NORTHWESTERN UNIVERSITY

Increasing Robustness in Koopman-Based Feedback Stabilization

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Mechanical Engineering

By

Jonathan M. Bosnich

EVANSTON, ILLINOIS

December 2024

ABSTRACT

Increasing Robustness in Koopman-Based Feedback Stabilization

Jonathan M. Bosnich

In this thesis, we develop a data-driven stabilizing feedback control algorithm for nonlinear control systems using applied Koopman operator methods. Our stabilizing control algorithm both offers increased robustness with respect to model error and minimizes control effort to achieve this robust stabilization. We use measurements of the unknown nonlinear control system to learn an approximate linear time-invariant (LTI) model using the algorithm described in [**1**]. This data-driven modeling algorithm is one of many that is built upon the power of the Koopman operator, which is an infinite-dimensional operator that transforms a finite-dimensional nonlinear system into an infinite-dimensional linear system. Koopman-based modeling and control approximates this infinite-dimensional system from measured data and a user-chosen dictionary of lifting functions, called observables. One of the prominent challenges with Koopman-based methods, and data-driven modeling methods in general, is the inability to verify how accurate the model is with respect to the true system, which is especially worrisome since it's been well-documented that Koopman-based methods can learn a model that is fundamentally different from the

system it's approximating. This issue is addressed for autonomous dynamical systems, i.e., systems with no control input, in [**2**] with the introduction of residual dynamic mode decomposition (ResDMD). The ResDMD algorithm produces error bounds on the distance of each eigenvalue of the data-driven model to the spectrum of the true nonlinear system, under assumptions on how much data is collected and how it's collected. We generalize this work to nonlinear systems with control input, which we call *ResDMD with control* and abbreviate as *ResDMDc*. Using the eigenvalue error bounds from ResDMDc, we synthesize a feedback controller on the data-driven model that can stabilize the underlying nonlinear system with robustness to model error. Furthermore, we synthesize this feedback controller such that it stabilizes the nonlinear system with minimal-norm feedback gains by generalizing the optimization algorithm and solver described in [**3**], which only considered stability of continuous-time systems. Our new algorithm computes $\mathcal{D}(0,\delta)$-stabilizing minimal-norm control gains, where $\mathcal{D}(0,\delta) \subset \mathbb{C}$ is a disk of radius δ centered at the origin, and an LTI system is said to be $\mathcal{D}(0,\delta)$-stable if all of its eigenvalues lie in $\mathcal{D}(0,\delta)$. Using basic example systems, we (1) verify the ResDMDc algorithm yields faithful error bounds, while also qualifying the results in light of the assumptions on data collection; (2) show that our new minimal-norm feedback algorithm reliably pushes eigenvalues into $\mathcal{D}(0,\delta)$; and (3) demonstrate the performance of our overall control algorithm. Finally, we describe future directions and improvements, including desired theoretical results and considerations for implementing this algorithm on a physical robot.

Acknowledgements

I would like to express my deepest gratitude to Professor Todd Murphey, my advisor, for his unwavering support, insightful guidance, and inspiring mentorship throughout the course of my PhD journey. I am equally grateful to Professor Niall Mangan for serving on my thesis committee and for providing invaluable feedback and encouragement that greatly shaped this work. My heartfelt thanks go to Dr. Giorgos Mamakoukas, whose collaboration and expertise were critical in advancing this research. I am also deeply appreciative of Allie Pinosky, my labmate and mentor, whose advice and camaraderie have been a source of strength and growth. To Jordan Keeley, my girlfriend, thank you for your love, patience, and steadfast support, which have been my anchor during the highs and lows of this journey. Finally, I am profoundly grateful to my family for their unwavering belief in me and for their endless encouragement, without which none of this would have been possible. This dissertation is as much a testament to their support as it is to my academic endeavors.

Nomenclature

The following mathematical notation will be used throughout this thesis.

$\mathbb{R}^n$	n-dimensional Euclidean space
$\mathbb{C}$	space of complex numbers
$\mathbf{x} \in \mathbb{R}^n$ or $\mathbb{C}^n$	all vectors will be in bold
$\mathbb{Z}^{\geq 0}$	the non-negative integers
$L^2(\Omega, \omega)$	space of square-integrable functions over Ω with measure ω
$\mathcal{K}$	infinite-dimensional Koopman operator
$\mathbb{K}$	finite-dimensional Koopman operator approximation
$\Re(z)$	real part of a complex number $z \in \mathbb{C}$
$\lvert z \rvert$	magnitude of a complex number $z \in \mathbb{C}$
$\overline{z}$	complex conjugate of a complex number $z \in \mathbb{C}$
A^*	complex conjugate transpose of a complex matrix $A \in \mathbb{C}^{n \times n}$
I_n	$n \times n$ identity matrix

Table 0.1. Table of mathematical notation.

Dedication

I dedicate this thesis to my sister, Katherine.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction and Organization

Modeling and control of nonlinear systems has been studied extensively and yet remains a grand challenge for controls engineering and applied science in general. A theoretically sound and practically applicable framework for nonlinear control will have a remarkably broad impact. Currently, the most common method for analyzing nonlinear systems and synthesizing control is to iteratively linearize the dynamics locally about a point, e.g., iLQR [**5**]. There are methods that globally linearize the dynamics, such as feedback linearization [**6**] and Carleman linearization [**7**], but these global linearizations can only be found under restrictive assumptions. There are many other nonlinear control techniques, such as Lyapunov methods [**6**] and sum of squares programming [**8**], each with their own pros and cons. Common among all of these nonlinear control methods is the necessity for an exact model of the underlying system derived from first principles. In most engineering applications, this is an infeasible task, creating a substantial barrier to real-world implementation. Alternatively, data-driven modeling and control overcomes this barrier by learning an approximate model of the system from measured data. The majority of data-driven modeling methods requires no knowledge of the underlying dynamics; some methods leverage partial knowledge of the dynamics to enhance accuracy of the model and computational efficiency. With advances in machine learning algorithms and methods to collect data, data-driven control has become an exciting and effective

paradigm for control theorists and engineers alike. However, there are challenges inherit to data-driven control.

1.1. Motivating Example: Naively Stabilizing the Data-Driven Model

Consider for example the ubiquitous control objective of stabilizing an unstable nonlinear system. This problem boils down to finding control input that, when applied to the system, moves the eigenvalues to a stable region of the complex plane. For example, for a continuous-time system, a stabilizing controller needs to push the eigenvalues into left half plane, i.e., $\Re(\lambda) < 0$ for every eigenvalue λ of the closed-loop dynamics. Now, suppose that we don't know and cannot practically derive from first principles the model of our system. All we know is that the system is unstable. Taking the data-driven approach, we measure data from the system and learn an approximate model of the system. There will necessarily be error between the data-driven model and the true system dynamics, and in particular, there will be a discrepancy between the eigenvalues of the data-driven model and the true system. Therefore, if we generate a feedback control law that stabilizes the data-driven model, it need not be the case that this control law stabilizes the true system. Figure 1.1 below illustrates this phenomenon with a cartoon.

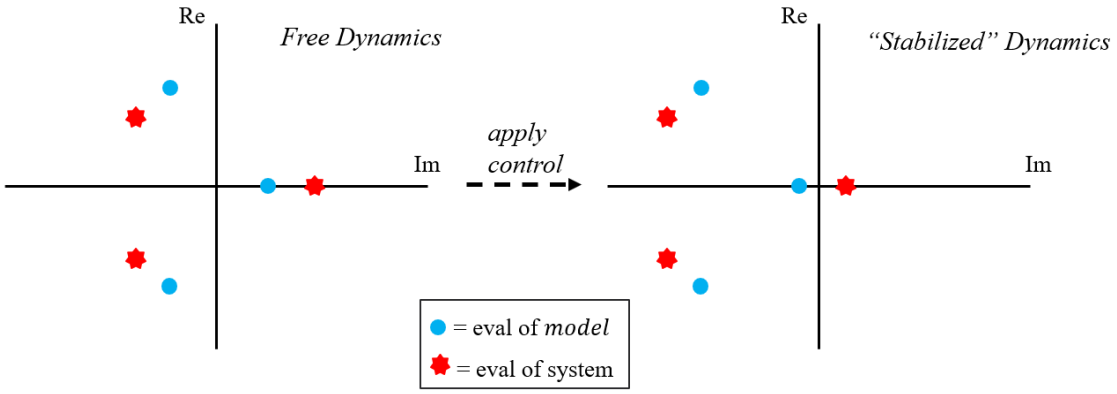

Figure 1.1. Deriving a stabilizing control law for the data-driven model and applying it to the true system won't necessarily lead to a stable closed-loop system! No matter how much data and how reliable the machine learning algorithm is, this data-driven modeling error will be present, and thus there is no guarantee that the true system will be stabilized.

1.2. Contributions and Organization

In order to guarantee that the eigenvalues of the closed-loop system lie in a stable region, one must be able to bound the error between the eigenvalues of the data-driven model and the spectrum of the underlying system. Guarantees of this sort in the context of data-driven methods are tough to come by, but we will see that the algorithm ResDMD, recently developed by Colbrook *et al.* in [**2, 9**], does exactly this for autonomous dynamical systems. In order t In this thesis, we extend this work for nonlinear control systems, specifically control-affine systems. We call this extension *ResDMDc*, ResDMD with control. ResDMDc gives us error bounds on the eigenvalues of the free dynamics of the data-driven model with respect to the free dynamics of the underlying nonlinear system. These error bounds define what we call an "error ball" about each eigenvalue,

where the ball has a radius equal to the error bound and is centered at the eigenvalue. Due to ResDMDc, we know that there exists at least one eigenvalue of the underlying nonlinear system in each of these error balls. The goal of our robust stabilizing feedback controller design can now be simple put: instead of synthesizing a control law that pushes the eigenvalues of the data-driven model into a stable region of the complex plane, we synthesize a control law that pushes the *error balls* into the stable region. This guarantees that the eigenvalues of the underlying system inside the error balls now completely lie inside the stable region. Pushing the error balls into a stable region is what gives us the "robustness" in the title of this thesis. This is to be interpreted as robustness with respect to data-driven model error, as opposed to the more common use of the word "robust" in the context of robust control theory [**10**].

Remark 1 (ResDMD Assumptions)**.** *There are assumptions on the amount of data collected and the sampling protocol for collecting data that underpins the error bounds given by ResDMD(c). In particular, there are two assumptions that make these error bounds exact, as opposed to approximate error bounds. One is that there is an infinite amount of training data, which is never satisfied. The other is an assumption on the sampling protocol, which is impossible to check. We will discuss both of these assumptions in more detail later. We state them here from the onset to qualify our claims and caution blind trust in ResDMD(c), especially if one wanted to apply ResDMD(c) to a safety-critical system, such as an autonomous vehicle.*

The second ingredient that plays an important role in this thesis is minimal-norm stabilizing feedback optimization. Recent work in applied linear algebra and optimization by

Gillis *et al.* has produced new characterizations of stability that lead to computationally efficient optimization algorithms for finding the nearest stable LTI system to an unstable LTI system, both in the context of discrete-time stability [**11**] and continuous-time stability [**4**]. To be clear, an autonomous LTI system is given by

$$\dot{x} = Ax \tag{1.1}$$

$$x_{k+1} = Ax_k, \quad k \in \mathbb{Z}^{\geq 0}, \tag{1.2}$$

where (1.1) is in continuous time and (1.2) is in discrete time. An LTI system is said to be stable if all of the eigenvalues of A lie in a stable region of the complex plane, where this stable region is the left half plane for continuous-time systems and the unit disk for discrete-time systems. Thus, finding the nearest stable LTI system means solving a matrix nearness optimization problem to find the nearest matrix A' with all stable eigenvalues. This optimization problem is ill-posed due to the fact that the feasible set, i.e., the set of all stable matrices in either continuous or discrete time, is highly nonconvex and neither open nor closed. Gillis *et al.* address this by establishing a new parameterization of stable matrices that makes the feasible set convex and closed, which enables standard constrained gradient descent algorithms to solve the problem. In [**12**] this stability parameterization and optimization framework is extended to the problem of finding the nearest neighbor $\Omega-$stable matrix, meaning that the eigenvalues of the new matrix lie inside a region of the complex plane $\Omega \subset \mathbb{C}$, which could be a conic sector, vertical strip, disk, and any intersection of the three. Finally, in [**3**] Gillis *et al.* consider LTI systems with control input and use the stability opimization framework described above to compute minimal-norm stabilizing feedback gains. However, in this work the

authors only considered continuous-time LTI control systems and the continuous-time notion of stability. In this thesis, we extend this continuous-time minimal-norm feedback stabilization to minimal-norm $\mathcal{D}(\delta)$-stabilizing feedback, where $\mathcal{D}(0,\delta) \subset \mathbb{C}$ is a disk of radius δ centered at the origin. That is, when control is applied, all of the eigenvalues of the closed-loop system lie inside $\mathcal{D}(0,\delta)$. Note that $\mathcal{D}(0,1)$-stabilizing feedback is equivalent to discrete-time stabilizing feedback. This generalization becomes readily useful for our overall Koopman-based controller design as we can incorporate the error bound associated with each eigenvalue of learned model, as given by ResDMDc. Ultimately, this leads to our stabilizing feedback controller that is able to place the eigenvalues' error balls into a stable region of the complex plane with minimal control effort.

We explicitly state the contributions of this thesis, as discussed in the two preceding paragraphs:

(C1) We generalize the ResDMD framework to include nonlinear systems with control input, which we call ResDMDc.

(C2) We extend the optimization algorithm in [**3**] to be able to find minimal-norm $\mathcal{D}(0,\delta)$-stabilizing feedback gains.

(C3) We combine (C1) and (C2) to derive the final data-driven stabilization algorithm stabilizes nonlinear systems with minimal control effort while accounting for model error.

The rest of the thesis is as follows. In Chapter 2 we introduce the Koopman operator and data-driven methods for approximating the Koopman operator, which ultimately yield an LTI control system that approximates the underlying nonlinear system. In Chapter 3 we review the ResDMD algorithm for autonomous systems and introduce

our ResDMDc formulation for controlled systems. In Chapter 4 we present the optimization algorithm for minimal-norm stabilizing feedback for LTI systems, both in continuous and discrete time, and show how we generalized this to minimal-norm $\mathcal{D}(0, \delta)$-stabilizing feedback. In Chapter 5 we put these two ingredients together to synthesize a data-driven feedback controller that is able to push the eigenvalues *and* their error balls into a the unit disk. We show our controller in action in a few basic example systems in Chapter 6 (hopefully). Finally, we describe future directions for this research, both in terms of theory and application.

CHAPTER 2

Data-Driven Control Using the Koopman Operator

With the confluence of machine learning advances, computational power, and an abundance of data and data collection techniques, data-driven methods have gained considerable research interest and momentum in the dynamical systems and control community, as well as just about every other scientific field. In contrast to model-based control, which requires the often infeasible task of developing high-fidelity, physically-accurate models of complicated dynamics from first principles, data-driven control views complicated dynamics as a black box, where the only important information comes from measured system data. Using this data, one is able to either develop a data-driven model that can be used in traditional model-based control, e.g. proportional-integral-derivative (PID) control or linear quadratic (LQR/LQG) control. It is also possible to not learn any model and by using neural network architectures develop control input solely from the measured data. Given the wide array of machine learning, data collection, and control methodologies, the field of data-driven has become truly immense and diverse. We will not be able to adequately survey all of the different methods, but we do want to point out that applied Koopman operator methods only represents one of many fruitful methodologies for data-driven control. For a comprehensive survey of data-driven control, refer to the review article [**13**] and the textbook [**14**].

2.1. Literature Review of Koopman-Based Control

Over the past two decades, applied Koopman operator methods have gained immense popularity as a new methodology for globally linearizing nonlinear dynamics by lifting to an infinite dimensional space where the Koopman operator propagates functions of the state linearly. To make this infinite-dimensional object computationally manageable, data-driven techniques for deriving finite-dimensional approximate Koopman dynamics such as dynamic mode decomposition (DMD) and its many variants have enabled the Koopman modeling method to be applied across a wide range of disciplines and problems therein. In particular, many data-driven control approaches have been develop using applied Koopman operator methods. One common approach is to learn a bilinear Koopman model and then search for a control lyapunov function (CLF) [**15**], [**16**], [**17**]. Another approach is to linear a linear Koopman model and use it as the model for model predictive control [**18, 19**]. While most of these papers only exhibit their controllers working on generic nonlinear systems in simulation, Koopman-based control has also proven to be quite effective for controlling a variety of physical platforms in real-world experiments, such as trajectory tracking for the end effector of a robotic arm [**20**], trajectory tracking for a robotic fish [**21**], control of a wheeled robot [**22**], and control of soft robotics [**23, 24, 25**], which is particularly impressive given the highly nonlinear and contact compliant nature of the soft material.

There are still many challenges and downsides to Koopman-based methods in modeling and control. One of the most pressing challenges is that, depending on the chosen observable functions and the measured data, the learned Koopman dynamics may be fundamentally different from the true nonlinear dynamics. It has been well documented

that DMD methods may learn spurious eigenvalues, which are eigenvalues of the Koopman model that have no physical relevance to the underlying system. A particularly worrisome example of this phenomenon is that the learned Koopman dynamics may be unstable even though the nonlinear dynamics are stable [**26, 27**]. This challenge is especially problematic in the context of control theory, as the goal of synthesizing control is to achieve with high confidence, ideally a theoretical guarantee, that the system behaves as desired when the control is applied. It is also worth noting that in the vast majority of Koopman-based control papers, this approximation error is not taken into account. In this thesis, we do take into account approximation error and use it to synthesize a more robust data-driven stabilizing controller.

While most Koopman-based methods neglect accounting (or caring) for the approximation error of their Koopman model, recent work has made great progress towards addressing this issue. In [**28**] a clever choice of observable functions as increasing order time derivatives of the state lead to a global error bound of the approximate Koopman operator via the Taylor remainder theorem. In [**29**] probabilistic bounds on the accuracy of the approximate Koopman operator are established in terms of the number of training data points, and this error bound is able to be applied broadly to both ordinary and stochastic differential equations. In the context of Koopman-based control, the very recent work by Strässer *et al.* [**17**] uses robust control theory to account for the approximation error of the Koopman operator and synthesize a robustly stabilizing CLF. Further, they are able to explicitly compute the region of attraction of the closed-loop system; that is, which initial state positions will lead to a trajectory that converges to the equilibrium. However, this error is never explicitly computed; it is only treated as an arbitrary value,

and so it is impossible to prescribe as the user. Moreover, in Assumption 2 they assume that the subspace formed by the span of the chosen observables is invariant, which is almost never satisfied in practice [**30**].

In this work, we do not make any assumption on the invariance of the subspace formed by our observables. Further, we explicitly compute an error bound on the distance of each eigenvalue of the Koopman model to the spectrum of the true nonlinear system, which we will do by generalizing ResDMD, discussed in detail in Chapter 3. This error bound is different than error bounds from other works because we our looking only at the error associated with the eigenvalues of the learned Koopman model, as opposed to the error of the Koopman model itself with respect to the true Koopman operator.

Ultimately, there are a myriad of approaches, both theoretically and computationally, for data-driven modeling and control using the Koopman operator. For more information, refer to [**30**] for a concise and fairly comprehensive survey of Koopman methods for data-driven modeling and control.

2.2. The Koopman Operator

In [**31**] Bernard Koopman showed that for Hamiltonian dynamical systems, functions of the state evolve according to a linear operator, now called the *Koopman operator*. The power of the Koopman operator is that it globally linearizes a nonlinear dynamical system by embedding it in a function space, typically $L^2(\Omega, \omega)$, where the Koopman operator acts linearly on these functions of the state; the challenge is that the Koopman operator is typically infinite-dimensional.

Remark 2. *Not every nonlinear system can be globally linearized by a Koopman embedding. For the precise mathematical conditions under which a nonlinear system can be globally linearized by a Koopman embedding, refer to* [**32**].

For the mathematical definition of the Koopman operator, consider an autonomous discrete-time nonlinear dynamical system

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k), \quad k \in \mathbb{Z}^{\geq 0}, \tag{2.1}$$

where $\mathbf{x}_k \in \Omega \subset \mathbb{R}^d$ is the state vector, Ω is a measurable state space with measure ω, and $F : \Omega \to \Omega$ is a nonlinear function governing the dynamics. Given a complex-valued function of the state $g : \Omega \to \mathbb{C}$ that is an element of $L^2(\Omega, \omega)$, the Koopman operator $\mathcal{K}$ acts on g by the composition

$$\mathcal{K}g = g \circ F. \tag{2.2}$$

Note that in the Koopman literature this function g is often called an *observable*, and we will use this terminology throughout. With the composition action of the Koopman operator, we have that

$$[\mathcal{K}g](\mathbf{x}_k) = g(F(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{2.3}$$

For a schematic illustration of the Koopman lifting and action, refer to Figure 2.1 below.

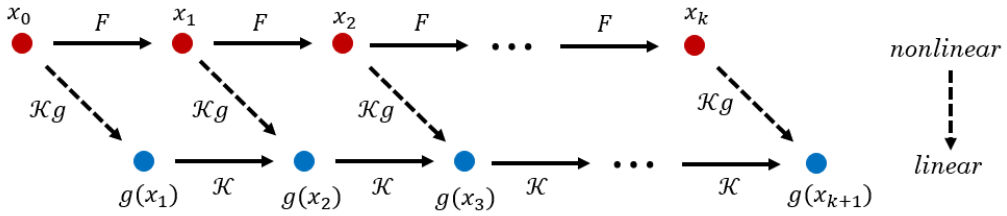

Figure 2.1. Schematic flow of nonlinear state-space dynamics and linear Koopman dynamics.

Thus it can be seen that the Koopman operator evolves functions of the state one step forward in time. It is not hard to show that the Koopman action is linear; that is,

$$[\mathcal{K}(\alpha g + \beta f)](\mathbf{x}) = \alpha[\mathcal{K}g](\mathbf{x}) + \beta[\mathcal{K}f](\mathbf{x}), \tag{2.4}$$

for arbitrary $f, g \in L^2(\Omega, \omega)$ and $\alpha, \beta \in \mathbb{R}$. Hence, the Koopman operator evolves functions of the state *linearly* forward in time.

The takeaway message here is that the behavior of the dynamics of (2.1) is encoded in the dynamics of (2.3); in particular, the spectral properties of the underlying system are fully captured by the spectral properties of $\mathcal{K}$. We can even think of (2.3) as defining a new infinite-dimensional discrete-time dynamical system, where the "state variable" is the function of the state $g(\mathbf{x})$ and the dynamics are governed by $\mathcal{K}$. We refer to this dynamical system as the "infinite-dimensional Koopman dynamics", which again describes the same fundamental behavior as the system (2.1). Note that in later sections, we will refer to the "finite-dimensional approximate Koopman dynamics" as just the "Koopman dynamics" or "Koopman model".

Remark 3. *The Koopman operator can also be easily defined for continuous-time dynamics, see for example* [**30**]*. The reason we introduced it with discrete-time dynamics is because it is more natural for the upcoming discrete data-driven process for approximating the Koopman operator. Whether or not the underlying system is continuous or discrete, we will measure discrete snapshots of data, which will ultimately lead to approximate Koopman dynamics in discrete time. Additionally, in engineering practice all continuous-time systems are first discretized.*

While the Koopman operator and the associated infinite-dimensional Koopman dynamics unleash the power of linear systems theory to be used on nonlinear systems, a clear roadblock is that the Koopman operator is this infinite-dimensionality, which makes it impossible to use the Koopman operator for any computational task. The key to unlocking the full potential of the Koopman operator for applied computational science is due to the data-driven learning algorithms that accurately capture the most important spectral information of the Koopman operator in a finite-dimensional matrix. The first and most popular of these algorithms are called dynamic mode decomposition (DMD) [**33**] and the more general extended dynamic mode decomposition (EDMD) [**34**]. In the next section we introduce the EDMD framework.

2.3. Data-Driven Approximations of the Koopman operator

We now set up the problem of learning the dynamics of (2.1) when F is unknown and we have access to measured (experimental or simulated) data. Assume that we have M snapshots of data $\{\mathbf{x}_m, \mathbf{y}_m\}_{m=1}^{M}$, where $\mathbf{y}_m = F(\mathbf{x}_m)$. Now, we choose a dictionary of N observable functions $\{\psi_1(\mathbf{x}), \psi_2(\mathbf{x}), \ldots, \psi_N(\mathbf{x})\}$. Ultimately, EDMD uses the snapshot

data to generate a matrix $\mathbb{K} \in \mathbb{C}^{N\times N}$ that is the approximation of $\mathcal{K}$ on the finite-dimensional subspace $V_N = \text{span}\{\psi_1(\mathbf{x}), \psi_2(\mathbf{x}), \ldots, \psi_N(\mathbf{x})\}$. That is, with the EDMD algorithm, we will get $\mathbb{K}$ such that

$$[\mathcal{K}\psi_j(\mathbf{x})] = \psi_j(F(\mathbf{x})) \approx \sum_{i=1}^{N} [\mathbb{K}]_{ij}\psi_i(\mathbf{x}). \tag{2.5}$$

The choice of observables is a critical step in EDMD, and it is completely up to the user to find a dictionary that is rich enough to capture the underlying dynamics. See the following remark.

Given the user-chosen dictionary, we assemble the observables into a vector-valued observable vector

$$\Psi(\mathbf{x}) = \begin{bmatrix} \psi_1(\mathbf{x}) & \psi_2(\mathbf{x}) & \cdots & \psi_N(\mathbf{x}) \end{bmatrix} \in \mathbb{C}^{1\times N}. \tag{2.6}$$

By definition, for every function $g(\mathbf{x}) \in V_N$, we can write

$$g(\mathbf{x}) = \sum_{j=1}^{N} \psi_j(\mathbf{x})c_j = \Psi(\mathbf{x})\mathbf{c} \tag{2.7}$$

for some vector of scalar coefficients $\mathbf{c} = \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix}^\top \in \mathbb{C}^N$. Now, applying $\mathcal{K}$ to (2.7) yields

$$[\mathcal{K}g](\mathbf{x}) = \Psi(F(\mathbf{x}))\mathbf{c} = \Psi(\mathbf{x})(\mathbb{K}\mathbf{c}) + r(\mathbf{x}), \tag{2.8}$$

where $r(\mathbf{x})$ is a residual term since V_N is typically not an invariant subspace of $\mathcal{K}$ [**34**], and so there doesn't exist a matrix $\mathbb{K}$ that makes $r(\mathbf{x})$ zero. The EDMD Koopman operator

is then the matrix that minimizes the residual

$$\mathbb{K} = \underset{K \in \mathbb{C}^{N \times N}}{\arg\min} \frac{1}{2} \sum_{m=1}^{M} |r(\mathbf{x}_m)|^2 = \underset{K \in \mathbb{C}^{N \times N}}{\arg\min} \frac{1}{2} \sum_{m=1}^{M} |\Psi(y_m) - \Psi(\mathbf{x}_m)K|^2. \tag{2.9}$$

This is a least-squares problem which has a global minimizer (or a family of equivalent minimizers). A solution to (2.9) is given by

$$\mathbb{K} = G^{\dagger} A, \tag{2.10}$$

where

$$G = \frac{1}{M} \sum_{m=1}^{M} \Psi(\mathbf{x}_m)^* \Psi(\mathbf{x}_m), \tag{2.11}$$

$$A = \frac{1}{M} \sum_{m=1}^{M} \Psi(\mathbf{x}_m)^* \Psi(y_m) \tag{2.12}$$

and where † denotes the pseudoinverse. Note that (2.10) is always unique. Further, (2.10) gives us the N-dimensional linear Koopman dynamics

$$\Psi(\mathbf{x}_{k+1}) = \Psi(\mathbf{x}_k)\mathbb{K}. \tag{2.13}$$

Since working with states that are column vectors is more natural in the context of linear systems theory, an equivalent representation of the Koopman dynamics is

$$\mathbf{z}_{k+1} = \mathbb{K}^{\top} \mathbf{z}_k, \tag{2.14}$$

where $\mathbf{z}_k = \Psi(\mathbf{x}_k)^{\top}$ is an $N-$dimensional column vector.

2.3.1. Convergence of EDMD in the Large Data Limit

We present a theoretical result regarding the convergence of $\mathcal{K}$ in the large data limit, i.e., as $M \to \infty$. This result, particularly its underpinning assumption on data collection, will be important for understanding and qualifying the claims of ResDMD. Below is an abbreviated version of the assumption and theorem, where only notation has been changed to reflect the notation in this thesis.

Assumption 1 (Assumption 1 in [**35**])**.** *The data $\{\mathbf{x}_m, \mathbf{y}_m\}_{m=1}^M$ is sampled independently from a probability distribution μ that is not supported on a zero level set of a linear combination of the observables. That is, given observables $\psi_1, \psi_2, \ldots, \psi_N$, we assume that*

$$\mu\left\{\mathbf{x} \in \Omega \mid \sum_{j=1}^{N} c_j \psi(\mathbf{x}) = 0\right\} = 0, \tag{2.15}$$

where $c_j \neq 0$ for all $j = 1, \ldots, N$.

Theorem 1 (Theorem 2 in [**35**])**.** *Recall that V_N is the subspace spanned by the dictionary of observables $\{\psi_1, \psi_2, \ldots, \psi_N\}$. Then, if Assumption 1 holds, it follows with probability one that*

$$\lim_{M\to\infty} \|\mathbb{K} - \mathcal{P}_{V_N}\mathcal{K}\mathcal{P}_{V_N}^*\| = 0, \tag{2.16}$$

where $\mathcal{P}_{V_N}$ is the orthogonal projection onto the subspace V_N and $\|\cdot\|$ is any operator norm. Further, we have

$$\lim_{M\to\infty} dist(\sigma(\mathbb{K}), \sigma(\mathcal{P}_{V_N}\mathcal{K}\mathcal{P}_{V_N}^*)) = 0, \tag{2.17}$$

where $\sigma(\cdot) \subset \mathbb{C}$ denotes the spectrum of an operator and dist$(\cdot,\cdot)$ is the Hausdorff metric on subsets of $\mathbb{C}$.

The proof of this theorem is a result of the law of large numbers. See [**35**] for the complete proof. The reason we present this theorem here is because there will be a similar "large data limit" convergence result for ResDMD in Chapter 3 using the same assumptions. Note that for practical purposes, this convergence in the large data limit doesn't really give us much. First, it is obviously never the case that we have an infinite amount of training data. But also, Assumption 1 is slightly concerning. The interpretation is that we are assuming the measured data is in a sense decorrelated from the choice of basis functions, which is not the case, for example, when sampling data from a simulated trajectory. Further, it is practically impossible to check if Assumption 1 is satisfied or not. We will return to these qualms when we discuss ResDMD in the large data limit.

We conclude this subsection with a remark on the connection between DMD and EDMD.

Remark 4. *The standard DMD framework is very similar to that of EDMD; in particular, DMD is a special case of EDMD* [**34**]*. In DMD, the observables are not chosen, rather, they are defined to be the state variables, i.e., $\psi_i(\mathbf{x}) = \mathbf{x}_i$ for all $i = 1, \ldots, d$. In EDMD, as we just saw, we are able to get a richer set of observables by allowing for user-chosen nonlinear functions of the state. Note that it is still common to include the state in the dictionary of observables in addition to the nonlinear functions in EDMD, which, among other things, makes state reconstruction trivial. In our control design, we will include the state in our dictionary of observables, see Assumption 3.*

2.3.2. Spectral Properties of $\mathbb{K}$

Since the dynamic properties of a system are encoded in its spectrum, the Koopman operators spectrum is of great importance to our understanding of the Koopman dynamics and the underlying nonlinear dynamics. In particular, we will see how to compute the Koopman eigenvalues μ_j, eigenfunctions $\phi_j(\mathbf{x})$, and modes $\mathbf{v}_j$ for $j = 1, 2, \ldots, N$. Let $(\mu_j, \phi_j(\mathbf{x}))$ be an eigenvalue-eigenfunction pair of $\mathcal{K}$. By definition, it follows that

$$[\mathcal{K}\phi_j](\mathbf{x}) = \mu\phi_j(\mathbf{x}). \tag{2.18}$$

Now, there exists a modal decomposition of every observable in the subspace V_N. That is, given some observable $g(\mathbf{x}) \in V_N$, we have that $g(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{v}_j\phi_j(\mathbf{x})$, where $\mathbf{v}$ is the Koopman mode. Thus, we can then represent the time of evolution of (2.3) by the following decomposition

$$g(\mathbf{x}_{k+1}) = g(F(\mathbf{x}_k)) = [\mathcal{K}g](\mathbf{x}_k) = \sum_{j=1}^{N} \mathbf{v}_j[\mathcal{K}\phi_j](\mathbf{x}_k) = \sum_{j=1}^{N} \mu_j\mathbf{v}_j\phi_j(\mathbf{x}_k). \tag{2.19}$$

We are able to approximate the Koopman eigenvalues, eigenfunctions, and modes from the spectral analysis of $\mathbb{K}$, the EDMD finite-dimensional approximation of the Koopman operator. First, the eigenvalues of $\mathbb{K}$ directly approximate the eigenvalues of $\mathcal{K}$, that is, the jth eigenvalue λ_j of $\mathbb{K}$ is the approximation of the jth eigenvalue μ_j of $\mathcal{K}$. The approximation of the eigenfunction $\phi_i(\mathbf{x})$ is given by $\phi_i(\mathbf{x}) = \sum_{j=1}^{N} [\mathbf{g}_i]_j\psi_j(\mathbf{x}) = \Psi(\mathbf{x})\mathbf{g}_i$, where $\mathbf{g}_i$ is the right eigenvector of $\mathbb{K}$ associated with the eigenvalue λ_i. There's plenty more to be said about the spectral properties of $\mathbb{K}$, but this is sufficient for what we will need to develop ResDMD(c). For more details, see [**34**].

2.4. EDMD with Control (EDMDc)

We now turn to the problem of modeling a system with control inputs, where we are no longer just interested in predicting the system in the future, we want to control its future behavior through a control input $\mathbf{u}$. Consider the unknown control-affine dynamics

$$\mathbf{x}_{k+1} = f_0(\mathbf{x}_k) + \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_k)[\mathbf{u}_i]_k. \tag{2.20}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ is the state, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $f_0(\mathbf{x})$ is the drift vector field which describes the free dynamics of the system, and $f_i(\mathbf{x})$ for i in some index set $\mathcal{I}$ are the control vector fields. Extending Koopman-based modeling to control systems of the form (2.20) was first done in [**36**] with dynamic mode decomposition with control (DMDc). Generalizing the EDMD methodology for control systems was established in [**1**], which we will call EDMDc even though this is not an acronym that appears in the paper. We will also use notation from EDMDc presented in [**20**]. Note that EDMDc algorithm in these works yield Koopman dynamics in the form of an LTI control system. Learning LTI Koopman dynamics, as opposed to bilinear Koopman dynamics, is discussed in Remark 5.

In order to incorporate control into any sort of Koopman-based modeling algorithm, there are two critical additions:

(1) The dictionary of observables $\Psi(\mathbf{x})$ must now be expanded to include the control input $\mathbf{u}$ and cross-terms in $\mathbf{u}$ and observables of $\mathbf{x}$. For example, a new observable could be $\psi_{new}(\mathbf{x}, \mathbf{u}) = x_1^2 u_2$. This expanded dictionary is denoted $\Psi(\mathbf{x}, \mathbf{u})$.

(2) Since we now have a observables dependent on $\mathbf{u}$, control input measurements must be collected during the training process.

Now, we present EDMDc. In addition to collecting measurements of the system $\{\mathbf{x}_m, \mathbf{y}_m\}_{m=1}^{M}$, we also collect measurements of the control input at each time step including the control at $M+1$, $\{\mathbf{u}_m\}_{m=1}^{M+1}$. Then we choose observables of the control and state and construct a new observable vector

$$\Psi_{\mathbf{u}} = \begin{bmatrix} \psi_1(\mathbf{x}, \mathbf{u}) & \psi_2(\mathbf{x}, \mathbf{u}) & \cdots & \psi_{N_{\mathbf{u}}}(\mathbf{x}, \mathbf{u}) \end{bmatrix} \in \mathbb{C}^{1 \times N_{\mathbf{u}}}. \tag{2.21}$$

Recall we have already chosen an observable vector of functions only dependent on the state

$$\Psi_{\mathbf{x}} = \begin{bmatrix} \psi_1(\mathbf{x}) & \psi_2(\mathbf{x}) & \cdots & \psi_{N_{\mathbf{x}}}(\mathbf{x}) \end{bmatrix} \in \mathbb{C}^{1 \times N_{\mathbf{x}}}. \tag{2.22}$$

We concatenate $\Psi_{\mathbf{x}}$ and $\Psi_{\mathbf{u}}$ to form our total dictionary of observables

$$\Psi(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \Psi_{\mathbf{x}} & | & \Psi_{\mathbf{u}} \end{bmatrix} \in \mathbb{C}^{1 \times N}, \tag{2.23}$$

where $N = N_{\mathbf{x}} + N_{\mathbf{u}}$. We now perform EDMD using the observables (2.23) and measured data, which yields the approximate Koopman operator

$$\mathbb{K} = G^{\dagger} A, \tag{2.24}$$

where now

$$G = \frac{1}{M}\sum_{m=1}^{M} \Psi(\mathbf{x}_m, \mathbf{u}_m)^* \Psi(\mathbf{x}_m, \mathbf{u}_m), \tag{2.25}$$

$$A = \frac{1}{M}\sum_{m=1}^{M} \Psi(\mathbf{x}_m, \mathbf{u}_m)^* \Psi(\mathbf{y}_m, \mathbf{u}_{m+1}). \tag{2.26}$$

The associated Koopman dynamics are

$$\Psi(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})^\top = \mathbb{K}^\top \Psi(\mathbf{x}_k, \mathbf{u}_k)^\top \tag{2.27}$$

where $\Psi(\mathbf{x}_k, \mathbf{u}_k)^\top$ is our Koopman state in column vector form. Observe that (2.27) does not disentangle the evolution of the state and the influence of the control input since both $\mathbf{x}$ and $\mathbf{u}$ are present in the Koopman state vector. In order to separate the dynamics of the observables which are only dependent on $\mathbf{x}$ and the observables which are also dependent on $\mathbf{u}$, we decompose the total matrix $\mathbb{K}$ into submatrices

$$\mathbb{K}^\top = \begin{bmatrix} \mathbb{K}_{\mathbf{x}} & \mathbb{K}_{\mathbf{u}} \\ \cdot & \cdot \end{bmatrix}, \tag{2.28}$$

where $\mathbb{K}_{\mathbf{x}} \in \mathbb{C}^{N_{\mathbf{x}} \times N_{\mathbf{x}}}$ and $\mathbb{K}_{\mathbf{u}} \in \mathbb{C}^{N_{\mathbf{x}} \times N_{\mathbf{u}}}$. Note that the $(\cdot)$ terms in (2.28) evolve the observations on the control input, which is unambiguous as this evolution is determined by the controller and not something we need to predict. Using the decomposition in (2.28), the Koopman dynamics can now be rewritten as the following LTI system

$$\mathbf{z}_{k+1} = \mathbb{K}_{\mathbf{x}} \mathbf{z}_k + \mathbb{K}_{\mathbf{u}} \mathbf{v}_k, \tag{2.29}$$

where the Koopman state vector is now $\mathbf{z}_k = \Psi_{\mathbf{x}}(\mathbf{x}_k)^\top \in \mathbb{C}^{N_{\mathbf{x}} \times 1}$, i.e., the state-dependent observables, and $\mathbf{v}_k = \Psi_{\mathbf{u}}(\mathbf{x}_k, \mathbf{u}_k)^\top \in \mathbb{C}^{N_{\mathbf{u}} \times 1}$ which can be thought of as the Koopman control input. In practice, if one makes sure to structure their observables as (2.23), i.e., with $N_{\mathbf{x}}$ state-dependent observables first followed by $N_{\mathbf{u}}$ state- and control-dependent observables, one can always compute the decomposition (2.28) and generate the LTI system (2.29).

2.4.1. A remark on learning an LTI representation of the Koopman dynamics

Remark 5. *We make assumption that the dynamics in the Koopman space (i.e. the space of observables) has the form of an LTI system. As a consequence, this means that the Koopman dynamics must obey the principle of linear superposition, which can be a restrictive assumption. In order to learn a more general Koopman system, many works choose rather to represent the Koopman dynamics as a bilinear model* [**37**]. *A case for learning an LTI system is made in* [**1**].

2.5. Challenges with EDMD

While EDMD(c) has proven to be an effective data-driven modeling methodology for a wide variety of dynamical systems, as alluded to in the introduction, there are well-documented issues with EDMD. been effective for a myriad of systems and engineering We now discuss some well-known issues with the EDMD algorithm. That is, when EDMD approximation to the Koopman operator and Koopman dynamics does not match the true nonlinear system.

(1) Spectral Pollution: The model generated by EDMD can have eigenvalues that are not physically related to the eigenvalues of the underlying system. These eigenvalues are often called *spurious*, and can lead to the behavior of the Koopman model being fundamentally different from the behavior of the underlying system. For example, it has been observed the EDMD can generate a Koopman model that is unstable even when the underlying system is stable [**26**]. This happens because there is a spurious eigenvalue of the Koopman model in an unstable region of the complex plane.

(2) Spectral Invisibility: Since the spectrum of a nonlinear system is generally comprised of a countably infinite point spectrum and an uncountably infinite continuous spectrum, a finite-dimensional approximation of the system, which only has finite point spectrum, will always fail to capture all of the spectral information. This is especially problematic for stability analysis using the approximate dynamics because one can never fully guarantee that the approximate dynamics capture all of the unstable eigenvalues.

(3) Lack of verification: As discussed in the introduction, there is very little literature on verifying the accuracy Further, the few methods that do exist give error bounds on the overall discrepancy between $\mathbb{K}$ and $\mathcal{K}$, not error bounds on the distance between an eigenvalue of $\mathbb{K}$ to the spectrum of $\mathcal{K}$. Therefore, one can imagine a situation where there are two spurious eigenvalues "offset" and result in a overall $\mathbb{K}$ that is close to $\mathcal{K}$, but the behavior of $\mathbb{K}$ can still be drastically off.

In [**2, 9**] Colbrook *et al.* develop an algorithm, residual dynamic mode decomposition (ResDMD), which fully resolves (3), gives a criterion for removing spurious eigenvalues to address (1), and enables better visibility of the spectrum to address (2).

CHAPTER 3

Residual Dynamic Mode Decomposition (ResDMD)

Before describing the mathematical formulation and results of ResDMD, we cut to the chase and state the powerful result that it yields. The ResDMD algorithm, at no extra computational cost nor further assumptions than is already present for EDMD, generates error bounds on the eigenvalues of $\mathbb{K}$ with respect to the true spectrum of $\mathcal{K}$. That is, for each eigenvalue λ_i, $i = 1, \ldots, N$ of $\mathbb{K}$, ResDMD computes an error bound r_i such that an element of the spectrum of $\mathcal{K}$ is guaranteed to lie within the ball $B(\lambda_i, r_i) = \{z \in \mathbb{C} \mid \|\lambda - z\| < r\}$, i.e., all points in $\mathbb{C}$ that are within a magnitude of r_i from λ_i. Returning to our cartoon of the motivating example 1.1, ResDMD allows us to draw error balls about each eigenvalue of the data-driven model, as shown below.

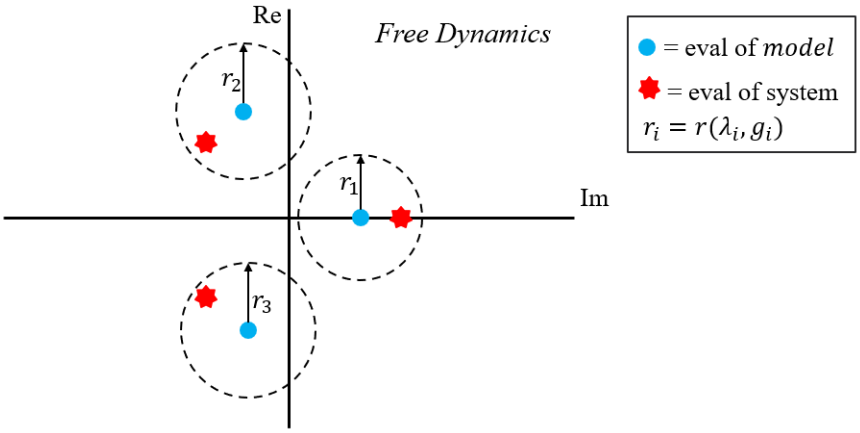

Figure 3.1. ResDMD guarantees that an eigenvalue of $\mathcal{K}$ lies in each of the error balls drawn.

Now we describe how ResDMD produces these error bounds.

3.1. Original Formulation

First, let λ and $\mathbf{g}$ be an eigenvalue-eigenvector pair of $\mathbb{K}$. The resulting approximate eigenfunction is $\phi(\mathbf{x}) = \sum_{j=1}^{N} g_j \psi_j(\mathbf{x}) = \Psi(\mathbf{x})\mathbf{g}$. If $(\lambda, \phi(\mathbf{x}))$ was a true eigenvalue-eigenfunction pair of $\mathcal{K}$, it would hold that

$$[\mathcal{K}\phi](\mathbf{x}) = \lambda\phi(\mathbf{x}). \tag{3.1}$$

Measuring the error or deviation of the pair $(\lambda, \phi(\mathbf{x}))$ from satisfying (3.1) can be done with the relative residual, which is given by

$$\begin{aligned} \mathrm{res}_{exact}(\lambda, \phi(\mathbf{x}))^2 &= \frac{\|[\mathcal{K}\phi](\mathbf{x}) - \lambda\phi(\mathbf{x})\|^2_{L^2(\Omega,\omega)}}{\|\phi(\mathbf{x})\|^2_{L^2(\Omega,\omega)}} \\ &= \frac{\int_\Omega |[\mathcal{K}\phi](\mathbf{x}) - \lambda\phi(\mathbf{x})|^2 d\omega(\mathbf{x})}{\int_\Omega |\phi(\mathbf{x})|^2 d\omega(\mathbf{x})} \\ &= \frac{\langle \mathcal{K}\phi, \mathcal{K}\phi\rangle - \lambda\langle\phi, \mathcal{K}\phi\rangle - \overline{\lambda}\langle\mathcal{K}\phi, \phi\rangle + |\lambda|^2\langle\phi,\phi\rangle}{\langle\phi,\phi\rangle}. \end{aligned} \tag{3.2}$$

We can't feasible compute (3.2) because of the infinite-dimensional inner products, but we can approximate them using the same quadrature numerical approximation technique as done in EDMD. We call this approximation $\mathrm{res}(\lambda, \phi(\mathbf{x}))$ and often refer to it as the residual, as opposed to the exact residual in 3.2. The formula for $\mathrm{res}(\lambda, \phi(\mathbf{x}))$ is given by

$$\mathrm{res}(\lambda, \phi(\mathbf{x})) = \sqrt{\frac{\mathbf{g}^* \left(L - \lambda A^* - \overline{\lambda}A + |\lambda|^2 G\right)\mathbf{g}}{\mathbf{g}^* G \mathbf{g}}}, \tag{3.3}$$

where we have the matrices

$$L = \frac{1}{M}\sum_{m=1}^{M} \Psi(y_m)^* \Psi(y_m), \tag{3.4}$$

$$G = \frac{1}{M}\sum_{m=1}^{M} \Psi(\mathbf{x}_m)^* \Psi(\mathbf{x}_m), \tag{3.5}$$

$$A = \frac{1}{M}\sum_{m=1}^{M} \Psi(\mathbf{x}_m)^* \Psi(y_m). \tag{3.6}$$

Note that both A and G have already been computed to find $\mathbb{K}$. Thus, the only additional computation required for (3.3) is the matrix L, which does not increase the overall computational expense and requires no additional sampled data. Note that $\text{res}(\lambda, \phi(\mathbf{x}))$ is a measure of how far the pair $(\lambda, \phi(\mathbf{x}))$ is from being a true eigenvalue-eigenfunction pair of $\mathcal{K}$. For our purposes, we want an error bound on the eigenvalue λ alone; that is, a measure of how far λ is to a true point of the spectrum of $\mathcal{K}$. We find this error bound through the following

$$\begin{aligned} \text{res}(\lambda) = \text{dist}(\lambda, \sigma(\mathcal{K})) &= \inf_f \frac{\|[\mathcal{K}f](\mathbf{x}) - \lambda f(\mathbf{x})\|_{L^2(\Omega,\omega)}}{\|f(\mathbf{x})\|_{L^2(\Omega,\omega)}} \\ &\leq \frac{\|[\mathcal{K}\phi](\mathbf{x}) - \lambda \phi(\mathbf{x})\|_{L^2(\Omega,\omega)}}{\|\phi(\mathbf{x})\|_{L^2(\Omega,\omega)}} \\ &\approx \text{res}(\lambda, \phi(\mathbf{x})). \end{aligned} \tag{3.7}$$

In the large data limit, the approximation in the final line of (3.7) becomes exact, as describe below.

3.1.1. ResDMD in the large data limit

Similar to the result that the EDMD Koopman operator $\mathbb{K}$ converges to the true Koopman operator projected onto V_N in the large data limit, the residual $\text{res}(\lambda, \phi(\mathbf{x}))$ converges to $\text{res}_{exact}(\lambda, \phi(\mathbf{x}))$. If Assumption 1 holds, we have that

$$\lim_{M\to\infty} \text{res}(\lambda, \phi(\mathbf{x})) = \text{res}_{exact}(\lambda, \phi(\mathbf{x})) = \frac{\|[\mathcal{K}\phi](\mathbf{x}) - \lambda\phi(\mathbf{x})\|_{L^2(\Omega,\omega)}}{\|\phi(\mathbf{x})\|_{L^2(\Omega,\omega)}}. \tag{3.8}$$

In addition to Assumption 1, Colbrook *et al.* make the assumption that the quadrature rule converges, which is also implicitly assumed for Thm 1. See Subsection 4.1.3 of [**2**] for more details.

We now, again, describe the caution that should be used when interpreting and using ResDMD. As discussed in Subsection 2.3.1, we will never have an infinite amount of data and cannot verify that any of our data is sampled in such a way that satisfies Assumption 1. Therefore, the final "≈" in (3.7) will never become a "=". That is, ResDMD will always only be able to produce *approximate* error bounds. Note, however, that the inequality in (3.7) is not tight, and hence even with the approximation in the last line, it is certainly possible that $\text{res}(\lambda) \leq \text{res}(\lambda, \phi(\mathbf{x}))$. Nevertheless, one should proceed with caution! For example, if one only collects a small amount of data using i.i.d. sampling, as opposed to information-maximizing sampling, such as ergodic data collection [**38**], we suggest that the error bounds ResDMD spits out should not be trusted. This is especially the case if one wishes to apply ResDMD to saftey-critical systems, such as autonomous driving, human-robot collaboration in close quarters, stability analysis of a nuclear reactor, and this list goes on.

We end this section with a remarks on how ResDMD can be used in various ways to design Koopman-based algorithms.

Remark 6. *There's a number of ways to use ResDMD in an overall Koopman-based modeling design. The most basic algorithm is a "clean-up" procedure removing spurious eigenvalues from the EDMD Koopman model. Since* $\mathbb{K}$ *is diagonalizable by construction, one can discard eigenvalue eigenvector pairs of* $\mathbb{K}$ *if their residual is too high (algorithm 1 of* [**9**]*). One can also use ResDMD as a tolerance for computing pseudospectra, which are elements within a prescribed tolerance of being true spectral elements (Algorithm 2 of* [**9**]*).*

In our control design, we do neither of these procedures. In fact, part of the utility of our controller is that it takes into account large residuals (i.e. eigenvalue error bounds) by design. That is, if there exist large residuals, say, for instance, due to a poor choice of observables or an insufficient amount training data, our controller will be more conservative because it will have to push the eigenvalues of the Koopman model further into the unit disk to fully get the error balls inside the unit disk. This point will be made more clear in Chapter 5.

3.2. ResDMD with Control (C1)

In its original formulation, ResDMD was defined for dynamical systems with no external control input. We extend ResDMD to nonlinear control-affine systems and show that one can still derive error bounds on the eigenvalues of the free dynamics, which is of importance for designing closed-loop feedback control to move these eigenvalues to desired areas of the complex plane. We call ResDMD with control *ResDMDc* (disclaimer: until

this work is published, "ResDMDc" is free for the taking! We have no rights to such a catchy acronym as of now).

Consider again the control-affine dynamics (2.20). We now assume that the drift and control vector fields are unknown. Let $\mathcal{K}_x$ be the Koopman operator associated with $\mathbf{x}_{k+1} = f_0(\mathbf{x}_k)$, i.e., the free dynamics of the unknown system. Recall that for EDMDc, we have the observables ordered as $\Psi(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \Psi_{\mathbf{x}} & | & \Psi_{\mathbf{u}} \end{bmatrix}$. It will again be important that we keep track of the observables which are only functions of the state. After applying EDMDc, we get the Koopman LTI control system (2.29). The goal of ResDMDc is to compute an error bound on each eigenvalue of $\mathbb{K}_{\mathbf{x}}$ with respect to the spectrum of $\mathcal{K}_x$. That is, we want to find error bounds on the eigenvalues of the free dynamics of the Koopman model with respect to the spectrum of the free dynamics of the underlying nonlinear system. This will enable us to analyze the stability of the underlying system without control input, and then develop a controller that pushes the eigenvalues of the Koopman system *and* their error balls into a stable region of the complex plane.

In order to do this, we must again separate the dictionary into observables that are only functions of the state and observables that are dependent on both the state and control. If we didn't do this and proceeded with ResDMD using the full dictionary $\Psi(\mathbf{x}, \mathbf{u})$, we would get error bounds on eigenvalues of the total matrix $\mathbb{K}$, which are uninterpretable for designing control since control is already embedded in $\mathbb{K}$. Therefore, when computing the residuals in ResDMDc, we will only use the state-dependent observables $\Psi_{\mathbf{x}}$. Let $V_{N_{\mathbf{x}}} = \text{span}\{\psi_1(\mathbf{x}), \psi_2(\mathbf{x}), \ldots, \psi_{N_{\mathbf{x}}}(\mathbf{x})\}$. Let $(\lambda, \mathbf{g})$ be an eigenvalue-eigenvector pair of $\mathbb{K}_{\mathbf{x}}$. The associated approximate eigenfunction is $\phi(\mathbf{x}) = \Psi(\mathbf{x})\mathbf{g} \in V_{N_{\mathbf{x}}}$. The measure of

how close the pair $(\lambda, \phi(\mathbf{x}))$ is to being a true eigenvalue-eigenfunction pair of $\mathcal{K}_x$ is given by

$$\text{res}(\lambda, \phi(\mathbf{x})) = \sqrt{\frac{\mathbf{g}^* \left(L_{\mathbf{x}} - \lambda A_{\mathbf{x}}^* - \overline{\lambda} A_{\mathbf{x}} + |\lambda|^2 G_{\mathbf{x}} \right) \mathbf{g}}{\mathbf{g}^* G_{\mathbf{x}} \mathbf{g}}}, \tag{3.9}$$

where we have the state-dependent matrices

$$L_{\mathbf{x}} = \frac{1}{M} \sum_{m=1}^{M} \Psi_{\mathbf{x}}(\mathbf{y}_m)^* \Psi_{\mathbf{x}}(\mathbf{y}_m), \tag{3.10}$$

$$G_{\mathbf{x}} = \frac{1}{M} \sum_{m=1}^{M} \Psi_{\mathbf{x}}(\mathbf{x}_m)^* \Psi_{\mathbf{x}}(\mathbf{x}_m), \tag{3.11}$$

$$A_{\mathbf{x}} = \frac{1}{M} \sum_{m=1}^{M} \Psi_{\mathbf{x}}(\mathbf{x}_m)^* \Psi_{\mathbf{x}}(\mathbf{y}_m). \tag{3.12}$$

Note that $G_{\mathbf{x}}$ and $A_{\mathbf{x}}$ differ from the matrices G and A used for EDMDc as these matrices use the full dictionary of observables $\Psi(\mathbf{x}, \mathbf{u})$, refer back to (3.6). We again use $\text{res}(\lambda, \phi(\mathbf{x}))$ to bound the error of the eigenvalue λ of $\mathbb{K}_{\mathbf{x}}$ to the spectrum of $\mathcal{K}_x$ as before

$$\begin{aligned} \text{res}(\lambda) = \text{dist}(\lambda, \sigma(\mathcal{K}_x)) &= \inf_f \frac{\| [\mathcal{K}_{\mathbf{x}} f](\mathbf{x}) - \lambda f(\mathbf{x}) \|_{L^2(\Omega,\omega)}}{\| f(\mathbf{x}) \|_{L^2(\Omega,\omega)}} \\ &\leq \frac{\| [\mathcal{K}_{\mathbf{x}} \phi](\mathbf{x}) - \lambda \phi(\mathbf{x}) \|_{L^2(\Omega,\omega)}}{\| \phi(\mathbf{x}) \|_{L^2(\Omega,\omega)}} \\ &\approx \text{res}(\lambda, \phi(\mathbf{x})). \end{aligned} \tag{3.13}$$

Using the same arguments as in Subsection 3.1.1, we have that the approximation in the last line of (3.13) becomes an equality, and thus $\text{res}(\lambda) \leq \text{res}(\lambda, \phi(\mathbf{x}))$, as $M \to \infty$. This error bound being exact uses the same assumptions, i.e., an infinite amount of data that

is sampled according to Assumption 1, and so we again emphasize that $\text{res}(\lambda, \phi(\mathbf{x}))$ is only an approximate error bound on $\text{res}(\lambda)$ and should be treated as such.

3.2.1. Analyzing the Accuracy of ResDMDc with an Example

In order to verify that the error bounds on the eigenvalues given by ResDMDc are faithful, we perform ResDMDc on a known *linear* control system. We use an LTI system in order to know the ground truth eigenvalues of the system, namely, the eigenvalues of the state matrix A. We analyze the accuracy of ResDMDc as follows: for every eigenvalue λ_i of $\mathbb{K}_{\mathbf{x}}$, find the distance from λ_i to the nearest eigenvalue of A, and verify that this distance is less than the error bound $\text{res}(\lambda_i)$. Consider the following continuous-time LTI control system

$$\dot{x} = A\mathbf{x} + Bu = \begin{bmatrix} -1 & 1 \\ 1 & -25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u, \tag{3.14}$$

where u is a scalar control input. The eigenvalues of A are $\lambda_1 = -0.96$ and $\lambda_2 = -25.04$. Note that these are both stable, and since $|\lambda_1|$ is considerably smaller than $|\lambda_2|$, the state trajectory will slowly converge along the stable manifold associated λ_1 and will quickly converge along the stable manifold associated with λ_2. Now, we discretize (3.14) using a fourth-order Runge-Kutta scheme with a time step of $dt = 0.01$. The eigenvalues of the discretized system are $e^{\lambda_1 dt} = 0.99$ and $e^{\lambda_2 dt} = 0.78$. To obtain training data, we simulate the discretized system for a time span of $t \in [0, 3]$ seconds for 200 distinct trajectories, where the initial conditions $[x_1]_0, [x_2]_0$ are each sampled from a uniform distribution over $[-10, 10]$, and control is sampled from a uniform distribution between $[-3, 3]$ at each time

step. We analyze the accuracy of ResDMDc with three different choices of observables, which will each exhibit slightly different behavior of ResDMDc.

3.2.2. Dictionary #1: ResDMDc Is Faithful

Let the first dictionary of observables be

$$\Psi_1(\mathbf{x}, u) = \begin{bmatrix} x_1 & x_2 & \sin(x_1) & \cos(x_2) & u \end{bmatrix}. \tag{3.15}$$

With this choice of dictionary, ResDMDc yields error bounds that small and faithful.

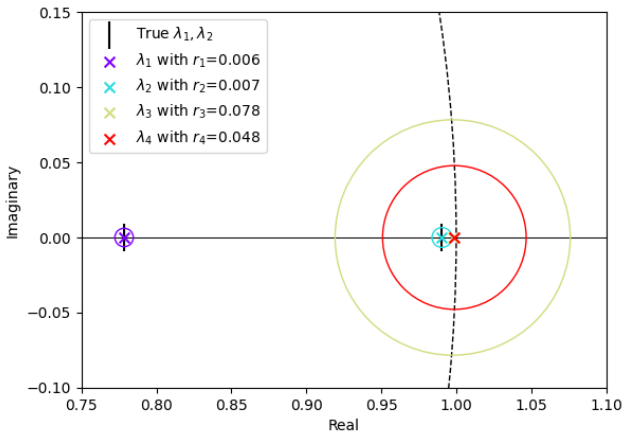

Figure 3.2. The eigenvalues of the true system lie inside the error balls about the eigenvalues of the Koopman model; that is, ResDMDc is working! Further, notice that the eigenvalues of the Koopman model capture both of the true eigenvalues of the true system.

3.2.3. Dictionary #2: ResDMDc Is Faithful But Overly Conservative

The second choice of observables is only slightly modified from (3.15), with $\cos(x_2)$ being replaced with $\sin(x_2)$. This small change will yield ResDMDc error bounds that, while

faithful, are extremely conservative and would lead to an extremely aggressive feedback controller. Let the second dictionary be

$$\Psi_2(\mathbf{x}, u) = \begin{bmatrix} x_1 & x_2 & \sin(x_1) & \sin(x_2) & u \end{bmatrix}. \tag{3.16}$$

This choice of dictionary yields the following figure.

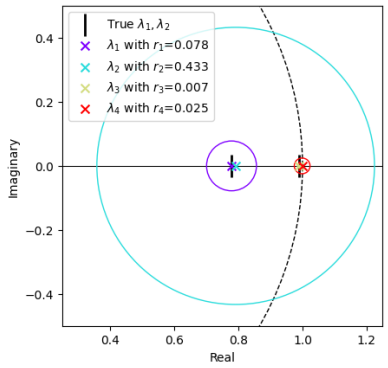

Figure 3.3. It is again the case in each error ball there exists an eigenvalue of the true system; ResDMDc is still producing faithful error bounds. However, the error bound on λ_2 is massive, even though λ_2 is quite close to the true eigenvalue λ_2. Thus, if we were to synthesize a control law that pushes the error balls inside the unit disk, it would be needlessly aggressive in pushing res(λ_2) inside when λ_2 is already stable and accurate.

3.2.4. Dictionary #3: ResDMDc Is Not Faithful

Finally, we show an example where ResDMDc breaks. That is, there exists an eigenvalue with an error ball that does not contain an eigenvalue of the true system. Let the third

choice of dictionary be

$$\Psi_3(\mathbf{x}, u) = \begin{bmatrix} x_1 & x_2 & x_1^2 & x_2^2 & u \end{bmatrix}. \tag{3.17}$$

With this choice of observables, and note that polynomials are a very common choice of observables, we will see in the figure below that one of the error bounds from ResDMDc is not faithful.

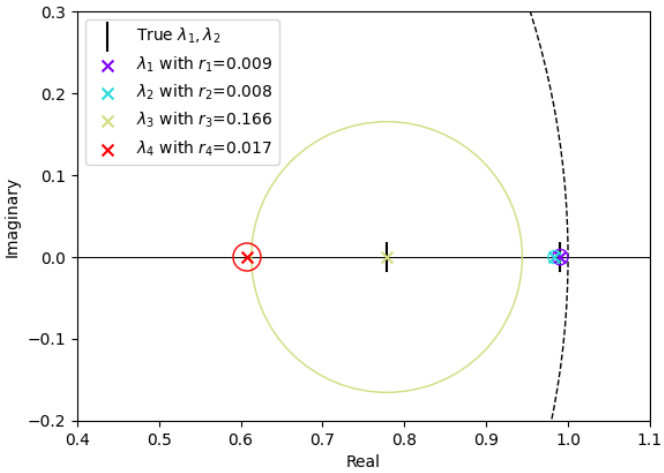

Figure 3.4. The error ball about λ_4 does not contain an eigenvalue of the true system, and so it is not a faithful error bound. Despite clearly being a spurious eigenvalue, λ_4 still has a relatively small residual. Note that we also see the phenomenon present in Figure 3.3, where even though λ_3 is almost exactly equal to the true λ_2, it still has a large error bound. It is counterintuitive that the error bounds on λ_3 and λ_4 are not flipped, i.e., r_4 for λ_3 and r_3 for λ_4

.

These three experiments reaffirm the claim that the ResDMD(c) error bounds should be used with extreme caution. It also, frankly, suggests that there may be a bug in our implementation of ResDMDc in Python. However, digging through each line of code has not unearthed any insidious bugs thus far.

3.2.5. Concluding Remarks

To recap this chapter, we have developed a methodology for computing an LTI Koopman system $\mathbf{z}_{k+1} = \mathbb{K}_{\mathbf{x}}\mathbf{z}(\mathbf{x}_k) + \mathbb{K}_{\mathbf{u}}\mathbf{v}(\mathbf{x}_k, u_k)$ using EDMDc, and computing error bounds on the eigenvalues of $\mathbb{K}_{\mathbf{x}}$ using ResDMDc. We have seen that ResDMDc can yield error bounds exactly like it's supposed to in Figure 3.2, error bounds that are faithful but needlessly large in Figure 3.3, and error bounds that are not faithful Figure 3.4. Nevertheless, while taking great care to appropriately qualify our claims, we will proceed with our controller design assuming that the ResDMDc bounds are faithful and reasonable. Hence, in theory, we can now design a robust stabilizing controller, where the control objective is to push the eigenvalues of $\mathbb{K}_{\mathbf{x}}$ far enough inside a stable region so that every error ball also lies within that region. However, this would require control synthesis via pole placement [**39**], which requires the user to prescribe the location of *all* of the eigenvalues of the closed-loop system. There are clear drawbacks to a pole placement approach. First, it would have to be the case that the pair $(\mathbb{K}_{\mathbf{x}}, \mathbb{K}_{\mathbf{u}})$ is completely controllable. That is too strong of a requirement as we only need the system to be stabilizable, i.e., the unstable modes must be controllable. Further, pole placement design is often energy inefficient and needlessly aggressive, so much so that it may actually exceed actuator saturation limits in real systems. Instead, we use a more pragmatic and structured control approach, one

that is both robust while being maximally conservative in control effort. That is, we use control that guarantees the error balls will be in the stable region *and* does so with minimal control effort. This control algorithm is developed in the next chapter.

CHAPTER 4

Minimal-Norm Stabilizing Feedback

In this chapter we describe the recent collection of work by Gillis et al [**11, 12, 3**]. This work develops efficient computational methods for solving the optimization problem of finding the nearest stable system to an unstable one. In particular, they give a novel parameterization of stable systems and design algorithms to solve the nearest stable system optimization problem, both in continuous and discrete time. They also reframe this problem as a finding minimal-norm stabilizing feedback for the continuous time case. We extend this method to minimal-norm $\mathcal{D}(0, \delta)$-stabilizing control (C2).

4.1. Introduction and Preliminaries

The problem of stabilizing an unstable or marginally stable dynamical system to a compact attractor, which is always a single point for LTI systems, is ubiquitous in control theory and engineering. We begin by making precise what we mean by "stable region of the complex plane" by defining the criterion for an LTI system to be stable, in both continuous and discrete time.

Definition 1. *The continuous-time linear system*

$$\dot{\mathbf{x}} = A\mathbf{x}, \tag{4.1}$$

is said to be stable if $\Re(\lambda) < 0$ *for all eigenvalues* λ *of* A*, and marginally stable if* $\Re(\lambda) \leq 0$ *for all* λ*.*

Definition 2. *The discrete-time autonomous linear system*

$$\mathbf{x}_{k+1} = A\mathbf{x}_k, \quad k \in \mathbb{Z}^{\geq 0} \tag{4.2}$$

is said to be stable if $\rho(A) < 1$ *and marginally stable if* $\rho(A) \leq 1$*, where* $\rho(A) = \max_\lambda |\lambda|$ *is the spectral radius of* A*.*

Thus, for a continuous-time linear system, the stable region of the complex plane is the left half plane, excluding the imaginary axis. For a discrete-time system, the stable region of the complex plane is the unit disk, excluding the unit circle. These stable regions include the imaginary axis and unit circle for marginal stability of continuous-time and discrete-time systems, respectively.

Remark 7. *We will use interchangeably stability of the "system" and stability of the "state matrix" A since these mean the same thing. That is, we say a linear system is stable if its state matrix A is stable.*

Remark 8. *In the work we're building off of, Gillis et al. use the word "stability" to mean "marginal stability" and "asymptotically stability" to mean "stability", with respect to our definitions above. This is frankly pretty confusing.*

With these notions of stability, the problem of finding a feedback control law to stabilize an unstable LTI system is as follows. Consider the control systems in continuous

time

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu \tag{4.3}$$

and discrete time

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + Bu_k, \quad k \in \mathbb{Z}^{\geq 0} \tag{4.4}$$

where $u \in \mathcal{U} \in \mathbb{R}^m$ is the control input and $y \in \mathbb{R}^l$ is the output of the system. If eigenvalues of the free dynamics are unstable, i.e., the eigenvalues of A lie in the right half plane of $\mathbb{C}$ or lie outside the unit disk in $\mathbb{C}$ in (4.3) or (4.4), respectively, then we seek feedback gains $K \in \mathbb{R}^{m \times n}$ such that the feedback law

$$u(\mathbf{x}) = -K\mathbf{x} \tag{4.5}$$

stabilizes the closed-loop dynamics, i.e., $\dot{\mathbf{x}} = (A - BK)\mathbf{x}$ or $\mathbf{x}_{t+1} = (A - BK)\mathbf{x}_t$ is stable.

Remark 9. *Note that under our stabilizing feedback law, the state trajectory of the closed-loop dynamics will exponentially converge to the origin. If convergence to some other fixed point* $\hat{\mathbf{x}}$ *is desired, one can make the coordinate transformation* $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ *so that closed-loop state trajectory in the new coordinates converges to the origin.*

In order to set up the optimization problem for finding stabilizing control gains K such that the 2-norm $\|K\|_2$ is minimized, we first look at the matrix nearness optimization problem of finding the nearest stable system to an unstable system. This optimization problem is formulated as follows: given an unstable matrix $A \in \mathbb{R}^{n \times n}$, the nearest stable

matrix $A^\star$ is the solution to the following constrained optimization problem

$$\begin{aligned} A^\star = \underset{X \in \mathbb{R}^{n\times n}}{\arg\min} \quad & \|A - X\|_F^2 \\ \text{s.t.} \quad & X \text{ is stable,} \end{aligned} \tag{4.6}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbb{S}_n$ is the set of $n \times n$ stable state matrices, which will be different continuous and discrete dynamics. The challenge with solving this problem is that for both continuous and discrete dynamics, the set of stable state matrices $\mathbb{S}_n$ is highly nonconvex and neither closed nor open. To alleviate this challenge, Gillis *et al.* derive a parameterization of the set of stable matrices that transforms the feasible set of (4.6) into a convex and closed one. This parameterization and optimization problem for the nearest stable matrix and minimal-norm stabilizing feedback for continuous-time LTI systems will be presented in the next subsection. After that, we look at the same optimization problem but for $\mathcal{D}(0,\delta)$-stability and stabilizing feedback, where $\mathcal{D}(0,1)$-stability is exactly the notion of stability for discrete-time LTI systems. We will see that the only thing that changes is the parameterization for the set of $\mathcal{D}(0,\delta)$-stable matrices.

4.2. The Continuous-Time Stability Case

4.2.1. Nearest Stable System via Matrix Nearness Optimization

We develop the parameterization of continuous-time stable state matrices as done in [**11**]. The set of continuous-time stable state matrices is

$$\mathbb{S}_n = \{A \in \mathbb{R}^{n\times n} \mid \Re(\lambda) \leq 0 \text{ for all eigenvalues } \lambda \text{ of } A\} \tag{4.7}$$

Refer to [**11**] for discussion and examples on why $\mathbb{S}_n$ as defined in (4.7) is highly nonconvex and neither open nor closed. This makes computing the solution to (4.6) difficult, even though the objective function is a nice smooth, convex quadratic. It is much easier to solve a constrained optimization problem when the feasible set is convex and closed, even if it is at the cost of making the objective function nonconvex. This is what Gillis *et al.* do with the following parameterization.

Theorem 2 (Lemma 2 in [**11**])**.** *A continuous-time state matrix $A \in \mathbb{R}^{n\times n}$ is stable if and only if $A = (J - R)Q$ for some matrices $J, R, Q \in \mathbb{R}^{n\times n}$ such that $J = -J^\top$, $R \succeq 0$, and $Q \succ 0$.*

See [**11**] for the proof.

Remark 10. *There is a physical interpretation to the parameterization in Thm 2. Given matrices $J, R, Q \in \mathbb{R}^{n\times n}$ that satisfy $J = -J^\top$, $R \succeq 0$, and $Q \succ 0$, the system $\dot{\mathbf{x}} = (J-R)Q\mathbf{x}$ is called a dissipative Hamiltonian system, where Q describes the energy of the system, J describes the flux in energy storage, and R describes the energy dissipation.*

With this parameterization, the optimization problem (4.6) becomes

$$\begin{aligned}(J^\star, R^\star, Q^\star) = \underset{J,R,Q\in\mathbb{R}^{n\times n}}{\arg\min} \quad & \|A - (J - R)Q\|_F^2 \\ \text{s.t.} \quad & J = -J^\top, R \succeq 0, Q \succ 0\end{aligned} \tag{4.8}$$

The nearest stable state matrix, i.e., the optimizer of (4.6), is then given by

$$A^\star = (J^\star - R^\star)Q^\star. \tag{4.9}$$

Note that the feasible set is still neither open nor closed, due to the linear matrix inequalities (LMI) $R \succeq 0$ and $Q \succ 0$. The authors in [**4**] show that the optimal solution of (4.8) does not change when $Q \succ 0$ is replaced by $Q \succeq 0$. With the constraints now $J = -J^\top, R \succeq 0, Q \succeq 0$ we finally have a convex and closed feasible set. It is also easy to project onto these constraints, which will be necessary for the optimization algorithm used to solve 4.8. Given a square matrix Z, the projections are as follows

$$\mathcal{P}_{SS}(Z) = \frac{Z - Z^\top}{2} \tag{4.10}$$

$$\mathcal{P}_{\succeq}(Z) = U(\max(\Lambda, 0))U^\top, \tag{4.11}$$

where $\mathcal{P}_{SS}(\cdot)$ is the projection onto the set of skew-symmetric matrices and $\mathcal{P}_{\succeq}(\cdot)$ is the projection onto the set of positive semidefinite matrices, and $Z = U\Lambda U^T$ is an eigenvalue decomposition.

The algorithm used to solve (4.8) is block coordinate descent (BCD). In BCD, one fixes a subset of the variables being optimized over and optimizes over the remaining variables. This process continues until all variables have been optimized over. In their implementation, Gillis *et al.* choose the following subproblems for BCD: (subproblem 1) fix Q and optimize over (J, R), (subproblem 2) fix (J, R) and optimize over Q. Projected gradient descent is used to solve each of these subproblems. For more information on the implementation of BCD, for example, a discussion on various ways to initialize the algorithm, see [**11**].

4.2.2. Minimal-Norm Stabilizing Feedback

Consider a continuous-time LTI control system (4.3), and assume that the state matrix A is unstable. From the previous section, we have an optimization problem formulation and algorithmic solver to find the nearest stable matrix to A. Now, in the context of an LTI system with control input term Bu, we wish to find a control law $u(\mathbf{x}) = -K\mathbf{x}$ such that the closed-loop system $\dot{\mathbf{x}} = (A - BK)\mathbf{x}$ is stable, i.e. $A - BK \in \mathbb{S}_n$, with minimal-norm control gains K. This optimization problem can be formulated as follows

$$\begin{aligned} K^\star = \underset{K \in \mathbb{R}^{m\times n}}{\arg\min} \quad & \|K\|_F^2 \\ \text{s.t.} \quad & A - BK \in \mathbb{S}_n \end{aligned} \tag{4.12}$$

We again use the parameterization that $A - BK \in \mathbb{S}_n \iff A - BK = (J - R)Q$ for some $J = -J^\top$, $R \succeq 0$, and $Q \succ 0$. Solving for K, we get the formula for the control gains $K = B^\dagger(A - (J - R)Q)$. Substituting this into the objective function of (4.12) and using the parameterization for the feasible set, we get

$$\begin{aligned} (J^\star, R^\star, Q^\star) = \underset{J,R,Q\in\mathbb{R}^{n\times n}}{\arg\min} \quad & \|B^\dagger(A - (J - R)Q)\|_F^2 \\ \text{s.t.} \quad & J = -J^\top, R \succeq 0, Q \succ 0, \\ & (I_n - BB^\dagger)(A - (J - R)Q) = 0, \end{aligned} \tag{4.13}$$

where the last equality constraint enforces the stability of $A - BK$, as shown in Theorem 2 of [**3**]. Note again that we can replace $Q \succ 0$ with $Q \succeq 0$ because taking the closure of the feasible set doesn't change the optimal value. The minimal-norm stabilizing feedback

gains are then given by

$$K^\star = B^\dagger(A - (J^\star - R^\star)Q^\star). \tag{4.14}$$

The same BCD algorithm used in the previous subsection is used to solve (4.13).

Remark 11. *It is natural to ask under what conditions $K^\star$ exists. From a linear control theory perspective, if the pair (A, B) is stabilizable, then $K^\star$ is guaranteed to exist. However, Gillis et al. don't make any sort of stabilizability/controllability assumptions in their paper. Rather, they set up and solve a precursor optimization problem, where the problem of stabilizing the system is feasible if and only if the optimizer to that problem is zero. See Theorem 2 and Subsection 5.1.1 in* [**3**] *for more details.*

4.3. The $\mathcal{D}(0, \delta)$-Stability Case

We now present the same sort of results from Section 4.2.1 but for the case of $\mathcal{D}(0, \delta)$-stability. Note that the nearest neighbor discrete-time stable matrix optimization problem was solved in [**11**] and further generalized in [**12**]. We will present the more general case of Ω-stability, where $\Omega \subset \mathbb{C}$ could be a conic sector, vertical strip, disk, or any intersection of the three. We will specifically look at $\Omega = \mathcal{D}(0, \delta) \subset \mathbb{C}$. While the nearest $\mathcal{D}(0, \delta)$-stable matrix problem was done in [**12**], the minimal-norm $\mathcal{D}(0, \delta)$-stabilizing feedback problem has not been done until now, though it was alluded to in Remark 5.3 of [**3**]. This extension is the second contribution of this thesis (C2).

4.3.1. Nearest Stable System via Matrix Nearness Optimization

We present the work on $\mathcal{D}(0,\delta)$-stability developed in [**12**]. A matrix A is said to be $\mathcal{D}(0,\delta)$-stable if all of the eigenvalues of A lie in or on the boundary of $\mathcal{D}(0,\delta)$. The set of all $n \times n$ matrices that are $\mathcal{D}(0,\delta)$-stable is

$$\mathbb{S}_n = \{A \in \mathbb{R}^{n\times n} \mid \rho(A) \leq \delta\}. \tag{4.15}$$

Note that for our controller design, it will be important that δ is allowed to be less than one. Further, note again that we have a feasible set that is nonconvex and neither open nor closed. This is fixed with another parameterization of $\mathbb{S}_n$ for $\mathcal{D}(0,\delta)$-stability, which is similar to the parameterization for continuous-time stable matrices.

Theorem 3 (Theorem 3 in [**12**])**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is $\mathcal{D}(0,\delta)$-stable if and only if $A = (J-R)Q$ for some matrices $J, R, Q \in \mathbb{R}^{n\times n}$ such that $J = -J^\top$, $R^\top = R$, $Q \succ 0$, and*

$$\begin{bmatrix} \delta Q^{-1} & -(J-R) \\ -(J-R)^\top & \delta Q^{-1} \end{bmatrix} \succ 0. \tag{4.16}$$

See [**12**] for the proof. Comparing this parameterization with the parameterization for continuous-time stable matrices in Thm 2, the condition on R is relaxed from being positive semidefinite to only being symmetric, and we have the extra LMI constraint (4.16).

Given a matrix A that is not $\mathcal{D}(0,\delta)$-stable, the nearest matrix to A that is $\mathcal{D}(0,\delta)$-stable is found by the optimization problem

$$
\begin{aligned}
(J^\star, R^\star, Q^\star) = \underset{J,R,Q\in\mathbb{R}^{n\times n}}{\arg\min} \quad & \|A-(J-R)Q\|_F^2 \\
\text{s.t.} \quad & J=-J^\top, R^\top = R, Q \succ 0, \\
& \begin{bmatrix} \delta Q^{-1} & -(J-R) \\ -(J-R)^\top & \delta Q^{-1} \end{bmatrix} \succ 0,
\end{aligned} \tag{4.17}
$$

where the ultimate solution is $A^\star = (J^\star - R^\star)Q^\star$. Note that we can again take the closure of the feasible set without changing the optimizer. More importantly, however, observe that the feasible set isn't yet convex due to the Q^{-1} terms. It is advantageous to have a convex feasible set, even at the expense of losing convexity of the objective function. Therefore, the authors make the transformation $P = Q^{-1}$ so that (4.18) becomes

$$
\begin{aligned}
(J^\star, R^\star, P^\star) = \underset{J,R,P\in\mathbb{R}^{n\times n}}{\arg\min} \quad & \|A-(J-R)P^{-1}\|_F^2 \\
\text{s.t.} \quad & J=-J^\top, R^\top = R, P \succeq 0, \\
& \begin{bmatrix} \delta P & -(J-R) \\ -(J-R)^\top & \delta P \end{bmatrix} \succeq 0.
\end{aligned} \tag{4.18}
$$

The nearest $\mathcal{D}(0,\delta)$-stable matrix to A is thus

$$
A^\star = (J^\star - R^\star)(P^\star)^{-1} \tag{4.19}
$$

4.3.2. Minimal-Norm Stabilizing Feedback (C2)

We develop and solve this optimization problem using the technique suggested (but not implemented) in Section 5.3 of [**3**]. We set up this optimization problem with the exact same structure as (4.13), except now with the $\mathcal{D}(0,\delta)$-stable parameterization of the feasible set. Hence, we have

$$
\begin{aligned}
(J^\star, R^\star, P^\star) = \underset{J,R,P\in\mathbb{R}^{n\times n}}{\arg\min} \quad & \|B^\dagger(A-(J-R)P^{-1})\|_F^2 \\
\text{s.t.} \quad & J=-J^\top, R=R^\top, Q\succeq 0, \\
& \begin{bmatrix} \delta P & -(J-R) \\ -(J-R)^\top & \delta P \end{bmatrix} \succeq 0, \\
& (I_n - BB^\dagger)(A-(J-R)P^{-1}) = 0.
\end{aligned} \tag{4.20}
$$

We solve this with BCD as before to arrive at the minimal-norm $\mathcal{D}(0,\delta)$-stabilizing feedback gains

$$
K^\star = B^\dagger\left(A-(J^\star-R^\star)(P^\star)^{-1}\right). \tag{4.21}
$$

We show these feedback gains in action in the following subsection.

4.3.3. Example

We show an example to illustrate the discrete-time stability case, $\Omega(0,1)$, as well as $\mathcal{D}(0,\delta)$ for $\delta < 1$. Consider the following discrete-time system

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + Bu_k = \begin{bmatrix} 1 & 0 & 2 \\ 1 & -1 & -1 \\ 0 & 2 & -1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} u_k. \tag{4.22}$$

The eigenvalues of A are $\lambda_1 = 1.4883$, $\lambda_2 = -1.2442{+}1.7764i$, and $\lambda_3 = -1.2442{-}1.7764i$. Observe that all of these eigenvalues are unstable. We set up (4.20) and run the BCD solver, which converged in about 15 iterations, resulting in Figure 4.1 below.

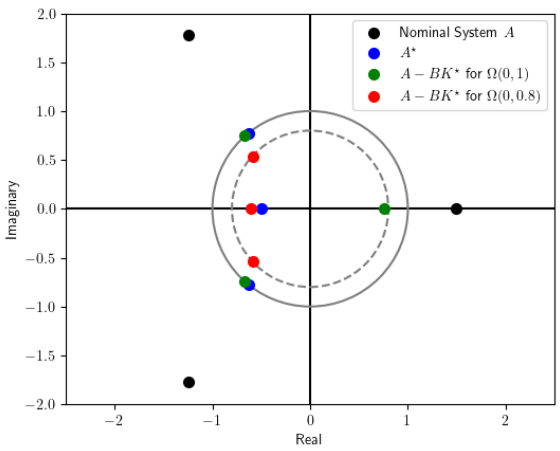

Figure 4.1. Plot of the eigenvalues of the nominal system (4.22), the solution to $(0,\delta)$-stabilizing feedback for both $\delta = 1$ and $\delta = 0.8$, and the nearest discrete-time stable matrix $A^\star$ generated by the code in [**4**] using only the state matrix A. Observe that two different minima were found, which is a property of the optimization problem formulation and solver, as discussed by Gillis *et al.* in [**4**].

Before we finally get to controller design, we make a few remarks on why we use the minimal-norm stabilizing feedback methods from this chapter.

Remark 12. *In order to use the minimal-norm stabilizing feedback algorithms, it is assumed that the system is of the form of an LTI control system. This is another, more practical reason for learning an LTI control system for the Koopman dynamics. If we learned a bilinear Koopman system, we would not be able to use any of this work.*

Remark 13. *It was necessary that we extend the minimal-norm stabilizing feedback from* [**3**] *to be able to generate minimal-norm $\mathcal{D}(0,\delta)$-stabilizing feedback. The approximate Koopman operator and the corresponding LTI Koopman dynamics may be framed in either continuous or discrete time because we have the relationship*

$$\widetilde{\mathbb{K}} = \frac{log(\mathbb{K})}{t_s},$$

where $\widetilde{\mathbb{K}}$ is the continuous-time Koopman operator and t_s sampling time. However, there is no mathematical transformation that can be done to the residuals from ResDMD that will give error bounds on the eigenvalues of the continuous Koopman dynamics.

CHAPTER 5

Putting It All Together: Our Controller (C3)

5.1. Introduction and Procedure

We now have all of the motivation and ingredients we need for our control design! Namely, ingredient #1 is that we can compute error bounds on the eigenvalues of the Koopman LTI system via ResDMDc, as discussed in Section 3.2; ingredient #2 is that we can stabilize a discrete-time LTI control system to the region $\mathcal{D}(0, \delta)$ with minimal-norm control gains, as discussed in Subsection 4.3.2. We can combine these ingredients to achieve the following: instead of designing stabilizing feedback to push the eigenvalues of the Koopman model into the unit disk, we design feedback to push the *error balls* about each eigenvalue of the Koopman model into the unit disk, all with minimal control effort. The following cartoon gives a pictorial description of the idea.

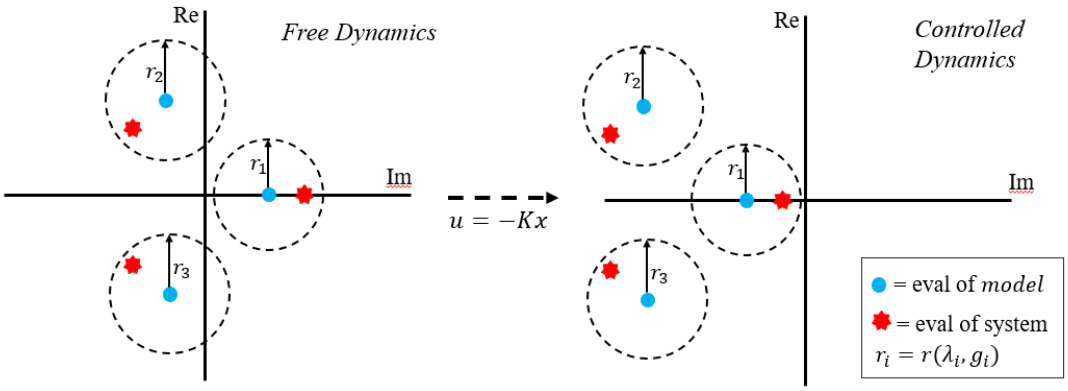

Figure 5.1. Here we illustrate the motto of our controller design—stabilize the error balls! Note that this stabilization is in the context of continuous systems, and we will exclusively look at discrete-time stability in what follows. So, the motto is actually to push the error balls inside the unit disk.

Keep this ultimate goal in mind as we proceed. Consider again a discrete-time control-affine dynamical system

$$\mathbf{x}_{k+1} = f_0(\mathbf{x}_k) + \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_k)[\mathbf{u}_i]_k \tag{5.1}$$

with state $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, control $\mathbf{u} \in \mathcal{U} \subset R^m$, and unknown drift vector field $f_0(\mathbf{x})$ and control vector fields $f_i(\mathbf{x})$ for $i \in \mathcal{I}$. We will perform EDMDc, ResDMDc, and apply minimal-norm $\mathcal{D}(0, \delta)$-stabilizing feedback for our controller design, but first, we state our assumptions.

Assumption 2. *For simplicity and ease of exposition, we assume that the control input u is a scalar, i.e., $u \in \mathcal{U} \subset \mathbb{R}$. Extending to the multi-input control case doesn't fundamentally change the design of our controller.*

Assumption 3. *We include the state* $\mathbf{x}$ *of the original system in the dictionary of observables. That is,* $\psi_i(\mathbf{x}) = \mathbf{x}_i$ *for* $i = 1, 2, \ldots, d$*, and for* $i > d$ *the observables* $\psi_i(\mathbf{x})$ *are user-chosen nonlinear functions of the state. Note that this makes state reconstruction from the Koopman dynamics trajectory trivial. Further, it must be the case that the size of the dictionary is at least the size of the state, i.e.,* $N \geq d$*. This isn't always the case, for example, in fluid dynamics where the state vector can be very large. For ResDMD analysis for the case when* $N < d$*, see* [**40**]*. Note that when it comes to application, this is a fairly restrictive assumption as it requires full observability of the state.*

From assumptions 2 and 3, we have the following dictionary of observables

$$\Psi(\mathbf{x}, u) = \begin{bmatrix} \mathbf{x}^\top & \psi_1(\mathbf{x}) & \psi_2(\mathbf{x}) & \cdots & \psi_N(\mathbf{x}) & u \end{bmatrix} \in \mathbb{C}^{1\times(d+N+1)} \tag{5.2}$$

We also make an assumption on the spectrum of the free dynamics of the underlying nonlinear system.

Assumption 4. *The drift vector field* $f_0(\mathbf{x})$ *has a pure point spectrum; that is, it has no continuous spectrum. For example, the system cannot be chaotic.*

Note that Assumption 4 is also made in [**1**].

The first step of our control algorithm is to compute a Koopman LTI system using EDMDc. This yields the system

$$\mathbf{z}_{k+1} = \mathbb{K}_{\mathbf{x}}\mathbf{z}_k + \mathbb{K}_{\mathbf{u}}\mathbf{v}_k, \tag{5.3}$$

where $\mathbb{K}_{\mathbf{x}} \in \mathbb{C}^{(d+N)\times(d+N)}$ and $\mathbb{K}_{\mathbf{u}} \in \mathbb{C}^{(d+N)\times 1}$. We run the ResDMDc algorithm concurrently with EDMDc. From ResDMDc, we get an error bound $r(\lambda_i)$, which is the residual

$\text{res}(\lambda_i, \phi_i(\mathbf{x}))$, on the eigenvalue λ_i of $\mathbb{K}_{\mathbf{x}}$ for each $i = 1, 2, \ldots, (d+N)$. Now, we want to apply minimal-norm $\mathcal{D}(0, \delta)$-stabilizing feedback control to get all of the error balls inside the unit disk. So what should we make δ? We define δ to be

$$\delta := 1 - \arg\max_{r(\lambda_i)} \left(|\lambda_i| + r(\lambda_i) \right). \tag{5.4}$$

Naively, one might think that we should just take $\delta = \max r(\lambda_i)$, but it may be the case that the eigenvalue λ_i is already deep inside the unit disk and, even if it has the largest associated residual, will be the case that $|\lambda_i| + r(\lambda_i) < 1$. How we have defined δ in (5.4) essentially states that δ should be taken to be one minus the residual $r(\lambda_i)$, where $|\lambda_i| + r(\lambda_i)$ sticks the furthest out from the unit circle in the complex plane. The following cartoon illustrates the choice of δ.

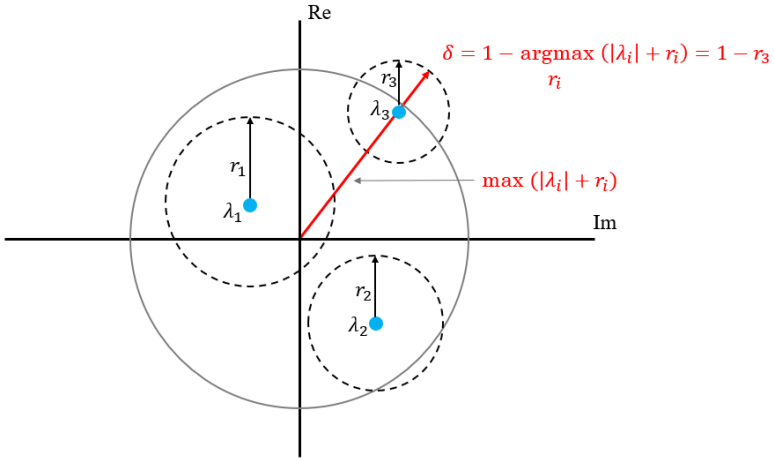

Figure 5.2. In this cartoon, there are three eigenvalues λ of the data-driven model, each with an error bound r. Observe that r_1 is the largest residual, however, the error ball is completely contained in the unit disk due to λ_1 already being quite stable. Instead, we choose $\delta = 1 - r_3$.

Once we have δ as defined in (5.4), we solve the minimal-norm $\mathcal{D}(0,\delta)$-stabilizing feedback optimization problem... and voila! We have pushed the error balls into the unit disk.

Remark 14. *There is an underlying assumption here, which is that the pair* $(\mathbb{K}_{\mathbf{x}}, \mathbb{K}_{\mathbf{u}})$ *must be stabilizable* $\mathcal{D}(0,\delta)$*-stabilizable. If this assumption doesn't hold, the optimization algorithm will return an error due to infeasibility.*

We summarize the algorithmic design of our controller in the following section.

5.2. Our Algorithm

(1) Choose nonlinear functions of the state $\psi_1(\mathbf{x}), \ldots, \psi_N(\mathbf{x})$ and construct dictionary of observables as in (5.2).

(2) Measure state data $\{\mathbf{x}_m, \mathbf{y}_m\}_{m=1}^{M}$, where $\mathbf{y}_m = \mathbf{x}_{m+1} = f_0(\mathbf{x}_m) + \sum_{i\in\mathcal{I}} f_i(\mathbf{x}_m)u_m$, and control input $\{u_m\}_{m=1}^{M+1}$.

(3) Perform EDMDc and ResDMDc concurrently to get the Koopman dynamics (5.3), and an error bound (residual) $r(\lambda_i)$ on each eigenvalue λ_i of $\mathbb{K}_{\mathbf{x}}$

(4) Define δ as in (5.4)

(5) Perform minimal-norm $\mathcal{D}(0,\delta)$-stabilizing feedback to get feedback gains $K^\star \in \mathbb{R}^{1\times(d+N)}$

(6) apply feedback $u(z) = -K^\star z$ to the underlying system (5.1)

CHAPTER 6

Simulation Experiment

Here we show our overall control algorithm described in Chapter 5 in action on a simple dynamical system. Consider the following planar nonlinear control-affine system

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \mu x_1 \\ \gamma(x_2 - x_1^2) \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} u, \tag{6.1}$$

where u is a scalar. We choose $\mu = -1$ and $\gamma = 0.1$ so that the origin is an unstable equilibrium point of (6.1). Further, note that because γ is small, the trajectory on the unstable manifold will only diverge slowly. This was necessary in order to get a sufficient amount of training data before the trajectories blew up.

The parameters of our simulation are as follows. We used a time span of $t \in [0, 3]$ seconds and time step $dt = 0.01$, and ran 300 separate simulations to collect data. The initial condition was sampled from a uniform distribution over $[-10, 10]$, and the control input at each time step was sample from a uniform distribution over $[-3, 3]$. The dictionary of observables we used was

$$\Psi(\mathbf{x}, u) = \begin{bmatrix} x_1 & x_2 & x_1^2 - 1 & x_2^2 - 1 & x_1^3 - 3x_1 & x_2^3 - 3x_2 & u \end{bmatrix}, \tag{6.2}$$

which are Hermitian polynomials of x_1 and x_2 up to third order, and a scalar control observable u. We used these observables and simulation data to learn a Koopman model

and run ResDMDc, which resulted in the following eigenvalues of the Koopman model and their associated ResDMDc error bounds.

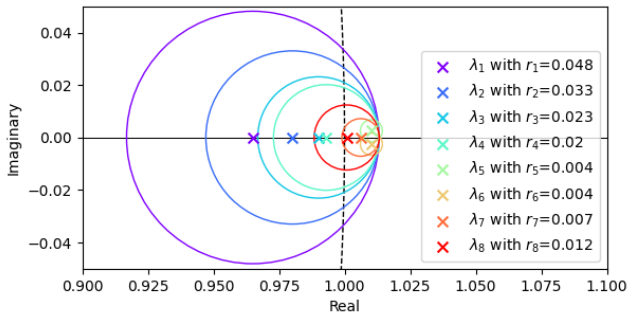

Figure 6.1. The eigenvalues of the Koopman model with their associated error bounds.

We compute the δ parameter from (5.4)

$$\delta = 1 - \arg\max_{r(\lambda_i)} \left(|\lambda_i| + r(\lambda_i)\right) = 1 - 0.004 = 0.996. \tag{6.3}$$

Even though r_5 is the smallest residual, it corresponds to the eigenvalue λ_5 that is the most unstable, so together $|\lambda_5| + r(\lambda_5)$ sticks the furthest out away from the unit circle. With this choice of δ, we run the minimal-norm $\mathcal{D}(0, \delta)$-stabilizing algorithm to yield feedback gains that when applied to (6.1) do successfully stabilize the system, as seen in the figure below.

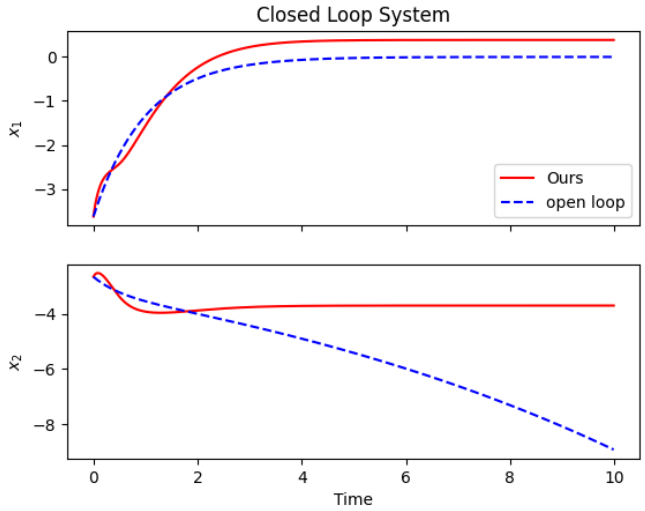

Figure 6.2. The closed loop trajectory is generated by applying the feedback control generated by our algorithm to the system (6.1. The open loop system is generated by simulating (6.1) with no control input. This plot shows that our controller does successfully stabilize an unstable nonlinear system!

To be transparent, depending on the choice of nonlinear system, dictionary of observables, and the amount of data to collect, our control algorithm broke many times. That is, it was unable to compute control gains because the $\mathcal{D}(0, \delta)$-stabilization was infeasible (so say the solver we used for this). We believe the root cause of this is the variability of the ResDMDc error bounds, as discussed in Subsection 3.2.1.

CHAPTER 7

Discussion and Future Work

While the preliminary analysis and numerical results suggest our controller is more robustly stabilizing nonlinear systems, there are still considerable holes in this work, both in terms of theoretical guarantees and real-world applicability. Before we discuss theoretical and practical extensions, we remark yet again on the limitations of ResDMD(c).

7.1. Guaranteeing stability

There is a reason we say "increasing robustness" in our title. The ultimate goal of this work is to guarantee stability under our feedback controller, and we have put forth considerable time and effort towards this goal, but we have thus far found more fundamental roadblocks than solutions. We have identified the following three roadblocks. To overcome these challenges (if possible), we will need to find assumptions on the underlying nonlinear system that are strong enough to do some heavy lifting in a theorem/proof, yet week enough that we don't restrict our class of systems down to a frictionless pendulum.

7.1.1. The curse of infinite-dimensionality

The most obvious obstruction is that the spectrum of $\mathbb{K}$ is discrete and finite, while the spectrum of $\mathcal{K}$ is comprised of a countably infinite point spectrum and an uncountably infinite continuous spectrum. Therefore, the spectrum of $\mathbb{K}$ will never be able to fully

capture the spectrum of $\mathcal{K}$, which results in the following possible scenario depicted in Figure below

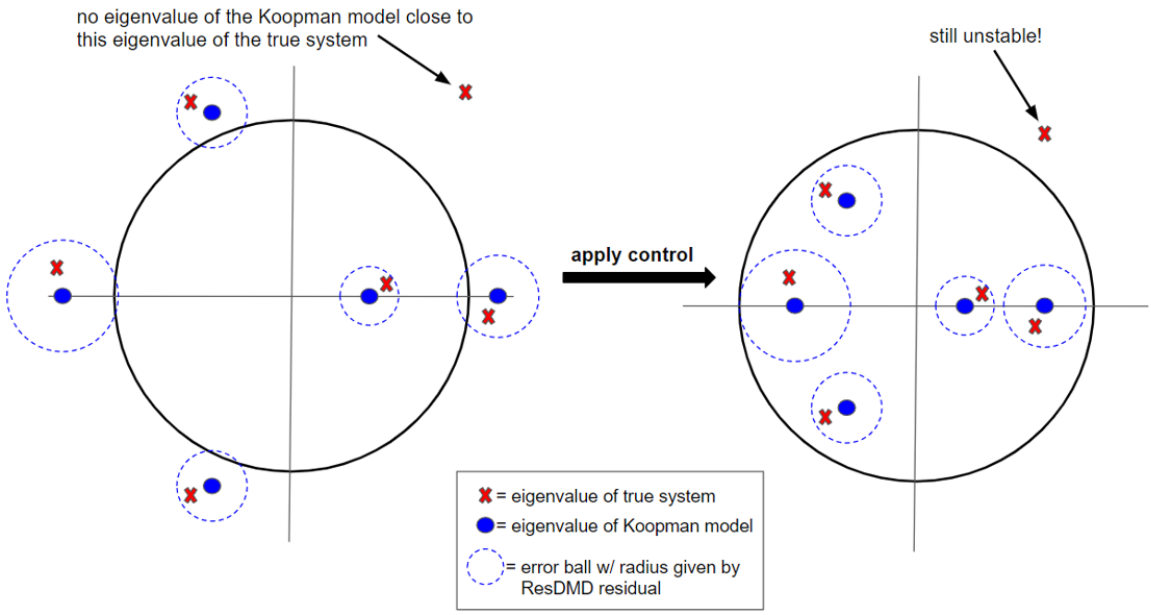

Figure 7.1. This is an example of the spectral invisibility issue in action. We cannot, at least theoretically, guarantee that we capture all of the unstable eigenvalues with our current framework.

Foundationally, computing the full spectrum of $\mathcal{K}$, or any general infinite-dimensional operator, is impossible. In the language of Colbrook and Hansen in [**41**], this problem is of the type Σ_1 in the Solvability Complexity Index hierarchy. In the language of a layman, this problem is of the type "forget about it." One idea to circumvent this curse of infinite-dimensionality while still establishing theoretical guarantees is to only analyze modes which have energy above some threshold. This assumption on the underlying

system may be hard to check, but it does seem reasonable to assume that even in an unstable system, there are only finitely many unstable modes with non-negligible energy.

Along the lines of not fully capturing the spectrum of $\mathcal{K}$ with $\mathbb{K}$, another future direction is to relax the assumption that the dynamical system we're stabilizing has pure point spectrum. We will need to fully consider the continuous spectrum of $\mathcal{K}$ if we wish to stabilize chaotic dynamical systems since a nontrivial continuous spectrum is a hallmark of chaotic dynamics. Since we aren't considering chaotic dynamical systems, we were able to restrict our analysis to only the point spectrum and disregard the continuous spectrum, see Remark 3 of [**42**]. In future work if we do try to stabilize a chaotic system, e.g. stabilize the canonical Lorenz system down to a stationary fixed point, we may be able to use the computation of spectral measures in [**2**]. These spectral measures are exactly used for approximating the continuous spectrum of a general Koopman operator.

7.1.2. Eigenvalues Staying Inside Error Balls Under Feedback

Another assumption that we have made is that the eigenvalue(s) of the true system that lie inside the ResDMD error balls stay inside the error ball when feedback is applied. We didn't realize we were making this assumption until my collaborator, Giorgos Mamkoukas, recently pointed it out, and it's totally true—the true nonlinear system and data-driven model are different and thus will behave differently under feedback!

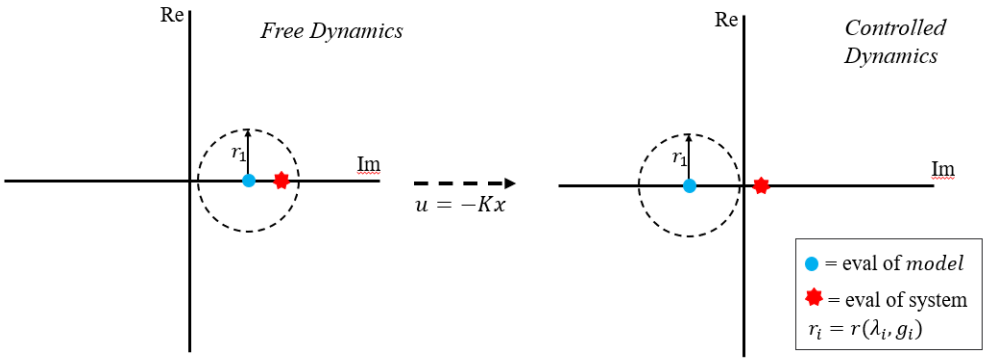

Figure 7.2. The true nonlinear system and the data-driven model will respond differently to feedback, e.g., the eigenvalues of one may move faster into a stable region than the eigenvalues of the other. As a result, we cannot guarantee that the eigenvalue(s) that originally lie in the ResDMD error ball stays in the error ball when feedback is applied.

Furthermore, we have not analyzed what feedback does to the error balls. We expect that under stabilizing feedback, the error balls will shrink in diameter (residuals will decrease) because it one can be more confident in EDMD(c) modeling a stable system than an unstable system. However, this is not a roadblock to stability as bigger error balls leads to a more aggressive stabilizing controller.

7.2. ResDMD(c) Interpretation and Caution

It is important to note that even if we could solve the above to challenges; that is, we could capture every unstable eigenvalue of a system with pure point spectrum and guarantee that these eigenvalues stayed in the error balls under feedback, we still

wouldn't be able to guarantee stability! The ResDMD error bounds are only approximate with finite data, and require Assumption 1 to be exact in the case where $M \to \infty$.

In conclusion, to be able to write down a formal proof guaranteeing stability under our feedback law, it needs to be the case that

(1) The underlying nonlinear system has pure point spectrum.
(2) Further, it has only finitely many unstable eigenvalues which are all captured by ResDMDc.
(3) We guarantee that these unstable eigenvalues remain in the error balls under feedback.
(4) Assumption 1 holds.
(5) We collect an infinite amount of data, sampled according to Assumption 1.

This list is clearly comical... Upshot: we can't guarantee stability.

7.3. Robustness with Respect To Noisy Measurements

Finally, we conclude with a remark on extensions to enable real-world application. At the end of the day, even though we can't guarantee stability, our controller can still be useful in practical engineering applications. However, there is also a considerable obstacle to using our controller on real systems. Currently, our training data is sampled from a simulated trajectory, so the measurements are uncontaminated by noise. This clearly won't be the case if we collected data from sensor measurements of a physical system, e.g., sensor measurements on a robot arm. In order to use measured experimental data in our controller design, we would also need to incorporate robustness with respect to

measurement noise. We may be able to get this noise robustness using Colbrook's newest ResDMD work in [**43**], which is a formulation of ResDMD for stochastic systems.

References

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

[2] M. J. Colbrook and A. Townsend, "Rigorous data-driven computation of spectral properties of koopman operators for dynamical systems," *Communications on pure and applied mathematics : a journal*, 2023-07-27.

[3] N. Gillis and P. Sharma, "Minimal-norm static feedbacks using dissipative hamiltonian matrices," *Linear Algebra and its Applications*, vol. 623, pp. 258–281, 2021.

[4] N. Gillis, M. Karow, and P. Sharma, "Approximating the nearest stable discrete-time system," *Linear Algebra and its Applications*, vol. 573, pp. 37–53, 2019.

[5] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[6] J.-J. E. Slotine, "Applied nonlinear control," *PRENTICE-HALL google schola*, vol. 2, pp. 1123–1131, 1991.

[7] W.-H. Steeb and F. Wilhelm, "Non-linear autonomous systems of differential equations and carleman linearization procedure," *Journal of Mathematical Analysis and Applications*, vol. 77, no. 2, pp. 601–611, 1980.

[8] S. Prajna, A. Papachristodoulou, and F. Wu, "Nonlinear control synthesis by sum of squares optimization: A lyapunov-based approach," in *2004 5th Asian control conference (IEEE Cat. No. 04EX904)*, vol. 1, pp. 157–165, IEEE, 2004.

[9] M. J. Colbrook, L. J. Ayton, and M. Szőke, "Residual dynamic mode decomposition: robust and verified koopmanism," *Journal of Fluid Mechanics*, vol. 955, p. A21, 2023.

[10] K. Zhou and J. C. Doyle, *Essentials of robust control*, vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[11] N. Gillis and P. Sharma, "On computing the distance to stability for matrices using linear dissipative hamiltonian systems," *Automatica*, vol. 85, pp. 113–121, 2017.

[12] N. Choudhary, N. Gillis, and P. Sharma, "On approximating the nearest ω-stable matrix," *Numerical Linear Algebra with Applications*, vol. 27, no. 3, p. e2282, 2020.

[13] W. Tang and P. Daoutidis, "Data-driven control: Overview and perspectives," in *2022 American Control Conference (ACC)*, pp. 1048–1064, IEEE, 2022.

[14] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[15] B. Huang, X. Ma, and U. Vaidya, "Feedback stabilization using koopman operator," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6434–6439, IEEE, 2018.

[16] S. Sinha, S. P. Nandanoori, J. Drgona, and D. Vrabie, "Data-driven stabilization of discrete-time control-affine nonlinear systems: A koopman operator approach," in *2022 European Control Conference (ECC)*, pp. 552–559, IEEE, 2022.

[17] R. Strässer, M. Schaller, K. Worthmann, J. Berberich, and F. Allgöwer, "Koopman-based feedback design with stability guarantees," *arXiv preprint arXiv:2312.01441*, 2023.

[18] A. Narasingam and J. S.-I. Kwon, "Closed-loop stabilization of nonlinear systems using koopman lyapunov-based model predictive control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 704–709, IEEE, 2020.

[19] I. Abraham, G. De La Torre, and T. D. Murphey, "Model-based control using koopman operators," in *2017 Robotics: Science and Systems, RSS 2017*, MIT Press Journals, 2017.

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

[21] 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.

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

[23] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2020.

[24] M. L. Castaño, A. Hess, G. Mamakoukas, T. Gao, T. Murphey, and X. Tan, "Control-oriented modeling of soft robotic swimmer with koopman operators," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1679–1685, IEEE, 2020.

[25] A. Volchko, S. K. Mitchell, T. G. Morrissey, and J. S. Humbert, "Model-based data-driven system identification and controller synthesis framework for precise control of siso and miso hasel-powered robotic systems," in *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pp. 209–216, IEEE, 2022.

[26] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning stable models for prediction and control," *IEEE Transactions on Robotics*, 2023.

[27] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche, "Diffeomorphically learning stable koopman operators," *IEEE Control Systems Letters*, vol. 6, pp. 3427–3432, 2022.

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

[29] F. Nüske, S. Peitz, F. Philipp, M. Schaller, and K. Worthmann, "Finite-data error bounds for koopman-based prediction and control," *Journal of Nonlinear Science*, vol. 33, no. 1, p. 14, 2023.

[30] S. E. Otto and C. W. Rowley, "Koopman operators for estimation and control of dynamical systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 59–87, 2021.

[31] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[32] M. D. Kvalheim and P. Arathoon, "Linearizability of flows by embeddings," *arXiv preprint arXiv:2305.18288*, 2023.

[33] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

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

[35] M. Korda and I. Mezić, "On convergence of extended dynamic mode decomposition to the koopman operator," *Journal of Nonlinear Science*, vol. 28, pp. 687–710, 2018.

[36] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[37] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021.

[38] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2015.

[39] C.-T. Chen, *Linear system theory and design*. Saunders college publishing, 1984.

[40] M. J. Colbrook, "Another look at residual dynamic mode decomposition in the regime of fewer snapshots than dictionary size," *arXiv preprint arXiv:2403.05891*, 2024.

[41] M. J. Colbrook and A. C. Hansen, "The foundations of spectral computations via the solvability complexity index hierarchy," *Journal of the European Mathematical Society*, vol. 25, no. 12, pp. 4639–4718, 2022.

[42] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[43] M. J. Colbrook, Q. Li, R. V. Raut, and A. Townsend, "Beyond expectations: residual dynamic mode decomposition and variance for stochastic dynamical systems," *Nonlinear Dynamics*, vol. 112, no. 3, pp. 2037–2061, 2024.