University of California
Santa Barbara

Learning and Control of Deep Koopman Eigenfunctions

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

Master of Science
in
Electrical and Computer Engineering

by

Alan Blake Cao

Committee in charge:

Professor Igor Mezić, Co-Chair
Professor João Hespanha, Co-Chair
Professor Jason Marden

March 2022

The Thesis of Alan Blake Cao is approved.

Professor Jason Marden

Professor João Hespanha, Committee Co-Chair

Professor Igor Mezić, Committee Co-Chair

December 2021

Learning and Control of Deep Koopman Eigenfunctions

by

Alan Blake Cao

Acknowledgements

First and foremost, I would to thank my advisor and mentor, Professor Igor Mezić for dedicated involvement in every step throughout this process.

I would like to express my deepest appreciation for Michael Banks. Our discussions have been invaluable throughout my time here at the University of California, Santa Barbara. He encouraged my ideas and fostered my growth as a researcher while always steering me in the right direction. Furthermore, I am indebted for his comments on this thesis.

Most importantly, I would like to thank my parents, Hsin-Ru Chang and Tuqiang Cao, and my other half, Irene Pattarachanyakul. Without their constant support and affection throughout my years of study, none of this would have been possible.

Abstract

Learning and Control of Deep Koopman Eigenfunctions

by

Alan Blake Cao

A nonlinear dynamical system can be represented by an infinite-dimensional linear operator known as the Koopman operator. Observables are scalar-valued functions of the state space that collectively form a linear vector space. Although all observables evolve linearly under the Koopman operator, special observables called eigenfunctions can be decoupled from other observables and span a Koopman-invariant subspace. Finding a finite approximation of the Koopman operator allows the application of well-developed linear systems methodologies to nonlinear systems. Numerical methods such as Dynamic Mode Decomposition (DMD) and its variants are widely used to produce finite approximations of the Koopman operator. Unfortunately, the approximations produced by these numerical methods are highly sensitive to the choice of observables, which are typically user-defined. In this work, we introduce a Koopman-inspired deep learning architecture that extracts the eigenfunctions of discrete spectrum systems, resulting in a diagonal representation of the Koopman operator. In numerical examples, the eigenfunctions learned using this framework exhibit a predictive performance superior to existing methods. Finally, we extend the architecture to *controlled* dynamical systems. Numerical examples show that the linear predictors obtained in this way can be readily used to design controllers that directly act on the Koopman modes of the system.

Contents

Chapter 1

Introduction

Dynamical systems theory describes properties of systems that evolve in time, and can be applied to a wide variety of fields including physics, biology, and economics. Dynamical systems can be mathematically represented as differential or difference equations that govern the evolution of a set of quantities, or states. One crucial motivation for understanding and modelling dynamical systems is the potential for control. The objective of control is to drive a system to some desired state by applying a control action. Control theory is crucial in most applications involving robotics, power systems, or chemical processes. Control is even irreplaceable in our daily lives: our senses provide reliable feedback allowing us to perform actions as desired. Although substantial theory has been developed for the analysis and control of linear systems, it does not easily translate to nonlinear systems. Nonlinearity is ubiquitous in physical phenomena, and representing nonlinear dynamics in a globally linear formulation would allow existing linear theory to be applied to a much larger class of systems.

A linearization of a nonlinear system is usually the first order Taylor expansion about a desired point. This yields a linear representation that becomes less accurate the further

away the system strays from the point of linearization. A globally linear representation would be superior and allow for control outside of a small neighborhood. Originally attributed to Koopman, Carleman, and Von Neumann, the idea of modelling nonlinear dynamics using infinite-dimensional linear operators offers an approach for finding a globally linear representation. We define an observable as a scalar-valued function of the state space, and note that the space of all observables forms a linear vector space. The idea is to lift nonlinear dynamics into the infinite-dimensional space of all observables where the evolution is described by the action of the so-called Koopman operator [6, 7]. Importantly, this operator is linear and the observables evolve linearly even though the underlying dynamics are nonlinear. In addition, the Koopman operator offers spectral information regarding coherent structures and frequencies in the underlying dynamics [17, 12]. Recent renewed interest in Koopman theory can be mainly attributed to theoretical advances [14, 3, 13, 2] and improved numerical methods such as Dynamic Mode Decomposition (DMD) [19, 20], Extended DMD (EDMD) [21], Hankel DMD [1], and deep DMD [9, 22]. These numerical methods produce finite-dimensional approximations of the Koopman operator from data that can be utilized in real-world applications. A dictionary of observables is used to lift the dynamics to a high-dimensional space where the evolution is approximately linear. Unfortunately, the observables are typically user-defined and highly affect the quality of the approximations. Of all possible choices for observables, the eigenfunctions of the Koopman operator are ideal because they can be decoupled from other observables and span a Koopman-invariant subspace. Identifying and representing the Koopman eigenfunctions has proven to be difficult in practice. Other methods have tried to use neural networks to learn Koopman eigenfunctions from data. In particular, [10] allows the eigenvalues to vary across the state space, resulting in a nonlinear representation of the dynamics. Moreover, these methods were applied to systems with continuous spectrum, in which there are no Koopman eigenfunctions ex-

cept for invariants. In contrast, our method directly learns true Koopman eigenfunctions for discrete spectrum systems. Importantly, our method learns Koopman eigenfunctions associated with constant eigenvalues to produce a spectral expansion of the Koopman operator. Our globally linear models can be integrated into linear control methodologies to control the underlying nonlinear dynamics.

In 1958, the idea of mathematically modelling biological neurons and their synaptic connections to perform pattern recognition was introduced via the concept of the perceptron [16]. Since then, Artificial Neural Networks (ANNs) have received a great deal of attention due to the development of learning algorithms, such as backpropagation [18], as well as the increasing abundance of data. ANNs consist of many simple interconnected nonlinear systems with weights that are adjusted to improve performance in a process known as learning. Increasing the depth of the ANNs enhances their performance and rate of learning, resulting in the hierarchical methods being termed deep learning. By the universal approximation theorem [4], ANNs are able to approximate arbitrarily complex functions, holding the potential for finding useful observables to produce superior representations of the Koopman operator. In this work we present an neural network architecture inspired by Koopman operator theory that approximates Koopman eigenfunctions and achieves improved prediction and control for nonlinear systems. Our model inherently decouples the Koopman modes of the system, performing a spectral decomposition of the underlying dynamics.

Chapter 2

Koopman Operator Theory

For some state space M we consider a nonlinear discrete-time *uncontrolled* dynamical system

$$\boldsymbol{x}^+ = T(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{x} \in M$ is the state of the system and $T : M \to M$ is the state transition mapping between successive time steps. The Koopman operator U is an infinite-dimensional operator that describes the evolution of all scalar-valued observables $h : M \to \mathbb{C}$ and is defined by

$$Uh(\boldsymbol{x}) = h \circ T(\boldsymbol{x}). \tag{2.2}$$

Importantly, the Koopman operator is globally linear even if the underlying dynamical system is highly nonlinear. This representation has the potential to improve nonlinear prediction and control by leveraging linear systems theory.

2.1 Numerical Methods - EDMD

A finite-dimensional approximation of the Koopman operator can be applied to a subset of observables to approximately predict the evolution of the underlying nonlinear dynamics. Unfortunately, obtaining an approximation of the Koopman operator that sufficiently captures the dynamics has proven to be difficult. Data-driven algorithms such as DMD and its variants are often used to find finite-dimensional representations of the Koopman operator. In particular, EDMD enhances the performance of DMD by providing the model with nonlinear functions of the measurements. The algorithm assumes there exists data in the form of tuples $(\boldsymbol{x}_i,\ \boldsymbol{x}_i^+),\ i = 0, 1, \ldots, N$ that satisfy $\boldsymbol{x}_i^+ = T(\boldsymbol{x}_i)$ and finds the optimal matrix A in a least-squares sense by minimizing

$$\sum_{i=0}^{N} \left\| f(\boldsymbol{x}_i^+) - Af(\boldsymbol{x}_i) \right\|_2^2 \tag{2.3}$$

where f is a vector lifted observable-functions. The observables must be chosen in a meaningful way to produce an adequate approximation of the Koopman operator.

2.2 Koopman Eigenfunctions

For systems with a discrete Koopman spectrum, the ideal choice for observables are the eigenfunctions of the Koopman operator ϕ, defined as

$$U\phi(\boldsymbol{x}) = \lambda\phi(\boldsymbol{x}). \tag{2.4}$$

The eigenfunctions span a Koopman invariant subspace [12], and lead to an exact finite-dimensional linear representation. The difficulty lies in choosing observables that approximately span this Koopman invariant subspace. Observable dictionaries consisting

of radial basis functions, monomials, and time delays have shown limited success [8]. The search for observables that approximate Koopman eigenfunctions motivates the use of deep learning methodologies.

Chapter 3

Learning Discrete Spectrum

Linear systems can be diagonalized to decouple the dynamics and to study the spectral properties. Representing nonlinear systems in the Koopman framework suggests a similar reduction [12]. For systems with a discrete Koopman spectrum, there exist a finite number of Koopman eigenfunctions that are able to sufficiently represent the underlying nonlinear dynamics. Lifting the original states into eigenfunction coordinates results in a globally diagonal representation that decouples the eigenfunction-eigenvalue pairs. This is the equivalent of performing an eigendecomposition for a linear system. Typically, observables used in EDMD are user-specified and are unable to sufficiently represent the dynamics. We show that our deep learning model finds superior observables by approximating Koopman eigenfunctions from data. By doing so, our model is able to generate predictions superior to existing models.

3.1 Network Architecture

We leverage deep learning to learn eigenfunctions of the Koopman operator from data. Importantly, we constrain our model to evolve the eigenfunctions through scalar multipli-

cation by an eigenvalue to find a linear representation in the lifted space of eigenfunctions. In order to generalize to systems with oscillatory behaviors, we equip our model to be able to learn complex-valued eigenfunctions and eigenvalues. We use two sets of real-valued weights to hold the complex quantities. The first set of weights corresponds to the real components while the second set corresponds to the imaginary components. Because all the operations can be reduced to scalar additions and multiplications, the resulting complex quantities can be computed according to the relationships

$$\begin{aligned}(a+bi)+(c+di) &= (a+c)+(b+d)i \\ (a+bi)(c+di) &= (ac-bd)+(ad+bc)i.\end{aligned} \tag{3.1}$$

To learn the dynamics, our model is trained on data in the form of trajectories produced from a set of initial conditions. For every trajectory, the loss functions used in training are:

1. Reconstruction Loss. This loss enforces that the model behaves as an autoencoder, where f is an encoder and f^{-1} is a decoder. The encoder lifts the states to a space of eigenfunctions and the decoder unlifts the eigenfunctions back to the original state space. Neural networks are used for the encoder and the decoder.

$$\left\|\boldsymbol{x}_0 - f^{-1}(f(\boldsymbol{x}_0))\right\|_{\text{MSE}} \tag{3.2}$$

2. Prediction Loss. This loss drives the network to learn eigenfunctions that can sufficiently predict the nonlinear evolution of the states. The prediction loss is enforced over n time steps that can capture the essential characteristics of the dynamics. By taking the prediction loss over multiple time steps, the compounded error of the repeated multiplication of the eigenvalues are taken into account to produce eigenfunctions well

suited for long-term predictions. A neural network is used for the diagonal matrix D of eigenvalues.

$$\frac{1}{n}\sum_{k=1}^{n}\left\|\boldsymbol{x}_k - f^{-1}(D^k \circ f(\boldsymbol{x}_0))\right\|_{\text{MSE}} \tag{3.3}$$

Our autoencoder-based neural network model shown in Figure 3.1 is able to discover a set of nonlinear observables on which the dynamics evolve linearly. The encoder lifts the original states into the space of eigenfunctions and the decoder recovers the original states. As opposed to enforcing linearity in the cost function [10], our network is constrained to only evolve in time by repeated multiplication by eigenvalues. Thus, the architecture constrains the model to approximate eigenfunctions and eigenvalues while sufficiently representing the dynamics. Importantly, a diagonal representation of the dynamics is obtained that decouples the principle characteristics of the nonlinear dynamics. Our models takes the form of

$$\begin{bmatrix} z_1' \\ z_2' \\ \vdots \\ z_N' \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}, \tag{3.4}$$

where $z_i = f_i(\boldsymbol{x})$ are the learned eigenfunctions and λ_i are the corresponding eigenvalues for $i = 1, \ldots, N$. We use the Pytorch framework [15] and the Adam optimizer [5] for all training purposes. The neural networks used are composed of 2 hidden layers each with 20 neurons. Each hidden layer in our model is followed by a rectified linear unit (ReLU) nonlinear activation function.

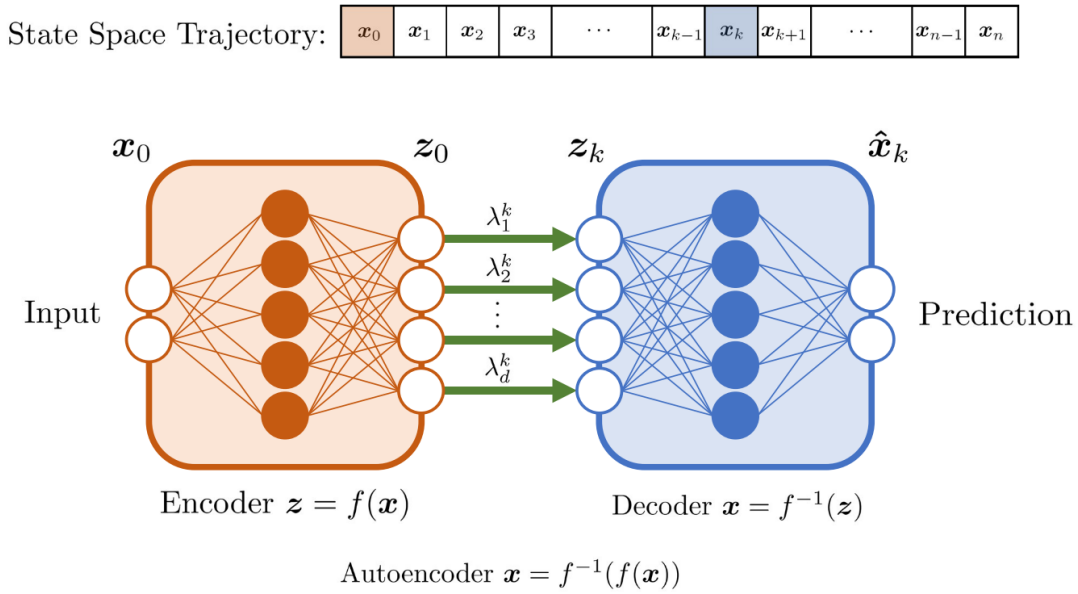

Figure 3.1: Visualization of our deep learning architecture used for approximating Koopman eigenfunctions and eigenvalues. The model is based on an autoencoder that is able to reconstruct the original inputs through a latent variable space we call the eigenfunction space. In addition to the autoencoder loss, we minimize the prediction error so that the model learns to predict the the dynamics. The green arrows represent repeated multiplication of each eigenfunction by its associated eigenvalue. These eigenvalues are also weights that are learned during training. To obtain an eigenfunction k steps in the future, the current eigenfunction must be multiplied by its eigenvalue k times. k must be chosen to sufficiently capture the dynamics of interest. The network is trained by minimizing the prediction error of the states at all future steps up to a desired time horizon.

3.2 Initialization with EDMD

We initialize the model with approximations of the eigenvalues and eigenfunctions of the Koopman operator obtained from performing EDMD on time delay observables. Although a better EDMD model can be constructed using more carefully chosen observables, the time delay observables can be readily obtained from the training data. The eigenvalues of the model are directly initialized with the eigenvalues acquired from EDMD, while the encoder and decoder are trained to represent the numerical eigenfunc-

tions produced by EDMD. Training the network intialized in this way allows the networks to start off with a better representation of the dynamics and decreases the amount of training time necessary.

Chapter 4

Uncontrolled Numerical Examples

We demonstrate our deep learning model's ability to approximate Koopman eigenfunctions on example systems with discrete real and complex spectra. For a system with complex spectra, we consider the Van der Pol oscillator which exhibits a highly nonlinear limit cycle.

4.1 Discrete Real Spectrum

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

$$\begin{aligned} \dot{x}_1 &= \alpha x_1 \\ \dot{x}_2 &= \beta(x_2 - x_1^2) \end{aligned} \tag{4.1}$$

For this simple system, it is not difficult to determine the eigenfunctions ϕ_1 and ϕ_2:

$$\begin{aligned} \phi_1 &= x_1 \\ \phi_2 &= x_2 - \frac{\beta}{\beta - 2\alpha} x_1^2 \end{aligned} \tag{4.2}$$

with corresponding eigenvalues $\lambda_1 = e^{\alpha \Delta t}$ and $\lambda_2 = e^{\beta \Delta t}$. For our choice of $\alpha = -0.05$ and $\beta = -1$, there is a stable manifold at $x_2 = x_1^2$. We create our datasets by solving the systems of differential equations in (4.1) using a Runge-Kutta (4, 5) solver. 200,000 initial conditions are generated randomly from $[-0.5, 0.5] \times [-0.5, 0.5]$ and solved for $k = 256$ time steps of $\Delta t = 0.1$. 10% of the data are used for validation and the rest are used for training. The discrete eigenvalues corresponding to $\alpha = -0.05$ and $\beta = -1$ are determined to be:

$$\lambda_1 = e^{\alpha \Delta t} = e^{(-0.05)(0.1)} \approx 0.99501 \tag{4.3}$$

$$\lambda_2 = e^{\beta \Delta t} = e^{(-1)(0.1)} \approx 0.90484 \tag{4.4}$$

4.1.1 Learning Real Eigenfunctions

We expect a model with two eigenfunctions to learn the eigenfunctions and eigenvalues described in (4.2) in order to adequately predict the dynamics. Because this system possesses purely real eigenvalues and eigenfunctions, our model can be constrained to use purely real values. Before training the model on the dynamics, we use DMD to initialize the weights and the eigenvalues. The results of applying DMD for initialization are shown in Figure 4.1. Although the eigenvalues produced by DMD are very close to the expected values, the corresponding eigenfunctions do not capture the nonlinearity due to the poor choice of observables. After training, our model learns the eigenfunctions of the nonlinear dynamics that are qualitatively similar to the expected analytical eigenfunctions.

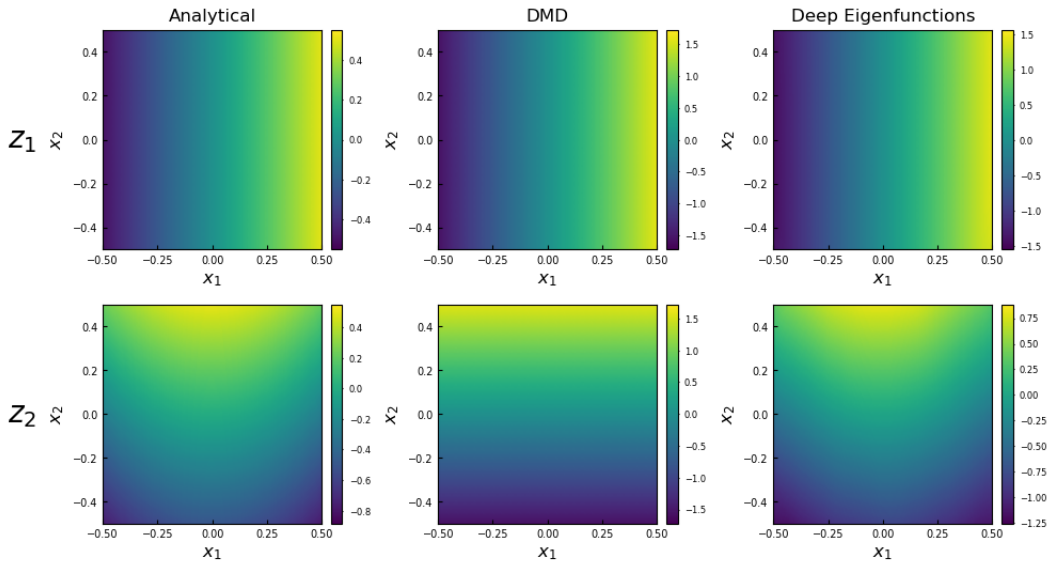

Figure 4.1: Left - Analytic eigenfunctions in (4.2) visualized over the state space. Middle - Numerical eigenfunctions computed using DMD. The eigenvalues are learned to be 0.99501 *and* 0.90507*. Right - Numerical eigenfunctions produced using our method. The eigenvalues are learned to be* $0.99503 + 0.00013i$ *and* $0.90217 - 0.00990i$*. Note that Koopman eigenfunctions are invariant to scaling.*

The state space can be partitioned into level sets of ϕ_2 where the dynamics rapidly converge to $\phi_2 = 0$ at the rate of λ_2. The slow manifold shown in Figure 4.2 is the zero level set of ϕ_2. After the slow manifold has been reached, the dynamics slowly converge to the origin at the rate of λ_1 while passing through level sets of ϕ_1.

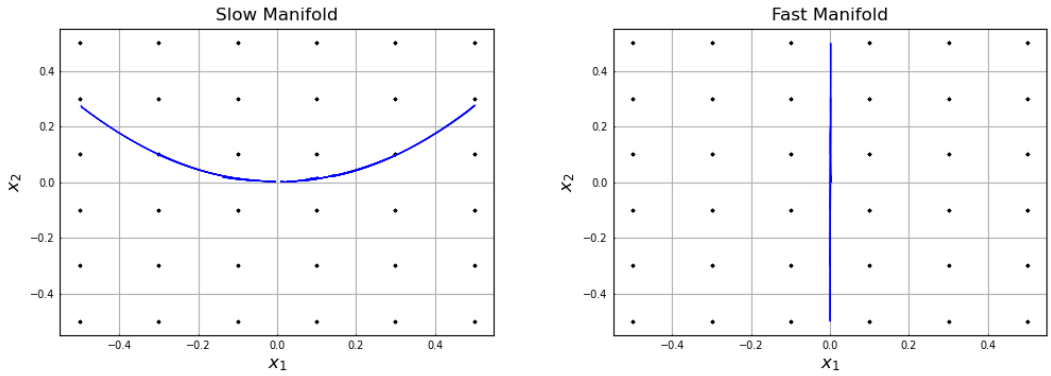

Figure 4.2: Zero-level sets of the eigenfunctions are the slow and fast manifolds.

Given initial conditions within the training bounds, our model is able to forecast trajectories for long time horizons shown in Figure 4.3 using a compact 2×2 diagonal matrix representation:

$$\begin{bmatrix} \phi_1' \\ \phi_2' \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}. \tag{4.5}$$

Our model is able to identify the nonlinear slow manifold at $x_2 = x_1^2$ that DMD is unable to recognize. The eigenvalues change very little during training, indicating that a good initialization was given that was able to accelerate the learning process.

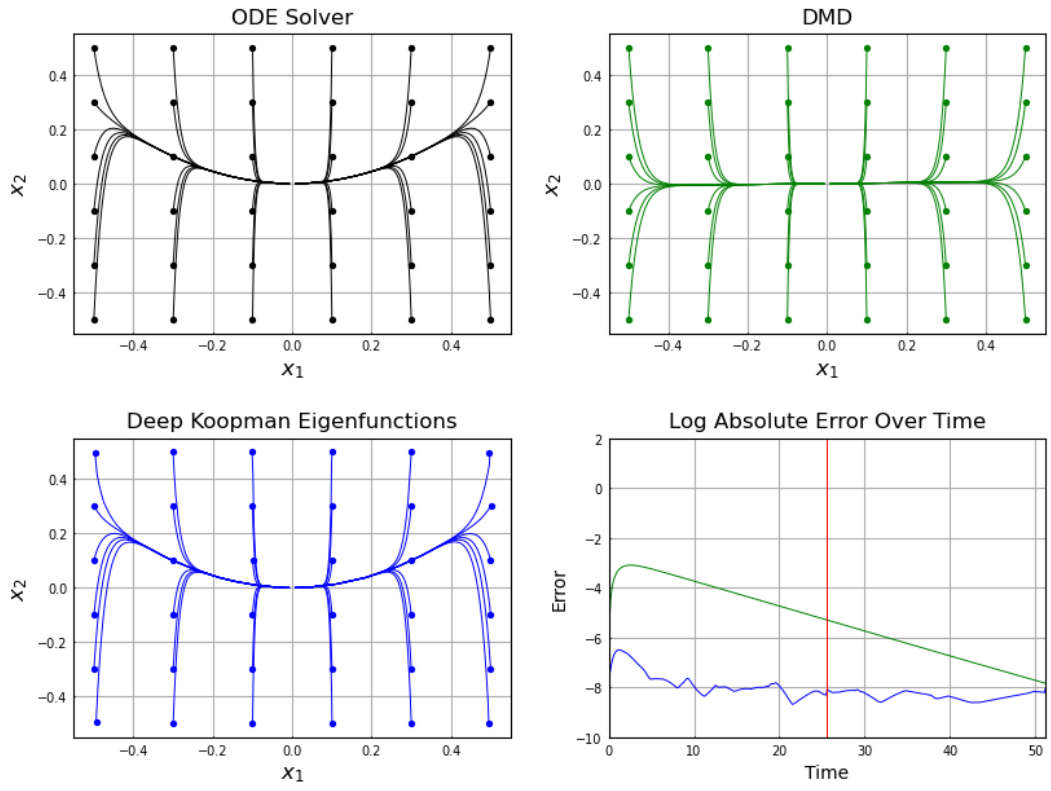

Figure 4.3: Trajectories of 512 steps generated from a 6×6 grid of initial conditions. Black - Trajectories generated using an ODE solver on the nonlinear equations in (4.1). Green - Results from using a 2×2 linear model produced by DMD. Blue - Results from using our 2×2 diagonal matrix representation in (4.5). Red - Training horizon. Log absolute prediction error is averaged over the 36 trajectories.

4.1.2 Redundant Eigenfunctions

Applying a model with redundant eigenfunctions, we expect to learn a set of eigenfunctions generated by the two eigenfunctions in (4.2). The results of training the network to learn sixteen eigenfunctions are shown in Figure 4.4. Considering the ℓ^2 norm of the eigenfunctions over the region of interest, we can see that three eigenfunctions have a magnitude above our cutoff threshold of $\epsilon = 0.1$.

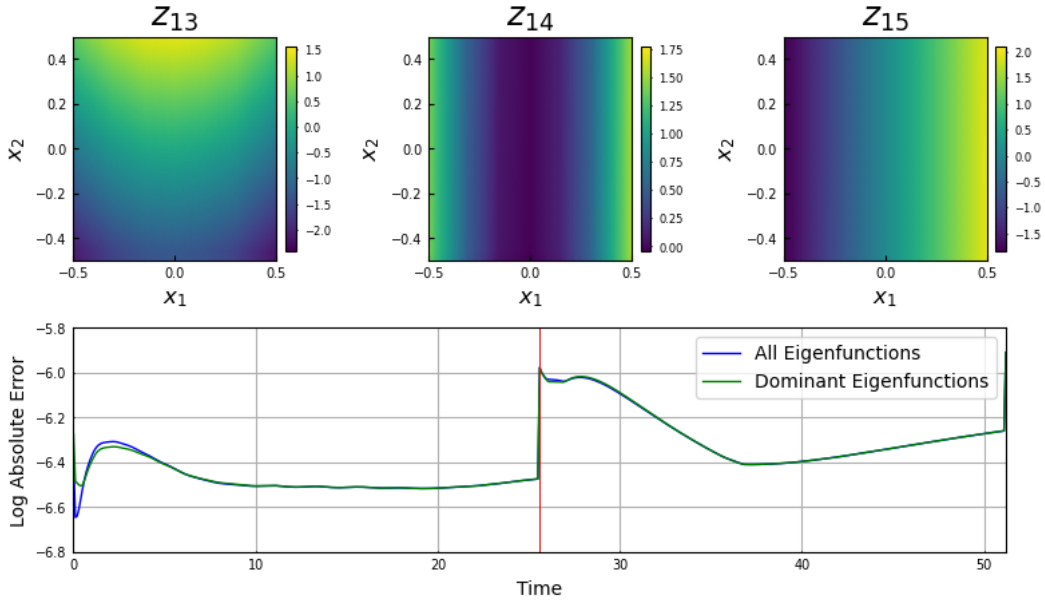

Figure 4.4: Results from learning redundant eigenvalues. Although most of the eigenvalues jump around during training, three eigenvalues remain very close to their initialization values. It turns our that these three eigenfunctions also have a significantly larger ℓ^2 norm and can completely capture the dynamics. Top - The 15th and 13th eigenfunctions appear similar to the expected eigenfunctions. They have corresponding eigenvalues at $0.99505 + 0.00011i$ and $0.90459 + 0.00641i$. We also find the 14th eigenfunction to have a significant magnitude and a corresponding eigenvalue at $0.98973 + 0.00203i$. The eigenfunction appears to be x_1^2, which has an expected eigenvalue at $e^{2\mu\Delta t} = e^{2(-0.05)(0.1)} \approx 0.99005$. Bottom - Average log absolute prediction error over a 6×6 grid of initial conditions. Using only the three dominant eigenfunctions results in a very close result to using all sixteen eigenfunctions.

Not surprisingly, the three corresponding eigenvalues move very little relative to their initialization compared to the other thirteen during training. Using only these three

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

4.2 Van der Pol Oscillator

Next, we consider the Van der Pol oscillator. This is a system with complex eigenvalues that exhibits a limit cycle.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -0.5(x_1^2 - 1)x_2 - x_1 \end{aligned} \tag{4.6}$$

We expect that two eigenfunctions should sufficiently capture the general behavior of the Van der Pol system. One complex eigenfunction is expected due to the periodic motion of the limit cycle, and a second eigenfunction is necessary to describe the stability of the limit cycle. The level sets of these eigenfunctions are the isochrons and the isostables of the system [11]. For some given phase θ, the isochron consists of all points such that the trajectory through such points at $t = 0$ asymptotically approaches the trajectory that belongs to a point on the limit cycle with phase θ at $t = 0$. From the period of the limit cycle ≈ 6.3807, we expect a corresponding discrete time eigenvalue at:

$$e^{(\frac{2\pi i}{6.3807})(0.1)} \approx 0.99518 \pm 0.09802i \tag{4.7}$$

Performing the computation of the so-called Fourier average

$$f_\omega^*(\boldsymbol{x}) = \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} (f \circ T^j)(\boldsymbol{x}) e^{-i\omega j} \tag{4.8}$$

for the observable $f(x_1, x_2) = x_1 + x_2$ with the period of the limit cycle results in the isochrons f^*_ω on the Van der Pol oscillator visualized in Figure 4.5. On the other hand, isostables consist of all the points that share the same asymptotic convergence toward a fixed point. A second eigenvalue corresponds to the stability of the limit cycle and can be computed numerically. Taking a surface of section, we can see that the trajectory exponentially approaches the limit cycle at a rate of decay of ≈ -0.5 with a corresponding discrete eigenvalue at:

$$e^{(-0.5)(0.1)} \approx 0.9512 \tag{4.9}$$

We create our datasets by solving the systems of differential equations in (4.6) using a Runge-Kutta (4, 5) solver. 200,000 initial conditions are generated randomly from $[-3.0, 3.0] \times [-3.0, 3.0]$ and solved for k = 256 time steps of $\Delta t = 0.1$. 10% of the data is used for validation and the rest is used for training.

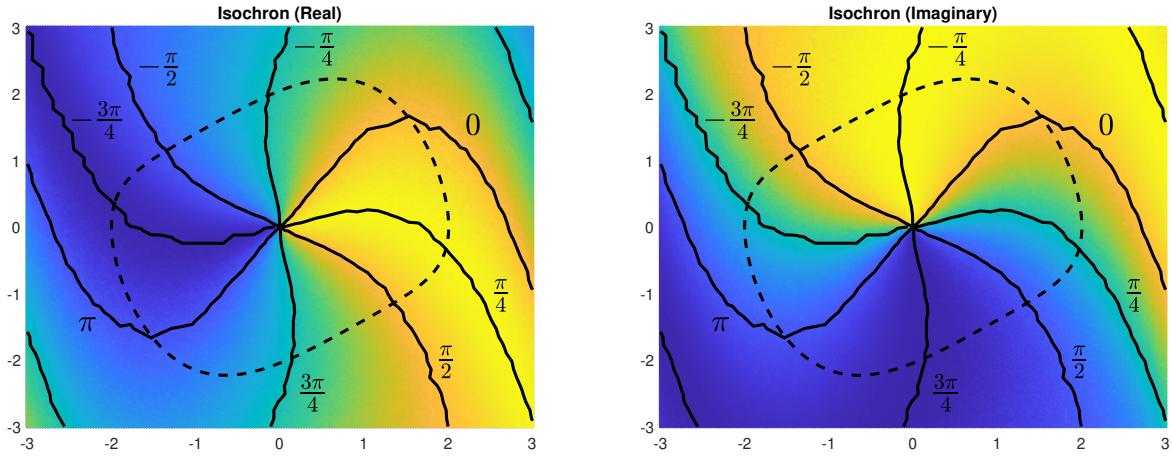

Figure 4.5: Isochrons of the Van der Pol oscillator shown as level sets of a complex eigenfunction. The dashed line represents the limit cycle and the solid lines represent level sets of the argument $\angle f$.

4.2.1 Complex Eigenfunctions

The results of using DMD to initialize the model for the Van der Pol system are shown in Figure 4.6. DMD produces a complex conjugate pair of eigenvalues with approximately the expected periodicity. However, the original coordinates are poor choices of observables and the resulting model from DMD completely fails to capture the presence of the limit cycle. The results for learning two eigenfunctions are shown in Figure 4.6. The first eigenvalue is learned to be associated with the periodic motion of the system and approaches the expected value in (4.7). The zero-level set of the eigenfunction is the limit cycle itself, shown in Figure 4.7.

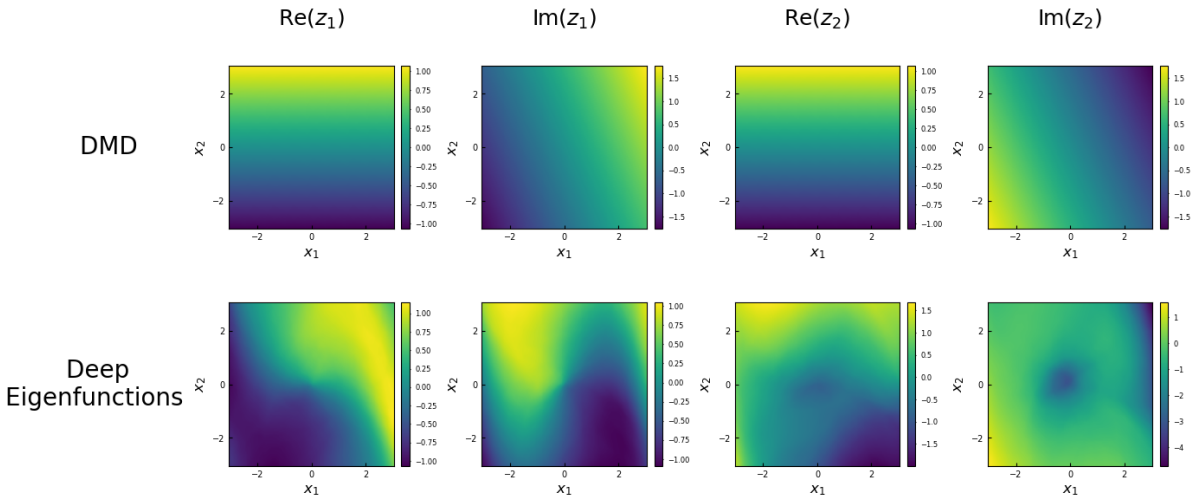

Figure 4.6: Top - Numerical eigenfunctions computed using DMD. The eigenvalues are learned to be $0.97052 \pm 0.09703i$*. Bottom - Numerical eigenfunctions produced using our method. The eigenvalues are learned to be* $0.99518 - 0.09831i$ *and* $0.95168 - 0.00024i$*. Note that Koopman eigenfunctions are invariant to scaling.*

The second eigenvalue is associated with the stability of the limit cycle and approaches the expected value in (4.9). Notably, the level sets of the first eigenfunction learned by our neural network model highly resembles the isochrons shown in 4.5.

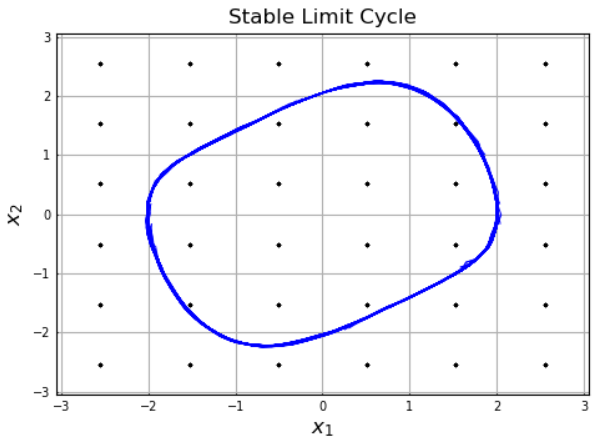

Figure 4.7: Limit cycle of the Van der Pol oscillator can be obtained as the zero-level set of the eigenfunction associated with the periodic motion of the system .

In Figure 4.8, our 2×2 diagonal linear method in (4.5) is compared to a 2×2 linear model produced by DMD. Our linear model is able to successfully predict trajectories that traverse the limit cycle for multiple periods. EDMD involves lifting into higher-dimensions to produce improved approximations of the Koopman operator. These methods produce large sparse matrices that exhibit relatively poor long-term predictive performance. In contrast, our compact diagonal representation (blue) is shown in Figure 4.9 to have superior long-term predictive performance, even when compared to larger representations that depend on user-defined observables. In particular, we compare our model to the predictors developed in [8] with 100 thin plate spline radial basis functions as the lifting functions. We also compare our results to a Hankel-DMD representation built on 10 time-delays. We find diminishing returns when constructing a model using more than 10 time-delays.

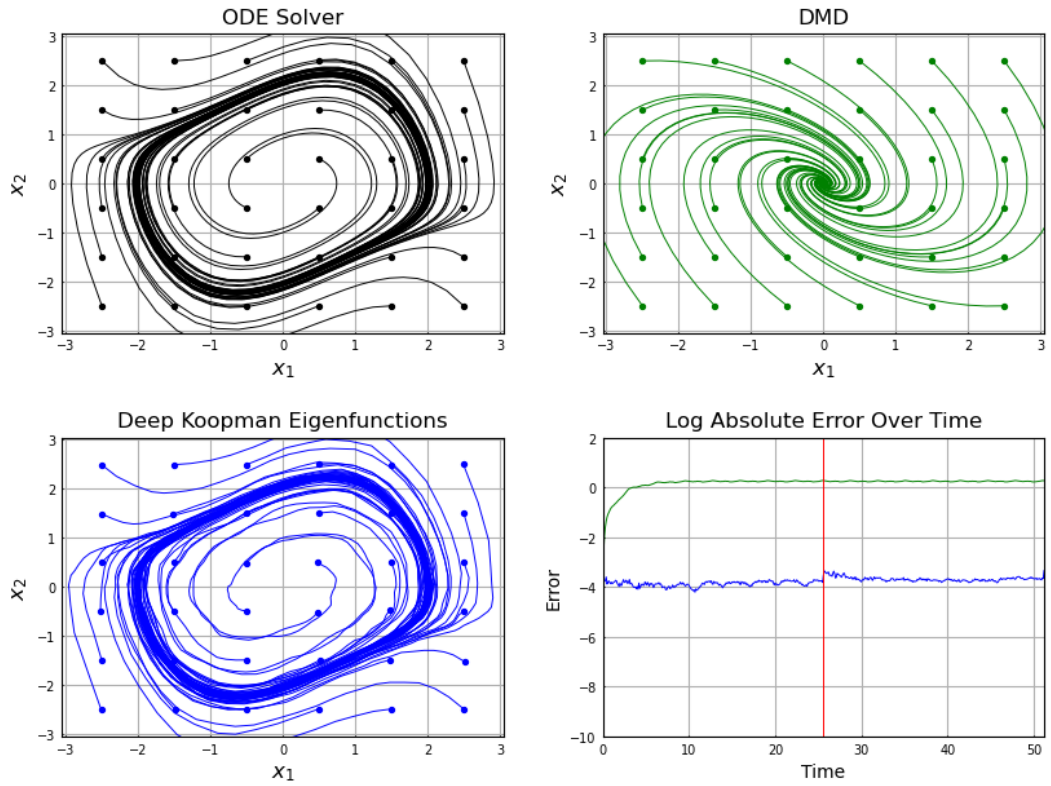

Figure 4.8: Trajectories of 512 steps generated from a 6×6 grid of initial conditions. Black - Trajectories generated using an ODE solver on the nonlinear equations in (4.6). Green - Results from using a 2×2 linear model produced by DMD. Blue - Results from using our 2×2 diagonal matrix representation. Red - Training horizon. Log absolute prediction error is averaged over the 36 trajectories.

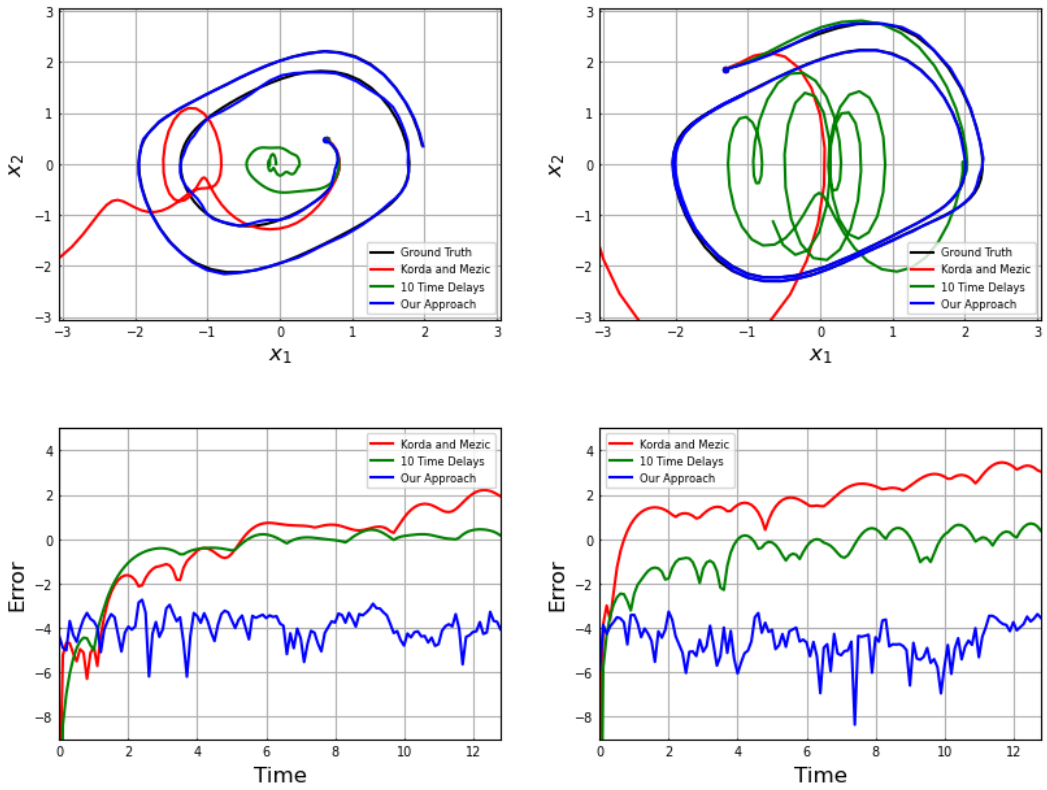

Figure 4.9: Trajectories of 128 steps generated from two initial conditions. Left - Predictions given an initial condition inside the limit cycle. Right - Predictions given an initial condition outside the limit cycle. Black - Trajectories generated using an ODE solver on the nonlinear equations in (4.6). Red - Results from using a 102×102 *linear predictor developed in [8]. Green - Results from using a* 22×22 *linear model produced by Hankel-DMD with 10 time-delays. Blue - Results from using our* 2×2 *diagonal matrix representation.*

Chapter 5

Koopman Modal Control

A key motivation for developing a globally linear model for nonlinear dynamics is the application of linear control theory. For some state space M and set of admissible inputs $\mathcal{U}$, we now consider a nonlinear discrete-time *controlled* dynamical system

$$\boldsymbol{x}^+ = S(\boldsymbol{x}, \boldsymbol{u}), \tag{5.1}$$

where $\boldsymbol{x} \in M$ is the state of the system, $\boldsymbol{u} \in \mathcal{U}$ is the control input, and $S : M \times \mathcal{U} \to M$ is the mapping between successive time steps. We would like to predict the trajectory of the controlled dynamical system given an initial condition $\boldsymbol{x}_0$ and the control input signal $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \dots\}$. A linear representation which sufficiently captures the nonlinear dynamics can be used to produce an optimal linear controller that is close to the optimal controller for the nonlinear dynamics. Importantly, due to the uncoupling of the Koopman modes, the different modes within the system can be directly controlled.

5.1 Linear Predictors

We are interested in predictors that possess a structure that can be integrated into linear control frameworks. Similar to the model used to learn the Koopman eigenfunctions, a second autoencoder model $\boldsymbol{u} = g^{-1}(g(\boldsymbol{u}))$ can be used to learn the appropriate lifting on the inputs in order to produce the optimal linear predictor in the space of the learned eigenfunctions. We assume a linear predictor that takes the form of a linear dynamical system

$$\boldsymbol{z}^{+} = A\boldsymbol{z} + \boldsymbol{v}, \tag{5.2}$$

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

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

form

$$\begin{bmatrix} z_1' \\ z_2' \\ \vdots \\ z_N' \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}, \tag{5.3}$$

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

5.2 Closed-Loop Control

The ultimate goal of developing an accurate model is the prospect of designing a controller so that the system dynamics behave as desired. Because our linear model acts on the lifted space of eigenfunctions, linear methodologies produce a control law for the lifted inputs $\boldsymbol{v}$. Fortunately, because an autoencoder structure was used for lifting the inputs, we have trained a mapping g^{-1} from the lifted space of inputs back to the original input space. We use the control law

$$\boldsymbol{v} = -K\boldsymbol{z} \quad \Longrightarrow \quad \boldsymbol{u} = g^{-1}(-Kf(\boldsymbol{x})) \tag{5.4}$$

as feedback into the system given in (5.1), where K represents some feedback gains computed by optimal control algorithms such as LQR. Evaluation of the deep learning model for lifting the states and unlifting the inputs can be rapidly performed, and this framework can be integrated for real-time control.

Chapter 6

Controlled Numerical Examples

We use deep learning to find linear predictors on the space of the approximate eigenfunctions and use linear control methodologies to perform control objectives.

6.1 Discrete Real Spectrum with Input Forcing

We consider the system in (4.1) modified by the addition of a control effort u.

$$\begin{aligned} \dot{x_1} &= \alpha x_1 \\ \dot{x_2} &= \beta(x_2 - x_1^2) + u. \end{aligned} \tag{6.1}$$

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

$$\begin{bmatrix} \phi_1' \\ \phi_2' \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{6.2}$$

Starting at some arbitrary initial condition, Figure 6.1 shows the prediction generated using a random input signal is compared to the effect of that signal on the actual dynamics.

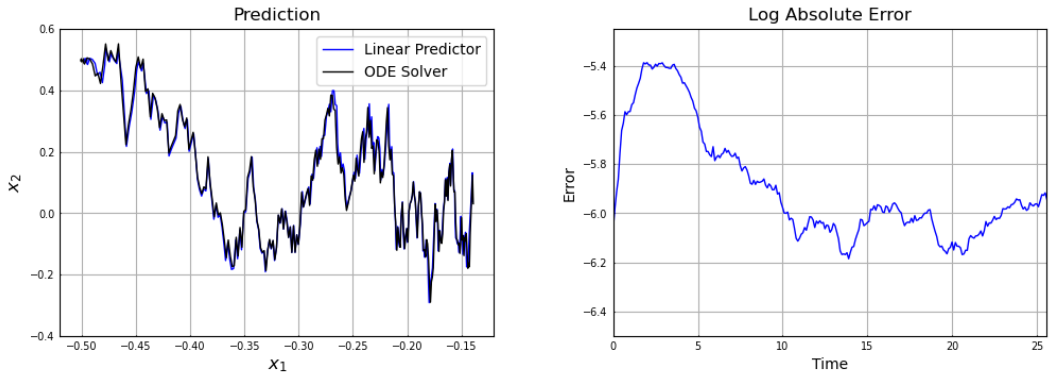

Figure 6.1: Results from learning to predict the discrete spectrum system with random inputs. A random input signal is generated that was not used in the training of the model, and the trajectory of the linear predictor is compared with the trajectory of the dynamics.

Now that a linear predictor has been created for the controlled dynamical system, we can apply feedback control. Here, our control objective is to achieve a specific rate of convergence. This can be easily done by enforcing an eigenvalue in our model using feedback control. Because the two lifted states in this system are decoupled and the input is only applied in the x_2 direction, only ϕ_2 is controllable. Figure 6.2 shows the results of the control effort applied to the original nonlinear dynamics. By learning the Koopman eigenfunctions of the system, the level sets, including the slow manifold, are preserved. Because of the high performance of the linear predictor, we are able to control the system far from the original dynamics.

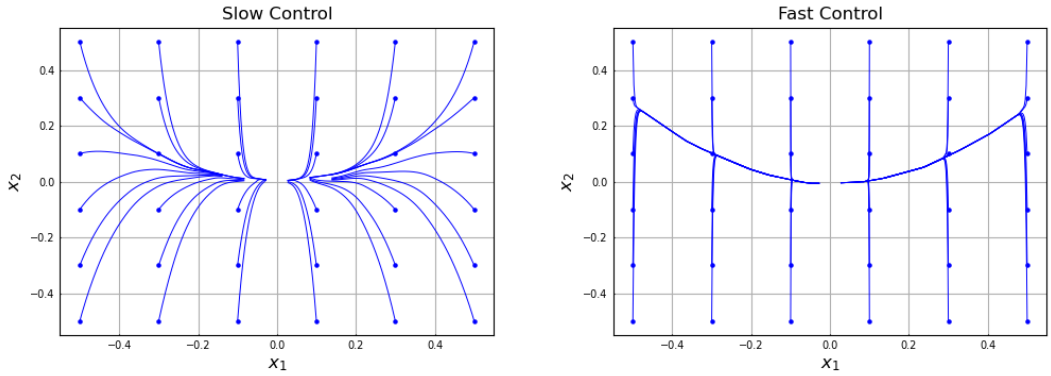

Figure 6.2: Results from applying a control effort to decrease and increase the rate of decay of ϕ_2. Left - Control with eigenvalue 0.999*. Right - Control with eigenvalue* 0.2*.*

6.2 Van der Pol with Input Forcing

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

$$\dot{x_1} = x_2 + u_1 \tag{6.3}$$

$$\dot{x_2} = -0.5(x_1^2 - 1)x_2 - x_1 + u_2. \tag{6.4}$$

Similar to the discrete real spectrum system, we simulate trajectories using random input signals with uniform distribution over the interval $[-1, 1] \times [-1, 1]$ to learn the lifted-controlled system in (6.2). However, because the Van der Pol system exhibits periodic behavior, the assumption that the approximated eigenfunctions are linearly dependent on the inputs becomes a dangerous assumption. The lack of linear dependence results in a poor linear predictor that fails to capture the controlled dynamics. Figure 6.3 shows the prediction generated using a random input signal compared to the effect of that signal on the actual dynamics. Because the linear model is poor, the errors compound and the model quickly diverges.

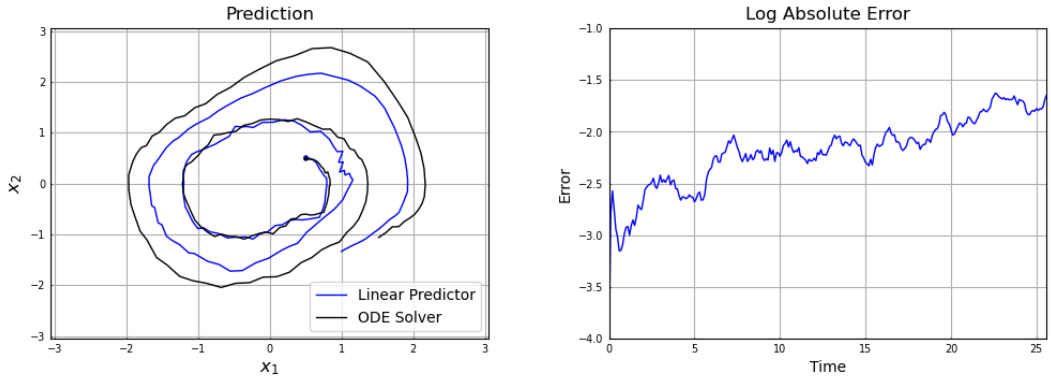

Figure 6.3: Results from learning the predict the Van der Pol system with random inputs. A random input signal is generated that was not used in the training of the model, and the trajectory of the linear predictor is compared with the trajectory of the dynamics.

Although the linear predictor has poor long-term predictions, the model is sufficient for controlling the Koopman eigenvalues and eigenfunctions using finite time-horizon methods. Our control objective is to achieve a specific periodicity on the limit cycle. This can be easily done by enforcing an eigenvalue with unit magnitude associated with the periodic motion of the system. Figure 6.4 shows the results of the control effort applied to the original nonlinear dynamics. By learning the Koopman eigenfunctions of the Van der Pol system, the isostables and isochrons can be preserved. This allows the shape of the limit cycle to be retained while applying some control effort. Due to the poor quality of the linear predictor shown in 6.3, it is expected that applying control to the system can affect the shape of the limit cycle. Controlling the period of the limit cycle near the original period reduces the distortion of the attractor.

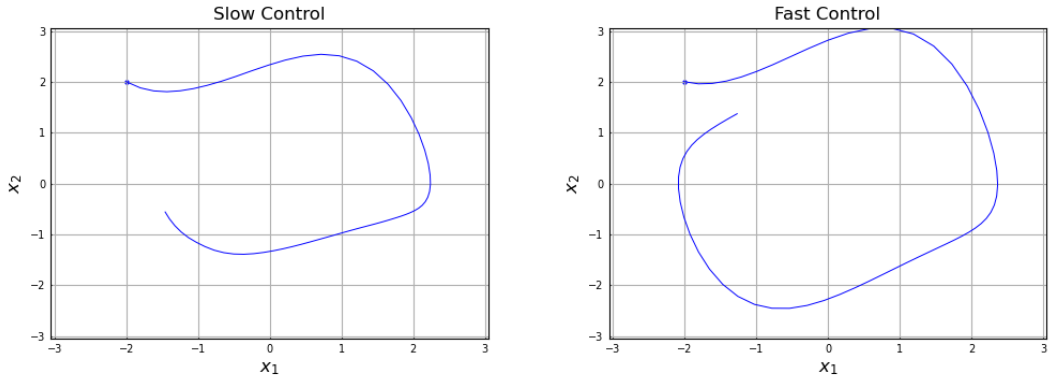

Figure 6.4: Results from applying control effort to decrease and increase the period of the limit cycle. Left - Control with eigenvalue $0.998 - 0.06321js$. *Right - Control with eigenvalue* $0.994 - 0.10938i$.

We consider using a bilinear representation as a future direction. Our current model experiences difficulties representing *controlled* dynamical systems. This is likely due to the fact that the eigenfunctions produced by our model are not linearly dependent on some function of the inputs in general. Providing a bilinear term that takes the current state in addition to the current input into account should improve the performance of our linear predictor.

Chapter 7

Conclusions and Outlook

A method is described that leverages deep learning to approximate Koopman eigenfunctions for arbitrary nonlinear dynamical systems with discrete spectrum. The underlying idea is a nonlinear lifting of the dynamics to a space of eigenfunctions where the evolution is exactly linear. The lifted linear systems found using this framework exhibit superior prediction compared to traditional methods that suffer from compounding errors over the prediction horizon. Lifting the inputs to the same space, linear predictors for controlled dynamical systems are obtained that can be readily applied to linear control design frameworks. This diagonal linear representation enables individual control of Koopman modes.

Our method exhibits superior performance compared to traditional data-driven methods that seek to find an approximately linear evolution on a user-defined lifted space. EDMD approximates the Koopman operator based on the error of a single time step, which results in poor long-term performance due to the significance of compounding errors. Our method produces superior long-term predictions by performing optimization over a user-defined time horizon. In addition, this method creates a compact low-dimensional linear representation by extracting the key dynamics instead of finding the dynamics on a much

higher-dimensional lifted space. Furthermore, by decoupling the Koopman modes, the spectral properties can be quickly interpreted and certain structures (such as steady-state limit cycles) can be readily identified.

In constructing the framework for the controlled linear predictor, we assumed that the approximated eigenfunctions are linearly dependent on some lifting of the inputs. However, this linear dependence does not hold in general. For systems with complex dynamics like the Van der Pol oscillator's limit cycle, the values of the inputs alone are inadequate to predict the effect on the dynamics. In general, information about the current state as well as the inputs is necessary. Future work should focus on developing a bilinear predictor that can provide improved predictions and control. More work needs to be done to generalize the architecture to systems with continuous Koopman spectra.

Bibliography

[1] Hassan Arbabi and Igor Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, jan 2017.

[2] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems. *ArXiV Preprint*, February 2021.

[3] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied koopmanism. 22(4):047510, dec 2012.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. 2(4):303–314, dec 1989.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, may 1931.

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

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

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

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

[11] A. Mauroy, I. Mezić, and J. Moehlis. Isostables, isochrons, and koopman spectrum for the action–angle representation of stable fixed point dynamics. 261:19–30, oct 2013.

[12] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, aug 2005.

[13] Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. 45(1):357–378, jan 2013.

[14] Igor Mezić and Andrzej Banaszuk. Comparison of systems with complex behavior. 197(1-2):101–133, oct 2004.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[16] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. 65(6):386–408, 1958.

[17] Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, nov 2009.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. 323(6088):533–536, oct 1986.

[19] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, jul 2010.

[20] Jonathan H. Tu, , Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz and. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

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

[22] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. *ArXiV Preprint*, August 2017.