University of California
Santa Barbara

Koopman Representations in Control

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

Doctor of Philosophy
in
Mechanical Engineering

by

Michael James Banks

Committee in charge:

Professor Igor Mezić, Chair
Professor Elliot Hawkes
Professor Bassam Bamieh
Professor Michael Gordon

June 2023

The Dissertation of Michael James Banks is approved.

Professor Elliot Hawkes

Professor Bassam Bamieh

Professor Michael Gordon

Professor Igor Mezić, Committee Chair

June 2023

Koopman Representations in Control

by

Michael James Banks

To my family

Acknowledgements

I would like to thank my advisor Dr. Igor Mezić. I have come to greatly respect and admire you for your kindness and mentorship over the past few years. I would also like to thank Dr. Elliot Hawkes, Dr. Ervin Kamenar, and David Haggerty for their extensive support through the soft robot project. I would also like to thank my research group Gowtham San Seenivasaharagavan, Mathias Wanner, and Alan Cao. I've learned so much from all of them. Lastly, I would like to thank my parents for their endless support through all of this. Without their encouragement, I would never have made it this far.

Curriculum Vitæ

Michael James Banks

Education

2023	Ph.D. in Mechanical Engineering (Expected), University of California, Santa Barbara.
2019	M.S. in Aerospace Engineering, University of Illinois at Urbana-Champaign.
2016	B.S. in Aerospace Engineering, University of Illinois at Urbana-Champaign.

Publications

- D.A. Haggerty, M.J. Banks, E. Kamenar, A.B. Cao, P.C. Curtis, I. Mezić, and E.W. Hawkes, *Control of Inertial Soft Robots*, *Science Robotics*, Under Review
- Paper in preparation with A.B. Cao and I. Mezić
  "Learning and Control of Deep-Neural Koopman Eigenfunctions"
- Paper in preparation with I. Mezić
  "Input-Parameterized Koopman Eigenfunctions"
- Paper in preparation with Y. Wang, T. Hikihara, and I. Mezić
  "Prediction and Control of Swarm Formation in Unknown Nonlinear Potential Using Dynamic Mode Decomposition"
- S. Bidadi, M.J. Banks, and D.J. Bodony, *Advances in adjoint-based methods: application towards multiphase systems*, *ILASS Paper*, Presented at the ILASS-Americas 29th Annual Conference on Liquid Atomization and Spray Systems Atlanta, GA, May 2017
- M.J. Banks, P. Sashittal, and D.J. Bodony, *Towards Data-driven Control of Multiphase Flows*, *ICLASS Paper*, Presented at the 14th International Conference on Liquid Atomization and Spray Systems, Chicago, IL, July, 2018

Conference Presentations

- SIAM DS 2023 Mini-Symposium
  Presented "Input-Parameterized Koopman Eigenfunctions"
  Co-authored "Koopman-Based Modeling and Control of Nonlinear Soft Robots"
  Co-authored "Prediction and Control of Swarm Formation in Unknown Nonlinear Potential Using Dynamic Mode Decomposition"
- SIAM DS 2021
  "Koopman Mode Analysis and Control of a Soft Robotic Arm"

- ICLASS 2019
  "Control-informed Dynamic Mode Decomposition applied to two-dimensional liquid jet breakup"
- ICLASS 2018
  "Towards Data-driven Control of Multiphase Flows"
- APS DFD 2018
  "Control-Informed Dynamic Mode Decomposition"
- APS DFD 2017
  "Data-based adjoint and H2 optimal control of the Ginzburg-Landau equation"

Abstract

Koopman Representations in Control

by

Michael James Banks

The Koopman operator describes the time-evolution of scalar-valued functions under the action of a dynamical system. These functions are called observables, and their evolution is always linear, even if the underlying dynamical system is nonlinear. The linearity of the Koopman operator framework is attractive to both dynamical systems theorists who study the spectral properties of these operators as well as to control theorists who leverage linearity to simplify control design. Recently, the theory of Koopman *representations* has emerged, with researchers gradually exploring the benefits of alternate, potentially nonlinear ways of representing these systems. In this thesis, we explore three distinct ways of representing the Koopman operator and explore their application to control design.

In the first part of this dissertation, we develop the mathematical underpinning of Koopman representation theory. The evolution from the state-space representation of a dynamical system to the Koopman operator is described, and its spectral content is explored. Next, nonlinear representations of the Koopman operator and its extension to systems with input is described. Finally, we introduce some of the numerical approximation schemes for the representations that are used in this paper. This chapter is meant to give the reader the mathematical background necessary to appreciate the results presented in the remaining chapters.

In the second part of this dissertation, we demonstrate a linear representation of the Koopman operator which fully leverages spectral objects such as eigenfunctions and

eigenvalues. The eigenfunctions are special observables which evolve under the action of the Koopman operator via multiplication by a complex scalar, the eigenvalues. This is analogous to the eigenvectors and eigenvalues of a linear transformation. A collection of eigenfunctions forms a finite-dimensional, linear representation of a dynamical system, and their evolution spans a Koopman-invariant subspace. Finding this finite-dimensional representation allows for the application of well-developed linear systems methodologies to nonlinear systems such as spectral analysis and linear optimal control methods. In this work, we introduce a deep learning architecture that learns the Koopman eigenfunctions of a dynamical system from data and constructs the resulting finite-dimensional, linear representation of the Koopman operator. In numerical examples, the eigenfunctions learned using this framework exhibit a predictive performance superior to popular fixed-basis methods such as Extended Dynamic Mode Decomposition (EDMD). Finally, we extend the architecture to *controlled* dynamical systems by simultaneously learning the eigenfunctions of the natural dynamics with special system-decoupling observables on the inputs. Numerical examples show that the linear predictors obtained in this way can be readily used to design controllers that act directly on the Koopman modes of the system.

In the third part of this dissertation, we introduce our first use of the *static Koopman operator* in control. In our application, this is a linear operator which maps the space of functions of static poses of a soft robotic arm to the space of functions of the pressures in the arm's actuating muscles. We use static Koopman operator as a pregain term in our optimal control implementation alongside a traditional *dynamic Koopman operator*. Using both Koopman representations, we advance the modeling and control of soft robots into the inertial, nonlinear regime. We control motions of a soft, continuum arm with velocities 10x larger and accelerations 40x larger than those of previous work, and do so for high-deflection shapes with over 110 degrees of curvature. This work advances rapid modeling and control for soft robots from the realm of quasi-static to inertial, laying the

groundwork for the next generation of compliant and highly dynamic robots.

Lastly, in the fourth chapter, we introduce a *nonlinear Koopman representation* which leverages so-called input-parameterized Koopman eigenfunctions. In the control of systems with multiple fixed points, it is typical to use piecewise control methods and local Koopman models. In contrast, our input-parameterized eigenfunction representation is accurate globally and enables a finite dimensional model which can handle these control problems without ad-hoc piecewise methods. We illustrate this on the control between the basins of attraction of the Duffing oscillator with dissipation.

Contents

Chapter 1

Koopman Representation Theory

In the traditional theory of geometric dynamics, a dynamical system is viewed as a rule which describes the time-evolution of a vector of states on a manifold. This representation of dynamical systems has seen extensive application to the study of the evolution of natural processes ranging as far as the orbital motion of planets to the population dynamics of living things. Although this methodology has been successful in many applications, the nonlinearity inherent in the dynamics of many systems of interest poses difficult challenges to this type of representation. Furthermore, the dynamics are often unknown in principle, with only experimental data available.

Originally attributed to Koopman, Carleman, and Von Neumann, Koopman Operator Theory is an alternative representation of a dynamical system which gives the evolution of observables on an infinite dimensional function space [1, 2]. Instead of acting directly on the states of the system, the Koopman operator acts on observables which are scalar-valued functions of the state space. This operator is always linear, even if the state space representation of the system is nonlinear. When the Koopman operator can be approximated (often from data), or better yet, a finite dimensional *representation* is found, then it can be used to study the underlying physics of the system. For example, the Koopman modes were used to study fluid flows in [3, 4]. In particular, the stability properties of a nonlinear dynamical system can be studied using methods analogous to the analysis of the spectral properties of linear dynamical systems [5].

The search for a finite-dimensional representation of the Koopman operator usually leverages the existence of Koopman eigenfunctions. These are special choices of observables which evolve analogously to the eigenvectors of a linear dynamical system in state space [4]. When enough eigenfunctions exist, it is possible to construct a representation of the Koopman operator which is linear and finite dimensional. This greatly simplifies prediction and analysis of the underlying physics of nonlinear systems. Other desirable properties of Koopman representations include faithfulness and efficiency. A faithful rep-

resentation is one which effectively distinguishes points in state-space, and an efficient representation uses no redundant observables [6].

Eigenfunctions can be difficult do derive analytically, so many data-driven approximation methods have been proposed. The most common method of these involve approximating the eigenfunctions as a linear combination of a predefined dictionary of observables. These include Dynamic Mode Decomposition (DMD) [3, 7, 8], extensions of DMD such as EDMD [9] and Hankel DMD [10], as well as Koopman Reduced Order Nonlinear Identification and Control (KRONIC) [11]. Methods which do not require a predefined basis of observables include deep learning [12, 13, 14], Generalized Laplace Analysis (GLA) [15, 16, 17, 18], and the methods in [19].

The Koopman operator framework can be extended to systems with input [20, 21]. The extension of DMD to the estimation of the controlled Koopman operator is called Dynamic Mode Decomposition with control (DMDc). The (bi)-linearity of the resulting Koopman representation allows for the application of well-known optimal control methodologies such as the Linear Quadratic Regulator (LQR) and linear Model Predictive Control (MPC) [22].

1.1 Representations of Dynamical Systems

Consider the space $\mathcal{M}$ with elements $\mathbf{x} \in M$. In general, this can be taken as an abstract measure space. For our purposes, we will assume that $\mathcal{M}$ is an n-dimensional manifold, and we will it "state space" with "states" $\mathbf{x}$. In order to introduce the concept of a dynamical system on $\mathcal{M}$, we also need to introduce the parameter t. When $t \in \mathbb{R}$, the corresponding system is said to be in continuous-time, and $t \in \mathbb{Z}$ corresponds to a discrete-time system.

Definition 1 *If $t \in \mathbb{R}$ and $\mathcal{M}$ is smooth, then the* ***continuous-time state space representation*** *of a dynamical system is given by a Lipschitz continuous mapping $F : \mathcal{M} \to T\mathcal{M}$ such that*

$$\dot{\mathbf{x}} = F(\mathbf{x}) \tag{1.1}$$

where $\dot{\mathbf{x}} = \frac{\partial}{\partial t}\mathbf{x}$ is the time derivative of the state and $T\mathcal{M}$ is the tangent bundle of $\mathcal{M}$.

Definition 2 *If $t \in \mathbb{Z}$, then the* ***discrete-time state space representation*** *of a dynamical system is given by a mapping $T : \mathcal{M} \to \mathcal{M}$ such that*

$$\mathbf{x}^+ = T(\mathbf{x}). \tag{1.2}$$

Here $T : \mathcal{M} \to \mathcal{M}$ is called the state transition mapping and $\mathbf{x}^+$ is the time-shifted state.

With these definitions, the state can be treated as a function of time $\mathbf{x}(t)$.

Definition 3 *Given an initial condition $\mathbf{x}_0 = \mathbf{x}(0)$, the* ***flow*** *$S^t$ of the dynamical system is given by*

$$\mathbf{x}(t) = S^t(\mathbf{x}_0). \tag{1.3}$$

In discrete-time, the flow over a single time step is given by $S^1 = T$. When F and T are nonlinear, their state space representations can be difficult to work with, so we turn to the Koopman operator framework.

Define the set $\mathcal{O}$ of all complex-valued functions $f(\mathbf{x}) : \mathcal{M} \to \mathbb{C}$. We call this the space of observables. If the domain of f has infinitely many points, then the underlying functions space $\mathcal{O}$ is infinite-dimensional. It is usually taken to be a Hilbert space. The Koopman representation of a dynamical system describes the evolution of these observables under the action of the dynamics [1, 2].

Definition 4 *The group of* ***Koopman operators*** U^t*, parameterized by time* $t \in \mathbb{R}^+$*, gives the evolution of observables* $f \in \mathcal{O}$ *under the flow of the dynamical system* S^t *or* T *according to the rule*

$$U^t f = f \circ S^t \tag{1.4}$$

in continuous-time. In discrete-time, this is

$$Uf = f \circ T. \tag{1.5}$$

Even though the underlying state space system is nonlinear, the Koopman operator is linear [1, 4, 16]. The proof of this fact is reproduced below.

Proposition 1 *The Koopman operator* U^t *is linear.*

Proof: We will prove the discrete-time case. For all $f_1, f_2 \in \mathcal{O}$ and $a, b \in \mathbb{C}$, we have

$$\begin{aligned} U(af_1(x) + bf_2(x)) &= (af_1 + bf_2) \circ T(x) \\ &= af_1(T(x)) + bf_2(T(x)) \\ &= aUf_1(x) + bUf_2(x). \end{aligned} \tag{1.6}$$

■

Much of the active research in Koopman operator theory involves looking for ways to represent the Koopman operator as a finite dimensional object, so that it can be stored and manipulated on a computer.

Definition 5 *A* ***finite*** (n)***-dimensional representation*** *of the dynamics* F *on* $\mathcal{O}$ *is defined by the pair* $(\mathbf{f}, \mathcal{F}^t)$*. Here* $\mathbf{f} = (f_1, ..., f_n)$ *is a set of* n *observables and* $\mathcal{F}^t : \mathbb{C}^n \to \mathbb{C}^n$ *is a mapping which satisfies*

$$U^t \mathbf{f} = \mathcal{F}^t \circ \mathbf{f}. \tag{1.7}$$

In order to find a finite-dimensional representation of the Koopman operator, we require a set of observables $\mathbf{f}$ to span a Koopman-invariant subspace of $\mathcal{O}$.

Definition 6 *Define $\mathcal{O}|_{\mathbf{f}}$ to be the subspace of $\mathcal{O}$ generated by linear combinations of $f_1, \ldots, f_n$. $\mathcal{O}|_{\mathbf{f}}$ is a **Koopman-invariant subspace** if*

$$g \in \mathcal{O}|_{\mathbf{f}} \implies U^t g \in \mathcal{O}|_{\mathbf{f}} \, \forall t. \tag{1.8}$$

If the action of the Koopman operator causes an observable of interest to leave $\mathcal{O}|_{\mathbf{f}}$, then this is called **leakage**, and $\mathbf{f}$ is insufficient to build a finite-dimensional representation of U^t.

We often want to choose observables which allow us to extend the linearity of the Koopman operator to $\mathcal{O}|_{\mathbf{f}}$.

Definition 7 *$(\mathbf{f}, \mathcal{F}^t)$ is a **linear representation** if $\mathcal{F}^t : \mathbb{C}^n \to \mathbb{C}^n$ is a linear mapping. This mapping can then be rewritten as $\mathcal{F}^t = e^{At}$ where A is called the **eigenmatrix** of U^t.*

When $\mathbf{f}$ is injective, this embedding is able to resolve all points in state space. A Koopman representation with this property is called faithful.

Definition 8 *A Koopman representation $(\mathbf{f}, \mathcal{F}^t)$ is **faithful** if for all $\mathbf{x}_1, \mathbf{x}_2 \in M$ we have*

$$\mathbf{f}(\mathbf{x}_1) = \mathbf{f}(\mathbf{x}_2) \implies \mathbf{x}_1 = \mathbf{x}_2. \tag{1.9}$$

If a representation is not faithful, it is called **reduced**. Whether or not a given representation is faithful will be a concern when we turn our attention to systems with multiple fixed points in Section 4.2.2.

Our next consideration is whether a Koopman representation has any redundant observables. For this, we turn to the concept of efficiency.

Definition 9 *A Koopman representation* $(\mathbf{f}, \mathcal{F}^t)$ *is* ***efficient*** *if* $\nexists H : \mathbb{C}^{n-1} \to \mathbb{C}$ *and* j *such that*

$$f_j = H(f_1, \ldots, f_{j-1}, f_{j+1}, \ldots, f_n). \tag{1.10}$$

The desire to build a representation of the Koopman operator which is linear, finite-dimensional, faithful, and efficient motivates the study of the spectrum of the Koopman operator.

1.2 Spectrum of the Koopman Operator

The linearity of the Koopman operator gives it a rich spectral structure.

Definition 10 *The* ***spectrum*** *of the Koopman operator* $\sigma(U^t)$ *is the set of scalars* $\lambda \in \mathbb{C}$ *such that* $U^t - \lambda I$ *is not invertible.*

Infinite dimensional operators such as the Koopman operator can have **continuous** and **residual** spectra. These concepts are not explored in this dissertation. Instead, we are concerned with the case where $U^t - \lambda I$ is not injective. In this case, $\sigma(U^t)$ is a so-called **discrete** spectrum. In this case, the scalars λ are called **eigenvalues** because they resemble the eigenvalues of finite-dimensional linear transformations. Koopman eigenvalues are associated with special choices of observables called Koopman eigenfunctions.

Definition 11 *A Koopman* ***eigenfunction*** $\phi(\mathbf{x})$ *with* ***eigenvalue*** $\lambda \in \mathbb{C}$ *is an observable whose Koopman evolution is given by*

$$U^t \phi = e^{\lambda t} \phi \tag{1.11}$$

in continuous-time and

$$U\phi = \lambda\phi \tag{1.12}$$

in discrete-time.

If the initial condition of the system exists in the span of a collection of known eigenfunctions $\mathbf{x}_0 \in \text{span}\{\phi_1, \ldots, \phi_n\}$, then the evolution of the system is fully determined by the corresponding eigenvalues. This means that the span of a set of eigenfunctions defines a Koopman invariant subspace of the dynamics [4]. Due to this, they may be used to build an exact finite-dimensional linear representation of the dynamics, applicable to any initial condition inside the invariant subspace.

Definition 12 *Define the vector of eigenfunctions $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_n]^T$, the diagonal matrix of eigenvalues*

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

*and the mapping $\Lambda^t(\boldsymbol{\phi}) = e^{\Lambda t}\boldsymbol{\phi}$ in continuous-time or $\Lambda(\boldsymbol{\phi}) = \Lambda\boldsymbol{\phi}$ in discrete-time. The pair $(\boldsymbol{\phi}, \Lambda^t)$ (or $(\boldsymbol{\phi}, \Lambda)$) is an **eigenfunction representation** of the Koopman operator. This representation is linear and finite dimensional.*

Methods for computing the eigenfunctions of dynamical systems are described in Sections 1.6.1 and 1.6.2.

1.3 Spectral Expansions

A collection of eigenfunctions $\{\phi_1, \ldots, \phi_n\}$ and eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ is enough to reconstruct the dynamics of a nonlinear system. We do this by employing the so-called

spectral expansion of the Koopman operator. As an example, we will give a brief overview of this spectral expansion applied to a system with a globally stable equilibrium at $\mathbf{x} = \mathbf{0}$ and analytic observables. More detail is given in [23]. Under certain assumptions on the eigenvalues of the gradient of the vector field at $\mathbf{0}$, an analytic vector field implies the existence of analytic eigenfunctions of the Koopman operator. The collection of these eigenfunctions forms the coordinates $\mathbf{z} = [\phi_1, \ldots, \phi_n]^T$. Next, define the Koopman modes

$$\mathbf{v}_{k_1 \ldots k_n} = \frac{1}{k_1! \ldots k_n!} \left. \frac{\partial^{k_1 \cdots k_n} \mathbf{z}^{-1}}{\partial^{k_1} \phi_1 \cdots \partial^{k_n} \phi_n} \right|_0 \tag{1.13}$$

and

$$\mathbf{v}_i = \left. \frac{\partial \mathbf{z}^{-1}}{\partial \phi_i} \right|_0 . \tag{1.14}$$

Then, the evolution of the state observable $\mathbf{f}(\mathbf{x} = \mathbf{x}$ forward N time steps is given by

$$\begin{aligned} U^N \mathbf{x} = &\sum_{i=1}^{n} \phi_i(\mathbf{x}) \mathbf{v}_i \lambda_i^N \\ &+ \sum_{\substack{\{k_1, \ldots, k_n\} \in \mathbb{N}^n \\ k_1 + \cdots + k_n > 1}} \phi_1^{k_1}(\mathbf{x}) \cdots \phi_n^{k_n}(\mathbf{x}) \mathbf{v}_{k_1 \ldots k_n} \left(\lambda_1^{k_1} \cdots \lambda_n^{k_n} \right)^N . \end{aligned} \tag{1.15}$$

Thus the dynamics are fully described by the so-called **spectral triple** $(\lambda_i, \mathbf{v}_i, \phi_i)$ of eigenvalues, Koopman modes, and eigenfunctions. In this context, the functions ϕ_i are often called the principal eigenfunctions. Note that the powers and products of the principal eigenfunctions are eigenfunctions themselves and become important in nonlinear dynamics, but the influence of these modes is often small because of the $\frac{1}{k_1! \ldots k_n!}$ term in Eq. 1.13. These extra eigenfunctions generated by the principal eigenfunctions become important in Section 2.3.1. When the dynamics are linear, the following simplified evolution equation is sufficient

$$U^N \mathbf{x} = \sum_{i=1}^{n} \phi_i(\mathbf{x}) \mathbf{v}_i \lambda_i^N \tag{1.16}$$

1.4 Nonlinear Representations

In some cases, a finite-dimensional, linear representation of U^t is either unknown or doesn't exist, making $\mathcal{F}^t$ necessarily nonlinear. This can occur when the spectrum of the system is necessarily continuous, or in some cases when input is introduced to the system. The problem of finding a suitable representation $(\mathbf{f}, \mathcal{F}^t)$ for general nonlinear maps was introduced in [6] as the "Koopman representation eigenproblem".

Definition 13 *The* ***representation eigenproblem*** *is the problem of finding a finite dimensional pair* $(\mathbf{f}, \mathcal{F}^t)$ *such that*

$$U^t \mathbf{f} = \mathbf{f} \circ S^t = \mathcal{F}^t(\mathbf{f}). \tag{1.17}$$

This is analogous to the search for eigenfunctions and eigenvalues when $\mathcal{F}^t$ is a linear mapping. Because of this, $\mathcal{F}^t$ is called the **eigenmap**. We propose a nonlinear representation of the Koopman operator and apply it to the control of multi basin of attraction systems in Section 4.1.1.

1.5 The Control Koopman Operator

A key motivation for developing a globally linear representations of nonlinear dynamical systems is the application of linear control methods. For some state space $\mathbf{x} \in \mathcal{M}$ and input space $\mathbf{u} \in \mathbb{C}^P$, we now consider a nonlinear discrete-time *controlled* dynamical

system

$$\mathbf{x}^+ = T(\mathbf{x}, \mathbf{u}), \tag{1.18}$$

where $T : \mathcal{M} \times \mathbb{C}^P \to \mathcal{M}$ is the mapping between successive time steps.

The Koopman operator framework can be extended to systems with inputs [24]. In full generality, the Koopman operator for systems with input acts on observables of the form $\bar{h} : M \times \mathcal{U} \to \mathbb{C}$ where $\mathcal{U}$ is the space of all control sequences indexed by time $\bar{\mathbf{u}}(\cdot) : \mathbb{N} \to \mathbb{C}^p$. We introduce the left shift operator $V : \mathcal{U} \to \mathcal{U}$ which simply chooses the next input in a sequence $(V\bar{\mathbf{u}})(k) = \bar{\mathbf{u}}(k+1)$.

Definition 14 *U is called the **control Koopman operator** if*

$$(U\bar{h})(\mathbf{x}, \bar{\mathbf{u}}(\cdot)) \coloneqq \bar{h}(T(\mathbf{x}, \bar{\mathbf{u}}(0)), V\bar{\mathbf{u}}(\cdot)). \tag{1.19}$$

Elements of $\mathcal{U}$ are infinite dimensional, which puts the observables $\bar{h} : M \times \mathcal{U} \to \mathbb{C}$ on an infinite dimensional domain, so they cannot be manipulated on a computer. We introduce the simplifying assumption that knowing only the input at the current time step is enough to predict the future dynamics. We can now define observables of the form $h : M \times \mathbb{C}^p \to \mathbb{C}$. This results in a Koopman operator $\mathcal{K}$ defined by

$$(Uh)(\mathbf{x}, \mathbf{u}) \coloneqq f(T(\mathbf{x}, \mathbf{u}), \mathbf{u}) \tag{1.20}$$

We seek a finite dimensional linear input/output system which approximates the action of U on a finite set of chosen observables. This process is described in Section 1.6.3.

In our applications, we will also make the simplifying assumption that the observables can be decoupled into pure state observables $f : \mathcal{M} \to \mathbb{C}$ and pure input observables

$g : \mathbb{C}^P \to \mathbb{C}$ such that

$$\{h_j(\mathbf{x}, \mathbf{u})\}_{j=1}^{n+n_u} = \{f_j(\mathbf{x})\}_{j=1}^{n} \cup \{g_j(\mathbf{u})\}_{j=n+1}^{n+n_u} \tag{1.21}$$

where n_u is the number of input observables.

1.6 Data-Driven Methods

Computing a finite-dimensional approximation of the resulting infinite-dimensional Koopman operator is essential for manipulation on a computer and is an active area of research. A method for the direct computation of Koopman eigenfunctions using Laplace averages is described in Section 1.6.1.

The most popular data-driven Koopman approximation method is called Extended Dynamic Mode Decomposition (EDMD). This method is described in Section 1.6.2. This method can be extended to systems with input as described in Section 1.6.3. This results in a finite-dimensional representation of the Koopman operator with an additive input term which enables the application of linear optimal control methods in nonlinear dynamical systems.

The Koopman eigenfunctions can also be computed from the EDMD-approximated Koopman operator (see Eq. 1.31). The convergence of the EDMD-approximated eigenfunctions to the true eigenfunctions is described in detail in [25]. The accuracy of these approximations is held back by the need to specify a basis. The search for basis-free methods for computing Koopman eigenfunctions motivates the use of the deep learning methodologies in Chapter 4.

1.6.1 Generalized Laplace Analysis

A common method of computing Koopman eigenfunctions $\phi(\mathbf{x})$ involves computing the Laplace averages $f^*_\lambda(\mathbf{x})$ associated with a continuous observable $f \in C^1$ and a complex eigenvalue λ [15]. Assume the vector field F from Eq. 1.1 is a C^2 function with flow S^t. Suppose this system has a stable fixed point $\mathbf{x}^*$ with basin of attraction $\mathcal{B}(\mathbf{x}^*)$ and a Jacobian matrix $J(\mathbf{x}) = \partial_{\mathbf{x}} F(\mathbf{x})$ with all stable eigenvalues $\mathcal{R}\{\lambda_j\} < 0 \; \forall j$ at $\mathbf{x}^*$. Suppose also that we choose the observable f such that $f(\mathbf{x}^*) = 0$ and $\langle \nabla f(\mathbf{x}^*), \mathbf{v}_1 \rangle \neq 0$ and define λ_1 to be the slowest-decaying eigenvalue ($\mathcal{R}\{\lambda_j\} \leq \mathcal{R}\{\lambda_1\} < 0 \; \forall j \neq 1$).

Definition 15 *With the above conditions met, the* ***Laplace average*** *of the observable f corresponding to the eigenvalue λ_1 is given by the formula*

$$f^*_{\lambda_1}(\mathbf{x}) = \lim_{T \to \infty} \frac{1}{T} \int_0^T (f \circ S^t)(\mathbf{x}) e^{-\lambda_1 t} dt. \tag{1.22}$$

A method for the numerical computation of the Laplace averages is given in [15].

The eigenfunction $\phi(\mathbf{x}) = f^*_{\lambda_1} / \langle \nabla f(\mathbf{x}^*), \mathbf{v}_1 \rangle$ contains information about the so-called isostables and isochrons of the system. In particular, the level sets of $|\phi(\mathbf{x})|$ are the isostables of the system. These give information about the rate of descent of trajectories toward the attractor. All initial conditions on an isostable with $|\phi(\mathbf{x})| = \alpha_1$ will arrive at the isostable $|\phi(\mathbf{x})| = \alpha_2$ in time

$$T = \frac{1}{|\mathcal{R}\{\lambda_1\}|} \ln\left(\frac{\alpha_1}{\alpha_2}\right). \tag{1.23}$$

Similarly, if the system has periodic behavior, then the level sets of the complex argument $\angle\phi(\mathbf{x})$ are the isochrons. These contain information about the phase of the system. In particular, in systems with limit cycles, the points on an isochron all converge to the same limiting trajectory [26, 27]. We use Equation 1.22 to compute eigenfunctions

in Section 4.3.1. Our control designs use the time between isostables given in Equation 1.23.

1.6.2 Extended Dynamic Mode Decomposition - EDMD

First developed in [7], data-driven algorithms such as DMD and its variants are often used to find finite-dimensional representations of the Koopman operator. In particular, EDMD enhances the performance of DMD by enriching the model with a user-specified vector of nonlinear observables $\mathbf{f} : \mathcal{M} \to \mathbb{R}^M$ [25]. The algorithm assumes there exist data vectors in the form of tuples $\{\mathbf{x}_i,\ \mathbf{x}_i^+\},\ i = 1, \ldots, N$ that satisfy $\mathbf{x}_i^+ = T(\mathbf{x}_i)$. The data vectors form the columns of the data matrices

$$X = [\mathbf{x}_1\ ...\ \mathbf{x}_N], \tag{1.24}$$

$$Y = [\mathbf{x}_1^+\ ...\ \mathbf{x}_N^+]. \tag{1.25}$$

It is then assumed that there exists a matrix A such that

$$Y \approx AX. \tag{1.26}$$

One then chooses a vector of observables f and performs the lifting of the data matrices

$$X_{\text{lift}} = [\mathbf{f}(\mathbf{x}_1)\ ...\ \mathbf{f}(\mathbf{x}_N)], \tag{1.27}$$

$$Y_{\text{lift}} = [\mathbf{f}(\mathbf{x}_1^+)\ ...\ \mathbf{f}(\mathbf{x}_N^+)]. \tag{1.28}$$

The DMD algorithm finds the optimal matrix A in a least-squares sense by solving

the following minimization problem

$$\min_A \left\| Y_{\text{lift}} - AX_{\text{lift}} \right\|_F^2 \tag{1.29}$$

where F denotes the Frobenius norm. This has solution

$$A = Y_{\text{lift}}(X_{\text{lift}})^\dagger \tag{1.30}$$

where † is the Moore-Penrose pseudo-inverse.

Approximations of the spectral properties of the system can be deduced from the matrix A. The eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ of A correspond to approximations of the eigenvalues of the Koopman operator and the Koopman modes. The adjoint Koopman modes $\mathbf{w}_i$ can then be used to compute the so-called "numerical eigenfunctions". First, normalize every $\mathbf{w}_i$ such that $\langle \mathbf{v}_i, \mathbf{w}_j \rangle = \delta_{ij}$. The numerical eigenfunctions are then given by the complex inner product

$$\hat{\phi}_i(\mathbf{f}(\mathbf{x})) = \langle \mathbf{f}(\mathbf{x}), \mathbf{w}_i \rangle. \tag{1.31}$$

Under certain assumptions on the sampling and choice of observables, the numerical eigenfunctions and eigenvalues approach the true spectrum of the Koopman operator [25]. Choosing observables that allow for quick convergence is difficult in practice. We use DMD-computed eigenfunctions to initialize our neural net-approximated eigenfunctions and eigenvalues before training. This is described in Section 2.2.2.

The magnitude of an eigenfunction $|\phi_j(\mathbf{f}(\mathbf{x}))|$ is called the "mode power" and gives the relative importance of the j^{th} Koopman mode to the dynamics. We use the mode powers to compare the influences of individual Koopman modes to the dynamics. For example, the coloring of the eigenvalues in Fig. 3.6 are shown scaled to the maximum value of the

mode power attained over the entire training data set. This is used to determine which modes are important to the dynamics of the soft robotic arm in Chapter 3.

1.6.3 Approximation of Koopman Operators for Control Systems: EDMDc

Here, we extend EDMD to systems with input, following the process outlined in [24]. The first step is to choose some finite dictionaries of observables for both the states $\{f_j(\mathbf{x})\}_{j=1}^{n}$ and inputs $\{g_j(\mathbf{u})\}_{j=1}^{n_u}$. It is simple to allow arbitrary input observables, but we only deal with the case where $g_j(\mathbf{u}) = u_j$ in our implementation in Chapter 3. We can now define vectors of observables $\mathbf{z}(\mathbf{x}) = [f_1(\mathbf{x}) \cdots f_n(\mathbf{x})]^T$ and $\mathbf{v}(\mathbf{u}) = [g_1(\mathbf{u}) \cdots g_{n_u}(\mathbf{u})]^T$ called the lifted state and the lifted inputs, respectively. This allows us to represent our dynamics as a linear input-output system

$$\mathbf{z}^+ = A\mathbf{z} + B\mathbf{v}. \tag{1.32}$$

Here A and B are the state transition and input matrices, respectively. This linear, finite dimensional representation has the benefit of enabling the later use of the fast and efficient linear optimal control methods described in Section 3.3.1.

The states are retrieved from the observables using the output equation

$$\mathbf{x} = C\mathbf{z} \tag{1.33}$$

where C is the output matrix.

The approximation of the matrices A, B, and C was introduced in [24] and is a simple modification of the method in Eq. 1.30. Along with our N measurements of the states $\{\mathbf{x}_i, \mathbf{x}_i^+\}$, we also collect N measurements of the inputs $\{\mathbf{u}_i\}$ collected at the same times

as our unshifted state measurements. Next, we build the lifted data matrices

$$X_{\text{lift}} = [\mathbf{f}(\mathbf{x}_1) \ ... \ \mathbf{f}(\mathbf{x}_N)] = [\mathbf{z}_1 \ ... \ \mathbf{z}_N], \tag{1.34}$$

$$X^+_{\text{lift}} = [\mathbf{f}(\mathbf{x}^+_1) \ ... \ \mathbf{f}(\mathbf{x}^+_N)] = [\mathbf{z}^+_1 \ ... \ \mathbf{z}^+_N], \tag{1.35}$$

$$U_{\text{lift}} = [\mathbf{g}(\mathbf{u}^+_1) \ ... \ \mathbf{g}(\mathbf{u}^+_N)] = [\mathbf{v}^+_1 \ ... \ \mathbf{v}^+_N]. \tag{1.36}$$

The desired matrices A and B satisfy the equation

$$X^+_{\text{lift}} = AX_{\text{lift}} + BU. \tag{1.37}$$

In order to approximate A and B, we recast this equation as a minimization problem

$$\min_{A,B} \|X^+_{\text{lift}} - AX_{\text{lift}} - BU\|_F \tag{1.38}$$

which has the solution

$$[A \ B] = X^+_{\text{lift}} \left(\begin{bmatrix} X_{\text{lift}} \\ U \end{bmatrix} \right)^\dagger \tag{1.39}$$

where † is the Moore-Penrose pseudoinverse.

Since we prescribe our first n observables to be the states $\mathbf{x} \in M$, we can compute the output matrix using a partial identity matrix

$$C = \begin{bmatrix} I_{n\times n} & 0_{n\times n_u - n} \\ 0_{n_u - n\times n} & 0_{n_u - n\times n_u - n} \end{bmatrix}. \tag{1.40}$$

Otherwise, C can be computed using another pseudo-inverse

$$C = XX^\dagger_{\text{lift}}. \tag{1.41}$$

The action of the matrices A and B on the lifted state via equation 1.32 approximates the action of the Koopman operator U in equation 1.20. Under certain assumptions, this representation of the Koopman operator converges to the true Koopman operator [25]. True convergence requires infinite data samples which are uniformly distributed in state space and a collection of observables which span an invariant subspace of the Koopman operator's underlying function space.

1.7 Permissions and Attributions

1. The content of chapter 2 is the result of a collaboration with Alan Cao and Dr. Igor Mezić, and a previous version appeared in the Alan Cao's master's thesis. It is reproduced here with the permission of the University of California Santa Barbara.

2. The content of chapter 3 is the result of a collaboration with David A. Haggerty, Dr. Ervin Kamenar, Alan B. Cao, Patrick C. Curtis, Dr. Igor Mezić, and Dr. Elliot W. Hawkes. This work is in review in the journal Science Robotics.

Chapter 2

Learning and Control of Deep Koopman Eigenfunctions

This chapter explores the application of deep neural nets to approximate the eigenfunctions of the Koopman operator. I initiated this project after realizing some of the shortcomings of existing work in this direction. The application of neural nets to the discovery of input observables is novel and came as a natural evolution of these methods to control problems. After I built the first working prototypes of this method, Alan B. Cao significantly extended the codebase and refined our applications. This work eventually became his master's thesis. An updated version of that thesis is reproduced below with permission from Alan. Our work was advised by professor Igor Mezić.

2.1 Introduction

Choosing suitable observables to be used in EDMD is difficult to do when the dynamics is unknown. Poorly chosen observables will result in poor approximations of the dynamics. Of all possible choices of observables, the eigenfunctions are the best because they span a Koopman-invariant subspace. Data-driven eigenfunction approximation methods exist, including those based on the so-called Laplace averages [15]. These methods assume the existence of a stable, attractive fixed point or limit cycle and involve a poorly-conditioned computation on long trajectories of data. There also exists the more direct data-driven method of [19], but this method assume non-periodic dynamics and requires the choice of a basis of so-called "boundary functions".

In 1958, the idea of mathematically modelling biological neurons and their synaptic connections to perform pattern recognition was introduced via the concept of the perceptron [28]. Since then, Artificial Neural Networks (ANNs) have received a great deal of attention due to the development of learning algorithms, such as backpropagation [29], as well as the increasing abundance of data. ANNs consist of many simple interconnected nonlinear systems with weights that are adjusted to improve performance in a process

known as learning. Increasing the depth of the ANNs enhances their performance and rate of learning, resulting in the hierarchical methods being termed deep learning. By the universal approximation theorem [30], ANNs are able to approximate arbitrarily complex functions, holding the potential for finding useful observables to produce superior representations of the Koopman operator.

Existing methods have tried to use neural networks to learn Koopman eigenfunctions from data. In particular, [14] learns observables resembling eigenfunctions, but allows the eigenvalues to vary across the state space, resulting in a nonlinear representation of the dynamics. These methods were successfully applied to systems with continuous spectrum, in which there are no Koopman eigenfunctions except for invariants. However, the resulting representations do not lend themselves readily to linear control methodologies.

In this work, we present a neural network architecture that learns the true Koopman eigenfunctions and eigenvalues from data. This results in a finite-dimensional, globally linear representation of nonlinear dynamical systems. We then extend this representation to systems with input by learning special choices of observables on the input which transform the system into a fully decoupled set of linear evolution equations. This control representation allows for direct pole placement and optimal control of the Koopman modes.

2.2 Deep Koopman Eigenfunctions

Here, we describe our network architecture which learns the Koopman eigenvalues and eigenfunctions of an uncontrolled, nonlinear dynamical system.

2.2.1 Network Architecture

We leverage a deep-neural network to simultaneously learn eigenfunctions and eigenvalues of the Koopman operator from data. In order to achieve this, we use the architecture outlined in Figure 2.1 and minimize the sum of two loss functions $l = \alpha_1 l_1 + \alpha_2 l_2$ (Equations 2.1 and 2.3). In our experiments, we chose $\alpha_1 = \alpha_2 = 1$. Our model is trained on data in the form of trajectories $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_m\}$. For every trajectory, the loss functions used in training are:

1. Reconstruction Loss. First, we introduce a loss function designed to train a vector of nonlinear functions f. This loss enforces that the model behaves as an autoencoder, where $\mathbf{f}(\mathbf{x}) : \mathcal{M} \to \mathbb{C}^n$ is an encoder and the decoder $\mathbf{f}^{-1} : \mathbb{C}^n \to \mathcal{M}$ is learned simultaneously in order to enable the recovery of the original states from the observables.

$$l_1 = \left\| \mathbf{x}_0 - \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}_0)) \right\|_{\text{MSE}} \tag{2.1}$$

2. Prediction Loss. Next, we introduce a loss function that learns a diagonal matrix of eigenvalues

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{2.2}$$

where each $\lambda_i \in \mathbb{C}$ is a constant, and simultaneously constrains the learned observables $\mathbf{f}$ to be eigenfunctions.

$$l_2 = \frac{1}{m} \sum_{k=1}^{m} \left\| \mathbf{x}_k - \mathbf{f}^{-1}(\Lambda^k \mathbf{f}(\mathbf{x}_0)) \right\|_{\text{MSE}} \tag{2.3}$$

Starting at the initial condition $\mathbf{x}_0$, this loss function lifts the state into the space of observables and evolves the observables forward in time n time steps via multiplication by the scalars λ_i. The predicted state is then compared to the true state using the decoder $\mathbf{f}^{-1}$. As opposed to enforcing linearity in the cost function [14], our network is constrained to only evolve in time by repeated multiplication by eigenvalues. Thus, the architecture constrains the model to approximate eigenfunctions and eigenvalues while sufficiently representing the dynamics. By taking the prediction loss over multiple time steps, the compounded error of the repeated multiplication of the eigenvalues are taken into account to produce eigenfunctions well suited for long-term predictions.

In order to generalize to systems with oscillatory behaviors, we equip our model to be able to learn complex-valued eigenfunctions and eigenvalues. We use two sets of real-valued weights to hold the complex quantities. For each eigenvalue/eigenfunction pair, we learn real constants a and b and real-valued functions c and d such that $\lambda_i = a + bi$ and $\phi_i(\mathbf{x}) = c(\mathbf{x}) + d(\mathbf{x})i$.

The final model takes the form N decoupled linear systems, each describing the evolution of an eigenfunction. This takes the form of

$$\begin{bmatrix} z_1^+ \\ z_2^+ \\ \vdots \\ z_n^+ \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}, \tag{2.4}$$

where $z_i = f_i(\mathbf{x})$ are the learned eigenfunctions and λ_i are the corresponding eigenvalues for $i = 1, \ldots, M$. We use the Pytorch framework [31] and the Adam optimizer [32] for all training purposes. The neural networks used are composed of 2 hidden layers each with 20 neurons. Each hidden layer in our model is followed by a rectified linear unit (ReLU)

nonlinear activation function.

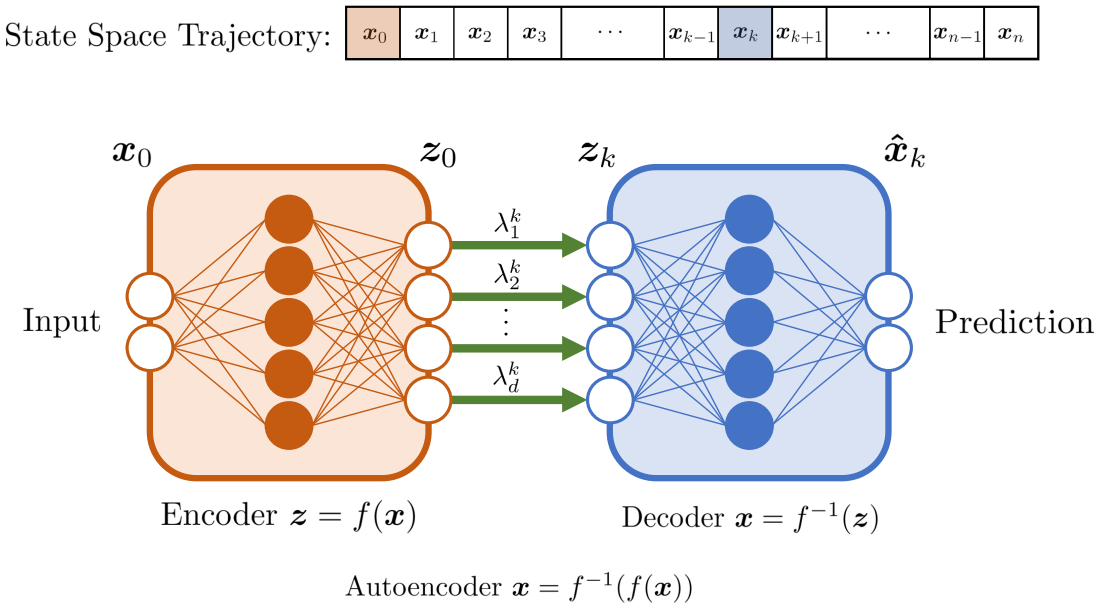


Figure 2.1: Visualization of our deep learning architecture used for approximating Koopman eigenfunctions and eigenvalues. The learned functions $\mathbf{f}$ and $\mathbf{f}^{-1}$ form an autoencoder structure mapping from the data $\mathbf{x}_i$ to a vector of eigenfunctions $\mathbf{z} = \mathbf{f}(\mathbf{x})$, and back. Prediction is done in the eigenfunction space via multiplication by the learned eigenvalues λ_i. The green arrows represent the k-times repeated multiplication of each eigenfunction by its associated eigenvalue.

2.2.2 Initialization with EDMD

We initialize the model with approximations of the eigenvalues and eigenfunctions of the Koopman operator obtained from performing EDMD on time delay observables. Although a better EDMD model can be constructed using more carefully chosen observables, the time delay observables can be readily obtained from the training data. The eigenvalues of the model are directly initialized with the eigenvalues of the EDMD matrix A, while the encoder and decoder are trained to represent the corresponding numerical eigenfunctions $\hat{\phi}$ (Eq. 1.31). Training the network initialized in this way significantly

reduces training time.

2.3 Uncontrolled Numerical Examples

We demonstrate our deep learning model's ability to approximate Koopman eigenfunctions on example systems with discrete real and complex spectra. The learning of real spectra is demonstrated on a simple nonlinear system with known analytic eigenfunctions. For a system with complex spectra, we consider the Van der Pol oscillator which exhibits a highly nonlinear limit cycle and is resistant to fixed-basis techniques such as EDMD.

2.3.1 Discrete Real Spectrum System

We consider a simple nonlinear system with a single fixed point at the origin.

$$\begin{aligned} \dot{x}_1 &= \alpha x_1 \\ \dot{x}_2 &= \beta(x_2 - x_1^2) \end{aligned} \tag{2.5}$$

For this simple system, the eigenfunctions are known:

$$\begin{aligned} \phi_1 &= x_1 \\ \phi_2 &= x_2 - \frac{\beta}{\beta - 2\alpha} x_1^2 \end{aligned} \tag{2.6}$$

with corresponding discrete-time eigenvalues $\lambda_1 = e^{\alpha \Delta t}$ and $\lambda_2 = e^{\beta \Delta t}$ where Δt is the time step. When $\alpha, \beta < 0$, there is a stable manifold at $x_2 = x_1^2$. We choose $\alpha = -0.05$ and $\beta = -1$ for our demonstrations.

We create our datasets by solving the systems of differential equations in (2.5) using a Runge-Kutta (4, 5) solver. We generate a random, uniformly distributed sampling of

200,000 initial conditions in the box $\mathbf{x}_0 \in [-0.5, 0.5] \times [-0.5, 0.5]$ and evolve the solution $m = 256$ time steps with $\Delta t = 0.1$. We separate the data into 90% training and 10% validation sets.

The discrete-time eigenvalues corresponding to $\alpha = -0.05$ and $\beta = -1$ are:

$$\lambda_1 = e^{\alpha \Delta t} \approx 0.99501 \tag{2.7}$$

$$\lambda_2 = e^{\beta \Delta t} \approx 0.90484 \tag{2.8}$$

Deep-Neural Eigenfunctions: Real Case

Here, we use our neural net architecture to learn $n = 2$ eigenfunctions and eigenvalues. Before training the model on the dynamics, we use DMD with state-only observables to initialize the weights and the eigenvalues. For comparison, the numerical eigenfunctions from DMD and the true eigenfunctions from Eq. 2.6 are shown in Figure 2.2 alongside the deep-neural eigenfunctions. The eigenvalues are compared in Table 2.1. Although the eigenvalues produced by DMD are very close to the expected values, the corresponding eigenfunctions do not capture the nonlinearity of the stable manifold. However, the eigenfunctions learned by our neural net architecture are similar to the true eigenfunctions, up to a scalar multiple.

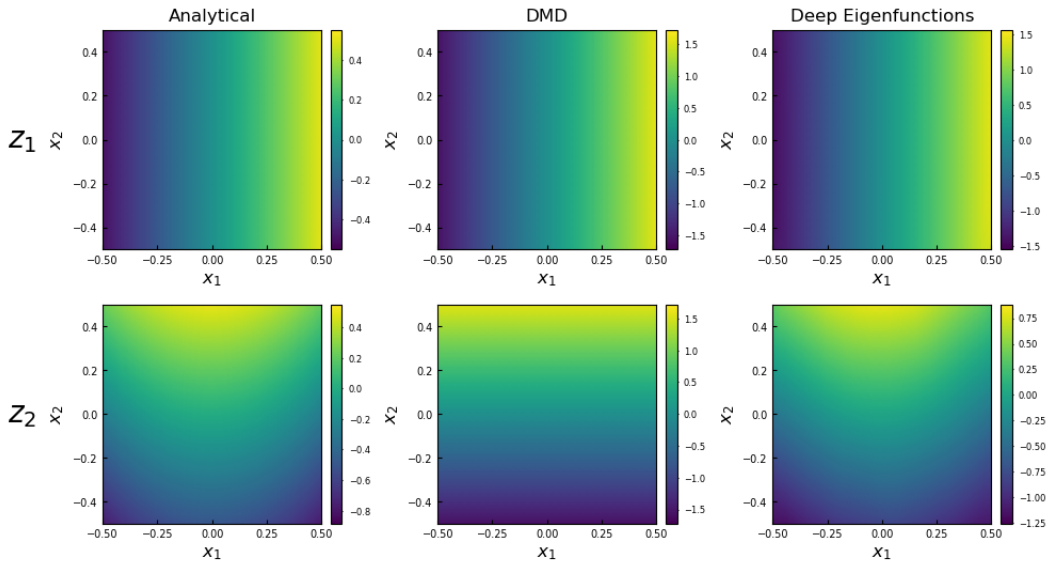


Figure 2.2: Left - Analytic eigenfunctions in (2.6) visualized over the state space. Middle - The numerical eigenfunctions computed using DMD do not capture the parabolic nonlinearity. Right - Deep-neural eigenfunctions produced using our method match the true eigenfunctions.

	λ_1	λ_2
True	0.99501	0.90484
DMD	0.99501	0.90507
Neural Net	0.99503	0.90217

Table 2.1: Comparison of the DMD and our neural net architecture's approximated eigenvalues to the true eigenvalues corresponding to the eigenfunctions given in Eq. 2.8. Note that both methods agree to a high precision, but the deep-neural eigenfunctions are far superior to the DMD-approximated numerical eigenfunctions.

The state space can be partitioned into level sets of ϕ_2 where the dynamics rapidly converge to $\phi_2 = 0$ at the rate of λ_2. The slow manifold shown in Figure 2.3 is the zero level set of ϕ_2. After the slow manifold has been reached, the dynamics slowly converge to the origin at the rate of λ_1 while passing through level sets of ϕ_1.

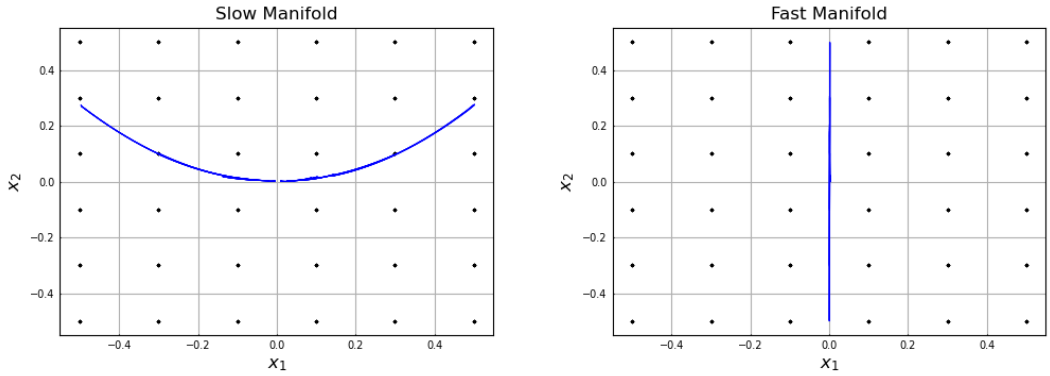


Figure 2.3: The zero-level sets of the learned eigenfunctions reveal the slow and fast manifolds.

Given initial conditions within the training bounds, our model is able to forecast trajectories for long time horizons shown in Figure 2.4 using a compact 2×2 diagonal matrix representation:

$$\begin{bmatrix} \phi_1^+ \\ \phi_2^+ \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}. \tag{2.9}$$

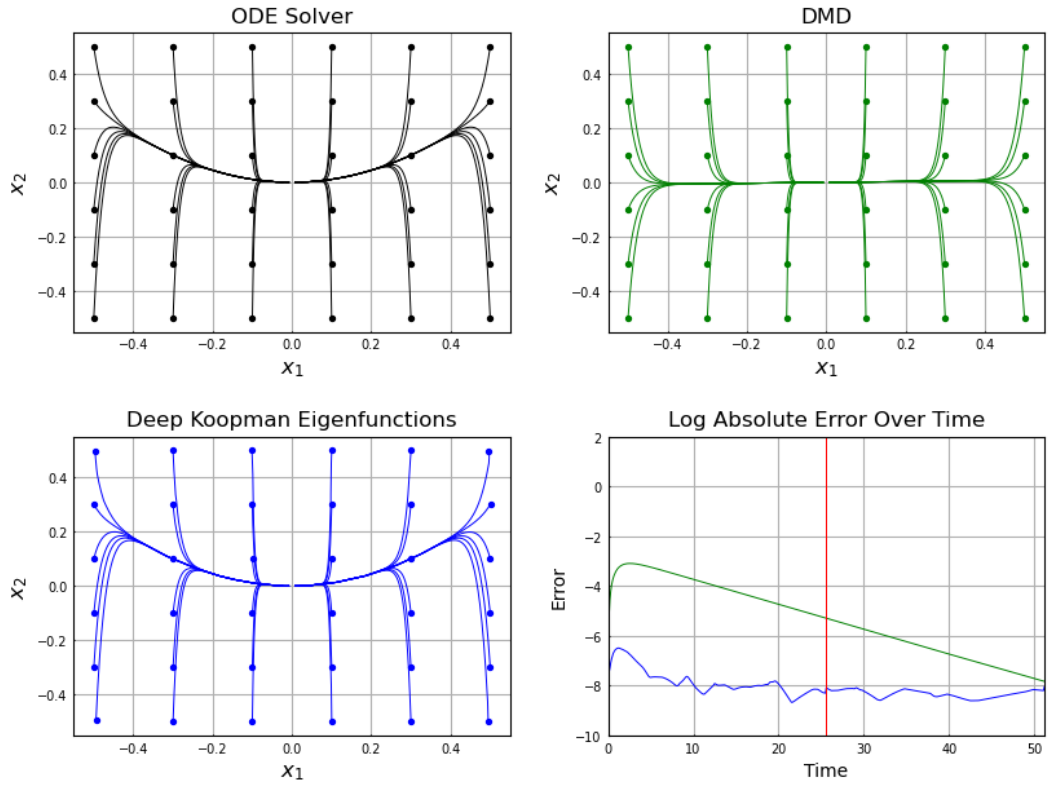


Figure 2.4: Trajectories of 512 steps generated from 36 initial conditions. Black - Trajectories generated using an ODE solver on the nonlinear equations in (2.5). Green - Results from using a 2×2 linear model produced by DMD. Blue - Results from using our 2×2 diagonal matrix representation in (2.9). Red - Training horizon. Log absolute prediction error is averaged over the 36 trajectories.

Redundant Eigenfunctions

The eigenfunctions discovered in Figure 2.2 are called the principal eigenfunctions. If our function space $\mathcal{O}$ is closed under pointwise multiplication of functions, then many more eigenfunctions exist beyond these. In this case, any pair of eigenfunctions ϕ_1 and ϕ_2 with eigenvalues λ_1 and λ_2 generate new eigenfunctions of the form $\phi_1^i \phi_2^j$ with eigenvalues $\lambda_1^i \lambda_2^j$ (see [33]). This applies to all $i, j \in \mathbb{R} \setminus \{0\}$ in general, and $i, j \in \mathbb{R}$ if ϕ_1 and ϕ_2 vanish nowhere. These additional eigenfunctions are not necessary to describe the evolution of the dynamics.

The deep-neural eigenfunction architecture provides a way to discover the number of eigenfunctions required to represent the dynamics. Simply choose n large, learn the eigenfunction representation, then trim away the eigenfunctions with small ℓ^2 norm in the region of interest.

We train our architecture to learn sixteen eigenfunctions for the system in Equation 2.5. The results are shown in Figure 2.5. If we set a cutoff threshold of $\epsilon = 0.1$ and ignore all eigenfunctions with $\|\phi_i(\mathbf{x})\|_2^2 < \epsilon$ in the training region $\mathbf{x} \in [-0.5, 0.5] \times [-0.5, 0.5]$, we see that three eigenfunctions remain.

Using only these three eigenfunctions to generate predictions, we see very little difference compared to using all sixteen. This suggests that our model has the ability to perform dimensional reduction and extract a few key eigenfunctions.

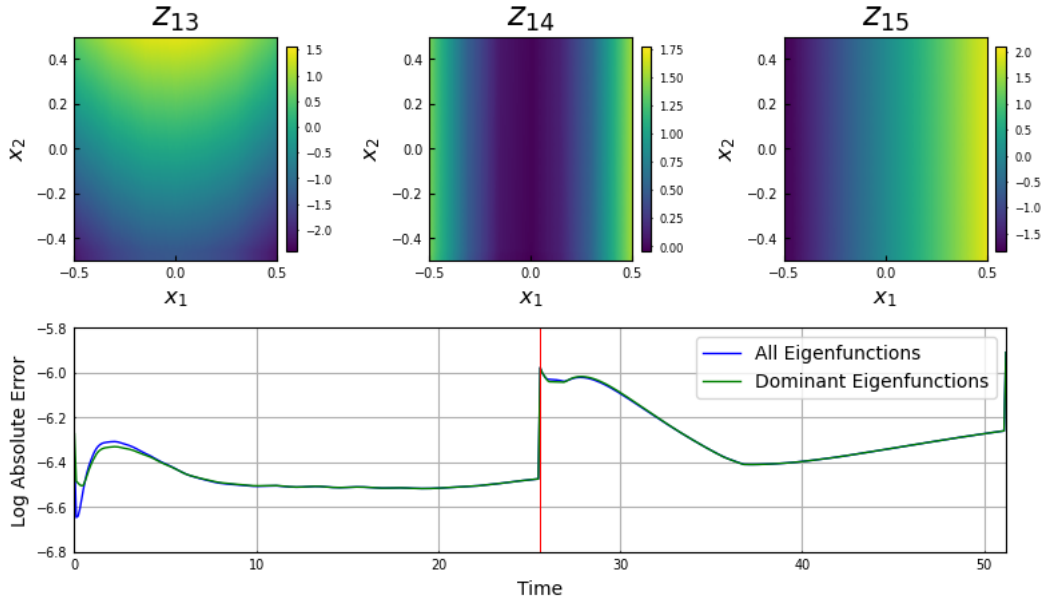


Figure 2.5: Top - The 15th and 13th eigenfunctions have the largest ℓ^2 norm and appear similar to the expected eigenfunctions. They have corresponding eigenvalues at 0.99505 and 0.90459. We also find the 14th eigenfunction to have a significant magnitude and a corresponding eigenvalue at 0.98973. The eigenfunction appears to be x_1^2, which has an expected eigenvalue at $e^{2\alpha\Delta t} \approx 0.99005$. Bottom - Average log absolute prediction error over $N = 36$ initial conditions. Using only the three dominant eigenfunctions results in a very close result to using all sixteen eigenfunctions.

2.3.2 Van der Pol Oscillator

Next, we consider the Van der Pol oscillator. This is a classical example of a nonlinear dynamical system with a limit cycle.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -0.5(x_1^2 - 1)x_2 - x_1 \end{aligned} \tag{2.10}$$

All initial conditions converge to the limit cycle in time, so we should expect any eigenfunction to capture the shape of the limit cycle as well as the rate of decay of trajectories toward the limit cycle. Trajectories also experience periodic motion around the limit cycle, so we expect a complex component that captures the evolution of the phase of the solution. Therefore, we design our deep-neural eigenfunction architecture to detect two eigenfunctions with complex eigenvalues.

The level sets of these eigenfunctions are the isochrons and the isostables of the system [15]. For some given phase θ, the isochron consists of all points such that the trajectory through such points at $t = 0$ asymptotically approaches the trajectory that belongs to a point on the limit cycle with phase θ at $t = 0$. From the period of the limit cycle $T \approx 6.3807$, we expect a corresponding discrete time eigenvalue at:

$$\lambda = e^{\frac{2\pi i}{T}\Delta t} \approx 0.99518 \pm 0.09802i \tag{2.11}$$

Performing the computation of the Laplace average (Section 1.6.1) for the observable $f(x_1, x_2) = x_1 + x_2$ with the period of the limit cycle results in the isochrons f_ω^* on the Van der Pol oscillator visualized in Figure 2.6. On the other hand, isostables consist of all the points that share the same asymptotic convergence toward a fixed point. A second eigenvalue corresponds to the stability of the limit cycle and can be computed

numerically. Taking a surface of section, we can see that the trajectory exponentially approaches the limit cycle at a rate of decay of $\tau \approx -0.5$ with a corresponding discrete-time eigenvalue at:

$$\lambda = e^{\tau \Delta t} \approx 0.9512 \tag{2.12}$$

We create our datasets by solving the systems of differential equations in (2.10) using a Runge-Kutta (4, 5) solver. 200,000 initial conditions are generated randomly from $[-3.0, 3.0] \times [-3.0, 3.0]$ and solved for $m = 256$ time steps of $\Delta t = 0.1$. We separate the data into 90% training and 10% validation sets.

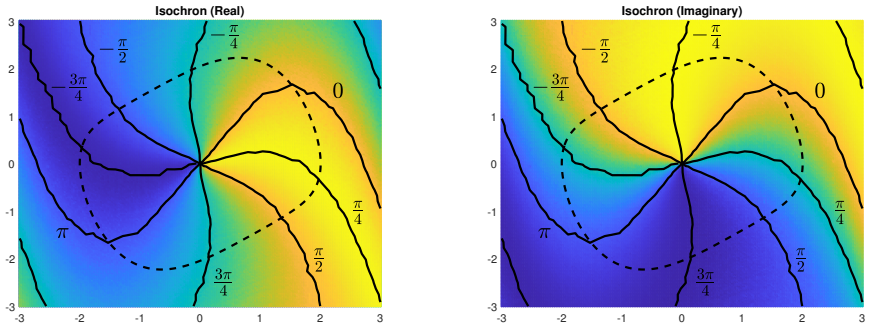


Figure 2.6: Isochrons of the Van der Pol oscillator shown as level sets of a complex eigenfunction. The dashed line represents the limit cycle and the solid lines represent level sets of the complex argument $\angle f$.

Deep-Neural Eigenfunctions: Complex Case

The results of both the DMD-initialized numerical eigenfunctions and the resulting deep-neural eigenfunctions are shwon in Figure 2.7. DMD produces a complex conjugate pair of eigenvalues with approximately the expected periodicity. However, the original coordinates are poor choices of observables and the resulting model from DMD completely fails to capture the presence of the limit cycle. However, the deep-neural eigenfunctions do succeed in capturing the limit cycle. In fact, the zero-level set of the eigenfunction z_1

is the limit cycle itself, shown in Figure 2.8. The first eigenvalue λ_1 captures the periodic motion of the system and is similar to the expected value in Eq. 2.11. Notably, the level sets of the first eigenfunction learned by our neural network model resemble the isochrons shown in Figure 2.6. The second eigenvalue λ_2 is associated with the stability of the limit cycle and approaches the expected value in Eq. 2.12.

Next, we test the predictive power of our eigenfunctions by building the 2×2 linear, uncoupled system of equations from Eq. 2.9. As shown in Figure 2.9, the deep-neural eigenfunction model is able to successfully predict trajectories that traverse the limit cycle for multiple periods.

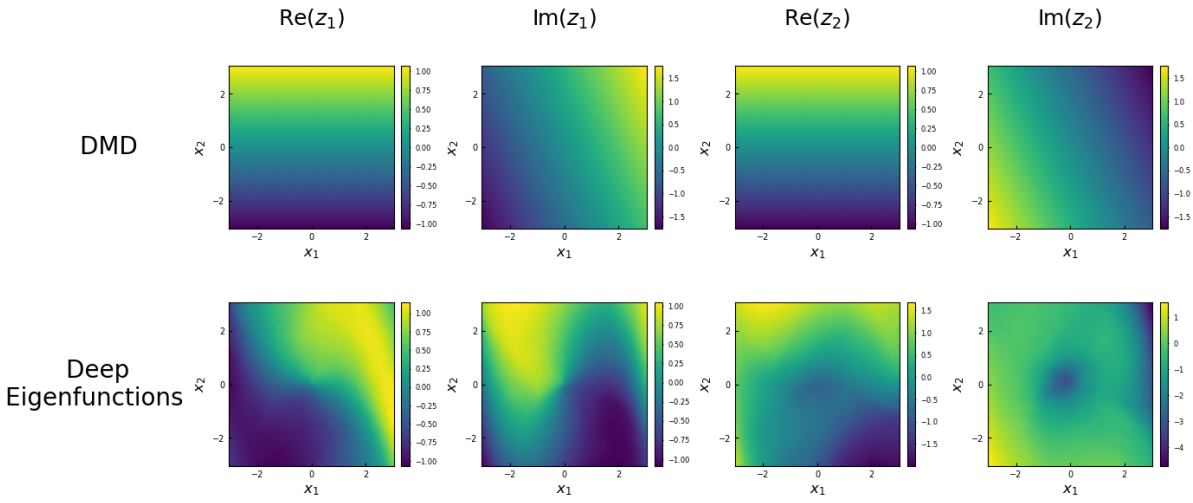


Figure 2.7: Top - Numerical eigenfunctions computed using DMD. The eigenvalues are learned to be $0.97052 \pm 0.09703i$. Bottom - Numerical eigenfunctions produced using our method. The eigenvalues are learned to be $0.99518 - 0.09831i$ and $0.95168 - 0.00024i$. Note that Koopman eigenfunctions are invariant to scaling.

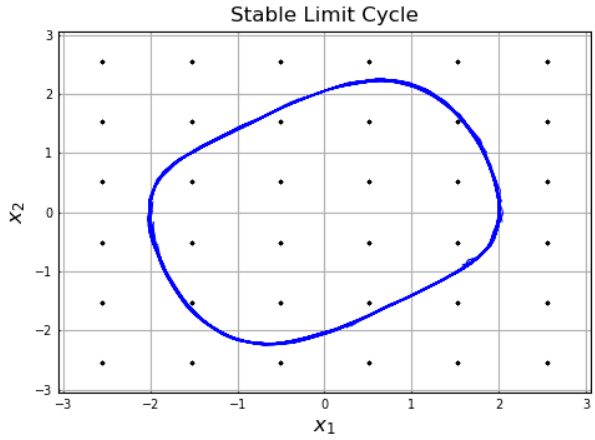


Figure 2.8: The limit cycle of the Van der Pol oscillator obtained from the zero-level set of the eigenfunction associated with the periodic motion of the system.

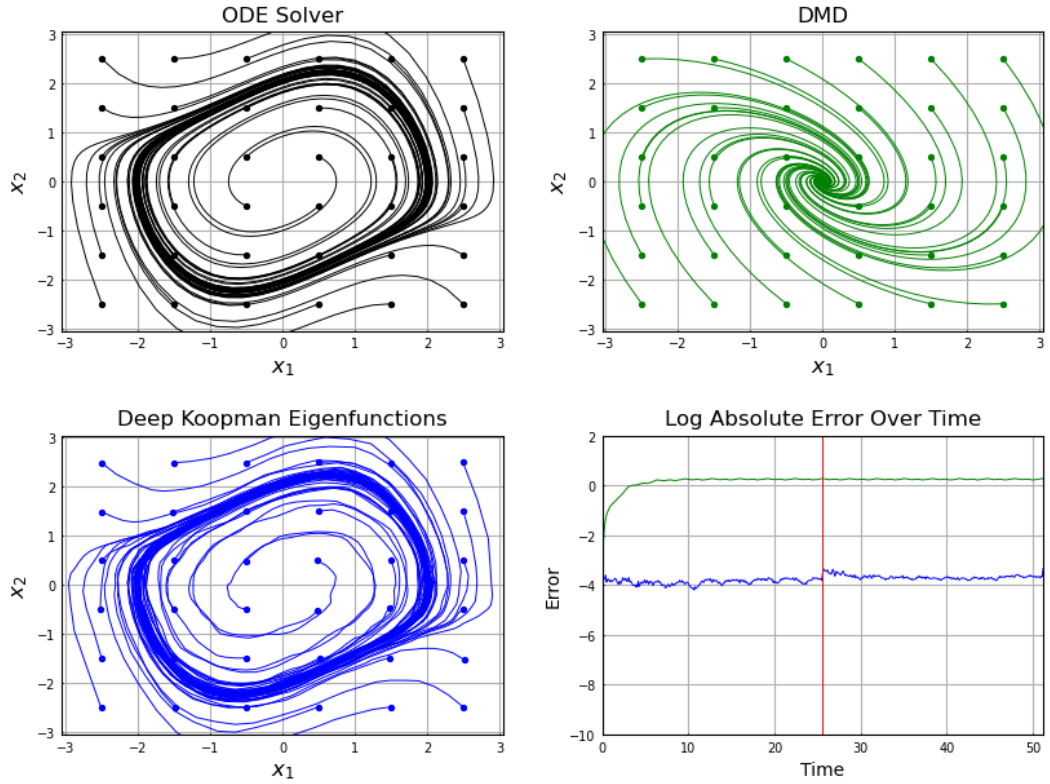


Figure 2.9: Trajectories of 512 steps generated from a 6×6 grid of initial conditions. Black - Trajectories generated using an ODE solver on the nonlinear equations in (2.10). Green - Results from using a 2×2 linear model produced by DMD. Blue - Results from using our 2×2 diagonal matrix representation. Red - Training horizon. Log absolute prediction error is averaged over the 36 trajectories.

EDMD involves lifting into higher-dimensions to produce improved approximations of the Koopman operator. These methods produce large sparse matrices that exhibit relatively poor long-term predictive performance. In contrast, our compact diagonal representation (blue) is shown in Figure 2.10 to have superior long-term predictive performance, even when compared to larger representations that depend on user-defined observables. In particular, we compare our model to the predictors developed in [24] with 100 thin plate spline radial basis functions as the lifting functions. We also compare our results to a Hankel-DMD representation built on 10 time-delays. We find diminishing returns when constructing a model using more than 10 time-delays.

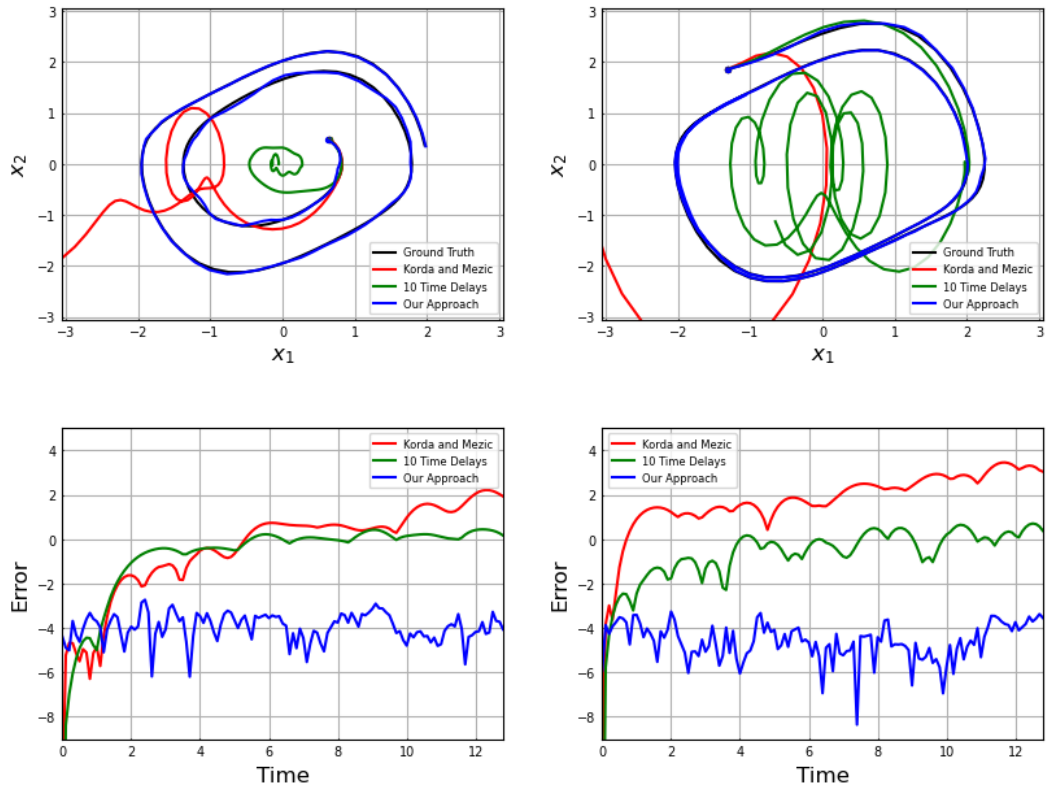


Figure 2.10: Trajectories of 128 steps generated from two initial conditions. Left - Predictions given an initial condition inside the limit cycle. Right - Predictions given an initial condition outside the limit cycle. Black - Trajectories generated using an ODE solver on the nonlinear equations in (2.10). Red - Results from using a 102×102 linear predictor developed in [24]. Green - Results from using a 22×22 linear model produced by Hankel-DMD with 10 time-delays. Blue - Results from using our 2×2 diagonal matrix representation.

2.3.3 Deep-Neural Eigenfunctions with Input Observables

We are interested in predictors that possess a structure amenable linear control frameworks. To achieve this, we will simultaneously learn the n eigenfunctions of the uncontrolled system as in Section 1.2 as well as a collection of n_u input observables g_i. Similar to the model used to learn the Koopman eigenfunctions, a pair of vector-valued functions $\mathbf{g} : \mathbb{R}^P \to \mathbb{R}^{n_u}$ and $\mathbf{g}^{-1} : \mathbb{R}^{n_u} \to \mathbb{R}^P$ which form a second autoencoder model $\mathbf{u} = \mathbf{g}^{-1}(\mathbf{g}(\mathbf{u}))$.

We assume a linear predictor that takes the form of a linear dynamical system

$$\mathbf{z}^+ = \Lambda \mathbf{z} + \mathbf{v}, \tag{2.13}$$

where $\mathbf{z} = \mathbf{f}(\mathbf{x})$ is the lifted state, $\mathbf{v} = \mathbf{g}(\mathbf{u})$ is the lifted input, and Λ is a diagonal matrix of learned complex numbers. Note that this resembles the classical linear MIMO state-space representation of a dynamical system, without the typical input matrix B. Instead of learning an input matrix, we choose to allow the input lifting to handle all transformations required. This requires selecting $n_u = n$, giving us n uncoupled, linear SISO systems which can be handled separately. Each lifted state z_i is uncoupled from the rest, and is influenced only by a single lifted input v_i.

To predict future states, we use the state decoder $\mathbf{f}^{-1}$ to obtain $\hat{\mathbf{x}}^+ = \mathbf{f}^{-1}(\mathbf{z}^+)$. It is important to note that in general, the lifted states are not linearly dependent on some function of the inputs. However, if the linear predictor provides accurate predictions for a sufficient time horizon, the predictions can be used in linear control methodologies such as model predictive control (MPC) and linear quadratic regulator (LQR) that require finite time horizons of predictions.

We use the following three loss functions to learn the functions $\mathbf{f}$, $\mathbf{f}^{-1}$, $\mathbf{g}$, $\mathbf{g}^{-1}$, and the constants along the diagonal of Λ.

$$l_1^{\text{input}} = \left\| \mathbf{x}_0 - \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}_0)) \right\|_{\text{MSE}} \tag{2.14}$$

$$l_2^{\text{input}} = \left\| \mathbf{u} - \mathbf{g}^{-1}(\mathbf{g}(\mathbf{u})) \right\|_{\text{MSE}} \tag{2.15}$$

$$l_3^{\text{input}} = \frac{1}{m}\sum_{k=1}^{m} \left\| \mathbf{x}_k - \mathbf{f}^{-1}(\Lambda^k \mathbf{f}(\mathbf{x}_0) + \mathbf{g}(\mathbf{u})) \right\|_{\text{MSE}} \tag{2.16}$$

The complete loss function minimized by our architecture is $l^{\text{input}} = \alpha_1 l_1^{\text{input}} + \alpha_2 l_2^{\text{input}} + \alpha_3 l_3^{\text{input}}$. In our experiments, we chose $\alpha_1 = \alpha_2 = \alpha_2 = 1$.

With this architecture, the dynamics are decomposed into uncoupled eigenfunctions with corresponding eigenvalues that make up the diagonal matrix A. Lifting the control effort to the same dimension as the eigenfunctions establishes a one-to-one correspondence between approximated eigenfunctions and lifted inputs. Our linear predictor takes the form

$$\begin{bmatrix} z_1^+ \\ z_2^+ \\ \vdots \\ z_n^+ \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \tag{2.17}$$

where $v_i = g_i(\mathbf{u})$ for $i = 1, \ldots, n$ are lifted inputs that correspond to the ith eigenvalue-eigenfunction pair. By posing the linear model in this diagonal form, it becomes straightforward to control specific eigenvalues or eigenfunctions using feedback control.

2.3.4 Closed-Loop Control

The decoupled system in Eq. 2.17 allows for the application of linear control design in the lifted inputs $\mathbf{v}$. For example, a feedback control law will take the form of

$$\mathbf{v} = -K\mathbf{z} \tag{2.18}$$

where K is a feedback gain matrix computed by optimal control algorithms such as LQR or pole placement.

The autoencoder structure and the existence of the learned function $\mathbf{g}^{-1}$ allows for control laws developed for the lifted inputs to be transformed back to the original input space. The final feedback control law takes the form

$$\mathbf{u} = \mathbf{g}^{-1}(-K\mathbf{f}(\mathbf{x})). \tag{2.19}$$

2.4 Controlled Numerical Examples

Here, we demonstrate applications of the deep-neural Eigenfunction architecture to linear feedback control design in nonlinear systems. First, we revisit the control of the discrete real spectrum system in Section 2.4.1. Next, we apply our methods to the Van der Pol system with forcing in Section 2.4.2.

2.4.1 Discrete Real Spectrum with Input

We consider the system in (2.5) modified by the addition of a control effort u.

$$\begin{aligned} \dot{x_1} &= \alpha x_1 \\ \dot{x_2} &= \beta(x_2 - x_1^2) + u. \end{aligned} \tag{2.20}$$

To learn the lifting of the control effort, we simulate trajectories using random input signals with uniform distribution over the interval $u \in [-1, 1]$ to learn the lifted-controlled

system:

$$\begin{bmatrix} \phi_1^+ \\ \phi_2^+ \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{2.21}$$

Figure 2.11 shows the prediction of a trajectory generated using a random input signal. This is in close agreement with the true dynamics.

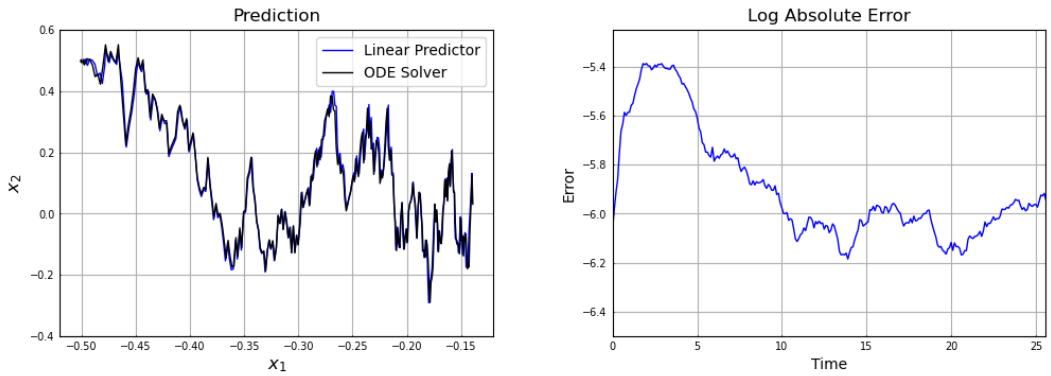


Figure 2.11: Prediction results for the system described in Eq. 2.21. Left: The response of the deep-neural eigenfunction model under random forcing agrees well with the true response. Right: The prediction error is shown in log scale as a function of time.

We now demonstrate feedback control applied to our linear predictor. Our control objective is to modify the rate of convergence of trajectories onto the stable manifold $x_2 = x_1^2$. Suppose we want to achieve a rate of convergence of γ so that $\phi_2^+ = \gamma\phi_2$. We simply choose

$$u = \mathbf{g}^{-1}((\gamma - \lambda_2)\phi_2). \tag{2.22}$$

This is simply the classical linear control technique of pole placement, applied onto our transformed coordinates. Figure 2.12 shows the results of the control effort applied to the system.

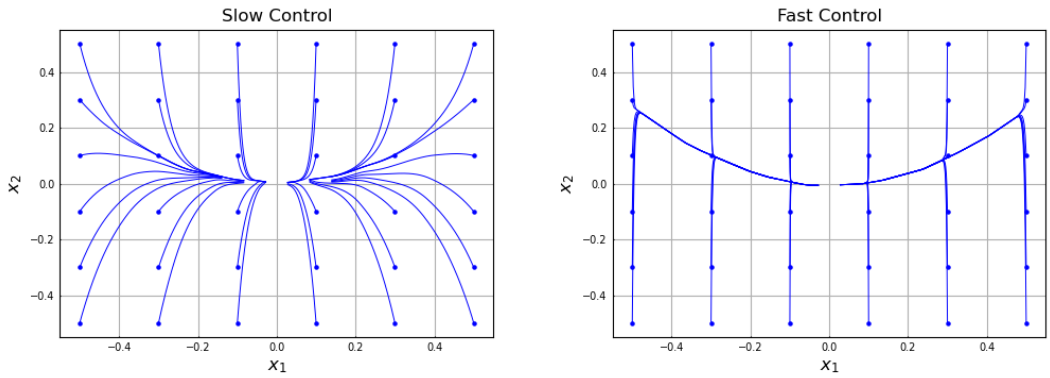


Figure 2.12: Results from applying a control effort to modify the rate of decay of trajectories onto the stable manifold $x_2 = x_1^2$. Left - Control with $\gamma = 0.999$ slows the stabilization of trajectories. Right - Control with $\gamma = 0.2$ speeds up the stabilization of trajectories.

2.4.2 Van der Pol with Input Forcing

Next we investigate the forced Van der Pol Oscillator with dynamics given by

$$\dot{x_1} = x_2 + u_1 \tag{2.23}$$

$$\dot{x_2} = -0.5(x_1^2 - 1)x_2 - x_1 + u_2. \tag{2.24}$$

Similar to the discrete real spectrum system, we simulate trajectories using random input signals with uniform distribution over the interval $\mathbf{u} \in [-1, 1] \times [-1, 1]$ to learn a lifted-controlled system in the same form as (2.21). Figure 2.13 shows the prediction generated using a random input signal compared to the effect of that signal on the actual dynamics. The accuracy of the predictor is limited to short horizons because there do not necessarily exist lifted inputs which completely linearize the input term.

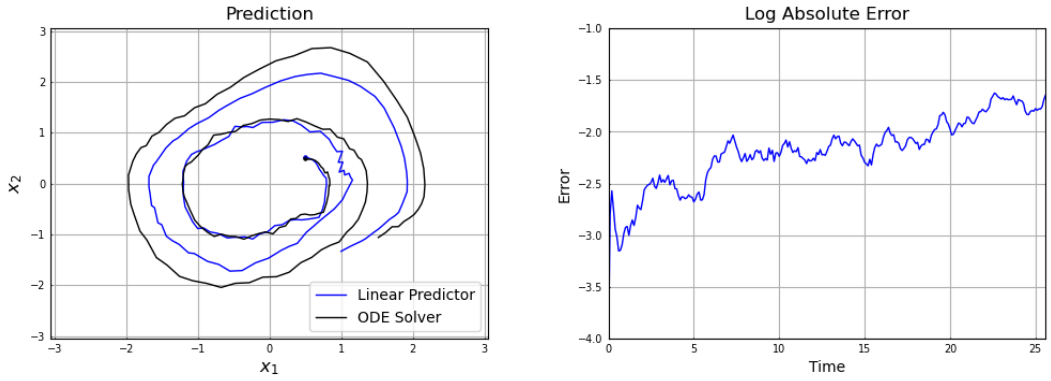


Figure 2.13: Results from learning the predict the Van der Pol system with random inputs. A random input signal is generated that was not used in the training of the model, and the trajectory of the linear predictor is compared with the trajectory of the dynamics.

Here, we demonstrate that this model can be used in prediction. Our control objective is to achieve a specific periodicity on the limit cycle. This can be easily done by modifying the gain associated with the eigenfunction that influences the periodic motion of the system. Figure 2.14 shows the results of the control effort applied to the original nonlinear dynamics.

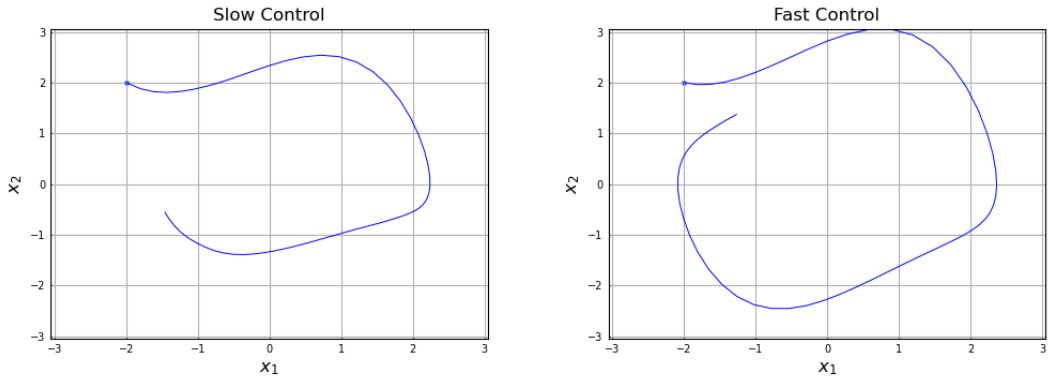


Figure 2.14: Results from applying control effort to decrease and increase the period of the limit cycle. Left - Control with $\gamma = 0.998 - 0.06321js$ designed to slow the periodic motion around the limit cycle. Right - Control with $\gamma = 0.994 - 0.10938i$ designed to speed up the periodic motion around the limit cycle. Both trajectories run for an equal amount of time, but the fast control trajectory outpaces the slow control trajectory.

Chapter 3

Control of Soft Robots with Inertial Dynamics

This work applies both dynamic and static Koopman Operator Theoretic to the data-driven optimal control of a soft robotic arm. This was a collaboration with David A. Haggerty, Ervin Kamenar, Alan B. Cao, Patrick C. Curtis, Igor Mezić, and Elliot W. Hawkes. My main contribution was in building the modeling and control codes, although all authors participated in the project at every level. The idea of the static Koopman pregain term in our control law was mine. The main benefit of this methodology is in its speed, simplicity, and versatility. Future work will involve applying this to increasingly sophisticated robots.

3.1 Introduction

The automation and robotics revolution has transformed manufacturing and heavy industry, leading to higher throughput, repeatability, and quality across numerous sectors [34, 35]. Unfortunately, robots are most often relegated to cages and isolated sections of manufacturing sites due to the inherent danger they present to human operators through their fast-moving, heavy, and rigid structures. Efforts towards allowing these robots to perform safely with human collaborators have focused on software control, but absolute guarantees of safety are not possible [36, 37, 38, 39].

In contrast, soft robots are safe by construction due to their low stiffness and mass, but modeling and control of these systems is challenging [40, 41, 42, 43, 44, 45]. This is due to their inherent nonlinearity, high dimensionality, and the imprecise measurement of their position in space. Past work has sought to overcome these obstacles through a variety of modeling methods, each of which constrains the design of control implementations. The majority of these modeling approaches fall into two categories: analytical Reduced Order Modeling (ROM), and machine learning (ML).

In soft robot ROM for control, the aim is to develop an analytical model based on

simplifying assumptions such as (piecewise) constant curvature ((P)CC) deformations [46, 47, 48, 49]. For an approximately constant-curvature system, this approach allows for the accurate prediction of dynamics given appropriate estimation of parameters. However, developing these analytical models is nontrivial and labor intensive, and each model applies only to the single system that was modeled. These models tend to be valid only in a neighborhood around the equilibrium point where the system has been linearized [47]. Controllers based on ROM models have been applied to soft robots in the past, but these have yet to achieve the real-time control of fast, inertial motions [50, 51, 52].

In the ML modeling of soft robot dynamical behaviors, many neural net-based approaches exist. Most of this work focuses on the development of predictors using neural nets such as Long Short Term Memory (LSTM) [53, 54] or recurrent neural networks [50, 52]. These methodologies generate highly accurate predictors of the dynamics. However, training these systems has a high computational cost. Moreover, their structure is nonlinear, requiring specialized control algorithms [43]. One example is a feedforward neural net controller which has been successfully coupled to a model-free closed-loop controller and applied to a high-deflection, yet quasi-static soft arm [55, 56]. Additionally, there are approaches which leverage a neural-net-based dynamical model in closed-loop control [50, 51]. However, neural net approaches to soft robot modeling have not yet resulted in the closed-loop control of high-speed, inertial and nonlinear dynamics.

Koopman Operator Theory-based ML has two qualities that make it attractive strategy for soft robot control: it is data-driven, eliminating the need for complicated analytical models, and it identifies a globally linear model, allowing for fast and efficient control design. A linear Koopman representation contrasts with the common method of state-space linearization, which builds a linear approximation of the nonlinear dynamics only valid in a small region of the workspace. The Koopman methodology has been applied to control systems, with the majority of work combining a Koopman operator approxima-

tion method, Dynamic Mode Decomposition, with control (DMDc) [57, 58, 59, 25, 60, 61]. In particular, model predictive control (MPC) is commonly used [24]. When DMDc and MPC are applied to soft robots, [62, 63, 64, 65] the control is accurate, but only shown so in quasi-static control in a low-deflection regime (approx. 18° of curvature). It is important to note that simple linearized models are likely to work at these low deflections because the full nonlinearity of the dynamics may only be explored at high deflections. In fact, references [62, 63, 64] show imperfect yet functional controllers using purely state-space linear MPC, suggesting the quasi-linearity of these systems.

The combination of the Koopman operator and the Linear Quadratic Regulator (K-LQR) optimal control scheme has shown promise in rigid robot applications [66, 67] and the control of fluid dynamics problems [68]. Notably, Mamakoukas et al. [69] show promise in a 1-DoF soft robotic fish application employing a similar Koopman structure.

Even with these many advances in the field, existing soft arm control implementations [70, 62, 63, 64, 69] have yet to be demonstrated in the inertial, non-linear regime. In order to compare with other works, we introduce the following definitions of the "inertial regime" and "nonlinear dynamics." We define the inertial regime for soft arms to be when the inertial force experienced by the tip F_{tip} is of the order of its weight $F_{\text{tip}} = ma_{\text{tip}} \approx mg$, meaning $a_{\text{tip}} \approx g$. Here m is the mass of the tip of the arm and a_{tip} is the acceleration of the tip during closed-loop control. We define nonlinear dynamics to be motions that fail to be adequately captured by a state-space linearization (see Fig. 3.6 for nonlinearity of output versus input for our system). Thus, an open challenge remains: modeling and control of inertial dynamics in highly nonlinear soft robots.

3.1.1 Our Contribution

In this work, we offer the following three contributions, shown visually in Fig. 3.1:

1. *We advance modeling and control of soft, continuum arms into the inertial regime.* Previous work has considered quasi-static motions, with accelerations below $0.03g$ where $g = 9.81\frac{m}{s^2}$. Our work demonstrates movements in closed-loop control with accelerations greater than $1g$ (see Table 3.1).

2. *We control these inertial movements in a highly nonlinear, high-deflection regime.* We demonstrate curvatures up to 110 degrees in Robot #1 and up to 180 degrees in Robot #2.

3. *We introduce the concept of a static Koopman pregain.* We construct a static Koopman operator, which maps held inputs to converged robot configurations. After being learned from data, we use it as a pregain term in the LQR implementation. The static Koopman pregain greatly increases the accuracy of static pointing tasks and improves the stability of dynamic tasks.

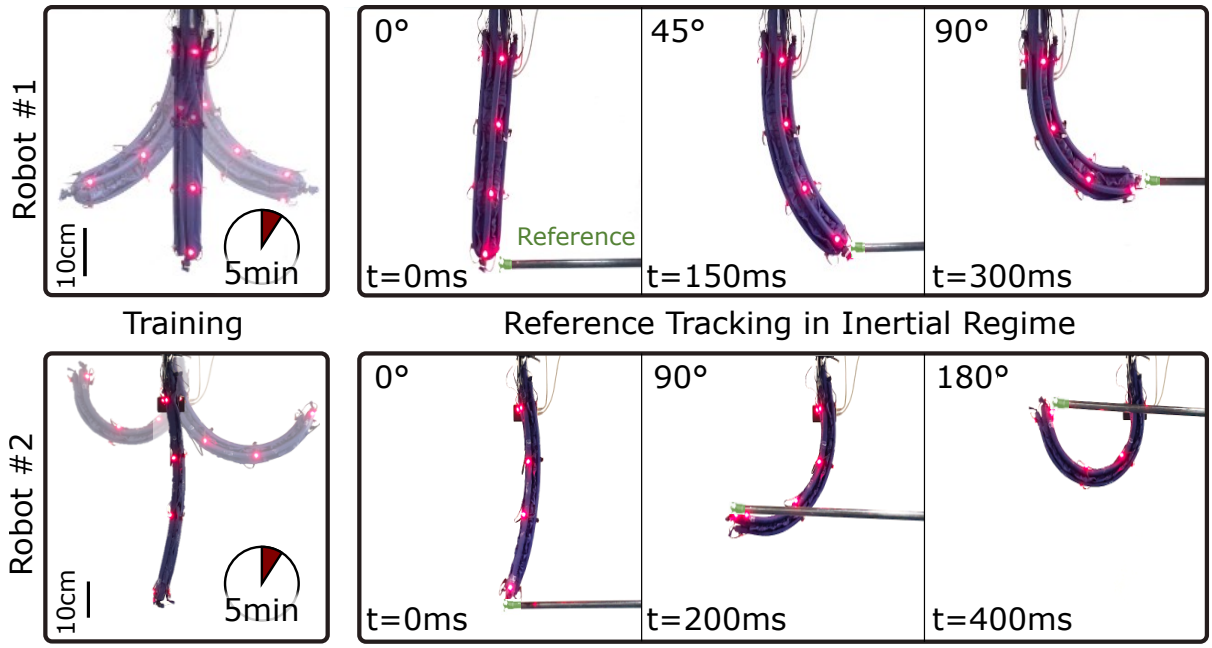


Figure 3.1: **Inertial, nonlinear soft arm control.** Using a combined static and dynamic Koopman framework, we achieve the closed-loop control of soft robotic arms exceeding 10x the tip speed, 40x the tip acceleration, and 6x the angular displacement of existing soft arms. This achievement brings soft robotics into the inertial, nonlinear regime. Only five minutes of training is required to achieve an optimal controller capable of high-deflection, high-accuracy closed-loop tracking of a reference (the tip of a pole moved rapidly by a human). The same methodology is applied to both a low-slenderness-ratio, four-muscle arm (Robot #1) and a high-slenderness-ratio, three-muscle arm (Robot #2). Both arms achieve their highest deflection in under half a second.

Importantly for feasibility in the realm of soft robotics, we show our approach requires minimal training and low computational cost, both for determining the model and controlling the robot. Collecting our training data takes less than 5 minutes and the computation of the model takes less than a second, as opposed to the long training times required by many neural net-based approaches. Our approach estimates both the static and dynamic control Koopman operators, enabling the use of low latency, efficient optimal control methods; this enables real-time tracking of fast-moving reference positions, even if field-deployed on a low-power microcontroller. Additionally, we demonstrate our

Table 3.1: **Comparison with existing soft, coninuum arms shows advances in speed, acceleration, and deflection during closed-loop control.** This work demonstrates a 10x increase in reference tracking tip speed [**Speed**] and a 4x improvement in tip deflection angle [**Deflection**] and advances closed-loop control of soft robot arms into the inertial regime $a_{\text{tip}} > g = 9.81\frac{m}{s^2}$. The acceleration [**Accel**] of the soft arm's tip a_{tip} is computed using the centripetal acceleration of soft arms for which circular reference tracking data is available. The distance from the base to the tip of each arm is also given [**Length**]. Note that closed-loop deflection data does not include the large-deflection open-loop tests present in some works. Acronyms: LQR - Linear Quadratic Regulator, RNN - Recurrent Neural Network, ROM - (analytical) Reduced Order Model, PCC - Piecewise Constant Curvature, MPC - Model Predictive Control, LSTM - Long Short-Term Memory, TRPO - Trust Region Policy Optimization, GPR - Gaussian Process Regression, TO - Trajectory Optimization, FFC - feedforward compensator, SM - sliding mode, AF - analytical feedback, R1: Robot #1, R2: Robot #2.

Robot	**Length** $[m]$	**Speed** $[\frac{m}{s}]$	**Accel** $[\frac{m}{s^2}]$	**Deflection** $[deg]$	**Model**	**Control Method**
This Work	0.37	1.52	11.6	R1: 110 R2: 180	Koopman	LQR
[50]	0.4	0.15		21	RNN	TO
[56]	0.3	0.12	0.065	45	None	NN FFC
[71]	0.3	0.1	0.1	20	ROM	SM
[65]	0.15	0.094	0.29	18	Koopman	MPC
[48]	0.38	0.09	0.032	27	PCC ROM	AF
[51]	0.44	0.05		19	LSTM	TRPO
[70]	0.25	0.035	0.012	7	Koopman	MPC
[64]	0.7	0.03	0.032	8	Koopman	MPC
[52]	0.22	0.002	0.0016	11	RNN	GPR

approach's generalizabiltiy to at least two variations of our soft arm, each with different dimensions, numbers of actuators, and workspaces, as shown in Fig. 3.1.

3.2 Results

In this section, we first outline our approach that enables modeling and control in the inertial, nonlinear regime, yet requires relatively little training data and low computational power. Next, we systematically test the speed and accuracy of the resulting

closed-loop controller in a series of circular reference tracking tests. The soft arm is further tested in a tip tracking test with a rapidly changing, user-defined reference position designed to test the soft arm's responsiveness to changes in commands in real time. Lastly, we test our methodology on the dynamic catching and throwing of a ball. This leverages the inertial dynamics of our soft arm to demonstrate its effectiveness in real-world tasks.

3.2.1 Static and Dynamic Koopman Operator Optimal Control

The successful real-time control of a soft arm in the inertial and nonlinear regime requires both a model that captures these dynamics and a control methodology that adapts to the motion of the robot in real time. We achieved this by building a controller which leverages both the static and dynamic Koopman operators of the soft arm system. The data is collected through a series of training experiments, performed by commanding step inputs with randomly distributed magnitudes. This training data is partitioned into dynamic and static components which are used to train the two separate Koopman operators (see Sec. 3.3) Both the training and model computation processes are fast, requiring only 5 minutes ($\approx 18,000$ samples at 60Hz collection rate) for training data collection (See Fig. 3.7) and the matrix pseudo-inverses used in the model construction take less than a second on an ordinary laptop computer.

The observables used to train the dynamic Koopman model are time delayed measurements of the position of motion tracking points placed on the soft arm. This turned out to be sufficient to build a linear model of its nonlinear dynamics. Previous work considered adding a single time delay to hundreds of monomials [70]. However, inspired by the fact that for ergodic systems, the limit of infinitely many time-delay observables results in DMD's convergence to the true Koopman operator [61, 60, 18], we included only

time-delay observables. Our results show that time-delay-only observables are sufficient to capture the dynamics of this nonlinear system (see Supplementary Fig. 3.9), without the added computational cost of many monomial observables. This also eliminates the large tails associated with monomials which magnify noisy measurements far from the origin. Indeed, without time delays, the eigenvalues in the high frequency and dissipative regions of the unit circle and their corresponding Koopman modes are missing (Fig. 3.6). We express this dynamic Koopman operator as a pair of matrices A and B giving the uncontrolled and controlled dynamics, respectively. These can be used to build the Koopman-LQR controller described in Sec. 3.3.

The resulting feedback controller is able to command the soft arm to follow a fast-changing reference position, but suffers from steady state error. Introducing integral control (e.g. Linear Quadratic Integral Control) is one of the commonly used approaches to minimizing steady-state error [72]. This method, however, is sensitive to measurement noise and requires a trade-off between speed of response and tracking accuracy. As a consequence, the implementation in this work – accurate control of highly dynamic tasks – resulted in poor tracking performance outside of the quasi-linear and quasi-static regimes. Instead, we address this by introducing a static Koopman pregain, a control concept we developed for the current work. The static Koopman operator was first formally described in our recent modeling work [73], but no connection to control design was made. Unlike the dynamic Koopman operator, this operator is a map between functions defined on two different spaces. In our application, the static Koopman operator is used to map functions defined on the space of inputs to functions defined on the space of robot configurations. We learn this operator from the static partition of the training data so that static positions in the workspace of the soft arm correspond to the values of the inputs required to reach those positions after all transient motions dissipate. This operator is then used as a pregain term that augments the LQR controller. Sensor

noise is known to cause tracking issues in soft robots attempting to perform real-time tracking of aggressive control inputs [44]. Our control structure mitigates this problem by balancing the noise-sensitive dynamic Koopman LQR term with the sensor-agnostic static Koopman pregain.

The construction of the controller and the computation of the optimal input are also fast processes which have low computational overhead. The solution of the Riccati equation involved in computing the LQR control gain takes less than a second, and computing the optimal input at a given time step only requires two small matrix multiplications. This is easily achieveable in real-time on a low-cost microcontroller.

3.2.2 Closed-loop circle-tracking in inertial, nonlinear regime

With our control architecture in place, we first sought to characterize the performance across a range of deflections and soft arm speeds in a planar circular reference tracking (smooth changes in reference position). We commanded the soft arm's tip to trace out circular paths in the X-Y plane with three radii (100mm, 180mm, and 220mm) and six frequencies ($0.1, 0.3, 0.5, 0.7, 0.9$, and 1.1Hz), as shown in Fig. 3.2. The same controller was used for all references, as described in 3.3.

These results show that the soft arm tracks the reference with consistent performance throughout the full range of deflections and speeds tested (Fig. 3.2, *left*, Video S2). The fastest and highest deflection circle-tracking result demonstrates a tip speed of 1.5m/s, a speed to length ratio of 3.23s^{-1}, and a tip acceleration of 11.6m/s^2 in closed-loop control. This is approximately an order of magnitude faster than any soft arm of which we are aware (see Table 3.1). Importantly, the system was trained exclusively on step inputs, and as such the model had no *a priori* knowledge of the control objective nor had it been trained on circular behaviors.

Additionally, we show that the relative contribution of the dynamic Koopman LQR input versus the static Koopman pregain increases with increasing speed and deflection (Fig. 3.2, *right*). For relatively low speeds and deflections, the dynamic Koopman LQR input is quite small, and the static Koopman pregain dominates. As accelerations increase and inertia becomes non-negligible, the dynamic component increases in magnitude to compensate for the static term's inability to account for inertial effects. This suggests that for any soft robot performing a non-inertial task, the incredibly simple static Koopman pregain could be sufficient for control.

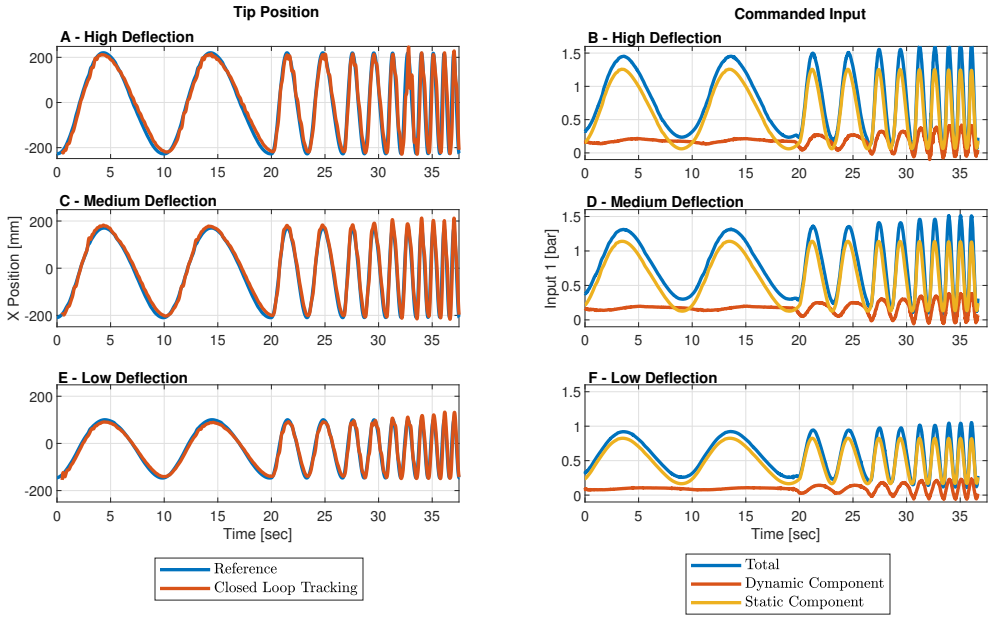


Figure 3.2: **Closed-loop reference tracking experiments show the soft arm's ability to track in real-time**. The soft arm tracked circular reference trajectories in the X-Y plane with frequencies ranging from 0.1 to 1.1 Hz (0.2 Hz step) at: (**A**-**B**) high, (**C**-**D**) medium, and (**E**-**F**) low deflections. Plots **A**, **C**, and **E** show the X positions (red) over time compared to their respective references (blue). The Y and Z positions are shown in Supplementary Fig. 3.11. Plots **B**, **D**, and **F** show the relative contributions of the static Koopman pregain (yellow) and dynamic Koopman LQR (red) to the total input (blue). At quasi-static speeds, only the static Koopman pregain is required for effective performance (i.e. the quantity $\mathbf{x}-\mathbf{x}_{\text{ref}}$ is approximately zero); as inertial effects increase, the LQR component increases its contribution to maintain performance. Only the commanded inputs to one of the four side muscles is shown, but the results are similar for all muscles.

3.2.3 Closed-loop, real-time reference-tracking in inertial, nonlinear regime

We next sought to characterize the controller performance for a less structured and more challenging control objective: tracking a real-time, user-defined reference. To do so, we commanded the controller to decrease the euclidean distance between the tip of the soft arm and a motion tracker point located on the tip of a pole. A human operator moved the pole across random trajectories within the reachable workspace of the soft arm, including both slow and rapid motions. Throughout the test, the robot remains in contact almost continuously while achieving speeds exceeding 0.7m/s (as shown in Fig. 3.3 and Videos S1 and S4).

To demonstrate the generalizability of our approach for different soft arms, we also tested our approach on a morphologically different second arm. The second arm is longer, more slender, and has three instead of four side muscle. This results in larger curvatures and a helical actuation pattern, as discussed in Section 3.3.4. Despite these differences, no changes were needed in the learning and control algorithm, aside from updating the number of inputs. This second system was exposed to 5 minutes of step input training data, the model and controller were calculated and deployed, and the system was commanded to again track the tip of the user-operated pole. Results of this test are shown in supplementary materials (Video S5), and stills from the testing are shown in Fig. 3.1.

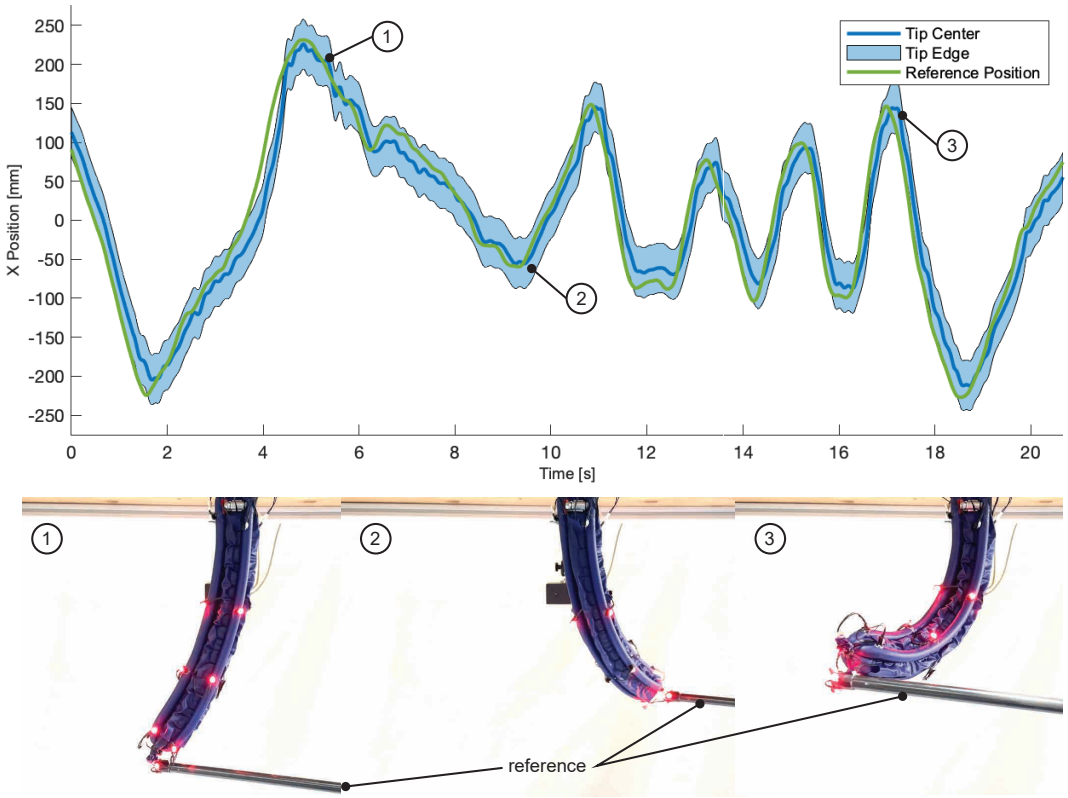


Figure 3.3: **Performance of our controller subjected to a series of arbitrary reference positions spanning the high-deflection workspace.** (**A**) The X position of the soft arm tip is shown as it tracks a moving reference commanded randomly by an operator. Contact between the green lines and blue band indicates points where the soft arm is touching the reference marker. (**1-3**) show images of the soft arm performing this behavior. Of note, the robot rarely loses contact with the moving reference.

3.2.4 Dynamic Throwing and Catching

With the viability of our method shown in the above characterization tests, we finally demonstrated how its capabilities translate to sample robotic tasks. Inspired by the demonstrations of a two-link arm [74], we challenged our soft continuum arm in two ways: (i) to catch a ball swinging through the air as we demonstrate in Fig. 3.4, and (ii) to receive an object from an operator, and to throw it into a reference bin as shown in Fig. 3.5. Both tests are shown in Video S3. This demonstration is similar to the ball

catching performed in [74] by a two-link arm with a soft joint. To our knowledge, our work is the first demonstration of throwing and ball catching in a fully soft continuum robot arm.

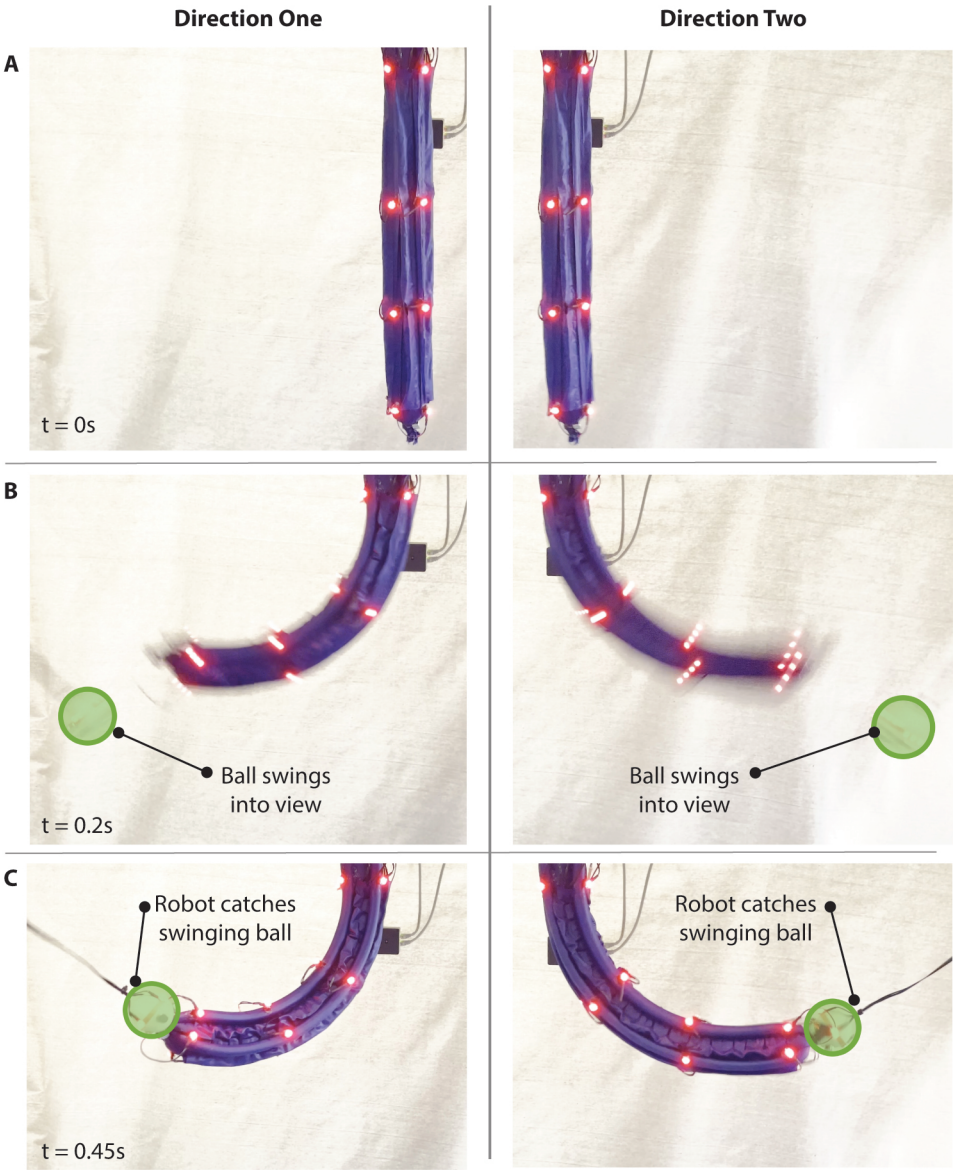


Figure 3.4: **Performance of our controller when attempting to minimize error on an arbitrary trajectory (catching a swinging ball).** (A) The soft arm stays in the neutral position while the ball is outside the workspace. (B) Once visible, the soft arm rapidly responds to reach the ball (outlined swinging into the workspace). (C) The soft arm tip intercepts the ball and catches it (with small magnets on both the soft arm tip and swinging ball.)

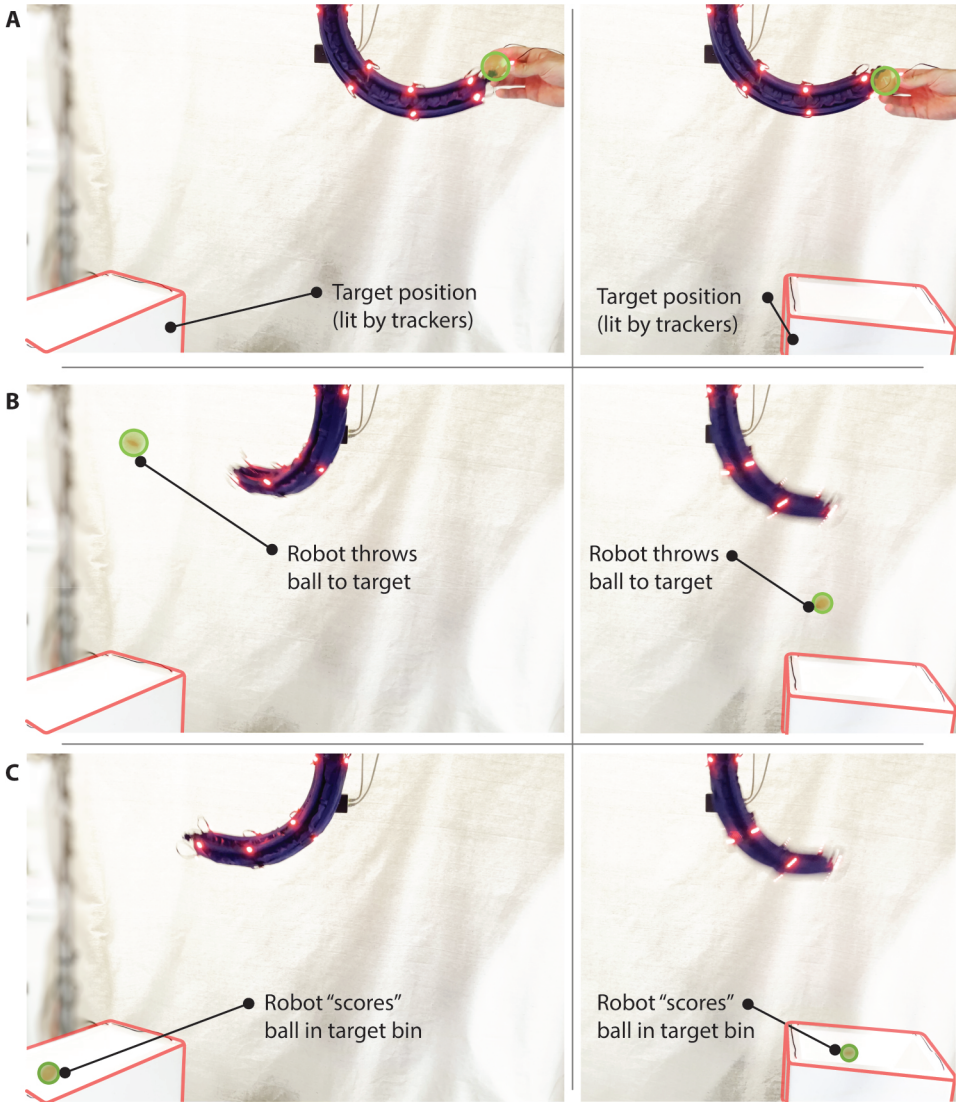


Figure 3.5: **Demonstration of implications of the developed methodology: completing example tasks** (A) The soft arm identifies the objective and approaches it (operator's hand). (B) After the operator's hand is removed and the ball is supported by the soft arm, the objective changes to the bin (lighted bin in bottom left and right, respectively). The soft arm now flings the ball at the objective. (C) The ball successfully enters the bin in two different, arbitrary locations, achievable only by working in the inertial regime.

3.3 Materials and Methods

Here, we first introduce 1.1, the mathematical underpinning of our modeling effort. In Section 1.6.3, we describe a practical method to build the model for our control system from data. In Section 3.3.1 we describe how this model is embedded into a real-time feedback controller. Our modeling and control insight is the addition of a static Koopman operator pregain described in Section 3.3.2. The design of our soft arms are given in Section 3.3.4 and their fabrication procedure is given in Section 3.3.5. The design of the pneumatic circuit is explained in Section 3.3.6. A block diagram detailing the full training process, modeling, and control architecture is given in Supplementary Fig. 3.12.

3.3.1 Koopman-LQR (K-LQR)

To date, similar investigations have used model predictive control (MPC) to control their soft robotic systems [62, 63, 64]. Using predictions of the dynamics and a tunable prediction horizon, this architecture calculates input sequences which move the system toward a desired reference position. This enables the use of explicit input and state constraints, but the real-time constrained optimizations involved in this method demand a high computational overhead.

In our inertial soft arm controller, explicit constraints are less important than keeping computational cost and latency low. For unconstrained linear optimal control problems with quadratic cost, the linear quadratic regulator (LQR) provides an analytical solution which does not require predictions of the dynamics in real time [72]. For our controller, we begin with the application of LQR to the dynamic Koopman representation of a dynamical system (previously demonstrated for a robotic fish [69]), and augment it via the introduction of the static Koopman term, described in Section 3.3.2.

Here we describe the dynamic Koopman LQR control law. While originally intro-

duced for linear dynamical systems in state space, LQR can also be applied to a vector of observables $\mathbf{z}$ of a nonlinear control system as long as a linear, finite dimensional representation of the Koopman operator (A, B) exists. Given the system

$$\mathbf{z}^+ = A\mathbf{z} + B\mathbf{u} \tag{3.1}$$

$$\mathbf{x} = C\mathbf{z}, \tag{3.2}$$

we define the global cost function

$$J = \sum_{i=1}^{m} \left[(\mathbf{z}_i - \mathbf{z}_{\text{ref}})^T Q (\mathbf{z}_i - \mathbf{z}_{\text{ref}}) + \mathbf{u}_i^T R \mathbf{u}_i \right] \tag{3.3}$$

where $\mathbf{x}_{\text{ref}} = C\mathbf{z}_{\text{ref}}$ is the desired position and Q and R are diagonal lifted state and input penalty matrices, respectively.

The computation of the minimizing control input is a classical method in optimal control [72] and is given by $\mathbf{u}_i = -K(\mathbf{z}_i - \mathbf{z}_{\text{ref}})$ where the matrix K is the LQR gain. This control law results in steady state errors in much of the soft arm's workspace. This is remedied in the next section by the addition of a pregain term based on the static Koopman operator.

3.3.2 Static Koopman Pregain

Unfortunately, Dynamic Koopman LQR alone resulted in significant disagreement between reference positions and the resulting states. This is because the nonzero inputs required to hold these positions result in a nonzero input penalty term. Any attempt to decrease the input penalty resulted in system instability. The addition of a pregain term is a classical method in control theory that addresses this problem. In this section, we introduce a data-driven method to compute the pregain using a static Koopman operator,

which we term the static Koopman pregain.

A core assumption of this component of our model is that when held for enough time, all transient dynamics dissipate, and the robot achieves a static pose. Therefore, the set of admissible step inputs $\mathbf{u}_{\text{static}}$ corresponds to a set of input-mediated fixed points $\mathbf{x}_{\text{static}}$. We seek a mapping from the data matrix of step inputs, U_{static}, to the data matrix of stationary states, X_{static}. Ideally, this mapping would be linear to enable us to use fast, optimal control. The Koopman framework usually requires the domain and range to be the same, but this requirement can be relaxed if we consider the static Koopman operator [73]. The static Koopman operator contrasts with the dynamic Koopman operator, which describes the evolution of observables $f : \mathcal{M} \to \mathbb{R}$ under the action of the mapping $T : \mathcal{M} \to \mathcal{M}$. If we define observables on the inputs as $g : \mathbb{R}^p \to \mathbb{R}$, the static Koopman operator $\mathcal{K}_{\text{stat}}$ is defined as

$$\mathcal{K}_{\text{stat}} f(\mathbf{x}_{\text{stat}}) = g(\mathbf{u}_{\text{stat}}).$$

We desire to approximate the action of the static Koopman operator with a finite dimensional matrix G. To do so, we first construct the data matrix U_{static} with unique step inputs as the columns of the matrix. By feeding these inputs to the system and allowing transient dynamics to dissipate, we are left with a unique stationary state, $\mathbf{x}_{\text{static}}$; these states represent the columns of X_{static}. We then lift the stationary states $\mathbf{z}_{\text{static}} = \mathbf{f}(\mathbf{x}_{\text{static}})$ and inputs $\mathbf{v}_{\text{static}} = \mathbf{g}(\mathbf{u}_{\text{static}})$ and build the corresponding lifted stationary data Z_{lift} and input V_{lift} matrices. The matrix G is then computed using

$$G = V_{\text{static}} Z^{\dagger}_{\text{static}}. \tag{3.4}$$

In our implementation, we take $\mathbf{v}_{\text{static}} = \mathbf{g}(\mathbf{u}_{\text{static}}) = \mathbf{u}$, so $V_{\text{lift}} = U_{\text{lift}}$. The matrix G serves as a linear mapping from lifted stationary states to lifted inputs.

Finally, we are ready to bias our control law with the addition of a feedforward pregain term $G\mathbf{z}_{\text{ref}}$, resulting in

$$\begin{aligned} \mathbf{u}_i &= -K(\mathbf{z}_i - \mathbf{z}_{\text{ref}}) + G\mathbf{z}_{\text{ref}}. \\ \mathbf{z}_{i+1} &= A\mathbf{z}_i + B\mathbf{u}_i \\ \mathbf{x}_i &= C\mathbf{z}_i. \end{aligned} \tag{3.5}$$

This signal is the optimal stabilizing solution taking the present initial state to the desired state, $\mathbf{x}_{\text{ref}}$.

As shown in Fig. 3.2, the pregain term $\mathbf{u}_{\text{stat}} = G\mathbf{z}_{\text{ref}}$ outweighs the dynamic term $\mathbf{u}_{\text{dyn}} = -K(\mathbf{z}_i - \mathbf{z}_{\text{ref}})$ in most tests. This allows the input penalty weights in the dynamic term to be optimized without fear of sacrificing steady-state error. Also, the static Koopman term provides enough of a steady input to counter the fluctuations caused by measurement noise introduced by the state measurements in the dynamic term. This is the reason our system does not experience the destabilizing effects of noise in fast-moving reference tests described in [44].

3.3.3 Training and Observables

With the mathematical underpinning of our modeling and control methodology described (see Supplementary Fig. 3.12), we now turn to the particular choices made to suit our particular robotic applications. Given the soft arms described in Section 3.3.4 we collect training data through a series of experiments, performed by commanding step inputs with randomly distributed magnitudes. The only prior knowledge of the soft arm's dynamics required is an upper bound for the length of time required for the dissipative dynamics to die down while inputs are held. Each step input is held for this amount of time so that the soft arm converges to a steady state, efficiently probing both the dynamic and static response. The data is separated into training and validation sets,

and the training data is further partitioned into dynamic and static components which are used to train dynamic and static Koopman operators (see Sec. 3.2.1).

Choosing observables is difficult in practice. We choose to implement DMDc with time delay observables (also known as Hankel DMDc) because of their provable convergence as the number of time delays goes to infinity under certain assumptions on the dynamics [61, 60, 18]. In reality, adding more time delays gives a diminishing return in prediction accuracy (see Fig. 3.7A). A single time delay with hundreds of monomials is used in [64, 63, 62, 70], but we find that time-delay-only observables offer better results, with improvements in reconstruction with up to ten observables (See Fig. 3.7A). To create our observables, we use the current measurement of the X-Y-Z positions of the motion trackers $\mathbf{x}_i$ and append two time-delayed versions of the same states $\mathbf{z}_i = [\mathbf{x}_i\ \mathbf{x}_{i-1}\ \mathbf{x}_{i-2}]^T$. Each time delay looks 1/60 seconds into the past. This proves to be sufficient for closed-loop control. For reconstruction, more time delays give further increases to the model's accuracy, as shown in Fig. 3.7A.

The synergy of step inputs and time delays allows the discovery of system eigenvalues in the important 1 to 5Hz range (the span of natural frequencies of the arm), as shown in Fig. 3.6. Without time delays, these eigenvalues and their corresponding Koopman modes are missed (Fig. 3.6). For comparison to the Koopman model used in [70], we tested the addition of monomial observables was tested up to order 4 with no new dynamic modes of any significant mode power learned. Monomial observables also failed to give any improvement to the reconstruction or closed-loop pointing accuracy of the model and controller (Fig. 3.7).

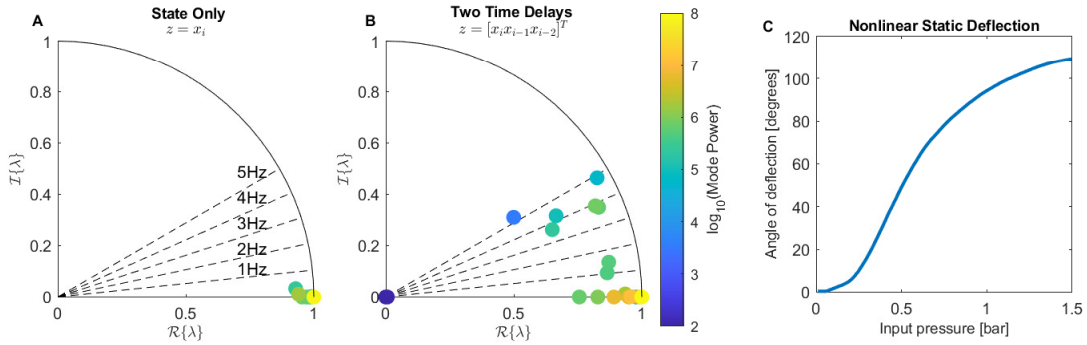


Figure 3.6: **Nonlinear and Inertial Dynamics of the Soft Arm.** The eigenvalue plots for Koopman models with state only (**A**) and state plus time delay observables (**B**) are shown. The dashed radial lines signify sections of the unit circle corresponding to modes with $1 - 5$Hz dynamics. The eigenvalues are shaded corresponding to the logarithm of their maximum achieved mode power evaluated over the training data (see 1.6.2). Using state-only observables results in a simple linearized model which does not capture any transient dynamics. The addition of two time delay observables allows the modeling of dynamics up to 5Hz. This is the model we choose for our experiments. (**C**) Presentation of the input-output nonlinearty of the system, which exhibits a sigmoidal deflection response. Modeling this nonlinearity is essential for acceptable reference tracking performance in the high-deflection regime.

With the goal of minimizing training time and model complexity, we found that up to five time delays and one minute of step input training is best for modeling our system before considering control, but only two time delays and five minutes of step input training is ideal when control is considered. We first compared the prediction ability of different dynamic Koopman models as we varied the number of time delays and total training time (Fig. 3.7A). The addition of a single time delay substantially reduced error, however additional time delays continued to offer marginal improvements up to five delays. We also found that after only approximately one minute of training, the model reached its minimum error. Second, we built Koopman-LQR controllers as described in Section 3.3.1, augmented with a static Koopman operator as a pregain term, with varied time delays and training time. We then quantified the error with

closed-loop control (Fig. 3.7B). In this case, two time delays outperformed one delay, but was comparable to three or more, resulting in our decision to use two delays for control. We also found that after approximately five minutes of training (fifty unique step inputs), the error converged; we used this amount of training time for the remaining experiments. Note: a direct linearization of the system was unstable during controlled motions, suggesting the nonlinearity of the system.

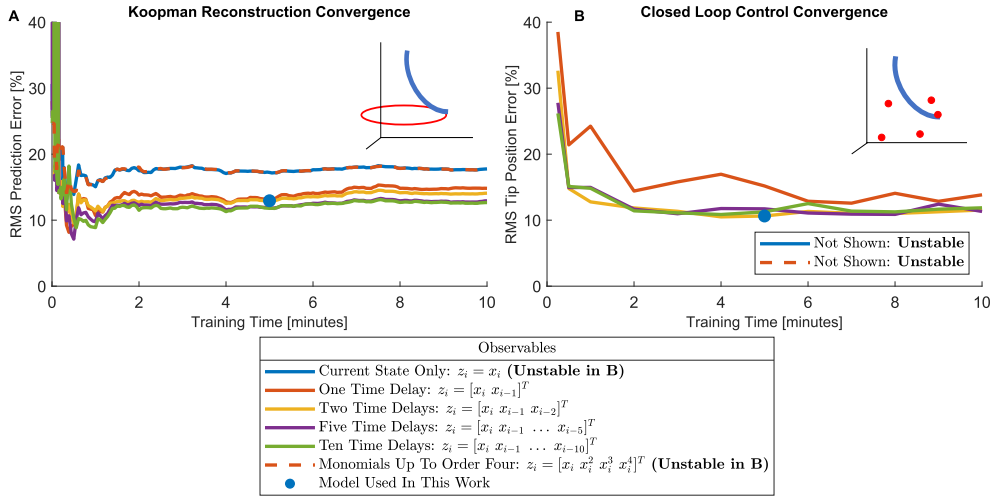


Figure 3.7: **Convergence of the Koopman model and control system.** (**A**) The dynamic Koopman model requires the addition of five time delay observables and only one minute of training data to reach minimum prediction error. To determine this error, the single-step prediction error of the dynamic Koopman model is collected for all points as the soft arm moves on a circular path in the X-Y plane (inset), and the root-mean-square (RMS) average is taken. For comparison, a Koopman model using monomials of the state up to order four gives no improvement over the state only model. This reconstruction is performed on a model trained on zero sinusoidal trajectories. (**B**) In closed-loop control, the combination static/dynamic Koopman controller requires only five minutes of training data and two time delays to reach minimum prediction error; accordingly, we use this controller design for every experiment. Each controller was commanded to move the soft arm's tip to a sequence of points in the workspace of the soft arm, and the average RMS error for all these points was calculated. Using zero time delays resulted in the soft arm being unable to stabilize at any reference position, so that line is not shown.

3.3.4 Robot Design

For this investigation, we constructed two distinct soft arms to evaluate the viability of the proposed methodology across nonlinear dynamical systems. For each, we aimed to meet the following objectives: a) high-deflection, nonlinear dynamics for which linearization fails; b) inertial dynamics, for which quasi-static approximations fail; c) enough morphological diversity such that their analytical models would be not readily transferrable.

To this end, the first arm was designed to have four actuators (two antagonistic pairs) longitudinally aligned with the main body to produce planar actuation. This design is behaviorally similar to others present in the literature ([75, 48, 49]). When fabricated with appropriate pretension, this construction allows for approximately 110° of curvature when fully actuated. With a length of 45 cm and a maximum diameter (main body diameter plus the diameter of the fully inflated muscles) of 6.25 cm, the slenderness ratio of this device was 7.2 (the ratio of length to max diameter).

The second arm was designed with three actuators, all of which were affixed to the body such that a torsional deflection would be induced when inflated. This produces a helical actuation that is markedly different from that of the first embodiment. With a length of 53 cm and a maximum diameter of 3.8 cm, this device exhibited a slenderness ratio of 13.9. The muscles were affixed with pretensions such that, when fully actuated, this device is capable of achieving approximately 180° of curvature.

For objective a), with an angle of curvature of at least 110° for both arms, the nonlinearity metric is well achieved (See Fig. 3.6). For objective b), both systems were fabricated out of airtight fabric, utilizing fabric pneumatic artificial muscles (fPAMs) as described in [76], which exhibit a fast response time and low hystersis (on the order of 1%), achieving accelerations in excess of g. For c), the factor of approximately two

difference in slenderness ratios, the change in actuator numbers, and the inclusion of helical actuation all combine to produce two systems with significantly different behavior (see, for example, the model presented in [77] compared to [46]).

3.3.5 Robot Fabrication

Both arms were constructed out of 30 Denier silicone-polyurethane impregnated ripstop nylon (Sil-nylon, Rockywoods Fabrics), actuated by fabric pneumatic artificial muscles (fPAMs) [76] built out of the same material. The main body was fabricated such that one side of the fabric weave cell was parallel to the longitudinal axis, the other perpendicular. This orientation makes the soft arm axially and transversely stiff, but torsionally compliant. The muscles were fabricated such that each side of the cell was offset by approximately 45° with respect to the longitudinal axis, which instead makes the actuator torsionally stiff but compliant axially and transversely. Moreover, when these muscles are inflated, they shorten in the longitudinal direction as a McKibben does, up to 35% based on the pretensioning induced during adhesion to the main body.

Each of these components was cut from a sheet of fabric, rolled into a tube, and sealed with a lap joint using RTV silicone adhesive (Smooth-on Silpoxy). Once each component was fashioned, a jig was produced to hold the main body and pretensioned muscles in place while the RTV cured. Finally, in between each muscle a fabric sleeve, exhibiting the same fabric bias as the muscles, was attached to the main body to allow for motion capture tracker wires to be routed without occluding the view of the LEDs.

3.3.6 Pneumatic Circuit Design

Each soft arm body was held at a constant pressure of approximately 1 bar for the entirety of testing, supplied by a discrete source. For each muscle of both soft arms,

Festo VEAB-L-26-D2-Q4-V1-1R1 proportional pressure valves were used to command individual pressures continuously. These three-port valves were chosen for three reasons: their fast response times (<10ms); accurate response (0.75% full-scale absolute accuracy, 0.4% full-scale repeatability error); and the ability to accept forced exhaust through their third port. However, this accuracy requires a lower flow rate, which precluded the use in the much larger main body (due primarily to persistent leaks). Additional information on the general control circuitry configuration can be found in the Supplemental Information.

3.A Supplementary Materials

3.A.1 Testing Apparatus

The experiments conducted in the framework of the study are performed on an experimental system shown in Fig. 3.8.

A 1.8x1.8x1.5 m 80/20 frame was assembled to support the testing apparatus. To this frame the motion capture cameras, soft arm, and control hardware were affixed. Information about the position and shape of the manipulator is gathered via motion capture (PhaseSpace Inc. Impulse X2E). This investigation utilized the motion capture system with 8 detectors (cameras) and 4 sets of trackers evenly spaced along the backbone of the soft arm; four LEDs are attached along the axis of each muscle. The same motion capture system was used for both data collection used in offline model construction and for closed-loop position feedback in control experiments. In closed-loop experiments that are performed without predefined trajectory, additional four LED trackers are mounted on an external object (a pole), their coordinates are averaged in real-time in order to determine the central point, which then served as an arbitrary reference generator.

Festo VEAB-L-26-D2-Q4-V1-1R1 proportional pressure regulators with 0.01 to 2 bar

output range and approximately 15 liters/min of flow at 1 bar pressure, are used to control the pressure in the arm's muscles. The body is held to a constant pressure of approximately 1.5 bar

The software used for running the system was LabVIEW 2019 with myRIO toolkit and real-time module, while LabVIEW Python node is used to acquire the real-time data from motion capture system. These information is then fed trough the fast network protocol to a myRIO 1900 control hardware. The same control hardware is also used to drive the pressure valves whereas an additional circuitry based on operational amplifiers is used to adjust 0-5V voltage levels generated by MyRIO hardware to be compatible with used proportional valves whose input range is 0-10V. Exhaust air ports of the valves are connected to vacuum so as to improve the dynamical response of the system.

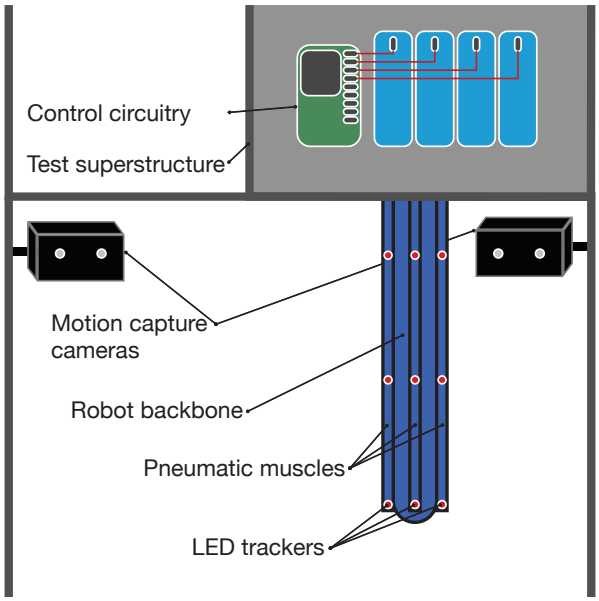


Figure 3.8: **Schematic representation of the experimental setup and its components.** The soft arm is mounted from above to the test superstructure. The soft arm backbone provides stiffness and the pneumatic muscles generate movement. Four layers of LED trackers are tracked by motion capture cameras positions around the arm.

3.A.2 Model Performance Metrics

We perform a convergence study on the reconstruction power of our Koopman models as a function of the number of snapshots for a range of observables. This process allowed us to develop a dictionary of observables suitable for our system. Given a particular choice of observables and number of training samples, we build the corresponding linear input-output system with A, B, and C matrices. This linear model is applied to $N = 8000$ samples of sinusoidal verification data over a range of deflection amplitudes and speeds. These particular samples are not included in the training data in order to give us a fair evaluation of the predictive power of our models. The linear system produced via (1.39) and (1.32) evaluate the evolution of these initial conditions over a single time step. The single-step reconstruction error is given by

$$e_i = \frac{\|x_i^{+,\text{ predict}} - x_i^{+,\text{ actual}}\|_2}{L}.$$

where $x_i^{+,\text{ actual}}$ is the evolution of x_i measured by the motion capture system, $x_i^{+,\text{ predict}}$ is the evolution predicted by the DMD model, and L is the length of the soft arm. We use the root mean square (RMS) of the individual e_i errors to score our model:

$$e_{\text{RMS}} = 100\sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}.$$

3.A.3 Reconstruction of sinusoidally forced motion

After choosing time delay observables, we attempted to reconstruct the movement of the soft arm under a sinusoidal inputs with six different frequencies ($0.1, 0.2, 0.4, 0.8, 1$, and 1.1Hz). We begin this process by providing a step input to the muscles that corresponds to a static position on the sinusoid with an arbitrary phase, and then complete

two revolutions at a given frequency before incrementing up in speed. The position of the physical system was recorded via motion capture system, and these inputs were provided to our above-developed model. This process was repeated for low (similar to [70, 62, 63], approximately 15°), medium (similar to [46, 47], approximately 25°), and high deflection (the single-actuator maximum of our system (robot 1), approximately 110°), with results reported in Fig. 3.9.

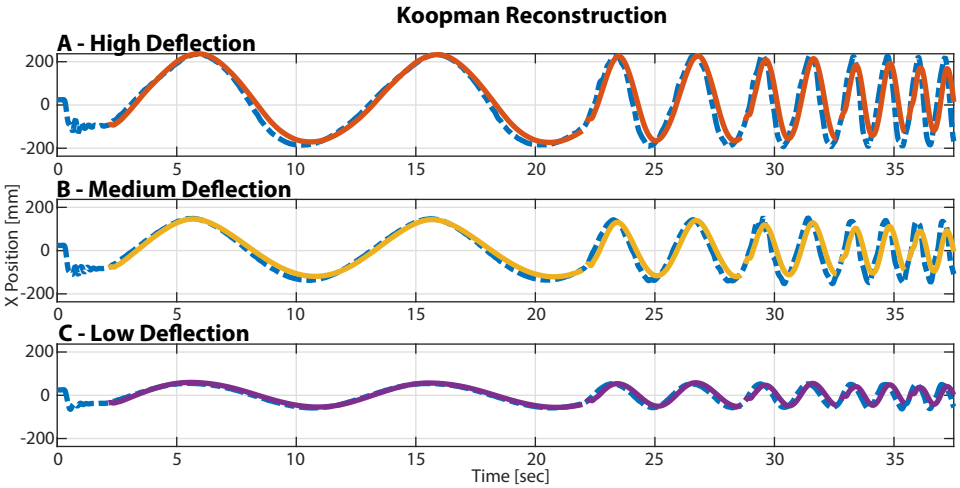


Figure 3.9: **Koopman reconstruction of circular motion.** The dynamic Koopman model is given a collection of sinusoidal inputs with a range of amplitudes and speeds and is tasked with reconstructing the motion of the soft arm. The true trajectories are shown in dashed blue, and the high (**A**), medium (**B**), and low (**C**) deflection reconstructions are given in red, yellow, and purple, respectively. The reconstruction is restarted every time the frequency changes. The reconstruction agrees with the true frequency, but is missing some of the amplitude in the fast regime. The static Koopman operator and feedback control account for the improvement in performance between this plot and Fig. 3.2

3.A.4 Disturbance Rejection

Finally, to evaluate the disturbance rejection capabilities of our system and to further distinguish the contribution of the static Koopman pregain, G, and the dynamic Koopman LQR gain, K, we commanded both stationary and circular references for the soft arm tip and subjected the system to disturbances. The control effort was recorded

and compared to the control effort expected from the pregain term alone. Given a static reference, the control effort from the pregain alone is constant in time. The results of the tests are shown in Fig. 3.10, clearly capturing the contribution of K, proportional to the disturbance.

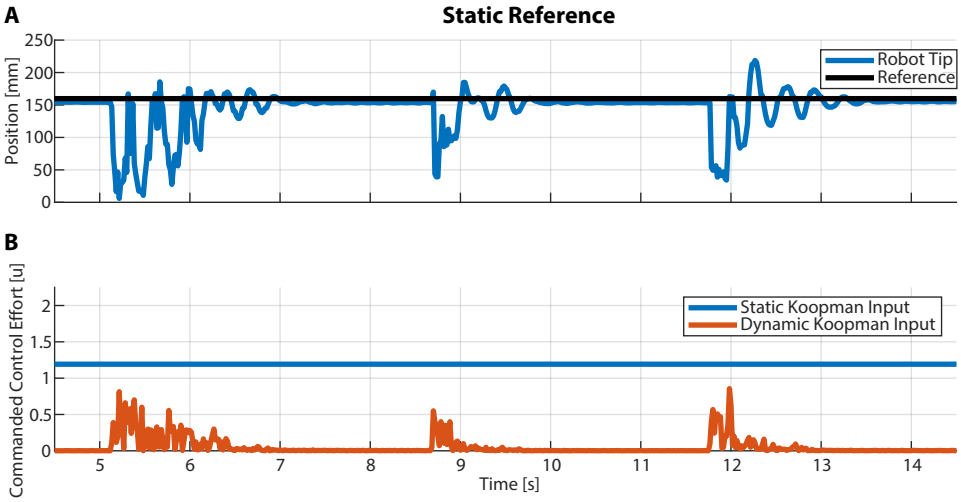


Figure 3.10: **Demonstration of our controller's ability to reject impulse disturbances in real-time. A)** The position over time of the end effector is shown relative to a predefined reference. The soft arm returns to the reference position after three large disturbances are applied. **B)** The magnitude of the commanded control effort is shown. Note that the static Koopman input component comes from the pregain term and is constant because the static reference doesn't change. The dynamic Koopman input adapts in real time to the disturbances.

3.B Supplemental Figures

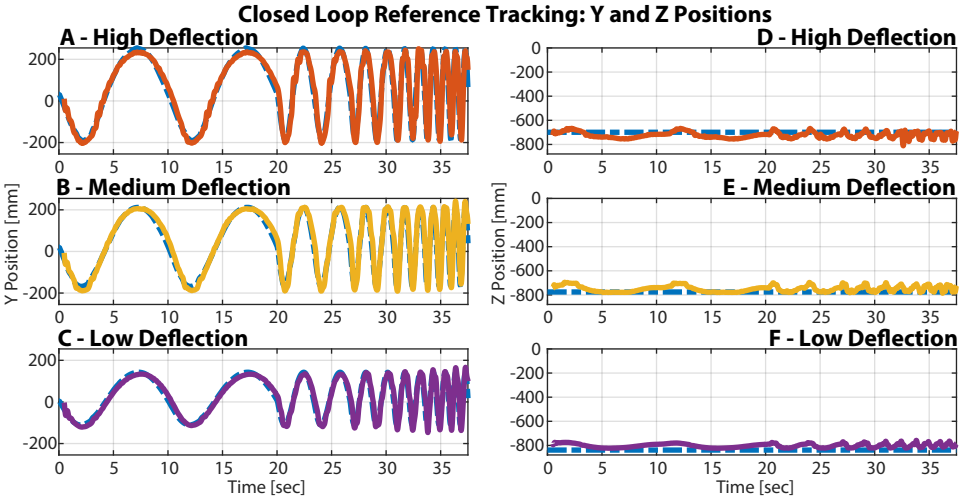


Figure 3.11: **Y and Z components of the real-time closed-loop reference tracking experiments**. The soft arm tracked circular reference trajectories in the X-Y plane with frequencies ranging from 0.1 to 1.1 Hz (0.2Hz step) at: (**A**,**D**) high, (**B**,**E**) medium, and (**C**,**F**) low deflection magnitudes. Plots show the Y (left column) and Z (right column) positions over time compared to their respective references. The commanded references are dashed lines and the control results are solid lines.

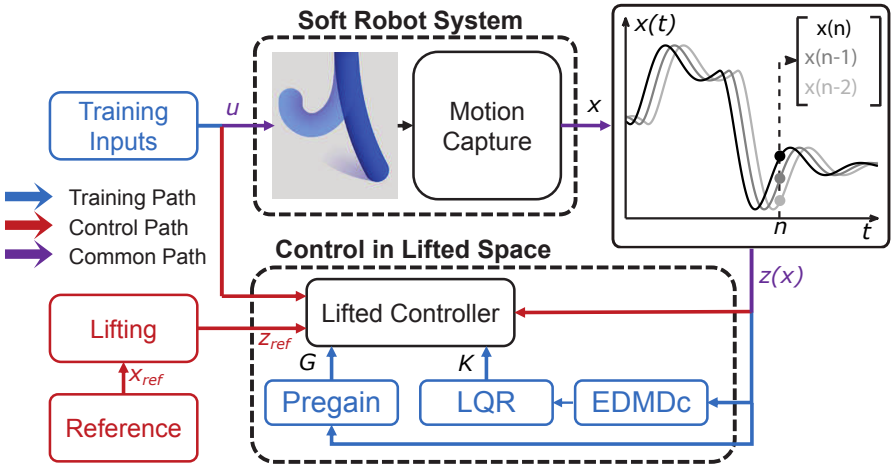


Figure 3.12: **Block diagram of the system, training method and K-LQR control approach.** Training inputs representing voltage signals are fed into pressure valves and 3D positions of the soft arm are measured by using a motion capture system. The lifting procedure of the position data provides the inputs needed to determine the Koopman model of the system and to calculate optimal lifted controller parameters. The position of the soft arm is finally controlled in 3D space by using obtained K-LQR.

Chapter 4

Input-Parameterized Koopman Eigenfunctions

This work applies the theory of nonlinear representations of the Koopman operator to control problems. This work was advised by professor Igor Mezić.

4.1 Introduction

In order to use the EDMDc-based Koopman approximation methods introduced in Section 1.6.3, we require the system to admit a finite dimensional control term that is linear or bilinear in the eigenfunction coordinates. This restricts these methods to systems which are time-discretizations of continuous-time control-affine systems [78], but this representation can still be necessarily infinite dimensional. The requirement of finite dimensionality puts further restrictions on the dynamics.

There are situations where finite dimensional bilinear Koopman representations do not exist or are unknown. For example, control-induced switching between multiple basins of attraction in the control-affine Hamiltonian setting is addressed in [11] using Koopman eigenfunctions, but only in a piecewise manner. Koopman models have also been applied to switching control systems [79] where a small number of discrete input choices is used to define multiple, distinct Koopman operators. For bistable monotone systems, an alternative approach is presented in [80, 81]. Here, the system is subjected to temporal pulses, and the resulting trajectories are used to compute isostables. These are level sets of Koopman eigenfunctions that contain information about the time required for the system to pass within a certain distance of a stable fixed point.

The idea that input-driven systems generate new eigenfunctions was also proposed in [58] but not used. We expand on this idea and show that these input-parameterized Koopman eigenfunctions and their respective eigenvalues can be used to construct novel representations of the Koopman operator. These are examples of the nonlinear representations of the Koopman operator proposed in [6]. These representations can sometimes

be finite dimensional when finite dimensional linear representations are impossible, but the structure of the model needs to be carefully chosen to allow for interpretability and control. We show that the nonlinear input-parameterized eigenfunction representation we propose is useful in the construction of control algorithms beyond the monotone systems of [80, 81].

To compute this representation, we train on trajectories in which the system in question is subjected to a constant input, held for all time. From this perspective, each choice of input generates a new autonomous system, from which we extract the eigenfunction information using Laplace analysis. From here, control design requires only inspection of the eigenfunctions. This representation gives a globally accurate predictor for systems which experience change in fixed point location and can model the transition between basins of attraction caused by inputs.

Our proposed control algorithms are given in Section 4.2, and their implementation on the Duffing oscillator with dissipation is shown in Section 4.3.

4.1.1 Input-Parameterized Eigenfunctions

We wish to retain the benefits of the linear eigenfunction representation $(\boldsymbol{\phi}(\mathbf{x}), \Lambda^t)$ of a nonlinear dynamical system for systems with input $\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}(t))$, without restricting to systems with linear or bilinear input dependence. Here the input is of the form $\mathbf{u}(t) : \mathbb{R}^+ \to \mathbb{C}^p$. To do this, we decompose our dynamical system into a series of systems $\dot{\mathbf{x}} = F_{\mathbf{u}}(\mathbf{x})$ parameterized by constant inputs, held for all time $\mathbf{u}(t) = \mathbf{u}$. Each system $F_{\mathbf{u}}$ has flow $S^t_{\mathbf{u}}$.

Definition 16 *The input-parameterized Koopman operator $U^t_{\mathbf{u}}$ gives the evolution of ob-*

servables f under the action of the dynamical system $F_{\mathbf{u}}$ according to the rule

$$U_{\mathbf{u}}^t f = f \circ S_{\mathbf{u}}^t. \tag{4.1}$$

If $\dot{\mathbf{x}} = F_{\mathbf{u}}(\mathbf{x})$ is a discrete spectrum system for all choices of $\mathbf{u}$, then we can define the input-parameterized eigenfunctions. Note that this will not always be the case. A classic example is when the stability parameter of the Lorenz system is treated as an input. Certain choices of this parameter cause the system to become chaotic, in which case no eigenfunctions exist.

Definition 17 *An input-parameterized eigenfunction $\phi_{\mathbf{u}}(\mathbf{x})$ with input-parameterized eigenvalue $\lambda(\mathbf{u})$ is an observable whose Koopman evolution is given by*

$$U_{\mathbf{u}}^t \phi_{\mathbf{u}}(\mathbf{x}) = e^{\lambda(\mathbf{u})t} \phi_{\mathbf{u}}(\mathbf{x}). \tag{4.2}$$

Define the vector of input-parameterized eigenfunctions $\boldsymbol{\phi}_{\mathbf{u}}(\mathbf{x}) = [\phi_{\mathbf{u},1}(\mathbf{x}) \dots \phi_{\mathbf{u},n}(\mathbf{x})]^T$, the diagonal matrix of eigenvalues

$$\Lambda_{\mathbf{u}} = \begin{bmatrix} \lambda_1(\mathbf{u}) & & \\ & \ddots & \\ & & \lambda_n(\mathbf{u}) \end{bmatrix}, \tag{4.3}$$

and the mapping $\Lambda_{\mathbf{u}}^t(\boldsymbol{\phi}) = e^{\Lambda_{\mathbf{u}} t}\boldsymbol{\phi}$ form an input-parameterized collection of eigenfunction representations of the Koopman operator $(\boldsymbol{\phi}_{\mathbf{u}}, \Lambda_{\mathbf{u}}^t)$.

This linear representation of the Koopman operator predicts the evolution of this system under any constant input. In order to include arbitrary inputs $\mathbf{u}(t)$, we need to construct a nonlinear representation. For simplicity, we will assume that all inputs are piecewise constant in time. This assumption allows us to consider any input that can be

feasibly stored on a computer since it will most likely be stored in discrete time. Take an input $\mathbf{u}(t)$ which is piecewise constant and changes value at the times t_i for $i = 1, \ldots, k$ and define the following coordinate change operator.

Definition 18 *Given input-parameterized eigenfunction coordinates* $\mathbf{y} = \boldsymbol{\phi}_{\mathbf{u}(t_i)}$ *and* $\mathbf{y}^+ = \boldsymbol{\phi}_{\mathbf{u}(t_{i+1})}$*, the coordinate change operator is given by*

$$\mathcal{G}_{t_i}(\cdot) = \mathbf{y}^+(\mathbf{y}^{-1}(\cdot)). \tag{4.4}$$

Assuming invertible eigenfunctions for all $\mathbf{u}$, this operator transforms out of the coordinate system defined by the eigenfunctions generated by $\mathbf{u}(t_i)$ and into those generated by $\mathbf{u}(t_{i+1})$. If we now define the evolution mapping

$$\mathcal{F}^{t_k}(\boldsymbol{\phi}(\mathbf{x}(t_0)), \mathbf{u}(t)) = \mathcal{G}_{t_{k-1}}(\ldots(\Lambda^{t_2-t_1}_{\mathbf{u}(t_1)}\mathcal{G}_{t_0}(\Lambda^{t_1-t_0}_{\mathbf{u}(t_0)}\boldsymbol{\phi}_{\mathbf{u}(t_0)}(\mathbf{x}(t_0))))\ldots), \tag{4.5}$$

then the nonlinear Koopman representation $(\boldsymbol{\phi}_{\mathbf{u}}, \mathcal{F}^{t_k})$ is a predictor for arbitrary piecewise continuous input sequences $\mathbf{u}(t)$. An illustration of this predictor is given in Figure 4.1.

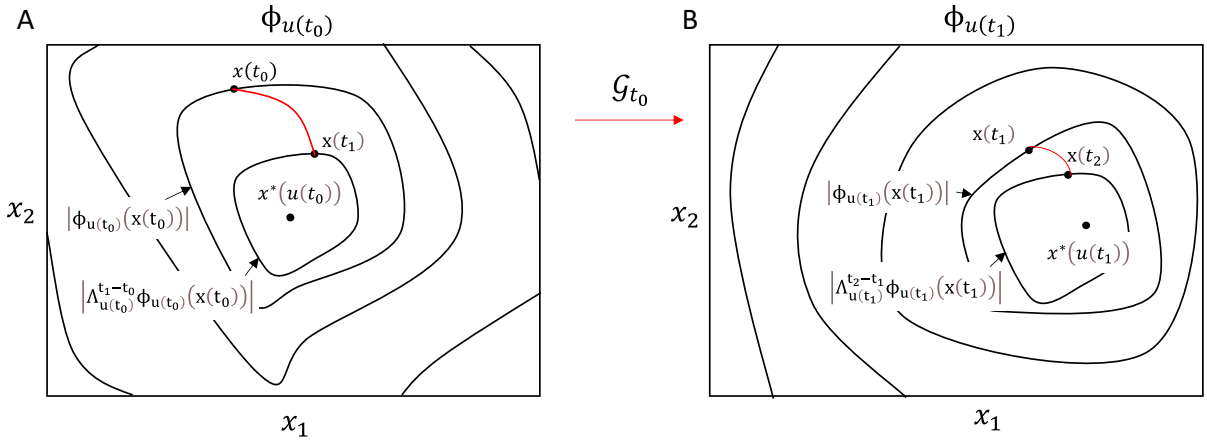


Figure 4.1: This figure shows the evolution of $\mathcal{F}^{t_0}$ given an initial state $\mathbf{x}(t_0)$ and input sequence $\mathbf{u}(t) = \begin{cases} \mathbf{u}(t_0) & t_0 \leq t < t_1 \\ \mathbf{u}(t_1) & t_1 \leq t < t_2 \end{cases}$. The dynamical system is chosen to have an attractive, stable fixed point which is a function of the input $\mathbf{x}^*(\mathbf{u})$. The eigenfunction for all choices of $\mathbf{u}$ is chosen to be the Laplace average (Eq. 1.22). The level sets of ϕ are the isostables and are shown as concentric contour lines. In Pane A, we show the linear evolution of the dynamics from $\mathbf{x}(t_0)$ to $\mathbf{x}(t_1)$ in the $\phi_{\mathbf{u}(t_0)}$ coordinates with held input $\mathbf{u}(t_0)$. When the input switched to $\mathbf{u}(t_1)$ at time t_1, the dynamics are no longer linear in the $\phi_{\mathbf{u}(t_0)}$ coordinates. At this time, the transformation $\mathcal{G}_{t_0}$ is used to transform to the $\phi_{\mathbf{u}(t_1)}$ coordinates. In Pane 2, the linear evolution from $\mathbf{x}(t_1)$ to $\mathbf{x}(t_2)$ in the $\phi_{\mathbf{u}(t_1)}$ coordinates with held input $\mathbf{u}(t_1)$ is shown.

This non-autonomous representation of the Koopman operator is nonlinear because no restrictions are made on the dependence of the mapping or the eigenfunctions on $\mathbf{u}$. The representation can be finite-dimensional as long as each input is associated with a finite dimensional autonomous representation and a finite number of inputs is chosen. However, we can still gain insights about the dynamics of the system by analyzing the spectra of the individual eigenfunction representations. We illustrate the usefulness of this type of model in control design in section 4.2.

4.2 Control Algorithms

The input-parameterized eigenfunctions contain rich information on the response of the system to input. This can be leveraged to design novel control algorithms.

4.2.1 Single, u-Independent Fixed Point

As a warm up, we consider a system with known input-parameterized Koopman eigenfunction representation $(\phi_{\mathbf{u}}, \mathcal{F}^{t_k})$ with a single fixed point $\mathbf{x}^*$. Assume also that the input-parameterized eigenfunction $\phi_{\mathbf{u}}$ was computed using Laplace averages (or any other suitable method) with respect to the slowest decaying eigenvalue and normalized as in Section 1.6.1. Therefore, the levels sets of $|\phi_{\mathbf{u}}(\mathbf{x}(t))|$ are the isostables.

Assume that this fixed point is not a function of the input, and that within a small enough region around the fixed point, the effect of the input on the dynamics is small. Suppose this assumption applies in the neighborhood around $\mathbf{x}^*$ defined by $|\phi_{\mathbf{u}}(\mathbf{x}(t))| < \epsilon$ for some $\epsilon > 0$. Call the boundary of this neighborhood the (ϵ)-isostable of the dynamics. The effect of the input on the location of this isostable is assumed to be negligible.

Inspection of the input-parameterized eigenfunctions provides a method of designing a piecewise constant sequence of inputs $\mathbf{u}(t)$ which bring the initial condition $\mathbf{x}_0 = \mathbf{x}(t_0)$ toward the fixed point $\mathbf{x}^*$ by minimizing the time required for the state to cross the (ϵ)-isostable. This is because we can compute

$$T(\mathbf{x}(t), \mathbf{u}) = \frac{1}{|\mathcal{R}\{\lambda_1(\mathbf{u})\}|} \ln\left(\frac{|\phi_{\mathbf{u}}(\mathbf{x}(t))|}{\epsilon}\right) \tag{4.6}$$

which is the time it will take the state $\mathbf{x}(t)$ to enter the (ϵ)-isostable while holding the input $\mathbf{u}$ constant. Algorithm 1 outlines a process for minimizing the time to enter the (ϵ)-isostable by searching for the choice of $\mathbf{u}$ which minimizes T at every time step.

Algorithm 1: Control algorithm which steers the initial condition toward the a single, u-independent fixed point $\mathbf{x}^*$.

Result: The input sequence $\mathbf{u}(t)$ bringing the initial state $\mathbf{x}_0$ within the (ϵ)-isostable around $\mathbf{x}^*$

Choose tolerance ϵ, the time step Δt, and the initial time $t = t_0$;

while $|\boldsymbol{\phi_0}(\mathbf{x}(t))| > \epsilon$ **do**

Compute $\mathbf{u}^\dagger = \arg\min_{\mathbf{u}} T(\mathbf{x}(t), \mathbf{u})$;

Compute $\mathbf{x}(t + \Delta t) = S^{\Delta t}(\mathbf{x}(t), \mathbf{u}^\dagger)$;

Set $\mathbf{u}(\tau) = \mathbf{u}^\dagger$ in the time interval $t \leq \tau < t + \Delta t$;

Set $t = t + \Delta t$;

end

4.2.2 Systems with Multiple Fixed Points

Here, we introduce the application of the Koopman eigenfunction embedding to systems with multiple fixed points which will be helpful in our discussion of the Duffing oscillator in Section 4.3.

Often, the state space $X \subset \mathbb{C}^n$ of a nonlinear dynamical system $\dot{\mathbf{x}} = F(\mathbf{x})$ can be partitioned into N basins of attraction $\{\mathcal{B}_1, \ldots, \mathcal{B}_N\}$. Let each $\mathcal{B}_i$ be a subset of a compact domain $\mathcal{D}$. Give a Lebesgue measure μ, the union of these basins fills the compact domain $\mu(\bigcup_i \mathcal{B}_i) = \mu(\mathcal{D})$ in all but a set of measure zero given by the boundaries between the basins.

Each $\mathcal{B}_i$ has fixed point $\mathbf{x}_i^*$ and can be associated with the eigenfunctions ϕ_i whose evolution is given by

$$U_i^t \phi_i = e^{\lambda_i t} \phi_i \tag{4.7}$$

where the Koopman operators $U_i^t : \mathcal{O}|_{\mathcal{B}_i} \to \mathcal{O}|_{\mathcal{B}_i}$ are given by the restrictions of the global Koopman operator $U^t : \mathcal{O} \to \mathcal{O}$ to the space of observables with support on $\mathcal{B}_i$

given by $\mathcal{O}|_{\mathcal{B}_i} = \{ f|_{\mathcal{B}_i} : \mathcal{B}_i \to \mathbb{C} \big| f \in \mathcal{O}\}$ [5]. Here, $\lambda_i \in \mathbb{C}$ are the eigenvalues and $(\phi_i, \mathcal{F}_i^t)$ is a (potentially complex) linear representation of the Koopman operator U_i^t of the system $\dot{\mathbf{x}} = F(\mathbf{x})$ with mapping $\mathcal{F}_i^t(\phi_i) = e^{\lambda_i t}\phi_i$ restricted to $\mathbf{x} \in \mathcal{B}_i$.

We can stitch these local Koopman operators into a global Koopman operator by concatenating the eigenfunctions of each basin into $\Phi = \begin{bmatrix} \phi_1 & \dots & \phi_N \end{bmatrix}^T$ and defining the diagonal matrix of eigenvalues Λ such that

$$U^t\Phi = e^{\Lambda t}\Phi = \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_N t} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}. \tag{4.8}$$

Each ϕ_j has support in only one basin. If $\mathbf{x} \in \mathcal{B}_i$, then the eigenfunctions in the other basins $\phi_j(\mathbf{x})\ \forall j \neq i$ contain no information about the evolution of the state and can be set to zero. By first defining the mapping $\mathcal{F}^t(\Phi) = e^{\Lambda t}\Phi$, we can construct the linear representation $(\Phi, \mathcal{F}^t)$. This representation is almost faithful because it is faithful everywhere except the measure zero boundaries between basins of attraction.

As pointed out in [11], this representation does not uniquely describe the state of the system in all locations in state space. When the state reaches one of the fixed points, all eigenfunctions ϕ_i with eigenvalue $\mathcal{R}\{\lambda_i\} \neq 0$ will be identically zero. This means care must be taken to ensure that eigenfunction-based representations of multiple fixed point systems are faithful. Non-faithful representations are termed "reduced" [6].

We need to add an observable to our representation which can resolve the differences between these points. This will be another eigenfunction, computed by taking time averages of continuous functions h (Eq. 4.9).

$$h^*(\mathbf{x}) = \lim_{\bar{t}\to\infty} \frac{1}{\bar{t}} \int_0^{\bar{t}} h(S^\tau(\mathbf{x}))d\tau \tag{4.9}$$

Define the subset $F \subset M$ to be the set of points on which the time averages of continuous functions do not exist and the subset $C \subset M/F$ to be the set on which these time averages exist and are constant. Then, there exists an invariant measure μ_C such that

$$h^*(\mathbf{x}) = \int_C h d\mu_C \tag{4.10}$$

for any continuous function h and any $\mathbf{x} \in C$.

Proposition 2 *Let the system $\dot{\mathbf{x}} = F(\mathbf{x})$ with $\mathbf{x} \in M$ and flow S^t have a finite number of attractors with basins of attraction $\mathcal{B}_i$, $i = 1, ..., N$, such that $\mu(\bigcup_i \mathcal{B}_i) = \mu(M)$, where μ is the Lebesgue measure. Let the time averages of continuous functions exist everywhere on a set M/F, where $\mu(F) = 0$. The time average $h^*(\mathbf{x})$ of a continuous function $h : \mathcal{D} \to \mathbb{R}$ is a piecewise constant eigenfunction of U^t with $\lambda = 0$ defined almost everywhere with respect to μ.*

Proof: For any point $\mathbf{x} \in \mathcal{B}_i$, there exists a set C such that

$$\begin{aligned} U^t h^*(\mathbf{x}) &= U^t \lim_{\bar{t}\to\infty} \frac{1}{\bar{t}} \int_0^{\bar{t}} h(S^\tau(\mathbf{x})) d\tau \\ &= \int_C U^t h d\mu_C \\ &= \int_C h \circ S^t d\mu_C \\ &= \int_C h d\mu_C \\ &= h^*(\mathbf{x}) \end{aligned}$$

The second line used Eq. 4.10 and the fourth line used the fact that μ_C is invariant under S. This proves that h^* is an eigenfunction with eigenvalue 0. Since the basin of attraction $\mathcal{B}_j$ is arbitrary with $\mu(\bigcup_i \mathcal{B}_i) = \mu(M)$, the proposition is proven. ■

With this new eigenfunction appended to our vector of observables, the evolution of any $\mathbf{x} \in X$ is now given by

$$U^t\Phi = e^{\Lambda t}\Phi = \begin{bmatrix} e^{\lambda_1 t} & & & \\ & \ddots & & \\ & & e^{\lambda_N t} & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_1 \\ \vdots \\ \boldsymbol{\phi}_N \\ h^* \end{bmatrix}. \tag{4.11}$$

This defines an almost faithful, linear, and finite dimensional representation of the Koopman operator of $\dot{\mathbf{x}} = F(\mathbf{x})$ on all $\mathbf{x} \in \bigcup_i \mathcal{B}_i$. We will use representations of this form to tackle the problem of control between basins of attraction is systems with multiple attractive fixed points in Section 4.2.3. This will expand the work of [81] to the case of nonmonotone system, albeit without the optimality guarantees.

4.2.3 Multiple, u-Dependent Fixed Points

We now turn to control design for systems that may have more than one fixed point, and the location and number of fixed points might vary based on the input. For a given input, the global input-parameterized eigenfunctions may show multiple fixed points, but the current location of the state $\mathbf{x}(t)$ is destined to fall into only one of these fixed points if that input is held for all time. Therefore, we may write the location of the fixed point as a function of both the current state and a candidate input $\mathbf{x}^*(\mathbf{x}(t), \mathbf{u}) = \mathbf{x}^*(\mathbf{x}(t); \mathbf{u})$.

Often, we desire a sequence of inputs $\mathbf{u}(t)$ which will deliver an initial state $\mathbf{x}_0 = \mathbf{x}(t_0)$ to another point in space $\mathbf{x}^\dagger = \mathbf{x}^*(\mathbf{x}(t_M); \mathbf{u}(t_M))$ which is a fixed point when the input $\mathbf{u}(t_M)$ is held at the final time t_M. In this case, the simplest method of control is to search through all choices of held inputs $\mathbf{u} \in \mathbb{C}^p$, find one $\mathbf{u}^\dagger$ such that $\mathbf{x}^\dagger = \mathbf{x}^*(\mathbf{x}_0, \mathbf{u}^\dagger)$, and hold it for all time. This is not always possible. In Section 4.3.3, we give an example

of a control application where it is impossible to switch from the initial condition to the desired fixed point with a constant input held for all time. In this case, it is necessary to consider the time dependent sequence of inputs $\mathbf{u}(t)$.

In Algorithm 2, we propose a method which is able to cause this switching of fixed points. At every time step, use the input-parameterized eigenfunctions to compute the set of fixed points $\{\mathbf{x}^*(\mathbf{x}(t), u^0), \ldots, \mathbf{x}^*(\mathbf{x}(t), u^K)\}$ associated with the current state $\mathbf{x}(t)$ and some finite sampling of held inputs $U = \{u^0, \ldots, u^K\}$. Next, cluster these fixed points based on their Euclidean distance from each other. We use the K-means clustering available in [82]. If a group of inputs $\mathcal{C} \subset U$ all result in fixed points that are relatively close to each other, then it is likely that these fixed points all correspond to the same basin of attraction.

As in Section 4.2.1, we use Eq. 4.6 to search for the minimizer of the settling time to the (ϵ)-isostable. If at any time an input exists which results in the current state moving to the desired state faster than the current input, switch to that input. This is the algorithm employed in Section 4.3.3.

Algorithm 2: This control algorithm enables switching between the basins of attraction of a dynamical system with multiple stable fixed points. We leverage the power of our nonlinear Koopman representation by extracting time-to-fixed-point information from the input-parameterized eigenfunctions.

Result: The input sequence $\mathbf{u}(t)$ bringing the initial state $\mathbf{x}_0$ within the (ϵ)-isostable of $\mathbf{x}^\dagger = \mathbf{x}^*(\mathbf{x}_0, \mathbf{u}^\dagger)$.

Choose tolerance ϵ, the time step Δt, and the initial time t_0;

while $|\phi_{\mathbf{u}(t)}(\mathbf{x}(t))| > \epsilon$ **do**

Given a sampling of inputs $U = \{u^0, \ldots, u^K\}$, compute the $N_{\mathcal{C}}$ K-means clusters $\mathcal{C}_i$ for $i = 1, \ldots, N_{\mathcal{C}}$;

Compute $\mathcal{C}^\dagger$, the cluster which minimizes the difference between its centroid and $\mathbf{x}^\dagger$;

Compute $\mathbf{u}^\dagger = \arg\min T(\mathbf{x}(t), \mathbf{u})$ with $\mathbf{u} \in \mathcal{C}^\dagger$;

Compute $\mathbf{x}(t + \Delta t) = S^{\Delta t}(\mathbf{x}(t), \mathbf{u}^\dagger)$;

Set $\mathbf{u}(\tau) = \mathbf{u}^\dagger$ in the time interval $t \leq \tau < t + \Delta t$;

Set $t = t + \Delta t$;

end

4.3 Duffing with Dissipation

Consider the Duffing oscillator with dissipation

$$\ddot{x} = x - x^3 - \mu\dot{x} + u \tag{4.12}$$

where $\mu = 0.1$ is a dissipation coefficient and $u \in \mathbb{R}$ is the input forcing term. Define the phase space vector $\mathbf{x} = \begin{bmatrix} x & \dot{x} \end{bmatrix}^T$.When $u = 0$, this system has two basins of attraction as shown in Figure 4.2. The basins are computed by taking level sets of the time averages of

$h = x^*$ using Eq. 4.9. We call the basin of attraction with fixed point at $\mathbf{x} = \begin{bmatrix} -1 & 0 \end{bmatrix}^T$ the "left" basin denoted by $\mathcal{B}_l$ and the basin with fixed point $\mathbf{x} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ is the "right" basin denoted $\mathcal{B}_r$.

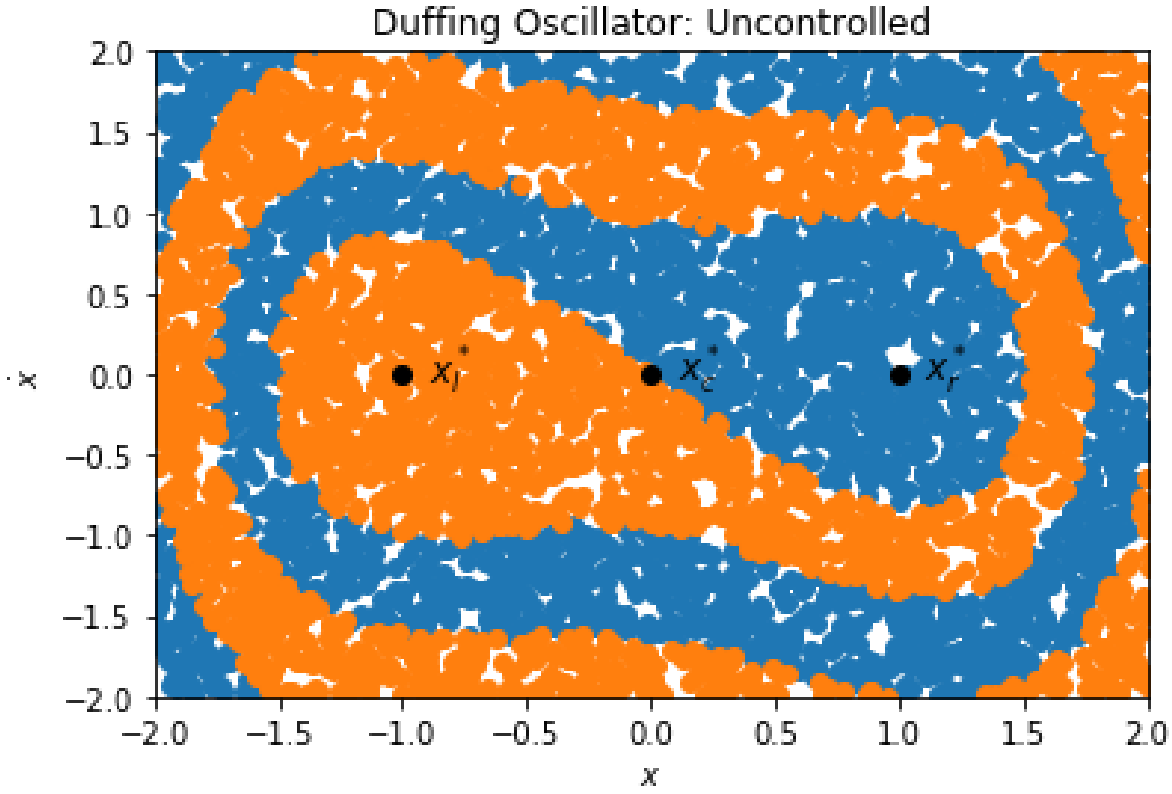


Figure 4.2: The basins of attraction shown in phase space for the Duffing oscillator with dissipation coefficient of $\mu = 0.1$. The random points in phase space are colored based on the time average of the function $h(\mathbf{x}) = x$. This provides a partitioning of the state space into basins of attraction. The left basin $\mathcal{B}_l$ is shown in orange while the right basin $\mathcal{B}_r$ is blue.

Because of dissipation, both basins of attraction of the uncontrolled system are stable with fixed points $\mathbf{x}_l^*$ and $\mathbf{x}_r^*$. There is also a third, central fixed point $\mathbf{x}_c^* = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ which is associated with the measure zero set along the boundary between the left and right basins of attraction. When inputs are introduced, the locations of these fixed points changes. At certain values of u, two fixed points will collide and disappear, leaving only

one as shown in Figure 4.3.

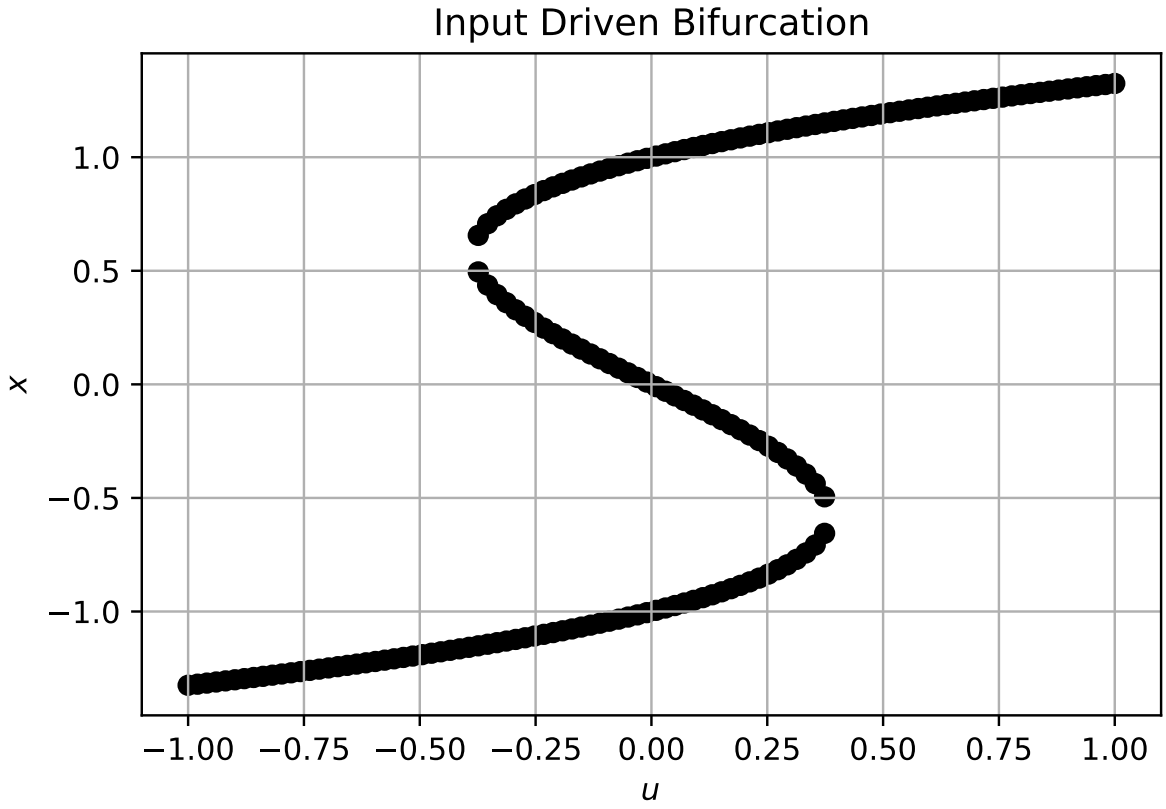


Figure 4.3: The location of the fixed points as a function of the input u. Note the input-driven bifurcation that causes the number of fixed points to change from 1 to 3 and back down to 1 as u increases. This is an input driven bifurcation.

Using the methods from Sections 4.2.2 and 4.1.1, we build a global nonlinear Koopman representation for this system and demonstrate its usefulness in control in Section 4.3.3.

4.3.1 Input-Parameterized Eigenfunction Computation

For each choice of input u, we wish to compute separate eigenfunction embeddings $\phi_l(\mathbf{x})$ and $\phi_r(\mathbf{x})$ for the basins $\mathcal{B}_l$ and $\mathcal{B}_r$, respectively. To do this, we consider $p = 25$ choices of u in the range $-0.5 < u < 0.5$ and compute the eigenfunctions of the autonomous systems defined by holding each choice of u constant. For each choice of u,

we compute the Laplace Averages from Eq. 1.22 for $10,000$ initial conditions with 2000 time samples and a time step of $\Delta t = 0.1$. Given points inside $\mathcal{B}_l$, the corresponding eigenfunction $\phi_l(\mathbf{x})$ is computed using the methods developed in [15] and outlined in Section 1.6.1. Outside $\mathcal{B}_l$, we set $\phi_l(\mathbf{x}) = 0$. The same is done for the right basin so that each eigenfunction is only supported on its corresponding basin of attraction. This combined with the time averages $h^*(\mathbf{x}) = x$ gives us the input-parameterized Koopman eigenfunctions of the Duffing system (Eq. 4.12) as shown in Figure 4.4.

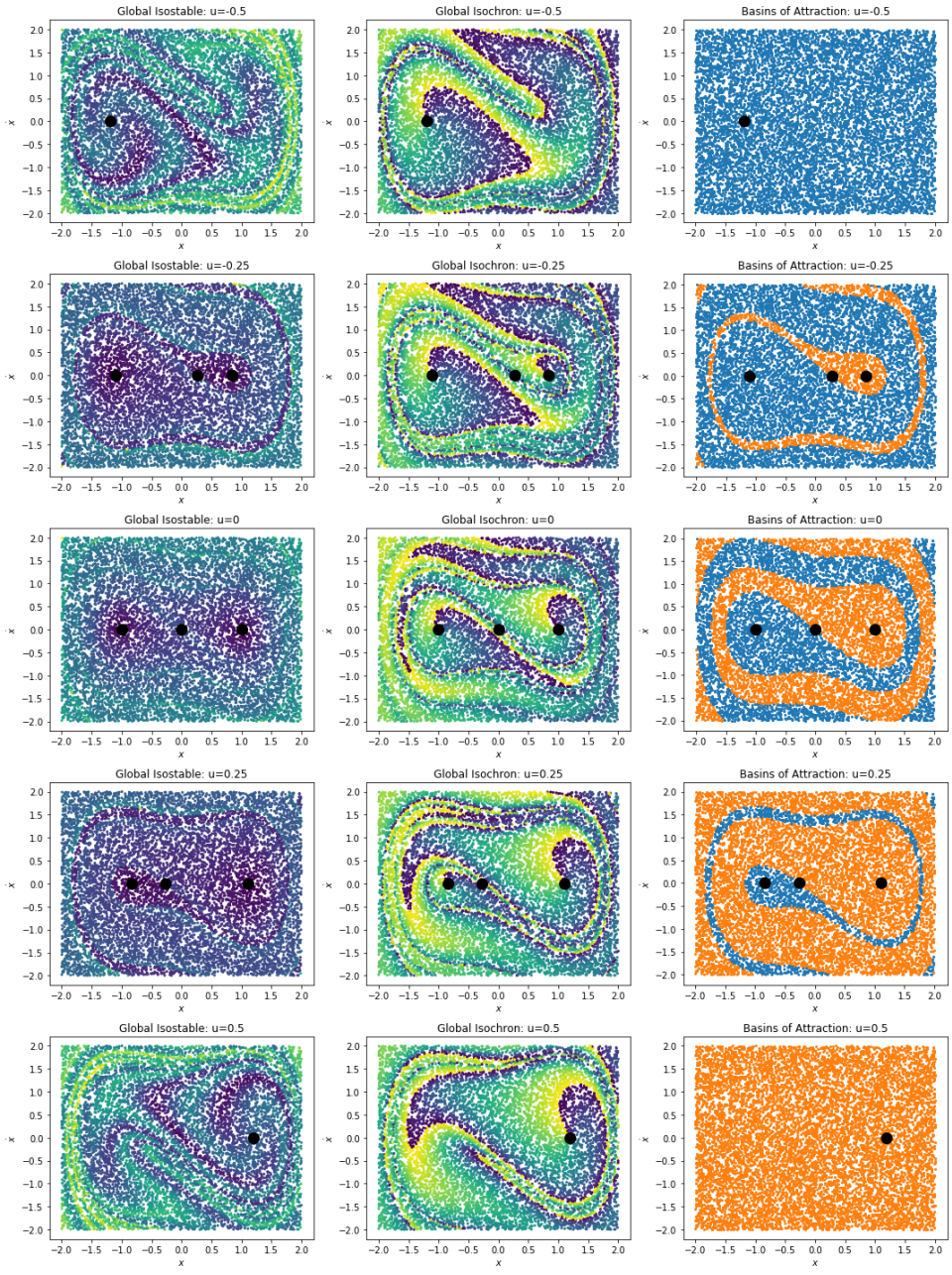


Figure 4.4: The input-parameterized eigenfunctions corresponding to the inputs $u = \{-0.5, -0.25, 0, 0.25, 0.5\}$ are shown. The level sets of the first and second columns are the isostables and isochrons of the system, respectively. The third column shows the dependence of the left and right basins of attraction on the input, with the locations of the fixed points shown as block dots. As $|u|$ becomes large, a pair of fixed points move gradually closer until they touch and disappear. Increasing $|u|$ further results in a system with only a single fixed point and basin of attraction.

4.3.2 Nonlinear Koopman Predictor

Let $\phi_l(\mathbf{x}; u^*)$and $\phi_r(\mathbf{x}; u^*)$ be the Laplace averages corresponding to the left and right basins of attraction ($\mathcal{B}_l(u^*)$ and $\mathcal{B}_r(u^*)$, respectively) as computed in Section 4.3.1 with the held input u^*. Define

$$\mathcal{B}(\mathbf{x}, u^*) = \begin{cases} 0 & \text{if} \quad \mathbf{x} \in \mathcal{B}_l(u^*) \\ 1 & \text{if} \quad \mathbf{x} \in \mathcal{B}_r(u^*) \end{cases}.$$

This is the function that points out the location of the basin of attraction of the state. This is an eigenfunction with discrete-time eigenvalue 1 in the uncontrolled case. Combine all of these together into one vector of observables

$$\Phi(\mathbf{x}; u^*) = \begin{bmatrix} \phi_l(\mathbf{x}; u^*) \\ \phi_r(\mathbf{x}; u^*) \\ \mathcal{B}(\mathbf{x}; u^*) \end{bmatrix}.$$

We can now build a collection of global, linear Koopman representations parameterized by the inputs u^*. Each representation gives a predictor of the dynamics of the Duffing oscillator with the input held constant for all time. These are of the form

$$\dot{\Phi} = e^{\Lambda(u^*)t}\Phi = \begin{bmatrix} e^{\lambda_l(u^*)t} & 0 & 0 \\ 0 & e^{\lambda_r(u^*)t} & 0 \\ 0 & 0 & 1 \end{bmatrix} \Phi \tag{4.13}$$

where $\lambda_l(u^*)$ and $\lambda_r(u^*)$ are functions of the input representing the eigenvalues of the left and right eigenfunctions, respectively. This linear ODE is a global predictor of Equation 4.12. This predictor is demonstrated in Figure 4.5 for the case where $u = 0$ is held constant for all time.

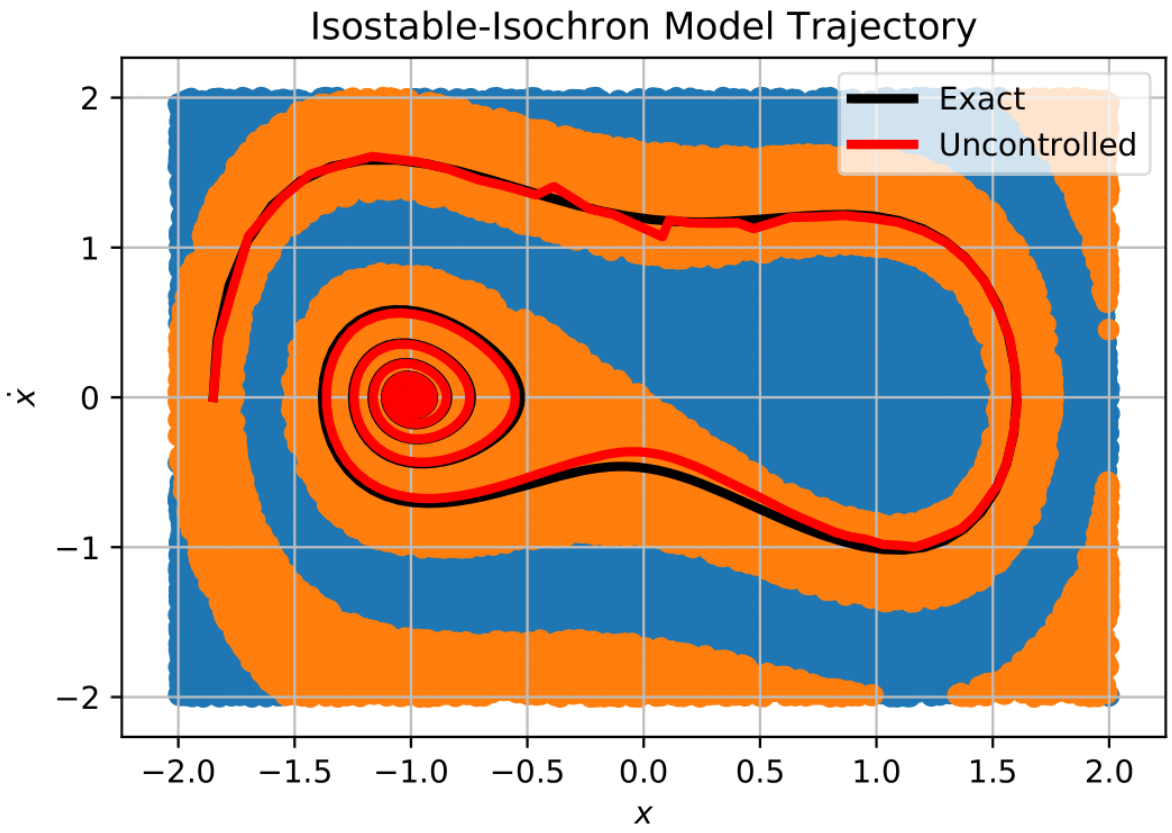
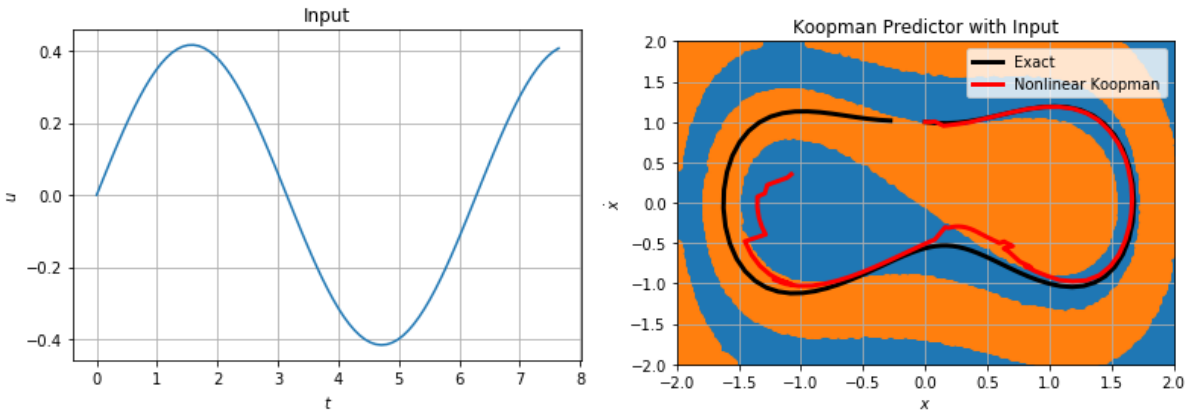


Figure 4.5: The trajectories of the true (exact) evolution of the Duffing system (Eq. 4.12) compared to the evolution of the linear Koopman representation (Eq. 4.13) with initial condition $\mathbf{x}_0 = \begin{bmatrix} -1.8 & 0 \end{bmatrix}^T$ and $u^* = 0$. This linear, faithful Koopman representation successfully predicts the evolution of this system using only three (complex-valued) eigenfunctions.

Although the linear Koopman representations are each applicable when the input is held are constant for all time, the input-parameterized eigenfunctions enable the construction of a predictor for arbitrary time-varying inputs using the nonlinear Koopman representation introduced in Section 4.1.1. A sinusoidal input is chosen to test the prediction capabilities of this representation in Figure 4.6. We approximate this as a piecewise constant input by performing nearest-neighbor interpolation at every time step $\Delta t = 0.1$ of the sinusoid onto the $p = 25$ inputs used to build the eigenfunctions $\phi_{\mathbf{u}}$. The prediction is done by applying Eq. 4.5 every time a discrete change in input occurs.



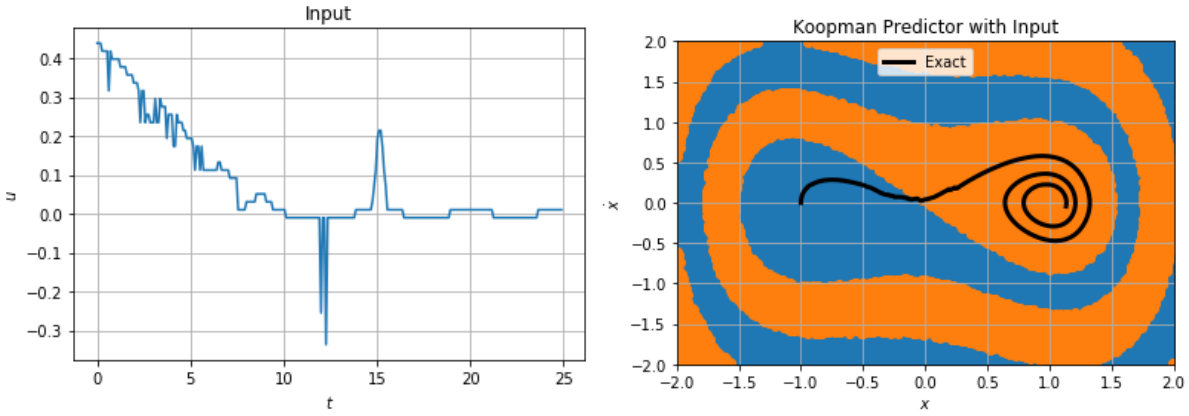
(a) (b)

Figure 4.6: The predictor with input model the crossing between basins of attraction. The prescribed input is sinusoidal and the initial condition is $\mathbf{x}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$.

4.3.3 Control Between Basins of Attraction

Our choice of embedding lends itself well to designing control strategies between the basins of attraction. This controller that achieves this by inspecting the input-parameterized eigenfunctions at every time step. This leverages our global Koopman predictor to create a controller without using any piecewise control strategies. This process is described in Algorithm 2.

We apply this method to the task of taking the initial state in the left fixed point $\mathbf{x}(t_0) = \mathbf{x}_l^*$ to the final state in the right fixed point $\mathbf{x}^\dagger = \mathbf{x}_r^*$. Asymptotic convergence to $\mathbf{x}_r^*$ requires $\mathbf{u} = 0$. However, holding $\mathbf{u} = 0$ for all time will not move the state away from the initial condition $\mathbf{x}_r^*$. Therefore, it is necessary to use Algorithm 2. We demonstrate the control result in Figure 4.7.



(a) (b)

Figure 4.7: Control result with initial condition $x_0 = [-1\ 0]^T$ and objective $x^* = [1\ 0]$. We now have a controller which uses nothing but information about the Koopman model to move between basins of attraction. The controller commands a positive but decreasing input to drive the state toward the central fixed point, a strong negative kick to push the state through the origin, and finally lets the state fall naturally into the right fixed point. No prior information about the system is needed to design this controller.

Chapter 5

Conclusion

In this dissertation, we explored three distinct types of representations of the Koopman operator. These included an eigenfunction representation learned with deep neural nets, a dual Koopman operator representation capturing both the dynamic and static behavior of a soft robot, and a nonlinear representation constructed using input-parameterized Koopman eigenfunctions. We then explored the application of these three representations in control design. Below, we give summaries of these three chapters.

5.1 Summary of "Learning and Control of Deep Koopman Eigenfunctions"

We presented a method that leverages deep learning to approximate Koopman eigenfunctions for nonlinear dynamical systems with discrete spectrum. When a nonlinear dynamical system is lifted to a space of eigenfunctions, the time evolution of the system is described by multiplication by a complex scalar called the eigenvalue. The resulting eigenfunctions and eigenvalues together form a linear representation of the Koopman operator. The resulting deep neural eigenfunction representations were shown to outperform EDMD-based Koopman representations which require a user-specified dictionary of lifting functions. We perform long-horizon predictions for a simple discrete spectrum toy system as well as the Van der Pol oscillator. We then used deep learning to discover lifting functions which linearize the inputs. This resulted in a decoupled system of linear Single-Input-Single-Output systems which are amenable to simple pole placement and linear optimal control techniques.

5.2 Summary of "Control of Soft Robots with Inertial Dynamics"

We presented a data-driven framework for the modeling and control of inertial and nonlinear soft robots. We used Koopman Operator Theory to enable the application of linear control methods to this highly nonlinear, inertial system. We introduce a Koopman-LQR with static Koopman pregain capable of accurately controlling two different inertial soft robots exhibiting high deflections and high velocities during arbitrary trajectories. Advancing the state of the art, the proposed method allows the construction and deployment of both a model and optimal controller from less than 5 minutes of training data - to the best of the authors knowledge, the shortest in soft robotics. Compared to existing MPC-based controllers, K-LQR is computationally less expensive and can be deployed on a simple microprocessor, enabling cheap and scalable use in a variety of environments outside the research laboratory. Despite its simplicity, our controller allows our soft arm to demonstrate inertial behavior, with accelerations greater than gravity and substantially greater than previous works.

Although the presented demonstration of our modeling and control paradigm focused on soft robots, its impact could be much broader. The paradigm's ability to explore the dynamical features of a complex, nonlinear, inertial system could offer advantages in modeling and control of myriad robotic systems. Further, its speed, versatility, low computational cost, and ease of use potentially expand the accessibility of robotics to new user groups. As such, we believe our paradigm has the potential to make field-deployable, dynamical, soft robotic systems significantly closer to realization.

5.3 Summary of "Input-Parameterized Koopman Eigenfunctions"

In this work, we introduced the concept of nonlinear representations of the Koopman operator to control design for strongly nonlinear systems. Our framework leverages the concept of input-parameterized Koopman eigenfunctions. These are observables which undergo linear evolution under constant input, but are allowed to transform arbitrarily under changing inputs. This Koopman representation is useful for systems whose finite-dimensional Koopman representations do not readily admit linear of bilinear control terms. This Koopman representation allows for the construction of control algorithms merely by inspection of the properties of the observables. This is demonstrated in the control of the switching between multiple basins of attraction of the Duffing oscillator with dissipation. Future work will involve categorizing and applying the many different types of nonlinear Koopman representation that might be useful in control.

Bibliography

[1] B. O. Koopman, *Hamiltonian systems and transformation in hilbert space*, *Proceedings of the National Academy of Sciences* **17** (may, 1931) 315–318.

[2] B. O. Koopman and J. v. Neumann, *Dynamical systems of continuous spectra*, *Proceedings of the National Academy of Sciences* **18** (mar, 1932) 255–263.

[3] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, *Spectral analysis of nonlinear flows*, *Journal of Fluid Mechanics* **641** (nov, 2009) 115–127.

[4] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, *Nonlinear Dynamics* **41** (aug, 2005) 309–325.

[5] A. Mauroy and I. Mezic, *Global stability analysis using the eigenfunctions of the koopman operator*, *IEEE Transactions on Automatic Control* **61** (nov, 2016) 3356–3369.

[6] I. Mezic, *Koopman operator, geometry, and learning*, *ArXiV Preprint* (Oct., 2020) [arXiv:2010.0537].

[7] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, *Journal of Fluid Mechanics* **656** (jul, 2010) 5–28.

[8] J. H. Tu, , C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. K. and, *On dynamic mode decomposition: Theory and applications*, *Journal of Computational Dynamics* **1** (2014), no. 2 391–421.

[9] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data–driven approximation of the koopman operator: Extending dynamic mode decomposition*, *Journal of Nonlinear Science* **25** (jun, 2015) 1307–1346.

[10] H. Arbabi and I. Mezić, *Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator*, *SIAM Journal on Applied Dynamical Systems* **16** (jan, 2017) 2096–2126.

[11] E. Kaiser, J. N. Kutz, and S. L. Brunton, *Data-driven discovery of koopman eigenfunctions for control*, *Machine Learning: Science and Technology* **2** (jun, 2021) 035023.

[12] E. Yeung, S. Kundu, and N. Hodas, *Learning deep neural network representations for koopman operators of nonlinear dynamical systems*, in *2019 American Control Conference (ACC)*, pp. 4832–4839, IEEE, 2019.

[13] Q. Li, F. Dietrich, E. M. Bollt, and I. 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** (oct, 2017) 103111.

[14] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, *Nature Communications* **9** (nov, 2018).

[15] A. Mauroy, I. Mezić, and J. Moehlis, *Isostables, isochrons, and koopman spectrum for the action–angle representation of stable fixed point dynamics*, *Physica D: Nonlinear Phenomena* **261** (oct, 2013) 19–30.

[16] I. Mezić and A. Banaszuk, *Comparison of systems with complex behavior*, *Physica D: Nonlinear Phenomena* **197** (Oct., 2004) 101–133.

[17] I. Mezić, *Analysis of fluid flows via spectral properties of the koopman operator*, *Annual Review of Fluid Mechanics* **45** (jan, 2013) 357–378.

[18] I. Mezić, *On numerical approximations of the koopman operator*, *Mathematics* **10** (apr, 2022) 1180.

[19] M. Korda and I. Mezic, *Optimal construction of koopman eigenfunctions for prediction and control*, *IEEE Transactions on Automatic Control* **65** (dec, 2020) 5114–5129.

[20] A. Mauroy, I. Mezić, and Y. Susuki, eds., *The Koopman Operator in Systems and Control*. Springer-Verlag, Feb., 2020.

[21] D. Bruder, X. Fu, and R. Vasudevan, *Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics*, *IEEE Robotics and Automation Letters* **6** (jul, 2021) 4369–4376.

[22] H. Arbabi, M. Korda, and I. Mezić, *A data-driven koopman model predictive control framework for nonlinear partial differential equations*, in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, dec, 2018.

[23] I. Mezić, *Spectrum of the koopman operator, spectral expansions in functional spaces, and state-space geometry*, .

[24] M. Korda and I. Mezić, *Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control*, *Automatica* **93** (jul, 2018) 149–160.

[25] M. Korda and I. Mezić, *On convergence of extended dynamic mode decomposition to the koopman operator*, *Journal of Nonlinear Science* **28** (nov, 2017) 687–710.

[26] M. Sabatini, *Non-periodic isochronous oscillations in plane differential systems*, *Annali di Matematica Pura ed Applicata* **182** (nov, 2003) 487–501.

[27] A. Mauroy and I. Mezić, *On the use of fourier averages to compute the global isochrons of (quasi)periodic dynamics*, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **22** (sep, 2012) 033112.

[28] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, .

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, .

[30] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, .

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, 2019.

[32] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[33] M. Budišić, R. Mohr, and I. Mezić, *Applied koopmanism*, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **22** (2012), no. 4 047510.

[34] I. Ermolov and A. G. Kravets, *Industrial Robotics Review*, pp. 195–204. Springer International Publishing, Cham, 2020.

[35] W. Ji and L. Wang, *Industrial robotic machining: a review*, *The International Journal of Advanced Manufacturing Technology* **103** (2019) 1239–1255.

[36] A. S. Bisen and H. Payal, *Collaborative robots for industrial tasks: A review*, *Materials Today: Proceedings* **52** (2022) 500–504. International Conference on Smart and Sustainable Developments in Materials, Manufacturing and Energy Engineering.

[37] S. Bragança, E. Costa, I. Castellucci, P. M. Arezes, J. S. Baptista, M. P. Barroso, P. Carneiro, P. Cordeiro, N. Costa, R. B. Melo, A. S. Miguel, and G. Perestrelo, *A Brief Overview of the Use of Collaborative Robots in Industry 4.0: Human Role and Safety*, pp. 641–650. Springer International Publishing, Cham, 2019.

[38] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, *Interactive, collaborative robots: Challenges and opportunities.*, in *IJCAI*, pp. 18–25, 2018.

[39] F. Vicentini, *Collaborative robotics: a survey*, *Journal of Mechanical Design* **143** (2021), no. 4.

[40] D. Rus and M. T. Tolley, *Design, fabrication and control of soft robots*, *Nature* **521** (2015), no. 7553 467–475.

[41] C. Majidi, *Soft robotics: a perspective—current trends and prospects for the future*, *Soft robotics* **1** (2014), no. 1 5–11.

[42] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, *Control strategies for soft robotic manipulators: A survey*, *Soft robotics* **5** (2018), no. 2 149–163.

[43] J. Wang and A. Chortos, *Control strategies for soft robot systems*, *Advanced Intelligent Systems* **4** (2022), no. 5 2100165.

[44] L. Shi, Z. Liu, and K. Karydis, *Koopman operators for modeling and control of soft robotics*, *arXiv preprint arXiv:2301.09708v1* (2023).

[45] O. Yasa, Y. Toshimitsu, M. Y. Michelis, L. S. Jones, M. Filippi, T. Buchner, and R. K. Katzschmann, *An overview of soft robotics*, *Annual Review of Control, Robotics, and Autonomous Systems* **6** (2022).

[46] R. K. Katzschmann, C. Della Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, *Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model*, in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 454–461, IEEE, 2019.

[47] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, *Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer*, in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 717–724, IEEE, 2019.

[48] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, *Model-based dynamic feedback control of a planar soft robot: Trajectory tracking and interaction with the environment*, *The International Journal of Robotics Research* **39** (2020), no. 4 490–513.

[49] C. Della Santina, R. L. Truby, and D. Rus, *Data–driven disturbance observers for estimating external forces on soft robots*, *IEEE Robotics and Automation Letters* **5** (2020), no. 4 5717–5724.

[50] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, *Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators*, *IEEE Transactions on Robotics* **35** (2019), no. 1 124–134.

[51] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, *Closed-loop dynamic control of a soft manipulator using deep reinforcement learning*, *IEEE Robotics and Automation Letters* **7** (2022), no. 2 4741–4748.

[52] X. Wang and N. Rojas, *A data-efficient model-based learning framework for the closed-loop control of continuum robots*, 2022.

[53] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, *Soft robot perception using embedded soft sensors and recurrent neural networks*, *Science Robotics* **4** (2019), no. 26.

[54] R. L. Truby, C. Della Santina, and D. Rus, *Distributed proprioception of 3d configuration in soft, sensorized robots via deep learning*, *IEEE Robotics and Automation Letters* **5** (2020), no. 2 3299–3306.

[55] S. Neppalli, B. Jones, W. McMahan, V. Chitrakaran, I. Walker, M. Pritts, M. Csencsits, C. Rahn, and M. Grissom, *Octarm - a soft robotic manipulator*, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007.

[56] D. Braganza, D. M. Dawson, I. D. Walker, and N. Nath, *A neural network controller for continuum robots*, *IEEE Transactions on Robotics* (2007).

[57] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, *SIAM Journal on Applied Dynamical Systems* **15** (2016), no. 1 142–161, [https://doi.org/10.1137/15M1013857].

[58] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Generalizing koopman theory to allow for inputs and control*, *SIAM Journal on Applied Dynamical Systems* **17** (jan, 2018) 909–930.

[59] L. Shi and K. Karydis, *Acd-edmd: Analytical construction for dictionaries of lifting functions in koopman operator-based nonlinear robotic systems*, *IEEE Robotics and Automation Letters* **7** (2021), no. 2 906–913.

[60] H. Arbabi and I. Mezic, *Computation of transient koopman spectrum using hankel-dynamic mode decompoisition*, *APS* (2017) G1–009.

[61] H. Arbabi and I. Mezic, *Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator*, *SIAM Journal on Applied Dynamical Systems* **16** (2017), no. 4 2096–2126.

[62] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, *Modeling and control of soft robots using the koopman operator and model predictive control*, *arXiv preprint arXiv:1902.02827* (2019).

[63] D. Bruder, C. D. Remy, and R. Vasudevan, *Nonlinear system identification of soft robot dynamics using koopman operator theory*, in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6244–6250, IEEE, 2019.

[64] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, *Koopman-based control of a soft continuum manipulator under variable loading conditions*, *arXiv preprint arXiv:2002.01407* (2020).

[65] J. Chen, Y. Dang, and J. Han, *Offset-free model predictive control of a soft manipulator using the koopman operator*, *Mechatronics* **86** (oct, 2022) 102871.

[66] H. Yin, M. Welle, and D. Kragic, *Policy learning with embedded koopman optimal control .*, *Proceedings of Machine Learning Research* **144** (2018).

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

[68] A. Gibson, *Application of koopman linear quadratic regulator to the control of a spherical microbubble*, Master's thesis, University of Colorado Colorado Springs, 2022.

[69] G. Mamakoukas, M. L. Castano, X. Tan, and T. D. Murphey, *Derivative-based koopman operators for real-time control of robotic systems*, *IEEE Transactions on Robotics* **37** (2021), no. 6 2173–2192.

[70] D. Bruder, X. Fu, G. Brent, D. Remy, and R. Vasudevan, *Data-driven control of soft robots using koopman operator theory*, *IEEE Transactions on Robotics* **37** (2021), no. 3 948–961.

[71] A. Kazemipour, O. Fischer, Y. Toshimitsu, K. W. Wong, and R. K. Katzschmann, *A robust adaptive approach to dynamic control of soft continuum manipulators*, *arXiv preprint arXiv:2109.11388* (2021).

[72] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[73] I. Mezić, *Koopman operator, geometry, and learning of dynamical systems*, *Notices of the American Mathematical Society* **68** (2021), no. 7 1087–1105.

[74] Y. Huang, M. Hofer, and R. D'Andrea, *Offset-free model predictive control: A ball catching application with a spherical soft robotic arm*, in *2021 International Conference on Intelligent Robots and Systems (IROS)*, pp. 563–570, IEEE/RSJ, 2021.

[75] M. M. Coad, L. H. Blumenschein, S. Cutler, J. A. R. Zepeda, N. D. Naclerio, H. El-Hussieny, U. Mehmood, J.-H. Ryu, E. W. Hawkes, and A. M. Okamura, *Vine robots: Design, teleoperation, and deployment for navigation and exploration*, *IEEE Robotics & Automation Magazine* **27** (2019), no. 3 120–132.

[76] N. D. Naclerio and E. W. Hawkes, *Simple, low-hysteresis, foldable, fabric pneumatic artificial muscle*, *IEEE Robotics and Automation Letters* **5** (2020), no. 2 3406–3413.

[77] L. H. Blumenschein, N. S. Usevitch, B. H. Do, E. W. Hawkes, and A. M. Okamura, *Helical actuation on a soft inflated robot body*, in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 245–252, IEEE, 2018.

[78] A. Surana, *Koopman operator based observer synthesis for control-affine nonlinear systems*, in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, Dec., 2016.

[79] S. Peitz and S. Klus, *Koopman operator-based model reduction for switched-system control of PDEs*, *Automatica* **106** (aug, 2019) 184–191.

[80] A. Sootla, A. Mauroy, and D. Ernst, *Optimal control formulation of pulse-based control using koopman operator*, *Automatica* **91** (may, 2018) 217–224.

[81] A. Sootla, G.-B. Stan, and D. Ernst, *Solving optimal control problems for monotone systems using the koopman operator*, in *Lecture Notes in Control and Information Sciences*, pp. 283–312. Springer International Publishing, 2020.

[82] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.