UNIVERSITY OF CALIFORNIA
RIVERSIDE

Data-Driven Modeling and Control via Koopman Operator Theory in Robotic Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Lu Shi

June 2023

Dissertation Committee:

Dr. Konstantinos Karydis, Chairperson
Dr. Amit K. Roy-Chowdhury
Dr. Jorge Cortes

The Dissertation of Lu Shi is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

This dissertation is the result of years of effort and the basis of my future work. Along the way, there were a number of individuals and organizations I would like to acknowledge for their invaluable support and guidance.

First and foremost, I am immensely grateful to my supervisor, Dr. Konstantinos Karydis, not only for his expert instructions on my research but also for his encouragement, and patience that instilled in me a sense of confidence in myself and my work. He never hesitates to share his knowledge, experience, and suggestions that could help me in my career, which plays a pivotal role in shaping my research journey. His belief in my abilities has inspired me to delve deeper into the subject matter and find the direction I want to dedicate my career to. I am truly grateful for the guidance and continuous encouragement.

I am grateful to the faculty members who have provided invaluable help, guidance, and support throughout my studies. I would like to extend my sincere thanks to Dr. Amit K. Roy-Chowdhury and Dr. Jorge Cortes for serving on my committee. I am appreciative of all the professors whose courses have been a source of inspiration, expanding my knowledge and enhancing my skill sets. I am the kind of person who likes to raise questions during the lecture and I would like to thank all the kind explanations and answers from professors.

I consider myself extremely fortunate to have had the opportunity to work alongside an exceptional team of lab mates: Xinyue Kan, Zhouyu Lu, Zhichao Liu, Hanzhe Teng, Keran Ye, Cody Simons, Amel Dechemi, Mohamed Karim Bouafouram, Caio Mucchiani, Dimitris Chatziparaschis, Georgia Kouvoutsakis, Pamodya Peiris, Yipeng Wang, Tomas Olvera Hale, Xiao'ao Song and Mehrnoosh Ayazi. They have consistently provided me with

invaluable technical support whenever I needed it. Additionally, during times when I faced confusion and challenges, they have been a great source of mental support, offering guidance and encouragement. I am truly grateful for their contributions to my journey. Besides, I would like to express my gratitude to my dear family, and my friend Chengyun Shi, Jiayue Tian and Mina Rashetnia who support me when I feel stressed about my research.

Lastly, I would like to express my sincere gratitude to the National Science Foundation (NSF) and the Office of Naval Research (ONR) for their generous financial support. Their funding has played a vital role in enabling me to pursue my research endeavors. Additionally, I would like to extend my thanks to the Department of Electrical Engineering at UCR for providing me with funding support. These supports have allowed me to focus wholeheartedly on my research without concerns about financial matters.

I feel incredibly fortunate to have had such remarkable support and assistance throughout my 5-year Ph.D. journey. Thanks to the unwavering support and help from my colleagues, mentors, and friends, this period of my life has become one of the most fulfilling and enjoyable experiences. I genuinely cherish each day and appreciate the opportunity to engage in meaningful research and personal growth.

感谢我的父母石岩生先生及柴雅莲女士对我一如既往的的爱，鼓励与支持。

ABSTRACT OF THE DISSERTATION

Data-Driven Modeling and Control via
Koopman Operator Theory in Robotic Systems

by

Lu Shi

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2023
Dr. Konstantinos Karydis, Chairperson

With the development of more sophisticated robots that are increasingly aimed to venture outside of the lab, the higher dimensionality, complexity and nonlinearities of the underlying robotic systems as well as environmental uncertainty pose challenges to reliance on nominal models of robots obtained from first principles. Koopman operator theory, a tool first introduced for data-driven model extraction and global linearization, and its various extensions have been widely implemented in robot modeling and control. This dissertation contributes fundamental theory and practical algorithms for the implementation of the Koopman operator theory across distinctive types of robots.

Specific contributions of this dissertation span over four key sub-phases. These include data collection, model extraction, controller design, and physical implementation. First, the practical scenario of working with noisy data for modeling is considered. Prediction errors because of noisy measurements when estimating the model via the data-driven Koopman operator-based approaches for control are derived. Then, the explicit quantification of the error is embedded into an existing Koopman-based data-driven robot modeling

and control architecture to enhance its robustness without making significant changes to current parts of the underlying structure. Second, considering the importance of the space-lifting process in the Koopman theory, we propose a general and analytical algorithm to formalize the construction of the lifting functions based on characteristic properties of robots - namely their configuration space or, in the case of soft robots, their workspace. The resulting design of the lifting functions is proven to be complete and leads to an approximated Koopman operator with provable guarantees of convergence to the true one. Finally, we develop and present Koopman-based controllers and implement them to drive and/or improve the performance of robots. The first design is an online modeling and control approach via the Koopman operator theory for grasping tasks using soft multi-fingered robotic grippers. Then, a hierarchical structure that refines the reference signal sent to an existing high-rate, pre-tuned low-level controller to deal with uncertainty is presented, which aims to decrease the effect of environmental disturbance in real time.

The advancements in data analysis, model extraction, controller design, and their applications to robots, as presented in this dissertation, lay the groundwork for the practical implementation of the Koopman operator theory in physical robotic platforms. The theoretical and experimental explorations, along with the enhancements made, open up new possibilities for modeling and controlling nonlinear robots operating in uncertain environments. These advancements enable the achievement of more intricate objectives and can contribute to the overall progress in the field of robotic control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Cognitive models, that try to represent the robot, its environment, and how the two interact, build the basis of the control theory of various robots. Models constructed directly from first principles or physical derivations are always used as the dynamics constraints in model-based controllers. However, in practical implementation, such nominal models can be inaccurate and possibly invalid. There are two key reasons associated with this observation. First, there exist many instances in which robots interact (physically) with their uncertain environment. Examples include robots operating in partially-known dynamic environments [1, 2, 3]; legged robots traversing non-smooth terrains [4]; quadrotors flying under the influence of uncertain aerodynamic effects [5, 6]; underwater robots affected by uncertain ocean currents [7], etc. Thus, employing pre-selected models may restrict the capability to predict actual robot behaviors when operating under uncertainty. On the other hand, the models can be sophisticated and computationally difficult to be

obtained directly, e.g., the dynamics of soft robots [8] which has infinite degrees of freedom, and the models of complicated legged robots [9] that present high nonlinearity, etc. The two aforementioned reasons introduce the need for new methodologies to obtain models of the robots and their environments as well as possible interactions between the two.

With the development of new tools to obtain, analyze and store data, data-driven approaches become the new trend to get descriptions of robotic systems. Machine learning methods, such as Gaussian Process [10, 11], or Deep Neural Networks [12], can be used to either model the system [13, 14] or determine control inputs [15] following an input-output training procedure. However, as state dimensionality and system structure complexity increase (e.g., by using 'deeper' neural networks), the methods may be challenged when it comes to being implemented in real-time for robotics research (e.g., [16, 17, 18]). Despite their high expressiveness, neural network-based systems continue to require large training data sets and might lead to instability and unpredictable outputs because of overfitting.

Besides (deep) neural networks, dimensionality reduction and spectral approaches play an important role in data-driven algorithms [19]. Methods like the Proper Orthogonal Decomposition [20, 21], Dynamic Mode Decomposition (DMD) [22], Extended DMD (EDMD) [23], and their various extensions [24, 25, 26, 27] have been successfully applied across areas. Most of the techniques turn out to be strongly related to the Koopman operator theory [28]. Koopman spectral theory [29] has emerged as a promising data-driven methodology over the past decade, in which nonlinear dynamics are represented in terms of an infinite-dimensional linear operator acting on the space of all possible observable functions of the system (Fig. 1.1). Advances in data-driven estimation approaches of the

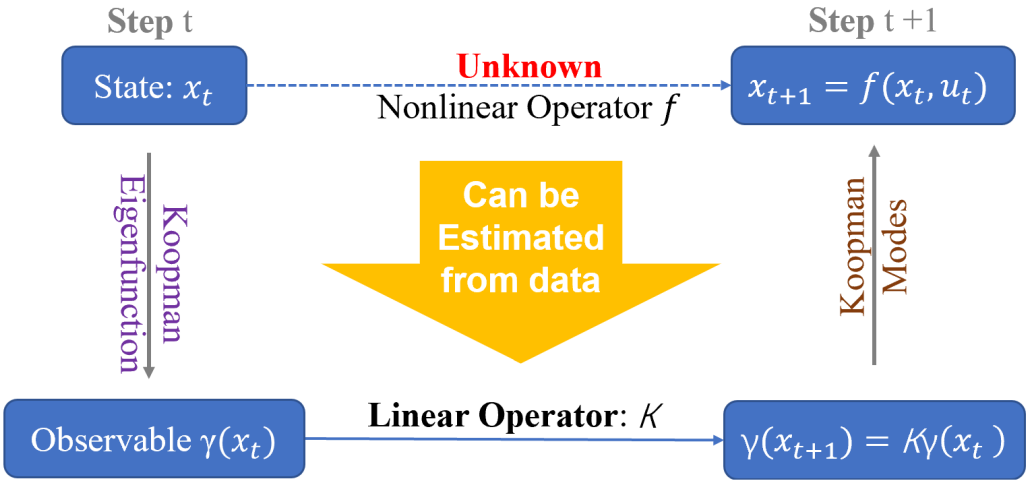

Figure 1.1: Overview of the Koopman operator theory, where the nonlinear (unknown) dynamic system is lifted to a higher dimensional linear space. The linear space is propagated by a linear operator: the Koopman operator, which can be approximated from data.

Koopman operator enable embedding the Koopman operator theory in robotic systems [30]. Instead of locally linearizing a system, the Koopman operator is able to evolve a nonlinear system with full fidelity throughout the state space (i.e. *global linearization*). This makes Koopman operators an attractive approach for obtaining linear representations of complex and/or nonlinear systems. By expressing a system's dynamics as a linear model, various well-designed reliable controllers can be easily obtained in the lifted space, and the stability properties can be more readily analyzed with existing well-designed theories. Owing to mapping the original unknown system to the Koopman space and approximating the operator from measurements, the Koopman operator theory can also be used as a *data-driven model extraction* method. Both of the two aforementioned properties make the Koopman operator well-suited for implementation in robotics to address the complexity of modeling as well as control and nonlinearities.

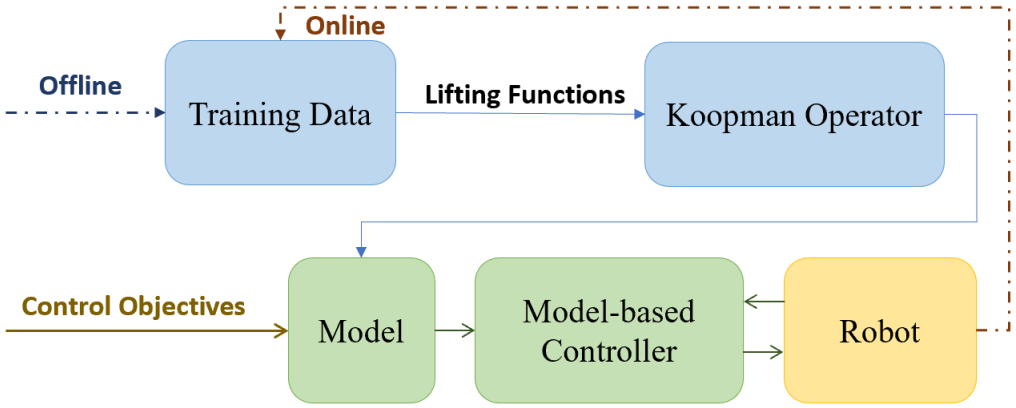

Figure 1.2: Overall procedure of modeling and control of robotic systems with the Koopman operator theory.

In this dissertation, we will introduce and present theoretical and algorithmic improvements when utilizing the Koopman operator theory in robotic platforms (Fig. 1.2 shows the overall procedure followed herein), including the analysis of data efficiency, enhancement on the accuracy of model extraction, control structure design, and practical applications in robots. Each part will be introduced and discussed in the following sections.

1.2 State of the Art

1.2.1 Data Efficiency

The quality of measurements used in the data-driven architecture is important as it will not only affect the feedback signal used for control but also the accuracy of the estimated model. As in other data-driven modeling approaches, methods employing the Koopman operator require analysis of safety and accuracy guarantees. Existing works include investigation of convergence of estimation [31, 32, 33] and global error bounds for the operator [34]. Another design [35] to minimize the effect of modeling uncertainties and

external disturbance is to include the usage of the Kalman filter. First, an augmented model is derived to include the disturbance model in the ordinary Koopman structure, and then a Kalman filter is adapted to estimate the observables and disturbances simultaneously. Other than the disturbed measurements, the cases for stochastic models are considered. In [36], a deep stochastic Koopman operator is introduced to guarantee the stability of nonlinear stochastic systems. A probabilistic neural network is designed to estimate the distribution of the observables and the distribution is propagated by the Koopman matrices. The parameters of the probabilistic neural network as well as the Koopman operator are optimized during the training process. Besides, [37] computes the nearest stable matrix solution to a least-squares reconstruction error to satisfy the safety guarantee.

Instead of dealing with the uncertainty and safety issues in the process of modeling directly, researchers also investigate the approaches to consider stability and model mismatch in controller design. For example, the effect of modeling error can be handled by using a constraint tightening approach in the proposed tracking Model Predictive Controller (MPC) strategy [38], which can lead to a recursively feasible and input-to-state stable system in the presence of bounded modeling errors. Another way to address this problem is by constructing a conformant model that combines trace conformance and reach-set conformance [39], which can then be used to construct controllers that are safe despite process noise and measurement errors acting on the real system. Additionally, to construct a robust MPC structure, it is possible to approximate a Lipschitz constant during training and utilize it to bound the prediction error along the planning horizon via a robust model-based controller online [40].

1.2.2 Model Extraction

A critical challenge inherent to all methods employing Koopman operator theory is the choice of a proper set of lifting functions. The lifting functions are crucial because they serve as the basis to construct an infinite-dimensional linear approximation of a (nonlinear) system's state evolution. Poor choice of the lifting function can significantly impact the estimation accuracy of the Koopman operator and the lifted linearized dynamics.

Existing works regarding the design of the lifting functions fall under several main directions. The first one is empirical selections [23]. For example, Legendre polynomials can make the observation matrix block diagonal, Hermite polynomials are best suited to problems where data are normally distributed, and radial basis functions are effective for systems with complex geometry. However, empirical approaches can be time- and effort-demanding and cannot guarantee generalization to and efficiency in new scenarios. Another direction is to rely on machine learning tools to derive the dictionary [41, 42, 43]. While such methods have stronger generalization capacity, they require significant tuning (e.g., in the case of neural networks the number of layers, number of units per layer, etc.), and large amounts of training data. The latter can in practice pose a significant challenge in robotics applications where data are in principle small. A third direction is when the original model has a special structure or some underlying fundamental model is known (or assumed). For example, elementary functions can be used to map a system to an equivalent polynomial form via Lie derivatives [44]. However, algorithms of that nature require the original system to be a linear combination of elementary functions. Another instance is to analyze the geometric relation between a system model and its Koopman operator to

obtain a dictionary [45]. While this algorithm can be useful in global linearization, controller design, and dimension reduction, it cannot offer model identification. Another direction is to compare and optimize over multiple sets of lifting functions to find a proper basis set, but still the question of how to select the sets to begin with in an efficient manner remains.

Each of the aforementioned methods has its own benefits and disadvantages. The selection should be determined based on the system and measurements. If the dataset is diverse and large enough, in other words, the offline training data cover as various cases as possible, the Machine-Learning-based dictionary could achieve high accuracy with sufficient time. Otherwise, the lifting functions can be designed based on experience or selected from multiple candidate dictionaries. However, a general algorithm to design the dictionary of lifting functions for robotic platforms is expected, which is also one of the main contributions of this dissertation which we are going to present in the following contents.

1.2.3 Controller Design

A system model enables the design of model-based controllers that leverage model predictions to choose suitable control inputs for a given task. We can obtain either a linear or a nonlinear model from measurements via the Koopman operator theory, which can be easily embedded into the design of model-based controllers. The key component of, and the main item that affects the performance of those controllers is the "model." The Koopman operator, which takes into account new measurements and/or offline measurements, can serve as the model constraint in the control structure formulation. The Koopman in model-based control is first introduced in [46]. Koopman MPC [32] and Koopman NMPC [47] are then proposed. The Koopman MPC illustrates how to formulate a Model Predictive Control

(MPC) structure with the Koopman operator theory, which not only provides the data-driven description of the underlying unknown system but also decreases the computation complexity. By estimating and/or updating models with online measurements or via offline learning through data collected from physical environments, the Koopman-model-based controller can achieve control objectives of unknown or partially known robots [48]. Designs are also embedded into the optimal control to work as the model description, e.g., Linear Quadratic Regulator (LQR) [49, 50]. Besides, [51, 52] provide a systematic approach for the design of stabilizing feedback controllers for nonlinear control systems using the Koopman operator framework. A control Lyapunov function (CLF)-based approach for the design of stabilizing feedback controllers for this Koopman-based bilinear system is then proposed and solved as a convex optimization problem. Another example [53] is to include an active learning strategy for robotic systems that take into account task information. It increases the rate of information about the Koopman operator and enables fast learning, which allows control to be readily synthesized by taking advantage of the Koopman operator representation.

1.2.4 Applications to Robotics

The properties of the Koopman-based methods, explainable learning, data-driven modeling, and intelligent control are important technical directions in the field of robotics. With the aforementioned research basis of dynamics extraction and controller design, the Koopman operator theory has gained more and more research attention being implemented among varying robotic systems. The characteristics of each robotic platform are emphasized accordingly.

The nonlinearity of the aerial robots makes them hard to be accurately controlled, especially when they are landing or flying at a low altitude, the surface effects (ground effects) can corrupt the nominal model and lead to unexpected performance. The data-driven Koopman-based approaches address the problem and decrease the influence of the ground effects by online extracting the models and linearly controlling the robot. The idea can be achieved via different types of design and implementations. For example, the authors use episodic learning to estimate the Koopman eigenfunction pairs and obtain the resulting control inputs in real-time to handle the ground effect when a multirotor is landing [54]. A sequence of Koopman eigenfunctions nonlinear control signals are iteratively learned from data and improved from the nominal control law. Another implementation of the Koopman operator theory in the aerial robots is to jointly learn a function dictionary and the lifted Koopman bilinear model to achieve quadrotor trajectory tracking at a low altitude [55]. A neural network is combined with the Koopman operator theory to update both the lifted states and inputs of the robot with online measurements.

Another one of the most popular research platforms is ground robots, which operate in contact with different types of terrain. The wheeled (differential-drive) mobile robot is a typical ground robot that operates based on a nonlinear non-holonomic model. In many applications, we expect wheeled robots to track and navigate smoothly and efficiently over varying types, and even rugged, deformable terrains. For tracking purposes, we usually formulate it as an optimization problem over a horizon of trajectory, where a model of the underlying system in the changing environment is required. The Koopman operator theory, which is purely data-driven, can serve as the model extraction method

without requirements on a priori knowledge. Then, a model-based controller, e.g., Model Predictive Controller (MPC), can be designed based on this data-driven model to achieve the tracking objective [38]. Other considerations including rotational motions of the mobile wheeled robots are emphasized via coordinate transformation by introducing the virtual control input [56]. Legged robots constitute another type of ground robots that move with articulated limbs. They can traverse different and more complicated terrains than wheeled robots. However, at the same time, the legged robots usually have highly nonlinear and sophisticated dynamics. An initial attempt at modeling of legged robots with the Koopman operator is done with the quadruped leg dynamics on deformable terrains [57]. An experimental framework is proposed to obtain a data-driven Koopman model of the quadruped's leg dynamics as a switched dynamical system. Experimental results show that the learned switched system model can be used to predict gait trajectories on unknown terrain. Though current developments in this area are sparse, more explorations in this area are expected for further developments like the controllers' design with this Koopman-based model, which might become a potentially powerful tool in the investigation of legged robot locomotion.

Other than rigid robots, soft robots are composed of compliant materials, instead of rigid links and joints. The compliance provides soft robots' flexibility, softness as well as safety when working in close contact with environments and humans. But these properties also make it hard to obtain model representations for analysis and control. One way to solve the problem is to embed the Koopman operator theory into model extraction. A primary benefit of the Koopman-based techniques is that a description of an input-output relationship can be obtained from data without explicitly defining a system state. This

is especially useful for obtaining reduced-order models of soft robots that have essentially infinite-dimensional kinematics. The Koopman operator-based modeling approaches have emerged as a dominant perspective for soft dynamics extraction over the past decade [58]. The Koopman-based MPC is widely used in soft platforms for either linear MPC [59, 60, 35, 61] or nonlinear MPC [59]. In [62], a soft inverted pendulum is controlled to be suspended stably in the inverted position. The Koopman-based MPC is also implemented in the soft continuum robots, the soft gripper to track desired trajectories [36, 61, 63, 35]. In these cases, the cost functions are always defined to minimize the difference between the predicted states obtained using the Koopman operator and desired trajectories or a target position. Besides the MPC, in [64], a reduced-order linear model of a helically actuated, inertial soft arm is obtained by the Koopman operator and serves as the model constraints to solve the discrete-time algebraic Riccati equation. The resulting control signal drives the soft arm to achieve two desired states.

1.3 Contributions

This dissertation contributes new theory and algorithms aimed at improving the deployment of Koopman operator theory in robotic systems. Overall, the contributions of the work cover the four key sub-phases of **data collection**, **model extraction**, **controller design**, and **physical implementation** during the process of utilizing the Koopman operator in the modeling and control of robotic platforms as shown in Fig. 1.2.

In terms of the analysis of the data efficiency, the first contribution is the robustness enhancement when the measurements for modeling are noisy [65]. We explicitly derived the

prediction error because of the noisy data when estimating the model via the data-driven Koopman operator-based approach for control. The explicit quantification of the error is then inserted into the existing Koopman-based data-driven robot control structure that endows robustness without making significant changes to the underlying architecture. The algorithm is validated via a parametric simulation study using a Van der Pol oscillator and experimentally tested in Gazebo with a non-holonomic wheeled robot (ROSbot2.0) under distinct noise levels.

The second contribution is the design of the lifting functions [66] for model extraction. We propose a general and analytical methodology to formalize the construction of the lifting functions based on system's characteristic properties of robots. The fundamental topological spaces and Cartesian products are analyzed and mapped to a set of lifting functions for a robotic system. The resulting dictionary of the lifting functions is proven to be complete and leads to an approximated Koopman operator with provable guarantees of convergence to the true one. We evaluate the efficacy of the proposed analytical approach in both simulated and hardware experiments, including a differential drive robot in both simulation and physical experimentation, a rigid robotic arm with two revolute joints in simulation, and a soft robotic leg. Comparison with other related algorithms are also presented.

The third contribution is the controllers' design and their implementations in different types of robots. First, we propose an online modeling and control approach via the Koopman operator theory and implement it for the task of grasping using a soft multi-fingered robotic grippers [67]. Second, a hierarchical structure is designed to deal with

uncertainty [68] that refines the reference signal sent to the existing low-level controller. A model of the reference and the actual output of the disturbed robot is learned via the Koopman operator theory and is utilized in the outer controller to decrease the effect of environmental disturbance in real time. The architecture is tested and compared with other related methods in both simulation and physical aerial robots under the ground effect.

The remainder of the dissertation is organized as follows. Background and preliminary knowledge of the Koopman operator theory and related data-driven estimation approaches are introduced in Chapter 2. Then, the contents are illustrated following the overall implementation procedure depicted in Fig. 1.2 and in ordered presented in the contributions. First, the improvement in the **Data** efficiency is discussed in Chapter 3. Then, a general algorithm for constructing the lifting functions used in **Model** extraction is presented in Chapter 4. Designs of **Controllers** and **Applications** to robotic platforms are given in Chapter 5. Finally, we summarize the conclusions and discuss future directions in Chapter 6. For convenience and clarification, key notations used in this document are included in Tab. 1.1.

Table 1.1: **Key Notations used in the Thesis.**

Notation	Description
x	State
u	Input
N_x	Dimension of state x
N_u	Dimension of input u
t	Index for online system propagation
γ	Observable function
$\mathcal{K}$	Koopman operator
K	Approximated Koopman operator via EDMD
K_A	Approximated Koopman operator via DMD
v_q	The q-th Koopman mode
λ_q	The q-th Koopman eigenvalue
φ_q	The q-th Koopman eigenfunction
Q	Dimension of observables' dictionary; Size of estimated Koopman operator
ξ_q	The q-th right eigenvector of Koopman operator
w_q	The q-th left eigenvector of Koopman operator
ψ_q	The q-th lifting function
$\mathbf{\Psi}$	Vector-valued observation

Chapter 2

Background

The Koopman operator is an infinite-dimensional linear operator that governs the evolution of observables $\gamma(x_t, u_t)$ of the original states. The evolving operator f of the original system can be represented by Koopman modes, eigenvalues, and eigenfunctions. In this section, we give an overview of key relevant results on Koopman operator theory and related data-driven approximation approaches.

2.1 Koopman operator

Consider a dynamical system with state vector $x \in \mathbb{R}^{N_x}$,

$$x_{t+1} = f(x_t) \text{ or } \frac{d}{dt}x = f(x) \ , \tag{2.1}$$

for the discrete-time and continuous-time settings, respectively. Define a set of observables $\gamma \in \mathcal{F}$ to be the *mapped* or *lifted* states. The evolution of observables γ using the infinite-

dimensional Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ can be written as

$$\mathcal{K}\gamma(x_t) = \gamma(f(x_t)) \text{ or } \frac{d}{dt}\gamma(x) = \mathcal{K}\gamma(x) \ .$$

If we further define the lifted states $\gamma(x)$ as *new* states $z = \gamma(x)$ in this linear Koopman space, the original nonlinear problem is transferred into a linear problem in this new space and various traditional linear controllers can be designed and implemented.

The Koopman operator can also be utilized to identify the dynamical system of the nonlinear form. Under the full-observability assumption that there exists one of the observables $\gamma(x) = x$, q Koopman-based items including Koopman modes v_q, eigenvalues λ_q and eigenfunctions φ_q can be obtained by decomposing the Koopman operator [23]. Finally, the unknown nonlinear operator f in (2.1) can be approximated as the combination of the Koopman-based items as

$$x_{t+1} = \gamma(f(x_t)) = \mathcal{K}\gamma(x_t) \to x_{t+1} = \sum_{q=1}^{Q} v_q \lambda_q \varphi_q(x_t) \ . \tag{2.2}$$

In its original formulation, Koopman operator theory applies to unforced systems. There are ways to generalize to forced systems. One is to lift states and inputs into two spaces separately, and then design controllers for the lifted system [69]. The other way is to extend the Koopman operator theory with control for systems with nonlinear input-output characteristics [70]. In the rest of this dissertation, we adopt the most general formulation in which inputs are generated from an exogenous forcing term. For details on other formulations, the reader is referred to [70, Section 3.1].

2.2 Dynamic Mode Decomposition (DMD)

In practice, we typically have access to the discretely sampled measurement data of the system, which are used to obtain a finite-dimensional approximated matrix of the infinite-dimensional Koopman operator. Although approximating the Koopman operator induces errors in the system propagation, it has been shown that the linear model is able to evolve the original system with acceptable accuracy [71, 31].

Similar to the Koopman operator, the DMD is introduced first in the fluids community. The DMD modes, which describe the spatial structure of the system and the DMD eigenvalues, which define the growth/decay rates and oscillation frequencies of each mode [23], are strongly related to the Koopman modes and eigenvalues. DMD analysis can be considered to be a numerical approximation to the Koopman spectral analysis. Of the most relevance and simplicity, DMD can be considered as getting a linear approximation of the original system and can be easily extended for inputs: DMDc [72]. Numerically, the goal of DMDc is to analyze the relationship between a future system measurement X_{t+1}, a current measurement x_t, and the current input u_t. For each triplet of measurement data (x_{t+1}, x_t, u_t), a pair of linear operators $K_A \in \mathbb{R}^{N_x \times N_x}$, and $B \in \mathbb{R}^{N_x \times N_u}$ is determined to approximate

$$x_{t+1} \approx K_A x_t + B u_t \ . \tag{2.3}$$

Operators K_A and B are selected as the best-fit solution for all triplets of available data. Given observations and inputs up to time instant $M+1$ arranged in vectors $X = [x_1, x_2 \ldots, x_M]$, $X_P = [x_2, x_3 \ldots, x_{M+1}]$, and $U = [u_1, u_2 \ldots, u_M]$, approximation (2.3) can

be rewritten compactly as

$$\begin{bmatrix} K_A & B \end{bmatrix} \begin{bmatrix} X \\ U \end{bmatrix} = A_P \Omega^T \approx X_P \ . \tag{2.4}$$

Then we could seek the best-fit solution as:

$$A_P = X_P (\Omega^T)^\dagger \ ,$$

where † denotes the pseudo-inverse. The problem can be solved immediately by Singular Value Decomposition (SVD), or QR decomposition, among others.

2.3 Extended Dynamic Mode Decomposition (EDMD)

Of all the approximation approaches, one of the most popular algorithms is the Extended Dynamic Mode Decomposition (EDMD) approach [23], which is the 'general' or 'extended' version of DMD. It describes how to learn those Koopman-based items in (2.2) from data. Given state history $X = [x_1, x_2, \ldots, x_M, x_{M+1}]$ and input history $U = [u_1, u_2, \ldots, u_M, u_{M+1}]$ (commonly referred to as "snapshots"), $\mathcal{K}$ can be expressed as a finite-dimensional approximation $K : \mathcal{F}_Q \to \mathcal{F}_Q$ of the Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ via EDMD. To do so, we need a dictionary of functions that lift state variables to the higher-dimensional space where the observable dynamic is approximately linear. If the lifting dictionary is chosen as $\mathbf{\Psi}(x_m) = [e_1^T x, \ldots, e_{N_x}^T X_m]$, where e_i is the $i-th$ unit vector in $\mathbb{R}^{N_x}$, the estimated Koopman operator generated by the following procedure is same as that obtained by DMD. Otherwise in general, define the dictionary of observables of size Q, $\mathcal{D} = \{\psi_1, \psi_2, \ldots, \psi_Q\}$, where each dictionary element ψ_q, $q = 1, \ldots, Q$ is a differentiable function containing x_m and u_m terms, set the vector-valued dictionary as $\mathbf{\Psi} = [\psi_1, \ldots, \psi_Q]$,

the Koopman operator can be approximated by minimizing the total residual between snapshots,

$$J = \frac{1}{2} \sum_{m=1}^{M} \left(\boldsymbol{\Psi}_{m+1} - \boldsymbol{\Psi}_m K \right)^2 \quad . \tag{2.5}$$

This can be solved by truncated singular value decomposition, yielding

$$K \triangleq \boldsymbol{G}^{\dagger} \boldsymbol{A}, \text{ where } \begin{cases} \boldsymbol{G} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{\Psi}_m^* \boldsymbol{\Psi}_m \quad , \\ \boldsymbol{A} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{\Psi}_m^* \boldsymbol{\Psi}_{m+1} \quad , \end{cases} \tag{2.6}$$

with †, T and * denoting pseudoinverse, transpose and conjugate transpose, respectively. Obtained K via (2.6), we get

$$\begin{cases} v_q = (w_q^* B)^T \quad , \\ \lambda_q \xi_q = K \xi_q \quad , \\ \varphi_q = \boldsymbol{\Psi}_t \xi_q \quad , \end{cases} \tag{2.7}$$

where ξ_q is the q-th eigenvector, w_q is the q-th left eigenvector of K scaled so $w_q^T \xi_q = 1$, and B is a weight matrix such that $x = (\boldsymbol{\Psi} B)^T$ [23]. The evolution of the original nonlinear system using the estimated Koopman operator is described by replacing expressions (2.7) into (2.2) [70].

Chapter 3

Data: Enhance Robustness

For all methods that extract models from data, the quality of measurements should be addressed. Indeed, when physical implementations with robotic platforms are considered, noise is always present in collected data. Thus, investigating the prediction error of, or providing robustness guarantees for, the perturbed systems' performance when the data used for modeling is noisy is of much significance. In this section, we first present the procedure for deriving loose and tight bounds of the prediction error using Dynamic Mode Decomposition (DMD). Then, we explore the general version of the data-driven approach, Extended DMD, and further obtain the explicit prediction error because of noisy measurements. Finally, we illustrate how the proposed derivation procedure and the resulting prediction error can be utilized in controller design. Evaluation of the final controller's efficacy and generalizability are tested with a non-holonomic wheeled robot.

3.1 Sensitivity Analysis of DMDc

Recall the introduction in Sec. 2.2, DMDc is to determine the matrices $[K_A, B] = A_P$ by calculating the best-fit solution of $A_P = X_P(\Omega^T)^\dagger$ for all available observations. X_P and Ω are collected measurements on states and inputs as used in (2.4). The noise in the measurement matrices X_P and Ω will influence the approximation of the system propagation operator A_P. Below we analyze the sensitivity to noisy data of DMDc and derive two estimation error bounds.

3.1.1 Sensitivity to noisy data

For convenience and understandability, we will first derive on a set of general equations and then apply the results directly to the proposed problem (2.4).

Lemma 2.1. Consider the equations

$$\begin{aligned} H\mathbf{z} &= \mathbf{s} \ , \qquad H \in \mathbb{R}^{p\times q},\ \mathbf{s} \in \mathbb{R}^{p\times t} \\ (H+\delta H)\hat{\mathbf{z}} &= \mathbf{s}+\delta\mathbf{s} \ , \quad \delta H \in \mathbb{R}^{p\times q},\ \delta\mathbf{s} \in \mathbb{R}^{p\times t} \end{aligned} \tag{3.1}$$

The sensitivity of system (3.1) to perturbations in $\mathbf{s}$ and H is

$$\frac{\|\hat{\mathbf{z}}-\mathbf{z}\|}{\|\mathbf{z}\|} \le \left(\kappa(H)^2\tan(\theta)+\kappa(H)\right)\frac{\|\delta H\|}{\|H\|}+\kappa(H)\sec(\theta)\frac{\|\delta\mathbf{s}\|}{\|\mathbf{s}\|} \ . \tag{3.2}$$

where $\kappa(H)$ is the condition number of linear operator H, and $\theta = \arccos\frac{\|H\mathbf{z}\|}{\|\mathbf{s}\|}$.

Proof of Lemma 2.1: Start with the normal equations

$$H^TH\mathbf{z} = H^T\mathbf{s} \ .$$

The first-order perturbation relation is

$$\delta H^TH\mathbf{z}+H^T\delta H\mathbf{z}+H^TH\delta\mathbf{z} = \delta H^T\mathbf{s}+H^T\delta\mathbf{s} \ ,$$

which we re-arrange to get

$$\delta\mathbf{z} = (H^TH)^{-1}\delta H^T(\mathbf{s} - H\mathbf{z}) + (H^TH)^{-1}H^T(\delta\mathbf{s} - \delta H\mathbf{z}).$$

Define $r = \mathbf{s} - H\mathbf{z}$, we can obtain

$$\|\delta\mathbf{z}\| \leq \|(H^TH)^{-1}\|\|\delta H\|\|r\| + \|(H^TH)^{-1}H^T\|\left(\|\delta\mathbf{s}\| + \|\delta H\|\|\mathbf{z}\|\right) \ .$$

Let $\sigma_1, \ldots, \sigma_p$ be singular values of H. Then $\|H\| = \sigma_1$, $\|(H^TH)^{-1}\| = 1/\sigma_q^2$, we can derive [73]

$$\|\delta\mathbf{z}\| \leq \frac{\|\delta H\|}{\sigma_q^2}\|r\| + \frac{1}{\sigma_q}(\|\delta\mathbf{s}\| + \|\delta H\|\|\mathbf{z}\|) \ .$$

Dividing both sides by $\|\mathbf{z}\|$ and after some algebraic manipulation so that $\|\delta H\|$ and $\|\delta\mathbf{s}\|$ only appear in ratios of $\frac{\|\delta H\|}{\|H\|}$ and $\frac{\|\delta\mathbf{s}\|}{\|\mathbf{s}\|}$, as $\kappa(H) = \sigma_1/\sigma_q$, we get

$$\frac{\|\delta\mathbf{z}\|}{\|\mathbf{z}\|} \leq \kappa(H)^2\frac{\|r\|}{\|H\|\|\mathbf{z}\|}\frac{\|\delta H\|}{\|H\|} + \kappa(H)\left(\frac{\|\mathbf{s}\|}{\|H\|\|\mathbf{z}\|}\frac{\|\delta\mathbf{s}\|}{\|\mathbf{s}\|} + \frac{\|\delta H\|}{\|H\|}\right) \ .$$

Then with

$$\frac{\|r\|}{\|H\|\|\mathbf{z}\|} \leq \frac{\|r\|}{\|H\mathbf{z}\|} = \tan(\theta) \ ,$$
$$\frac{\|\mathbf{s}\|}{\|H\|\|\mathbf{z}\|} \leq \frac{\|\mathbf{s}\|}{\|H\mathbf{z}\|} = \sec(\theta) \ ,$$

we arrive at (3.2). ■

Corollary 2.2. The estimation error of problem (2.4) is bounded by the measurements' noise intensity, i.e.

$$\frac{\|\delta A_P\|}{\|A_P\|} \leq \left(\kappa(\Omega)^2\tan(\theta) + \kappa(\Omega)\right)\frac{\|\delta\Omega\|}{\|\Omega\|} + \kappa(\Omega)\sec(\theta)\frac{\|\delta X_P\|}{\|X_P\|}, \tag{3.3}$$

where $\kappa(\Omega)$ is the condition number of observation matrix Ω, and $\theta = \arccos\frac{\|A_P\Omega\|}{\|X_P\|}$.

Proof of Corollary 2.2: The proof follows directly from Lemma 2.1 by setting $H = \Omega$, $\mathbf{z} = A_P^T$, and $\mathbf{s} = X_P^T$. ■

Lemma 2.3 [A tighter bound]. Let $\epsilon = \frac{\|\delta \mathbf{s}\|}{\|\mathbf{s}\|}$. In the case that $\|\delta \mathbf{s}\| = \|\delta H\|$, the sensitivity of (3.1) is reduced to

$$\frac{\|\hat{\mathbf{z}} - \mathbf{z}\|}{\|\mathbf{z}\|} \leq \epsilon \kappa(H)(1 + \|\hat{\mathbf{z}}\|) \ . \tag{3.4}$$

Proof of Lemma 2.3: From system (3.1), we have

$$H(\hat{\mathbf{z}} - \mathbf{z}) = \delta \mathbf{s} - \delta H \hat{\mathbf{z}} \ ,$$

and it follows that

$$\hat{\mathbf{z}} - \mathbf{z} = (H^T H)^{-1} H^T \delta \mathbf{s} - (H^T H)^{-1} H^T \delta H \hat{\mathbf{z}} \ .$$

Then,

$$\|\hat{\mathbf{z}} - \mathbf{z}\| = \|(H^T H)^{-1} H^T\|(\|\delta \mathbf{s}\| + \|\delta H\|\|\hat{\mathbf{z}}\|) \ .$$

Setting $\|\delta \mathbf{s}\| = \|\delta H\|$ yields

$$\begin{aligned} \|\hat{\mathbf{z}} - \mathbf{z}\| &= \|(H^T H)^{-1} H^T\|\|\delta \mathbf{s}\|(1 + \|\hat{\mathbf{z}}\|) \\ &= \epsilon \|(H^T H)^{-1} H^T\|\|\mathbf{s}\|(1 + \|\hat{\mathbf{z}}\|) \\ &\leq \epsilon \|(H^T H)^{-1} H^T\|\|H\|\|\mathbf{z}\|(1 + \|\hat{\mathbf{z}}\|) \\ &= \epsilon \kappa(H)\|\mathbf{z}\|(1 + \|\hat{\mathbf{z}}\|) \ , \end{aligned}$$

which concludes the proof. ■

Corollary 2.4. When the disturbance occurs in observed states, i.e. $\|\delta \Xi_P\| = \|\delta \Xi\| = \|\delta \Omega\|$ a tighter error bound is

$$\frac{\|\delta A_P\|}{\|A_P\|} \leq \kappa(\Omega) \frac{\|\delta \Xi_p\|}{\|\Xi_p\|}(1 + \|\hat{A}_p\|) \ . \tag{3.5}$$

Proof of Corollary 2.4: The proof follows directly from Lemma 2.3 by setting $H = \Omega$, $\mathbf{z} = A_P^T$, and $\mathbf{s} = X_P^T$. ■

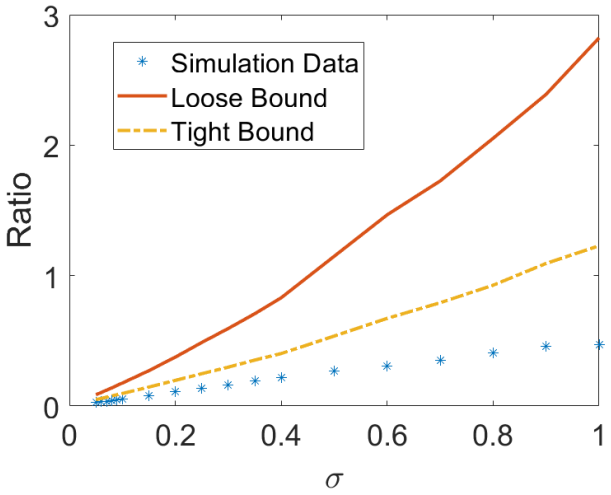

Figure 3.1: Estimation sensitivity to perturbation $\frac{\|\delta A_P\|}{\|A_P\|}$ and predicted bounds as noise magnitude increases.

3.1.2 Toy Example:

Consider the simple linear system

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} u_t \ . \tag{3.6}$$

We add zero-mean white noise to corrupt state observations, i.e. $N(0, \sigma^2)$, with $\sigma \in [0.05, 1]$ in 0.05 increments. The initial condition is set at $[0.01, 0.01]$. We perform a Monte Carlo simulation for $10,000$ trials to produce the average error. With reference to Fig. 3.1, the effect of the disturbance increases as variance increases (as expected). The general bound given in (3.3) expands at higher rates than the tighter one we obtain via (3.5) as the noise variance increases.

3.2 Sensitivity Analysis of EDMD

Extended Dynamic Mode Decomposition (EDMD) is the general nonlinear extension of DMD. It is also the most popular and main approach to estimating the Koopman operator. As illustrated in Sec. 2.3, when training measurements are noisy, the approximated operator, the resulting model, and the prediction will be inaccurate.

To explicitly demonstrate the effect of the noise in data, we derive the error in steps. We first investigate the sensitivity of the approximated Koopman operator K to noisy training measurements, followed by the sensitivity of the eigendecomposition to disturbed elements of K. Then, we combine them together to quantify the prediction error caused by noisy data when Koopman operator theory is employed.

3.2.1 Sensitivity Analysis of Koopman Operator

Perturbations on an element x_m^i (i.e. the i^{th} state of measurement at snapshot m with $i \in \{1, \ldots, N_x\}$), will cause an error ΔK^i. This can be obtained by chain rule on (2.6) as

$$\Delta K^i = [\Delta k_{ab}^i]_{Q\times Q} = \sum_{m=1}^{M+1} \{(\frac{\partial G^\dagger}{\partial x_m^i} A + G^\dagger \frac{\partial A}{\partial x_m^i})\Delta x_m^i\} \ , \tag{3.7}$$

where [1]

$$\frac{\partial G^\dagger}{\partial x_m^i} = -G^\dagger \frac{\partial G}{\partial x_m^i} G^\dagger + G^\dagger (G^\dagger)^* \left(\frac{\partial G^*}{\partial x_m^i}\right)\left(I - GG^\dagger\right) + \left(I - G^\dagger G\right)\left(\frac{\partial G^*}{\partial x_m^i}\right)(G^\dagger)^* G^\dagger$$

[1] For details on the proof of the following pseudo-inverse terms see [74].

$$\frac{\partial A}{\partial x_m^i} = \begin{cases} \frac{1}{M}\mathbf{\Psi}_{m+1}^* \frac{\partial \mathbf{\Psi}_m}{\partial x_m}\mathbf{e_i}, & \text{if } m = 1 \\ \frac{1}{M}\mathbf{\Psi}_{m-1}^* \frac{\partial \mathbf{\Psi}_m}{\partial x_m}\mathbf{e_i}, & \text{if } m = M+1 \\ \frac{1}{M}(\mathbf{\Psi}_{m+1}^* + \mathbf{\Psi}_{m-1}^*)\frac{\partial \mathbf{\Psi}_m}{\partial x_m}\mathbf{e_i}, & \text{else} \end{cases}$$

and

$$\frac{\partial G}{\partial x_m^i} = \begin{cases} 0, & \text{if } m = M+1 \\ \frac{2}{M}\mathbf{\Psi}_m^* \frac{\partial \mathbf{\Psi}_m}{\partial x_m}\mathbf{e_i}, & \text{else} \ . \end{cases}$$

Here $\mathbf{e_i}$ represents the i^{th} unit vector in $\mathbb{R}^{N_x}$.

3.2.2 Sensitivity Analysis of Eigendecomposition

As described in [75], if a generic element k_{ab} is perturbed, the eigenvalues and eigenvectors of the matrix $K = [k_{ab}] \in \mathbb{R}^{Q\times Q}$ are affected. The sensitivity of left eigenvectors w_q, right eigenvectors ξ_q, and eigenvalues λ_q can be written as

$$\begin{cases} c_{\lambda_q}^{ab} = \frac{\partial \lambda_q}{\partial k_{ab}} = w_q^a \xi_q^b, \ \ (a,b,q = 1,2,\ldots,Q) \ , \\ c_{\xi_q}^{ab} = \frac{\partial \xi_q}{\partial k_{ab}} = \sum_{\substack{j=1 \\ j\neq q}}^{Q} h_{jq}^{ab}\xi_j \ \ , (a,b,q = 1,2,\ldots,Q) \ , \\ c_{w_q}^{ab} = \frac{\partial w_q^*}{\partial k_{ab}} = -\sum_{\substack{j=1 \\ j\neq q}}^{Q} h_{qj}^{ab} w_j^* \ \ , (a,b,q = 1,2,\ldots,Q) \ , \end{cases} \tag{3.8}$$

where w_q^a denotes the a^{th} element of the q^{th} left-eigenvector w_q, ξ_q^b is the b^{th} element of the q^{th} right-eigenvector ξ_q, and h_{qj}^{ab} represents the a^{th} row and b^{th} column element of matrix H_{qj} (similar for h_{jq}^{ab}). The matrix $H_{qj} = \left[h_{qj}^{ab}\right]_{Q\times Q}$ is computed by $\frac{w_q \xi_j^*}{\lambda_j - \lambda_q}$. That is, to get the eigendecomposition sensitivity of each element k_{ab}, we first need to obtain $(Q\times Q - Q)H$ matrices, and then every entry of these matrices will be used as parameters in (3.8).

3.2.3 Sensitivity Analysis of Predicted Output

We are now ready to derive the perturbation in predicted outputs as a result of measurement noise. Setting $[I_{N_x\times N_x}, O_{N_x\times N_u}]$ as R, where $I_{N_x\times N_x}$ denotes the $N_x \times N_x$ identity matrix and $O_{N_x\times N_u}$ denotes a $N_x \times N_x$ zero matrix. For the fully-observable forced nonlinear discrete time system (2.2), we can describe its deviation as

$$\Delta x_{t+1} = R\sum_{m=1}^{M+1}\left(\sum_{q=1}^{Q}\sum_{i=1}^{N_x}\frac{\partial(v_q\lambda_q\varphi_q(x_t,u_t))}{\partial x_m^i}\Delta x_m^i\right) \tag{3.9}$$

$$= R\sum_{m=1}^{M+1}\left(\sum_{a=1}^{Q}\sum_{b=1}^{Q}\sum_{q=1}^{Q}\sum_{i=1}^{N_x}\frac{\partial(v_q\lambda_q\varphi_q(x_t,u_t))}{\partial k_{ab}^i}\frac{\partial k_{ab}^i}{\partial x_m^i}\Delta x_m^i\right)$$

$$= R\sum_{a}^{Q}\sum_{b}^{Q}\left((\sum_{q}^{Q}\sum_{i}^{N_x}\frac{\partial(v_q\lambda_q\varphi_q(x_t,u_t))}{\partial k_{ab}^i})(\sum_{m}^{M+1}\frac{\partial k_{ab}^i}{\partial x_m^i}\Delta x_m^i)\right).$$

Letting

$$\Delta k_{ab}^i = \sum_{m=1}^{M+1}\frac{\partial k_{ab}^i}{\partial x_m^i}\Delta x_m^i,$$

we deduce

$$\Delta x_{t+1} = R\sum_{a=1}^{Q}\sum_{b=1}^{Q}\left(\sum_{q=1}^{Q}\sum_{i=1}^{N_x}\frac{\partial(v_q\lambda_q\varphi_q(x_t,u_t))}{\partial k_{ab}}\Delta k_{ab}^i\right) \quad . \tag{3.10}$$

Note that as illustrated in (3.8), for a dynamical system evolving by K, $\frac{\partial\lambda_q}{\partial k_{ab}} = c_{\lambda_q}^{ab}$, $\frac{\partial\xi_q}{\partial k_{ab}} = c_{\xi_q}^{ab}$, and $\frac{\partial w_q^T}{\partial k_{ab}} = c_{w_q}^{ab}$ are finite (bounded) constants. Then, for every k_{ab} we have

$$\begin{aligned}\frac{\partial(\boldsymbol{v}_q\lambda_q\varphi_q(x_t,u_t))}{\partial k_{ab}} &= \frac{\partial(\boldsymbol{v}_q)\lambda_q\varphi_q}{\partial k_{ab}} + \frac{\boldsymbol{v}_q\partial(\lambda_q)\varphi_q}{\partial k_{ab}} + \frac{\boldsymbol{v}_q\lambda_q\partial(\varphi_q)}{\partial k_{ab}} \\ &= ((c_{w_q}^{ab})^*B)^T\lambda_j\boldsymbol{\Psi}_t\xi_q + (w_q^*B)^Tc_{\lambda_q}^{ab}\boldsymbol{\Psi}_t\xi_q + (w_q^*B)^T\lambda_q\boldsymbol{\Psi}_tc_{\xi_q}^{ab}.\end{aligned} \tag{3.11}$$

Using matrix format of (3.11) and $\Delta K^i = [\Delta k_{ab}^i]_{Q\times Q}$ from (3.7), we can first compute the dot product of these two matrices and then obtain the prediction error as the sum of each

element of the dot product, that is

$$\Delta x_{t+1} = Re^T\{\left[\sum_{q=1}^{Q}\frac{\partial(\boldsymbol{v}_q\lambda_q\varphi_q(x_t,u_t))}{\partial k_{ab}}\right]_{Q\times Q}\cdot(\sum_{i=1}^{N_x}\Delta K^i)\}e \ , \tag{3.12}$$

where $e=[1,1,\ldots,1]^T$.

3.2.4 Parametric Testing with a Van der Pol Oscillator

To evaluate the proposed procedure of approximating the prediction error, we test it using the Van del Pol oscillator [32],

$$\begin{cases} \dot{x}_1 = 2x_2 \\ \dot{x}_2 = -0.8x_1 + 2x_2 - 10x_1^2x_2 + u \ . \end{cases} \tag{3.13}$$

The system (3.13) is discretized with period $T_s = 0.01$ s. $M+1$ pairs of data generated from the system are used to estimate Koopman operator K, and are further implemented as dynamic constraints in the controller. A random input vector is applied to propagate the system.

The whole process is implemented in a parametric study as follows. We consider four distinct cases for measurement sets with $M=\{2*10^3, 1*10^4, 2*10^4, 2*10^5\}$. We also consider three distinct noise levels for the disturbance. The amplitudes of the disturbances are chosen randomly from uniform distributions over the intervals $c_1=[0,0.1x(0)]$, $c_2=[0,0.2x(0)]$ and $c_3=[0,0.4x(0)]$ for '**low** (10%),' '**middle** (20%)' and '**high** (40%)' noise levels, respectively. These lead to a total of 12 distinct case studies. For each case study, we train the system according to the selected measurements and noise settings, then give a random input vector to propagate the trained system for 50 time steps, collect the output, and finally measure the true and predicted errors. To avoid bias and randomness, we

evaluate the trained system under each case study in 5 repeated trials, and averaged results are reported. For illustration, Fig. 3.2 shows the example of the average predicted error and the average true error when $M = 2 * 10^4$ and under the c_2 noise case (middle).

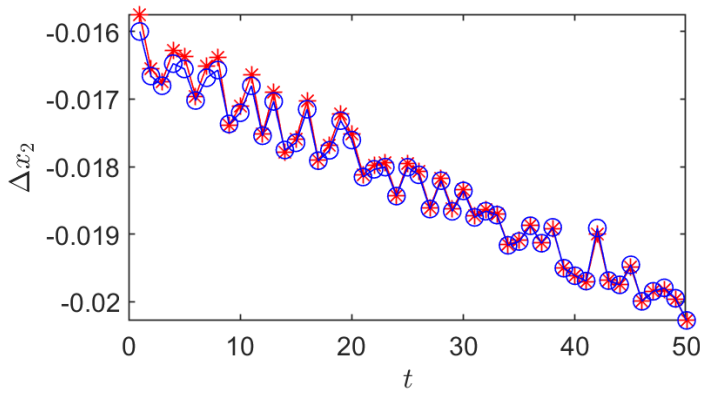

Figure 3.2: Prediction (blue circle) and true (red star) error under middle level noise (c_2) when $M = 2 * 10^4$.

To compare overall the cases, we define the Mean Squared Error (MSE) between the **estimated** and **true** prediction error as

$$MSE = \frac{1}{50}\sum_{t=1}^{50}(\Delta x_2^{\text{prediction}}(t) - (\Delta x_2^{\text{true}}(t))^2 \ . \tag{3.14}$$

Results (averaged MSE) are depicted and compared in Fig 3.3. We observe that 1) low density noise leads to smaller prediction error for most choices of M. 2) Our method performs better with a large enough size of training data set (e.g., greater than $2 * 10^4$ as shown in the simulation).[2] 3) If we keep increasing the size (e.g., $5 * 10^4$ in this one), its accuracy may not be improved further.

[2] In practice, good results may be achieved with fewer data, as we show next in simulation with a wheeled robot. Better understanding the interplay between training data set size and performance is part of ongoing work.

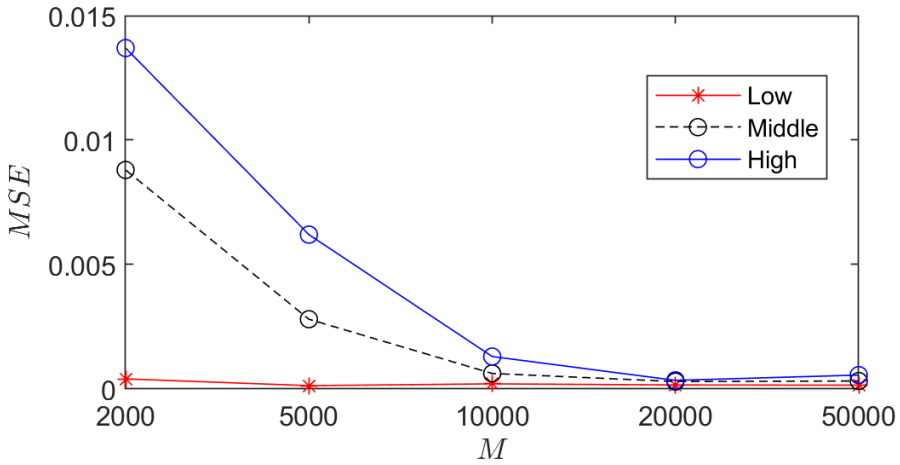

Figure 3.3: Mean Squared Error (MSE) of estimated and true prediction error under different noise levels.

3.3 Algorithmic Implementation

The proposed quantification of the prediction error approach describes analytically how to accommodate for measurement noise in data-driven control systems that are based on the Koopman operator. It is worth noting that in our proposed formulation in (3.12), ΔK^i is estimated using only measurements and information about the noise, so it can be obtained offline. Similarly, the eigendecomposition sensitivity analysis of K is predetermined and can yield the complex parameters $c^{ab}_{w_q}$, $c^{ab}_{\lambda_q}$, $c^{ab}_{\xi_q}$ ahead of time. These allow part of our approach to be executed during training and enable the computation of the prediction error to take place at run-time, to adjust to measurement errors and thus endow robustness to the overall data-driven robot control architecture (Fig. 3.4).

To make the approach amenable to robot control applications, where real-time implementation is required, we demonstrate in this section how the most computationally-intense aspects of the proposed approach can in fact be precomputed. Then, we present the algorithmic implementation of our approach in Algorithm 1.

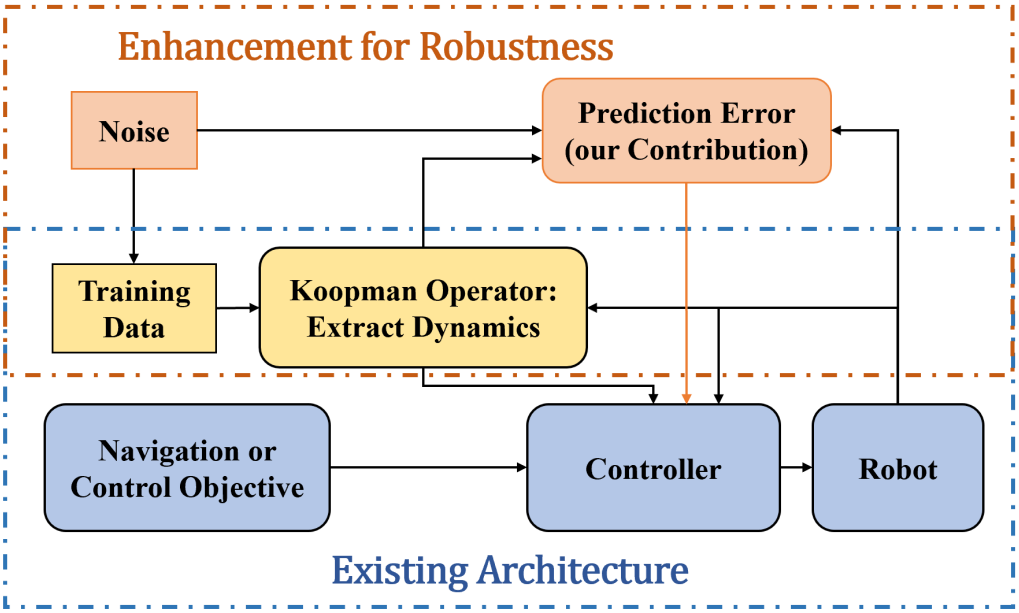

Figure 3.4: Overview of the enhanced robot control structure proposed in this work.

3.3.1 Enabling Real-time Operation

Reformulation of dynamics-extraction expression: Recall that only the observation at current time t, $\mathbf{\Psi}_t$, requires information gathered during run-time for making a prediction. We can thus rewrite (2.2) to extract the terms that can be precomputed.

Consider the dynamics $x_{t+1} = [I_{N_x \times N_x}, O_{N_x \times N_u}] \sum_{q=1}^{Q} v_q \lambda_q \varphi_q(x_t, u_t)$, combining (2.7) yields

$$\sum_{q=1}^{Q} v_q \lambda_q \varphi_q(x_t, u_t) = \sum_{q=1}^{P} (w_q^* B)^T \lambda_q \mathbf{\Psi}_t \xi_q = \left(\sum_{q=1}^{Q} (\mathbf{\Psi}_t \xi_q)^T \left((w_q^* B)^T \lambda_q \right)^T \right)^T \tag{3.15}$$

$$\rightarrow x_{t+1} = [I_{N_x \times N_x}, O_{N_x \times N_u}] (\mathbf{\Psi}_t F)^T \quad ,$$

where $F = \sum_{q=1}^{Q} \xi_q \lambda_q w_q^* B$. Thus, we have separated the Koopman prediction of the next state (2.2) into offline (F) and online ($\mathbf{\Psi}_t$) parts.

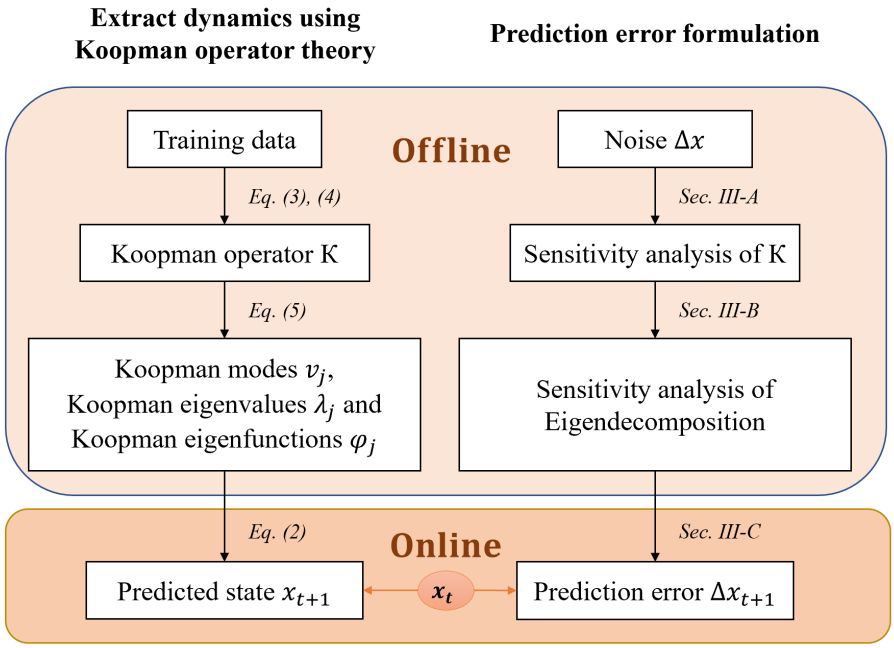

Figure 3.5: Illustration of the proposed approach's pipeline to adding robustness to data-driven robot control design (c.f. top system sub-block in Fig. 3.4). Our approach can be decomposed into an offline and an online component for real-time implementation, specific details of which are given in Algorithm 1.

Information of noise: In the original formulation (3.9), we require the noise in measurements Δx_m at every time instant. However, in practice, we only have access to (or make assumptions about) noise statistics, for example, mean $E(\Delta x)$ and standard deviation $\delta(\Delta x)$ in the case of Gaussian noise. We can thus replace each Δx_m by $E(\Delta x)$ directly. Also, we can randomly sample a training dataset $\Delta X \sim (E(\Delta x), \delta^2(\Delta x))$, applying it to (3.9), and repeat the process several times to get the average values for accuracy. (The case of Gaussian noise herein is used as an example. Our proposed approach does not depend on the type of noise employed. Rather, noise statistics is a hyper-parameter that can be manually selected by the user or approximated via any available training data.)

3.3.2 Algorithm

We summarize the whole procedure for (1) extracting dynamics using Koopman operator and (2) estimate the prediction error when we want to use the Koopman based data-driven methods for control. Each procedure is clarified in "offline" and "online" phases as shown in Fig. 3.5 and Algorithm 1.

Algorithm 1 Robust Koopman-based Control Algorithm

Initialization: Collect measurements of states X and inputs U that might be noisy.

Offline training:

Extracting dynamics:

- Utilize X, U to estimate the Koopman operator K as well as matrices G, A with (2.6).
- Compute parameter matrix F in (3.15) with estimated K.

Estimation of prediction error:

- Obtain ΔK^i with information of noise in measurements and the approximated K, G, A using (3.7).
- Get the parameters for eigendecomposition sensitivity $c_{\lambda_q}^{ab}$, $c_{\xi_q}^{ab}$ and $c_{w_q}^{ab}$ of the estimated Koopman operator K through (3.8).

Online propagation for t do

Robotic System:

- Use learned prediction error Δx_t at last timestep to complement model constraints used in controller design.
- Drive the robot with the signal calculated by the updated controller.

Extracting dynamics:

- Use F to estimate the system dynamics and do prediction as described in (3.15).

Estimate prediction error:

- Get eigen-sensitivity matrix (3.11) with pretrained $c^{ab}_{\lambda_q}$, $c^{ab}_{\xi_q}$, $c^{ab}_{w_q}$ and current observation $\mathbf{\Psi}_t$.
- Compute prediction error Δx_{t+1} with ΔK and the eigen-sensitivity matrix by (3.12).

$t \leftarrow t + 1$

3.3.3 Evaluation with a Wheeled Robot

Simulation experiments are conducted in the Gazebo environment using a ROSbot 2.0 robot, a differential-drive wheeled robot (Fig. 3.6). The ROSbot 2.0 is an autonomous, open-source robot platform and the embedded software allows us to send motion commands

by manipulating the x component of the linear speed vector as u_x and the z component of the angular speed vector as u_z. The state vector contains the geometric center position $\{p_x, p_y\}$ and orientation θ.

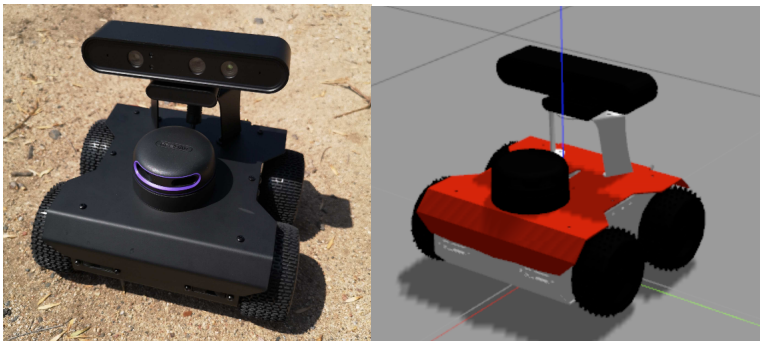

Figure 3.6: ROSbot 2.0 and its Gazebo model.

We test the efficiency of our proposed approach by building on top of a Koopman operator-based controller that we are going to explain more about in the following Chapter 5, which is the data-driven hierarchical structure that refines the nominal input u_o as $\hat{u} = u(K, u_o)$ to improve the performance of the controller under uncertainty. We highlight that the relation between inputs u and the states x is learned using the Koopman operator, and then the learned model is used to get a predicted (refined) input signal $\hat{u}$. Accurate and noisy data are used to generate the deterministic Koopman operator K and stochastic one K_n. Then, these two are worked as model constraints to design the '**Nominal**' and '**Noisy**' controllers separately. To reduce the effect of noise in the learned model K_n, which is the contribution of our paper, we try to learn the prediction error generated by K_n using Algorithm 2. Then, this prediction error is used to complement the 'Noisy' controller to yield the '**Proposed**' control signal.

We compute the approximated Koopman operator using data captured when the robot is operating in an open loop based on random chattering linear velocity and angular velocity input signals. We perform 10 such open-loop trials. The noisy training data set is created by adding disturbance in the position states p_x and p_y. We sample the magnitude of noise for each state from the uniform distribution over the interval $c_1 = [0, 0.25\max(x^r)]$, $c_2 = [0, 0.50\max(x_r)]$, $c_3 = [0, 0.75\max(x^r)]$, $c_4 = [0, 1.00\max(x^r)]$.

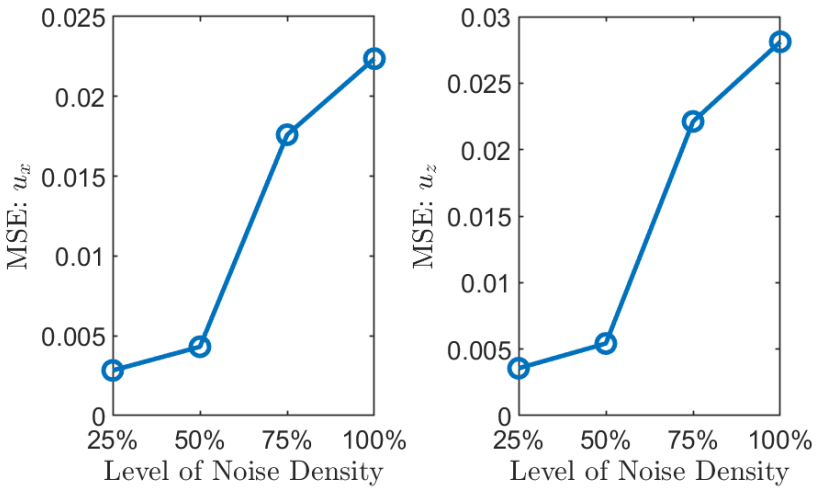

Figure 3.7: Mean Squared Error (MSE) of the estimated and true prediction error for linear velocity (left panel) and angular velocity (right panel) under different levels of noise.

During testing, the robot is required to follow a semicircle trajectory for length $\frac{T}{2}$, i.e. $[p_x^r, p_y^r] = [\sin(\frac{2\pi}{T}t), 1 - cos(\frac{2\pi}{T}t)]$. To avoid bias, testing trials are repeated for 10 consecutive times for an average with the same training set. We compute the Mean Squared Error (MSE) for the direct prediction error of u_x and u_z as defined in (3.2.4), where the **prediction** term is Δu and the **true** term is $(u(K_n) - u(K)$ and the results are illustrated in Fig. 3.7. We also obtain the difference among all trials for the output trajectories and show the results of position p_x as well as p_y in Fig. 3.8.

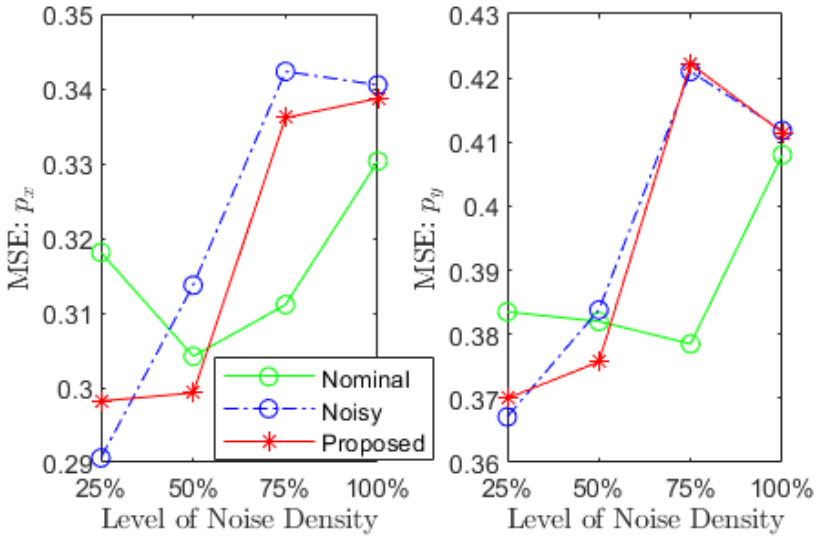

Figure 3.8: MSE of output trajectories' evolution along the p_x axis (left panel) and p_y axis (right panel) under distinct noise levels.

It can be observed that 1) our proposed method can approximate the prediction uncertainty because of noise with a small error (Fig. 3.7). 2) When we implement the estimated uncertainty to a Koopman-based data-driven structure, our '**Proposed**' controller can drive the '**Noisy**' one closer to a desired trajectory. 3) As the noise level increases, our approach performs worse in terms of the prediction error estimation while performing better when we use the estimation for control. 4) Our algorithm remains bounded to the performance of the nominal controller (as the '25%' noise level case in Fig. 3.8). Ongoing work focuses on implementation in different types of Koopman operator-based nominal controllers.

3.4 Summary

In this section, we focus on the sensitivity analysis of the data-driven approximation approaches of the Koopman operator. We investigate the influence of noisy measurements on the accuracy of the approximated model, the model-based prediction, and further the prediction-based controller design.

We first propose the loose and tight bounds for the basic Dynamic Mode Decomposition (DMD) and evaluate the result with a linear example. Moreover, we derive the prediction error explicitly of perturbed systems' performance when the data used for training the Koopman operator via Extended DMD are noisy. The utilization of the prediction error is evaluated in developing an enhanced motion control algorithm that endows robustness to existing Koopman-based data-driven control architectures. We also show how certain aspects of the approach can happen offline, thus making the proposed algorithm amenable to real-time implementation. The performance is illustrated in a parametric simulated study using a Van der Pol oscillator as the number of training data and the level of noise corrupting them vary. We further test in Gazebo simulation with a non-holonomic wheeled robot tasked to track a reference trajectory. Results from both types of testing confirm that the proposed approach can apply across systems, including practical robotics.

In all, the work introduced in this section emphasizes the considerations on the quality of '**Measurements**' of the whole Koopman-based data-driven structure. It builds the step toward estimating more accurate models and developing robust control architectures for robotic platforms.

Chapter 4

Model: Lifting Functions Design

Once we have obtained reliable data, we can estimate the Koopman operator and model the underlying system following the procedures introduced in Sec. 2. For all approaches employing the Koopman operator theory, the way to lift the original system is of high importance because they work as the basis to construct the infinite-dimensional linear approximation of the nominal (nonlinear) system's model and the approximation accuracy of the operator as well as the resulting lifted dynamics can be effected by a poor choice. As discussed in Chapter 1, several different algorithms have been proposed for the selection of the observable functions, while a general structure of the design for robotic systems is lacking.

In this section, a new method to analytically construct dictionaries of lifting functions is proposed for Koopman operator-based data-driven nonlinear robotic systems. Different from existing related methods, this method exploits the fact that robotic systems have certain characteristic properties that can be acquired without knowledge of their ex-

act dynamical models. Properties considered herein are the system's configuration space and workspace. These properties reveal fundamental information regarding the system's states and dynamics and can provide intuition about how to generate the lifting functions required for Koopman operator approximation. We show how fundamental topological spaces and Cartesian products thereof can be mapped to a basis of Hermite polynomials and Kronecker products thereof which serve as the dictionary of lifting functions. Moreover, it is shown that the resulting dictionary is complete and leads to an estimated Koopman operator with provable guarantees of convergence to the true one, in the limit and provided that the observables are weighted bounded. At the same time, the resulting dictionary is simple to implement.

We evaluate the efficacy of this proposed analytical method using a series of both simulated and hardware experiments. We consider a differential drive robot in both simulation and physical experimentation, and a rigid robotic arm comprising two revolute joints in simulation. In these cases, we use the configuration space of the robots. The method can also apply to soft robots (whereby their configuration space is ill-defined), by considering their workspace instead. Physical experimentation using a soft robotic leg that can bend and extend confirms that our approach can apply uniformly across rigid and soft robots, and further demonstrates its practical utility. Results obtained by our algorithm are also compared against other nonlinear dynamics identification approaches in terms of prediction accuracy and training time, to demonstrate our method's efficacy.

4.1 Dictionary Construction

In this section, we present the main technical result, the ACD-EDMD, an Analytical Construction of Dictionaries of appropriate lifting functions for a range of data-driven Koopman-operator-based nonlinear robotic systems. The key insight is that fundamental topological spaces and Cartesian products thereof can be mapped to a basis of Hermite polynomials and Kronecker products thereof. The latter produces the dictionary of lifting functions, which, as we show in the following, enjoys provable convergence guarantees of estimated Koopman operator to the true one when the observables are weighted bounded. Fundamental topological spaces in the context of robotics include the robot's configuration space and its workspace.

Importantly, in our approach we consider both rigid and soft robots. This serves two key purposes. 1) To demonstrate that employing a robot's fundamental topology to yield lifting functions is general and can apply across robotic embodiments. 2) To offer a baseline in which the configuration space or the workspace topology might be preferred one over another. For rigid robots we consider the configuration space. However, soft robots are often considered to contain an infinite number of degrees of freedom, hence the problem of constructing their configuration space is ill-defined [76]. Thus, we consider the workspace instead for the case of soft robots. A flowchart of the proposed method is given in Fig. 4.1.

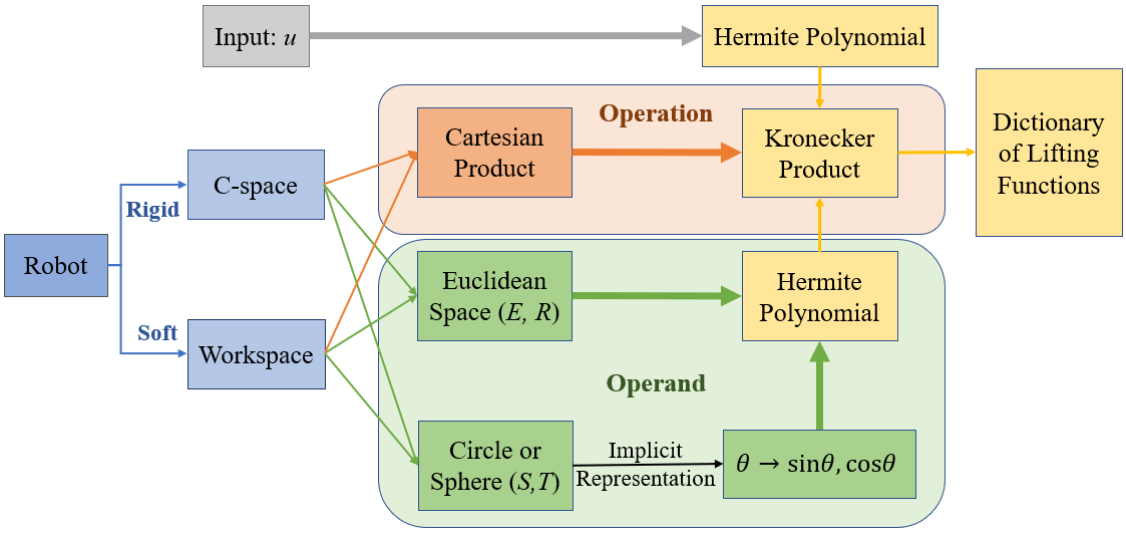

Figure 4.1: Illustration of our proposed method pipeline. Thin arrows indicate the computing sequence and the thick arrows represent how to map fundamental topological spaces to lifting functions. Hermite polynomials establish lifting functions for lower-dimensional spaces that work as the 'operand.' The dictionary of higher-dimensional spaces is formed as the Kronecker product (the 'operation') of sets of Hermite polynomials.

4.1.1 Analytical Construction for Fundamental Topological Spaces

The Euclidean space $\mathbb{E}^n$ is very frequently used in the context of robotics. We use the Hermite polynomials to construct the basis functions. Intuitively, the Hermite polynomials form an orthogonal basis of the Euclidean space (or the Hilbert for higher dimensions).[3] Non-Euclidean spaces which are often employed include the circle S^1, the sphere S^n, and the torus T^n. On the other hand, we elect to represent non-Euclidean spaces by an implicit parameterization of unit complex numbers so that the explicit variables are mapped to a higher dimensional space, e.g., an angle $\theta \in S^1$ will be mapped to $[\sin(\theta), \cos(\theta)] \in \mathbb{E}^2$. Doing so enables the use of Hermite polynomials for a range of both Euclidean and non-Euclidean spaces.

[3] There are two different definitions of Hermite polynomials [77]. We consider the "probabilist's" Hermite polynomial; yet, the same analysis applies to the "physicist's" version too.

Higher-dimensional spaces can be expressed as the Cartesian product of lower-dimensional spaces that contains the union of these spaces. For topological spaces constructed as the Cartesian product, we compute the Kronecker product of the lifting functions of the lower-dimensional spaces as the dictionary for the higher-dimensional space. The Kronecker product can be viewed as a form of vectorization (or flattening) of the outer product so it contains the results of all the elements multiplied from the two sets.

Table 4.1: Configuration Space Topology and Associated Lifting Functions

System	Topology	Sample Representation	# of states
Point on a Line	$\mathbb{E}^1$ or $\mathbb{R}^1$	x	1
Point on a Plane	$\mathbb{E}^2$ or $\mathbb{R}^2$	(x, y)	2
Point on a 3D Space	$\mathbb{E}^3$ or $\mathbb{R}^3$	(x, y, z)	3
Spherical Pendulum	$\mathbb{S}^2$	(θ, ϕ)	2
2R Robot Arm	$\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$	(θ, ϕ)	2
Rotating Sliding Knob	$\mathbb{E}^1 \times \mathbb{S}^1$	(x, θ)	2
Wheeled Robot	$\mathbb{R}^2 \times \mathbb{S}^1$	(x, y, θ)	3

System	Lifting Function $D(x)$ (recall notation: $H_1(x) = [H^0(x), H^1(x)]$)
Point on a Line	$H_1(x)$
Point on a Plane	$kron(H_1(x), H_1(y))$
Point on a 3D Space	$kron(H_1(x), H_1(y), H_1(z))$
Spherical Pendulum	$kron\left(H_1(\sin(\theta)), H_1(\cos(\theta)), H_1(\sin(\phi)), H_1(\cos(\phi))\right)$
2R Robot Arm	$kron\left(H_1(\sin(\theta)), H_1(\cos(\theta)), H_1(\sin(\phi)), H_1(\cos(\phi))\right)$
Rotating Sliding Knob	$kron(H_1(x), H_1(\sin(\theta)), H_1(\cos(\theta)))$
Wheeled Robot	$kron(H_1(x), H_1(y), H_1(\sin(\theta)), H_1(\cos(\theta)))$

We first construct the lifting functions for the state, $D(x)$; some examples of lifting functions are listed in Table 4.1. Note that we have considered zero- and first-order Hermite polynomials (denoted by H^0 and H^1, respectively) but ACD-EDMD may apply when considering higher-order terms as well; investigating this direction is part of

future work. For clarity, we set $H_1(x) = [H^0(x), H^1(x)]$. For the control input, $D(u)$ is computed by the zero- and first-order Hermite polynomials and Kronecker products thereof similarly to the $\mathbb{R}^n$ cases for states as in Table 4.1. This way to include the control input in the dictionary is consistent with that of other related methods that append the control input to the state [78]. Having computed $D(x)$ and $D(u)$, the complete dictionary is then formed as the Kronecker product between lifting functions for states and inputs, i.e. $\mathcal{D} = kron(D(x), D(u))$ (see Fig. 4.1).

4.1.2 Theoretical Analysis of ACD-EDMD

The lifting functions comprising dictionary D are Kronecker products of Hermite polynomials in a single dimension. This set of basis functions is **simple to implement**, and conceptually related to approximating the Koopman eigenfunctions with a Taylor expansion [23]. Furthermore, because they are orthogonal with respect to Gaussian weights, the matrix G in (2.6) will be diagonal if the data are drawn from a normal distribution, which can be beneficial numerically. To work as a general architecture of the dictionary of lifting functions, we also want the construction to be complete and converge to the actual Koopman operator in certain conditions. In this section, we present the completeness and convergence of our proposed structure. The existing remarks are given and then we give out our derived theorems.

Firstly, it has been shown that as $M \to \infty$ (M: size of measurements) the operator $K_{Q,M}$ converges to K_Q (Q: dimension of the observables' dictionary), the orthogonal projection of $\mathcal{K}$ on the subspace spanned by the lifting functions [71]. That result has been extended to analyze the convergence of K_Q to the actual Koopman operator $\mathcal{K}$ [31].

In Remark 1 below we summarized the conditions needed to be satisfied for convergence, which we build upon in our own theoretical analysis presented next. Thus, if the chosen dictionary of lifting functions satisfies the following two conditions, the EDMD-estimated operator converges to the actual one.

Remark 1 *(Adapted from [31]) If 1) the Koopman operator $\mathcal{K}$ is bounded; 2) the lifting functions $\psi_1, \ldots, \psi_N$ are selected from an orthonormal basis of $\mathcal{F}$, then the sequence of operators $\mathcal{K}_Q$ converges to $\mathcal{K}$ as $Q \to \infty$.*

On the other hand, Hermite polynomials are the main functions we used in our algorithm, and they serve as an orthonormal basis (complete orthonorml set) for the Hilbert space [79, 80], which is useful in our theoretical analysis. Remark 2 below elaborates.

Remark 2 *Hermite polynomials form an orthogonal basis of the Hilbert space of functions $g(x)$ satisfying $\int_{-\infty}^{\infty} |g(x)|^2 w(x) dx < \infty$, in which the inner product is given by the integral $\langle g_1, g_2 \rangle = \int_{-\infty}^{\infty} g_1(x) \overline{g_2(x)} w(x) dx$, including the Gaussian weight function $w(x)$.*

Based on the technical preliminaries discussed above, the Hermite polynomials form a **complete** orthogonal basis of the Hilbert space of the weights with exponential decay, i.e. the linear span of the basis is dense [77]. We can approximate the Koopman operator $\mathcal{K}$ with arbitrarily high accuracy by using a sufficiently large number of terms of the basis functions when the observables are weighted bounded in the Euclidean (Hilbert) space. In other words, the EDMD operator using the proposed dictionary enjoys provable conditioned **convergence** guarantees. We deduce the theorem and proof as follows.

Theorem 3.1 *If the* ***observable functions*** $g \in \mathcal{F}$ *satisfy*

$$\int_{-\infty}^{\infty} |g(x)|^2 w(x) dx < \infty \tag{4.1}$$

with the inner product defined as $\langle g_1, g_2 \rangle = \int_{-\infty}^{\infty} g_1(x)\overline{g_2(x)}w(x)dx$ *, where* $\overline{g}$ *denotes the conjugate function, and weighting function* $w(x)$ *is the Gaussian weight function, then the operator* $K_{Q,M}$ *estimated by ACD-EDMD converges to* $\mathcal{K}$ *as the number of samples* M *and number of used lifting functions* N *go to infinity.*

Proof: The proof is decomposed into four steps. 1) The convergence of $K_{Q,M}$ to K_Q is proven in [71] as $M \to \infty$. 2) Condition (4.1) dictates that observables $g(x)$ are bounded. This implies that the Koopman operator $\mathcal{K}$ calculated by these observables is also bounded and hence satisfies the first condition of Remark 1. 3) Per Remark 2, the Hermite polynomials form an orthogonal (which implies orthonormal) basis of the Hilbert space for all the weighted bounded observables g, and hence the second condition of Remark 1 is also satisfied. 4) We have established so far that individual Hermite polynomials satisfy the two conditions of Remark 1. Here we consider the Kronecker product of Hermite polynomials, which nonetheless does not affect the aforementioned properties. Thus, with the proposed lifting functions and if the observables are well-designed (weighted bounded), we have $K_{Q,M} \longrightarrow \mathcal{K}$ as $M \longrightarrow \infty$ and $Q \longrightarrow \infty$. ■

4.2 Experimental Evaluation

We evaluate the efficacy of our proposed analytical method to generate lifting functions for Koopman-operator-based nonlinear robotic systems using a series of both simulated and hardware experiments. We consider a differential drive robot (ROSbot 2.0) in

both simulation and physical experimentation, a rigid robotic arm comprising two revolute joints in simulation, and a soft robotic leg [81, 82] that can bend and extend in physical experimentation.

4.2.1 Data-driven approaches for comparison

EDMD-DL *(EDMD with Dictionary Learning):* For high-dimensional and highly nonlinear systems, machine learning methods can help make selections on lifting functions [41]. In this method, the lifting functions $\mathbf{\Psi}$ are trained and represented by an artificial neural network. Thus, EDMD is coupled with the trainable dictionary. In EDMD-DL [41], given data measurements X the dictionary vector $\mathbf{\Psi}(x_m)$ is parameterized by a universal function approximator, i.e. $\Psi(x) = \Psi(x;\theta)$ for $\theta \in \Theta$ to be varied. Then, a feedforward 3-layer neural network is designed to approximate Ψ that solves the minimization problem (2.3) as

$$\Psi(x) = W_{out}h_3 + b_{out}$$

$$h_{k+1} = tanh(W_k h_k + b_k), \;\; k = 0, 1, 2 \;\; .$$

The set of all trainable parameters is $\theta = \{W_{out}, b_{out}, \{W_k, b_k\}_{k=0}^2\}$. By iterating over two steps: (1) fix θ, optimize K as a least-square problem; then (2) fix K, optimize θ as a standard machine learning problem, the dynamics can be estimated until convergence.

SINDy *(Sparse Identification of Nonlinear Dynamical Systems):* SINDy [83] has been proposed for extracting governing equations of nonlinear systems from data. The method considers that only a few important terms govern the dynamics of an underlying model and uses sparse regression to determine the fewest terms required to accurately

illustrate the system's state evolution based on observed (time-varying) data series. Letting the system be $\dot{x}_t = f(x_t)$, where $f(x_t)$ represents the dynamic constraints that describe propagation rules and is to be identified by data, snapshots of states x_t and their derivatives $\dot{x}_t$ are collected or estimated and arranged into two data matrices X, $\dot{X}$. Then, a library $\Theta(X)$ consisting of candidate nonlinear functions of the column of X is designed. Entries in this matrix of nonlinearities can be chosen with significant freedom. One example consists of constant, polynomial and trigonometric terms [83], i.e.

$$\Theta(X) = \begin{bmatrix} 1 & X & X^2 & \sin(X) & \cos(X) \end{bmatrix} \tag{4.2}$$

A sparse regression problem to calculate the sparse vectors of coefficients $\Xi = [\xi_1, \xi_1, \ldots, \xi_{n_x}]$ is setup as $\dot{X} = \Theta(X)\Xi$. Once Ξ has been determined, the system can be approximated as $\dot{x} = f(x) = \Xi^T(\Theta(x^T))^T$. The process can also be extended to include inputs [84].

Results from ACD-EDMD are compared against those attained by other related methods that are just introduced: 1) Classical EDMD with the dictionary that contains the direct sum of Hermite polynomials of up to second-order terms in all states; 2) EDMD-DL with 25 dictionary outputs (includes one constant non-trainable map, the coordinate projection non-trainable maps and the rest trainable lifting functions); 3) SINDy with nonlinear functions introduced in (4.2) solved with least absolute shrinkage and selection operator (LASSO) [85].

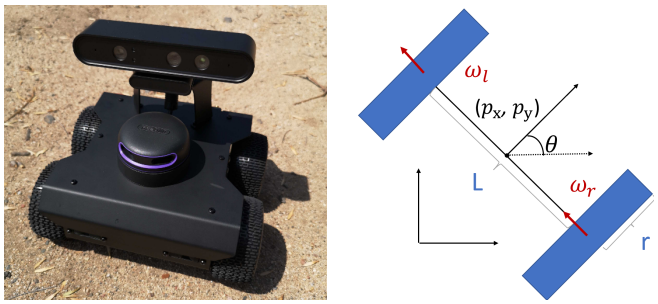

Figure 4.2: ROSbot 2.0 (left) and the differential drive model (right).

4.2.2 Differential Drive Robot - Simulation

The differential drive robot model is [4]

$$\begin{cases} \dot{p}_x = \frac{r}{2}(\omega_r + \omega_l)\cos(\theta) \\ \dot{p}_y = \frac{r}{2}(\omega_r + \omega_l)\sin(\theta) \\ \dot{\theta} = \frac{r}{L}(\omega_r - \omega_l) \end{cases} \tag{4.3}$$

where parameters $r = 0.062$ m and $L = 0.228$ m match those of the physical ROSbot 2.0 (Fig. 4.2). The control input is $u = [\omega_r, \omega_l]$, with ω_r and ω_l being the angular velocities of the right and left wheel, respectively. The state vector contains position $\{p_x, p_y\}$ and orientation θ.

The configuration of the chassis of the differential drive robot model is the Cartesian product of two points and a circle, i.e. $\mathbb{R}^2 \times \mathbb{S}^1$. We first map the orientation θ to the implicit parameterization $\{\sin\theta, \cos\theta\}$. Then, the dictionary of lifting functions is constructed as the Kronecker product of Hermite polynomials of the four variables $\{p_x, p_y, \sin\theta, \cos\theta\}$ of up to first order terms (as shown in Table 4.1).

[4] The model is used as the baseline to create synthetic training data and to get the 'true state' output in simulated testing and evaluation.

For operator learning we simulate 100 trajectories each lasting for 50 sec with sampling rate $T = 0.1s$. To construct the training data, we input a random signal to (4.3) that is sampled from the normal distribution $u \sim \mathcal{N}([0,0]^T, 9^2 I_{2\times 2})$. The covariance magnitude is selected so as to decrease chances of generating control inputs that would be unattainable by the robot or damage it, or could lead to damage to, the physical robot. The learned operator is validated by picking random input signals of length $L = 50$ sampled from the same normal distribution as in the training set. We calculate the Mean Squared Error (MSE) between predicted (superscript p) and true (superscript t) states per $MSE = \frac{1}{L}\sum_1^L([p_x^p, p_y^p, \theta^p] - [p_x^t, p_y^t, \theta^t])^2$. The MSE of the validation set is very low, $MSE =$ [0.0097 mm, 0.0000 mm, 0.0013 rad].

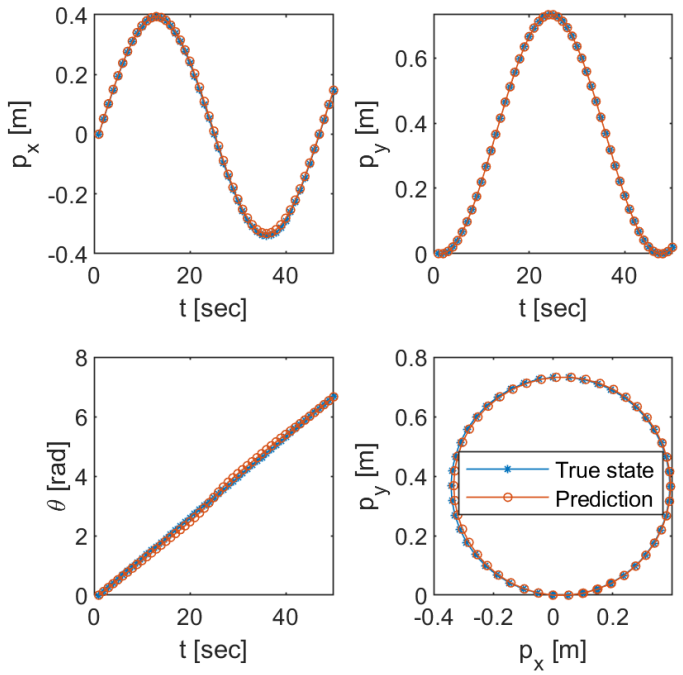

Figure 4.3: Results from testing ACD-EDMD's generalization capacity in making a simulated differential-drive robot follow a circular trajectory using random-input-signal simulated training data.

We then evaluate the learned operator's generalization capacity by testing with input $u = [12.1369, 6.7310]$ rad/sec. This input is designed to make the robot follow a counter-clockwise circular trajectory (starting from the origin). The predicted (by our method) and true (via (4.3)) states are shown in Fig. 4.3. Comparison results with other approaches are listed in Table 4.2. Results indicate that the model learned from the Koopman operator via ACD-EDMD is able to predict a trajectory very distinct from what it was trained on with very small errors. In contrast, the direct combination of Hermite polynomials leads to large errors, while EDMD-DL and SINDy can achieve small errors (though still much larger than ACD-EDMD) but require much longer training times.

Table 4.2: Comparative Results in Simulated Differential Drive Robot

Method	MSE [mm; mm; rad]	Training Time [sec]
Hermite Polynomials	[291.063; 110.825; 0]	2.94
EDMD-DL	[71.616; 1.119; 0.0624]	2080.11
SINDy	[1.495; 1.729; 0]	2631.00
ACD-EDMD (ours)	[0.035; 0; 0.006]	1.97

4.2.3 Experiments with the Physical ROSbot 2.0 Robot

Next, we move on with evaluating the learned operator's generalization capacity by testing using actual experimental data from the physical robot. Note that the dictionary is the same as in the simulated case above, constructed on the basis of random inputs using (4.3). The training dataset is captured based on a random chattering linear velocity and angular velocity input signals for 10 continuous trials. We consider two evaluation cases; **Case 1**: predicting the same counter-clockwise circular trajectory as above; and **Case 2**: predicting a sharp turn trajectory. All experimental data were collected using a

motion capture camera system. We wish to highlight that both trajectories are very distinct from what the learned operator has been trained upon, and contain noise (e.g., the effect of unmodeled dynamics such as friction) which is also not captured by training using the nominal model (4.3).

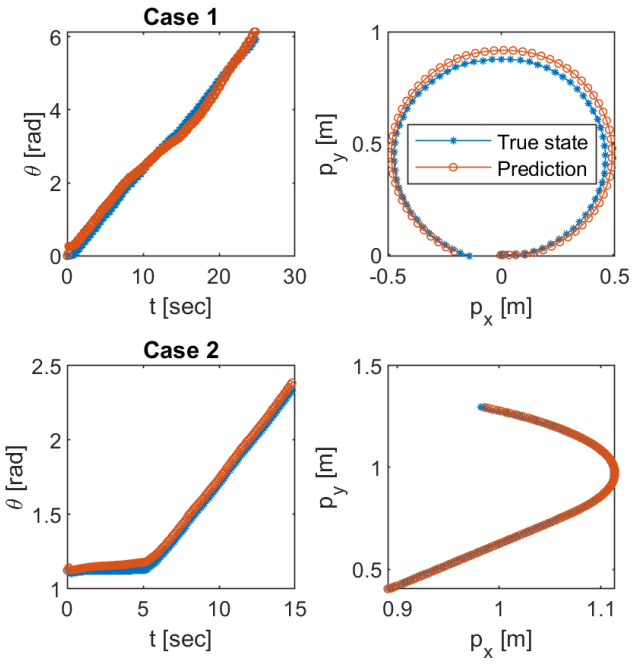

Figure 4.4: Results from testing ACD-EDMD's generalization capacity in making ROSbot follow a circular (top panels) and a sharp turn trajectory (bottom panels) using random-input-signal simulated training data.

The true states (obtained by remote-controlling the physical robot) and the predicted states (obtained by our method using logged control inputs from the experiment) are shown in Fig. 4.4. Results indicate that the model learned from the Koopman operator using random-input simulated data is able to predict *experimental trajectories*, which are also very distinct from the training dataset, with very small error (Case 1 $MSE = [0.0008$ m, 0.0011 m, 0.0211 rad]; Case 2 $MSE = [0.0016$ mm, 0.0154 mm, 0.0011 rad]). There exist

some small discrepancies in the circular trajectory (Case 1, top panels in Fig. 4.4), but overall results predicted using our proposed method capture accurately experimentally the observed trajectories.

4.2.4 Two Revolute Joint (2R) Rigid Robotic Arm - Simulation

We further evaluate our method by testing with a topologically distinct from the wheeled robot system; a 2R rigid robotic arm. The input signal contains the joint torques, that is $\tau = [\tau_1, \tau_2]$. The state vector contains the joint angles $\theta = [\theta_1, \theta_2]$ and their first derivatives (joint angular velocities) $\dot{\theta} = [\dot{\theta}_1\ \dot{\theta}_2]$. The Lagrangian method [86] yields $M(\theta) =$

$$\begin{bmatrix} m_1 L_1^2 + m_2(L_1^2 + 2L_1 L_2 \cos\theta_2 + L_2^2) & m_2(L_1 L_2 \cos\theta_2 + L_2^2) \\ m_2(L_1 L_2 \cos\theta_2 + L_2^2) & m_2 L_2^2 \end{bmatrix},$$

and

$$c(\theta, \dot{\theta}) = \begin{bmatrix} -m_2 L_1 L_2 \sin\theta_2 (2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^2) \\ m_2 L_1 L_2 \dot{\theta}_1^2 \sin\theta_2 \end{bmatrix},$$

$$g(\theta) = \begin{bmatrix} (m_1 + m_2) L_1 g \cos\theta_1 + m_2 g L_2 \cos(\theta_1 + \theta_2) \\ m_2 g L_2 \cos(\theta_1 + \theta_2) \end{bmatrix},$$

then the forward dynamics is given by

$$\ddot{\theta} = M^{-1}(\theta)(\tau - c(\theta, \dot{\theta}) - g(\theta)) \ . \tag{4.4}$$

The configuration space of the 2R arm is the 2-D torus and is homeomorphic to the Cartesian product of two circles, i.e. $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$. We first express the two joint angles as the unit complex number, that is $\theta_1 \mapsto \{\sin\theta_1, \cos\theta_1\}$, and $\theta_2 \mapsto \{\sin\theta_2, \cos\theta_2\}$. Then, the dictionary of lifting functions is constructed as the Kronecker product of Hermite

polynomials of the four variables $\{\sin\theta_1, \cos\theta_1, \sin\theta_2, \cos\theta_2\}$ of up to first order terms (as shown in Table 4.1).

For operator learning we simulate 100 trajectories each lasting for 2 sec with sampling rate $T_a = 0.01$ sec; thus, each trajectory time series has length $L_a = 200$. To construct the training data, we input a random signal to (4.4), sampled from the standard normal distribution (i.e. $\tau \sim \mathcal{N}([0,0]^T, I_{2\times 2})$).

The learned operator is validated by picking random input (time series) signals of length $L_a = 100$ sampled from the standard normal distribution as in the training set. We calculate the MSE between predicted (superscript p) and true (superscript t) states per $MSE = \frac{1}{L_a}\sum_1^{L_a}([\theta_1^p, \theta_2^p] - [\theta_1^t, \theta_2^t])^2)$. The MSE of the validation set is very low, $MSE =$ [0.0003 rad, 0.0009 rad].

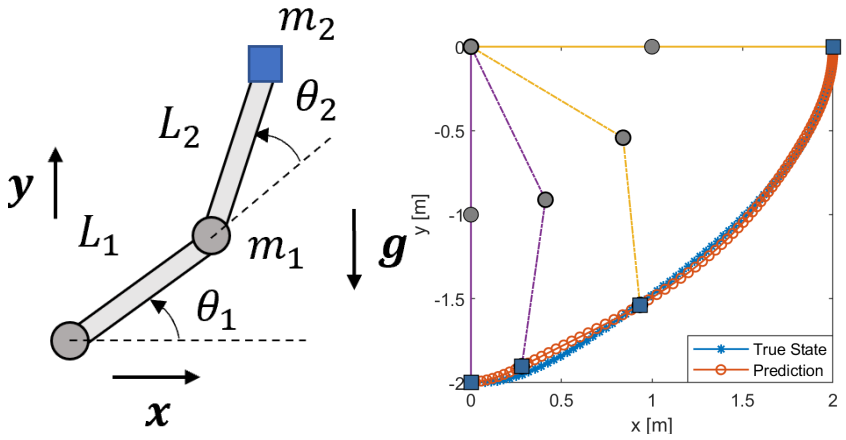

Figure 4.5: Model of 2R robotic arm (left panel). Results from testing generalization capacity in driving the arm to follow a clockwise quadrant trajectory (right panel). The yellow and purple solid lines indicate the initial and final positions. The dashed lines indicate intermediate states of the arm.

We then evaluate the learned operator's generalization capacity by testing with input $\tau = [-1, 0]$ Nm. This input is designed to make the arm's end-effector follow a

Table 4.3: Comparative Results in Simulated 2R Arm

Method	MSE [rad; rad]	Training Time [sec]
Hermite Polynomials	[0.0016; 0.0594]	0.59
EDMD-DL	[0.0000; 0.0000]	280.45
SINDy	[0.0000; 0.0016]	576.92
ACD-EDMD (ours)	[0.0015; 0.0036]	0.96

clockwise circular trajectory (from a horizontal to a vertical configuration in the fourth quadrant). The predicted (by our method) and true (via (4.4)) states are shown in Fig. 4.5. Comparison results are illustrated in Table 4.3. Results indicate that the model learned via ACD-EDMD is able to predict a trajectory very distinct from what it was trained on with very small error. While EDMD with Hermite polynomials requires the shortest time (approximately half of ACD-EDMD) it leads to larger (over one order of magnitude) errors in θ_2. In contrast, EDMD-DL and SINDy produce smaller errors but at the expense of increasing the training time by more than two orders of magnitude.

4.2.5 Experiments with a Soft Robotic Leg

Lastly yet importantly, we turn our attention to soft robotic systems. In contrast to rigid robotic systems whereby their configuration space can be uniquely determined, soft robotic systems do not possess such property. However, the workspace of a soft robot can be determined uniquely according to the control objective and hence we propose employing the workspace topology in the case of soft robotic systems.

The soft robotic system we consider is a soft pneumatic robotic leg [81, 82]. The leg comprises of 1) the bending part, and 2) the extension part (Fig. 4.6). When the two parts

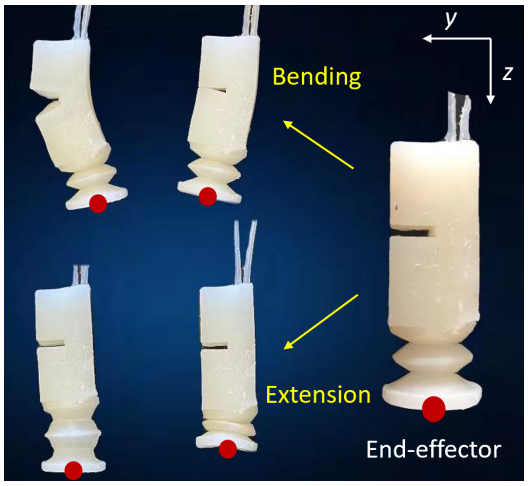

Figure 4.6: The soft robotic leg considered herein shown at distinct operating settings (left: pressurization; middle: depressurization; right: idle).

are simultaneously pressurized, the leg both bends and extends. Pressurization is controlled by two DC 12V 2-way normally closed electric solenoid air valves, one for bending ($|u_1|$) and one for extension ($|u_2|$). Two vacuum pumps are utilized for input (+) and exhaust (-). A motion capture camera system is used to collect position data (y, z) of the tip of the soft leg. The workspace of the tip is $\mathbb{R}^2$ and we observe both y and z. Hence, the dictionary to be used in ACD-EDMD is formed as the Kronecker product of $H_1(y)$ and $H_1(z)$.

We adopt a two-pronged training and evaluation method. First, we conduct individual tests for the two parts by setting one input at zero. Then, we pressurize/depressurize both bending and extension parts to mimic a foot path [81, 82]. In training, a constant voltage input signal is given to the valves that control active leg parts. Each signal lasts for 2 sec; training inputs for different tests are listed in Table 4.4. Note that in Case 3 we train the system with both positive and negative listed values. Collected data are post-processed by a moving-average filter with window length of 5.

Table 4.4: Training Inputs for the Soft Robotic Leg Experiments

Case 1: Individual test for bending part	$u_1 = +[3.12, 3.61, 3.64, 5.92, 7.97]V$ $u_2 = [0.00, 0.00, 0.00, 0.00, 0.00]V$
Case 2: Individual test for extension part	$u_1 = [0.00, 0.00, 0.00, 0.00, 0.00]V$ $u_2 = +[3.12, 3.64, 4.30, 5.92, 6.92]V$
Case 3: Test on the the whole robot leg	$\|u\| = [(3.85, 4.55), (3.85, 12.03), (4.63, 4.63),$ $(6.16, 6.16), (9.30, 9.30), (11.24, 4.55), (11.24, 12.03)]V$

To evaluate the generalization capacity of ACD-EDMD, we design experiments in three cases. In Cases 1 and 2, the input signal is a combination of two constant voltages (not used in training) each lasting for 1 sec. In Case 1, we focused on the bending part. The testing input is designed as $u_1 = 4.30$ V for 1 sec followed by $u_1 = 3.19$ V for another 1 sec, while $u_2 = 0$ V for 2 sec of the whole process. The extension part is evaluated in Case 2. The input sequence is $u_1 = 0$ V for 2 sec, while $u_2 = 3.22$ V for 1 sec and $u_2 = 3.61$ V for the following 1 sec.

Results in Fig. 4.7 suggest that ACD-EDMD (in spite of using a very small dataset) offers a good prediction in the individual parts tests (Case 1 $MSE = [0.1423, 0.0068]$ mm and Case 2 $MSE = [0.0005, 0.0122]$ mm). In Case 3 where both the bending and extension parts are actuated, we first drive the robot to curl up ($u = [-3.88, -7.74]$ V) then extend as well bend ($u = [4.63, 3.50]$ V), and finally curl up again ($u = [-4.54, -3.88]$ V) to complete one foot trajectory cycle. Results shown in Fig. 4.8 demonstrate that the proposed method, despite its reduced performance compared to the single-part tests in Cases 1 and 2, can still predict a relatively accurate trajectory for the full actuator.

In Table 4.5, we report comparative results in Case 3 between ACD-EDMD and the other related approaches. Results show that our approach achieves similar error as

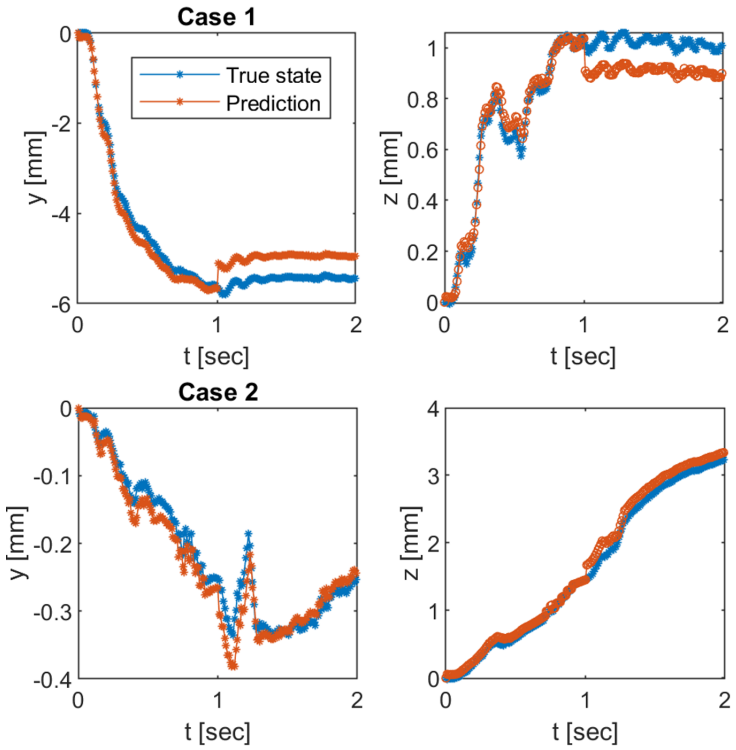

Figure 4.7: Results from testing generalization capacity when actuating only the bending part (top panels) and only the extension part (bottom panels).

SINDy, but at significantly lower (two orders of magnitude) training time. EDMD-DL attains the smallest error but the training time is four orders of magnitude larger compared to ACD-EDMD's training time. Direct use of Hermite polynomials in EDMD results to the lowest accuracy.

Table 4.5: Comparative Results in Soft Robotic Leg Experiments

Method	MSE [mm; mm]	Training Time [sec]
Hermite Polynomials	[96.357; 2.940]	0.11
EDMD-DL	[0.095; 0.013]	151.19
SINDy	[0.507; 0.216]	2.77
ACD-EDMD (ours)	[0.464; 0.616]	0.08

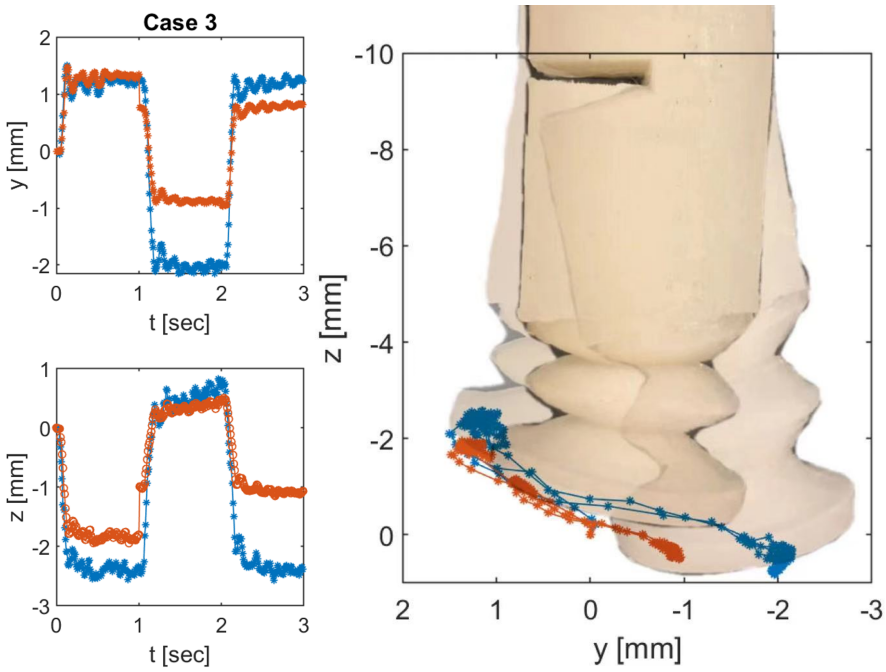

Figure 4.8: Test results for Case 3 that the robotic leg follows a walk trajectory. The blue curve with star indicates the true state and the red curve with circle represents the prediction. In the right panel we show how the leg moves; less transparent snapshots indicate later states. (Best viewed in color.)

4.3 Summary

In this chapter, we address the challenge we meet when estimating the model of robotic systems with the Koopman operator theory - lifting functions selection. The main scientific premise is that robotic systems exhibit certain characteristic properties which can be exploited to inform the design of appropriate lifting functions for use in the context of modeling and control of data-driven Koopman-operator-based robotic systems. Selection of an appropriate set of lifting functions directly affects the Koopman operator's ability to extract dynamics from data, and to-date several successful approaches using Koopman operator theory in the context of robotics employ empirically-selected lifting functions.

We provide an analytical way to design lifting functions based on fundamental topological spaces for robots (i.e. their configuration space if rigid and their workspace if soft) using Kronecker products of Hermite polynomials. Our proposed method, termed ACD-EDMD, is simple to implement and enjoys provable guarantees of completeness and convergence when the observables are weighted bounded. Evaluation results using a range of diverse robotic systems and tested both in simulation and via physical hardware experiments reveal that our proposed method can generalize well and predict accurately the robot's state evolution when compared with related model estimation approaches. This finding is persistently observed especially in rigid robots, and even when testing the method's generalization capacity in very different cases. An example is the wheeled robot learning to follow specific patterns (e.g., a circular trajectory) while trained on (simulated) random-input-signal trajectories.

Comparison with other related dynamics identification methods indicates that ACD-EDMD can achieve high prediction accuracy and low training times in all tested cases. Prediction accuracy is overall much higher than the baseline method of using directly EDMD with a sum of Hermite polynomials, on par to SINDy and in some cases higher than EDMD-DL (although the latter requires manual tuning of hyper-parameters to perform well). Training time is overall on par to the baseline case and at least two orders of magnitude faster than SINDy and two-four orders of magnitude faster than EDMD-DL. The fast training times of ACD-EDMD, compared with its high accuracy, make it beneficial for implementation in physical robots and toward online Koopman-based modeling and control architectures.

With this analytical and general structure of the lifting functions design, we can now formulate the problem and get a '**Model**' estimation of the robot via the Koopman operator theory. This is the key component of our overall developed architecture and also the basis of the following controllers' design.

Chapter 5

Control: Design and Applications to Robotic Platforms

A system model plays a crucial role in enabling the design of model-based controllers that utilize model predictions to determine appropriate control inputs for a given task. As discussed in earlier chapters, the Koopman operator theory allows us to derive either linear or nonlinear models from measurements. These models can be seamlessly integrated into the design of model-based controllers. The performance of these controllers heavily relies on the quality of the underlying model. By considering new measurements and/or offline measurements, the Koopman operator can be employed as a model constraint within the control structure formulation, thereby influencing the overall performance of the controllers.

In this chapter, we will introduce two Koopman-based control architectures. First, we present the Koopman in Model Predictive Control (MPC), which is a widely used and

popular model-based structure. In this approach, the control input is optimized over a finite time horizon, applied for a single timestep, and the optimization process is repeated. We develop an online control algorithm that utilizes the learned and continuously-refined Koopman-based models in real time. This algorithm adapts control inputs for the MPC structure accordingly. To validate the effectiveness of the approach, we conduct experiments using two soft grippers to grasp objects with varying shapes and weights. Notably, our algorithm does not require any explicit prior information about the objects. Second, we illustrate a hierarchical structure that leverages the Koopman operator to handle uncertainty. This approach relies on the aforementioned technique to learn a higher-level controller. The higher-level controller then generates a refined reference to enhance the performance of an existing lower-level controller. We prove the stability of this design and evaluate the structure using simulations and experiments with micro-aerial robots.

5.1 Online Modeling and Control of Soft Grippers

In this section, we present an online modeling and control approach for soft multi-fingered robotic grippers based on Koopman operator theory. First, dictionaries of lifting functions are designed based on the workspace of the soft gripper. Then, the approach ACD-EDMD described in the previous section 4, is used to learn and update the model at every time step in real-time without any complicated dictionary learning. Updated models, in turn, provide the dynamics constraints within an MPC structure to help achieve desired control objectives. The algorithm is evaluated and compared via physical experiments the prediction accuracy of different modeling methods among different datasets using two design of soft grippers.

Figure 5.1: Instance of the soft gripper considered herein grasping a soft ball using the proposed online learning-based method. The method is designed to be agnostic to the shape and/or weight of the object to be grasped.

5.1.1 Soft Gripper System Design and Properties

In this work we consider soft multi-fingered robotic grippers that are pneumatically actuated. A rigid sub-system is used for gross transport and positioning of the wrist plate, which contains the soft fingers (actuators) that are responsible for object grasping (Fig. 5.1). Note that motion of the rigid sub-system is not included in the model we aim to learn. Instead, we only consider the grasping behavior of the soft fingers in this paper.

Soft Gripper Design and Integration

The soft gripper comprises three identical bending actuators (Fig. 5.2) which are retrofitted PneuNets [87]. These actuators use pneumatic pressurization and depressurization applied to in-series connected deformable chambers to create rotational motion. The soft actuators are made via silicone (Smooth-On Dragon Skin 30) casting in 3D-printed molds (with carbon-fiber reinforced Onyx filament), and are attached to 3D-printed plates. We consider two distinct plates for arranging the soft fingers, one where the three fingers are arranged in a symmetric manner and one that is asymmetric (Fig. 5.3).

The soft gripper assembly is attached to the ReactorX 150, a rigid manipulator with four links and four joints.[5] The plate acts as a wrist affixed to the manipulators' endpoint, which first moves the soft gripper to a desired position and then lifts the soft gripper after it has grasped an object. Note that the focus of this work is on the soft gripper but not the rigid manipulator, assessing how our proposed Koopman operator-based algorithm can enable the soft gripper to grasp objects of different shapes and weights.

[5] The original fifth joint of the robot is a rigid pincher which is removed for the purposes of this work.

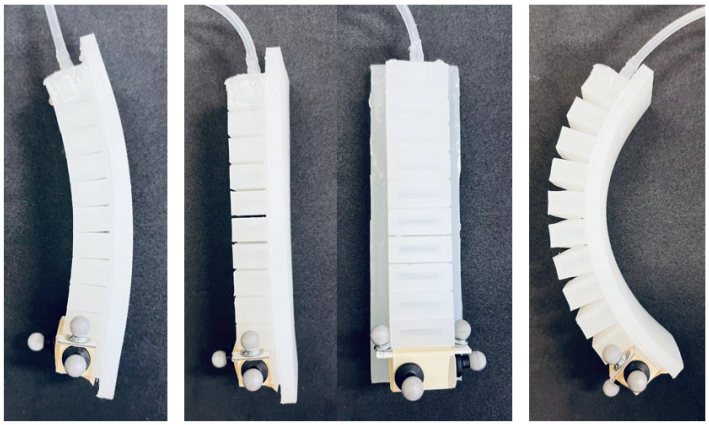

Figure 5.2: Soft finger depressurized (left), idle (middle) and pressurized (right).

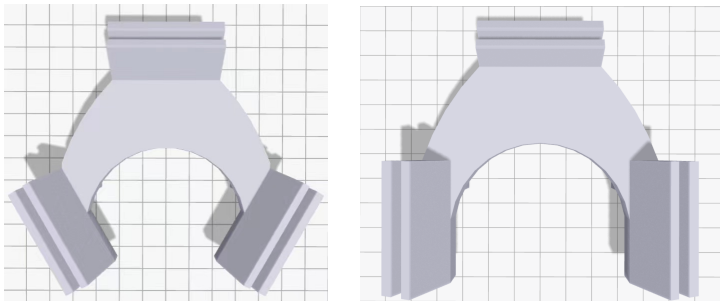

Figure 5.3: CAD renderings of the rigid attachment plates considered herein that afford symmetric (left) and asymmetric (right) soft finger arrangements.

Pneumatic Actuation

The soft fingers are pneumatically actuated using Programmable-Air boards (Fig. 5.4). The board is based on the Arduino nano (ATMega328P), and has a maximum pressure of $\pm$ 50 kPa and flow rate of 2 $liters/min$. Pressurization is controlled by changing the PWM signal sent to pumps. Two such boards were considered for the experiments; the detailed configuration is discussed in Section 5.1.3.

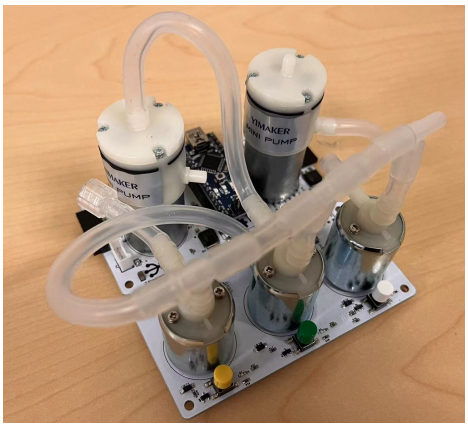

Figure 5.4: The Programmable-Air Pneumatic Control Board.

Workspace Construction

As discussed in Chapter 4, the workspace of soft robots needs to be defined to obtain the dictionary of lifting functions, which is used in the Koopman operator-based modeling approach. The workspace of a robot is the set of positions that it can reach. For soft multi-fingered grippers, the position of the tips of the fingers can be considered to construct the workspace. The state of the soft gripper is thus defined by the concatenation of the positions of each tip as $\xi_1 = [x_1, y_1, z_1]$, $\xi_2 = [x_2, y_2, z_2]$, and $\xi_3 = [x_3, y_3, z_3]$, expressed in terms of an inertial frame set at the center of the plate (Fig. 5.5).[6]

We can further simplify the workspace used for operator learning by exploiting the physical constraints underlying the $\{x_i, y_i, z_i\}, i = 1, 2, 3$ state vectors. The types of actuators considered herein are constrained to create planar motion and can only bend about the normal vector to the plane they are constrained to create motion at (which is determined herein based on the arrangement of the rigid attachment plate). This way,

[6] Although this work uses a 3-fingered soft gripper, the proposed approach can be directly extended to account for general n-fingered soft grippers.

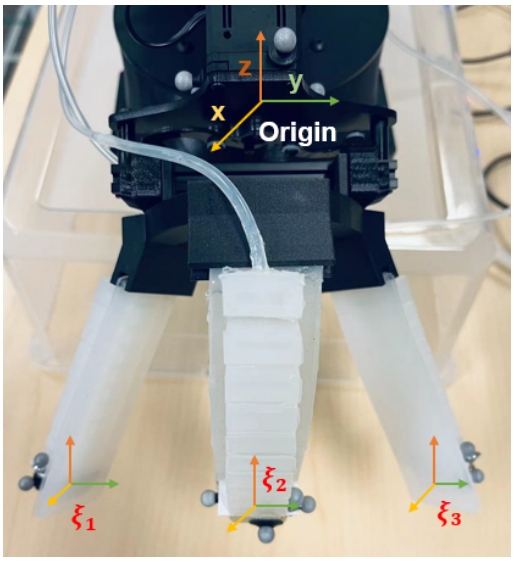

Figure 5.5: Coordinate systems and motion capture marker configuration.

only one of $\{x_i, y_i, z_i\}$ for each finger suffices, thus giving rise to a workspace for the whole soft gripper of $\mathbb{R}^1 \times \mathbb{R}^1 \times \mathbb{R}^1$. Without loss of generality, here we choose z_i as the representations of sub-workspace for each of the fingers, but any other state component would work as well. Besides the Kronecker product of sub-dictionaries, we also include in the library observations of other states and inputs for full-observability. We define $D_\xi = [1, z_1, z_2, z_3, z_1z_2, z_1z_3, z_2z_3, z_1z_2z_3]$. The dictionary is computed as $D = [\xi_1, \xi_2, \xi_3, kron(\mathbf{u}, D_\xi)]$. After eliminating redundant terms, the final dictionary contains 29 terms.[7]

[7] It is beneficial to seek to construct the smallest possible dictionary that best describes the system at hand for computational expediency. At times, doing so relies on exploiting engineering intuition and/or any system symmetries. In this work we take advantage of the bending-only motion afforded by the specific fingers we consider. However, our method is general and can directly apply to any type of soft fingers that afford unconstrained position (and orientation) control of their tips in full 3D space, albeit at the expense of increasing the size of the constructed dictionary.

5.1.2 Online Modeling and Control

Unlike rigid robotic systems whereby it is known how to construct their dynamics, soft robotic systems possess complicated and, most often, infinite-dimensional dynamics. Thus, data-driven approaches may be a better fit for extracting models for soft robotic systems directly from data. However, models learned offline or in simulation (a technique often employed in relevant works) may be invalidated at runtime due to the presence of uncertainty, unless a very delicately designed experimental environment and operation process are imposed for the dynamics learned offline to remain valid for online model-based control. This process can get very tedious and is not scalable. Instead, online control should require limited data and limited time to enable online computation, which are key properties enabled via Koopman operator-based methods [88, 68]. Thus, we propose an online modeling approach using Koopman operator theory to update dynamics constraints in real time and adjust inputs accordingly within an MPC structure. Note that while we consider MPC in this work, the continuously refined learned model can be used in any other online model-based control algorithm.

MPC determines control inputs at each time step by solving an optimization problem among a range of prediction horizon H_p constrained by robot dynamics. The general optimization algorithm can be formulated as

$$\min_{u_i} \;\; J((u_i)_{i=0}^{H_p-1}, (\xi_i)_{i=0}^{H_p},),$$

$$\begin{cases} \xi_{i+1} = f(\xi_i, u_i), \;\; i = 0, \ldots, H_p - 1, \\ \xi_0 = \xi_{initial}. \end{cases}$$

In this work, the soft grippers' dynamic constraints are obtained via ACD-EDMD in real time. In other words, at each time step, we solve an optimization problem using an updated learned model. The cost function J is usually user-designed. Herein, we set a desired trajectory r and minimize the difference between predicted states and the desired points. With these considerations in place, the MPC problem solved here at each time step t is expressed as

$$\min_{u_i} \quad J = ((u_i)_{i=0}^{H_p-1}, (\xi_i - r_i)_{i=0}^{H_p},), \tag{5.1}$$

$$\begin{cases} \xi_{i+1} = \text{ACD-EDMD}(\xi_i, u_i), \quad i = 0, \ldots, H_p - 1, \\ \xi_0 = \xi_t. \end{cases}$$

We are now ready to present the proposed online modeling and control algorithm (Alg. 2). After initialization, at each time step t, control signals are obtained from the MPC controller with the updated model constraints learned based on data from the previous time step. Then, the physical robot is driven by the control inputs and observations are collected to the online training dataset. A refined model estimated from the updated dataset via ACD-EDMD is then used to create updated dynamic constraints for the following step. The process is repeated until the control task (here grasping different objects) is achieved.

5.1.3 Experimental Testing and Results

To test the efficiency of our online modeling and control algorithm, we evaluate separately modeling accuracy and control performance. First, we compare the performance of different model-estimation approaches in online and offline datasets to show the benefits of training with online observations. We further compare the ACD-EDMD based online

Algorithm 2 Online Modeling and Control

Initialization: Evolve the system with random valid inputs for the first M time steps. Then, compute the estimated Koopman operator from the observations:

$$K = \text{ACD-EDMD}(\{\xi\}_1^{M+1}, \{u\}_1^M)$$

for $t \geq M$ **do**

- ***MPC Controller***: Calculate the control inputs with the model constraints learned in the previous step by solving optimization (5.1).
- ***Plant***: Propagate the soft robotic gripper with $u = [u_1, u_2]$: $\xi_{t+1} = f(\xi_t, u)$, where f is the evolving law of the real plant.
- ***Model***: Collect new data from observations and update the ACD-EDMD model.

$t \leftarrow t + 1$

modeling estimation accuracy with SINDy (Sparse Identification of Nonlinear Dynamical Systems (SINDy) [83]) and LSTM (Long Short-Term Memory [89]) using online training datasets. Then, we implement our online control algorithm in the soft gripper and evaluate it in grasping different objects.

Motion capture system (12-camera Prime13 Optitrack) is used to collect position data (x, y, z) of the tip of each finger. To ensure consistency among all training data, we set the origin of the inertial frame at the geometric center of the attachment plate (Fig. 5.5). In

both of the grippers, one control board controls soft fingers 1 and 3, and the other control board controls finger 2. Thus, together with the PWM signals of two control boards, 9 states and 2 inputs of the robotic soft gripper are observed.

Settings of Different Approaches to Compare Against

SINDy is a data-driven approach to extracting governing equations of nonlinear systems as introduced in Sec. 4.2.1. It builds on the assumption that only a few key nonlinear candidates govern the evolution of the original system. Sparse regression determines the weights of those nonlinear terms from observation data. Here we adopt the example in [83] as the library $\Theta(X)$ of candidate nonlinear functions, consisting of constant, polynomial and trigonometric terms, that is $\Theta(X) = \begin{bmatrix} 1 & X & X^2 & \sin(X) & \cos(X) \end{bmatrix}$. The sparse regression problem is solved by least absolute shrinkage and selection operator (LASSO) [85].

LSTM is an artificial recurrent neural network that offers feedback connections and the ability to deal with entire datasets rather than single data points. It has been used in robotics applications for state prediction (e.g., [90]), notably in-hand robotic manipulation [91]. In this paper, we use the LSTM layer in MATLAB Deep Learning Toolbox to construct the network. A single layer of 20 hidden units is applied as designed in [92].

Model Estimation Accuracy

Firstly, we test the model estimation accuracy of the aforementioned approaches among different settings of datasets to illustrate the benefits of utilizing online data for model extraction of unknown soft robotic components. Different datasets for training are collected as follows.

1. **Unloaded Offline training data**: Constant PWM signals are given by the control board. Each signal lasts for 16 sec with a sampling rate of 2 Hz (0.5 sec). Preliminary testing revealed that the finger can function effectively when the PWM is within the range of 20% to 35%. Thus, we collect the position and voltage signals data by either pressurizing or depressurizing all the fingers when the gripper is unloaded within that range. A total number of 32 sets are collected as the 'Unloaded Offline' training data. A moving-average filter with window size of 5 is applied to the offline training data.

2. **Loaded Offline training data**: Data from four trials of grabbing plush oranges (object 1 in Fig. 5.6), two trials of grabbing the smaller rectangular box (object 2 in Fig. 5.6) and two trials of grabbing the larger square box (object 3 in Fig. 5.6) are collected with length of 26 points for each trial. The 'Loaded Offline data' case is constructed as a combination of these 8 data sets.

3. **Online training data**: Online observations are collected when the gripper attempts to grasp the plush orange. Let N_T be the training data length; every $\xi^t_{t-N_T-1}$ observations are used to predict ξ_{t+1}.

The three aforementioned datasets are used to predict ξ_{t+1} for the whole online data trial of length L. We calculate the Mean Squared Error (MSE) between predicted (superscript p) and true (superscript t) states per $MSE = \frac{1}{L-N_T}\sum_{N_T}^{L}([\xi_1^p, \xi_2^p, \xi_3^p] - [\xi_1^t, \xi_2^t, \xi_3^t])^2$ for the whole trajectory. Results are contained in Table 5.1. For all algorithms, predictions computed from the *loaded* data are more accurate than those made based on *unloaded* data. Further, **online** approaches lead to significantly smaller error than all **offline** approaches.

Figure 5.6: Objects of different shape and/or weight considered in this work. Sample trials can be viewed at `https://youtu.be/i2hCMX7zSKQ`.

We also compare the efficiency of all methods with online training data by evaluating repeated grasping of object 1 for 8 times to avoid bias. We tested all three approaches with different training data size, $N_T = \{3, 5, 10\}$. Results are depicted in Table 5.2 (training time) and Fig. 5.7 (prediction accuracy). Our algorithm can run $2\times$ and $3\times$ **orders of magnitude faster** than SINDy and LSTM, respectively, per each time step. This performance is attainable because ACD-EDMD leverages fundamental characteristics of the dynamical system at hand (herein the workspace of the soft gripper) which leads to a simultaneously smaller and more descriptive dictionary to use.

Further, with reference to Fig. 5.7, our method can achieve similar accuracy with LSTM for a smaller training data size, and comparable accuracy for a larger training data size. (More specifically, our method can lead to more variable accuracy compared to LSTM for $N_T = 10$, but mean values are close.) The combination of comparable accuracy but substantially faster training time makes our proposed approach appropriate for online learning-

Table 5.1: Prediction accuracy of different approaches

Method	*Dataset*	MSE($\lvert\xi_1\rvert$)	(magnitude)
SINDy	Offline (Unloaded)	$[0.00, 556.66, 0.062]$	$\times 10^3$
ACD-EDMD	Offline (Unloaded)	$[0.26, 256.62, 5.76]$	$\times 10^2$
SINDy	Offline (Loaded)	$[364.17, 38.57, 0.74]$	$\times 10^1$
ACD-EDMD	Offline (Loaded)	$[211.90, 75.52, 0.0026]$	$\times 10^0$
SINDy	Online	$[0.0047, 0.0113, 0.0006]$	$\times 10^0$
LSTM	Online	$[0.58, 1.40, 0.0136]$	$\times 10^{-2}$
ACD-EDMD	Online	$[0.32, 2.16, 1.58]$	$\times \mathbf{10^{-4}}$

Method	*Dataset*	MSE($\lvert\xi_2\rvert$)	(magnitude)
SINDy	Offline (Unloaded)	$[340.17, 0.44, 188.47]$	$\times 10^3$
ACD-EDMD	Offline (Unloaded)	$[180.83, 7.12, 133.35]$	$\times 10^2$
SINDy	Offline (Loaded)	$[0.95, 0.99, 26.59]$	$\times 10^1$
ACD-EDMD	Offline (Loaded)	$[0.43, 0.13, 0.47]$	$\times 10^0$
SINDy	Online	$[1.3600, 1.2400, 0.0900]$	$\times 10^0$
LSTM	Online	$[0.38, 0.76, 0.60]$	$\times 10^{-2}$
ACD-EDMD	Online	$[0.17, 0.93, 0.20]$	$\times \mathbf{10^{-4}}$

Method	*Dataset*	MSE($\lvert\xi_3\rvert$)	(magnitude)
SINDy	Offline (Unloaded)	$[12.84, 327.78, 0.014]$	$\times 10^3$
ACD-EDMD	Offline (Unloaded)	$[7.91, 141.33, 0.69]$	$\times 10^2$
SINDy	Offline (Loaded)	$[122.65, 0.13, 3.25]$	$\times 10^1$
ACD-EDMD	Offline (Loaded)	$[62.36, 15.21, 0.05]$	$\times 10^0$
SINDy	Online	$[0.4400, 0.0022, 0.0300]$	$\times 10^0$
LSTM	Online	$[1.45, 1.81, 0.87]$	$\times 10^{-2}$
ACD-EDMD	Online	$[0.40, 1.35, 0.15]$	$\times \mathbf{10^{-4}}$

based modeling and control tasks. The bottom row panels of Fig. 5.7 show that the prediction error obtained by SINDy is much larger than the other two approaches.[8] SINDy's worse than expected performance may be because of the small training data size used in the online procedure, or the candidate nonlinear functions selected based on the original publication [83]. This is in fact one key point of departure from SINDy in that it requires selection of candidate nonlinear functions which can affect the overall performance.

[8] For error bars to be visible, we omitted those of SINDy in $N_T = \{3, 5\}$.

Table 5.2: Training times per each time step for different approaches

Training Size \ Approach	ACD-EDMD	SINDy	LSTM
$N_T = 3$	0.00028 s	0.0260 s	0.7240 s
$N_T = 5$	0.00030 s	0.0583 s	0.7837 s
$N_T = 10$	0.00044 s	0.1542 s	0.8468 s

Online Modeling and Control: Grasping

For the final evaluation of our method, we apply Alg. 2 to grasp objects using the two soft grippers with different finger configurations. The online training size N_T is 5 and the prediction horizon H_p is set at 3. In this experiment, we manually designed desired trajectories for each finger.[9] The same desired trajectory is used for grasping objects of different shape and same shape but different weight (Fig. 5.6), without any information about the objects. Each grasping task is repeated 10 times to avoid bias. Successful grasps are considered those that the gripper grasps the object, holds it until fully lifted and keeps it for 3 sec after lifting.

Table 5.3: Success rates of grasping different objects

Object	1	2	3	4	5	6
Gripper	Symmetric Gripper					
Success Rate	100%	100%	90%	10%	0%	0%
Gripper	Asymmetric Gripper					
Success Rate	90%	80%	100%	90%	70%	0%

Results show that the asymmetric gripper functions better than the symmetric one in terms of grasping heavier objects, whereas the symmetric gripper performs better

[9] Note that this experiment is only one example to show how our online modeling and control approach can be implemented in practice. Other model-based controllers and desired trajectories can be used as well, and one can embed our online modeling method into their design directly, which is one key benefit of our proposed method.

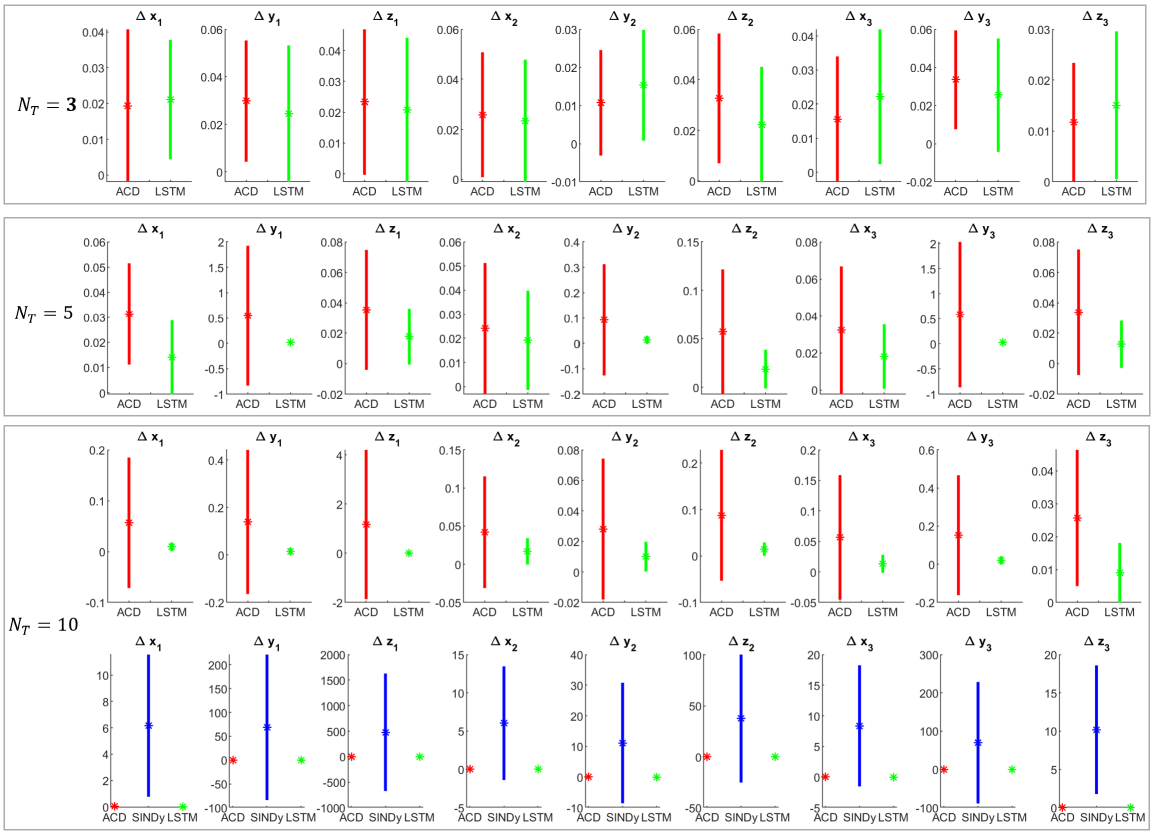

Figure 5.7: 1-σ error prediction accuracy of all online methods. SINDy (blue) has large errors; to keep the bars visible, one case ($N_T = 10$) is shown.

on grasping objects of varying shape (Table 5.3). However, we can still deduce that our algorithm can be easily implemented into different robots to grasp different objects without requiring any knowledge about the robot and the objects.

5.2 Data-driven Hierarchical Control (DHC)

In this section, we propose a Data-driven Hierarchical Control (DHC) architecture. This approach can be particularly appropriate in practice when a low-level, high-rate, pre-tuned controller is already in place, and a second higher-level controller is wrapped around the lower-level one to allow for the system to operate under uncertainty.

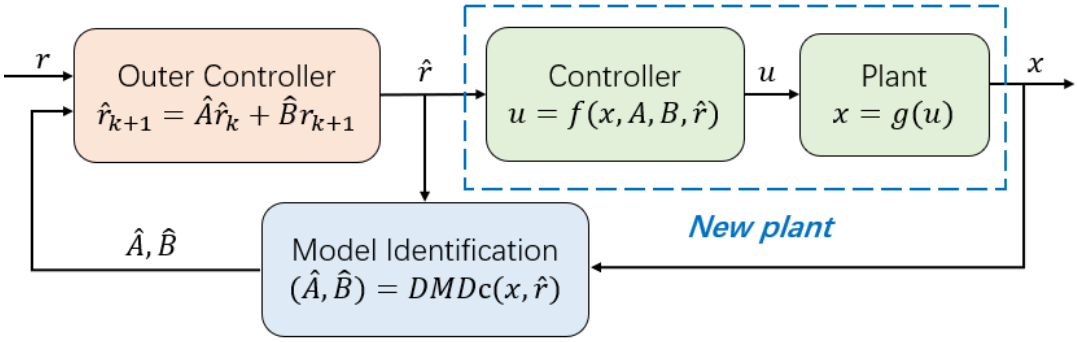

Figure 5.8: Overview of the Data-driven Hierarchical Control (DHC) structure proposed in this work. Model identification (via spectral methods) is combined with a linear controller to adjust the reference input given to an existing (pre-tuned) lower-level controller.

5.2.1 Controller Structure

Given an existing low-level controller for a plant of interest, the proposed DHC structure adds a model identification block and a higher-level ("outer") controller to extract uncertainties and remedy for them in real-time, respectively (Fig. 5.8). The low-level controller and the original plant are combined into a new plant. The model identification block is built from DMDc and estimates linear operators $\hat{A}$ (which is notated as K_A in (2.3)) and $\hat{B}$ from new plant output and reference signals. The identified operators are then used by the higher-level controller to refine the reference signal sent to the new plant.

The control procedure can be specified as follows. At timestep $t+1$, the outer controller refines the original reference r_{t+1} given to the lower-level controller as per $\hat{r}_{t+1} = \hat{A}_t\hat{r}_t + \hat{B}_t r_{t+1}$. To estimate $\hat{A}_t$ and $\hat{B}_t$ using DMDc, we treat the measured state x_t as the "input" and the refined references $\hat{r}_t$ and $\hat{r}_{t+1}$ as the evolving "output." The reason for doing so is that the controller operates on the reference signal, which in turn needs to be "learned" based on the observed system state.

Then, DMDc operates on the linear approximation $\hat{r}_{t+1} = \hat{A}_t\hat{r}_t + \hat{B}_t x_t$. Note that to avoid singularities, we check the rank of the measurement matrices used to approximate models. After an initialization for M steps, the identification process repeats to refine the estimated model until the control task is finished (Algorithm 3). From a general standpoint, (2.4) can be solved when $N_x + N_u + 1$ linearly independent measurements are available. Thus, a lower bound for M is $N_x + N_u + 1$. In this work we set $M = N_x + N_u + 1$.

5.2.2 Stability Analysis

In this part, we will prove that the hierarchical structure would not have ana effect on the stability of the original low-level controller. Let the underlying low-level controller be stable, and consider the origin as the reference, i.e. $r_t = 0$ (other cases can be straightforwardly generalized via states transformation). Given the Lyapunov function $V(x_t) = x_t^T x_t$, we calculate

$$\Delta V(x_t) = V(x_{t+1}) - V(x_t) = g(\hat{r}_{t+1}, w_{t+1})^T g(\hat{r}_{t+1}, w_{t+1}) - x_t^T x_t.$$

When $\hat{r}_{t+1} = \hat{A}_t\hat{r}_t + \hat{B}_t r_{t+1}$, we have $g(\hat{r}_{t+1}, w_{t+1}) = r_{t+1} = 0$. Then,

$$\Delta V(x_t) = r_{t+1}^T r_{t+1} - x_t^T x_t = 0 - x_t^T x_t \rightarrow \Delta V(x_t) = -x_t^T x_t \ .$$

Thus, V decays over time and satisfies $V(0) = 0$. From the standard Lyapunov argument, the equilibrium $r_t = 0$ is then stable. Hence, wrapping the outer controller around the underlying plant respects the stability properties of the original lower-level controller.

Algorithm 3 DHC Procedure

---

initialize: Set $\hat{r}_i = r_i$ and evolve the system for the first M time steps.

Then compute $(\hat{A}_0, \hat{B}_0) = DMDc(\{\hat{r}_i\}_{i=1}^{M}, \{x_i\}_{i=1}^{M})$

for $t \geq M$ **do**

- ***Outer Controller***: Refine reference based on approximated model:

$$\hat{r}_{t+1} = \hat{A}_t \hat{r}_t + \hat{B}_t r_{t+1} \ .$$

- ***Plant***: Propagate the system with $\hat{r}_{t+1}$:

$$x_{t+1} = g(\hat{r}_{t+1}, w_t) \ ,$$

  where g is the evolving law of real plant and w_t the uncertainty.

- ***Model***: Check rank of measurement matrices.

  if *full column rank* **then**

  $(\hat{A}_t, \hat{B}_t) = DMDc(\{\hat{r}_i\}^{t+1}, \{x_i\}^{t+1})$

  else $(\hat{A}_t, \hat{B}_t) = (\hat{A}_{t-1}, \hat{B}_{t-1})$

$t \leftarrow t + 1$

5.2.3 Simulation in a Planar Quadrotor

We demonstrate the utility and evaluate the performance of DHC in simulating a planar quadrotor with different noise intensities in this part. While quadrotor dynamics is well understood, there is a certain limit on the degree of variations in parameters like mass and moment of inertia that model-based controllers can handle. Such variations make the utilized model inaccurate, which in turn may render the controller unstable. Online adjustment can help quantify model errors and inform the controller so as to maintain stable operation.

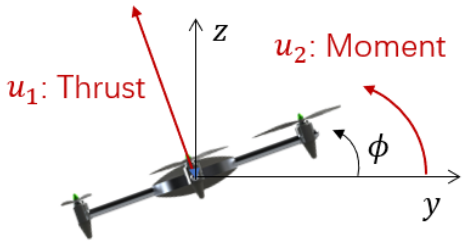

Figure 5.9: Planar quadrotor.

System Description:

The planar quadrotor depicted in Fig. 5.9 can be modeled as

$$\begin{cases} m\ddot{y} = -u_1 \sin\phi \\ m\ddot{z} = u_1 \cos\phi - mg \\ I_{xx}\ddot{\phi} = u_2 \end{cases} .$$

Linearization around hover configurations yields

$$\dot{\xi} = \begin{bmatrix} & \mathbf{0}_{3\times3} & & \mathbf{I}_{3\times3} \\ 0 & 0 & -g & \\ 0 & 0 & 0 & \mathbf{0}_{3\times3} \\ 0 & 0 & 0 & \end{bmatrix} \xi + \begin{bmatrix} & \mathbf{0}_{4\times2} \\ 1/m & 0 \\ 0 & 1/I_{xx} \end{bmatrix} u \ , \tag{5.2}$$

where $\xi = [y, z, \phi, \dot{y}, \dot{z}, \dot{\phi}]$ and $u = [u_1 - mg, u_2]$. In all simulations, nominal parameter values match those of the physical Crazyflie robot used later for experiments, that is, $m = 0.03kg$, $I_{xx} = 1.43 \times 10^{-5} kgm^2$, and $g = 9.8m/s^2$.

Controllers Design:

Besides our proposed DHC structure, we design and compare against a State Feedback Controller (SFC), a linear Model Predictive Controller (MPC), and a Model Reference Adaptive Controller (MRAC).

- State Feedback Controller (SFC): With the closed-loop system poles set at [0.9;0.8;-0.9;-0.8;0.95;-0.95], we get

$$K_p = \begin{bmatrix} -0.0001 & -0.0243 & -0.0008 & 0.0012 & 0 & 0.0149 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & 0 & -0.0000 \end{bmatrix}.$$

- Model Predictive Control (MPC): A linear MPC is designed directly with model (5.2) at hand. The prediction and control horizon is set to be 10 and 2, and the pitch angle ϕ is constrained in the interval $[-\pi/2, \pi/2]$.

- Model Reference Adaptive Control (MRAC): A parameter adaptation rule is used together with the previously described MPC to adjust the controller parameter so

that the output of the plant tracks the output of the reference model having the same reference input. The adaptation rule is designed based on [93] as

$$\frac{dm}{dt} = a_1(y_d - y) + b_1(z_d - z)$$

$$\frac{dI_{xx}}{dt} = a_2(y_d - y) + b_2(z_d - z)$$

where the weights were selected empirically to $a_1 = b_1 = 0.1$ and $a_2 = b_2 = 10^{-4}$. To tune the weights, we first chose their order to be consistent with the physical quantity they relate to. Then, we solved a sequential optimization problem to identify those parameter values that maximize MRAC's performance.

- DHC: Our method also uses the linear MPC as the underlying controller. Then, we follow the steps in Algorithm 3 to simulate the system.

Simulation

The quadrotor's task is to reach position $[3; 5]$ m from initial condition $[0; 0]$ m. We analyze the performance of each controller in two cases: when parameters are uncertain with low noise intensity (c1) where $\sigma_m = 0.2m$ and $\sigma_{I_{xx}} = 0.2I_{xx}$, and with high noise intensity (c2) where $\sigma_m = 0.6m$ and $\sigma_{I_{xx}} = 0.6I_{xx}$.

Perturbations in model parameters are generated at the initialization of each simulated trial by sampling from normal distributions. Distribution means are the nominal values for m and I_{xx}. Distribution variances are selected as above. Once the perturbation is sampled, it remains constant for the duration of the trial. For every different condition, we perform 10 trials.

Table 5.4: Performance of different controllers in linear planar quadrotor model

Case	Controller	Settling time T_s (s)	Total control effort to stabilize $\Sigma\|u\|$	Steady state error $\|x_{desired} - x_{steady}\|$	Overshoot $\frac{\|x_{max} - x_{steady}\|}{\|x_{steady}\|}$
c1	SFC	1.95	1.2453	[0;0]	2.5828
	MPC	0.04	0.6020	[0.6846;0.7740]	0
	MRAC	0.72	11.1852	[0.2986;0.7038]	1.4575
	DHC	0.38	4.0847	[0;0.0011]	0
c2	SFC	Unstable	\	\	\
	MPC	0.04	0.6032	[1.6951;1.1906]	0
	MRAC	0.92	11.0816	[1.0351;1.0357]	0.9718
	DHC	0.38	3.6052	[0;0.0085]	0

Results and Discussion

Simulation results suggest that all controllers can stabilize the system in low noise intensity (see Table 5.4). However, as noise intensity increases SFC fails. Linear MPC converges faster than all other methods but produces an increasingly large steady-state error. MRAC can reduce such steady-state errors, at the expense of longer settling time (within 2%) and larger control effort, and could possibly cause a large overshoot as in SFC. Despite its efficiency shown here, tuning MRAC can be tedious, and its performance relies a lot on tuned weights. On the other hand, our proposed DHC structure keeps updating the model that implicitly includes noise. As such, it can quickly extract refined dynamics from uncertain data to maintain system stability faster, with less control energy and smaller steady-state error. Further, as the disturbance gets embedded in the learned model, the performance of the controller does not deteriorate significantly as noise intensity increases.

5.2.4 Controlling a Crazyflie Quadrotor Under Ground Effect

We evaluate experimentally our DHC structure on a Crazyflie quadrotor operating under the influence of ground effect over sustained periods of time. The ground effect manifests itself as an increase in the generated rotor thrust given constant input rotor power [94, 95] when operating close to the ground or over other surfaces. When operating in the ground effect, the dynamics of the robot-environment interaction change to a degree that an underlying model tuned for mid-air operation may no longer be valid. In turn, model mismatches caused by varying robot-environment interaction (i.e. a form of uncertainty) degrade the robot's performance, both in terms of stability and control.

As we show below, controllers unaware of the ground effect will tend to raise the height of the robot when the latter fly sufficiently close over an obstacle. This behavior may lead to crashes when operating in confined and cluttered environments. Wrapping DHC around such a controller can help the robot adjust the impact of the ground effect without any specific models for it.

Controller Design

We follow the controller structure shown in Fig. 5.8 (as in simulation). We test using two distinct lower-level baseline controllers: a nonlinear geometric controller [96], and a PID controller. Controller gains are those pre-tuned by the manufacturer. Hence, note that the controllers have been tuned for operation in mid-air, not in near-ground operation. The controllers are different from those in the simulation since 1) they are often used in practice, and 2) their underlying performance in simulation would depend on the quality of

tuning (especially for PID) and would thus not add significant value beyond the controllers already tested in simulation.

We evaluate the performance of both low-level controllers with and without DHC wrapped around them. The task is to fly at a constant height between an initial and a goal position (Fig. 5.10). During part of the trajectory, the robot flies over an obstacle which creates the sustained ground effect and influences robot behavior. Performance is quantified in terms of deviating from the desired height due to the ground effect.

Figure 5.10: (Left) Experimental setup, and (Right) the Crazyflie. Sample experiments can be viewed at `https://youtu.be/OznDCskVnJU`.

Experiment

The robot is commanded to follow 2.4 m straight-line constant-height trajectories. The height of the obstacle is $h_o = 0.290$ m. The robot propeller radius is $r = 0.023$ m. We collect data at three distinct heights $h \in \{h_o + r, h_o + 2r, h_o + 3r\}$, and while flying at six distinct forward speeds $v \in \{0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$ m/s. Experiments are run with the four aforementioned control structures. This leads to a total of 72 distinct case studies. For each case study, we collect data from 10 repeated trials. To minimize depleting battery effects, a fully charged battery is used at the beginning of each 10-trial experimental session. Position data are collected via motion capture (VICON).

Results and Discussion

Experimental results (Figs. 5.11 and 5.12) reveal that wrapping the proposed hierarchical structure (red curves in figures) around a low-level controller can help keep the quadrotor closer to the desired height when compared to the standalone original controller (blue curves), on average. This indicates that incorporating model identification and controller adaptation modules through DHC helps the overall control structure adjust better and faster to uncertain perturbations as, in this case, ground effects.

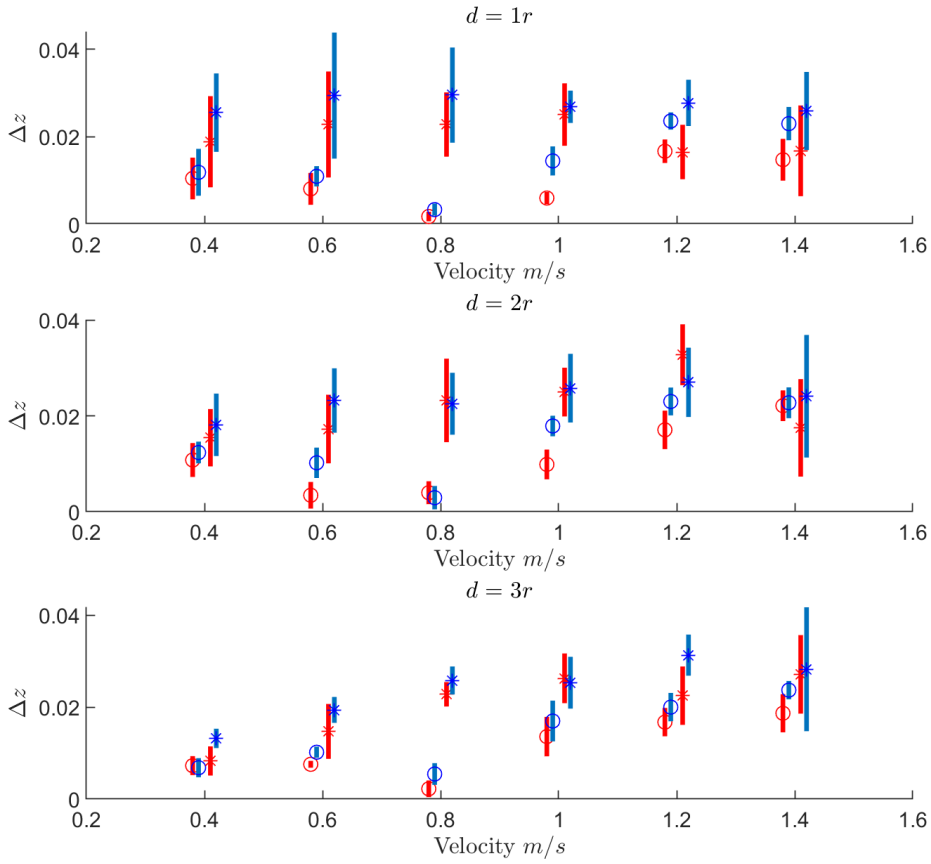

Figure 5.11: 1 σ error plots for the standalone geometric controller (blue circle) and PID controller (blue star), and with wrapped DHC (red circle and star).

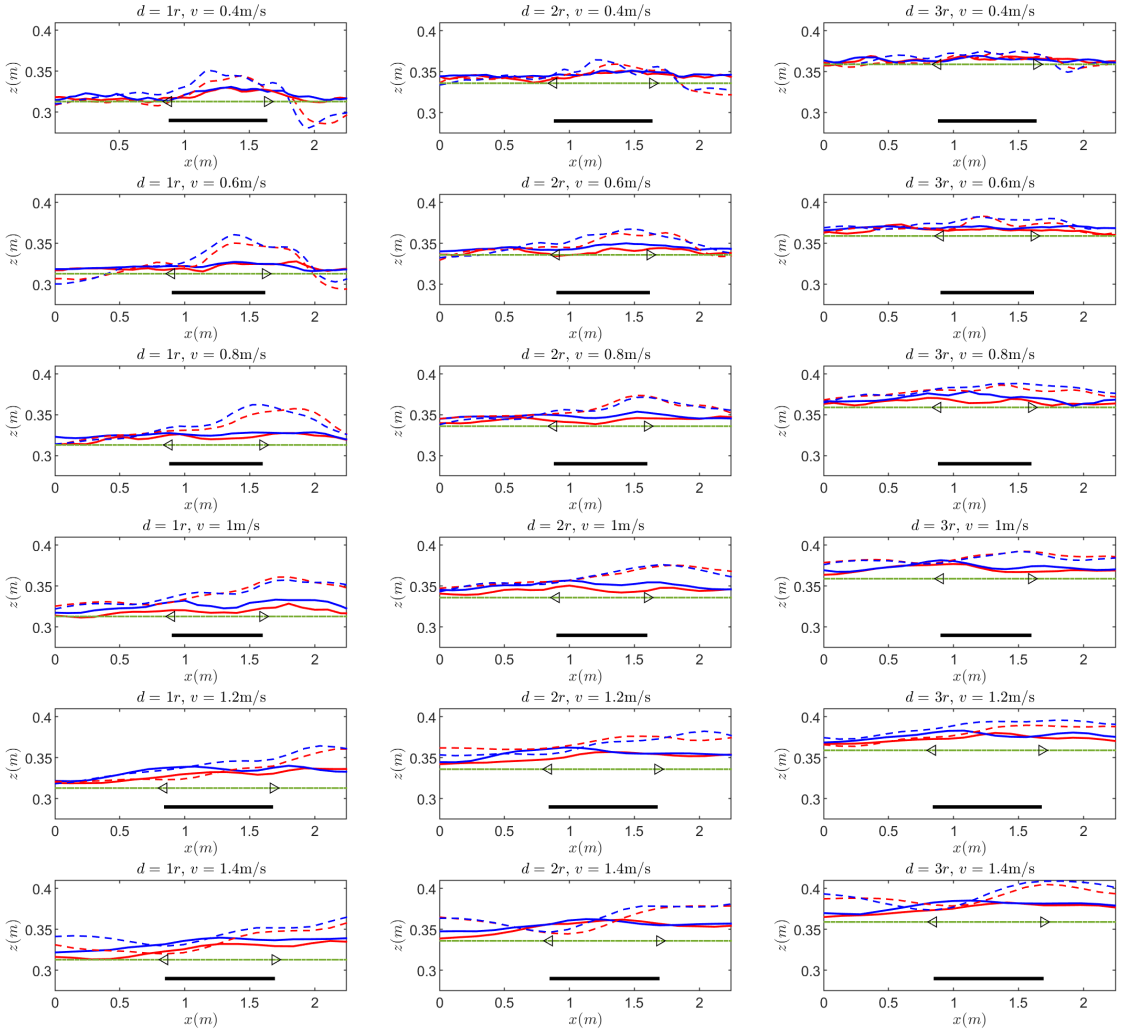

Figure 5.12: Trajectories of the Crazyflie tasked to fly at desired heights, d, over the obstacle (green dashed-dotted) and various forward speeds, v, under ground effect with standalone geometric controller (blue solid) and PID controller (blue dashed), and with wrapped DHC (red dashed and solid). Obstacle locations (black) are shown for clarity.

Furthermore, model/controller adaptation appears to perform better at low distances from the ground and as the forward velocity increases. This observation is in line with aerodynamics suggesting that 1) ground effect starts diminishing when the flying height above the obstacle increases to $d > 3r$, and 2) as forward speed increases, both the ground

effect and drag affect rotorcraft flight in still not well-understood ways. Hence, through this experiment, we show how DHC can be used to render a controller adaptive to environment changes that excite unmodeled dynamics.

The hierarchical structure may still remain bound to the low-level controller's possible limitations, as shown in the simulation. In the experiments, this is observed as forward speed increases. After clearing out the obstacle, the controller's apparent steady-state error seems to increase (c.f. bottom two rows of Fig. 5.12) at higher speeds. While this may be an artifact of the limited experimental volume, the settling time of the controller with and without DHC still increases. Future work will investigate the interplay between the lower-level controller and DHC, and consider different hierarchical adaptive control structures to change an underlying controller's behavior more drastically when needed.

5.3 Summary

In this section, we present two ways to implement the Koopman-learned model in control design. A direct model-based architecture illustrates the most straightforward implementation and is evaluated in soft robotic platforms. Then, a hierarchical controller with the Koopman operator is presented to deal with the uncertainty in the environment when a pre-defined lower-level controller exists. Both of them show the benefits of utilizing the Koopman operator theory in modeling and control of robots.

Soft multi-fingered grippers have been gaining momentum across applications. However, because of their infinite-dimensionality and non-linearities, necessary modeling and control algorithms are difficult to derive. Our proposed Koopman-based method can

achieve solid prediction accuracy while training significantly faster compared to related data-driven methods currently in-use. This method can be directly embedded into model-based online control of robotics systems, without requiring explicit knowledge of system dynamics except for their configuration space or workspace. Results by testing with two soft 3-fingered robotic grippers attest that our method can grasp a variety of objects of different shapes and/or weights without any a-priori knowledge. This work can be beneficial in applications like smart manufacturing and precision agriculture.

Then, we introduced a Data-driven Hierarchical Control (DHC) structure with two key functions. 1) DHC utilizes the online data-driven approach to learn a model of input reference and true outputs. 2) DHC utilizes the derived model to keep refining a reference signal given to an underlying low-level controller to ensure system performance in the presence of system and/or system-environment uncertainty. We showed that DHC retains the stability properties of the underlying lower-level controller. The utility and performance of the proposed approach are tested in simulation and experimentally with a quadrotor. The test cases were designed to excite specific types of uncertainty that are common in practice: 1) constant model parameter errors in simulation; and 2) variable uncertain robot-environment interaction experimentally through operation under ground effect. Results suggest that DHC can successfully be wrapped around existing controllers in practice, to improve their performance by discovering and harnessing unmodeled dynamics during deployment.

Chapter 6

Conclusions and Future Work

In this dissertation, we presented our theoretical and algorithmic contributions to employing Koopman theory across robotic systems. The contributions span over four key phases: noisy measurements analysis in the "Data Collection" process, lifting functions construction in the "Model Extraction" phase, new algorithm of a model-based controller as well as a hierarchical structure in the "Controller Design" phase, and implementations to different robotic platforms including aerial robots, ground robots, manipulators as well as soft robots, in the "Robotic Application" phase.

First, we develop approaches to quantify the prediction error because of noisy data when approximating a model for control of a data-driven system via the Koopman operator. The safety bound is analyzed for DMD and we extend the work to explicit derivation of prediction error for EDMD. The proposed formulation relies on studying the sensitivity of components of the Koopman operator to noisy data. Further, we propose an algorithm for an enhanced robot control structure that endows robustness to the data-driven

system at hand. The proposed algorithm is designed to work in unison with the existing Koopman-based data-driven robot control architecture to add robustness without replacing parts of the underlying architecture. We address practical aspects of how to deploy the proposed algorithm across nonlinear dynamical systems and mobile robots and show how to make (parts of) the algorithm run online to enable real-time implementation. We show evidence of generalizability via a parametric simulation study using a Van der Pol oscillator and experimentation in Gazebo with a non-holonomic wheeled robot (ROSbot2.0) under distinct noise levels.

Second, we propose a general and analytical methodology to formalize the construction of lifting functions based on system characteristic properties. We show how fundamental topological spaces and Cartesian products thereof can be mapped to a basis of Hermite polynomials and Kronecker products thereof which serve as the dictionary of lifting functions. We further show that the resulting dictionary is complete and leads to an estimated Koopman operator with provable guarantees of convergence to the true one, in the limit and provided that the observables are weighted bounded. At the same time, the set is simple to implement. We evaluate the efficacy of our proposed analytical method using a series of both simulated and hardware experiments. We consider a differential drive robot in both simulation and physical experimentation, and a rigid robotic arm comprising two revolute joints in simulation. In these cases, we use the configuration space of the robots. The method can also apply to soft robots (whereby their configuration space is ill-defined), by considering their workspace instead. Physical experimentation using a soft robotic leg that can bend and extend confirms that our approach can apply uniformly across rigid and

soft robots, and further demonstrates its practical utility. Results obtained by our algorithm are also compared against other nonlinear dynamics identification approaches in terms of prediction accuracy and training time, to demonstrate our method's efficacy.

Third, we present an online modeling and control approach for soft multi-fingered robotic grippers based on Koopman operator theory. Dictionaries of lifting functions are designed based on the workspace of the soft gripper. Then, the Koopman operator theory is used to learn and update the model at every time step in real-time. Updated models, in turn, provide the dynamics constraints within an MPC structure to help achieve desired control objectives. We evaluate and compare via physical experiments the prediction accuracy of different modeling methods among different datasets using two soft grippers. The proposed online modeling and control algorithm is implemented in the grippers to grasp different objects of varying shape and weight, without any explicit information provided to the algorithm a-priori.

Finally, fueled by the potential of hierarchical methods and dimensionality-reduction approaches, we illustrate a new structure to handle uncertainty. The approach hinges on DMD with Control to learn a higher-level controller and then generate a refined reference to improve an existing lower-level controller's performance. Our approach can be particularly appropriate in practice when a low-level, high-rate, pre-tuned controller is already in place, and a second higher-level controller is wrapped around the lower-level one to allow for the system to operate under uncertainty. We evaluate and validate the methodology using aerial robots both in simulation and experimentally that are able to learn how to harness uncertain aerodynamics, such as ground effect.

The works described in the dissertation build the foundation for implementing the Koopman operator theory into physical robotic platforms. In terms of future work, other than the extension of current results, e.g., investigate the trade-offs of including higher-order Hermite polynomial terms for prediction in higher-dimensional robotics problems when designing the lifting functions; performing stability analysis of the proposed method in the nonlinear system; and extend the implementation to multi-agent systems, we want to further research the way to improve the modeling of physical complicated robots and decrease the Sim-to-Real gap in the controller applications via the Koopman operator theory.

As robots tend to venture more and more outside of the lab, encounter sophisticated environments and are required to complete more complicated tasks, we expect the modeling of robots and their environments (or human-in-the-loop), to be accurate enough and the resulting controllers are powerful and robust. A large number of new nonlinear methods for modeling and controlling complex robots have been developed, such as reinforcement learning-based architectures and model-based intelligent controllers. However, the application of control methods to real robots is still limited to traditional control structures, which have limited adaptability in new or uncertain environments. The main challenges in applying these new control methods to real robots include the following: first, due to the presence of uncertainty in the interaction between robots and their environments, models and controllers derived from first principles and physical formulas may not be precise and universal enough in practical applications. Second, most neural network-based data-driven methods require a large amount of data to be learned during the "exploration" stage. However, due to safety and efficiency considerations, "exploration" is often only possible in

a simulated environment, making it urgent to reduce the effect of the difference between the simulated environment and the real robot, i.e. the Sim-to-Real gap. To address these issues, we propose to use the data-driven modeling method - Koopman operator theory, which considers the physical structure, and only requires limited measurement time and data, to extract models, thereby reducing the impact of the Sim-to-Real Gap on existing algorithms and improving their performance on real robots. Specifically, future work consists of the following three main parts.

The first part is about considering safety and robustness during the "exploration" and practical application stages. Unlike many other data-driven methods that have a black-box estimation issue, the data-driven modeling methods based on Koopman theory can obtain explicit model expressions. Therefore, we can analyze the properties of the specific models, e.g., controllability, observability, and safety guarantees. For example, we can design a Control Barrier Function (CBF) based on this model to ensure that the robot always stays in a safe space during the exploration process. Additionally, in the practical application of the algorithm, we can limit the robot's input based on this model to make it act within a safe range in real-time with online learning, thereby reducing potential harm to the robot and its surroundings.

The second part is applicable after we completed training in the simulation environment and obtained initial algorithms that are going to be applied to actual robots. By defining the 'state' and 'input' quantities properly in the formulation of modeling with Koopman, we can use the Koopman theory to "measure" the Sim-to-Real gap with limited data during the physical application. This "measurement" can then be used as a sup-

plementary or guiding quantity to be added to current algorithms' online updates, thereby accelerating the adaptation process of the algorithm obtained in the simulation environment to the practical application scenario and the achievement of the desired goal.

With the aforementioned two parts, in the practical application of model-based controllers in complex robots, the Koopman operator can be used online with limited data to estimate the evolution model of the environment or uncertainty. By incorporating the uncertainty in the robot-environment, robot-robot, and human-robot interaction processes mentioned earlier into the existing model, we can design an improved and robust controller. Finally, we aim to apply this complete improvement framework to complex nonlinear robot platforms that are highly susceptible to environmental influences, such as legged robots or soft robots. Besides, in multi-robot systems, we need to consider not only the structure of different machines themselves and the impact of the environment but also the interaction between robots within the system. Therefore, the online estimation based on current data updates can help us better analyze and control multi-robot platforms. We also hope to further promote the application of this Koopman-based framework to multi-robot platforms.

Bibliography

[1] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35:51–76, 2013.

[2] Xinyue Kan, Hanzhe Teng, and Konstantinos Karydis. Online exploration and coverage planning in unknown obstacle-cluttered environments. *IEEE Robotics and Automation Letters*, 5(4):5969–5976, 2020.

[3] Zhouyu Lu, Zhichao Liu, Merrick Campbell, and Konstantinos Karydis. Online search-based collision-inclusive motion planning and control for impact-resilient mobile robots. *IEEE Transactions on Robotics*, 2022.

[4] Alexander Shkolnik, Michael Levashov, Ian R Manchester, and Russ Tedrake. Bounding on rough terrain with the little dog robot. *The International Journal of Robotics Research*, 30(2):192–215, 2011.

[5] Konstantinos Karydis, Ioannis Poulakakis, Jianxin Sun, and Herbert G Tanner. Probabilistically valid stochastic extensions of deterministic models for systems with uncertainty. *The International Journal of Robotics Research*, 34(10):1278–1295, 2015.

[6] Sam Zarovy, Mark Costello, Ankur Mehta, Greg Gremillion, Derek Miller, Badri Ranganathan, J Sean Humbert, and Paul Samuel. Experimental study of gust effects on micro air vehicles. In *AIAA Atmospheric Flight Mechanics Conference*, page 7818, 2010.

[7] Arvind A Pereira, Jonathan Binney, Geoffrey A Hollinger, and Gaurav S Sukhatme. Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, 30(5):741–762, 2013.

[8] Josie Hughes, Utku Culha, Fabio Giardina, Fabian Guenther, Andre Rosendo, and Fumiya Iida. Soft manipulators and grippers: a review. *Frontiers in Robotics and AI*, 3:69, 2016.

[9] Justin Carpentier and Pierre-Brice Wieber. Recent progress in legged robots locomotion control. *Current Robotics Reports*, 2(3):231–238, 2021.

[10] Dario Piga, Marco Forgione, Simone Formentin, and Alberto Bemporad. Performance-oriented model learning for data-driven mpc design. *control systems letters*, 3(3):577–582, 2019.

[11] Karime Pereida and Angela P Schoellig. Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 7831–7837. IEEE, 2018.

[12] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. *CoRR*, 2018.

[13] Christopher D McKinnon and Angela P Schoellig. Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2977–2984, 2018.

[14] Tong Wang, Huijun Gao, and Jianbin Qiu. A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. *IEEE Transactions on Neural Networks and Learning Systems*, 27(2):416–425, 2015.

[15] Herwin Suprijono and Benyamin Kusumoputro. Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, pages 99–102, 2017.

[16] Scott A Nivison and Pramod P Khargonekar. Development of a robust deep recurrent neural network controller for flight applications. In *American Control Conf. (ACC)*, pages 5336–5342. IEEE, 2017.

[17] Bara J Emran and Homayoun Najjaran. Adaptive neural network control of quadrotor system under the presence of actuator constraints. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2619–2624, 2017.

[18] Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. Learning quadrotor dynamics using neural network for flight control. In *IEEE Conf. on Decision and Control (CDC)*, pages 4653–4660, 2016.

[19] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Data-driven approximations of dynamical systems operators for control. *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*, pages 197–234, 2020.

[20] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817, 2000.

[21] Konstantinos Karydis and M Ani Hsieh. Uncertainty quantification for small robots using principal orthogonal decomposition. In *International Symposium on Experimental Robotics*, pages 33–42. Springer, 2016.

[22] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: theory and applications. *"Journal of Computational Dynamics"*, 2014.

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

[24] Stefan Klus, Feliks Nüske, Sebastian Peitz, Jan-Hendrik Niemann, Cecilia Clementi, and Christof Schütte. Data-driven approximation of the koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[25] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2(3):035023, 2021.

[26] Carl Folkestad, Daniel Pastor, Igor Mezic, Ryan Mohr, Maria Fonoberova, and Joel Burdick. Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control. In *american control conference (acc)*, pages 3906–3913. IEEE, 2020.

[27] Steven L Brunton, Bingni W Brunton, Joshua L Proctor, and J Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2), 2016.

[28] Samuel E Otto and Clarence W Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, 2021.

[29] Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[30] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *SIAM Review*, 64(2), 2022.

[31] Milan Korda and Igor Mezić. On convergence of extended dynamic mode decomposition to the koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

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

[33] Sebastian Peitz and Stefan Klus. Koopman operator-based model reduction for switched-system control of pdes. *Automatica*, 106:184–191, 2019.

[34] Giorgos Mamakoukas, Maria L Castano, Xiaobo Tan, and Todd D Murphey. Derivative-based koopman operators for real-time control of robotic systems. *IEEE Transactions on Robotics*, 2021.

[35] Jie Chen, Yu Dang, and Jianda Han. Offset-free model predictive control of a soft manipulator using the koopman operator. *Mechatronics*, 86:102871, 2022.

[36] Minghao Han, Jacob Euler-Rolle, and Robert K Katzschmann. Desko: Stability-assured robust control with a deep stochastic koopman operator. In *International Conference on Learning Representations (ICLR)*, 2021.

[37] Giorgos Mamakoukas, Ian Abraham, and Todd D Murphey. Learning stable models for prediction and control. 2020.

[38] Ye Wang, Yujia Yang, Ye Pu, and Chris Manzie. Data-driven predictive tracking control based on koopman operators. *arXiv preprint arXiv:2208.12000*, 2022.

[39] Niklas Kochdumper and Stanley Bak. Conformant synthesis for koopman operator linearized control systems. In *IEEE 61st Conference on Decision and Control (CDC)*, pages 7327–7332. IEEE, 2022.

[40] Giorgos Mamakoukas, Stefano Di Cairano, and Abraham P Vinod. Robust model predictive control with data-driven koopman operators. In *American Control Conference (ACC)*, pages 3885–3892. IEEE, 2022.

[41] Qianxiao Li, Felix Dietrich, Erik M Bollt, and Ioannis G Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

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

[43] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1130–1140, 2017.

[44] Marcos Netto, Yoshihiko Susuki, Venkat Krishnan, and Yingchen Zhang. On analytical construction of observable functions in extended dynamic mode decomposition for nonlinear estimation and prediction. In *American Control Conference (ACC)*, pages 4190–4195. IEEE, 2021.

[45] Erik M Bollt. Geometric considerations of a good dictionary for koopman analysis of dynamical systems: Cardinality,"primary eigenfunction," and efficient representation. *Communications in Nonlinear Science and Numerical Simulation*, 100:105833, 2021.

[46] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-based control using koopman operators. In *Robotics: Science and Systems, RSS*. MIT Press Journals, 2017.

[47] Carl Folkestad and Joel W Burdick. Koopman nmpc: Koopman-based learning and nonlinear model predictive control of control-affine systems. In *International Conference on Robotics and Automation (ICRA)*, pages 7350–7356. IEEE, 2021.

[48] Hassan Arbabi, Milan Korda, and Igor Mezić. A data-driven koopman model predictive control framework for nonlinear partial differential equations. In *IEEE Conference on Decision and Control (CDC)*, pages 6409–6414, 2018.

[49] Tomoharu Iwata and Yoshinobu Kawahara. Controlling nonlinear dynamical systems with linear quadratic regulator-based policy networks in koopman space. In *60th IEEE Conference on Decision and Control (CDC)*, pages 5086–5091, 2021.

[50] Andrew J Gibson, Michael L Calvisi, and Xin C Yee. Koopman linear quadratic regulator using complex eigenfunctions for nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 21(4):2463–2486, 2022.

[51] Bowen Huang, Xu Ma, and Umesh Vaidya. Data-driven nonlinear stabilization using koopman operator. *The Koopman Operator in Systems and Control*, pages 313–334, 2020.

[52] Abhinav Narasingam, Sang Hwan Son, and Joseph Sang-Il Kwon. Data-driven feedback stabilisation of nonlinear systems: Koopman-based model predictive control. *International Journal of Control*, 96(3):770–781, 2023.

[53] Ian Abraham and Todd D Murphey. Active learning of dynamics for data-driven control using koopman operators. *IEEE Transactions on Robotics*, 35(5):1071–1083, 2019.

[54] Carl Folkestad, Daniel Pastor, and Joel W Burdick. Episodic koopman learning of nonlinear robot dynamics with application to fast multirotor landing. In *International Conference on Robotics and Automation (ICRA)*, pages 9216–9222. IEEE, 2020.

[55] Carl Folkestad, Skylar X Wei, and Joel W Burdick. Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking. In *International Conference on Robotics and Automation (ICRA)*, pages 1344–1350. IEEE, 2022.

[56] Chao Ren, Hongjian Jiang, Chunli Li, Weichao Sun, and Shugen Ma. Koopman-operator-based robust data-driven control for wheeled mobile robots. *IEEE/ASME Transactions on Mechatronics*, 2022.

[57] Alexander Krolicki, Dakota Rufino, Andrew Zheng, Sriram SKS Narayanan, Jackson Erb, and Umesh Vaidya. Modeling quadruped leg dynamics on deformable terrains using data-driven koopman operators. *IFAC-PapersOnLine*, 55(37):420–425, 2022.

[58] Lu Shi, Zhichao Liu, and Konstantinos Karydis. Koopman operators for modeling and control of soft robotics. *arXiv preprint arXiv:2301.09708*, 2023.

[59] Daniel Bruder, Xun Fu, R Brent Gillespie, C David Remy, and Ram Vasudevan. Data-driven control of soft robots using koopman operator theory. *IEEE Transactions on Robotics*, 37(3):948–961, 2020.

[60] Daniel Bruder, Xun Fu, R Brent Gillespie, C David Remy, and Ram Vasudevan. Koopman-based control of a soft continuum manipulator under variable loading conditions. *IEEE Robotics and Automation Letters*, 6(4):6852–6859, 2021.

[61] Daniel Bruder, Brent Gillespie, C David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, 2019.

[62] Naoto Komeno, Brendan Michael, Katharina Küchler, Edgar Anarossi, and Takamitsu Matsubara. Deep koopman with control: Spectral analysis of soft robot dynamics. In *IEEE Annual Conference of the Society of Instrument and Control Engineers (SICE)*, pages 333–340, 2022.

[63] Daniel Bruder, C David Remy, and Ram Vasudevan. Nonlinear system identification of soft robot dynamics using koopman operator theory. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6250, 2019.

[64] David A Haggerty, Michael J Banks, Patrick C Curtis, Igor Mezić, and Elliot W Hawkes. Modeling, reduction, and control of a helically actuated inertial soft robotic arm via the koopman operator. *arXiv preprint arXiv:2011.07939*, 2020.

[65] Lu Shi and Konstantinos Karydis. Enhancement for robustness of koopman operator-based data-driven mobile robotic systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2503–2510, 2021.

[66] Lu Shi and Konstantinos Karydis. Acd-edmd: Analytical construction for dictionaries of lifting functions in koopman operator-based nonlinear robotic systems. *IEEE Robotics and Automation Letters*, 7(2):906–913, 2021.

[67] Lu Shi, Caio Mucchiani, and Konstantinos Karydis. Online modeling and control of soft multi-fingered grippers via koopman operator theory. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1946–1952, 2022.

[68] Lu Shi, Hanzhe Teng, Xinyue Kan, and Konstantinos Karydis. A data-driven hierarchical control structure for systems with uncertainty. In *IEEE Conference on Control Technology and Applications (CCTA)*, pages 57–63, 2020.

[69] Matthew O Williams, Maziar S Hemati, Scott TM Dawson, Ioannis G Kevrekidis, and Clarence W Rowley. Extending data-driven koopman analysis to actuated systems. *IFAC-PapersOnLine*, 49(18):704–709, 2016.

[70] 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.

[71] S Klus and Ch Schütte. Towards tensor-based methods for the numerical approximation of the perron-frobenius and koopman operator. *Journal of Computational Dynamics*, 2016.

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

[73] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[74] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[75] T. Crossley and B. Porter. Eigenvalue and eigenvector sensitivities in linear systems theory. *International Journal of Control*, 10(2):163–170, 1969.

[76] Zhongliang Jing, Lingfeng Qiao, Han Pan, Yongsheng Yang, and Wujun Chen. An overview of the configuration and manipulation of soft robotics for on-orbit servicing. *Science China Information Sciences*, 60(5):050201, 2017.

[77] Parul Maheshwari, Gautam Mukhopadhyay, and Siddhartha SenGupta. Properties of tensor hermite polynomials. *arXiv preprint arXiv:1411.7398*, 2014.

[78] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Data-driven approximations of dynamical systems operators for control. In *The Koopman Operator in Systems and Control*, pages 197–234. Springer, 2020.

[79] Enrico Celeghini, Manuel Gadella, and Mariano A del Olmo. Hermite functions and fourier series. *Symmetry*, 13(5):853, 2021.

[80] B. Reed, M.; Simon. Functional analysis. *Academic Press:*, 1972.

[81] Zhichao Liu, Zhouyu Lu, and Konstantinos Karydis. Sorx: A soft pneumatic hexapedal robot to traverse rough, steep, and unstable terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 420–426, 2020.

[82] Zhichao Liu and Konstantinos Karydis. Position control and variable-height trajectory tracking of a soft pneumatic legged robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1708–1709, 2021.

[83] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[84] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[85] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[86] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[87] Robert F Shepherd, Filip Ilievski, Wonjae Choi, Stephen A Morin, Adam A Stokes, Aaron D Mazzeo, Xin Chen, Michael Wang, and George M Whitesides. Multigait soft robot. *Proceedings of the national academy of sciences*, 108(51):20400–20403, 2011.

[88] Majid Mazouchi, Subramanya Nageshrao, and Hamidreza Modares. Finite-time koopman identifier: A unified batch-online learning framework for joint learning of koopman structure and parameters. *arXiv preprint arXiv:2105.05903*, 2021.

[89] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[90] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters (RAL)*, 3(3):1544–1551, 2018.

[91] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research (IJRR)*, 39(1):3–20, 2020.

[92] Julian Zimmer, Tess Hellebrekers, Tamim Asfour, Carmel Majidi, and Oliver Kroemer. Predicting grasp success with a soft sensing skin and shape-memory actuated gripper. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7120–7127, 2019.

[93] R Ibarra, S Florida, W Rodríguez, G Romero, D Lara, and I Pérez. Attitude control of a quadrocopter using adaptive control technique. In *Applied Mechanics and Materials*, volume 598, pages 551–556. Trans Tech Publ, 2014.

[94] Konstantinos Karydis and Vijay Kumar. Energetics in robotic flight at small scales. *Interface focus*, 7(1):20160088, 2017.

[95] Xinyue Kan, Justin Thomas, Hanzhe Teng, Herbert G Tanner, Vijay Kumar, and Konstantinos Karydis. Analysis of ground effect for small-scale uavs in forward flight. *IEEE Robotics and Automation Letters*, 4(4):3860–3867, 2019.

[96] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.