University of California
Santa Barbara

Dynamic Model Abstractions for Identification and Control of Deep Neural and Biological Computation Networks

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

Doctor of Philosophy
in
Mechanical Engineering

by

Charles Addison Johnson

Committee in charge:

Professor Enoch Yeung, Chair
Professor Linda Petzold
Professor Jeff Moehlis
Professor Igor Mezić
Professor Steve Haase, Duke University

June 2026

The Dissertation of Charles Addison Johnson is approved.

Professor Linda Petzold

Professor Jeff Moehlis

Professor Igor Mezić

Professor Steve Haase, Duke University

Professor Enoch Yeung, Committee Chair

May 2026

Dynamic Model Abstractions for Identification and Control of Deep Neural and Biological Computation Networks

by

Charles Addison Johnson

Dedicated to Lauren, Benjamin and Indiana.

Acknowledgements

The work in Chapter 2 was supported partially by a Defense Advanced Research Projects Agency (DARPA) Grant No. FA8750-19-2-0502, PNNL Grant No. 528678, ICB Grant No. W911NF-19-D-0001 and No. W911NF-19-F-0037, and ROE Young Investigator Grant No. W911NF-20-1-0165.

The work in Chapter 3 primarily funded by the Army Young Investigator Program Award W911NF-20-1-0165 and the Army Research Office grants, W911NF-19-D-0001, W911NF22F0005, and W911NF-23-2-0006. This work was also supported in part by a subcontract awarded by the Pacific Northwest National Laboratory for the Secure Biosystems Design Science Focus Area "Persistence Control of Engineered Functions in Complex Soil Microbiomes" sponsored by the U.S. Department of Energy Office of Biological and Environmental Research.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government.

On a personal note, I would like to express my grattitude to Prof. Enoch Yeung for his mentorship and support. This support began during a summer undergraduate internship at PNNL. After I completed my Master's degree I was excited to have a second chance to work with Enoch as a PhD student. I have learned so much from his instruction, feedback and example. I would also like to thank my wife of 10 years, Lauren, who has been working on her doctorate at the same time. Her dedication to supporting me, as well as her research and knowledge as a social scientist have changed my life. During the PhD process we have both worked in acadamia and raised our two children, Benjamin and Indiana. This would not have been possible without support from our extended

family, including our parents, siblings and grandparents. I would also like to thank my collaborators in the Biological Control Lab at UCSB as well as out-of-department collaborators. Finally, I would like to thank my committee members for their support and feedback in this final step of my PhD journey.

Curriculum Vitæ

Charles Addison Johnson

Education

2026	Ph.D. in MEchanical Engineering (Expected), University of California, Santa Barbara.
2020	M.A. in Computer Science, Brigham Young University.
2017	B. A. in Mathematics with University Honors, Brigham Young University.

Publications

1. Tuning a Genetic Circuit with Double Negative Feedforward Loops to Approximate Square Waves; J Wu, C Johnson, E Yeung bioRxiv, 2025.11. 13.688370 2025
2. Heterogeneous Mixtures of Dictionary Functions to Approximate Subspace Invariance in Koopman Operators: Why Deep Koopman Operators Work; CA Johnson, S Balakrishnan, E Yeung Journal of Nonlinear Science 35 (4), 68 5 2025
3. Distributional Approach to Risk Preferences; B Jabarian, N Chemaya, C Johnson, E Yeung, G Charness 2025
4. A novel model class for bowtie biological networks with universal classification properties; CA Johnson, K Ho, E Yeung arXiv preprint arXiv:2505.13703 2025
5. Nonlinear data-driven control via koopman models: the interplay of stabilization and subspace closure error; CA Johnson, E Yeung 1 2024
6. Learning Invariant Subspaces of Koopman Operators–Part 1: A Methodology for Demonstrating a Dictionary's Approximate Subspace Invariance; CA Johnson, S Balakrishnan, E Yeung arXiv preprint arXiv:2212.07358 1 2022
7. Learning Invariant Subspaces of Koopman Operators–Part 2: Heterogeneous Dictionary Mixing to Approximate Subspace Invariance; CA Johnson, S Balakrishnan, E Yeung arXiv preprint arXiv:2212.07365 1 2022
8. The production of English syllable-level timing patterns by bilingual English-and Spanish-speaking children with cochlear implants and their peers with normal hearing; M Gibson, F Bunta, C Johnson, M Huárriz Journal of Phonetics 95, 101194 2 2022
9. Heterogeneous mixtures of dictionary functions to approximate subspace invariance in koopman operators; CA Johnson, S Balakrishnan, E Yeung arXiv preprint arXiv:2206.13585 1 2022
10. Characterizing network controllability and observability for abstractions and realizations of dynamic networks; CA Johnson, S Warnick IFAC-PapersOnLine 53 (2), 10987-10993 2 2020

11. Graph theoretic foundations of cyclic and acyclic linear dynamic networks; CA Johnson, N Woodbury, S Warnick IFAC-PapersOnLine 53 (2), 26-33 4 2020
12. Abstractions of Graph Models; CA Johnson Brigham Young University 2020
13. Ultrasound and nasometric evidence for controlled high vowel nasalization in Montreal French; M Dow, M Gibson, C Johnson 2019
14. Target control and source estimation metrics for dynamical networks; A Vosughi, C Johnson, M Xue, S Roy, S Warnick Automatica 100, 412-416 31 2019
15. A class of logistic functions for approximating state-inclusive koopman operators; CA Johnson, E Yeung 2018 Annual American Control Conference (ACC), 4803-4810 43 2018
16. Local control and estimation performance in dynamical networks: Structural and graph-theoretic results; A Vosughi, C Johnson, S Roy, S Warnick, M Xue 2017 IEEE 56th Annual Conference on Decision and Control (CDC), 4032-4037 12 2017
17. Net path and net effect of linear networks with dynamics; CA Johnson Brigham Young University 1 2017

Abstract

Dynamic Model Abstractions for Identification and Control of Deep Neural and Biological Computation Networks

by

Charles Addison Johnson

The underlying theme that characterizes my research arc has been that of phenomenological modeling of dynamical systems and mathematical abstraction of networks. By abstracting complicated networks (such as neural networks trained to learn dynamic system behavior), we can build much lower parameter and geometrically interpretable models that capture the same properties of the more expensive and harder to understand deep neural networks. Additionally, an abstracted vision of biochemical networks allows us to discover new mechanisms within cellular transcriptional networks to better understand how bacteria can adapt to and perform computation in challenging and changing environments.

In Chapter 2 we analyze the learned dictionaries from the deep network based Koopman system identification algorithm, deepDMD. We explore the theoretical basis for the strong performance of this algorithm. We do so by discovering dictionaries to satisfy a property of subspace approximation, which we define as uniform finite approximate closure. We also discover that structured mixing of heterogeneous dictionary functions drawn from different classes of nonlinear functions can be used to achieve the same accuracy and dimensional scaling as the deep-learning-based deepDMD algorithm. This mixed dictionary achieves similar accuracy and dimensional scaling to deepDMD with an order of magnitude reduction in parameters, while maintaining geometric interpretability.

In Chapter 3 we connect the concepts of Koopman system identification and Koopman

state feedback control. For control affine systems, we demonstrate examples of state-feedback controllers to illustrate that, when control is the objective, the Koopman model becomes subject to the control design process. This implies that certain scenarios merit data-driven control strategies different from system identification strategies that prioritize best fit. We also present a controller design framework to stabilize systems modeled with a data-driven Koopman model. Since Koopman models propose a linear state-space representation of the true dynamics, they often imply that a proportional control law applied to the states of the Koopman model can stabilize the origin. We compute error bounds on the minimal nonlinear control action needed to stablize a system that has already been assigned a Koopman model with stabilizing proportional control.

In Chapter 4, we shift gears to model cell sensory transcription networks, the intracellular computation structure that regulates and drives cellular activity. Here we explore the computation (and classification) potential of single-celled organisms. We present a model for identification and response to environmental changes in resilient bacteria. This model combines two known motifs in transcription networks: dense overlapping regulons (DORs) and single input modules (SIMs). Combined in a hybrid network motif we have processes for decision making, control actuation and feedback. In Chapter 4, we model this hybrid network motif (which we call the DOR2SIM motif) with a superposition of modular nonlinear functions to describe protein signaling in the network and basic mass action kinetics to describe the other chemical reactions in this process. We also introduce a notion of classification for dynamic systems to explain bacterial decision making in terms of classification. Given this definition, we provide sufficient conditions for models of the DOR2SIM motif to classify. These conditions suggest that relatively low monomer degradation rates as well as low expression of source node genes (the inputs to the DORs) at equilibrium enables classification.

Contents

Chapter 1

Introduction

This work represents a combination of modeling and controls research for general dynamic systems and for the investigation of biological computation and classification. Chapter 2 was first prepared as two distinct journal submissions, which were later on combined into a single journal paper and submitted to and accepted for publication in the Journal of Nonlinear Science. Chapter 3 was originally prepared as a conference paper, however, there was too much content and so I am expanding it into a journal paper as well. Before Chapter 3 is published we intend to add numerical simulation results and numerical analysis. At present, I am working with a third co-author to prepare these results[1]. The purpose of these numerical results will be to highlight the relationship between data-driven Koopman operator system identification and state-feedback control design. Chapter 4 was also prepared as a conference paper that I am expanding into a journal paper. Before submitting, I will add substantial data and modeling based on data that I am collecting in the wet lab. Throughout this dissertation, the main focus are theoretical underpinnings and mathematical modeling foundations. Each chapter is self-contained in presenting its motivation, overview and introducing related work. However, I will briefly

[1]The work in this dissertation does not include any contributions from said co-author.

introduce the work of this dissertation below, begninning with Koopman models.

Koopman operators model nonlinear dynamics as a linear dynamic system acting on a nonlinear function as the state. This nonstandard state is often called a Koopman observable and is usually approximated numerically by a superposition of functions drawn from a *dictionary*. In a widely used algorithm, *Extended Dynamic Mode Decomposition* (EDMD), the dictionary functions are drawn from a fixed class of functions. Deep learning combined with EDMD has been used to learn novel dictionary functions in an algorithm called deep dynamic mode decomposition (deepDMD) [1]. The learned representation both (1) accurately models and (2) scales well with the dimension of the original nonlinear system. In Chapter 2 we analyze the learned dictionaries from deepDMD and explore the theoretical basis for their strong performance. We explore State-Inclusive Logistic Lifting (SILL) dictionary functions to approximate Koopman observables. Error analysis of these dictionary functions show they satisfy a property of subspace approximation, which we define as uniform finite approximate closure. Typically, a Koopman dictionary's nonlinear functions are homogeneous. In Chapter 2, we discover that structured mixing of heterogeneous dictionary functions drawn from different classes of nonlinear functions achieve the same accuracy and dimensional scaling as the deep-learning-based deepDMD algorithm. We specifically show this by building a heterogeneous dictionary comprised of SILL functions and conjunctive radial basis functions (RBFs). This mixed dictionary achieves similar accuracy and dimensional scaling to deepDMD with an order of magnitude reduction in parameters, while maintaining geometric interpretability. These results strengthen the viability of dictionary-based Koopman models to solving high-dimensional nonlinear learning problems.

Koopman system identification techniques model nonlinear dynamics by using a dictionary of functions to learn a linear system that is higher-dimensional than the data used to train the model. These methods are data-driven and may be used to design control

laws for unknown dynamics. In Chapter 3, for control affine systems, we demonstrate examples of state-feedback controllers to illustrate that, when control is the objective, the Koopman model becomes secondary and subject to the control design process. This implies that certain scenarios merit data-driven control strategies different from system identification strategies that prioritize best fit. We also present a controller design framework to stabilize systems modeled with a data-driven Koopman model. Since Koopman models propose a linear state-space representation of the true dynamics, they often imply that a proportional control law applied to the states of the Koopman model can stabilize the origin. We explore the relationship between Koopman model accuracy and the degree of success applying proportional feedback control based on the Koopman model. We do this by computing error bounds on the minimal control action needed to stabilize a system after proportional control is applied to the Koopman model. These error bounds relate the Koopman model error to how well feedback control will stabilize the system. One potential application of these system identification and control strategies is to model and control transcriptional networks in cells.

Cell sensory transcription networks are the intracellular computation structure that regulates and drives cellular activity. Activity in these networks determines the the cell's ability to adapt to changes in its environment. Resilient cells successfully identify (classify) and appropriately respond to environmental shifts. We present a model for identification and response to environmental changes in resilient bacteria. This model combines two known motifs in transcription networks: dense overlapping regulons (DORs) and single input modules (SIMs). DORs have the ability to perform cellular decision making and have a network structure similar to that of a shallow neural network, with a number of input transcription factors (TFs) mapping to a distinct set of genes. SIMs contain a master TF that simultaneously activates a number of target genes. Within most observed cell sensory transcription networks, the master transcription factor of SIMs are

output genes of a DOR creating a fan-in-fan-out (bowtie) structure in the transcriptional network. In Chapter 4, we model this hybrid network motif (which we call the DOR2SIM motif) with a superposition of modular nonlinear functions to describe protein signaling in the network and basic mass action kinetics to describe the other chemical reactions in this process. We analyze this model's biological feasibility and capacity to perform classification, the first step in adaptation. We provide sufficient conditions for models of the DOR2SIM motif to classify constant (environmental) inputs. These conditions suggest that generally low monomer degradation rates as well as low expression of source node genes at equilibrium in the DOR component enable classification.

Chapter A is a collection of appendicies for the proceeding chapters. Material deemed to interrupt the flow of the chapters is afforded a section in the appendicies.

Chapter 2

Heterogeneous Mixtures of Dictionary Functions to Approximate Subspace Invariance in Koopman Operators: Why Deep Koopman Operators Work

Published in Volume 35 of the Journal of Nonlinear Systems in May of 2025

2.1 Introduction

Koopman operator theory considers an alternate representation of a dynamic system where the state evolution of a nonlinear system is linear. In this representation, the concepts of vibrational, growth, and decay modes in linear systems can be directly extended to nonlinear systems [2]. These modes address problems in the field of fluid mechan-

ics [3–5], disease modeling [6], attack identification in the power grid [7], programming the steady state of biological systems [8], and extracting new biosensors [9]. The spectral properties of the linear Koopman operator in this function space connects to classical methods for model reduction, validation, identification and control [10]. Since Mezić's first paper on the spectral properties of the Koopman operator, computing Koopman modes has become a major research focus [11, 12]. The central algorithm for computing these modes is dynamic mode decomposition (DMD). Extensions of DMD, such as extended dynamic mode decomposition (EDMD) [13] enable the approximation of strong nonlinearities in the model. Others, such as the robust approximation algorithm presented in [14], enable high fidelity modeling in the presence of both process and measurement noise. Increasing levels of sophistication are required to build large-scale, data-driven models of complicated, nonlinear dynamic processes. Koopman operator theory provides an alluring, though nuanced, approach to building such models [3–6, 8–12].

To build universally applicable models, as the scale of systems increases, model classes need to capture as many dynamics as possible in the smallest number of parameters and model dimensions. This requirement has restricted the viability of dictionary-based Koopman models such as the extended dynamic mode decomposition (EDMD [13]) algorithm. Such models become unwieldy because human defined dictionaries with nonlinearities placed in an evenly spaced grid scale, in dimension, exponentially with the dimension of the state space. However, Koopman models, in theory, do not need their parameters to scale exponentially with the dimension of the state. One approach to learning lower dimensional models is through the SINDy algorithm. This algorithm uses sparse regression to project to a lower dimensional subspace of the nonlinear function space initially chosen [15]. Another, Symmetric Subspace Decomposition, approximates the maximal Koopman-invariant subspace in an iterative fashion [16].

Some empirical results suggest deep-learning based models scale much better with

state dimension [1, 17–20]. One of these approaches is to use deep learning to learn a function dictionary or a set of observables during training [1]. The deepDMD algorithm uses a form of stochastic gradient descent (SGD) to train a deep artificial neural network to select observables. Because the neural network is a universal function approximator [21], it can learn a large class of continuous observables on a compact set and parameterize the approximate Koopman operator. The deepDMD algorithm has been generally successful at learning both small and large scale nonlinear systems [1].

Other approaches leverage deep learning to build efficiently parameterized Koopman models with high fidelity. Methods using linear recurrent autoencoders build effective, low-dimensional Koopman models that do not explicitly use linear combinations of the state as dictionary functions [17–19]. To do this, they pass observables from the learned function space, via the decoder, back to the original model's state-space. Other work has, under the assumption of ergodicity, delegated the process of choosing a dictionary for EDMD to deep learning [20].

Neural networks are typically black box models. In this work, we use dictionaries of functions inspired from deep learning to build Koopman models which both capture the system dynamics and approximate the dynamics of the nonlinear dictionary functions. The success of these models provides insight into why deep learning models have the capacity to approximate Koopman operators well. We explain this success by demonstrating that these dictionaries satisfy uniform finite approximate closure, a property tied to the subspace invariance of these dictionaries.

In sections 2.6 and 2.7 of this paper we demonstrate the *uniform finite approximate closure* of the state-inclusive logistic lifting (SILL) dictionary. The SILL dictionary was inspired by some of the dictionary functions learned by deepDMD, the deep-learning based Koopman model in [1]. To an extent, this analysis explains the success of deepDMD in learning. But, deepDMD learns more than this one type of function (see Fig. 3 of [1]).

Indeed, it can learn dictionary functions from multiple function classes simultaneously, and we have observed it to do so (see Fig. 2.1).

Inspired by these observations, we also contribute the first analysis of a heterogeneous, or mixed, basis dictionary function for Koopman learning. Traditionally, dictionary based methods for building Koopman models focus on homogeneous dictionaries. In such a dictionary, each of the nonlinear candidate dictionary functions comes from a specific function basis, such as a polynomial or Fourier basis [22]. The SILL dictionary, defined in Section 2.4 of this paper, is one such homogeneous dictionary. While mixing elements from different functional basis does increase the representational efficiency of the dictionary, the subspace invariance, or closure, properties of such a dictionary are not straightforward to understand.

Introducing heterogeneous basis leads to added complexity of the resulting dictionaries. This complexity is transferred to the representation of the vector field over which we take Lie-derivatives of each dictionary function (see the proof of Lemma 1 in Section A.1 of the appendix). We evaluate the vector field of the new class of Koopman models by computing the Lie-derivatives of their heterogeneous mixture of dictionary functions. We then work from these Lie-derivatives to argue that these Koopman models are approximately subspace invariant and that they will scale favorably with (1) system dimension (see sections 2.7.1 and 2.8.3) as well as (2) model dimension and dictionary steepness (see sections 2.6 and 2.8.1). Combining our new function dictionaries with an effective learning algorithm yields a training loss comparable to deepDMD, but with an order of magnitude reduction in model complexity.

In sections 2.2 and 2.3 we define the Koopman learning problem and provide foundational definitions and notation for this paper. In sections 2.4 and 2.5 we define deep-learning-inspired dictionaries for solving the Koopman learning problem. In sections 2.6, 2.7 and 2.8 we mathematically demonstrate the favorable modeling properties of these

dictionaries. In Section 2.9 we show these favorable properties numerically. Finally, we draw our conclusions in Section 2.10.

2.2 The Koopman generator learning problem

In this section we introduce the problem of choosing a function space and approximate Koopman generator to model a dynamic system. This problem is not convex, but, when approximated well, it gives an accurate, data-driven, linear model of a nonlinear system.

Consider a nonlinear, time-invariant, autonomous system with dynamics

$$\dot{x} = f(x) \tag{2.1}$$

where $x \in \mathbb{R}^n$ for $n \in \mathbb{N}$, $f : \mathbb{R}^n \to \mathbb{R}^n$. We introduce the concepts of a Koopman generator and its associated multi-variate Koopman semigroup, following the exposition of [11].

For continuous nonlinear systems, the Koopman semigroup is a semigroup, a set with an associative binary operation, $\mathcal{K}_{t\in\mathbb{R}}$ of linear but infinite dimensional operators, $\mathcal{K}_t$, that acts on a space of functions, Ψ, with elements $\psi : \mathbb{R}^n \to \mathbb{R}^N$, for $N \in \mathbb{N}$. Our function, ψ, is an observable because it is a function of the state, x. We say $\mathcal{K}_t : \Psi \to \Psi$ is an operator for each $t \geq 0$. The Koopman operator applies the transformation,

$$\mathcal{K}_t \circ \psi(x_0) = y \circ \Phi_t(x_0), \tag{2.2}$$

where $\Phi_t(x)$ is the flow map of the dynamical system (2.1) evolved forward up to time t, given the initial state x_0. Instead of examining the evolution of the state, the Koopman semigroup allows us to study the forward evolution of functions of state, $\psi(x)$ [13].

The action of the generator, $\mathcal{K}_\mathcal{G}$, on the observables, ψ, for the Koopman semigroup

is defined as

$$\mathcal{K}_{\mathcal{G}} \circ \psi \triangleq \lim_{t \to 0} \frac{\mathcal{K}_t \circ \psi - \psi}{t}. \tag{2.3}$$

When ψ and t are fixed, we see from Eq. (2.2) that $\mathcal{K}_t$ is a function of the state, x. Similarly, $\mathcal{K}_{\mathcal{G}}$ may be understood as a function of x. The Koopman generator is a state-dependant derivative operator, see Section 7.6 of [23]. It satisfies

$$\frac{d}{dt}\psi(x) = \mathcal{K}_{\mathcal{G}}(x) \circ \psi(x). \tag{2.4}$$

In rare cases the Koopman generator will not have a dependence on x, see Eq. (24) and the surrounding section in [24].

2.2.1 Learning Koopman operators from data

In discrete-time, data-driven Koopman operator learning (explored in Section 2.9) we have r pairs of measurements of the state,

$$(x_i, x_{i+1}) = (x_i, f(x_i)), \text{ for } i = 1, 2, ..., r. \tag{2.5}$$

Note that, here, each $x_i \in \mathbb{R}^n$ is a single full state measurement.

In continuous-time, data-driven Koopman generator learning (the topic of sections 2.6, 2.7 and 2.8) our pairs are instead measurements of the state and their derivatives as follows:

$$(x_i, \frac{dx_i}{dt}) = (x_i, f(x_i)), \text{ for } i = 1, 2, ..., r. \tag{2.6}$$

We briefly remark that, there are techniques for a continuous-time data-driven Koopman generator learning problem working with discrete measurements instead of measurements paired with their derivatives. For example, see [25].

Because we are working with numerical computation, the state (and its derivative, if applicable) is n-dimensional, where $n < \infty$. We assume all measurements of the state are noise-free.

Given our data, Eq. (2.5), it is customary to choose a function that encapsulates a finite set of N *dictionary functions*, $\psi : \mathbb{R}^n \to \mathbb{R}^N$, a constant matrix, $K \in \mathbb{R}^{N \times N}$ and a projection, $\mathtt{P} : \mathbb{R}^N \to \mathbb{R}^n$, to model f. We use ψ to refer to the dictionary functions here because, our final Koopman generator approximation will be acting on data-centered evaluations of the dictionary functions as observables.

Note that, in a data-driven setting, we choose ψ without knowing the equation for $f(x)$. Typically, we choose N such that $N > n$, and so $\psi(x)$ is a "lifting" of our data (see the use of "lifting" in [26]).

In numeric methods for Koopman modeling, we approximate f with the matrix-vector product, $K\psi(x)$. Effectively, in a data-driven setting, this amounts to projecting the action of an infinite dimensional Koopman operator as matrix multiplication on sampled data space. Thus we have

$$\mathtt{P}(\frac{d\psi(x)}{dt}) \approxeq f(x). \tag{2.7}$$

The matrix, K, is a linear operator, to ensure that it behaves like a true Koopman generator we choose it, in conjunction with ψ, to satisfy

$$\frac{d\psi(x)}{dt} \approxeq K\psi(x). \tag{2.8}$$

This, in summary, yields the following finite approximation to the Koopman generator equation

$$\frac{dx}{dt} = f(x) \approxeq \mathtt{P}(\frac{d\psi(x)}{dt}) \approxeq \mathtt{P}(K\psi(x)). \tag{2.9}$$

We are tasked to learn a numerical, finite-dimensional approximation K of the Koopman

generator, $\mathcal{K}_{\mathcal{G}}$ and the set of dictionary functions, ψ. This matrix K is the Koopman generator approximation that acts specifically on data-centered evaluations of dictionary functions $\psi(x)$ as observable functions of the state, x.

2.2.2 Problem Statement

To approximate the true Koopman generator, we solve the following optimization problem.

$$\min_{K,\psi\in\Psi} \sum_{i=1}^{r} \left\| \frac{d\psi(x_i)}{dt} - K\psi(x_i) \right\|, \tag{2.10}$$

where each $x_i \in \mathbb{R}^n$ is one of our r full state measurements.

This is an abstraction of the discrete-time problem statement. In practice, derivatives are difficult to numerically compute and measure.

In this optimization we need to select dictionary functions, as well as a real-valued matrix, K. This optimization problem is non-convex, since the form of $\psi(x)$ is unknown or parametrically undefined. The model dimension, N, is a hyperparameter of Eq. (2.10).

In EDMD the dictionary functions $\psi(x)$ are predefined, drawn from a *homogeneous* class of nonlinear functions, and assumed to be known. Under these assumptions, Eq. (2.10) is a convex optimization problem with a closed-form solution. By contrast, in deepDMD, the dictionary functions and K are learned simultaneously during iterative training. This is, of course, a nonlinear, non-convex optimization problem, for which we employ variants of the stochastic gradient descent algorithm from Tensorflow or Pytorch, such as adaptive gradient descent (AdaGrad, [27]) or adaptive momentum (ADAM, [28]). These numerical approaches provide no guarantee that low-parameter approximations to the learned set of dictionary functions $\psi_1(x), \psi_2(x), ..., \psi_N(x)$ are homogeneous, or drawn from the same function class. The outcomes of deepDMD often produce what appear to be heterogeneous dictionaries $\psi(x)$.

2.2.3 Finite closure

Previous work characterizes the closure and convergence of Koopman models as additional dictionary functions are appended to the model for the DMD and EDMD algorithms [26,29]. Other work connects the number of data-points in a data-driven learning context for specific learning algorithms to closure [30]. We explore closure as an inherent property of a dictionary. We begin by understanding the property of subspace invariance, which corresponds to finite exact closure. Closure, in this context, refers to how the action of the Koopman generator on the dictionary functions does not give any function outside of the dictionary functions' span. This span is the subspace that we refer to.

Let S, be the span of our dictionary functions. The set S is a subspace of the set of all functions that map x to $\mathbb{R}$.

Definition 1. *We say that our dictionary, $\psi(x)$, satisfies Koopman subspace invariance when $\mathcal{K}_{\mathcal{G}} \cdot \psi(x) \in S$. A Koopman subspace invariant dictionary is said to satisfy finite exact closure.*

If some element in a dictionary does not satisfy Koopman subspace invariance, then there exists some element in the dictionary that, when acted on by the Koopman operator, cannot be represented as a linear combination of dictionary functions. In the context of data-driven Koopman learning this means that when our dictionary contains N functions, no N by N matrix captures the precise action of the Koopman generator.

Finite exact closure (Koopman subspace invariance) is unlikely to be achieved, as (1) the Koopman generator may need to be infinite dimensional and (2) in the case that it does not, it is difficult to engineer models with exact closure even with an explicit knowledge of Eq. (2.1). So, we also consider an approximate notion of closure relevant to building models from data.

Definition 2. *We say $\psi(x) : \mathbb{R}^n \to \mathbb{R}^N$ achieves finite ϵ-closure or finite closure with $O(\epsilon)$ error when there exists a $K \in \mathbb{R}^{N\times N}$ and an $\epsilon > 0$ such that*

$$\frac{d(\psi(x))}{dt} = K\psi(x) + \epsilon(x), \tag{2.11}$$

for the vector field f.

Our dictionary $\psi(x)$ achieves finite approximate closure when, for the vector field, f, and every x, the function $\epsilon(x)$ is a bounded for every K such that $||K|| < \infty$.

We say that $\psi(x)$ achieves uniform finite approximate closure *for some set $\mathcal{R}$ when it achieves finite closure with $|\epsilon(x)| < B \in \mathbb{R}$ for all $x \in \mathcal{R}$.*

We are most interested by the property of uniform finite approximate closure, especially when we can bound the constant, B, to be arbitrarily low. When B can be bound in such a manner we have an accurate, data-driven model.

Example 1. *Unfortunately, closure does not come with every dictionary of observables, even if that dictionary spans the function space of the dynamic system. For example, a canonical polynomial basis, $\{1, x, x^2, ..., x^{N-1}\}$, used to approximate the one-dimensional system $f(x) = x^2$, spans the dynamic system, but no model using this basis will be closed. In fact, none will achieve uniform finite approximate closure. We illustrate what this lack of closure means when $x >> 1$. In that case, the Lie derivative of x^n is*

$$\frac{d(x^{N-1})}{dt} = \frac{d(x^{N-1})}{dx}\frac{dx}{dt} = (N-1)x^{N-2}x^2 = (N-1)x^N. \tag{2.12}$$

Approximating this derivative when $x >> 1$ as an N^{th} degree polynomial will dramatically fail as the error will be of order $O(x^N)$. This failure will cascade back through approximations of all the other dictionary functions. Closure is a crucial property of these models!

We want models with uniform finite approximate closure because, as the bounding constant goes to zero, $B \to 0$, we may use K to perform stability, observability and spectral analysis. We see this is true as, in the discrete-time formulation, as $B \to 0$, K approaches a projection of the action of the Koopman operator [26]. In the continuous-time formulation, we assume (without proof) this result implies, as $B \to 0$, that K approaches the action of the Koopman generator.

The dictionaries that we consider in this paper will be state-inclusive, meaning that their first n-dimensions are simply the first n state variables.

Definition 3. *A dictionary, $\psi(x)$, is state-inclusive when*

$$\begin{bmatrix} I & 0 \end{bmatrix} \psi(x) = x \tag{2.13}$$

when I is the $n \times n$ identity matrix.

When a dictionary is state-inclusive, it is trivial to project from ψ to x and its trajectory may yield stability insights. To keep our models meaningful in this way, all the dictionaries in this article are state-inclusive.

2.3 Notation

Our models, throughout this article, will have two classes of parameters. Each class refers to the distinct geometric properties of center and steepness. To create a clearer separation of function variables and parameters we will use the following notation

$$H(x; \mu_l, \alpha_l).$$

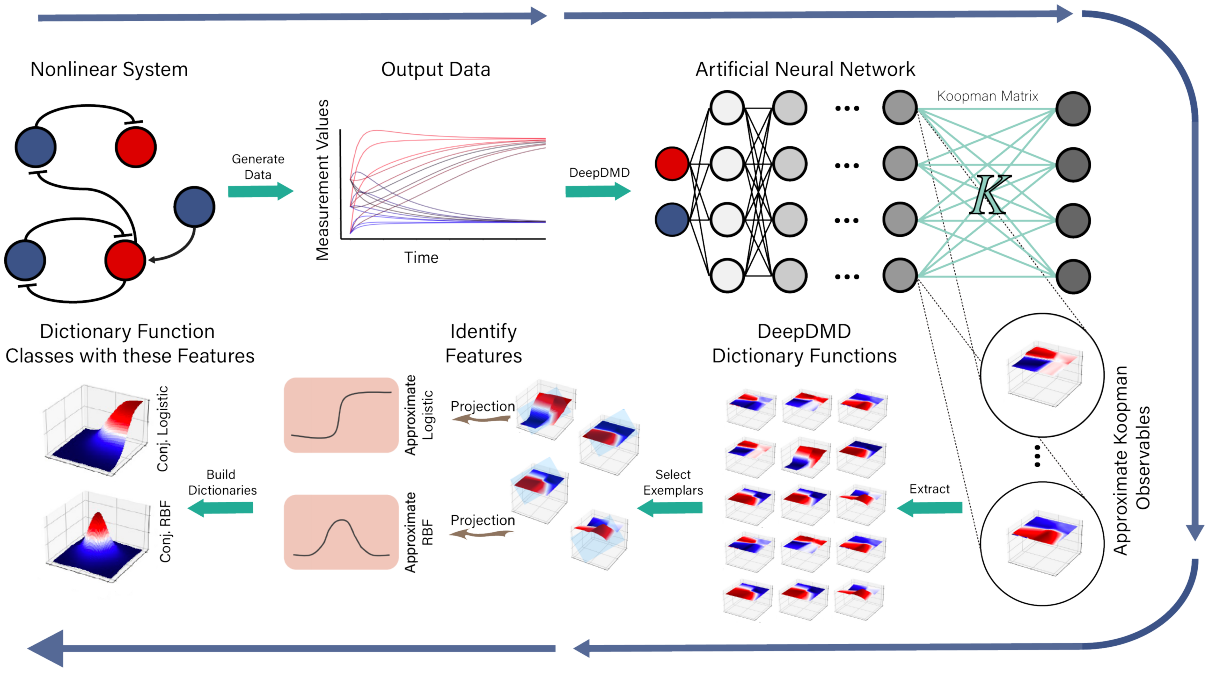

Figure 2.1: Process for the discovery of a novel mixed function dictionary with approximate subspace invariance. First, the deepDMD algorithm takes data from a nonlinear system to build approximate Koopman observables. Then, projections of these observables are analyzed to extract functional properties. Finally, high-dimensional functions, whose projections satisfy these properties, are developed and their closure properties are verified.

In this notation, H is a function whose variable is the vector x and whose parameters are vectors μ_l and α_l. The vectors μ_l and α_l refer to the geometrically distinct classes of parameters, μ for center and α for steepness. A second example would be

$$\eta(x_i; \mu_{li}, \alpha_{li}).$$

In this notation, η is a function whose variable is the scalar x_i, and whose parameters are the scalars μ_{li} and α_{li}. To make our equations less cumbersome we summarize all the distinct parameters with the label θ. For example,

$$H(x; \mu_l, \alpha_l) \triangleq H(x; \theta_l),$$

and

$$\eta(x_i; \mu_{li}, \alpha_{li}) \triangleq \eta(x_i; \theta_{li}).$$

In general, our notation uses the following conventions.

1. Integer i will be an index of state dimension.
2. Integer j will be an index of the first group of added dimensions.
3. Integer k will be an index of the second group of added dimensions.
4. Integer l will be an additional added dimension index.
5. Integer n will be the state dimension.
6. Integer N will be the added dimensions. When we mix two basis, the dimension of the first (conjunctive logistic functions) will be N_L and the dimension of the second (conjunctive radial basis functions) will be N_R. This means that $N_L + N_R = N$.
7. The $n \times N$ real-valued matrix, w, will be a matrix of weights. In our analysis it corresponds to a block of the Koopman Operator approximation matrix, K, which describes the flow of the state variables as a linear combination of nonlinear observables.

2.4 The SILL dictionary: A homogeneous dictionary model of deepDMD's learned dictionary

In this section, we define a new class of dictionary functions, $\psi(x)$, identified from deepDMD's solution to Eq. (2.10). We will call this class *State-Inclusive Logistic Liftings* (SILLs). We call them this because

1. they contain the state of the vector field $f(x)$. The dynamics of the governing equations are directly included in the dictionary, so the dictionary is state-inclusive.

2. The dictionaries in this class also contain nonlinear functions, all of which are conjunctive logistic functions, so these dictionaries are logistic in nature.

3. Finally, since each has at least one nonlinear dictionary function, the dictionary size, $N = 1 + n + N_L$, is greater than the state dimension, n. Since $N > n$, the model using this dictionary is "lifted" to a higher dimension than the original state.

We previously showed that Koopman models chosen from SILL dictionaries have successfully learned global nonlinear phase-space behavior of several simple, nonlinear systems [31]. In Section 2.6, we show, for the first time, that SILL dictionaries satisfy uniform finite approximate closure.

2.4.1 The SILL functions and deepDMD

We define a multivariate conjunctive logistic function, $\Lambda : \mathbb{R}^n \to \mathbb{R}$, for a given center parameter vector $\mu_j \in \mathbb{R}^n$ and steepness parameter vector $\alpha_j \in \mathbb{R}^n$ as follows

$$\Lambda(x; \mu_j, \alpha_j) \triangleq \prod_{i=1}^{n} \lambda(x_i; \theta_{ji}), \tag{2.14}$$

where the measurements are $x \in \mathbb{R}^n$ and the scalar logistic function, $\lambda : \mathbb{R} \to \mathbb{R}$, is defined as

$$\lambda(x_i; \mu_{ji}, \alpha_{ji}) \triangleq \frac{1}{1 + e^{-\alpha_{ji}(x_i - \mu_{ji})}}. \tag{2.15}$$

The parameter μ_{ji} is the center or the point of activation for $\Lambda(x; \theta_j)$ along dimension x_i. The parameter α_{ji} is a steepness or sensitivity parameter, and determines the steepness of the logistic curve in the i^{th} dimension for $\Lambda(x; \theta_j)$. Conjunctive logistic functions map

orthants of R^n to be nearly 1 and the rest of the space to be nearly 0. The boundary between this orthant and the rest of R^n is defined by the center parameters and the nature of the transition (gradual vs steep) comes from the steepness parameters.

To illustrate how this conjunctive logistic function works, consider what happens if you set the vector x to be constant in all but the l^{th} dimension. Then

$$\Lambda(x; \mu_j, \alpha_j) = (\prod_{i \neq l} c_i)\lambda(x_l; \mu_{jl}, \alpha_{jl}). \tag{2.16}$$

This is a constant times the logistic function in the l^{th} dimension. When we project dictionary functions learned by deepDMD to a single dimension we observe that they likewise approximate a scaled logistic function.

2.4.2 SILL functions and systems are smooth generalizations of step functions for piecewise constant dynamic systems

A conjunctive logistic function may be thought of as a smooth relaxation of an n-dimensional step function, $S : \mathbb{R}^n \to \mathbb{R}$, so that

$$S(x; \mu_j) = \begin{cases} 1 \text{ if } x_i \geq \mu_{ji}, \forall i = 1, ..., n, \\ 0 \text{ otherwise.} \end{cases} . \tag{2.17}$$

Specifically,

$$\lim_{\alpha \to \infty} \Lambda(x; \mu_j, \alpha_j) = S(x, \mu_j). \tag{2.18}$$

This is important to note because of the following proposition, which is proved in Section A.1 of the appendix.

Proposition 1. *Given a piecewise constant vector field, $f(x)$, that can be expressed as a finite combination of these n-dimensional step functions,*

$$f_i(x) = \sum_{j=1}^{N_L} w_{ij} S(x; \mu_j), \forall i = 1, ..., n, \tag{2.19}$$

there exists a finite state-inclusive dictionary including n-dimensional step functions that will satisfy exact closure for all $x \in \mathbb{R}^n$ except for a set of points on a set of Lebesgue measure zero.

This means that state-inclusive dictionaries of SILL functions are a relaxation of dictionaries with exact closure properties on a broad class of vector fields. Exploring the relationship between n-dimensional step functions and these SILL relaxations will be an important component of our analysis of the performance of SILL functions as Koopman dictionary functions.

2.4.3 The SILL Koopman dictionary

Given N multivariate logistic functions, we define a SILL dictionary as $\psi : \mathbb{R}^n \to \mathbb{R}^N$, so that:

$$\psi(x) \triangleq \begin{bmatrix} 1 & x^T & \bar{\Lambda}(x)^T \end{bmatrix}^T \tag{2.20}$$

where $\bar{\Lambda}(x) = [\Lambda(x; \theta_1), \ldots, \Lambda(x; \theta_N)]^T$ is a vector of conjunctive logistic functions and $N = 1 + n + N_L$. We then have that $K \in \mathbb{R}^{N \times N}$. This dictionary is state-inclusive. Ideally $\bar{\Lambda}(x)$, the state and a constant spans each dimension of the vector field over the region of interest.

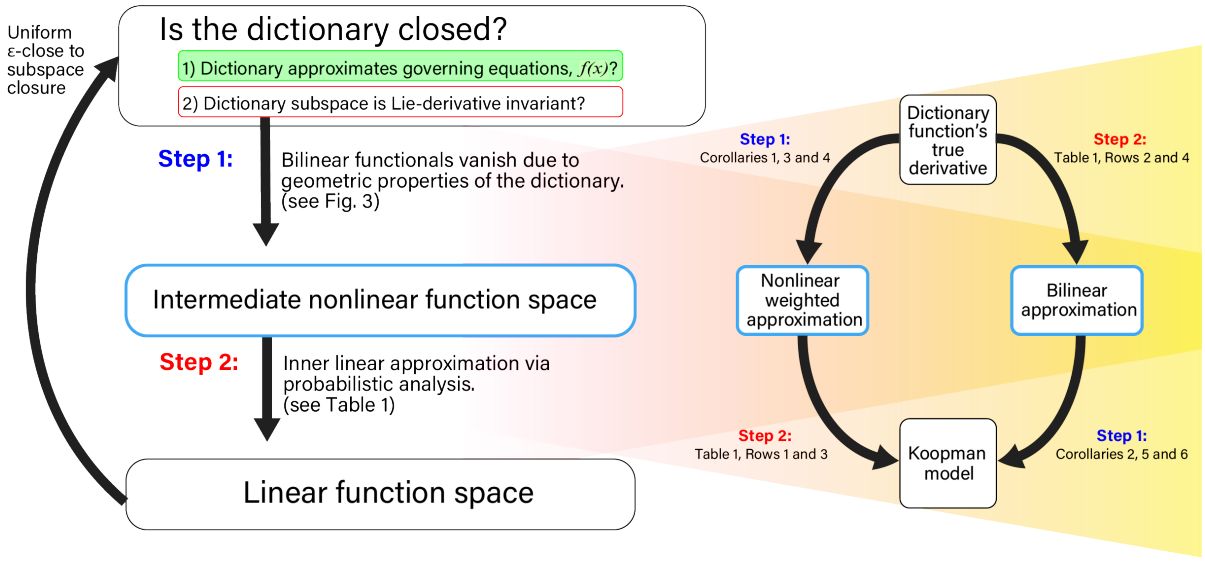

Figure 2.2: A visual outline of the (commutative) steps taken to prove uniform finite approximate closure of SILL dictionaries in sections 2.6 and 2.7. These same steps are applied in Section 2.8 to the heterogeneous augSILL dictionary. In summary, under the assumption that the governing equations are in the span of a given dictionary, there is a commutative two-step process to approximating the true system dynamics to a Koopman model. The work in this article describes error bounds for each step.

2.5 The AugSILL dictionary: A heterogeneous dictionary model of deepDMD's learned dictionary

In this section, we define a heterogeneous dictionary for Koopman learning. This dictionary, like the SILL dictionary, is inspired from the dictionary functions learned by deepDMD.

In our experience, projections of deepDMD's dictionary functions tended to match the profiles of logistic functions. However, in these dictionary function projections, there were some punctuated irregularities, divots and bumps not easy to represent as a logistic function (see Fig. 2.1). We choose to model these irregularities with radial basis functions (RBFs). We therefore augment our SILL dictionary with conjunctive multivariate RBFs (conjunctive RBFs). We define an RBF with a steepness of α_{ki} and a center of μ_{ki} as

follows:

$$\rho(x_i;\theta_{ki}) \triangleq \rho(x_i;\mu_{ki},\alpha_{ki}) \triangleq \frac{e^{-\alpha_{ki}(x_i-\mu_{ki})}}{(1+e^{-\alpha_{ki}(x_i-\mu_{ki})})^2}. \tag{2.21}$$

This RBF takes on its global maximum value of $\frac{1}{4}$ when the state, $x_i = \mu_{ki}$, the center parameter. It radially approaches zero at a rate determined by the value of the steepness parameter α_{ki}.

Note the following relationship between our RBF and logistic functions. The RBF $\rho(x_i;\theta_{ki}) = \lambda(x_i;\theta_{ki}) - \lambda(x_i;\theta_{ki})^2$. So, even in one dimension, exactly approximating an RBF would require an infinite linear combination of SILL functions (a piecewise linear spline on infinitesimally small intervals). Thus, conjunctive RBFs contribute distinct nonlinear features that are outside the span of the SILL function space. This mathematical observation is the rationale for referring to this mixture of dictionary functions as heterogeneous.

We define a conjunctive RBF to be $P : \mathbb{R}^n \to \mathbb{R}$ so that:

$$P(x;\theta_k) \triangleq P(x;\mu_k,\alpha_k) \triangleq \prod_{i=1}^{n} \rho(x_i;\theta_{ki}). \tag{2.22}$$

Conjunctive RBFs map their center to a value of 4^{-n} and radially around that center drop off to zero. The steepness parameters determine how quickly the drop off to zero occurs along each coordinate axis.

When we set the vector x to be constant in all but the l^{th} dimension, $P(x;\mu_j,\alpha_j) = (\prod_{i\neq l} c_i)\rho(x_l;\mu_{jl},\alpha_{jl})$. This is a constant times the RBF in the l^{th} dimension. When we project dictionary functions learned by deepDMD to a single dimension, we observe that many contours approximate logistic functions, others approximate RBFs and scaled sums of RBFs and logistic functions. Fig. 3 of [1] demonstrates some of these projected functions.

We propose the *augSILL* (augmented SILL) dictionary as the stacked vector of a mixture of dictionary functions of x

$$\psi(x) \triangleq \begin{bmatrix} 1 & x^T & \bar{\Lambda}^T & \bar{P}^T \end{bmatrix}^T, \tag{2.23}$$

where the vector

$$\bar{\Lambda} \triangleq [\Lambda(x;\theta_1), ..., \Lambda(x;\theta_{N_L})]^T \tag{2.24}$$

contains all conjunctive logistic functions, and

$$\bar{P} \triangleq [P(x;\theta_{N_L+1}), ..., P(x;\theta_{N_L+N_R})]^T \tag{2.25}$$

contains all conjunctive radial basis functions. In this article, the augSILL dictionary includes N_L conjunctive logistic functions and N_R conjunctive RBFs. The augSILL dictionary is of dimension $N = 1 + n + N_L + N_R$. To our knowledge this is the first time that anyone has analyzed the behavior of mixed dictionary functions for the numerical approximation of a Koopman operator or Koopman generator.

Can a mixture of heterogeneous functions form a coherent basis that preserves subspace invariance, or at least uniform finite approximate closure? Certainly the more varied basis facilitates approximating the vector field, $f(x)$, due to the increased degrees of freedom in the dictionary. But the subspace invariance properties are just as important to Koopman models as approximating the vector field (see Example 1). Using a mixed basis dictionary complicates the computation of the Lie derivatives of dictionary elements. This is because the representation of the vector field itself (a component in Lie derivative computation) has more degrees of freedom.

2.6 Uniform finite approximate closure of the SILL dictionary: Step 1

We showed in [31] that a pure, state-inclusive, SILL dictionary can make effective low-dimensional Koopman models. We demonstrate that the SILL dictionary satisfies uniform finite approximate closure (see Definition 2 in Section 2.2.3). In essence, uniform finite approximate closure guarantees that the dimensionality of the dictionary space does not need to diverge to infinity, while simultaneously approximating the vector field $f(x)$ sufficiently well. This property is what makes numerical approximation of the Koopman generator equation possible.

Recall from Example 1 that when considering a new set of dictionary functions, we also have to consider the effects of dictionary explosion. In Example 1, we computed the Lie derivatives of each dictionary function, which in turn generated new product terms that were not in the span of the existing dictionary. Thus, for any new class of dictionary functions, a key property requisite to uniform approximate finite closure is the ability to approximate products of dictionary functions as elements of the span of the dictionary.

In the context of the SILL dictionary, we need to show how approximate products of dictionary functions are elements in the span of our dictionary. We do this by showing convergence in steepness of *products* of conjunctive logistic functions to a *single* conjunctive logistic function. To show these bilinear terms are approximately in the span of our dictionary, we need to show that the product of two conjunctive logistic functions may be approximated as a single conjunctive logistic function as follows:

$$\Lambda(x;\theta_l)\Lambda(x;\theta_j) \approx \Lambda(x;\theta^*) \tag{2.26}$$

where $\theta^* = (\mu_{max}(l,j), \alpha_{max}(l,j))$ and

$$\mu_{max}(l,j) = \left(\max\{\mu_{l1}, \mu_{j1}\}, ..., \max\{\mu_{lm}, \mu_{jm}\}\right),$$

and $\alpha_{max}(l,j)$ are the α (steepness) values that correspond to the indices of the μ's.

Theorem 1 demonstrates that Eq. (2.26) is a good approximation in the limit of increasing steepness parameter α. Specifically, it says that a bilinear combination of conjunctive logistic functions can be approximated by a single conjunctive logistic function. This is important to show the uniform finite approximate closure of the SILL dictionary as it paves the way for Corollaries 1 and 2. These corollaries are key steps in understanding the error between the Koopman model using the SILL dictionary and the true Lie derivatives of these dictionary functions (see **Step 1** in Fig. 2.2). The proofs of Theorem 1 and Corollary 1 are in Section A.1 of the Appendix.

To characterize the value of our approximation we need to define the regions where our approximation's efficacy is capped. Theorem 1 will demonstrate that the approximation in Eq. (2.26) can be arbitrarily good outside of a special set of hyperplanes. Given a SILL dictionary, $\psi(x)$, we define a set of nN_L, $n-1$ dimensional hyperplanes in $\mathbb{R}^n$ corresponding to the centers of each $M_{\bar{\Lambda}} \triangleq \{x : \text{there exists } i \in \{1,2,...,n\} \text{ and } j \in \{1,2,...,N_L\} \text{ so that } x_i = \mu_{ji}\}$. This defines specific state values where the approximation in Eq. (2.26) has maximal error and the error cannot be reduced. In practice, this does not stop the SILL dictionary from achieving high levels of performance for system identification, see Section 2.9. This makes sense as the set $M_{\bar{\Lambda}}$ is a finite collection of $n-1$ dimensional sets in $\mathbb{R}^n$, and is therefore of Lebesgue measure zero.

Theorem 1. *Under Assumption 1, if the state, x, does not exactly match the SILL dictionary functions corresponding center parameters, $x \notin M_{\bar{\Lambda}}$, then, as the steepness parameters go to infinity, the product of two conjunctive logistic function will exponentially*

approach a single conjunctive logistic function in the dictionary, specifically, $\alpha \to \infty$*,*

$$\Lambda(x;\theta_l)\Lambda(x;\theta_j) - \Lambda(x;\theta^*) \to 0$$

at a rate of $e^{-c\alpha}$*, for some positive constant* c*.*

Theorem 1 implies an intermediate result to demonstrating uniform finite approximate closure of the SILL basis in approximating a Koopman generator. This result is given in Corollary 1 and it connects Theorem 1 to the Lie derivative of a SILL dictionary function.

Since our Koopman models are linear in nature, each Lie derivative will need to be approximated as a linear combination, or weighted sum, of dictionary functions. However, the Lie derivative of each dictionary function is a nonlinear weighted sum of bilinear terms. Corollary 1 replaces all of the bilinear terms in this nonlinear weighted sum with individual dictionary functions. This gives us an intermediate approximation that is nearly the linear combination of dictionary functions that we need for a Koopman model.

Corollary 1. *Under the assumptions of Theorem 1, when* f *is spanned by a SILL dictionary, the Lie derivative of a conjunctive logistic function exponentially approaches a finite weighted sum of conjunctive logistic functions as the steepnesses of the functions goes to infinity. Specifically,*

$$\dot{\Lambda}(x;\theta_l) \to \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij} (1 - \lambda(x_i;\theta_{li}))\Lambda(x;\theta^*) \tag{2.27}$$

exponentially as $\alpha \to \infty$*.*

The intermediate approximation to the Lie derivative of a conjunctive logistic function given in Corollary 1 will be further approximated as a Koopman model in Section 2.7.

This intermediate approximation is not a linear combination of dictionary functions like a Koopman model would be. However, the error between this intermediate approximation and the true Lie derivative is uniformly bounded by the intermediate bounding constant

$$\bar{B}_1 = \max_{x \in \mathbb{R}^n} \sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} (1 - \lambda(x_i; \theta_{li})) \varepsilon_{\Lambda_l \Lambda_j}(x, i, j), \tag{2.28}$$

where $\varepsilon_{\Lambda_l \Lambda_j}(x, i, j) \triangleq \Lambda(x; \theta_l)\Lambda(x; \theta_j) - \Lambda(x; \theta^*)$. The intermediate bound, $\bar{B}_1$, is finite as it is a finite sum and product of finite-valued functions. When $x \notin M_{\bar{\Lambda}}$ the bound decreases exponentially as α becomes increasingly positive because $\varepsilon_{\Lambda_l \Lambda_j}(x, i, j)$ does so as well. This error bound for the intermediate approximation will be used in Section 2.7.2 to bound the overall error of a Koopman model using the SILL dictionary.

We now show the second step in an alternate path to characterizing the finite closure properties of the SILL dictionary. This is to approximate the SILL observables' Lie derivative as a linear combination of SILL basis functions,

$$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \Lambda(x; \theta^*), \tag{2.29}$$

where $l \in \{1, 2, ..., N_L\}$. This step corresponds to the lower right arrow in the right side of Fig. 2.2 and is given in Corollary 2.

Corollary 2. *Under the assumptions of Theorem 1, the error between*

$$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \Lambda(x; \theta_l)\Lambda(x; \theta_j) \tag{2.30}$$

and Eq. (2.29) goes to zero exponentially as $\alpha \to \infty$.

Proof: The proof follows from a direct application of Theorem 1. ■

The error described for the approximation in Corollary 2 is uniformly bounded by

$$\tilde{B}_2 = \max_x \sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \varepsilon_{\Lambda_l \Lambda_j}(x, i, j), \tag{2.31}$$

where $\varepsilon_{\Lambda_l \Lambda_j}$ is defined as above, and this bound likewise goes to zero exponentially as α increases. The bounding constant B which we will use to demonstrate uniform finite closure will be a function of $\bar{B}_1$, $\tilde{B}_2$ as well as two other bounds. This relationship is in Section 2.7.2.

The end result of Eq. (2.27) is a nonlinear combination of dictionary functions and Eq. (2.30) is an approximation of the Lie derivative, $\dot{\Lambda}$. We explore the approximation quality in Section 2.7. Section 2.7 also ties these results to conclude that the SILL dictionary satisfies uniform finite approximate closure and that the bounding constant goes to zero, $B \to 0$, exponentially as steepness of the dictionary functions and the state dimension increase. Thus, SILL dictionary functions define a spanning set for Koopman observables.

Ref	**Approximation**	**Difference (Error)**	**Error Bound**
(2.27)	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \Lambda(x; \theta^*)$	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \lambda(x_i; \theta_{li}) \Lambda(x; \theta^*)$	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \frac{\nu_{ij}}{2^{n+1}}$
(A.9)	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \Lambda(x; \theta_l) \Lambda(x; \theta_j)$	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \lambda(x_i; \theta_{li}) \Lambda(x; \theta_l) \Lambda(x; \theta_j)$	$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \frac{\nu_{ij}}{2^{2n+1}}$

Table 2.1: Approximations to and properties of error bounds for the equations referred to in the **Reference** column. The reference equation is approximated as the corresponding equation in the **Approximation** column. We give the error of this approximation in the **Difference (Error)** column. The **Error Bound** column gives a bound on this error. Each constant ν_{ij} is bounded by the product of a weight and steepness parameter. For the purpose of these bounds, these parameters are samples from a uniform distribution defined on the interval $[-a, a]$ for some positive real number a.

2.7 Uniform finite approximate closure of the SILL dictionary: Step 2

This section simultaneously addresses the approximation of two related mathematical objects.

1. The nonlinear combination of SILL dictionaries in Equation (2.27) with a linear combination.

2. The bilinear combination in Equation (2.30) with the Lie derivatives of a conjunctive logistic function.

So, there are two distinct approximations, one for Corollary 1 and one for Corollary 2. The explicit approximations are given in Table 2.2. In the sense of the steps shown in Fig. 2.2, each of these approximations is a step closer to the Koopman model. The relationship between these approximations, Corollaries 1 and 2, and the uniform finite approximate closure of the SILL dictionary is schematically depicted in Fig. 2.2.

To show uniform finite approximate closure, we need to characterize the error,

$$\epsilon(x) = \frac{d\psi(x)}{dt} - K\psi(x). \tag{2.32}$$

We do so by choosing a specific approximation for the time derivative of each nonlinear dictionary function in the SILL basis.

Characterizing the closure of a dictionary means understanding the error between the Koopman model built with that dictionary and the true Lie derivative of each dictionary function. In Section 2.6 we used mathematical analysis to show that the approximation error of some models will go to zero in the limit of high steepness. Proposition 1 shows that this is enough to reach closure in the limit of infinite steepness and a piecewise

constant vector field. But, when the vector field is analytic, these models do not fully bridge the gap between our Koopman model and the true Lie derivative. This section characterizes additional approximation errors to fully connect our Koopman model to the true Lie derivative. Then, we uniformly bound those errors to show the uniform finite approximate closure of the SILL dictionary.

For all $l \in \{1, 2, ..., N_L\}$, the error of approximating Eq. (A.10) with Eq. (2.29) is:

$$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \lambda(x_i; \theta_{li}) \Lambda(x; \theta^*), \tag{2.33}$$

and the error of approximating Eq. (A.9) with Eq. (2.30) is:

$$\sum_{i=1}^{n} \sum_{j=1}^{N_L} \alpha_{li} w_{ij} \lambda(x_i; \theta_{li}) \Lambda(x; \theta_l) \Lambda(x; \theta_j). \tag{2.34}$$

To get a grasp on the kind of errors we can expect when modeling with the SILL basis we need to develop a conceptual model of the magnitude of Eq. (2.33) and Eq. (2.34).

We do so by sampling possible parameter and state values from uniform distributions over a symmetric interval and then computing distributions of one dimensional logistic functions. We then do the same for terms in the sums in Equations (2.33) and (2.34), under the assumption that $w_{ij} = 1$. In doing so, we notice an exponential decrease in expected error as the state dimension increases.

2.7.1 Expectation of approximation error vanishes

To understand the exponential decrease in expected approximation error we start with the expected values of a single dimensional logistic function. We do so with parameters and state values sampled from uniform distributions defined on the interval $[-a, a]$. We choose this statistical model for how our data and parameters are sampled, because 1) the

data and parameters are assumed to belong to a bounded continuum, and 2) the uniform distribution is the maximum entropy distribution for a continuous random variable on a finite interval. Since our error terms are weighted sums of products of logistic functions we, under the assumption of independence, can compute the expected value of our error terms as a weighted sum of the distinct expected values of logistic functions. This is justified by the linearity of and product rule of expectation under independence.

We cannot explicitly compute the probability density function (PDF) of our logistic functions, so, we compute the values of these integrals numerically. Intermediate steps and details of this approximation are in Section A.2 of the Appendix. In Fig. 2.4 we show their calculated expected values and variances for symmetric uniform distributions with different values of a.

We find that the expected value of a logistic function will be $1/2$ regardless of the sampling interval (see Fig. 2.5). Its variance, as we sample in a wider interval, tends to the functional extremes of zero and one. This is favorable for the linearity of our approximation since, for all $\varepsilon \in (0, 0.5]$, $(0.5-\varepsilon)(0.5+\varepsilon) = 0.25-\varepsilon^2 < 0.25 = (0.5)^2$. So, products of more extreme samples are lower in value than products of samples near the expected value. This demonstrates that our bound, and the error bounds that follow, computed using the expected value of $1/2$, are conservative.

We approximate a single term in the sum of the error function and extrapolate via the sum and product rule of expectation under the assumption of independence to see how nearly linear our approximation is (see Section A.2 of the Appendix). We can conservatively bound the expectation of the approximation error as a product that decreases exponentially with the state dimension. The expected value of a conjunctive logistic function is

$$E[\Lambda] < \frac{1}{2^n}. \tag{2.35}$$

We now compute the expected error given the expected value of the logistic and conjunctive logistic functions. The end result is that, applying the independence assumption, one can conservatively expect the error of each summation term in Eq. (2.33) to decrease in the state dimension, n, at a rate bound above by $\frac{1}{2^{n+1}}$ and the error of each summation term in Eq. (2.34) to decrease in n at a rate bound above by $\frac{1}{2^{2n+1}}$. In each case, these error bounds are exponentially decaying in n, the state dimension.

Explicitly, these intermediate error bounds are

$$\bar{B}_2 = \sum_{i=1}^{n}\sum_{j=1}^{N_L} \frac{\nu_{ij}}{2^{n+1}} \text{ and } \tilde{B}_1 = \sum_{i=1}^{n}\sum_{j=1}^{N_L} \frac{\nu_{ij}}{2^{2n+1}}, \tag{2.36}$$

where $\nu_{ij} \in \mathbb{R}$, and each $\nu_{ij} < a$ as each α_{li} is sampled from the interval $[-a, a]$ and w_{ij} is assumed to be 1. A summary is given in Table 2.2.

2.7.2 Error of the final model

Theorem 1 shows that in the limit of infinitely steep dictionary functions, the error of approximating the product of two conjunctive logistic functions with a single one goes to zero exponentially. Theorem 1 gives us two approximation corollaries that share exponentially decreasing error. Likewise, our probabilistic bound demonstrated that the error in approximating the Lie derivative of SILL functions as a Koopman model goes to zero exponentially, on average, as the state dimension increases. This means that the error, $\epsilon_l(x)$, of the approximation

$$\begin{aligned} \dot{\Lambda}(x;\theta_l) &= \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij} (1 - \lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)\Lambda(x;\theta_j) \\ &\approx \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij} \Lambda(x;\theta^*) \end{aligned} \tag{2.37}$$

will be bound by the sum $\bar{B}_1 + \bar{B}_2$, as well as the sum $\tilde{B}_1 + \tilde{B}_2$[1]. This means that the error will be uniformly bounded by the constant

$$\min\{\bar{B}_1 + \bar{B}_2, \tilde{B}_1 + \tilde{B}_2\} \triangleq B > 0. \tag{2.38}$$

The error will exponentially go to zero with increasingly steep logistic functions and increasing state dimension.

Remark 1. *Consider a data-driven scenario, where the true dimension of the state is unknown but there are multiple ways to take measurements of the system. In such a scenario there may be a trade-off between the cost of taking higher dimensional measurements (eg. through more expensive equipment or more extensive experiments) and the quality of the final model. The error bounds tied to the state dimension can be used to choose the lowest dimension of measurements needed to build a satisfactory Koopman model.*

The bound, B, on our closure error, $\epsilon(x)$, goes to zero, $B \to 0$, at the extremes of high state dimension and steep dictionary functions. This bound shows that the SILL dictionary satisfies uniform finite approximate closure. Given that A_2 is the approximation on row 1 of Table 2.1, then we have that

$$|\dot{\Lambda}_l - A_2| < B(\alpha, n) \leq \bar{B}_1(\alpha, n) + \bar{B}_2(\alpha, n) \to 0 \tag{2.39}$$

exponentially as α and n increase when the state is not in alignment with the center parameters, $x \notin M_{\bar{\Lambda}}$, and $|\theta|$, $|K|$ are bounded (see Footnote 1). These are reasonable assumptions as the set $M_{\bar{\Lambda}}$ is of measure zero in $\mathbb{R}^n$ and because $|\theta|$, $|K|$ being bounded

[1]If $|\dot{\Lambda} - A_1| < \bar{B}_1$ and $|A_1 - A_2| < \bar{B}_2$, where A_1 is the limiting approximation in Eq. 2.27 and A_2 is the approximation on row 1 of Table 2.2. Then $|\dot{\Lambda} - A_2| < \bar{B}_1 + \bar{B}_2$ by the triangle inequality. This argument applies to $\tilde{B}_1$ and $\tilde{B}_2$ without loss of generality where A_1 is the approximation on row 2 of Table 2.2 and A_2 is Eq. (2.29).

is required for any numerical application. Note that $B(\alpha, n)$ is constant given a set of possible state values and a bound on the model parameters. It only varies in model parameters and hyperparameters.

In practice, these limiting conditions of extreme steepness and state dimension are not required. This constant in state dimension, B, appears to be a wide upper bound. One captures strong nonlinear system features using the SILL basis for a low dimensional dictionary and nonlinear system with no restrictions on the steepness parameter [31].

The empirical results in [31] as well as Section 2.9 suggest that there are more nuanced relationships between the steepness, center and weight parameter values and the error of a Koopman model than the uniform bound we have demonstrated in this paper. In these scenarios there is the additional consideration of choosing dictionary functions whose linear combination approximates the underlying vector field. As of now, we rely on numerical optimization to find this balance, but there remains untapped research potential in determining what the ideal parameter combinations should be. For a more detailed commentary on this point, see Remark 2 in Section 2.8.3.

2.8 Uniform finite approximate closure of the AugSILL dictionary

In this section we analyze the subspace invariance properties of a mixed dictionary, the augSILL dictionary. To do so, we characterize the error of Eq. (2.32) for the augSILL dictionary to show that this dictionary satisfies uniform finite approximate closure.

In Sections 2.8.1 and 2.8.2 we establish theorems analogous to Theorem 1 and its corollaries for the augSILL basis. In Section 2.8.3 we demonstrate probabilistic results that combine with the corollaries in Section 2.8.2 to uniformly bind average error of Eq.

(2.32) with a constant, B, that can be arbitrarily small. This approach applies to the SILL and augSILL dictionaries to show uniform finite approximate closure.

2.8.1 Convergence in Steepness of Bilinear AugSILL Terms to the Span of the AugSILL Dictionary

When we construct a mixed dictionary and compute its Lie derivatives we find mixed bilinear terms involving both types of nonlinear dictionary functions. On the path to demonstrating uniform finite approximate closure, we need to approximate these bilinear terms with a single element of our dictionary.

In Section 2.6 we demonstrated that the product of two conjunctive logistic functions can be well approximated by a single conjunctive logistic function. Here we approximate the product of a conjunctive logistic and RBF. To make this approximation, we first define a decision function, H, as follows:

$$H(x;\theta_l,\theta_k) \triangleq \begin{cases} P(x;\theta_k) \text{ if } \mu_{li} < \mu_{ki}, \text{ for } i = 1,2,...,m \\ 0 \text{ otherwise,} \end{cases} \tag{2.40}$$

where $l \in \{1,2,...,N_L\}$ and $k \in \{N_L+1, N_L+2, ..., N_L+N_R\}$.

The decision function, H, uses the relative centers of Λ and P, μ_l and μ_k, to choose to take on the value of the conjunctive RBF (P) or zero. Both functions are in the augSILL dictionary as the zero function is trivially available.

To approximate the product of a conjunctive logistic and RBF, (see Fig. 2.3**c1-c2**) we intend to show that:

$$\Lambda(x;\theta_l)P(x;\theta_k) \approx H(x;\theta_l,\theta_k). \tag{2.41}$$

Eq. (2.41) is provably quite a poor approximation under the wrong conditions, for

example, when elements of α_l are negative. However, the following theorems will demonstrate, that when each element of α is positive, this is a reasonable approximation. This is argued by showing that when the elements of α approach infinity, the error of the approximation approaches zero exponentially for all but a set of points that lie on $n(N_L + N_R)$, $n-1$ dimensional hyperplanes in $\mathbb{R}^n$. These are the points where the state values exactly match up with the function centers. As we noted in Section 2.6, this collection of hyperplanes has Lebesgue measure zero in $\mathbb{R}^n$.

In the case where the center of $\Lambda(x;\theta_l)$ is less than the center of $P(x;\theta_k)$ in all dimensions we have that:

$$\begin{aligned}\varepsilon_{\Lambda_l P_k}(x) \triangleq \Lambda(x;\theta_l)P(x;\theta_k) - H(x;\theta_l,\theta_k) \\ = (\Lambda(x;\theta_l) - 1)P(x;\theta_k).\end{aligned} \tag{2.42}$$

Theorem 2 demonstrates that Eq. (2.42) goes to zero exponentially in the limit of the steepness parameters, α. This theorem's purpose is to argue that the decision function, H, approximates that product of a conjunctive logistic and RBF when the center parameters of the RBF are more positive than those of the logistic in each dimension. Since H, under these circumstances, decides to approximate this product with the conjunctive RBF to begin with, the approximation is already present in the augSILL dictionary. This fact will allow us to argue the subspace invariance of this dictionary further down the line.

Given an augSILL dictionary, $\psi(x)$, we define a set of $n(N_L + N_R)$, $n-1$ dimensional hyperplanes in R^n corresponding to the centers of each $M_{\bar{\Lambda},\bar{P}} \triangleq \{x : \text{there exists } i \in \{1,2,...,n\} \text{ and } j \in \{1,2,...,N_L+N_R\} \text{ so that } x_i = \mu_{ji}\}$. This defines specific state values where the approximations in Eq. (2.41) and (2.44) have maximal error.

Theorem 2. *When the state does not exactly align with the centers of the dictionary functions in any dimension and the center of the conjunctive RBF is more positive than*

the center of the conjunctive logistic function in each dimension, then their product exponentially converges to the conjunctive RBF as their steepness parameters go to infinity. Specifically, if $x \notin M_{\bar{\Lambda},\bar{P}}$ *and for all* $i = 1, 2, ..., n$, $\mu_{ki} \geq \mu_{li}$, *then as* $\alpha \to \infty$, $\Lambda(x;\theta_l)P(x;\theta_k) - H(x;\theta_l,\theta_k) \to 0$ *exponentially.*

Proof: As a preliminary, we note that $-1 \leq (\Lambda(x;\theta_l) - 1) \leq 0$ for any $x \in \mathbb{R}^n$ and for $\alpha > 0$.

We first show that when $\mu_{ki} \geq \mu_{li}$ $P(x;\theta_k) \to 0$ as $\alpha \to \infty$. We do so by considering the two possible cases for any i where $\mu_{ki} \geq \mu_{li}$.

Case 1, $x_i > \mu_{ki}$. In this case as $\alpha \to \infty$ we have that $e^{-\alpha_{ki}(x_i-\mu_{ki})} \to 0$ exponentially, and that $(1 + e^{-\alpha_{ki}(x_i-\mu_{ki})})^2 \to 1^2 = 1$. Thus the i^{th} term of $P(x;\theta_k)$ will go to 0 exponentially.

Case 2, $x_i < \mu_{ki}$. In this case as $\alpha \to \infty$ we have that $\frac{e^{-\alpha_{ki}(x_i-\mu_{ki})}}{(1+e^{-\alpha_{ki}(x_i-\mu_{ki})})^2} \to 0$. Thus the i^{th} term of $P(x;\theta_k)$ will go to 0 exponentially.

Each of these cases implies that $P(x;\theta_k)$ and therefore Eq. (2.42) goes to 0 exponentially. ■

Theorem 2 implies that the bilinear augSILL term, ΛP, a product of two dictionary functions, is well approximated by P, a single dictionary function when P's center parameters are more positive than Λ's. This fact will be key to showing that bilinear terms in the Lie derivatives of augSILL dictionary functions are approximated as linear terms in Section 2.8.2.

Theorem 3 implies that ΛP is well approximated by 0, when at least one of P's center parameters are more negative than Λ's. This, of course, is assuming that all the steepness parameters of these dictionary functions are positive, since the approximation is evaluated in the limit of high steepness.

When the center of $\Lambda(x;\theta_l)$ is greater than the center of $P(x;\theta_k)$ in one or more

dimensions we have that:

$$\begin{aligned}\varepsilon_{\Lambda_l P_k}(x) &\triangleq \Lambda(x;\theta_l)P(x;\theta_k) - H(x;\theta_l,\theta_k) \\ &= \Lambda(x;\theta_l)P(x;\theta_k)\end{aligned} \tag{2.43}$$

Theorem 3 demonstrates that the approximation in Eq. (2.43) is a good approximation in the limit of α.

Theorem 3. *When the state does not exactly align with the centers of the dictionary functions in any dimension and the center of the conjunctive logistic function is more positive than the center of the RBF in all dimensions, then their product exponentially converges to zero as their steepness parameters go to infinity. Specifically, if $x \notin M_{\bar{\Lambda},\bar{P}}$ and $x_i \neq \mu_{li}$ for all $i \in \{1,2,...,m\}$ so that $\mu_{ki} < \mu_{li}$, then as $\alpha \to \infty$,*

$$\Lambda(x;\theta_l)P(x;\theta_k) - H(x;\theta_l,\theta_k) \to 0$$

exponentially.

Proof: We assume that $\mu_{ki} < \mu_{li}$ and note that, in each case as $\alpha \to \infty$ the i^{th} term in Eq. (2.43) goes to zero as shown in Case 2 of the proof of Theorem 2. Thus the product of these terms will go to zero as $\alpha \to \infty$. For the sake of brevity we forgo the explicit computation of the limits which follow and speak in more general terms of rate of growth. We note that when we use "∞_c" we mean to say that the term grows to infinity with α with a rate of $e^{c\alpha}$, for some positive constant c. Furthermore, we use "0_c" to mean that the term goes to zero with a rate of $e^{-c\alpha}$, for some positive constant c.

Case 1, $x_i > \mu_{ki}$ *Sub-Case 1.1*, $x_i > \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{0_{c_k}}{1} = 0$. *Sub-Case 1.2*, $x_i = \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{0_{c_k}}{2} = 0$. *Sub-Case 1.3*, $x_i < \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{0_{c_k}}{\infty_{c_l}} = 0$.

Case 2, $x_i < \mu_{ki}$, thus $x_i < \mu_{li}$ and so as $\alpha \to \infty$ our i^{th} term goes to $\frac{\infty_{c_k}}{\infty^2_{c_k}\infty_{c_l}} = 0$. *Sub-Case 2.1*, $x_j > \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{\infty_{c_k}}{\infty^2_{c_k}} = 0$. *Sub-Case 2.2*, $x_i = \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{\infty_{c_k}}{2\infty^2_{c_k}} = 0$. *Sub-Case 2.3*, $x_i < \mu_{li}$ so as $\alpha \to \infty$ our i^{th} term goes to $\frac{\infty_{c_k}}{\infty^2_{c_k}\infty_{c_l}} = 0$.

Now we note that, in each case as $\alpha \to \infty$ the i^{th} term in Eq. (2.43) goes to zero exponentially. Thus the product of these terms will go to zero exponentially as $\alpha \to \infty$.

■

Between Theorems 2 and 3 we have that $\Lambda P \approx H$. Specifically, $\Lambda(x;\theta_l)P(x;\theta_k) \approx P(x;\theta_k)$ when $\theta_l \lesssim \theta_k$ and 0 otherwise. The only pathological case excluded from these theorems is when the conjunctive logistic and conjunctive RBF centers exactly match in some dimension. Distance from the pathology becomes relevant when steepness parameters, α, are too small.

Now, we approximate the product of two conjunctive RBFs, completing all the possible combinations of pairwise products between elements of our mixed basis. The product of two conjunctive logistic functions, $\Lambda_l\Lambda_k$, is covered in Section 2.6. To approximate P_lP_k (see Fig. 2.3**b**), Theorem 4 describes conditions under which

$$\varepsilon_{P_lP_k}(x) \triangleq P(x;\theta_l)P(x;\theta_k) \approx 0. \tag{2.44}$$

Because Theorem 4 shows that the approximation in Eq. (2.44) is a good approximation in the limit of α. It takes the bilinear term, P_lP_k, and approximates it with the zero function which is trivially in all augSILL dictionaries.

Theorem 4. *When the state does not exactly align with the centers of the dictionary functions in any dimension, then the product of two conjunctive RBFs converges exponentially to zero as their steepness increases. Specifically, if* $x \notin M_{\bar{\Lambda},\bar{P}}$*, then as* $\alpha \to \infty$*,* $P(x;\theta_l)P(x;\theta_k) \to 0$ *exponentially.*

The proof of Theorem 4, as well as a supporting lemma is presented in Section A.1 of the appendix.

We now have four approximation theorems for the bilinear mathematical terms which arise when computing the Lie derivatives of augSILL basis functions. We now apply them to these Lie derivatives with a set of approximation corollaries. These corollaries, together with the argument in Section 2.8.3, will demonstrate the uniform finite approximate closure of the augSILL dictionary.

2.8.2 Showing bilinear Lie derivatives can be approximated linearly to satisfy the Koopman generator equation

Corollary 3 approximates the Lie derivative of a conjunctive logistic function in the context of our mixed basis. The approximation that this corollary suggests is not a Koopman model itself (see Fig. 2.1). In Section 2.8.3 we demonstrate uniform finite approximate closure by approximating this intermediate approximation with a full Koopman model.

Corollary 3. *Under the assumptions of Theorem 1 and the assumption that the augSILL observables span the vector field we seek to model, the Lie derivative of a conjunctive logistic function exponentially approaches a nonlinear combination of augSILL functions, specifically,*

$$\begin{aligned}\dot{\Lambda}(x;\theta_l) \to &\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta^*) \\ &+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}(1-\lambda(x_i;\theta_{li}))H(x;\theta_l,\theta_k)\end{aligned} \tag{2.45}$$

exponentially as $\alpha \to \infty$.

Now that we have an intermediate approximation for the Lie derivative of a con-

junctive logistic function we need a similar result for conjunctive RBFs. Once we have these results the analysis in Section 2.8.3 can show a Koopman approximation of the derivatives of these two augSILL functions. This final approximation will be a linear combination of augSILL dictionary functions.

Corollary 4 approximates the Lie derivative of a conjunctive RBF in the context of the augSILL basis.

Corollary 4. *Under the assumptions of Theorem 1 and the assumption that the augSILL observables span the vector field we seek to model, the Lie derivative of a conjunctive RBF exponentially approaches a nonlinear combination of conjunctive RBFs, specifically,*

$$\dot{P}(x;\theta_l) \to \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij} (1 - 2\lambda(x_i;\theta_{li})) H(x;\theta_j,\theta_l) \tag{2.46}$$

exponentially as $\alpha \to \infty$.

Note that Equations (2.45) and (2.46) are not compatible with the Koopman model we seek to learn: $K\psi(x)$, where the dictionary functions, ψ, are augSILL functions and K is a real-valued matrix. In Section 2.8.3 we approximate Equations (2.45) and (2.46) with a mathematical form that is consistent with the approximated Koopman generator, Eq. (2.8).

One can approximate the Lie derivatives of an augSILL dictionary function as a linear combination of *products of pairs* of augSILL functions. The details on approximating these Lie derivatives as a weighted sum of these bilinear terms is detailed in Section 2.8.3.

Below, we approximate this weighted sum of bilinear terms (one of the intermediate approximations in Fig. 2.1) with a weighted sum of augSILL functions. The final weighted sum is of the form of Eq. (2.8), and therefore admits a Koopman operator

model.

Corollary 5. *Under the assumptions of Theorem 1, the sum of products*

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\Lambda(x;\theta_l)\Lambda(x;\theta_j)\\&+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\Lambda(x;\theta_l)P(x;\theta_k)\end{aligned}\tag{2.47}$$

approaches

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\Lambda(x;\theta^*)\\&+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}H(x;\theta_l,\theta_k),\end{aligned}\tag{2.48}$$

a weighted sum of augSILL functions, exponentially as $\alpha \to \infty$.

Proof: This result is a direct consequence of Theorem 1, Theorem 2 and Theorem 3. These approximations are outlined in Fig. 2.3 **(a)**, **(c1)** and **(c2)**. ■

Corollary 6. *Under the assumptions of Theorem 1, the sum of products*

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}P(x;\theta_l)\Lambda(x;\theta_j)\\&+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}P(x;\theta_l)P(x;\theta_k)\end{aligned}\tag{2.49}$$

approaches

$$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}H(x;\theta_j,\theta_l),\tag{2.50}$$

a weighted sum of conjunctive RBFs, exponentially as $\alpha \to \infty$.

Proof: This result is a direct consequence of Theorems 2, 3 and 4. These approximations are outlined in Fig. 2.3 **(c1)**, **(c2)** and **(b)**. ■

The resulting linear combinations from Corollaries 5 and 6 can be stacked and combined into the matrix K (whose entries would be the products of α_{**} and w_{**}). Thus, given an augSILL dictionary that admits a weighted bilinear approximation to the Lie derivatives of its functions, Corollaries 5 and 6 show the uniform finite approximate closure of that dictionary by showing that the error of approximating the weighted bilinear representation with a linear combination of dictionary functions goes to zero exponentially in steepness.

Corollaries 3, 4, 5 and 6 each imply an error bound for approximating the Lie derivative of an augSILL dictionary function with an intermediate approximation. These error bounds are listed explicitly in Table A.2. They combine with other error bounds in the same table to demonstrate uniform finite approximate closure following the argument given in Section 2.7.

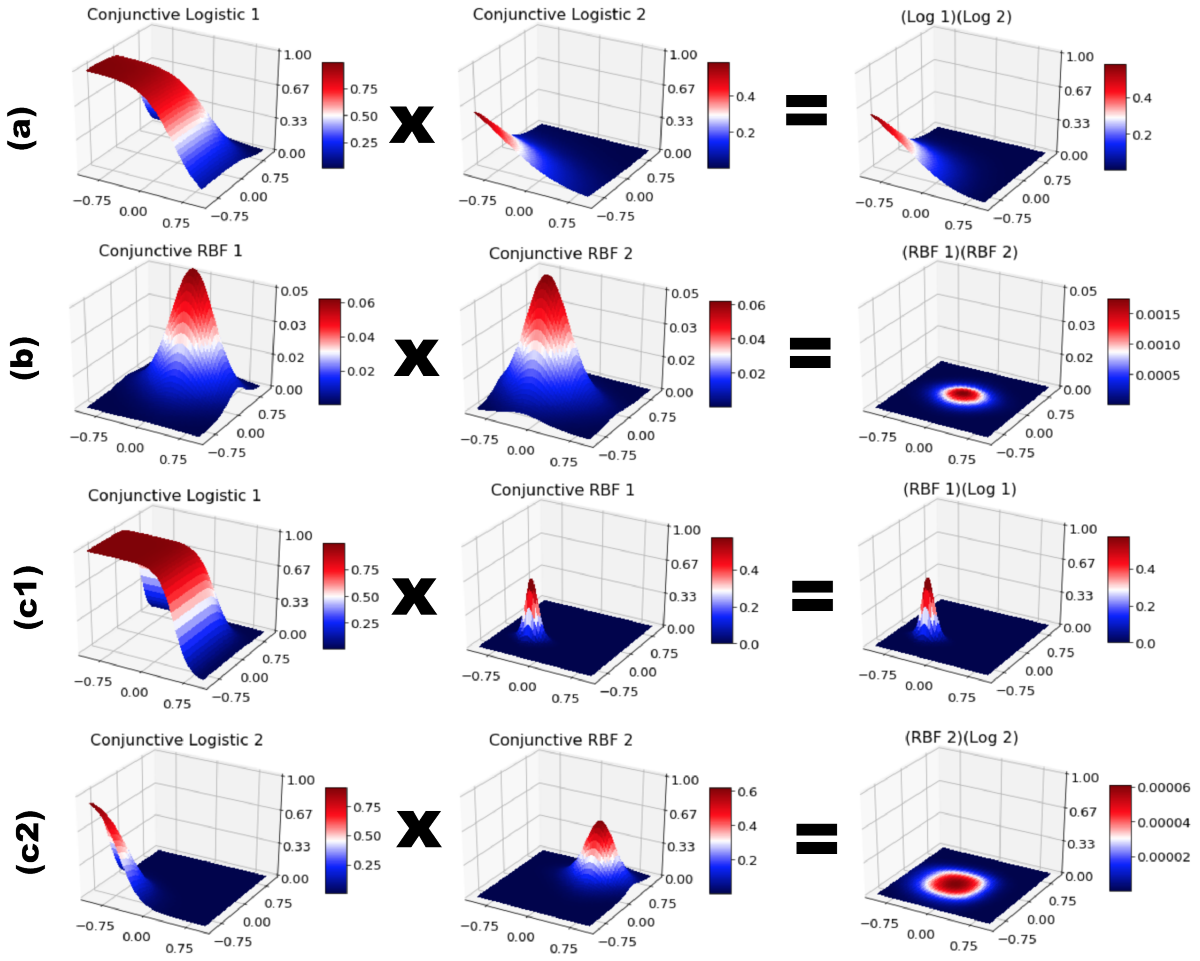

Figure 2.3: Visual representation of Theorem 1 and Theorems 2, 3 and 4. **(a)** Theorem 1: The product of two conjunctive logistic functions approximates the conjunctive logistic function whose centers are greater in each dimension. **(b)** Theorem 4: The product of two conjunctive RBFs is nearly zero unless the norm of the difference between their centers is very small. **(c1)** Theorem 2: The product of a conjunctive logistic function and a conjunctive RBF approximates the RBF if at least one dimension of the center in the logistic function is greater than its partner center in the RBF. **(c2)** Theorem 3: When this is not the case the product is nearly zero.

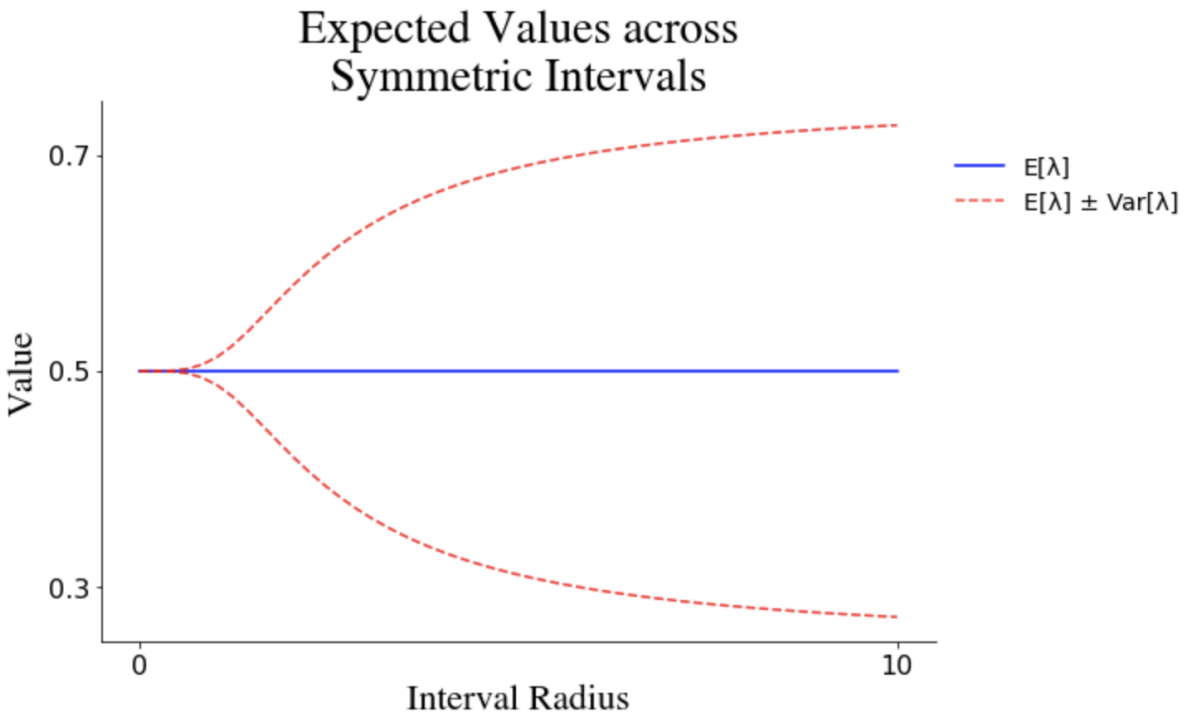

Figure 2.4: Expected values and variances of a logistic function with parameters and state values sampled from symmetric uniform distributions of various interval radii. Note that the expected value is always 1/2.

Fun.	**Ref.**	**Approximation**	**Difference (error)**	**Error bound**
Log.	(2.45)	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\Lambda(x;\theta^*)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}H(x;\theta_l,\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\lambda(x_i;\theta_{li})\Lambda(x;\theta^*)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\lambda(x_i;\theta_{li})H(x;\theta_l,\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{n+1}}$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{3n+1}}$
Log.	(A.16)	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\Lambda(x;\theta_l)\Lambda(x;\theta_j)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\Lambda(x;\theta_l)P(x;\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\lambda(x_i;\theta_{li})\Lambda(x;\theta_l)\Lambda(x;\theta_j)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\lambda(x_i;\theta_{li})\Lambda(x;\theta_l)P(x;\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{2n+1}}$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{3n+1}}$
RBF	(2.46)	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}H(x;\theta_j,\theta_l)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}2\alpha_{li}w_{ij}\lambda(x_i;\theta_{li})H(x;\theta_j,\theta_l)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{3n+1}}$
RBF	(A.20)	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\Lambda(x;\theta_j)P(x;\theta_l)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}P(x;\theta_i)P(x;\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\lambda(x_i;\theta_{li})\Lambda(x;\theta_j)P(x;\theta_l)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\lambda(x_i;\theta_{li})P(x;\theta_i)P(x;\theta_k)$	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{3n+1}}$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{4n+1}}$

Table 2.2: Approximations to and properties of error bounds for the four equations referred to in the **Ref.** column. The reference equation is approximated as the corresponding equation in the **Approximation** column. We give the error of this approximation in the **Difference (error)** column. The **Error bound** column gives a bound on this error. The **Fun.** column refers to the type of dictionary function approximated in the row. The right side of Fig. 2.2 shows where these approximations fit into showing uniform finite approximate closure. The integer n is the dimension of the state, x. Each constant ν_{ij} is bounded by the product of a weight and steepness parameter. For the purpose of these bounds, these parameters are samples from a uniform distribution defined on the interval $[-a, a]$ for some positive real number a.

2.8.3 Expectation of approximation error vanishes

This section simultaneously addresses the approximation of two related mathematical objects.

1. It addresses the approximation of Equation (2.45) with (2.48) and Equation (2.46) with Equation (2.50). This is the lower left step in the right side of Fig. 2.1.

2. It also addresses the approximation of Equations (2.47) and (2.49) with the Lie derivatives of conjunctive logistic and RBFs respectively. This is the upper right step in the right side of Fig. 2.1.

In total, there are four distinct approximations, see Table 2.2. Each case, as illustrated in Fig. 2.1, our approximation is a step closer to the Koopman model. We either go from:

1. an intermediate approximation of the Lie derivative that is a nonlinear combination of dictionary functions to a linear combination of the same dictionary functions, or

2. the Lie derivative itself to a bilinear intermediate approximation.

To understand our approximation error we compute the expected values of a single dimensional logistic and RBF. We do so with parameters and state values sampled from uniform distributions defined on the interval $[-a, a]$. We choose this statistical model for how our data and parameters are sampled, because 1) the data and parameters are assumed to belong to a bounded continuum, and 2) the uniform distribution is the maximum entropy distribution for a continuous random variable on a finite interval. Since our error terms are weighted sums of products of these functions we, under the assumption of independence, estimate the expected value of our error terms via the linearity and product rule of expectation.

We cannot explicitly compute the probability density function (PDF) of our logistic and RBFs, so, we compute the values of these integrals numerically. Intermediate steps and details of this approximation are in Section A.2 of the Appendix. The only difference is that an RBF is substituted for a logistic function. In Fig. 2.5 we show their calculated expected values and variances for symmetric uniform distributions with different values of a.

We find that the expected value of a logistic function will be $1/2$ (see Fig. 2.5). Its variance, as we sample in a wider interval, tend to the functional extremes of zero and one. This is favorable for the linearity of our approximation since, for all $\varepsilon \in (0, 0.5]$, $(0.5-\varepsilon)(0.5+\varepsilon) = 0.25 - \varepsilon^2 < 0.25 = (0.5)^2$. So, products of more extreme samples are lower in value than products of samples near the expected value. The expected value of an RBF will be no greater than $1/4$, and it decreases as a increases.

We approximate a single term in the sum of the error function and extrapolate via the sum and product rule of expectation under the assumption of independence to see how nearly linear our approximation is (see Section A.2 of the Appendix). We can conservatively bound the expectation of approximation error as a product that decreases exponentially with the state dimension. The error bounds for a conjunctive logistic and RBF are

$$E[\Lambda] < \frac{1}{2^n} \text{ and } E[P] < \frac{1}{4^n} = \frac{1}{2^{2n}}. \tag{2.51}$$

Since, H will be a conjunctive RBF in $\frac{1}{2^n}^{th}$ of the state-parameter space, its weight of decrease can be bounded by

$$E[H] < \left(\frac{1}{2^{2n}}\right)\left(\frac{1}{2^n}\right) = \frac{1}{2^{3n}}. \tag{2.52}$$

We record the full error terms and bounds in Table 2.2.

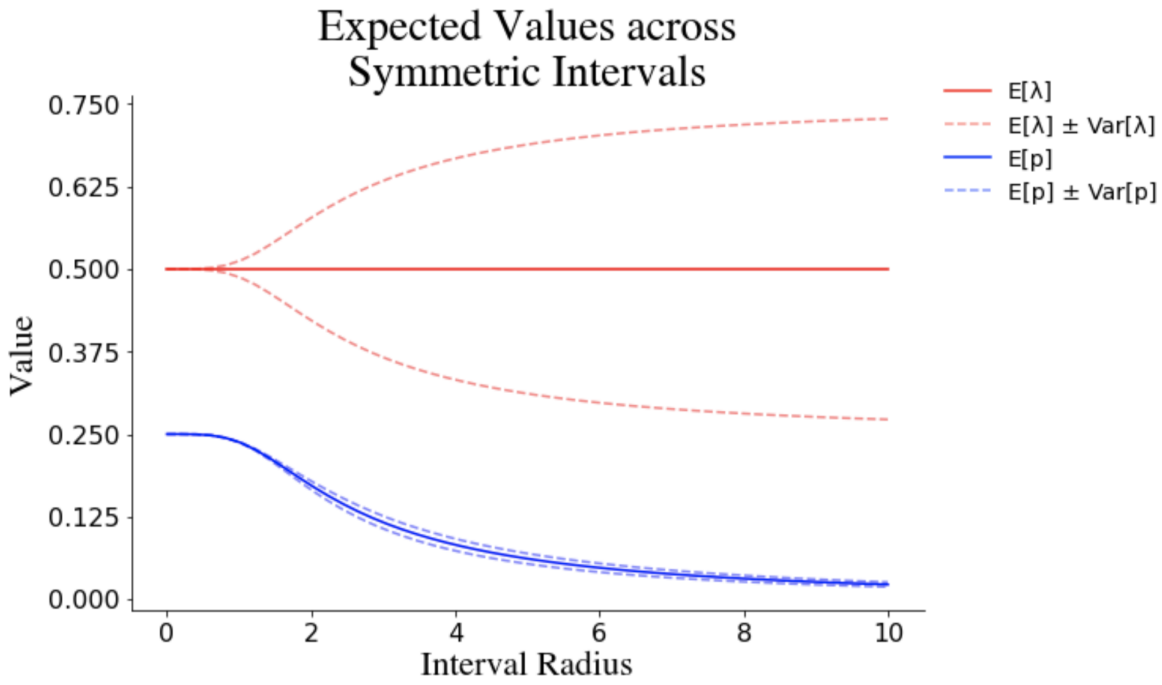

Figure 2.5: Expected values and variances of logistic and RBFs with parameters and state values sampled from symmetric uniform distributions of various interval radii. Note that the expected value of a logistic function is always 1/2, and that of the RBF is bounded above by 1/4.

These bounds are combined with the bounds that arise from Corollaries 1, 2, 3 and 4 to bound the approximation error of a Koopman model using the augSILL dictionary. A summary of all these bounds is given explicitly in Table A.2 of the Appendix.

In summary, the Koopman model approximation error is as well or better behaved for augSILL than SILL dictionaries. So, there is a uniform bound for augSILL dictionaries, much like there is for a standard SILL dictionary. This means that augSILL models must satisfy uniform finite approximate closure.

It is reasonable to assume that (1) the state value does not belong to the set $M_{\bar{\Lambda},\bar{P}}$, a subset of $\mathbb{R}^n$ of measure zero, and (2) the dictionary parameters and Koopman approximation matrix are bounded. Under these assumptions, augSILL dictionaries, in the limit of an increasing state dimension and increasing steepness of their dictionary functions,

have that their bounding constant, $B > 0$, approaches zero in expectation, $B \to 0$. Since the uniform bound on the error of this model can be arbitrarily small, the augSILL dictionaries can be used to build accurate Koopman models for any dynamic system of the from of Eq. (2.1).

Remark 2. *The empirical results in [31] as well as Section 2.9 suggest that there are more nuanced relationships between the steepness (α), center (μ) and weight (w) parameter values and the error of a Koopman model than the uniform bound we have demonstrated in this paper. In these scenarios there is the additional consideration of choosing dictionary functions whose linear combination approximates the underlying vector field. As of now, we rely on numerical optimization to find this balance. In Section 2.9 we characterize the error term, ϵ, from Equation (2.32), as the mean-squared error of two types of dictionary lifted data points: those from a future time-step and those from a present time-step after matrix multiplication with an approximate Koopman operator (plus regularization terms specific to the learning algorithm). The log of this error term is plotted in the first row of Figure 2.10. However, on the theoretical side there remains untapped research potential in determining what the ideal parameter combinations should be.*

For example, as steepness (α) increases, the conjunctive logistic functions in the dictionary will approach an $n-$dimensional step function, like that given in Equation (2.17). Finite linear combinations of step functions have provable dictionary closure properties when they are defined over a vector field of sums of step functions, see Proposition 1 of Section 2.4. So, when our dynamics are in the span of our dictionary (Assumption 2), we can see how some of those nice properties stay for conjunctive logistic functions. However, the theoretical results in this paper do not consider the case where the dynamics of the system are not exactly in the dictionary's span.

Similarly, as steepness (α) increases, the conjunctive radial basis functions in the

dictionary will approach an $n-$dimensional impulse function whose support can be moved by changing the center (μ) parameters. Finite linear combinations of such a function would be ill-suited for approximating any smooth hypersuface in $\mathbb{R}^n$, even though they have excellent subspace closure properties for any system that is a sum of impulse functions.

To summarize, in practice, the modeler may find finite model parameter values and finite dimensional models that work together to build a model whose error improves on the finite versions of the bounds that we have built. This is in contrast to the infinite dimensional models with infinite steepness parameters that we have produced and analyzed in the limit. However, finding those parameters is a non-convex problem. Connecting this work to choosing optimal parameters for a given class of system identification problem is an outstanding problem that future research will need to address.

2.9 Numerical examples

To come full circle, we need to reexamine deepDMD and compare it to heterogeneous dictionary models. DeepDMD uses a feedforward neural network to simultaneously parameterize the matrix K as well as the dictionary functions $\psi(x)$. The deepDMD algorithm has built accurate predictive models and its dimension ($N \in \mathbb{Z}^+$) scales well with that of the modeled system [1]. We build a novel comparison of deepDMD and a much lower-parameter model built from the augSILL basis. The lower parameter augSILL model learns as quickly and accurately as the deep-learning-based model.

To explain why algorithms like EDMD have variable success, we also contribute a head to head comparison of five dictionaries for Koopman learning. We see the deep-learning-inspired dictionaries vastly outperform orthogonal polynomial dictionaries. This suggests that issues with algorithms like EDMD may be resolved by selecting a dictionary proven to satisfy uniform finite approximate closure.

Unless specified otherwise we use simulated data generated from uniformly distributed initial states run with SciPy's ODE integration software. Each of these results is concerned with the discrete-time, data-driven problem statement. The system's state is directly measured at even time intervals.

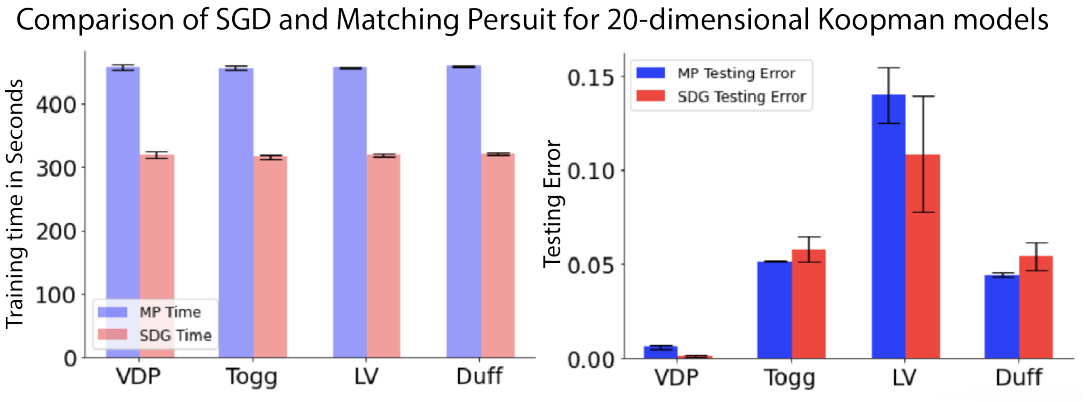

Figure 2.6: A comparison of the full matching pursuit algorithm to SGD on four nonlinear systems. We plot 5-step prediction error; see the paragraph containing Equation (2.53) for an explanation of this error. The SGD algorithm (in red) shows consistently better temporal scaling as well as comparable 5-step prediction error.

2.9.1 Choosing center and steepness parameters

Even when a dictionary class and model dimension are selected, each individual problem will warrant a unique parameterization of the dictionary. Given a dictionary, such as the augSILL dictionary, how do we choose the parameters of each dictionary function? We consider two algorithms, matching pursuit and stochastic gradient descent (SGD).

Matching pursuit [32] considers an expansive list of potential dictionary functions and greedily adds the function that lowers the value of the objective function (Eq. (2.10)) most. Matching pursuit adds one function at a time to the model.

We use SGD to attack a host of non-convex optimization problems. It is famous, in part, because of its use in training artificial neural networks. SGD can be directly

applied to parameterize a fixed number of dictionary functions from data, much like it learns the parameters of a neural network.

Which algorithm do we choose?

We compared two variations of matching pursuit, as well as SGD for learning augSILL models of four dynamic systems, the Van der Pol oscillator, the Duffing oscillator, the Lokta-Volterra model and the Gardner-Collins genetic toggle switch. The specific parametrization of these systems is given in Section A.3 of the Appendix.

The training data was generated from 100 random initial points uniformly sampled from the interval $[0, 0.5]$ for each system except for the Lotka-Volterra model, which was sampled from the interval $[1, 5]$. Samples from each trajectory were taken for $t = 0, 1, 2, ..., 14$, resulting in a total of $100 \times 15 = 1500$ data points per system for each trial. The testing data was generated from 50 random initial points uniformly sampled from the interval $[0, 1]$ for each system except for the Lotka-Volterra model, which was sampled from the interval $[1, 9]$. Samples from each trajectory were taken at the same times as the training data resulting in $50 \times 15 = 750$ data points per system for each trial. We ran 15 trials total per system.

We found that the full matching pursuit algorithm and SGD had similar testing error for a 20 dimensional Koopman operator (see Fig. 2.6). Testing error is defined to be half the 2-norm of the difference between the trajectory values at time $t + 5$, $x(t + 5)$, and the Koopman model predicted values at time $t + 5$,

$$x_{pred}(t + 5) \triangleq \begin{bmatrix} I_{n\times n} & 0 \end{bmatrix} K^5 \psi(x(t)), \tag{2.53}$$

where $n = 2$ for each of these four systems. We focus on accuracy and performance for 20 dimensional models as a step-in for modeling higher dimensional systems. Also, we note

that SDG was about 1.5 times faster when building a 20 dimensional model. We defined accuracy as the error between predictions Since SGD was the better choice for building larger Koopman models in terms of time to execute, and performed comparably in 5-step prediction error, we compare this algorithm to deepDMD. Since deepDMD utilizes SGD, we can compare model accuracy at each training epoch.

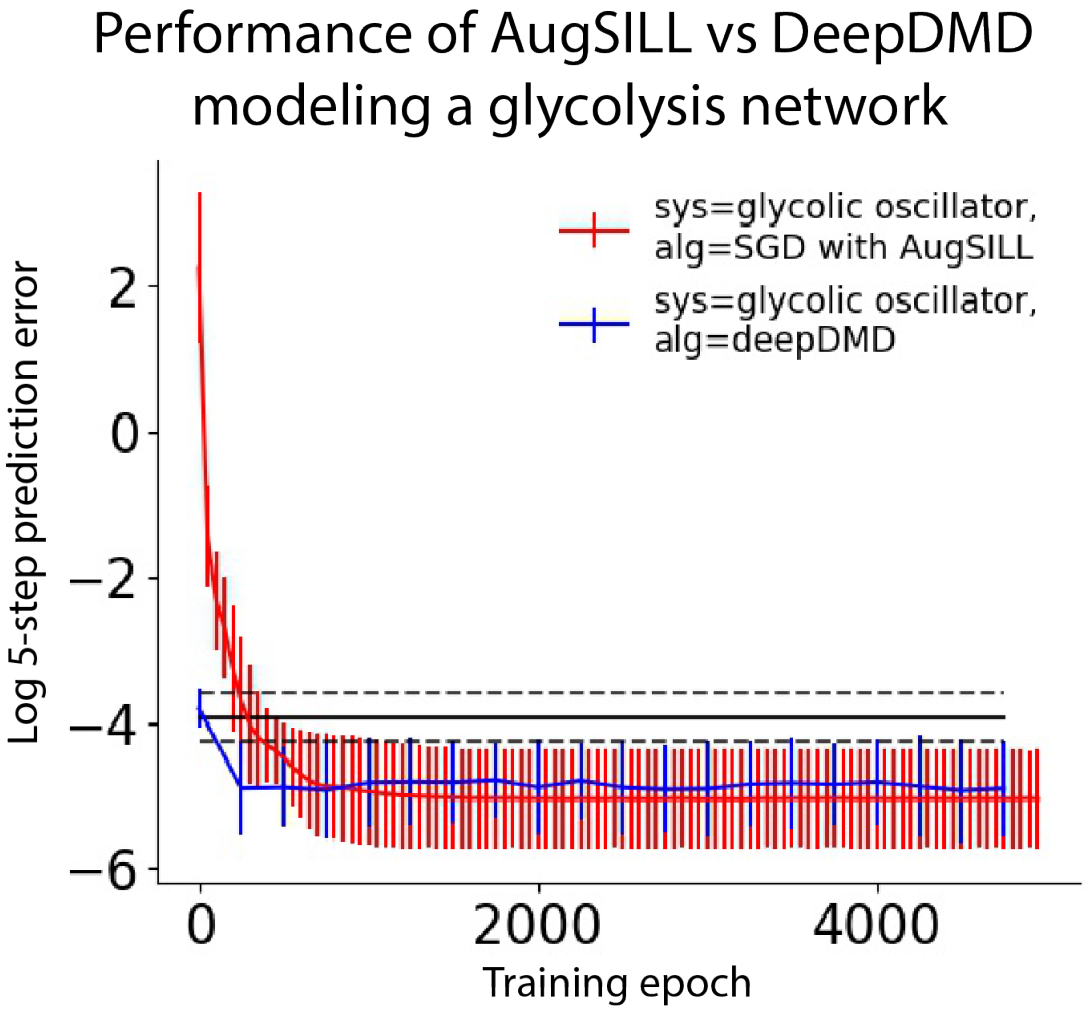

Figure 2.7: A comparison deepDMD to SGD with the augSILL basis on the seven dimensional model of glycolysis given in [33]. The solid horizontal bar is the mean performance of the DMD algorithm, the dotted bars are one standard deviation above and below this mean. The spines are error bars for the measured epochs. We plot 5-step prediction error as explained in the paragraph containing Equation (2.53).

2.9.2 AugSILL basis vs deepDMD

The SILL and augSILL dictionaries are a targeted study of the dictionary functions generated from deepDMD. Our visualization of deepDMD observables showed a convergence to sums of SILL and augSILL dictionary observables. Can we use these dictionaries to build a model on par with deep learning?

To challenge ourselves, we used a seven dimensional glycolysis model to generate our testing and training data [33]. This data was all generated from a single initial condition $x_0 = [1, 0.19, 0.2, 0.1, 0.3, 0.14, 0.05]$. From this initial condition the system takes about 3.5 seconds to settle into nonlinear oscillations (which have a period of roughly 1 second). Our training data consists of 700 evenly-spaced samples of the trajectory starting from x_0 from 0 to 7 seconds. The testing data consists of 300 evenly-spaced samples of the same trajectory from 7 to 10 seconds.

From this setup, the augSILL model reaches a comparable 5-step prediction error to deepDMD in under 1000 training epochs and neither significantly changes over the next 4000 epochs, see Fig. 2.7. Note that the model we learn using the augSILL basis has 995 parameters. All in all, the deepDMD model has a total of $3,949$ parameters. All of the augSILL parameters are easily interpreted as center, steepness and weight parameters of a logistic or RBF.

To provide more insight on the performance of these two models we also compare their short and long term predictive performance during the training process as shown in Fig. 2.10. For these specific models learning the glycolysis network, we find that the long-term prediction error stops falling off between 10 and 25 steps of prediction. We also found that the model performed better on its training data up until about 25 steps of prediction, at which point the performance of the train and test data, on average, was nearly the same.

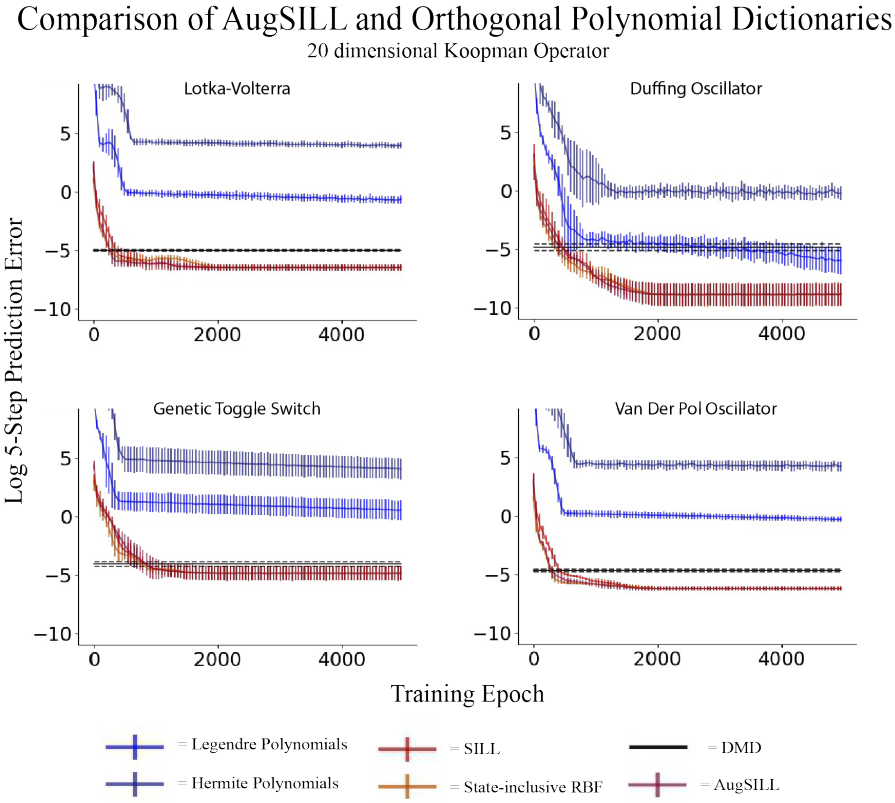

Figure 2.8: Five-step prediction accuracy (mean squared error over all the testing data) vs training epoch of 20-dimensional Koopman learning with SGD using various dictionaries. Plotted on a log scale. Note that the orthogonal (Hermite and Legendre) polynomial basis (in blue) have greater error throughout the training process. The spines are error bars at each epoch where 5-step error was measured.

2.9.3 Comparison of dictionaries for Koopman learning

Now we address the relationship between the choice of dictionary and the success of a Koopman model. What properties do successful Koopman dictionaries have in common? The augSILL basis performed on par with deepDMD. Would we have gotten similar results using orthogonal polynomial dictionaries?

We learn the systems in Eq. (A.28), (A.29), (A.30), and (A.31), parameterized with the SGD algorithm. We do so with the SILL, augSILL and summed one-dimensional RBFs, as well as two different orthogonal polynomial dictionaries (Legendre and Her-

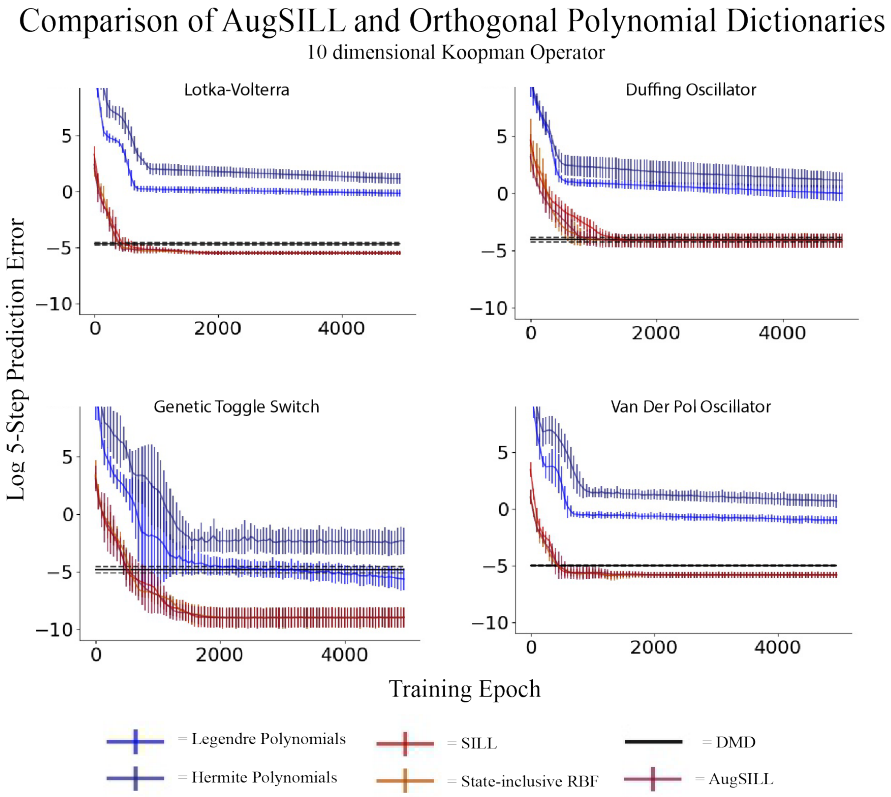

Figure 2.9: Five-step prediction accuracy (mean squared error over all the testing data) vs training epoch of 10-dimensional Koopman learning with SGD using various dictionaries. Plotted on a log scale. Note that the orthogonal (Hermite and Legendre) polynomial basis (in blue) have greater error throughout the training process. The spines are error bars at each epoch where 5-step error was measured.

mite). We compare their 5-step average prediction error for a 5, 10 and 20 dimensional Koopman model of each system.

The training data was generated from 10 random initial points sampled from the interval [0, 0.5] for each system except for the Lotka-Volterra model, which was sampled from the interval [1, 5]. A total of 101 samples, evenly-spaced in time, were taken from $t = 0$ to $t = 5$. The testing data was generated from 5 random initial points sampled from the interval [0, 1] for each system except for the Lotka-Volterra model, which was sampled from the interval [1, 9]. Just as for the training data, a total of 101 samples, evenly-spaced in time, were taken from $t = 0$ to $t = 5$. This gives a total of 1010 training

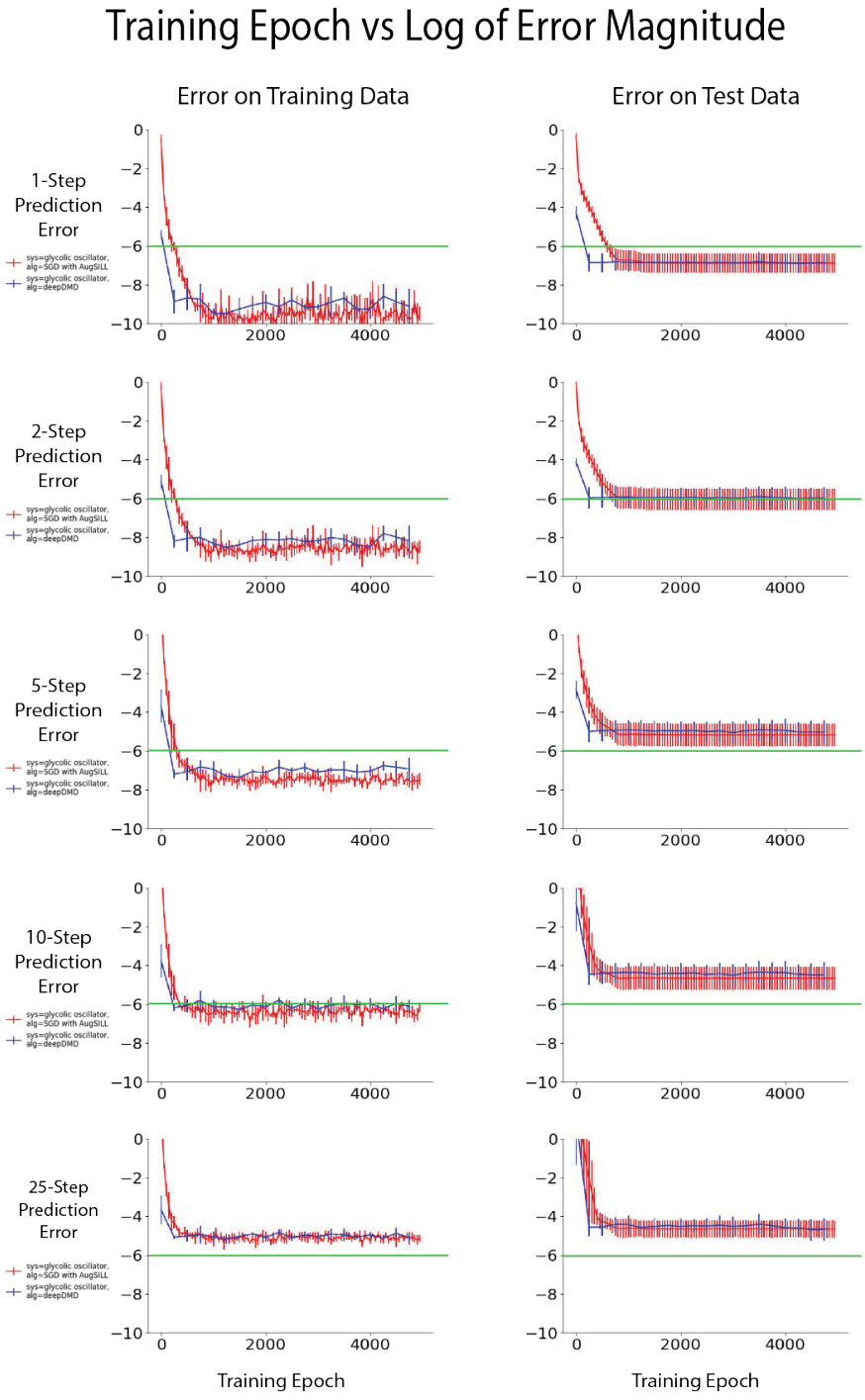

Figure 2.10: Prediction accuracy (mean squared error over all the testing data) of deep-DMD and the augSILL basis over shorter and longer periods of time on training and testing data. The first row of this plot (1-Step Prediction Error) approximates the log of the mean squared error of the discrete-time analog to the error term, ϵ, from Equation (2.32). The green line was added to provide a visual reference and facilitate comparison of the plots, it has no additional significance.

and 505 testing data points for each trial. This was repeated for four trials. All of the models used identical training and testing data, except for standard DMD which was only used to benchmark the results.

For the 5 dimensional models the choice of dictionary seemed mostly irrelevant. However, for a 5 dimensional model, SGD only outperformed standard DMD for the toggle switch. So, we don't have enough system dimensions with these dictionaries to want to use SGD in the first place.

Building a 10 and 20 dimensional Koopman model of each system, we find that the basis inspired from the outputs of deepDMD (SILL, augSILL and summed RBFs) have lower 5-step prediction errors (see Fig. 2.8 and Fig. 2.9). Each of these three basis had nearly identical errors, though the SILL basis can take more training iterations to perform comparably. The Legendre polynomial basis outperformed the Hermite polynomial basis in each case. For the 20 dimensional model of the Van der Pol system the augSILL basis was over 372 times more accurate than the Legendre polynomial basis, for the Duffing Oscillator it was over 221 times more accurate, for the Predator-Prey System it was over 18 times as accurate and for the Toggle Switch it was over 328 times as accurate.

2.10 Conclusion

Learning models in a data-driven setting using human-defined dictionaries results in high-dimensional, over-parameterized representations of what could be simple physical phenomena. DeepDMD and other ANN learning techniques address these issues but at the cost of extensive computational time. Moreover, the dictionaries learned by deepDMD are ad hoc, randomly constructed, and difficult to interpret.

To investigate how deepDMD finds successful Koopman invariant subspaces, we extracted simple features of the dictionary functions learned by deepDMD and looked for

properties to explain their success. We discovered a Koopman dictionary which we call the State-Inclusive Logistic Lifting (SILL) dictionary. We showed, in sections 2.6 and 2.7, that SILL model error drops exponentially when steepness parameters increase and higher dimensional state measurements are included in the data. This suggests that SILL dictionaries can be used to construct high-fidelity, low-dimensional, structured, and interpretable Koopman models.

The success of any state-inclusive Koopman dictionary depends on more than how the dictionary approximates the evolution of the state in time. It also depends on how well linear combinations of the dictionary functions approximate the dictionary function's Lie derivatives. The SILL dictionary is fully specified with closed-form analytical expressions (unlike deepDMD) and as a consequence of the theoretical results in this paper, satisfies a unique numerical property of uniform approximate finite closure. Our methodology provides a template for understanding how deep neural networks successfully approximate governing equations [15], the action of operators and their spectra [2], and dynamical systems [5]. Further, these results provide a pattern for improving scalability and interpretability of dictionary-based learning models for dynamical system identification.

More can be done to improve dimensional scalability of dictionary models. The deepDMD algorithm suggests further innovations to improve dictionary learning. One such innovation that we see in the dictionaries learned by deepDMD is the use of heterogeneous dictionaries. In this paper, we demonstrated that, even in a data-driven setting, one can glean real value from mixing basis functions and forming a heterogeneous dictionary for Koopman learning. This is an important insight as current dictionary based Koopman learning typically works with homogeneous dictionaries. The success of heterogeneous dictionaries helps to explain the success of the deepDMD algorithm, and may extend to other deep-learning-based Koopman learning algorithms [17–19].

The mixed-basis models can scale efficiently both in model parameters and model

dimension as the number of states used to construct the model grows. In this paper we demonstrated this point by constructing a heterogeneous dictionary, the augSILL dictionary, as a general solution to the data-driven Koopman learning problem. We then demonstrated the *uniform finite approximate closure* of this dictionary and showed it, in numerical simulations, on a number of distinct learning problems.

In under 1000 training epochs augSILL models matched the accuracy of trained deepDMD models. These models were anywhere from 18 to 372 times as accurate as models made using the polynomial dictionaries, which have no guarantees of uniform finite approximate closure. The augSILL dictionary performs like deepDMD to learn a dynamic system from data using an order of magnitude fewer parameters. Further, the augSILL dictionary is fully specified with closed-form analytical expressions (unlike deepDMD) and satisfies uniform approximate finite closure just like the SILL dictionary.

In studying the dictionary functions learned by deepDMD we found that the algorithm tended to learn redundant observables when the Koopman model was too high dimensional. A future line of work should explore this model-reduction property of DeepDMD, and see if augSILL models do the same.

Chapter 3

Trade-offs between Koopman model approximation and stabilizing control

3.1 Introduction

Koopman operator and Koopman generator theory consider alternate representations of a dynamic system. These representations exist when there is a one-to-one mapping between the system's solution and the solution to another linear, dynamic system (a one-to-one linear immersion) [34]. In this linear system, the vibrational, growth, and decay modes can be used to study the original nonlinear system [2]. These modes address problems in the field of fluid mechanics [3–5] and disease modeling [6], attack identification in the power grid [7], programming the steady state of biological systems [8, 35] and extracting new biosensors [9]. The spectral properties of the linear Koopman operator/generator in this function space connect to classical methods for model reduction, validation, identification and control [36–38].

Using Koopman models for controller design is an exciting approach to data-driven control, in part because it promises to be a one-size-fits-most solution to a very general problem. Recent work has developed data-driven approaches to using Koopman models for control. Summaries of available techniques for this work can be seen in [10,39]. These solutions include applying Model Predictive Control to the Koopman model [40]. This has potential to be more computationally efficient than model predictive control applied to other nonlinear models for the system dynamics. This method requires solving convex quadratic optimization problems to decide the control input at regular intervals. Other online controllers can be implemented from Koopman models as well, such as the policy iteration controller in [41].

Offline Koopman control techniques have been proposed as well [42,43], they typically perform in the context of a Koopman model that is fit to make accurate predictions based off of data. An advantage of offline feedback controllers, like the ones proposed in this paper, is that they are only solved once before integrating into a system. One challenge of offline controllers is that the true system may differ from the model against which the controller was designed. This challenge is especially pronounced in the context of Koopman modeling as nonlinear phenomena are approximated by a linear model.

This paper provides a treatment of Koopman-based control that explicitly accounts for errors in distinct components of a Koopman model. We use those errors to derive bounds on the success of offline controllers built on Koopman models. Through this treatment, we develop theoretical model-dependent controllers as well as bounds on minimal stabilizing compensatory control. These controllers and bounds expose several important lessons, such as

1. designing linear control with a Koopman model that is not exact can fail,

2. nonlinear controllers can compensate for Koopman model prediction error,

3. the model parameters of the Koopman model can be changed for the sake of control design with guaranteed stability results and

4. there are diminishing returns in improving Koopman model accuracy with respect to achieving stabilizing control.

When trying to stabilize the origin of a system with unknown dynamics, Koopman models are a data-driven class of black box models used to capture features of the dynamics for the sake of controller design. They typically involve the selection of nonlinear dictionary functions to choose the initial conditions of the higher-dimensional linear model. A sufficiently rich function dictionary can minimize the error of the Koopman model. However, even with explicit knowledge of the dynamics to be modeled it is typically not possible to choose dictionaries that make the error exactly zero. When there is any error, designing against the Koopman model without additional considerations can fail to stabilize the origin. In this paper, we show how to

1. account for model error when designing a nonlinear feedback controller both when the original system is explicitly known and in when the system is partially unknown (a data-driven scenario),

2. change the parameters of the Koopman model in conjunction with the design of a nonlinear controller to guarantee asymptotic stability of the origin, and

3. leverage the relationship between the model estimation and controller design to bound minimal correction needed to make proportional control laws work for an imperfect Koopman model.

These results highlight potential nuances in the problem of system identification for control. Theorem 5 can be used to show that, when stabilization of the unknown system is the goal, one can choose a model with less predictive accuracy to guarantee a control

objective. However, when a baseline controller is built to the specifications of a Koopman model, instead of the true dynamics, achieving stability may require additional control action. Theorems 6 and 7 give bounds on the additional control action needed to stabilize imperfectly represented systems.

In Section 3.2, we first state the control problem that we aim to solve. Then, in sections 3.3 and 3.4 we provide stabilizing controllers under different modeling assumptions. We analyze these controllers to explore the relationship between Koopman models and nonlinear stabilizing control. Finally, in Section 3.5, we bound the distance of proportional controllers, designed against a Koopman model, from successful stabilization as a function of the Koopman model error.

3.2 Koopman identification and control problem formulation

Assume we have a nonlinear dynamic system with affine control inputs

$$\dot{x} = f(x) + Bu, \tag{3.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $B \in \mathbb{R}^{n \times m}$ is full column-rank, $f : \mathbb{R}^n \to \mathbb{R}^n$ is analytic and $\dot{x}$ has an equilibrium point at the origin, $f(0) = 0$. Given this system, we first present a system identification problem statement.

3.2.1 System identification problem statement

In system identification, the objective is to find a low error model of Equation (3.1) given a particular family of models. In theory, this problem amounts to choosing the parameterization of a given mathematical model that is closest to the true dynamics

according to some metric. In practice, the governing equations for measured dynamics are unknown and sampled data is used to approximate the model's distance from the true dynamics. In this section, we present an idealized problem statement in order to build a framework which highlights how Koopman methods address both system identification and control.

The model parameters, which we denote here as θ, may be thought of as coordinates denoting a particular choice of model. The integer p is the number of model parameters, i.e. the dimension of θ. Our error function, $e : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$, is a function of the state, $x \in \mathbb{R}^n$, input, $u \in \mathbb{R}^m$, and model parameters, $\theta \in \mathbb{R}^p$. We sum up the identification problem as follows (see (a) in Figure 3.1):

$$\begin{aligned} &\min_{\theta \in \mathbb{R}^p} ||e(x, u, \theta)|| \\ &\text{where:} \\ &f(x) + Bu = P(\mu(x, u, \theta) + e(x, u, \theta)), \forall x \in \mathbb{R}^n. \end{aligned} \tag{3.2}$$

Here, $\mu : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^{n+N}$ is the model, $e : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^N$ is the model error and $P : \mathbb{R}^{n+N} \to \mathbb{R}^n$ is a projection function. Implementing the projection function, P, is often relevant to Koopman-based system identification as the final model typically is higher-dimensional than the original modeled dynamic system. Now that we have a general problem statement for system identification, we come back to the central theme of this paper: model-based, state-feedback (i.e. offline) control.

3.2.2 System control problem statement

In general, u is simply some function of t. However, in the state-feedback stabilization problem, u is explicitly constrained to be a function of the state. The state-feedback

stabilization problem may be written (see (b) in Figure 3.1):

$$\begin{aligned} &\min_{u} ||u(x)|| \\ &\text{so that:} \\ &\dot{x} = f(x) + Bu(x) \\ &\text{has a stable equilibrium point at the origin.} \end{aligned} \tag{3.3}$$

Here, u, is chosen to be some function of the state variable, $u : \mathbb{R}^n \to \mathbb{R}^m$.

3.2.3 System identification and control problem statement

When we interface with our dynamic system using a parameterizable model, we can write an alternative problem statement for state-feedback stabilization (see (c) in Figure 3.1):

$$\begin{aligned} &\min_{u,\theta \in \mathbb{R}^p} ||u(x,\theta)|| \\ &\text{so that:} \\ &\dot{x} = P(\mu(x, u(x,\theta), \theta) + e(x, u(x,\theta), \theta)) \\ &\text{has a stable equilibrium point at the origin and where:} \\ &f(x) + Bu = P(\mu(x, u(x,\theta), \theta) + e(x, u(x,\theta), \theta)), \forall x \in \mathbb{R}^n. \end{aligned} \tag{3.4}$$

In this problem formulation, the control input depends on the model itself. Thus, the input, u, is not longer a function of the state alone, but of the state and the model parameters, $u : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$. It is useful to tie the control action to the model parameters to analyze the controller's relationship to inaccuracy or uncertainty in the model itself (see Section 3.5). For example, this may be used to consider how bad a model can get while still resulting in the successful stabilization of the origin by a particular class of controllers.

3.2.4 Three model classes for system identification and control

There are a number of models that we could use to interface with our dynamics. In this section we will define three:

1. the *state-inclusive dictionary based mode*l,
2. the *augmented-state, state-inclusive dictionary based model* and
3. the *state-inclusive Koopman model*.

Each successive model we define will build on the definition of the previous model classes.

The first models are *state-inclusive dictionary-based models*. Some advantages of state-inclusive dictionary-based models include that they

1. can be used to recover the underlying physics of the system,
2. can be adapted to coincide with measured quantities of the state and
3. are exactly accurate whenever the true dynamics exist as a projection of the dictionary space.

A state-inclusive dictionary-based model has the form:

$$\dot{x} \approx \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x \\ \bar{\psi}(x) \end{bmatrix} + Bu, \tag{3.5}$$

where $K_1 \in \mathbb{R}^{n \times n}$, $K_2 \in \mathbb{R}^{n \times N}$, and $\bar{\psi} : \mathbb{R}^n \to \mathbb{R}^N$ where N is a nonnegative integer. In this model a linear combination of the state variables and functions of the state is used to estimate the flow of the dynamic system. The dictionary functions, $\bar{\psi}$, fill the role of approximating nonlinear components of the flow function, $f(x)$.

Our second model is the *augmented-state, state-inclusive dictionary-based model*. In this model, N state-variables ($\psi \triangleq [\psi_1, \psi_2, ..., \psi_N]^T$) are added to the original n state-variables. This model has the form:

$$\begin{aligned} \dot{x} &\approx \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + Bu \\ \dot{\psi} &\approx G(x, \psi, u). \end{aligned} \tag{3.6}$$

In this model all of the nonlinear evolution of the system flow is pushed in to the evolution of the added state variables, ψ. In this model, it is not necessary to define an algebraic expression for ψ in terms of the measured state, x, or time t. By contrast, $\bar{\psi}$, in the state-inclusive dictionary based model, is algebraically defined. We will use the relationship between these two state-inclusive dictionary based models to crystallize our notion of error in state-inclusive Koopman models in Section 3.2.5. So, what is the relationship between these two models?

If the initial condition of ψ is chosen so that $\psi(0) = \bar{\psi}(x(0))$ and the state evolution equation G satisfies

$$G(x, \psi, u) = G(x, u) = \frac{\partial \bar{\psi}(x)}{\partial x}(f(x) + Bu) \tag{3.7}$$

then, the augmented-state, state-inclusive dictionary-based model is mathematically equivalent to a state-inclusive dictionary-based model using $\bar{\psi}$ as its dictionary functions. Specifically, $\psi(t) = \bar{\psi}(x(t))$ for all $t > 0$. This particular case is important because we use it to develop our Koopman model error terms in Section 3.2.5.

Remark 3. *In our Koopman models (see Equation (3.9) of Section 3.2.5), ψ will represent added state-variables. In the Koopman model, ψ will approximate the true value of*

the dictionary functions, $\bar{\psi}$.

Both of these dictionary models are nonlinear. Koopman-based models attempt to rewrite the autonomous system dynamics

$$\dot{x} = f(x) \tag{3.8}$$

as a (potentially higher-dimensional) linear system.

Our final type of model is the *state-inclusive Koopman model*, a special class of augmented-state, state-inclusive dictionary based model. In this work, we assume that we have a state-inclusive Koopman model of the system in Equation (3.1). Our model is as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u \\ \psi_u(x,u) \end{bmatrix}, \tag{3.9}$$

where $\psi \in \mathbb{R}^N$. In our model, $K_3 \in \mathbb{R}^{N\times n}$, $K_4 \in \mathbb{R}^{N\times N}$, $B_{21} \in \mathbb{R}^{N\times m}$, $B_{22} \in \mathbb{R}^{N\times M}$, $\psi_u(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^M$, and M is a nonnegative integer. The input-dictionary function, ψ_u, is a component of the model that can be used to account for possible bilinear terms of the state and input that result from computing the time derivative $\dot{\psi}$ over the vector field given in Equation (3.1).

Note that the matrix given in Equation (3.9) is not the true infinite-dimensional Koopman operator for the dynamic system. This matrix is part of the Koopman-inspired model class for the system learned from data. In this model, the block matrices, K_1, K_2, K_3 and K_4 form a finite-dimensional approximation to the Koopman generator. Ideally, our K-matrix itself is near a projection of the action of our true infinite-dimensional Koopman operator on our sampled data space.

The block matrices B_{21} and B_{22} help us model the influence of inputs on the time

derivative of the dictionary functions, $\bar{\psi}(x)$. The function ψ_u is a function that accounts for the nonlinear component of the influence of u on ψ. When ψ_u is a linear function of the state and the input, x and u, then the state-inclusive Koopman model is, itself, a linear dynamic system of dimension $n + N$. Now we have a data-driven model that allows us to mathematically explore the usefulness of Koopman models on state-feedback control.

3.2.5 Two components of our Koopman model

A Koopman model needs to approximate the dynamics of $f(x)$. However, it must also approximate the dynamics of the dictionary functions evolving over the vector field, $f(x)$. To explicitly study these two properties, we split the Koopman model into two components:

1. *the model approximation component* which is tasked with directly modeling physics from data and

2. *the subspace closure component* which accounts for all other dynamics in the Koopman model. These dynamics come from the evolution of nonlinear dictionary functions. For continuous-time systems, this is the Lie derivative of the dictionary functions taken with respect to the flow of the original nonlinear system.

Since our dictionary, $\begin{bmatrix} x^T & \bar{\psi}^T \end{bmatrix}^T$, and its associated model state, $\begin{bmatrix} x^T & \psi^T \end{bmatrix}^T$, include the state dynamics directly, this component of the model is easy to examine independently of the rest of the Koopman model. We define this component and the remainder of the model below.

Definition 4. *The model approximation component of Equation (3.9) is*

$$\dot{x} \approx K_1 x + K_2 \psi + Bu. \tag{3.10}$$

The subspace closure component of Equation (3.9) is

$$\dot{\psi}(x) \approx K_3 x + K_4 \psi + B_{21} u + B_{22} \psi_u(x, u). \tag{3.11}$$

The model approximation error is then defined to be

$$e_1(x, \psi) \triangleq f(x) - (K_1 x + K_2 \psi), \tag{3.12}$$

the subspace closure error is defined to be

$$e_2(x, \psi) \triangleq \frac{\partial \bar{\psi}(x)}{\partial x} f(x) - (K_3 x + K_4 \psi) \tag{3.13}$$

and the subspace input error is defined to be

$$e_B(x, u) \triangleq \frac{\partial \bar{\psi}(x)}{\partial x} B u - (B_{21} u + B_{22} \psi_u(x, u)). \tag{3.14}$$

We refer to e_1 as the model approximation error as it can only be zero when the model lies within the span of x, ψ and u; the state, augmented state and input. We let e_2 denote the subspace closure error, since it can only be zero when the subspace of our Koopman model corresponding to the augmented state lies in the span of x, ψ, u and ψ_u, the state, augmented state, input and input-dictionary function. Stated another way, when the subspace closure error, e_2, is nonzero for all choices of K_3, K_4, B_{21} and B_{22}, then $\dot{\psi}$ is not in this span. The work in [44] proposes methods for building a Koopman model with low error in the subspace closure component. Traditionally, low subspace closure error is desirable as errors in approximating the evolution of the dictionary functions become errors in approximating the physics of the system.

Our Koopman model state-feedback stabilization problem is therefore formulated as

a specific class of the system identification and control problem given in Equation (3.4) and (c) in Figure 3.1. We state the problem as follows:

$$\min_{K\in\mathbb{R}^{(n+N)\times(n+N)},\theta_2\in\mathbb{R}^{p_2}} ||u(x,K,\theta_2)||$$

so that:

$$\dot{x} = \begin{bmatrix} I & 0 \end{bmatrix} \left(\begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u \\ \psi_u(x,u) \end{bmatrix} + \begin{bmatrix} 0 \\ e_B(x,u) \end{bmatrix} \right)$$

has a stable equilibrium point at the origin. (3.15)

The dynamics, including control action, that we will stabilize to provide a feasible solution to the problem statement in Equation (3.15) are:

$$\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u \\ \psi_u(x,u) \end{bmatrix} + \begin{bmatrix} 0 \\ e_B(x,u) \end{bmatrix}. \quad (3.16)$$

Please note that when Equation (3.16) has a stable equilibrium point at the origin, the constraint of Equation (3.15) is satisfied.

3.2.6 Solution methodology and implications

To the author's knowledge, up until now, no one has explicitly addressed the model approximation and subspace closure error in the context of building a stabilizing controller. In fact, explicit representation and error bounds on Equation (3.13) in the context of modeling are rare in the literature. Still, it has been discussed explicitly in Chapter 3 for certain classes of dictionaries. To explore the relationship between the model error terms and stabilizing control, in sections 3.3 and 3.4 we provide stabilizing controller

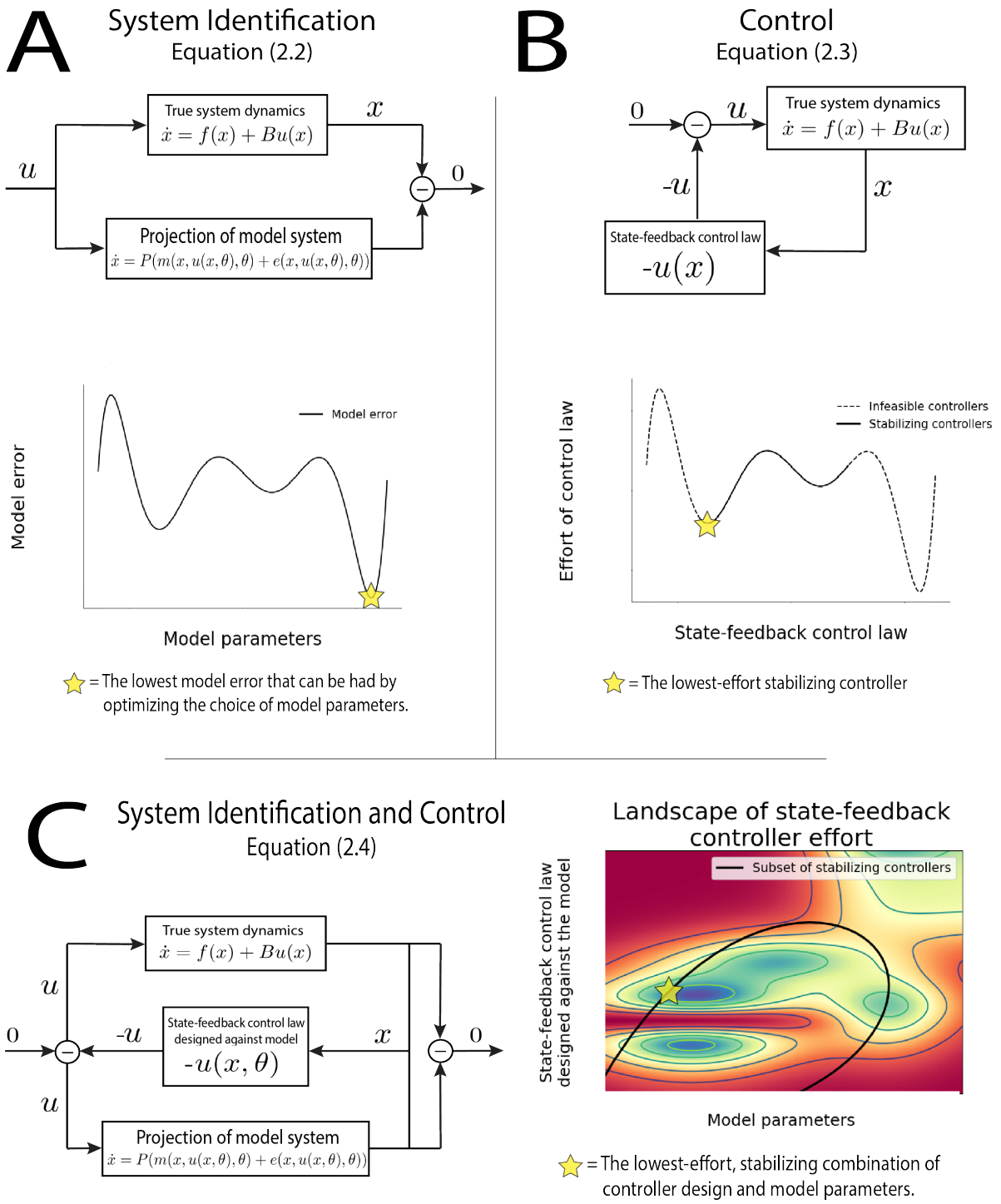

Figure 3.1: Graphic representations of the problem statements given in Section 2. Block diagrams for each problem are also included.

design strategies for Equation (3.16) under different assumptions relevant to distinct modeling scenarios.

Section 3.3 presents a solution when we assume that the Koopman model is perfect. It is the easiest version of the problem to solve, and is commonly solved in the literature [6, 39, 44–46]. However, when the augmented state dynamics are not exactly captured, control design under the assumption that the model is perfect can result in failure to stabilize the origin, see Example 3.

In Section 3.4 we operate with and without the assumption that the true model dynamics are known. In spite of this knowledge, choosing a dictionary with a subspace closure error of zero is typically impossible. This means that the Koopman subspace corresponding to the augmented state is not invariant with respect to the state and augmented state.

After Section 3.4, we address a lingering question. If one designs a controller based off of a Koopman model, will the controller successfully stabilize the true underlying dynamic system? To answer that question, in Section 3.5, we consider the success of a proportional dictionary controller when applied to a system that has been modeled in a Koopman framework. We use the model error to bound the size of additional compensatory control needed to guarantee stability of the origin.

3.3 Stabilizing control when the Koopman model is exact

When the model approximation error and the subspace closure error are zero (i.e. $e_1(x) = 0, e_2(x) = 0$) we simply need to design a stabilizing controller for the dynamic system:

$$\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u, \tag{3.17}$$

with $\psi(0) \triangleq \bar{\psi}(0)$. This can essentially be treated as stabilizing an LTI system with known parameters. For example, one can choose a linear state-feedback control law.

Example 2. *Successful control designed to stabilize the Koopman model. Given the dynamic system (the system state space and example trajectories are shown in Figure 3.2):*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_1 + u_1 \\ x_2 - x_1^3 + u_2 \end{bmatrix}, \tag{3.18}$$

with a Koopman model of the autonomous system of

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix}, \tag{3.19}$$

where $\psi(0) = x_1(0)^3$, then the control:

$$u(x) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix}, \tag{3.20}$$

will stabilize the system, see Figure 3.3.

If the subspace dynamics of the Koopman model are incorrect, stabilization via linear state-feedback of the Koopman model is not guaranteed to work.

Example 3. *(Failed control designed to stabilize the Koopman model.) Given the system*

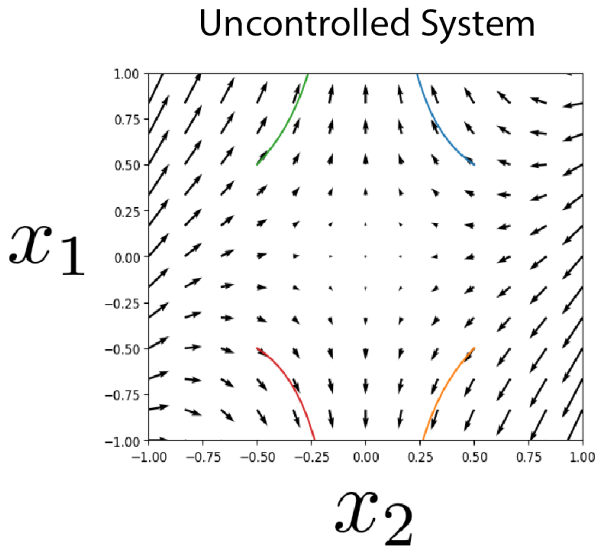

Figure 3.2: The state-space of the uncontrolled dynamic system given in Example 1. Four example trajectories are given.

in Equation (3.18), with an incorrect Koopman model of the autonomous system of

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & -1 \\ .1 & 1 & .1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix}, \tag{3.21}$$

where $\psi(0) = x_1(0)^3$*, then the control:*

$$u(x) = \begin{bmatrix} 0 & -1 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_1^3 \end{bmatrix}, \tag{3.22}$$

stabilizes the modeled system but not the true dynamics. The origin of the controlled system is unstable, see Figure 3.3.

Due to error in the Koopman models, online control techniques, such as Model Predictive Control [40, 47], are implemented to stabilize dynamic systems in this context.

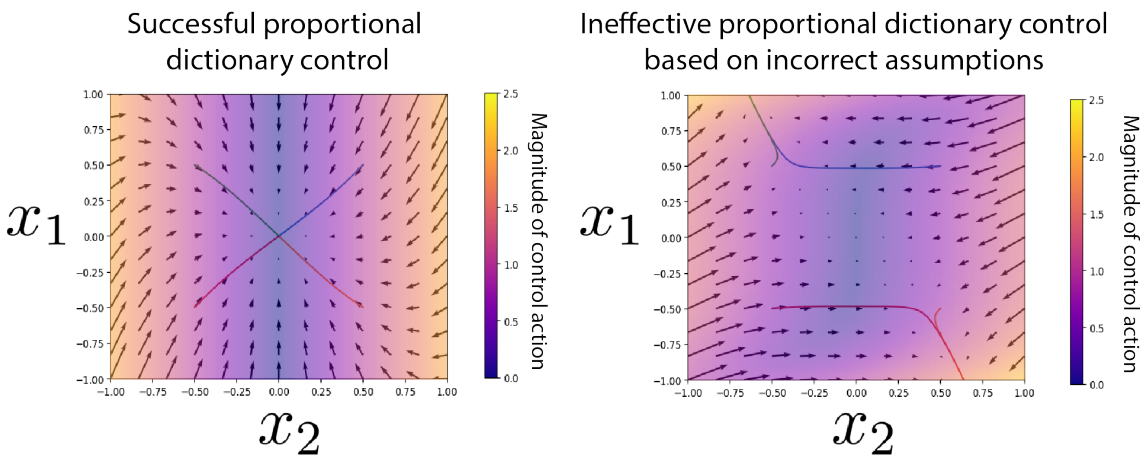

Figure 3.3: State-space and sample trajectories for the dynamic system given in Example 1 with the proportional controllers in equations (3.20) and (3.22). Color is used to indicate the magnitude of the control signal of each controller across the state-space.

This is because the condition that $e_1(x) = e_2(x) = 0$, the Koopman model is perfect for all $x \in \mathbb{R}^n$, requires that both

1. the vector field, $f(x)$, and

2. the time derivative of the augmented state, $\dot{\psi}$,

lie in the span of the state and augmented state, $[x^T, \psi^T]^T$. It is possible to construct specific examples where this is the case and the Koopman operator/generator is a finite dimensional matrix, see Example 2. But, in general, for both $f(x)$ and $\dot{\psi}$ to lie in the span of $[x^T, \psi^T]^T$, the Koopman operator/generator needs to be infinite dimensional [2,5]. Therefore learning an error-free Koopman model from data is infeasible.

In practice, the Koopman observables can be chosen to best fit the observed dynamics. If this is done well the model approximation error, e_1, should be small. However, the fact that the model approximation error e_1 is small or exactly zero does not imply that the subspace closure error, e_2, is also small or zero. The independance of e_1 and e_2 means that, even when the true system dynamics are known, the Koopman model will likely

have a nonzero subspace closure error, $e_2 \neq 0$. So, how do we account for subspace closure error e_2 when designing a controller?

3.4 Stabilizing controller design from a Koopman model given known dynamics

In this section we design a stabilizing controller for Equation (3.16) when the model approximation error, $e_1(x, \psi)$, and the subspace closure error, $e_2(x, \psi)$, are known. In particular, we are interested in the case when $e_1(x)$ is 0, since

1. if the modeler knows $e_1(x)$, then by subtracting out the model they can acquire all nonlinear terms in the dynamics, f, of Equation (3.1) and therefore

2. the modeler can choose dictionary functions that include the nonlinear terms in f when e_1 is known a priori.

We will first define some mathematical terms as a preliminary. Then we present a stabilizing controller. Finally, we analyze the mathematical dependencies of the controller to extract principles of Koopman control. We will see that classic nonlinear control strategies can be adapted to compensate for the failure of linear control for Koopman models.

Theorem 5. *Assume that $x = 0$ is an unstable equilibrium point for the nonlinear forced dynamic system:*

$$\dot{x} = f(x) + Bu, \tag{3.23}$$

with an associated Koopman model:

$$\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u \\ \psi_u(x,u) \end{bmatrix}. \tag{3.24}$$

The controller:

$$\bar{u}(x,\psi) = \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + w(x,\psi), \tag{3.25}$$

where $\psi(0) \triangleq \bar{\psi}(x(0))$ *and* $\dot{\psi} \triangleq \dot{\bar{\psi}} = \frac{d\bar{\psi}}{dx} \cdot f(x)$*, will stabilize the controlled system*

$$\dot{x} = f(x) + B\bar{u}(x,\psi) \tag{3.26}$$

if

1. *for some positive definite energy matrix:*

$$Q \triangleq \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \tag{3.27}$$

the matrix product:

$$\begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 & K_4 \end{bmatrix} \tag{3.28}$$

is negative definite and

2. *the inequality:*

$$F_1(x,\psi) + F_2(x,\psi)w(x,\psi) < 0 \tag{3.29}$$

is maintained in a domain of $\mathbb{R}^{n+N}$ *containing the origin. Here* $F_1(x,\psi) = (x^T Q_1 +$

$\psi^T Q_2^T) e_1(x,\psi) + (x^T Q_2 + \psi^T Q_3)\left(e_2(x,\psi) + \frac{\partial \bar{\psi}(x)}{\partial x}(BP_1 x + BP_2\psi)\right)$ *and* $F_2(x,\psi) = \left(x^T Q_1 + \psi^T Q_2^T + (x^T Q_2 + \psi^T Q_3)\frac{\partial \bar{\psi}(x)}{\partial x}\right) B$.[1]

Proof: We begin by relating the system in Equation (3.23) to the system model in Equation (3.24) by adding in the error terms to get:

$$\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} u \\ \psi_u(x,u) \end{bmatrix} + \begin{bmatrix} 0 \\ e_B(x,u) \end{bmatrix}, \quad (3.30)$$

where we have (from Definition 4) that:

$$\begin{aligned} e_1(x,\psi) &= f(x) - (K_1 x + K_2\psi) \\ e_2(x,\psi) &= \frac{\partial \bar{\psi}(x)}{\partial x} f(x) - (K_3 x + K_4\psi) \text{ and} \\ e_B(x,u) &= \frac{\partial \bar{\psi}(x)}{\partial x} Bu - (B_{21}u + B_{22}\psi_u(x,u)), \end{aligned} \quad (3.31)$$

where $B_{22} \in \mathbb{R}^{N\times M}, \psi_u : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^M$ and $\frac{\bar{\psi}(x)}{\partial x}$ is an $N \times n$ matrix for any x, we can choose the model-input matrix B_{22} and the dimension of the nonlinear input dynamics approximation function M so that $B_{22} = I^{N\times N}$.

This implies that if we choose

$$\psi_u(x,u) \triangleq \frac{\partial \bar{\psi}(x)}{\partial x} Bu - B_{21}u = (\frac{\partial \bar{\psi}(x)}{\partial x} B - B_{21})u, \quad (3.32)$$

then we have that $e_B(x,u) = 0$.

[1]The functions F_1 and F_2 are components of the time derivative of a Lyapunov function for the controlled system.

So, we choose our model to be:

$$\begin{aligned}
\begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & I \end{bmatrix} \begin{bmatrix} u \\ (\frac{\partial \bar{\psi}(x)}{\partial x} B - B_{21})u \end{bmatrix} \\
&= \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B & 0 \\ 0 & \frac{\partial \bar{\psi}(x)}{\partial x} B \end{bmatrix} \begin{bmatrix} u \\ u \end{bmatrix} \\
&= \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B \\ \frac{\partial \bar{\psi}(x)}{\partial x} B \end{bmatrix} u.
\end{aligned} \tag{3.33}$$

From here we can use a positive definite energy function (Lyapunov candidate function) to consider the stability of the origin of our model. We choose the following function:

$$V(x,\psi) = \frac{1}{2} \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}. \tag{3.34}$$

We then compute its time derivative to look for a sufficient condition for stability of the origin. The time derivative is:

$$\begin{aligned}
\frac{dV}{dt}(x,\psi,u) &= \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \left(\begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1(x,\psi) \\ e_2(x,\psi) \end{bmatrix} + \begin{bmatrix} B \\ \frac{\partial \bar{\psi}(x)}{\partial x} B \end{bmatrix} u \right) \\
&= \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} \\
&\quad + \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} e_1(x,\psi) + Bu \\ e_2(x,\psi) + \frac{\partial \bar{\psi}(x)}{\partial x} Bu \end{bmatrix}.
\end{aligned} \tag{3.35}$$

Now we consider that our state feedback controller is a sum of a proportional and

nonlinear controller of the following form:

$$u(x,\psi) = \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + w(x,\psi), \tag{3.36}$$

where $w : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$. Then we have that our time derivative can be represented as a sum of two components. The first component,

$$\begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}, \tag{3.37}$$

is assumed to be negative definite by our choice of Q and P. In practice, it can be set to be negative definite by using the Cholesky-decomposition to set up a linear system of equations to solve for such a Q and P (see Section A.5). The second component,

$$\begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} e_1(x,\psi) + Bw(x,\psi) \\ e_2(x,\psi) + \frac{\partial \bar{\psi}(x)}{\partial x} B \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \frac{\partial \bar{\psi}(x)}{\partial x} Bw(x,\psi) \end{bmatrix} \tag{3.38}$$

must be made negative definite by a choice of the function w. Since we have that

$$\begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} = \begin{bmatrix} x^T Q_1 + \psi^T Q_2^T & x^T Q_2 + \psi^T Q_3 \end{bmatrix} \tag{3.39}$$

the term in Equation (3.38) equals

$$\begin{aligned}
&(x^TQ_1+\psi^TQ_2^T)(e_1(x,\psi)+Bw(x,\psi))+(x^TQ_2+\psi^TQ_3)e_2(x,\psi)\\
&+(x^TQ_2+\psi^TQ_3)(\frac{\partial\bar{\psi}(x)}{\partial x})(BP_1x+BP_2\psi+Bw(x,\psi))\\
&=(x^TQ_1+\psi^TQ_2^T)e_1(x,\psi)+(x^TQ_2+\psi^TQ_3)\left(e_2(x,\psi)+\frac{\partial\bar{\psi}(x)}{\partial x}(BP_1x+BP_2\psi)\right)\\
&+\left(x^TQ_1+\psi^TQ_2^T+(x^TQ_2+\psi^TQ_3)\frac{\partial\bar{\psi}(x)}{\partial x}\right)Bw(x,\psi)\\
&\triangleq F_1(x,\psi)+F_2(x,\psi)w(x,\psi).
\end{aligned} \tag{3.40}$$

Our condition of negative definiteness is that the function w satisfies:

$$F_1(x,\psi)+F_2(x,\psi)w(x,\psi)<0. \tag{3.41}$$

One application of this theorem is that, when the dynamics are known, we can write explicit formulas for a nonlinear stabilizing controller. When the true dynamics are unknown, a controls engineer can use a data-driven method to build a controller that satisfies the conditions of Theorem 5 for all availible data.

Corollary 7. *For tuning parameter, $c>0$, and positive-definite function $\varepsilon(x,\psi)>0$, consider the nonlinear control functions*

1. $\bar{w}_1(x,\psi)\triangleq w_i(x,\psi)\triangleq\frac{-F_1(x,\psi)-c\varepsilon(x,\psi)}{mF_{2Bi}(x,\psi)}$, *and*
2. $\bar{w}_2(x,\psi)\triangleq\begin{cases}\frac{-F_1(x,\psi)-c\varepsilon(x,\psi)}{F_{2Bi}(x,\psi)} \text{ if } ||F_{2B}(x,\psi)||_\infty=|F_{2Bi}(x,\psi)|\\ 0 \text{ otherwise,}\end{cases}$.

If (for $i=1$ or $i=2$) we have that

$$F_1(x,\psi)+F_2(x,\psi)\bar{w}_i(x,\psi)\neq 0 \tag{3.42}$$

in a domain containing the origin of $\mathbb{R}^{n+N}$*, then the controller:*

$$\bar{u}(x,\psi) \triangleq \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \bar{w}_i(x,\psi) \tag{3.43}$$

will stabilize the system in Equation (3.23).

Proof: The function $F_1(x,\psi)$ is scalar-valued. On the other hand, $F_2(x,\psi)$ is a $1 \times n$ real-valued matrix. This means that the product $F_2(x,\psi)B$ is a $1 \times m$ real-valued matrix, which we call $F_{2B}(x,\psi)$. This means that we need to choose $w(x,\psi)$ so that the dot-product of $F_{2B}(x,\psi)$ and $w(x,\psi)$ is less than $-F_1(x,\psi)$. Specifically,

$$\begin{aligned} &F_1(x,\psi) + F_2(x,\psi)Bw(x,\psi) < 0 \\ &\quad \implies F_2(x,\psi)Bw(x,\psi) < -F_1(x,\psi). \end{aligned} \tag{3.44}$$

Using our new notation, we further simplify this expression to a dot product:

$$\begin{aligned} &F_{2B}(x,\psi)w(x,\psi) < -F_1(x,\psi) \\ \implies &\sum_{i=1}^{m} F_{2Bi}(x,\psi)w_i(x,\psi) < -F_1(x,\psi). \end{aligned} \tag{3.45}$$

There are a number of choices for $w(x,\psi)$ that satisfy this condition. For example, if we define

$$w_i(x,\psi) \triangleq \frac{-F_1(x,\psi) - c\varepsilon(x,\psi)}{mF_{2Bi}(x,\psi)}, \tag{3.46}$$

for some tuning parameter, $c > 0$, and some positive-definite function $\varepsilon(x,\psi) > 0$ then we satisfy Equation (3.41). Possible function for $\varepsilon(x,\psi)$ include $\varepsilon(x,\psi) \triangleq V(x,\psi)$ from Equation (3.34), or $\varepsilon(x,\psi) \triangleq \sum_{i=1}^{n} x_i^2 + \sum_{j=1}^{N} \psi_j^2$. However, this solution does have a singularity whenever (for any $i = 1, 2, ..., m$) we have that $F_{2Bi}(x,\psi) = 0$. If instead we

define

$$w_i(x,\psi) \triangleq \begin{cases} \frac{-F_1(x,\psi)-c\varepsilon(x,\psi)}{F_{2Bi}(x,\psi)} \text{ if } ||F_{2B}(x,\psi)||_\infty = |F_{2Bi}(x,\psi)| \\ 0 \text{ otherwise,} \end{cases} \tag{3.47}$$

for the same tuning parameter, $c > 0$, and positive-definite function $\varepsilon(x,\psi) > 0$ from Equation (3.46), then we satisfy Equation (3.44) and there is only a singularity when $F_{2B}(x,\psi) = \vec{0}$. This is also provides control that uses at most the same level of actuation as the choice of w in Equation (3.46).

So, given Q, K, the following controller, u, ensures that the time derivative of the Lyapunov candidate function, $V(x,\psi)$ is negative definite. Therefore, this controller,

$$u(x,\psi) = \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + w(x,\psi), \tag{3.48}$$

where $w_i(x,\psi)$ is defined in Equation (3.47) and P_1, P_2 satisfy Theorem 5, will stabilize the origin. By extension, this controller will stabilize the modeled Koopman system.

3.4.1 Implications of Theorem 5

The controller presented in Theorem 5 works by controlling the Koopman modeled dynamics with a state feedback controller and then applying nonlinear feedback control to compensate for the Koopman model error. The nonlinear component, w, accounts for both model approximation error and subspace closure error (see Definition 4).

We can use Theorem 5 to design a nonlinear state-feedback controller for a Koopman modeled system if we so desire. To do so one first uses the Koopman model and the process in Section A.5 to select the matrices $Q > 0$, P_1 and P_2. At this point, if we know the true system dynamics, we can use one of the proposed controller designs given in Equations (3.46) and (3.47) to choose the nonlinear control function, w. To do so

we need to choose two things: the tuning constant, $c > 0$, and the negative-definiteness guarantee function, $\varepsilon(x, \psi)$. Possible choices include $\varepsilon(x, \psi) \triangleq x^T \psi$ and $\varepsilon(x, \psi) \triangleq x^T Q \psi$. If the true system dynamics are unknown, then the modeler chooses the nonlinear control function, w, to satisfy Equation (3.41) for the existing data and may design to take into account any additional desired controller constraints. For example, the modeler could choose $w(x, \psi)$ to be a neural network and parameterize it to satisfy Equation (3.41) for all availible measurements with minimal norm activation weights.

The structure of this controller demonstrates that controllers exist for Koopman models with high error. If the Koopman model approximates zero dynamics, i.e. $K = 0$, then the nonlinear control component, w, can be designed against the nonlinear system. If the Koopman model is exact, i.e. $e_1 = 0$ and $e_2 = 0$, then $w(x, \psi) = 0$ will stabilize the system. This indicates that we can have a nuanced trade-off between solving the Koopman identification problem and solving the Koopman control problem. A more accurate Koopman model can restrict the size of the nonlinear component in the controller, however, this may or may not be optimal.

The controller prescribed by Theorem 5 is an example of proportional dictionary feedback control plus compensatory nonlinear state-feedback control. It is natural to assume that, when the Koopman model is more accurate (the error terms are small), the compensatory nonlinear-state feedback control can also be small. This raises an important question to modelers who use proportional dictionary feedback control designed against an imperfect Koopman model. How small can compensatory state-feedback control be and still guarantee stability of the origin?

3.5 Quantifying how much correction proportional dictionary feedback control requires to ensure stability

A Koopman model provides a linear representation of the system dynamics. Therefore, approaching control via linear techniques in the higher-dimensional space of a Koopman model is a natural step in stabilization. By considering the dictionary as the state of the linear model, a number of MIMO feedback control techniques for LTI dynamic systems can be applied to stabilize the origin. However, as shown in Example 3, doing so can fail. In this section, we develop error bounds to quantify the distance from success of linear dictionary feedback control $\Pi : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$, $\Pi(x, \psi) = P_1 x + P_2 \psi$.

Remark 4. *The* Π *in Section 3.5 stands for "proportional." We did not use "P" so that the symbol would not be confused with the* P *in Section 3.2, which stands for "projection." We hope that this is a clear way to structure our notation.*

To quantify the distance from successful stabilization of a proportional dictionary feedback controller, Π, we can take a nonlinear dictionary feedback controller, call it $w(x, \psi)$, $w : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$, so that the sum of the controllers ($u = \Pi + w$) stabilizes the true system in Equation (3.1). We assume that there exists at least one function, $w_{min} : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$, so that $u \triangleq \Pi + w_{min}$ stabilizes Equation (3.16) and $||w_{min}|| \leq ||w||$ for all w that stabilize Equation (3.16). In other words, the augmented state feedback function w_{min} is a minimal norm stabilizing control that is added to the proportional augmented state feedback control, Π. The norm of any additional stabilizing nonlinear controller, w, is then an upper bound on the norm of the minimal-norm function, w_{min}. If the proportional dictionary feedback controller, Π, already stabilizes the origin, then $w_{min} = 0$ and so $||w_{min}|| = 0$. The feedback control w_{min} (indeed any w that we consider

here) is a correction added to Π in order to ensure stability. This correction is typically nonlinear in the dictionary functions.

We present three bounds on w_{min}, the minimal nonlinear control action needed to stabilize Equation (3.1).

3.5.1 Bounds on correction to proportional dictionary control when B is invertible

When a Koopman model has error, designing proportional control action based off of that model can fail. However, as established by Theorem 5, additional nonlinear control can compensate for the erroneous proportional control, no matter how bad the Koopman model may be. This raises the question, how much additional corrective control is needed to properly stabilize the underlying dynamics? Under the assumption that the input matrix, B, is invertible, the following theorem provides an upper bound on the size of the minimal stabilizing corrective control.

Theorem 6. *Given Equation (3.16) where B is invertible and*

$$u(x,\psi) = \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + w(x,\psi), \tag{3.49}$$

where the system given in Equation (3.16) has a quadratic Lyapunov function then

$$w_{min} \triangleq \min_{w} ||w||, \text{ so that the origin of Equation (3.16) is stable} \tag{3.50}$$

will satisfy:

1. $\frac{||w_{min}||}{||\Pi||} \leq \frac{||B^{-1}f(x)||}{||\Pi||} + 1$ *where* $\Pi(x,\psi) \triangleq P_1 x + P_2 \psi$, *the proportional dictionary control component of the controller,* $\dot{x} = f(x) + Bu$ *and*

2. $||w_{min}|| \leq ||B^{-1}((K_1 + BP_1)x + (K_2 + BP_2)\psi)|| + ||B^{-1}e_1||$.

Proof: Given our choice of $u(x, \psi)$ we can rewrite Equation (3.16) as

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{\psi}\end{bmatrix} &= \begin{bmatrix}K_1 + BP_1 & K_2 + BP_2\\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2\end{bmatrix}\begin{bmatrix}x\\ \psi\end{bmatrix} + \begin{bmatrix}e_1\\ e_2\end{bmatrix} + \begin{bmatrix}B & 0\\ B_{21} & B_{22}\end{bmatrix}\begin{bmatrix}w\\ \psi_u\end{bmatrix} + \begin{bmatrix}0\\ e_B\end{bmatrix}\\ &= \begin{bmatrix}K_1 + BP_1 & K_2 + BP_2\\ \frac{\partial\bar{\psi}}{\partial x}(K_1 + BP_1) & \frac{\partial\bar{\psi}}{\partial x}(K_2 + BP_2)\end{bmatrix}\begin{bmatrix}x\\ \psi\end{bmatrix} + \begin{bmatrix}e_1\\ \frac{\partial\bar{\psi}}{\partial x}e_1\end{bmatrix} + \begin{bmatrix}B\\ \frac{\partial\bar{\psi}}{\partial x}B\end{bmatrix}w.\end{aligned} \tag{3.51}$$

We can perform the final step shown because

$$\dot{\psi} = \frac{\partial\bar{\psi}}{\partial x}\dot{x}. \tag{3.52}$$

To understand this, it may be helpful to recall the distinction between ψ and $\bar{\psi}$ given in Section 3.2.4. To summarize, ψ is an augmented state variable and $\bar{\psi}$ is a vector of nonlinear dictionary functions of the state variables.

Now we can examine the controlled system's quadratic Lyapunov function as follows, (where Q is positive definite)

$$V(x, \psi) = \frac{1}{2}\begin{bmatrix}x^T & \psi^T\end{bmatrix}\begin{bmatrix}Q_1 & Q_2\\ Q_2^T & Q_3\end{bmatrix}\begin{bmatrix}x\\ \psi\end{bmatrix}. \tag{3.53}$$

Then, for $t : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^n$ satisfying

$$t(x, \psi) \triangleq (K_1 + BP_1)x + (K_2 + BP_2)\psi + e_1(x, \psi), \tag{3.54}$$

we have that

$$\begin{aligned}\frac{dV}{dt} &= \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} t(x,\psi) + Bw \\ \frac{\partial \bar{\psi}}{\partial x}(t(x,\psi) + Bw) \end{bmatrix} \\ &\triangleq \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} h(x,\psi) \\ \frac{\partial \bar{\psi}}{\partial x} h(x,\psi) \end{bmatrix},\end{aligned} \tag{3.55}$$

where $h : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^n$ so that

$$h(x,\psi) \triangleq t(x,\psi) + Bw(x,\psi). \tag{3.56}$$

We define the two functions t and h for notational convenience.

Since we assume in the theorem statement that this quadratic Lyapunov function exists we know that there is a solution. Call this solution, $\bar{h}(x,\psi) \triangleq t(x,\psi) + B\bar{w}(x,\psi)$, where $\bar{w} : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$ is our solution-producing input and

$$\begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} \bar{h}(x,\psi) \\ \frac{\partial \bar{\psi}}{\partial x} \bar{h}(x,\psi) \end{bmatrix} < 0. \tag{3.57}$$

Then, we have that $\varepsilon\bar{h}(x,\psi)$ is also a solution for $\varepsilon > 0$. Furthermore, since we also assume that B is invertible, then, for any $\varepsilon > 0$, we can choose a new solution producing input, $\bar{w}_\varepsilon : \mathbb{R}^n \times \mathbb{R}^N \to \mathbb{R}^m$ so that $\bar{w}_\varepsilon(x,\psi) \triangleq \varepsilon\bar{w} + B^{-1}(\varepsilon - 1)t(x,\psi)$ then:

$$\begin{aligned} t(x,\psi) + B\bar{w}_\varepsilon &= t(x,\psi) + B(\varepsilon\bar{w} + B^{-1}(\varepsilon - 1)t(x,\psi)) \\ &= \varepsilon t(x,\psi) + \varepsilon B\bar{w}(x,\psi) \\ &= \varepsilon\bar{h}(x,\psi). \end{aligned} \tag{3.58}$$

We now let $w \triangleq \bar{w}_\varepsilon$. This choice of compensatory stabilizing input, w, will satisfy

$$\varepsilon\bar{h} = \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + e_1 + Bw. \tag{3.59}$$

If we choose w so that $\varepsilon \to 0$, then we have that:

$$0 = \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + e_1 + Bw. \tag{3.60}$$

We can then do some algebraic manipulation to get:

$$-w = B^{-1} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + B^{-1}e_1 + \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}. \tag{3.61}$$

Working from this expression, we could do different sorts of groupings of terms before taking the norms of both sides and looking for bounds on the norms. Two ways that we could work this out give us the two bounds in the theorem statement as follows.

1. First, we simplify as follows:

$$\begin{aligned} -w &= B^{-1} \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + B^{-1}e_1 + \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} \\ &= B^{-1}f(x) + \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} \\ &\triangleq B^{-1}f(x) + \Pi(x, \psi) \end{aligned} \tag{3.62}$$

so we have that

$$\begin{aligned} ||w|| &= ||B^{-1}f(x) + \Pi|| \\ &\leq ||B^{-1}f(x)|| + ||\Pi||, \end{aligned} \tag{3.63}$$

we then divide both sides by $||\Pi||$ to get $\frac{||w||}{||\Pi||} \leq \frac{||B^{-1}f(x)||}{||\Pi||} + 1$, so we have that

$$\frac{||w_{min}||}{||\Pi||} \leq \frac{||w||}{||\Pi||} \leq \frac{||B^{-1}f(x)||}{||\Pi||} + 1. \tag{3.64}$$

2. Now we group the terms a different way, so that:

$$\begin{aligned} -w &= B^{-1}\begin{bmatrix} K_1 & K_2 \end{bmatrix}\begin{bmatrix} x \\ \psi \end{bmatrix} + B^{-1}e_1 + \begin{bmatrix} P_1 & P_2 \end{bmatrix}\begin{bmatrix} x \\ \psi \end{bmatrix} \\ &= B^{-1}\begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \end{bmatrix}\begin{bmatrix} x \\ \psi \end{bmatrix} + B^{-1}e_1. \end{aligned} \tag{3.65}$$

Taking the norm of both sides and applying the triangle inequality gives:

$$||w_{min}|| \leq ||w|| \leq ||B^{-1}((K_1 + BP_1)x + (K_2 + BP_2)\psi)|| + ||B^{-1}e_1||, \tag{3.66}$$

the second bound.

We now consider the significance of these two bounds. The first bound is:

$$\frac{||w_{min}||}{||\Pi||} \leq \frac{||B^{-1}f(x)||}{||\Pi||} + 1 \tag{3.67}$$

where $\Pi(x, \psi) \triangleq P_1x + P_2\psi$, the proportional dictionary control component of the controller and $\dot{x} = f(x) + Bu$. The advantage of this bound is that it is, in a sense, scale-free as it provides an upper bound on the fold-change (comparing to the proportional dictionary component) of the non-Koopman component of the stabilizing controller. When the proportional dictionary component gets large the bound approaches 1. This indicates that choosing a large control signal may be a mistake that the nonlinear component would need to rectify via negation and that this rectification would overshadow any other ad-

justments the controller would need to make for stabilization.

The second bound is:

$$||w_{min}|| \leq ||B^{-1}((K_1 + BP_1)x + (K_2 + BP_2)\psi)|| + ||B^{-1}e_1||. \tag{3.68}$$

One advantage of this bound is highlighted by the following corollary.

Corollary 8. *Under the same set-up and assumptions of Theorem 6, there exists proportional constant matrices P_1^*, P_2^*, so that*

$$||w_{min}|| \leq ||B^{-1}e_1(x, \psi)||. \tag{3.69}$$

Proof: By our assumptions B is invertible, therefore we can choose to define P_1^* and P_2^* as follows,

$$P_1^* \triangleq -B^{-1}K_1 \text{ and } P_2^* \triangleq -B^{-1}K_2. \tag{3.70}$$

Then we have from Theorem 6 that

$$\begin{aligned} ||w_{min}|| &\leq ||B^{-1}((K_1 + BP_1^*)x + (K_2 + BP_2^*)\psi)|| + ||B^{-1}e_1|| \\ &= ||0|| + ||B^{-1}e_1|| \\ &= ||B^{-1}e_1||. \end{aligned} \tag{3.71}$$

Corollary 8 shows that (due to the invertibility of B) one can choose proportional constants so that, w_{min} is purely bounded by our model approximation error scaled by B. An interesting consequence is that when the model approximation error is zero, B is full rank and there exists some quadratic Lyapunov function for our system, then the proportional dictionary component can simply work with B to cancel out the Koopman model exactly to achieve stabilization. No thought needs to be given for the subspace

closure component of the model. Such a dramatic control action is typically overkill when stabilization is the objective. This indicates that the second bound in Theorem 6 is not tight.

3.5.2 A bound on correction to proportional dictionary control when B is rank-1

The resulting bounds from Theorem 6 apply when B is full rank. This implies that there are just as many inputs as true state-variables in the governing equations. This can be the case in many applications, however, it is often the case that there are more state-variables than inputs. In this section, we provide a distinct upper bound on the size of the minimal stabilizing corrective control from the bounds in Theorem 6 when the controlled system only has a single input, i.e. the input matrix, B, is rank 1.

Theorem 7. *Given Equation (3.16) where B is a single (nonzero) column, the system has a quadratic Lyapunov function*

$$V(x,\psi) = \frac{1}{2}\begin{bmatrix} x^T & \psi^T \end{bmatrix}\begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix}\begin{bmatrix} x \\ \psi \end{bmatrix}, \tag{3.72}$$

and

$$u(x,\psi) = \begin{bmatrix} P_1 & P_2 \end{bmatrix}\begin{bmatrix} x \\ \psi \end{bmatrix} + w(x,\psi), \tag{3.73}$$

then the minimum norm w that will stabilize the system will be bound above in norm, specifically

$$||w_{min}|| \leq ||\frac{-v_1^T e_1 - v_2^T e_2 - v_2^T t(x,\psi) - q(x,\psi)}{(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x})B}||, \tag{3.74}$$

where $t(x,\psi) = (\frac{\partial \bar{\psi}}{\partial x}B - B_{21})(P_1 x + P_2\psi)$, $v_1^T = x^T Q_1 + \psi^T Q_2^T$, $v_2^T = x^T Q_2 + \psi^T Q_3$ *and* $q(x,\psi) = \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}$.

Proof: Given our choice of $u(x,\psi)$ we can rewrite Equation 3.16 as

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} w \\ \psi_u \end{bmatrix} + \begin{bmatrix} 0 \\ e_B \end{bmatrix} \\ &= \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 + e_B \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} w \\ \psi_u \end{bmatrix}. \end{aligned} \tag{3.75}$$

Now we take our quadratic Lyapunov function,

$$V(x,\psi) = \frac{1}{2} \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}. \tag{3.76}$$

Then we have that

$$\frac{dV}{dt} = \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} \bar{K}_1(x,\psi) \\ \bar{K}_2(x,\psi) \end{bmatrix}, \tag{3.77}$$

where

$$\begin{bmatrix} \bar{K}_1(x,\psi) \\ \bar{K}_2(x,\psi) \end{bmatrix} \triangleq \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 + e_B \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} w \\ \psi_u \end{bmatrix}. \tag{3.78}$$

For simplicity of notation, we define the following quadratic term:

$$q(x,\psi) \triangleq \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 + B_{21}P_1 & K_4 + B_{21}P_2 \end{bmatrix} \begin{bmatrix} x \\ \psi \end{bmatrix}. \tag{3.79}$$

Then we can simplify the previous term as follows:

$$\begin{aligned} \frac{dV}{dt} &= \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \left[\begin{bmatrix} e_1 \\ e_2 + e_B \end{bmatrix} + \begin{bmatrix} B & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} w \\ \psi_u \end{bmatrix} \right] + q(x,\psi) \\ &= \begin{bmatrix} x^T & \psi^T \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} e_1 + Bw \\ e_2 + e_B + B_{21}w + B_{22}\psi_u \end{bmatrix} + q(x,\psi). \end{aligned} \tag{3.80}$$

Now we begin to directly compute this expression:

$$\begin{aligned} &\begin{bmatrix} x^TQ_1 + \psi^TQ_2^T & x^TQ_2 + \psi^TQ_3 \end{bmatrix} \begin{bmatrix} e_1 + Bw \\ e_2 + e_B + B_{21}w + B_{22}\psi_u \end{bmatrix} + q(x,\psi) \\ &\qquad \triangleq \begin{bmatrix} v_1^T & v_2^T \end{bmatrix} \begin{bmatrix} e_1 + Bw \\ e_2 + e_B + B_{21}w + B_{22}\psi_u \end{bmatrix} + q(x,\psi), \end{aligned} \tag{3.81}$$

where v_1 and v_2 are functions of x and ψ defined for convenience, much like q.

Here, we recall the definition of the subspace input error, $e_B(x,u)$:

$$e_B(x,u) = \frac{\partial \bar{\psi}}{\partial x} Bu - B_{21}u - B_{22}\psi_u(x,u). \tag{3.82}$$

Note that, when $u = P_1 x + P_2 \psi + w$, this implies

$$\begin{aligned} e_B + B_{21} w + B_{22} \psi_u &= \frac{\partial \bar{\psi}}{\partial x} B w + (\frac{\partial \bar{\psi}}{\partial x} B - B_{21})(P_1 x + P_2 \psi) \\ &\triangleq \frac{\partial \bar{\psi}}{\partial x} B w + t(x, \psi), \end{aligned} \tag{3.83}$$

where $t(x, \psi)$ is a function term defined for convenience, much like q, v_1 and v_2.

So we continue to compute our expression:

$$\begin{aligned} \text{Equation (3.81)} &= \begin{bmatrix} v_1^T & v_2^T \end{bmatrix} \begin{bmatrix} e_1 + Bw \\ e_2 + e_B + B_{21} w + B_{22} \psi_u \end{bmatrix} + q(x, \psi) \\ &= \begin{bmatrix} v_1^T & v_2^T \end{bmatrix} \begin{bmatrix} e_1 + Bw \\ e_2 + \frac{\partial \bar{\psi}}{\partial x} B w + t(x, \psi) \end{bmatrix} + q(x, \psi) \\ &= v_1^T e_1 + v_1^T B w + v_2^T e_2 + v_2^T \frac{\partial \bar{\psi}}{\partial x} B w + v_2^T t(x, \psi) + q(x, \psi) \\ &= (v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x}) B w + v_1^T e_1 + v_2^T e_2 + v_2^T t(x, \psi) + q(x, \psi) \\ &< 0, \end{aligned} \tag{3.84}$$

this implies that

$$(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x}) B w < -v_1^T e_1 - v_2^T e_2 - v_2^T t(x, \psi) - q(x, \psi). \tag{3.85}$$

Since B is a column vector the (normally vector-valued) function $(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x}) B$ is a scalar-valued function. So, if we set

$$w(x, \psi) = \frac{-v_1^T e_1 - v_2^T e_2 - v_2^T t(x, \psi) - q(x, \psi)}{(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x}) B} - \varepsilon \frac{||x|| + ||\psi||}{(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x}) B}, \tag{3.86}$$

for some $\varepsilon > 0$, then the Lyapunov function candidate will have a negative definite time

derivative. So, our minimal stabilizing w will be bound above by

$$||\frac{-v_1^T e_1 - v_2^T e_2 - v_2^T t(x,\psi) - q(x,\psi)}{(v_1^T + v_2^T \frac{\partial \bar{\psi}}{\partial x})B}||. \tag{3.87}$$

The above norm may be computed as a function of e_1 and e_2 as the remaining components are completely determined by the model.

The main challenge in applying this result is to know a priori the values of $e_1(x,\psi)$, $e_2(x,\psi)$ and Q. A train-test split of the data can be used to approximate $e_1(x,\psi)$ and an approximation of $e_2(x,\psi)$ can be computed from the approximation of e_1. The matrix Q may be chosen as an ansatz. The methods of computing $v_1(x,\psi), v_2(x,\psi), t(x,\psi)$ and $q(x,\psi)$ are given in the proof.

3.6 Conclusion

Koopman operator theory provides a number of powerful modeling tools. With effort and sufficient data, these models can perform high-fidelity predictions [18, 48] and can be used for control [40]. However, high-accuracy predictive models can be expensive and time-consuming to collect data for and produce. The theorems of this paper suggest that costly high-accuracy models are not required for successful control. But, this fact raises a further question. What degree of model accuracy is needed to design stabilizing feedback control in a data-driven setting?

To explore the relationship between Koopman model accuracy and control we have developed two theorems, each describing a model-dependent nonlinear controller that explicitly accounts for the accuracy of the Koopman model. One designs under the assumption that the true dynamics are known and the other does not. Theorem 5 shows that the parameters of the Koopman model can be tied to parameters of the state-

feedback controller to build model-dependent controllers. Model-dependent controllers are useful because they can be used to explore the point at which controlling the model alone will fail. The model-dependent controllers that we have developed here highlight how parameterizing the Koopman model to best fit the data is a potential waste of valuable modeling resources when control is the objective. This has been shown to be the case both when the true dynamics can be used to directly compute control action and when they are only tied to control action indirectly, a necessary condition for data-driven control.

Building on the idea of model-dependent controllers, we have developed three bounds (contained in Theorem 6 and Theorem 7) on the effectiveness of proportional controllers designed against Koopman models of dynamic systems. Each of these bounds can be approximated without explicit knowledge of the true dynamics of the system (from data). In the end we see that perfect Koopman models are not necessary for stabilizing a nonlinear dynamic system but determining the needed control action is nuanced. However, mathematical tools address this nuance.

Chapter 4

A novel model class for bowtie biological networks with universal classification properties

4.1 Introduction

To respond to changes in their environment, bacteria and other cells need to perform in-cell computation through networks of chemical reactions. The transcription and translation of messenger RNA into proteins interplays with environmental signals to identify and respond to new environments [49]. Successful cells will correctly classify their current environment and appropriately produce the proteins needed for that context [50]. Classification can be thought of as a control policy acting on a continuous signal space and reducing it to a discrete action from a finite or countably finite list of options. It is interesting because it is a black-box "analog-to-digital" or "continuous-to-discrete" transformation that biological cells have to make each day. We ask the question, "how is biological classification represented computationally and structurally in the architecture

and evolved design of living cells?" Answering this question opens the door to further inquiry. What are the energy and metabolic benefits of such control and how can these evolved computation structures be applied to improve the efficiency of human-designed systems such as AI?

Dedicated biologists and systems theorists have identified common network motifs implemented in almost all cells [51]. Two of these motifs are *dense overlapping regulons* and *single input modules*. Dense overlapping regulons have the ability to perform cellular decision making and have a network structure akin to that of a shallow neural network, with a number of input transcription factors (input TFs) mapping to a distinct set of genes [52]. Single input modules contain a master TF (typically auto regulated) that simultaneously activates a number of target genes. The activation of these target genes typically enables the sequential production of proteins which combine together to form an enzyme. The final product enzyme may have multiple functions, its primary function may be related to performance in a particular environment. Typically, its secondary function is to deactivate the master TF [52].

Within most observed cell sensory transcription networks, dense overlapping regulons are not layered in cascade with other dense overlapping regulons. This means that intracellular classification is not made by a deep network. Additionally, the master TF of single input modules are an output gene of a dense overlapping regulon [51]. This is an example of the bowtie motif in biological networks [53, 54].

We build this bowtie network by combining models of transcriptional networks from [52] and [55] to hypothesize a model for bacterial decision making. Given this model, we analyze its biological feasibility and basic capability for classification. Part of our interest in this motif stems from its network structure. Because all ambiguous components of our proposed model's network structure are feedforward; results from linear network reconstruction suggest that this model's topology and parameters could be uniquely

identifiable from time-series experimental data [56, 57].

In this paper, we study a class of biological networks that have the ability to convert continuous environmental signals into discrete control policies, namely that are able to perform a mode of biological classification. We introduce this class of models as the DOR2SIM model and provide de novo mathematical definitions and analyze their architectural properties. We consider a 11-state model that is constructed as the minimal set of signal TFs needed to build a distinguishable DOR motif and the minimal set of target genes to make a distinguishable SIM motif. We show that our 11-state DOR2SIM model is capable of classifying any constant input signal from the environment and thus, prove by inductive logic, that DOR2SIM models are a powerful class of models for universally classifying all positive constant signals. We then provide a sufficient model parameter condition to verify the ability of any DOR2SIM model to classify constant inputs based off of an experimentally measured or simulated equilibrium point of the system. This parameter condition suggests that the behavior of source nodes in the DOR motif at equilibrium leads to successful classification. As a preliminary, we define dynamic classification.

4.2 Dynamic classification of inputs

In this paper we consider the classification properties of a nonlinear dynamic system of the form

$$\dot{x} = f(x, u), \tag{4.1}$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m$. We define classification as the act of sorting input to the system into distinct classes. These classes are distinguished from each other by the systems state

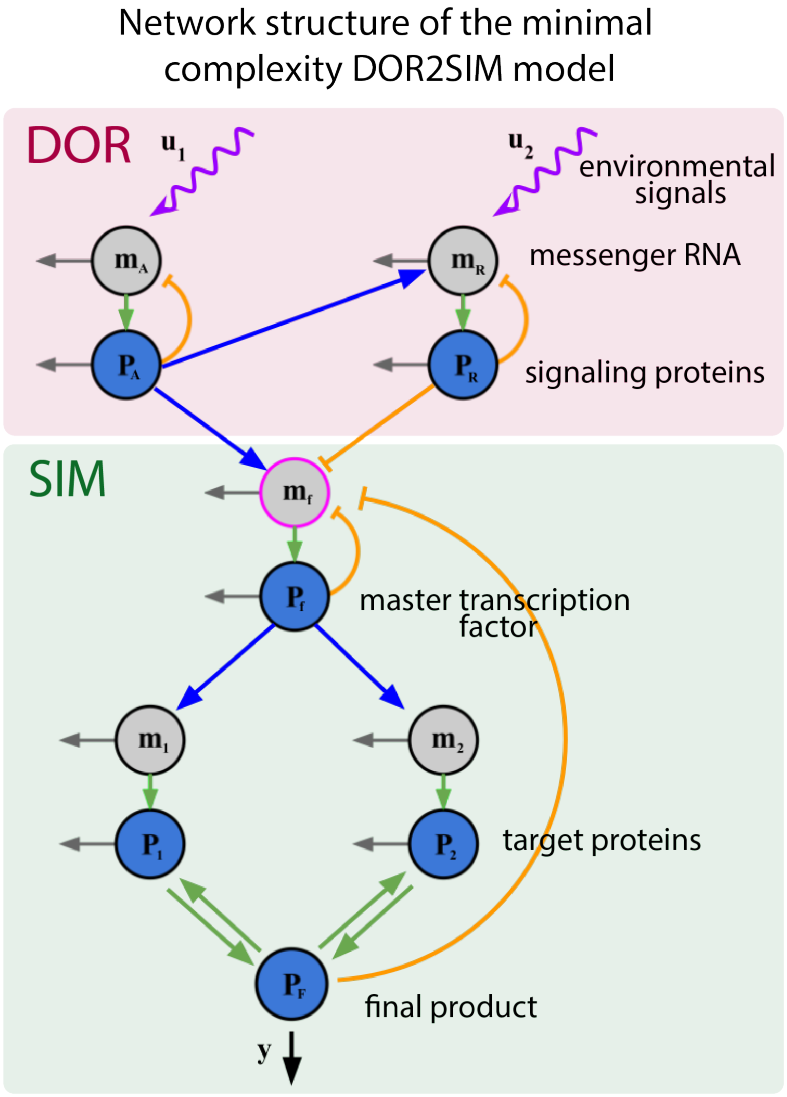

Figure 4.1: A diagram of the network structure of the DOR2SIM model. The dense overlapping regulon (DOR) and single input module (SIM) components are highlighted in red and green respectively. Nodes are labeled consistent with Equation (4.11).

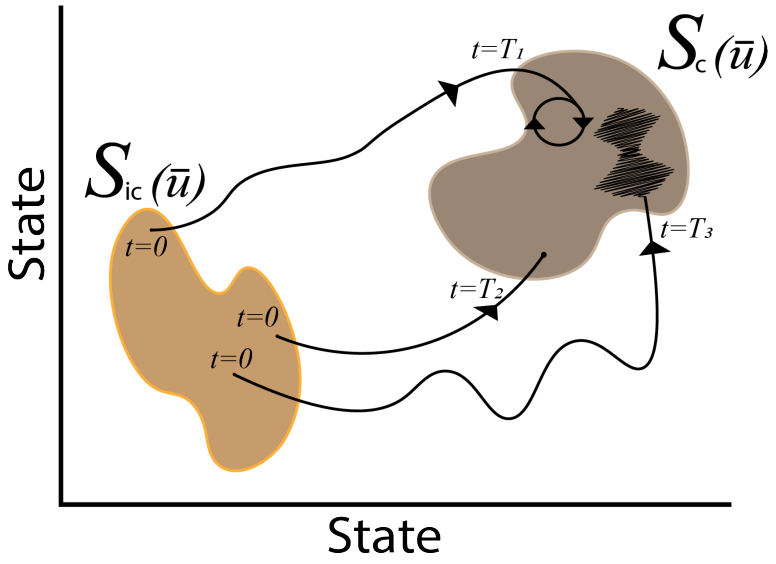

Figure 4.2: A visual schematic of Definition 5. Given a constant input, $\bar{u}$, trajectories that begin in S_{ic} will reach and remain in S_c in finite time.

variables. This information on the class of input should be stored in the system until the input changes, at which point it should reclassify the new input. This definition of the dynamic classification of inputs is given precisely below.

Definition 5. *The dynamic system in Equation (4.1) is defined to classify a constant-valued input, $\bar{u} \in \mathbb{R}^m$ over the initial point set $S_{ic} \subset \mathbb{R}^n$, to the class set $S_c \subset \mathbb{R}^n$ if, for all initial points in the initial point set, $x_0 \in S_{ic}$, we have that our constant valued input and the system dynamics evolved from our initial point, $\dot{x} = f(x, \bar{u})$, $x(0) = x_0$, implies that there exists a threshold time, $T \in \mathbb{R}^+$, so that for any time greater than the threshold time the state is in the class set, $\exists\ T > 0$ so that for all $t > T$ $x(t) \in S_c$. See Figure 4.2 for a visual schematic of this definition.*

Cellular combinatorial decision making in transcriptional networks is performed in dense overlapping regulons (DORs) and then applied in (among other motifs) single input modules (SIMs). This means that classification models of DORs connecting to SIMs should be able to classify any constant-valued inputs to an appropriate class set.

The definition of what set might be appropriate is open for discussion, however, in this paper "appropriate" sets will be simply connected sets. The most important feature in a dynamic system for classification is the ability to distinguish between class sets for different inputs. This means that, if a small perturbation is applied to the inputs, the intersection of the new class set with the class set of the unperturbed inputs should be the empty set. We formalize this notion in the following definition of an input classifying system.

Definition 6. *The dynamic system in Equation (3.1) is an input classifying system over an interval, $\mathcal{I}$, of $\mathbb{R}^m$ if it classifies all constant inputs in $\mathcal{I}$ to disjoint class sets.*

Before we explore the classification potential of our model of the DOR2SIM network motif, we introduce the dynamics of the DOR and SIM models separately and then we combine them.

4.3 Dense overlapping regulon model

In a dense overlapping regulon (DOR), environmental signals influence the production of messenger RNA that codes to transcription factors (TFs) that will influence the downstream expression of genes that respond to the environmental signal. This is the controller component of the DOR2SIM motif. In this model the dynamic states will be the concentrations of messenger RNA as well as the concentrations of the proteins that they code for. The dynamics can be given by a function, $f_{DOR} : \mathbb{R}^{2n} \times \mathbb{R}^n \to \mathbb{R}^{2n}$, and the outputs are the concentrations of the final TFs, $y_{DOR} : \mathbb{R}^{2n} \to \mathbb{R}^n$. These functions are defined as follows

$$\begin{aligned} x_{DOR} &\triangleq [m_1, ..., m_n, P_1, ..., P_n]^T \\ \dot{x}_{DOR} &= f_{DOR}(x_{DOR}, u), \end{aligned} \tag{4.2}$$

where:

$$\begin{aligned} \dot{m}_i &= F_{m_i}(P_i, P_{i+1}, ..., P_n) - \delta_i m_i + b_i u_i \\ \dot{P}_i &= k_{P_i} m_i - \delta_{n+i} P_i \\ y_{DOR} &= \begin{bmatrix} P_1 & P_2 & \dots & P_n \end{bmatrix}^T, \end{aligned} \tag{4.3}$$

where each δ, b and k is a constant parameter and each function F_m is a special nonlinear function which describes the binding of the input proteins to the promoter site for the specified messenger RNA.

So, what is the functional form of F_{m_i}, our nonlinear binding function? Well this depends on the type of promoter that we are assuming in our model. In this paper we assume that binding is competitive and only a single protein may bind at a time.

When the binding site only permits a single protein to bind at a time, the activation/repression dynamics in F_{m_i} are given (as derived in Chapter 2 Section 2.3 of [55]) by:

$$F_{m_i}(P_i, P_{i+1}, ..., P_n) = \frac{\beta_{m_i} \sum_{j \in A} (P_j/K_j)^{n_j} + \alpha_0}{1 + \sum_{j=i}^{n} (P_j/K_j)^{n_j}}, \tag{4.4}$$

where $A \subset \{i, i+1, ..., n\}$ is the set of all indices corresponding to activator proteins (the remaining indices correspond to repressor proteins), β_{m_i} is a constant, α_0 is a constant giving the rate of gene expression in the absence of activators on the promoter site and the parameters K_j and n_j are constants.

4.4 Single input module model

The single input module (SIM) is a motif in transcription networks where a single master TF, P_M, will activate the promoter sites for a number of other genes. These genes, in turn, produce proteins which combine with each other to form a single enzyme. In the DOR2SIM motif, this enzyme can perform multiple functions. Its primary pur-

pose is typically to respond to the environmental signaling which activated the master transcription factor (master TF) in the first place. Additionally, it often down-regulates the master TF to turn off the system once the enzyme concentration is high enough.

The dynamics for the SIM will be given by the function $f_{SIM} : \mathbb{R}^{3N+1} \times \mathbb{R}^n \to \mathbb{R}^{3N+1}$. The output for the SIM is simply the final enzyme, so $y_{SIM} : \mathbb{R}^{3N+1} \to \mathbb{R}$. The system dynamics are given below as follows:

$$\begin{aligned} x_{SIM} &\triangleq [m_M, P_M, m_1, ..., m_N, P_1, ..., P_{2N-1}]^T \\ \dot{x}_{SIM} &= f_{SIM}(x_{SIM}, u), \end{aligned} \tag{4.5}$$

where:

$$\begin{aligned} \dot{m}_M &= F_M(u_1, ..., u_n, P_M, P_{2N-1}) - \delta_{m_M} m_M \\ \dot{P}_M &= k_M m_M - \delta_{P_M} P_M \\ \dot{m}_{i \leq N} &= F_{m_i}(P_M) - \delta_{m_i} m_i \\ \dot{P}_1 &= k_1 m_1 + \delta_{P+N+1} P_{N+1} \\ &\quad - k_{N+1} P_1 P_2 - \delta_{P_1} P_1 \\ \dot{P}_{2 \leq i \leq N} &= k_i m_i + \delta_{P_{N+i-1}} P_{N+i-1} \\ &\quad - k_{i+N-1} P_i P_{i-1} - \delta_{P_i} P_i \\ \dot{P}_{N+1} &= k_{N+1} P_1 P_2 - \delta_{P_{N+1}} P_{N+1} \\ \dot{P}_{N+2 \leq i} &= k_i P_{i-N+1} P_{i-1} - \delta_{P_i} P_i \\ y_{SIM} &= P_{2N-1}, \end{aligned} \tag{4.6}$$

where each δ and k is a constant corresponding to degradation rate and reaction rate respectively. The function F_M and the functions F_i are nonlinear functions describing the binding of input proteins to the promoter site for messenger RNA.

Each function, F_i, takes the form of a single variable Hill activation function, $Hill$. In this paper we consider two types of Hill functions, Hill activation and Hill repressor

functions. Hill activation functions are defined as follows:

$$Hill_{act}(x) = \frac{\beta x^{\mathbf{n}}}{K^{\mathbf{n}} + x^{\mathbf{n}}}. \tag{4.7}$$

Each F_i is a Hill activation function. The function, F_n in the DOR model is a Hill repressor function, defined as follows:

$$Hill_{rep}(x) = \frac{\beta K^{\mathbf{n}}}{K^{\mathbf{n}} + x^{\mathbf{n}}}. \tag{4.8}$$

For both types of single-variable Hill functions the constants β, K and $\mathbf{n}$ are function parameters.

On the other hand, the functional form of F_M is more nuanced. Because of our underlying assumption that each promoter binding site only permits a single protein to bind at a time, F_M will be modeled according to Equation (4.4), much like the F_m functions in the DOR model.

4.5 The combined DOR2SIM model

If we interconnect a DOR with a SIM, ignoring feedback due to the final product of the SIM influencing the signals that stimulate the expression of the input TFs, we have an open-loop model of the system. This model is as follows:

$$\begin{aligned} \dot{x}_{DOR} &= f_{DOR}(x_{DOR}, u) \\ \dot{x}_{SIM} &= f_{SIM}(x_{SIM}, y_{DOR}(x_{DOR})) \\ y &= y_{SIM}(x_{SIM}). \end{aligned} \tag{4.9}$$

4.5.1 The minimal complexity DOR2SIM model

To study the dynamic classification properties of the DOR2SIM model class we will begin by proposing a minimal complexity instantiation of the DOR2SIM that retains all of the dynamic features of the model. This means that our model will include:

1. a feedforward connection between the signaling TFs
2. negative auto regulation in both the signaling and master TFs as well as
3. a higher order protein that is not coded for directly in the organism's DNA.

Complexity here is structural complexity and is measured as the number of signaling TFs in the DOR component and the number of target TFs in the SIM component. The minimal complexity DOR2SIM model will have only two signaling TFs in its DOR component and two target TFs in its SIM component. If we structurally simplify the model further by removing a signaling TF from the DOR or a target TF from the SIM, then the DOR and SIM motifs, respectively, would not be distinguishable from a cascade motif and so we no longer have a DOR2SIM model. Thus there are no further structural simplifications possible for this model.

However, we will operate under a number of non-structural simplifying assumptions. We will assume that

1. protein degradation rates for all monomer proteins are equal,
2. translation rates for all monomer proteins are equal,
3. input signals are unweighted ($b_1 = b_2 = 1$),
4. all promoters maximal production rate is 1 ($\beta = 1$ in Hill functions),

5. all promoters activate (or repress) at the promoter site after they form a dimer (which we model by setting the Hill coefficients to be 2 assuming high cooperativity [58]) and

6. half maximal repression for the source messenger RNA is reached when the concentration of the protein it codes for is at 1, this means that in Equation (4.11):

$$F_1(x_3) \triangleq \frac{\beta}{1+x_3^2}, \tag{4.10}$$

for some positive constant β.

Ultimately, our minimal complexity DOR2SIM model is as follows:

$$\begin{bmatrix} \dot{m}_a \\ \dot{m}_r \\ \dot{P}_a \\ \dot{P}_r \\ \dot{m}_f \\ \dot{m}_1 \\ \dot{m}_2 \\ \dot{P}_f \\ \dot{P}_1 \\ \dot{P}_2 \\ \dot{P}_F \end{bmatrix} \triangleq \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \end{bmatrix} = \begin{bmatrix} F_1(x_3) - \delta x_1 + u_1 \\ F_2(x_3, x_4) - \delta x_2 + u_2 \\ kx_1 - \delta x_3 \\ kx_2 - \delta x_4 \\ F_3(x_3, x_4, x_8, x_{11}) - \delta x_5 \\ F_4(x_8) - \delta x_6 \\ F_5(x_8)\delta x_7 \\ kx_5 - \delta x_8 \\ kx_6 - \delta x_9 + \Delta x_{11} - \bar{k} x_9 x_{10} \\ kx_7 - \delta x_{10} + \Delta x_{11} - \bar{k} x_9 x_{10} \\ \bar{k} x_9 x_{10} - \Delta x_{11} \end{bmatrix}, \tag{4.11}$$

where the parameters $\delta, \Delta, k, \bar{k}$ are each positive constants. A visual depiction of the network structure of this model is in Figure 4.1.

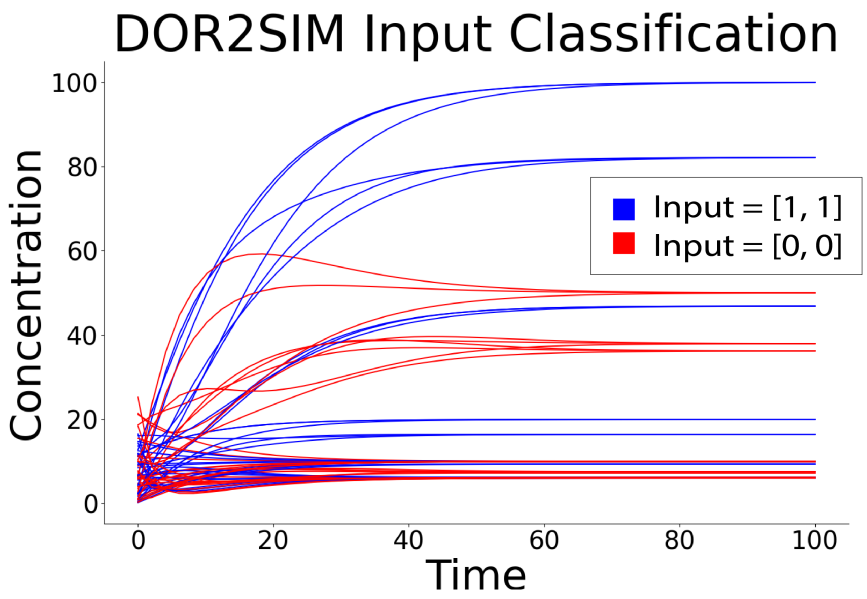

Figure 4.3: The evolution, in time, of multiple state trajectories from random initial conditions given distinct inputs for identical DOR2SIM models. Note that the equilibrium points depend on the input, rather than initial conditions demonstrating that this network classifies distinct inputs with with its state. The lines appear to overlap because we are plotting all 11 state trajectories vs time on the same axis. If it were an 11 dimensional graphic they would not overlap.

4.6 Classification properties of the DOR2SIM model

Our numerical exploration of this model indicates that, like other biological models arising from mass action kinetics, the DOR2SIM model is highly stable and that the equilibrium points vary with changes in input values, see Figure 4.3. These are encouraging results for the capacity of this model to perform classification. However, the nonlinear dynamics of these systems allow for these systems to potentially have multi-stability induced hysteresis. Multi-stability introduces a dependence between the ultimate state of the system on initial conditions. This dependence is problematic for classifying inputs since the same input could potentially, given the initial state of the system, map to distinct class sets. Thus, we delve into a mathematical analysis of the DOR2SIM model in order to clarify when it can model dynamic input classification in a biological context.

4.6.1 Defining biologically feasible initial conditions, inputs and class sets

Recall from Definition 5 that the DOR2SIM model will only be able to classify inputs from a set of initial conditions to a class set if any initial condition combined with the specified input results in a system trajectory that will reach the class set and remain in the class set in finite time. As such, our set of initial conditions will be the set of all biologically feasible initial conditions. Since each state variable in our system is the concentration of some compound, be it a protein or messenger RNA, biologically feasible initial conditions must be nonnegative and finite. So our biologically feasible initial conditions belong to the set:

$$S_{ic} \triangleq \{x \in \mathbb{R}^{11} : x_i \geq 0, \forall i = 1, 2, ..., 11\}. \tag{4.12}$$

Our inputs and class sets likewise must be nonnegative in order to be biologically feasible.

4.6.2 Sufficient conditions for the 11-state DOR2SIM model to classify any biologically feasible input

Theorem 8. *Given the minimal complexity DOR2SIM model and assuming that every biologically feasible initial condition will lie in the bastion of attraction of some stable equilibrium point, if the monomer degradation rate satisfies,*

$$\left(\frac{k^2}{3}u_1^2\right)^{\frac{1}{4}} < \delta < \frac{\sqrt{6}}{6} \approx 0.408, \tag{4.13}$$

then the model will classify the constant input $\bar{u} \triangleq [u_1, u_2]$ *over the set of all biologically feasible initial conditions,* S_{ic}*, to the class set* S_c *where*

$$S_c \triangleq \{x \in S_{ic} : x_1 = m_a^{eq}, x_3 = \frac{k}{\delta} m_a^{eq}\}, \tag{4.14}$$

for some $m_a^{eq} \in \mathbb{R}^+$.

Furthermore, the minimal complexity DOR2SIM model is an input classifying system over some sufficiently small intervals of the set of biologically feasible inputs that satisfies Equation (4.13).

Proof: In this proof our objective will be to demonstrate that any equilibrium point of this system will belong to the class set S_c. We begin by noting that at equilibrium,

$$0 = k x_1^{eq} - \delta x_3^{eq} \implies x_3^{eq} = \frac{k}{\delta} x_1^{eq} = \frac{k}{\delta} m_a^{eq}. \tag{4.15}$$

We also have that

$$0 = \frac{1}{1 + c^2 x_1^{eq2}} - \delta x_1^{eq} + u_1, \tag{4.16}$$

where we define $c \triangleq \frac{k}{\delta}$, to simplify notation. Multiplying both sides by $1 + c^2 x_1^{eq2}$ and combining like terms we obtain the following equivalent statement:

$$0 = -\delta c^2 x_1^{eq3} + u_1 c^2 x_1^{eq2} - \delta x_1^{eq} + (1 + u_1). \tag{4.17}$$

By Descartes rule of signs this polynomial will have 3 or 1 positive root and 0 negative roots. This means that there will be 0 or 2 complex roots. We will apply Sturm's Theorem to determine conditions where there will be exactly 1 positive root and 2 complex roots. To do this we build a Sturm sequence and verify the differences in the sign changes in the Sturm sequence at $x_1^{eq} = 0$ and as $x_1^{eq} \to \pm\infty$.

When we do so we have that the sequence signs are:

$$\begin{aligned}(+,-,-\varsigma_1,\varsigma_2) \text{ as } x_1^{eq} \to -\infty \\ (+,-,-,\varsigma_2) \text{ when } x_1^{eq} = 0 \\ (-,-,\varsigma_1,\varsigma_2) \text{ as } x_1^{eq} \to \infty,\end{aligned} \tag{4.18}$$

where

$$\begin{aligned}\varsigma_1 &= sign(\frac{-2c^2u_1^2+6\delta^2}{9\delta}) = sign(3\delta^2 - c^2u_1^2) \\ \varsigma_2 &= sign(\frac{P_{pos}(c,u_1,\delta)}{4c^2u_1^2 - 24c^2\delta^2u_1^2 + 36\delta^4}) \\ &= sign(c^2u_1^2 - 6c^2u_1^2\delta^2 + 9\delta^4),\end{aligned} \tag{4.19}$$

where P_{pos} is a 5^{th}-order polynomial in c, u_1 and δ with all positive coefficients. In order to have zero real negative roots and one real positive root, Sturm's Theorem implies that ς_1 and ς_2 both be positive. For ς_1 to be positive:

$$0 < 3\delta^2 - c^2u_1^2 \implies c^2u_1^2 < 3\delta^2 \implies \frac{k^2}{3}u_1^2 < \delta^4. \tag{4.20}$$

Since all the involved terms are positive we can maintain this inequality when we take the cubed root of both sides of the final inequality to get the left half of Equation (4.13).

To conclude our proof, when ς_1 is positive we will show that ς_2 will also be positive if $\delta < \frac{\sqrt{6}}{6}$. First, when ς_1 is positive we then have that, for some $\varepsilon > 0$, that

$$3\delta^2 = c^2u_1^2 + \varepsilon \implies 3\delta^2 - \varepsilon = c^2u_1^2. \tag{4.21}$$

This means that we can simplify our term related to ς_2 in Equation (4.19) as follows:

$$\begin{aligned} c^2u_1^2 - 6c^2u_1^2\delta^2 + 9\delta^4 &= (3\delta^2 - \varepsilon) - 6(3\delta^2 - \varepsilon)\delta^2 + 9\delta^4 \\ &= -9\delta^4 + (3 + 6\varepsilon)\delta^2 - \varepsilon. \end{aligned} \tag{4.22}$$

This polynomial is positive (as can be verified using the quadratic formula on with the substitution $z \triangleq \delta^2$ to find zeros and then checking intervals) whenever:

$$\frac{1 + 2\varepsilon}{6} - \frac{\sqrt{36\varepsilon^2 + 9}}{18} < \delta^2 < \frac{1 + 2\varepsilon}{6} + \frac{\sqrt{36\varepsilon^2 + 9}}{18}. \tag{4.23}$$

At this point we can substitute

$$\varepsilon = 3\delta^2 - c^2u_1^2 \tag{4.24}$$

into the above inequality and simplify to get:

$$9\delta^4 - 6c^2u_1^2\delta^2 + c^2u_1^2 > 0. \tag{4.25}$$

We then recall that $c \triangleq \frac{k}{\delta}$, so we substitute and simplify again to get:

$$9\delta^6 - 6k^2u_1^2\delta^2 + k^2u_1^2 > 0. \tag{4.26}$$

We then solve for u_1^2 and we get

$$u_1^2 > \frac{-9\delta^6}{k^2(1 - 6\delta^2)}. \tag{4.27}$$

We then note that whenever $1 - 6\delta^2 > 0$, that u_1 satisfies this condition for all positive

reaction rates, k. This finally brings us to the right side of Equation (4.13) as follows,

$$0 < 1 - 6\delta^2 \implies 6\delta^2 < 1 \implies \delta < \frac{1}{\sqrt{6}} = \frac{\sqrt{6}}{6}. \tag{4.28}$$

The final claim of the theorem is validated by x_1^{eq}'s smooth dependence on u_1 in Equation (4.17), so the relationship between x_1^{eq} and u_1 is bijective on a small enough interval that does not contain a local minimum or maximum.

The proof of Theorem 8 demonstrates that the positive constant m_a^{eq} is the concentration of the messenger RNA for the source node in the DOR's feedforward component of the DOR2SIM model. This same proof also demonstrates that m_a^{eq} is a function of the system parameters and the input to the same source node, u_1.

Theorem 8 implies that as long as the TFs and messenger RNA of the system do not degrade too rapidly with respect to the timescale of the transcription dynamics that this model will successfully classify sufficiently small, constant inputs. Theorem 8 is interesting in that monomer degradation rates are lower in cells that divide more slowly and previous research suggests a tradeoff between cell growth rate and environmental adaptation [50]. Theorem 8 is also encouraging as the negative auto regulation built into this model promotes a quick response time [52] and therefore the degradation parameter should be small by comparison. So this model should, under reasonable assumptions, successfully model dynamic classification for small enough inputs.

4.7 Using data to determine if higher-order DOR2SIM models classify

Numerical integration or experimental measurements of gene activity in a cell culture can be used to estimate a stable equilibrium point of the DOR2SIM model. However, the

existence of such an equilibrium point is not sufficient to establish a DOR2SIM model's ability to classify the constant input used to find that equilibrium point. It is possible for biological systems to showcase hysteresis [59]. This means that, although the measured or simulated trajectory (or trajectories) indicate a particular state that the system will map the input to, different initial conditions may be in the bastion of attraction for a different attracting set. Given a stable equilibrium point we provide sufficient results to verify that the DOR2SIM system will map all biologically viable initial conditions to the generalization of the class set, S_c, defined in Theorem 8. As before, S_c, will be defined in terms of the concentration of the source messenger RNA in the feedforward component of the DOR within the DOR2SIM model.

In this section, we retain a few of the our previous modeling assumptions, specifically, we assume that

1. all promoters activate (or repress) at the promoter site after they form a dimer (so all Hill coefficients are set to 2) and

2. half maximal suppression for the source messenger RNA is reached when the concentration of the protein it codes for is at 1.

Theorem 9. *Given a DOR2SIM system with a biologically feasible constant input, $\bar{u} = [u_1, ..., u_n]$, resulting in a biologically feasible equilibrium point at $x^{eq} \in \mathbb{R}^{2n+3N+1}$ assume that every biologically feasible initial condition will lie in the bastion of attraction of some stable equilibrium point and that the following condition is satisfied:*

$$k_{P_n}^2 x_n^{eq} (\frac{b_n u_n}{\delta_{2n}^2} - x_n^{eq}\delta_n)^2 - 4\delta_n \delta_{2n}^2 (\beta_n + b_n u_n) < 0, \tag{4.29}$$

where x_n^{eq} is the concentration of the source messenger RNA in the feedforward dynamics of the DOR at the equilibrium point, x^{eq}, and where $k_{P_n}, \delta_n, \delta_{2n}, b_n, \beta_n$ are all constant

parameters in the DOR2SIM model. Then the entire DOR2SIM system will classify all biologically feasible initial conditions to the class set:

$$S_c \triangleq \{x \in \mathbb{R}^{2n+3N+1} : x_n = x_n^{eq}, x_{2n} = \frac{k_{P_n}}{\delta_{2n}} x_n^{eq}\}. \tag{4.30}$$

Furthermore, this DOR2SIM model is an input classifying system over some sufficiently small intervals of the set of biologically feasible inputs that satisfies Equation (4.29).

Proof: We begin this proof in a form analogous to that of Theorem 8 and find that at equilibrium:

$$0 = k_{P_n} x_n - \delta_{2n} x_{2n} \implies x_{2n} = \frac{k_{P_n}}{\delta_{2n}} x_n. \tag{4.31}$$

We need to show that the n^{th} state at equilibrium must equal the n^{th} state of our known equilibrium point, specifically, that $x_n = x_n^{eq}$.

Again, as in Theorem 8, we define a convenience variable, $c \triangleq \frac{k_{P_n}}{\delta_{2n}}$ and derive another required condition at equilibrium point from the state-evolution of x_n:

$$0 = -\delta_n c^2 x_n^3 + b_n u_n c^2 x_n^2 - \delta_n x_n + (\beta_n + b_n u_n), \tag{4.32}$$

where x_n is the concentration of the n^{th} messenger RNA at equilibrium. Since all biologically feasible parameters and inputs to this system are positive Descartes rule of signs gives that this polynomial has 0 negative real solutions and 3 or 1 positive real solutions. We know that one positive real solution to this polynomial is $x_n = x_n^{eq}$. Therefore, to satisfy this equation with only one real solution, we must have that Equation (4.32) be writable as:

$$0 = (x_n - x_n^{eq})(\alpha x_n^2 + \beta x_n + \gamma), \tag{4.33}$$

with a negative discriminant:

$$\beta^2 - 4\alpha\gamma < 0. \tag{4.34}$$

We then divide out $x_n - x_N^{eq}$ from Equation (4.32) and equate the coefficients to α, β and γ to compute that:

$$\begin{aligned} \alpha &= -\delta_n c^2 \\ \beta &= b_n u_n c^2 - x_n^{eq}\delta_n c^2 \\ \gamma &= -(\beta_n + b_n u_n)/x_n^{eq}. \end{aligned} \tag{4.35}$$

Thus our negative discriminant condition, Equation (4.34), becomes Equation (4.29) and it implies that x_n^{eq} is the only real value for x_n which allows the system state x to be at equilibrium. Similar to the proof of Theorem 8, we note the smooth dependence of x_n^{eq} on u_n in Equation (4.32) to justify the final statement of the theorem.

Theorem 9 shows that we have found an architecture (the DOR2SIM model) that universally classifies biologically feasible environments under mathematical conditions that coincide a relatively strong activation of the source node in the DOR component compared to its concentration of messenger RNA at equilibrium ($\beta_n + b_n u_n$ large compared to x_n^{eq}). This suggests that the smaller the concentration of messenger RNA from the source node gene needed to keep the cell at equilibrium in one environment, the greater capacity that the cell has to classify based on smaller environmental signals and in spite of a less efficient promoter site. So this model suggests that the messenger RNA from source nodes in DOR2SIM motifs of highly adaptable bacteria will settle to low concentrations.

This theorem also provides a quick check to verify the ability of a particular DOR2SIM model to classify constant inputs of interest. This check could be used in conjunction with numerical simulations or a sequence of experiments that independently tweak the input values to look for possible limits of the system's ability to classify inputs.

4.8 Conclusion

In order to survive, bacteria and other cells need to rely on decision making that is computed via networks of chemical reactions that interplay with the central dogma of biology: the transcription and translation of messenger RNA. This means that these cells need to be able to classify chemical and other signals from their environments through these dynamics. We have combined typical biological motifs to propose the DOR2SIM model, a specific model for biological classification and response to environmental signals.

Our numerical analysis shows that this model is stable, consistent with experimentally measured transcriptomics and is capable of dynamic input classification. We have established a straightforward condition under which the minimal complexity DOR2SIM model is guaranteed to perform input classification. We have also established a sufficient condition to check the classification ability of an arbitrary DOR2SIM model receiving an arbitrary biologically consistent input. This condition is a simple computation that is made from the model parameters as well as an equilibrium point, which can be discovered through numerical simulation or from experimental data collection. The condition suggests that successful classification in these networks (and by extension, successful adaptation of cells to new environments) is tied to a maintaining a low concentration of messenger RNA from source nodes in the DOR at equilibrium. In future work we hope to leverage the network structure of the DOR2SIM model in order to identify this motif and dynamics in adaptable bacteria. We also hope to integrate the physical dynamics of DNA supercoiling into the model to understand the role that supercoiling plays in cell response to environmental stimulus.

Appendix A

Appendices

A.1 Proofs of theorems

In [31] we demonstrated that imposing a total order on conjunctive logistic basis functions implied some basic closure properties of the SILL dictionary. We show in this section proofs that apply this total order as steps to showing the uniform finite closure of the SILL and augSILL dictionaries. The needed total order is summed up in the following assumption.

Assumption 1. *There exists a total order on the set of conjunctive logistic functions,* $\Lambda(x;\theta_1), ..., \Lambda(x;\theta_l)$*, induced by the positive orthant* $\mathbb{R}^n_+$*, where* $\theta_l \gtrsim \theta_j$ *whenever* $\mu_j - \mu_l \in \mathbb{R}^n_+$*, and therefore* $\Lambda(x;\theta_l) \geq \Lambda(x;\theta_j)$*.*

Remark 5. *Given a finite SILL dictionary that does not satisfy Assumption 1, one can enforce that Assumption 1 holds by adding a finite number of additional conjunctive logistic functions to the basis.*

In these results we do not consider the error of approximating the vector field, f, we instead focus on the subspace invariance of a model built from a dictionary that already

captures the basic system dynamics. This assumption is given formally below.

Assumption 2. *The vector field, f, lies in the span of our SILL dictionary(or augSILL dictionary as appropriate).*

Proposition 1: Given a piecewise constant vector field, $f(x)$, that can be expressed as a finite combination of these n-dimensional step functions,

$$f_i(x) = \sum_{j=1}^{N_L} w_{ij} S(x; \mu_j), \forall i = 1, ..., n, \tag{A.1}$$

there exists a finite state-inclusive dictionary including n-dimensional step functions that will satisfy exact closure for all $x \in \mathbb{R}^n$ except for a set of points on a set of Lebesgue measure zero.

Proof: We prove this theorem by construction of the dictionary that satisfies exact closure for all $x \in \mathbb{R}^n$ except for a set of points on a set of Lebesgue measure zero. We define our dictionary as follows:

$$\psi(x) = \begin{bmatrix} x \\ \bar{S}(x) \end{bmatrix}, \tag{A.2}$$

where $\bar{S}$ is a vector of n-dimensional step functions whose centers are all possible combinations of μ_{ji} from Eq. (A.1). Now we need to show that $\frac{d\psi(x)}{dt}$ can be exactly approximated as a linear combination of elements of our dictionary, $\psi(x)$, on all of $\mathbb{R}^n$ except on a set with Lebesgue measure zero.

We first note that,

$$\frac{d\psi(x)}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{d\bar{S}(x)}{dt} \end{bmatrix} = \begin{bmatrix} f(x) \\ \frac{d\bar{S}(x)}{dt} \end{bmatrix}. \tag{A.3}$$

Due to Eq. (A.1), we know that $f(x)$ will be a linear combination of n-dimensional step

functions in $\bar{S}$, and therefore is in the span of our dictionary. Now we just need to show that

$$\frac{d\bar{S}(x)}{dt}$$

can be exactly approximated as a linear combination of elements of our dictionary, $\psi(x)$, on all of $\mathbb{R}^n$ except on a set with Lebesgue measure zero.

To do so we note that for all

$$x \notin M_{\bar{\Lambda}} \triangleq \{x : \exists i \in \{1, 2, ..., n\} \text{ and } j \in \{1, 2, ..., N_L\} \text{ so that } x_i = \mu_{ji}\}$$

the time derivative of the scalar function from Eq. (2.17) is

$$\frac{dS(x;\mu_l)}{dt} = \frac{dS(x;\mu_l)}{dx}\frac{dx}{dt} = (0)\frac{dx}{dt} = 0, \tag{A.4}$$

since, whenever $x \notin M_{\bar{\Lambda}}$, $dS(x;\mu_l)/dx = 0$. As $M_{\bar{\Lambda}}$ is a set of nN_L $n-1$ dimensional hyperplanes in $\mathbb{R}^n$, it has Lebesgue measure zero,

$$m(M_{\bar{\Lambda}}) = 0. \tag{A.5}$$

So, for each center parameter,μ_l, the Lie-derivative $\frac{dS(x;\mu_l)}{dt}$ is in the span of $\psi(x)$ on all of $\mathbb{R}^n$ except on a set of Lebesgue measure zero. This implies that $d\bar{S}(x)/dt$ can be exactly approximated as a linear combination of elements of our dictionary, $\psi(x)$, on all of $\mathbb{R}^n$ except on a set with Lebesgue measure zero.

Theorem 1: Under Assumption 1, if the dictionary functions do not exactly match their corresponding center parameters, $x_i \neq \mu_{ji}$ for all $i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., N\}$, then, as the steepness parameters go to infinity, the product of two conjunctive logistic function will exponentially approach a single conjunctive logistic function in the dictio-

nary, $\alpha \to \infty$,

$$\Lambda(x;\theta_l)\Lambda(x;\theta_j) - \Lambda(x;\theta^*) \to 0$$

exponentially for any $l \in \{1, 2, ..., N_L\}$.

Proof: We now investigate the error of the approximation. Eq. (2.26)'s error of approximation is

$$\begin{aligned} &\Lambda(x;\theta_l)\Lambda(x;\theta_j) - \Lambda(x;\theta_l) \\ &\quad = \frac{1 - \Lambda(x;\theta_j)^{-1}}{(\Lambda(x;\theta_l)\Lambda(x;\theta_j))^{-1}} \\ &\quad = \frac{1 - (1 + e^{-\alpha_{j1}(x_1 - \mu_{j1})})...(1 + e^{-\alpha_{jn}(x_n - \mu_{jn})})}{\prod_{i=1}^{n}(1 + e^{-\alpha_{li}(x_i - \mu_{li})})(1 + e^{-\alpha_{ji}(x_i - \mu_{ji})})}, \end{aligned} \tag{A.6}$$

whenever $\theta_l \gtrsim \theta_j$.

Without loss of generality, assume that $\theta_l \gtrsim \theta_j$. As we hold x constant and let $\alpha \to \infty$, for all $i \in \{1, 2, ..., n\}$, and any possible value of l, j we observe two cases. In these cases we denote a possible value of μ_{li}, μ_{ji} etc. as μ_*

Case 1, $x_i - \mu_* > 0$: As $\alpha \to \infty$ we have that $e^{-\alpha(x_i - \mu_*)} \to 0$ and so $\frac{1}{1+e^{-\alpha_i(x_i - \mu_*)}} \to 1$.

Case 2, $x_i - \mu_* < 0$: As $\alpha \to \infty$ we have that $e^{-\alpha_i(x_i - \mu_*)} \to \infty$ and so $\frac{1}{1+e^{-\alpha_i(x_i - \mu_*)}} \to 0$ exponentially.

So, if there exists $i \in \{1, 2, ..., n\}$ for every j, so that $x_i - \mu_{ji} < 0$, then Eq. (2.26) goes to 0 exponentially as $\alpha \to \infty$.

However, if $x_i - \mu_{ji} > 0$ for all i, then, since $\theta_l \gtrsim \theta_j$, for all i we have that $x_i - \mu_{li} > 0$ for all i as well, so Eq. (A.6) goes to $\frac{1-1}{1} = 0$ exponentially as $\alpha \to \infty$. ■

Corollary 1: Under the assumptions of Theorem 1, and assuming that f is spanned by a SILL dictionary, the Lie derivative of a conjunctive logistic function exponentially approaches a finite weighted sum of conjunctive logistic functions as the steepnesses of

the functions goes to infinity. Specifically,

$$\dot{\Lambda}(x;\theta_l) \rightarrow \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta^*) \tag{A.7}$$

exponentially as $\alpha \rightarrow \infty$.

Proof: We assume that f is spanned by our set of nonlinear dictionary functions.

This means that there exists a real-valued weighting matrix, $w \in \mathbb{R}^{n\times N_L}$, so that for any, $i \in \{1,2,...,n\}$, the i^{th} element of f, f_i, can be written as:

$$f_i(x) = \sum_{j=1}^{N_L} w_{ij}\Lambda(x,\theta_j). \tag{A.8}$$

Therefore, the time derivative of an arbitrary nonlinear observable in the SILL dictionary is:

$$\begin{aligned}
\dot{\Lambda}(x;\theta_l) &= (\nabla_x \Lambda(x;\theta_l))^T \frac{dx}{dt} = (\nabla_x \Lambda(x;\theta_l))^T f(x) \\
&= \sum_{i=1}^{n} \alpha_{li}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l) f_i(x) \\
&= \sum_{i=1}^{n} \alpha_{li}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l) \sum_{j=1}^{N_L} w_{ij}\Lambda(x;\theta_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)\Lambda(x;\theta_j).
\end{aligned} \tag{A.9}$$

Theorem 1 shows, that in the limit of steepness, the error between Eq. (A.9) and

$$\sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta^*) \tag{A.10}$$

will go to zero exponentially. ■

Lemma 1: If $||\mu_l - \mu_k||_\infty \neq 0$ or $\mu_{li} \neq \mu_{ki}$ for all $i \in \{1,2,...,m\}$ and all $k \in$

$\{N_L+1, N_L+2, ..., N\}$, then as $\alpha \to \infty$, $P(x;\theta_l)P(x;\theta_k) \to 0$ exponentially.

Proof: As we consider the approximation error of a product of two conjunctive RBFs with zero our error is the term:

$$P(x;\theta_l)P(x;\theta_k) = \prod_{i=1}^{n} \frac{e^{-\alpha_{li}(x_i-\mu_{li})}e^{-\alpha_{ki}(x_i-\mu_{ki})}}{(1+e^{-\alpha_{li}(x_i-\mu_{li})})^2(1+e^{-\alpha_{ki}(x_i-\mu_{ki})})^2}. \tag{A.11}$$

We proceed by looking at the i^{th} term in Eq. (A.11), we know how it develops as $\alpha \to \infty$ by examining the cases tabulated below. As in the proof of Theorem 3 we note that when we use "∞_c" we mean to say that the term grows to infinity with α with a rate of $e^{c\alpha}$, for some positive constant c. Furthermore, we use "0_c" to mean that the term goes to zero with a rate of $e^{-c\alpha}$ for some positive constant c.

Cases	$x_i < \mu_{li}$	$x_i = \mu_{li}$	$x_i > \mu_{li}$
$x_i < \mu_{ki}$	$\frac{\infty_{c_l}\infty_{c_k}}{\infty^2_{c_l}\infty^2_{c_k}} \to 0$	$\frac{\infty_{c_k}}{\infty^2_{c_k}} \to 0$	β_1
$x_i = \mu_{ki}$	$\frac{\infty_{c_l}}{\infty^2_{c_l}} \to 0$	$\frac{1}{16}$	$\frac{0_{c_l}}{2} \to 0$
$x_i > \mu_{ki}$	β_2	$\frac{0_{c_k}}{2} \to 0$	$\frac{0_{c_l}0_{c_k}}{1} \to 0$

We first comment that the case in the center will occur in at most $n-1$ of the terms of Eq. (A.11) by our assumption that $||\mu_l - \mu_k||_\infty \neq 0$. We then note that given β_1 we have three cases:

Case 1, $-\alpha_{li}(x_i-\mu_{li}) - \alpha_{ki}(x_i-\mu_{ki}) < 0$. In this case we have that as $\alpha \to \infty$ that our term goes to $\frac{0_{c_l-c_k}}{\infty^2_{c_k}} \to 0$.

Case 2, $-\alpha_{li}(x_i-\mu_{li}) - \alpha_{ki}(x_i-\mu_{ki}) = 0$. In this case we have that as $\alpha \to \infty$ that our term goes to $\frac{1}{\infty^2_{c_k}} \to 0$.

Case 3, $-\alpha_{li}(x_i-\mu_{li}) - \alpha_{ki}(x_i-\mu_{ki}) > 0$. In this case we have that as $\alpha \to \infty$ that our term goes to $\frac{\infty_{c_k-c_l}}{\infty^2_{c_k}} \to 0$. This fraction approaches zero exponentially as the constant, c_l, is positive by the assumption that $x_i > \mu_{li}$.

We now note that without loss of generality that these three cases cover β_2 (one

simply swaps α terms) and so we have that as $\alpha \to \infty$ at least one term in our product will go to zero exponentially and the rest at most will go to $\frac{1}{16}$. Thus the error term goes to zero exponentially. ■

Theorem 4: If $x_i \neq \mu_{ki}$ for all $i \in \{1, 2, ..., m\}$ and $k \in \{N_L + 1, N_L + 2, ..., N\}$, then as $\alpha \to \infty$, $P(x; \theta_l)P(x; \theta_k) \to 0$ exponentially.

Proof: This proof is very similar to the proof of Lemma 1. The only difference is that from the table of cases, only the four corner cases can occur. ■

In the following proofs we assume that our augSILL observables span the vector field we seek to model (see Assumption 2). Define $f_i(x)$ to be the i^{th} entry of the vector field, f. Then $f_i(y)$ may be written as the following weighted sum:

$$f_i(y) = \sum_{j=1}^{N_L} w_{ij}\Lambda(x; \theta_j) + \sum_{k=N_L+1}^{N_L+N_R} w_{ik}P(x; \theta_k). \tag{A.12}$$

It is reasonable to assume that $f_i(x)$ can be written as such as sum as we assume f and x to be analytic, which implies that f is smooth. This means that a sufficiently rich basis of logistic and RBFs will approximate each f_i in such a manner. In the real Koopman learning problem we would also have a constant and weighted sum of the state to help approximate each f_i. Using these extra terms to approximate f_i in our analysis makes the math more cumbersome, and so we represent f_i without them. The class of representable vector fields, fs, is extremely broad as both logistic and RBFs are universal function approximators [60], [61].

Corollary 3: Under the assumptions of Theorem 1 and the assumption that the augSILL observables span the vector field we seek to model, the Lie derivative of a conjunctive logistic function exponentially approaches a nonlinear combination of augSILL

functions, specifically,

$$\begin{aligned}\dot{\Lambda}(x;\theta_l) \rightarrow &\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta^*) \\ &+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}(1-\lambda(x_i;\theta_{li}))H(x;\theta_l,\theta_k)\end{aligned} \tag{A.13}$$

exponentially as $\alpha \rightarrow \infty$.

Proof: The derivative of a conjunctive logistic function, $\Lambda(x;\theta_l)$, with respect to time is

$$\dot{\Lambda}(x;\theta_l) = \left(\nabla_x \Lambda(x;\theta_l)\right)^T \frac{dx}{dt} = \left(\nabla_x \Lambda(x;\theta_l)\right)^T f(x), \tag{A.14}$$

where the i^{th} term of the gradient of $\Lambda(x;\theta_l)$ is

$$\begin{aligned}\left[\nabla_x \Lambda(x;\theta_l)\right]_i &= \alpha_{li}(\lambda(x_i;\theta_{li}) - \lambda(x_i;\theta_{li})^2)\frac{\Lambda(x;\theta_l)}{\lambda(x_i;\theta_{li})} \\ &= \alpha_{li}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l).\end{aligned} \tag{A.15}$$

Thus the time-derivative of $\Lambda(x_i;\theta_l)$ is

$$\begin{aligned}\dot{\Lambda}(x;\theta_l) &= \sum_{i=1}^{n}\alpha_{li}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)f_i(x) \\ &= \sum_{i=1}^{n}\alpha_{li}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)(\sum_{j=1}^{N_L}w_{ij}\Lambda(x;\theta_j) \\ &+ \sum_{k=N_L+1}^{N_L+N_R}w_{ik}P(x;\theta_k)) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)\Lambda(x;\theta_j) \\ &+ \sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}(1-\lambda(x_i;\theta_{li}))\Lambda(x;\theta_l)P(x;\theta_k).\end{aligned} \tag{A.16}$$

From Theorem 1, Theorem 2 and Theorem 3 we have that Eq. (A.16) goes to Eq. (2.45) as $\alpha \to \infty$. ■

Corollary 4: Under the assumptions of Theorem 1 and the assumption that the augSILL observables span the vector field we seek to model, the Lie derivative of a conjunctive RBF exponentially approaches a nonlinear combination of conjunctive RBFs, specifically,

$$\dot{P}(x;\theta_l) \to \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li} w_{ij} (1 - 2\lambda(x_i;\theta_{li})) H(x;\theta_j,\theta_l) \tag{A.17}$$

exponentially as $\alpha \to \infty$.

Proof: The derivative of a conjunctive RBF is:

$$\dot{P}(x;\theta_l) = (\nabla_x P(x;\theta_l))^T \frac{dx}{dt} = (\nabla_x P(x;\theta_l))^T f(x) \tag{A.18}$$

where the i^{th} term of the gradient of $P(x;\theta_l)$ is

$$\begin{aligned} [\nabla_x P(x;\theta_l)]_i &= \alpha_{li}\rho(x_i;\theta_{li})(1 - 2\lambda(x_i;\theta_{li}))\frac{P(x;\theta_l)}{\rho(x_i;\theta_{li})} \\ &= \alpha_{li}(1 - 2\lambda(x_i;\theta_{li}))P(x;\theta_l). \end{aligned} \tag{A.19}$$

So, the time-derivative of $P(x;\theta_l)$ is

$$\begin{aligned}
\dot{P}(x;\theta_l) &= \sum_{i=1}^{n} \alpha_{li}(1-2\lambda(x_i;\theta_{li}))P(x;\theta_l)f_i(x) \\
&= \sum_{i=1}^{n} \alpha_{li}(1-2\lambda(x_i;\theta_{li}))P(x;\theta_l)(\sum_{j=1}^{N_L} w_{ij}\Lambda(x;\theta_j) \\
&+ \sum_{k=N_L+1}^{N_L+N_R} w_{ik}P(x;\theta_k)) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{N_L} \alpha_{li}w_{ij}(1-2\lambda(x_i;\theta_{li}))P(x;\theta_l)\Lambda(x;\theta_j) \\
&+ \sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R} \alpha_{li}w_{ik}(1-2\lambda(x_i;\theta_{li}))P(x;\theta_l)P(x;\theta_k).
\end{aligned} \tag{A.20}$$

From Theorems 2, 3 and 4 we have that Eq. (A.20) goes to Eq. (2.46) as $\alpha \to \infty$. ■

A.2 PDF of X(Y-Z) and logistic functions

Given the symmetric uniform distributed random variables X, Y and Z. We can compute the PDF of X(Y-Z), a term in logistic functions. The random variables X and Z represent the scalar parameters and the random variable Y represents the scalar state value. So, X(Y-Z) corresponds to $\alpha(x-\mu)$. We choose X, Y and Z to be identically and independently distributed (iid) as the symmetric uniform distribution: $U(-a,a), a \in \mathbb{R}^+$. We then apply the law of the unconscious statistician (LOTUS) to compute the expected value of a logistic function.

We compute the PDF of X(Y-Z), where the random variables X, Y and Z are independently and identically distributed as the symmetric uniform distribution: $U(-a,a), a \in \mathbb{R}^+$.

We start by computing the PDF of (Y-Z). The PDF of each random variable is

$$f_U(x) = \begin{cases} \frac{1}{2a} & \text{if } x \in [-a, a] \\ 0 & \text{otherwise.} \end{cases} \tag{A.21}$$

Since the random variables are symmetrically distributed, this is the same distribution as (Y+Z), a symmetric triangular distribution. The PDF is

$$f_T(x) = \begin{cases} \frac{1}{2a} + \frac{x}{4a^2} & \text{if } x \in [-2a, 0) \\ \frac{1}{2a} - \frac{x}{4a^2} & \text{if } x \in (0, 2a] \\ 0 & \text{otherwise.} \end{cases} \tag{A.22}$$

Now we use the formula for the distribution of a product of random variables,

$$g(z) = \int_{-\infty}^{\infty} f_T(x) f_U(z/x) \frac{1}{|x|} dx, \tag{A.23}$$

to compute the PDF of X(Y-Z). So,

$$
\begin{aligned}
g(z) = \int_{-\infty}^{\infty} & \begin{cases} \frac{1}{2a} + \frac{x}{4a^2} & \text{if } x \in [-2a, 0) \\ \frac{1}{2a} - \frac{x}{4a^2} & \text{if } x \in (0, 2a] \\ 0 & \text{otherwise} \end{cases} \\
& \times \begin{cases} \frac{1}{2a|x|} & \text{if } \frac{z}{x} \in [-a, a] \\ 0 & \text{otherwise} \end{cases} dx \\
= & \int_0^{2a} (\frac{1}{2a} - \frac{x}{4a^2}) \begin{cases} \frac{1}{2ax} & \text{if } \frac{z}{x} \in [-a, a] \\ 0 & \text{otherwise} \end{cases} \\
& - \int_{-2a}^{0} (\frac{1}{2a} + \frac{x}{4a^2}) \begin{cases} \frac{1}{2ax} & \text{if } \frac{z}{x} \in [-a, a] \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{A.24}
$$

The following equivalences hold, however they are only equal to $g(z)$ when $\frac{|z|}{a} \leq 2a$ or, alternatively written, $z \in [-2a^2, 2a^2]$.

$$
\begin{aligned}
g(z) &= \frac{1}{4a^2} \int_{\frac{|z|}{a}}^{2a} (\frac{1}{x} - \frac{1}{2a}) - \frac{1}{4a^2} \int_{-2a}^{\frac{-|z|}{a}} (\frac{1}{x} + \frac{1}{2a}) \\
&= \frac{1}{4a^2} (\ln|x| - \frac{x}{2a})|_{x=\frac{|z|}{a}}^{x=2a} \\
&- \frac{1}{4a^2} (\ln|x| + \frac{x}{2a})|_{x=-2a}^{x=\frac{-|z|}{a}} \\
&= \frac{1}{4a^2} (\ln(2a) - \ln(\frac{|z|}{a}) - 1 + \frac{|z|}{2a^2}) \\
&- \frac{1}{4a^2} (\ln(\frac{|z|}{a}) - \ln(2a) - \frac{|z|}{2a^2} + 1) \\
&= \frac{1}{2a^2} (\ln\left(\frac{2a^2}{|z|}\right) + \frac{|z|}{2a^2} - 1).
\end{aligned} \tag{A.25}
$$

So, we have that our final PDF is

$$g(z) = \begin{cases} \frac{1}{2a^2}(\ln(\frac{2a^2}{|z|}) + \frac{|z|}{2a^2} - 1) & \text{if } z \in I \\ 0 & \text{else,} \end{cases} \tag{A.26}$$

where I is the interval $[-2a^2, 2a^2]$.

With this PDF we use the LOTUS to compute the expected value of a randomly sampled logistic function as

$$E[\lambda] = \int_{-\infty}^{\infty} g(x) \left(\frac{1}{1 + e^{-x}} \right) dx. \tag{A.27}$$

We compute the variance using the formula: $Var[X] = E[X^2] - E[X]^2$, which involves an additional application of the LOTUS.

To the authors' knowledge, none of these integrals has a closed form solution in terms of the sampling interval parameter, a. Hence, we use numeric simulation to understand this relationship (see Fig. 2.4 and Fig. 2.5).

A.3 Dynamic systems for numeric simulations

The 2D systems we test in Section 2.9 are:

1. the Van Der Pol Oscillator:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + c_1(1 - x_1^2)x_2, \end{aligned} \tag{A.28}$$

2. the Duffing Oscillator:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -c_2 x_2 - c_3 x_1 - c_4 x_1^3, \end{aligned} \tag{A.29}$$

3. the Predator-Prey System:

$$\begin{aligned} \dot{x}_1 &= c_5 x_1 - c_6 x_1 x_2 \\ \dot{x}_2 &= c_7 x_1 x_2 - c_8 x_2, \end{aligned} \tag{A.30}$$

4. and the Toggle Switch:

$$\begin{aligned} \dot{x}_1 &= \frac{c_9}{1 + x_2^{c_{11}}} - c_{13} x_1 \\ \dot{x}_2 &= \frac{c_{10}}{1 + x_1^{c_{12}}} - c_{13} x_2, \end{aligned} \tag{A.31}$$

where $c_1 = 1, c_2 = 0, c_3 = -1, c_4 = 1, c_5 = 1.1, c_6 = 0.5, c_7 = 0.1, c_8 = 0.2, c_9 = 2.5, c_{10} = 1.5, c_{11} = 1.4, c_{12} = 1.1,$ and $c_{13} = 0.25$.

A.4 Tabulated summary of error bounds for demonstrating uniform finite approximate closure (approximate subspace invariance)

We summarize the error bounds described in Section 2.8.3 on Table A.2. Please note the following.

Rows 1 and 2 with rows 5 and 6 gives a bound on the augSILL dictionary's ability to approximate the Lie derivatives of each nonlinear augSILL dictionary function. Rows 3 and 4 with rows 7 and 8 also give their own bound. The bounds' constant upper limit implies uniform finite approximate closure, and under reasonable assumptions each error bound exponentially approaches zero.

This is under the assumption that the Lie derivatives of the linear terms in the

augSILL dictionary are in the subspace defined by the augSILL dictionary. Note that the Lie derivative of the constant term in the augSILL dictionary is trivially in the subspace of the dictionary functions. Also note that the error term, $\varepsilon_{\Lambda_l \Lambda_j}$, and the set, M, in the third column are as defined in Section 2.6.

We approx.	With	The error bound is	Goes to zero as	When	See
$\dot{\Lambda}_l$	Eq. (2.45)	$\max_{x\in M}\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}(1-\lambda(x_i;\theta_{li}))\varepsilon_{\Lambda_l\Lambda_j}(x)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}(1-\lambda(x_i;\theta_{li}))\varepsilon_{\Lambda_l P_k}(x)$	$\alpha\to\infty$	$x\in\mathbb{R}^n - M_{\bar{\Lambda},\bar{P}}$	Corollary 1
$\dot{P}_l$	Eq. (2.46)	$\max_{x\in M}\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}(1-2\lambda(x_i;\theta_{li}))\varepsilon_{P_l\Lambda_j}(x)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}(1-2\lambda(x_i;\theta_{li}))\varepsilon_{P_l P_k}(x)$	$\alpha\to\infty$	$x\in\mathbb{R}^n - M_{\bar{\Lambda},\bar{P}}$	Corollary 2
Eq. (2.47)	Eq. (2.48)	$\max_{x\in M}\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\varepsilon_{\Lambda_l\Lambda_j}(x)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\varepsilon_{\Lambda_l P_k}(x)$	$\alpha\to\infty$	$x\in\mathbb{R}^n - M_{\bar{\Lambda},\bar{P}}$	Corollary 3
Eq. (2.49)	Eq. (2.50)	$\max_{x\in M}\sum_{i=1}^{n}\sum_{j=1}^{N_L}\alpha_{li}w_{ij}\varepsilon_{P_l\Lambda_j}(x)$ $+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\alpha_{li}w_{ik}\varepsilon_{P_l P_k}(x)$	$\alpha\to\infty$	$x\in\mathbb{R}^n - M_{\bar{\Lambda},\bar{P}}$	Corollary 4

Table A.1: Part 1 of a summary of the error bounds described in this paper to demonstrate uniform finite approximate closure. The variable, c, is a bounding constant. Recall that each constant ν_{ij} is bounded by the product of a weight and steepness parameter. For the purpose of these bounds, these parameters are samples from a uniform distribution defined on the interval $[-a, a]$ for some positive real number a. Finally, T1 is short for Table 1.

We approx.	With	The error bound is	Goes to zero as	When	See
Eq. (2.45)	T1: row 1, col 3	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{n+1}}+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{3n+1}}$	$n\to\infty$	$\lvert\theta\rvert,\lvert K\rvert<c$	T1: row 1
Eq. (A.16)	T1: row 2, col 3	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{2n+1}}+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{3n+1}}$	$n\to\infty$	$\lvert\theta\rvert,\lvert K\rvert<c$	T1: row 2
Eq. (2.46)	T1: row 3, col 3	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{3n+1}}$	$n\to\infty$	$\lvert\theta\rvert,\lvert K\rvert<c$	T1: row 3
Eq. (A.20)	T1: row 4, col 3	$\sum_{i=1}^{n}\sum_{j=1}^{N_L}\frac{\nu_{ij}}{2^{3n+1}}+\sum_{i=1}^{n}\sum_{k=N_L+1}^{N_L+N_R}\frac{\nu_{ik}}{2^{4n+1}}$	$n\to\infty$	$\lvert\theta\rvert,\lvert K\rvert<c$	T1: row 4

Table A.2: Part 2 of a summary of the error bounds described in this paper to demonstrate uniform finite approximate closure. The variable, c, is a bounding constant. Recall that each constant ν_{ij} is bounded by the product of a weight and steepness parameter. For the purpose of these bounds, these parameters are samples from a uniform distribution defined on the interval $[-a, a]$ for some positive real number a. Finally, T1 is short for Table 1.

A.5 Solving for positive and negative definite matrices

Satisfying the conditions of Theorem 5 requires discovering block matrices so that the matrix

$$\begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \tag{A.32}$$

is positive definite and the matrix

$$\begin{bmatrix} Q_1 & Q_2 \\ Q_2^T & Q_3 \end{bmatrix} \begin{bmatrix} K_1 + BP_1 & K_2 + BP_2 \\ K_3 & K_4 \end{bmatrix} \tag{A.33}$$

is negative definite.

Solving for these Q, P (and potentially K) block matrices is equivalent to solving a mostly unconstrained system of equations using the Cholesky decomposition of the desired positive and negative definite matrices. The problem is to solve for the full matrices F_1, F_2, P_1 and P_2 (and potentially K_1, K_2, K_3 and/or K_4 depending on the scenario) and the lower-triangular matrices L_1, L_2, L_3 and L_4, so that

$$\begin{aligned} L_3 L_3^T &= -L_1 L_1^T (K_1 + BP_1) - L_1 F_1^T K_3, \\ F_2 L_3^T &= -F_1 L_1^T (K_1 + BP_1) - (F_1 F_1^T + L_2 L_2^T) K_3, \\ L_3 F_2^T &= -L_1 L_1^T (K_2 + BP_2) - L_1 F_1^T K_4, \text{ and} \\ F_2 F_2^T + L_4 L_4^T &= -F_1 L_1^T (K_2 + BP_2) - (F_1 F_1^T + L_2 L_2^T) K_4. \end{aligned} \tag{A.34}$$

Then, let $Q_1 = L_1 L_1^T, Q_2 = L_1 F_1^T$ and $Q_3 = F_1 F_1^T + L_2 L_2^T$. The only constraint that one needs to enforce on the matrices in this system of equations is that each of the lower-triangular matrices has non-zero diagonals (zeros on the diagonals gives a positive

semi-definite Q matrix).

Bibliography

[1] 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.

[2] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, *Nonlinear Dynamics* **41** (2005), no. 1 309–325.

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

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

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

[6] 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** (2018), no. 1 909–930.

[7] S. P. Nandanoori, S. Kundu, S. Pal, K. Agarwal, and S. Choudhury, *Model-agnostic algorithm for real-time attack identification in power grid using koopman modes*, in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, IEEE, 2020.

[8] A. Hasnain, N. Boddupalli, S. Balakrishnan, and E. Yeung, *Steady state programming of controlled nonlinear systems via deep dynamic mode decomposition*, in *2020 American Control Conference (ACC)*, pp. 4245–4251, IEEE, 2020.

[9] A. Hasnain, S. Balakrishnan, D. M. Joshy, J. Smith, S. B. Haase, and E. Yeung, *Learning perturbation-inducible cell states from observability analysis of transcriptome dynamics*, *Nature Communications* **14** (2023), no. 1 3148.

[10] A. Mauroy, Y. Susuki, and I. Mezić, *The Koopman Operator in Systems and Control*. Springer, 2020.

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

[12] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, *A kernel-based approach to data-driven koopman spectral analysis*, *arXiv preprint arXiv:1411.2260* (2014).

[13] 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** (2015), no. 6 1307–1346.

[14] S. Sinha, B. Huang, and U. Vaidya, *Robust approximation of koopman operator and prediction in random dynamical systems*, in *2018 Annual American Control Conference (ACC)*, pp. 5491–5496, IEEE, 2018.

[15] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, *Proceedings of the national academy of sciences* **113** (2016), no. 15 3932–3937.

[16] M. Haseli and J. Cortés, *Learning koopman eigenfunctions and invariant subspaces from data: Symmetric subspace decomposition*, *IEEE Transactions on Automatic Control* (2021).

[17] C. Wehmeyer and F. Noé, *Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics*, *The Journal of chemical physics* **148** (2018), no. 24 241703.

[18] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, *Nature communications* **9** (2018), no. 1 1–10.

[19] S. E. Otto and C. W. Rowley, *Linearly recurrent autoencoder networks for learning dynamics*, *SIAM Journal on Applied Dynamical Systems* **18** (2019), no. 1 558–593.

[20] N. Takeishi, Y. Kawahara, and T. Yairi, *Learning koopman invariant subspaces for dynamic mode decomposition*, *arXiv preprint arXiv:1710.04340* (2017).

[21] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, *Neural networks* **4** (1991), no. 2 251–257.

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

[23] A. Lasota and M. C. Mackey, *Chaos, fractals, and noise: stochastic aspects of dynamics*, vol. 97. Springer Science & Business Media, 1998.

[24] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, *Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control*, *PloS one* **11** (2016), no. 2 e0150171.

[25] A. Mauroy and J. Goncalves, *Koopman-based lifting techniques for nonlinear systems identification*, *IEEE Transactions on Automatic Control* **65** (2019), no. 6 2550–2565.

[26] M. Korda and I. Mezić, *On convergence of extended dynamic mode decomposition to the koopman operator*, *Journal of Nonlinear Science* **28** (2018), no. 2 687–710.

[27] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, *Journal of machine learning research* **12** (2011), no. 7.

[28] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[29] 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.

[30] C. Zhang and E. Zuazua, *A quantitative analysis of koopman operator methods for system identification and predictions*, *Comptes Rendus. Mécanique* **351** (2023), no. S1 1–31.

[31] C. A. Johnson and E. Yeung, *A class of logistic functions for approximating state-inclusive koopman operators*, in *2018 Annual American Control Conference (ACC)*, pp. 4803–4810, IEEE, 2018.

[32] S. G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, *IEEE Transactions on signal processing* **41** (1993), no. 12 3397–3415.

[33] B. C. Daniels and I. Nemenman, *Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression*, *PloS one* **10** (2015), no. 3 e0119821.

[34] Z. Liu, N. Ozay, and E. D. Sontag, *On the non-existence of immersions for systems with multiple omega-limit sets*, *IFAC-PapersOnLine* **56** (2023), no. 2 60–64.

[35] J. Harrison and E. Yeung, *Stability analysis of parameter varying genetic toggle switches using koopman operators*, *Mathematics* **9** (2021), no. 23 3133.

[36] E. Yeung, Z. Liu, and N. O. Hodas, *A koopman operator approach for computing and balancing gramians for discrete time nonlinear systems*, in *2018 Annual American Control Conference (ACC)*, pp. 337–344, IEEE, 2018.

[37] A. Mauroy, Y. Susuki, and I. Mezić, *Koopman operator in systems and control*. Springer, 2020.

[38] S. Balakrishnan, A. Hasnain, R. Egbert, and E. Yeung, *The effect of sensor fusion on data-driven learning of koopman operators*, *arXiv preprint arXiv:2106.15091* (2021).

[39] S. E. Otto and C. W. Rowley, *Koopman operators for estimation and control of dynamical systems*, *Annual Review of Control, Robotics, and Autonomous Systems* **4** (2021) 59–87.

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

[41] A. Krolicki, S. Sutavani, and U. Vaidya, *Koopman-based policy iteration for robust optimal control*, in *2022 American Control Conference (ACC)*, pp. 1317–1322, IEEE, 2022.

[42] G. Mamakoukas, M. Castano, X. Tan, and T. Murphey, *Local koopman operators for data-driven control of robotic systems*, in *Robotics: science and systems*, 2019.

[43] H. Shi and M. Q.-H. Meng, *Deep koopman operator with control for nonlinear systems*, *IEEE Robotics and Automation Letters* **7** (2022), no. 3 7700–7707.

[44] E. Kaiser, J. N. Kutz, and S. L. Brunton, *Data-driven discovery of koopman eigenfunctions for control*, *Machine Learning: Science and Technology* **2** (2021), no. 3 035023.

[45] D. Goswami and D. A. Paley, *Global bilinearization and controllability of control-affine nonlinear systems: A koopman spectral approach*, in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 6107–6112, IEEE, 2017.

[46] D. Goswami and D. A. Paley, *Bilinearization, reachability, and optimal control of control-affine nonlinear systems: A koopman spectral approach*, *IEEE Transactions on Automatic Control* **67** (2021), no. 6 2715–2728.

[47] M. Korda and I. Mezić, *Optimal construction of koopman eigenfunctions for prediction and control*, *IEEE Transactions on Automatic Control* **65** (2020), no. 12 5114–5129.

[48] C. A. Johnson, S. Balakrishnan, and E. Yeung, *Subspace invariance in koopman operators–part 2: Heterogeneous dictionary mixing to approximate subspace invariance*, *Automatica* (2023).

[49] F. Jacob and J. Monod, *Genetic regulatory mechanisms in the synthesis of proteins*, *Journal of molecular biology* **3** (1961), no. 3 318–356.

[50] L. López-Maury, S. Marguerat, and J. Bähler, *Tuning gene expression to changing environments: from rapid responses to evolutionary adaptation*, *Nature Reviews Genetics* **9** (2008), no. 8 583–593.

[51] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, *Network motifs in the transcriptional regulation network of escherichia coli*, *Nature genetics* **31** (2002), no. 1 64–68.

[52] U. Alon, *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC, 2019. **Note:** We used models and information from chapters 1, 2 and 4 in this paper.

[53] M. Csete and J. Doyle, *Bow ties, metabolism and disease*, *TRENDS in Biotechnology* **22** (2004), no. 9 446–450.

[54] T. Friedlander, A. E. Mayo, T. Tlusty, and U. Alon, *Evolution of bow-tie architectures in biology*, *PLoS computational biology* **11** (2015), no. 3 e1004055.

[55] D. Del Vecchio and R. M. Murray, *Biomolecular feedback systems*. Princeton University Press Princeton, NJ, 2015. **Note:** We used models and information from chapters 1 and 2 in this paper.

[56] J. Gonçalves and S. Warnick, *Necessary and sufficient conditions for dynamical structure reconstruction of lti networks*, *IEEE Transactions on Automatic Control* **53** (2008), no. 7 1670–1674.

[57] D. Materassi and G. Innocenti, *Topological identification in networks of dynamical systems*, *IEEE Transactions on Automatic Control* **55** (2010), no. 8 1860–1871.

[58] M. Santillán, *On the use of the hill functions in mathematical models of gene regulatory networks*, *Mathematical Modelling of Natural Phenomena* **3** (2008), no. 2 85–97.

[59] D. Angeli, J. E. Ferrell Jr, and E. D. Sontag, *Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems*, *Proceedings of the National Academy of Sciences* **101** (2004), no. 7 1822–1827.

[60] A. R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, *IEEE Transactions on Information theory* **39** (1993) 930–945.

[61] M. D. Buhmann, *Radial basis functions: theory and implementations*, vol. 12. Cambridge university press, 2003.