Learning-Based Stability Certification and System Identification of Nonlinear Dynamical Systems

by

Ruikun Zhou

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Sze Zheng Yong Associate Professor Department of Mechanical and Industrial Engineering Northeastern University
Supervisor:	Jun Liu Professor, Department of Applied Mathematics
Internal Members:	Xinzhi Liu Professor, Department of Applied Mathematics
	Kirsten Morris Professor, Department of Applied Mathematics
Internal-External Member:	Jeff Orchard Associate Professor David R. Cheriton School of Computer Science

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The research presented in this thesis has resulted in several publications and submissions. The contributions of the author to each chapter are detailed below:

1. For Chapters 1 and 2, I am the sole author.

2. Chapter 3 is based on the paper published in *IEEE Control Systems Letters* [148], in which I led the conceptualization, methodology, algorithm design, numerical results and writing.

3. Chapter 4 is based on the work [150] published in *Advances in Neural Information Processing Systems (NeurIPS 2022)*. I contributed to all aspects except the theoretical developments of the neural approximation of unknown dynamics and Lyapunov functions in Section 4.4, which were carried out by my co-author, Thanin Quartz.

4. Chapter 5 is based on two papers: the conference version in [93] and an extended journal version currently under review at the *IEEE Transactions on Automatic Control*, for which a preprint is available in [94]. In this project, my contributions are: conceptualization, investigation, algorithm design, numerical results, and writing.

5. Chapter 6 is based on the work [149] that has been accepted for publication in the *Proceedings of the 2025 IEEE 64th Conference on Decision and Control (CDC)*, where I was responsible for all aspects, with limited co-author contributions to the conceptualization and assistance with writing and proofreading.

Abstract

In recent decades, by taking advantage of the abundance of sensory measurements, learning-based methods have been prevalent and shown their effectiveness in tackling challenging or intractable problems for classical approaches in systems and control. For instance, many systems with complex nonlinearities, high-dimensional state spaces, or unknown dynamics cannot be effectively handled by classical mathematical tools, and computing stability certifications for such systems is often intractable. This thesis aims to construct systematic approaches to perform system identification tasks and learning-based Lyapunov functions for nonlinear dynamical systems, with some extensions to optimal control.

The first aspect of this thesis is to develop an efficient method based on a special feedforward neural network structure, an extreme learning machine, to compute stability certificates for nonlinear systems by solving linear partial differential equations (PDEs), when the dynamics are accessible. Differing from the typical neural network-based approaches that require training on high-performance computing platforms, one only needs to solve a convex optimization problem. On top of that, the proposed method can also be used to efficiently solve the notable Hamilton-Jacobi-Bellman (HJB) equation via policy iteration to obtain optimal control policies for nonlinear systems.

The second aspect of this research is to tackle these issues for nonlinear systems with (partially) unknown dynamics. We first show that with two feedforward neural networks, the unknown system and a Lyapunov-based stability certificate can be learned simultaneously. With the help of satisfiability modulo theories (SMT) solvers, the resulting Lyapunov function can be formally verified to provide stability certificates for the unknown nonlinear system.

Alternatively, in the past two decades, the Koopman operator and its generator have demonstrated advantages in identifying discrete-time systems and continuous-time systems, respectively, requiring significantly less data while achieving better performance than most existing classical methods. For unknown continuous-time dynamical systems, we propose a novel resolvent operator-based learning framework to learn the Koopman generator, which is a linear operator that describes the infinitesimal evolution of the Koopman operator. The learned generator, thereafter, can be used to identify the vector field of the nonlinear systems. Moreover, with the learned high-accuracy Koopman generator, we can also construct a Lyapunov-based stability certificate for the unknown nonlinear system in the same function space. By formulating the linear PDEs as a linear least squares problem, Lyapunov functions can be computed efficiently. The learned Lyapunov functions can be formally verified using

an SMT solver and provide less conservative estimates of the region of attraction, compared to existing methods.

Taken together, these contributions provide a coherent pathway that begins with model-based stability certification computation and continues to fully data-driven system identification and thereafter computing Lyapunov-based stability certificates.

Numerical examples and simulations are presented throughout this thesis to illustrate the theoretical results and efficacy of the proposed algorithms.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Jun Liu, for his continuous support, guidance, and encouragement throughout my PhD journey. His profound knowledge and insightful feedback have been invaluable in shaping my research and academic growth.

I would also like to extend my heartfelt thanks to my committee members, Prof. Xinzhi Liu, Prof. Kirsten Morris, Prof. Jeff Orchard, and Prof. Sze Zheng Yong, for their valuable feedback and suggestions that have significantly improved the quality of this thesis.

I would like to express my sincere gratitude to my colleague, Yiming Meng, for his invaluable guidance and help throughout the journey. He has been both a mentor and a dear friend, guiding me through technical challenges, helping me solve research puzzles, and offering steadfast encouragement during difficult times. Most importantly, I feel fortunate to have worked alongside him, as his passion for research and constructive insights have inspired me to explore new ideas and pursue my work with renewed enthusiasm.

I am grateful to have had my wonderful co-workers, Thanin Quartz, Maxwell Fitzsimmons, and Zhexuan Zeng. I am also deeply thankful to my friends, especially my roommate, Jiazheng Han, for making my time at Waterloo both enjoyable and enriching.

Finally, I would like to express my deepest appreciation to my family for their unwavering support and encouragement throughout my academic journey. Their love and belief in me have been a constant source of strength.

Dedication

To my family, friends, and mentors who have supported me throughout this journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI4Science artificial intelligence for science 1, 2

DOA domain of attraction 13, 15, 42

EDMD Extended Dynamic Mode Decomposition 78

ELM extreme learning machine 22–25, 27–39, 105, 116, 121

FDM finite-difference method 58, 59, 64, 81, 83, 84, 87, 88, 90, 91, 94, 96, 98–100, 102, 103

GP Gaussian process 10, 19

HJB Hamilton-Jacobi-Bellman v, 6, 21, 29–31

KLM Koopman Logarithm Method 81, 83, 84, 87–91, 94, 96–103

LQR linear–quadratic regulator 7, 40, 44, 48, 50, 53–56

MILP mixed-integer linear program 18

MIP mixed-integer programming 18

MIQP mixed-integer quadratic program 18

MPC model predictive control 10, 11

PDE partial differential equation 24–28, 30–32, 36, 84, 105, 109–111, 114–119

PDEs partial differential equations v, 6, 21, 22, 24, 39, 95, 104–106, 110, 120–122

PI policy iteration 29–31, 36–38

PID proportional–integral–derivative 11

PINN physics-informed neural network 17, 24, 34–38, 109, 121

PINNs physics-informed neural networks 15, 32–35

RBF radial basis function 21

RL reinforcement learning 4, 12

RMSE root mean squared error 84, 87, 90, 91, 96–104

ROA region of attraction 2, 4, 12, 13, 15–17, 19, 21, 22, 27, 32, 34, 35, 39, 40, 43, 47, 50–56, 104, 105, 109, 111, 112, 116–118, 120, 121

RTM resolvent-type method 58, 81, 83–85, 87, 88, 90, 91, 94, 96–103

SGA successive Galerkin approximation 36–38

SINDy Sparse Identification of Nonlinear Dynamics 9, 83, 84, 87, 88, 90, 94, 98, 100–103, 114, 119

SMT satisfiability modulo theories v, vi, xiv, 2, 5, 6, 16–18, 28, 29, 31, 41, 45, 46, 48, 50, 51, 56, 105, 110, 117, 118, 120–122

SOS sum-of-squares 13, 21, 36, 40

SRTM Sparse RTM 83, 84, 88, 90, 94

UAS uniformly asymptotically stable 42, 43, 49–51

WSINDy Weak SINDy 88, 98, 100–103

List of Symbols

We have made considerable effort to maintain succinct and consistent notation throughout this thesis. Due to the complexity of the subject matter, some notations are inevitably overloaded. This list provides the main symbols and notations used throughout this thesis. Notations that are defined locally within specific sections are not included in this list.

General Notation

Ω: state space

f: vector field

$\phi(t, x)$: flow map (solution map)

$\mathbb{R}^d$: Euclidean space of dimension $d > 1$

$\mathbb{R}$: the set of real numbers

$\mathbb{C}$: the set of complex numbers

$\mathbb{R}^+$: the set of non-negative real numbers

$|\cdot|$: Euclidean norm

$\|\cdot\|_F$: Frobenius norm for matrices

$\cdot^\mathsf{T}$: matrix/vector transpose

$\cdot^\dagger$: matrix pseudoinverse

$\overline{A}$: closure of set A

$\text{int}(A)$: interior of set A

∂A: boundary of set A

$\mathcal{C}(\Omega)$: set of continuous functions with domain Ω

$\mathcal{C}^i(\Omega)$: set of i-times continuously differentiable functions with domain Ω

Koopman Operator Theory

For the reader's convenience, we provide a partial list of notations related to Koopman operator theory and the corresponding learning, specifically in Chapters 5 & 6, and the Appendix.

$\mathcal{F}$: function space of observable functions

$\mathcal{S}_t$: semigroup operator

Id: identity operator on any properly defined spaces

$\text{dom}(\mathcal{A})$: domain of operator $\mathcal{A}$

$\text{ran}(\mathcal{A})$: range of operator $\mathcal{A}$

$\mathcal{A}$: generator of semigroup

M: the number of sampled initial conditions

N: the number of dictionary observable functions

$\mathcal{Z}_N(x)$: the dictionary of N observable functions

$\mathcal{K}_t, \mathcal{L}$: the Koopman operator and its infinitesimal generator

K, L: the data-driven approximation of $\mathcal{K}_t$ (with a fixed t) and the $\mathcal{L}$, with dimension $N \times N$

τ_s: terminal observation instance for data collection

γ: observation/sampling frequency

Γ: number of snapshots for each trajectory during data collection, which is equal to $\gamma\tau_s$

τ: the sampling period, which is equal to $1/\gamma$

Chapter 1

Introduction

At the heart of dynamical systems theory is the study of natural and engineered systems that can be modelled by differential equations. In addition, control theory was developed to design control policies for dynamical systems to achieve desired properties, such as tracking performance, stability, and safety guarantees, typically with the help of feedback. In the last century, researchers primarily employed model-based control, designing controllers for dynamical systems based on accurate models of the plants using existing mathematical tools. However, as a matter of fact, most real-world dynamical systems, such as quadcopters and humanoid robots, are high-dimensional with uncertainties that cannot be accurately modelled with classical system identification tools. Furthermore, it is difficult to guarantee their stability and design optimal controllers using existing techniques, even in a practical sense.

Over the past decades, data-driven control using big data has become increasingly prevalent and has shown its effectiveness in tackling problems that are challenging or intractable for classical model-based control approaches. Such nonlinear systems include those with unknown dynamics, systems with complex nonlinearities, and high-dimensional systems [18]. Meanwhile, from the learning-based methods' perspective, data-driven control falls into the category of artificial intelligence for science (AI4Science), which has prevailed in various scientific problems, ranging from protein structure prediction with AlphaFold to climate modelling and prediction using deep neural networks. In the systems and control community, by leveraging an abundance of sensory measurements, learning-based methods can be used to approximate unknown functions or construct nonlinear controllers. For instance, neural networks are widely used to learn the unknown dynamics and nonlinear controllers for nonlinear dynamical systems [47].

Moreover, stability analysis is at the core of nonlinear systems, where stability properties of various nonlinear systems can be qualitatively characterized by Lyapunov functions with the notable Lyapunov stability theorems [58]. One common application of stability is the tracking task for a vehicle along a given trajectory, in which stability of the dynamics guarantees that the car tracks well. However, discovering a Lyapunov function remains a long-standing challenge in nonlinear dynamical systems, even for known systems. For unknown systems, this challenge is further compounded by the accuracy of data-driven system identification and the formal verification of the learned Lyapunov functions. On the other hand, if the hyperbolic equilibrium point of a given nonlinear system is asymptotically stable, a Lyapunov function is a scalar function mapping from the state space to $\mathbb{R}$, satisfying a set of Lyapunov conditions to guarantee the stability of the system. As a result, learning-based methods can be used not only to identify unknown dynamics, but also to learn Lyapunov functions that satisfy certain conditions called the Lyapunov conditions to provide stability guarantees for the dynamical systems. On top of that, the level sets of Lyapunov functions depict an estimate of the region of attraction (ROA) of dynamical systems, in which all states eventually converge to the equilibrium points. Being able to compute larger ROA estimates for nonlinear systems is itself of major interest in control theory. In a similar manner to the stability properties, the safety properties of a system can be characterized by barrier functions, or they can be unified with the stability ones via Lyapunov-barrier functions [90]. However, when one uses AI4Science techniques, the learned solutions typically lack formal guarantees due to the probabilistic nature of machine learning methods. To overcome this issue, verification techniques for the learned functions can be implemented when stability and safety are of ultimate importance, which is common in many cyber-physical systems that need safety-critical control, such as flight control and autonomous vehicles.

Above all, we can learn three objectives with learning-based approaches: unknown dynamics, nonlinear controllers, and Lyapunov functions. In the scenarios where formal guarantees are needed for the stability certificates, formal verification methods, such as SMT solvers, may be taken into account to verify the learned Lyapunov functions.

1.1 Learn Unknown Dynamics

We consider an n-dimensional nonlinear autonomous system of the following form:

$$\dot{x} = f(x), \tag{1.1}$$

where $x \in \Omega$ is the state of the system, and $\Omega \subseteq \mathbb{R}^n$ is the state space. We denote the unique solution to (1.1) from the initial condition $x(0) = x_0$ by $\phi(t, x_0)$ for $t \in J$, where J is the maximal interval of existence for ϕ.

In many practical scenarios, f on the right-hand side is unknown or partly known. How to learn the dynamics f, also known as system identification, has been a vital and intensively studied topic in the community of control theory, starting from the 1960s when Kalman published his well-known work on filtering, estimation and prediction of linear stochastic systems [54, 53]. In the last century, the study was dominated by classical model-based methods to identify the additive noise, disturbance, or a small amount of unknown part of nonlinear systems based on model-based control techniques. Thanks to the development of learning-based approaches since the 1990s, learning more complex unknown dynamics becomes possible with a sufficient amount of data generated in the process or by simulation. In some cases, only a few mild assumptions or almost no prior knowledge of the dynamical systems is needed. Taking neural networks as an example, the well-known universal approximation theorem states that it is possible to approximate any nonlinear function with a one-hidden-layer neural network [44]. Apart from that, recently, an operator theoretic approach, the Koopman operator, has been used as a powerful tool to identify the unknown dynamics by mapping the nonlinear systems into some linear operator space with the help of the autoencoder [82]. These learned dynamics are also known as data-driven models in some literature. A more detailed review of this topic will be provided in the next chapter.

1.2 Learn Nonlinear Controllers

Nonlinear control design is another difficult problem for nonlinear systems, even with known dynamics. The most common control design method implemented in real life is the feedback control technique, whose basic structure can be found in Fig. 1.1. How to learn the 'System' block using learning-based approaches has been discussed in the previous section. Nonlinear control design is to solve a high-dimensional and non-convex problem, while many machine learning methodologies are non-convex optimization to find the global/local minima as well, which makes machine learning control suitable for identifying controllers for most real-life control systems. Essentially speaking, nonlinear controllers are a set of nonlinear functions. Taking neural networks as the example again, it is also possible to have neural network controllers to stabilize the nonlinear systems, which has been well studied over the past forty years [68].

Referring to the definition of *data-driven control design* in [47], it is the synthesis of

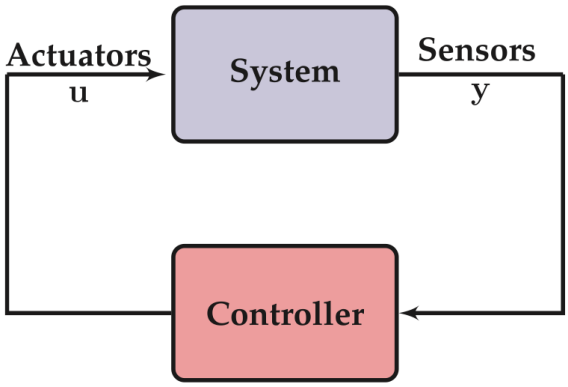


Figure 1.1: Illustration of a typical feedback control structure [18].

a controller using input and output data measured on systems to be controlled without explicit use of (non)parametric models of the systems. It is worth mentioning that nonlinear controllers can be developed with partly known or totally unknown systems. For the former one, neural networks and other learning-based methods can be used to design a nonlinear adaptive controller for stochastic systems with unknown disturbance or noise. Regarding the latter one, model-free reinforcement learning (RL), for example, Q-learning, can be a perfect fit to generate control policies [15].

1.3 Learn Lyapunov Functions and ROA Estimates

Finding the ROA of an asymptotically stable equilibrium point for a nonlinear dynamical system remains one of the most challenging tasks, especially for systems with uncertainty [84]. Even though the well-known Lyapunov theorems, which use the level sets of a Lyapunov function to estimate the ROA, were proposed more than a century ago, there is no general approach to finding a valid Lyapunov function for general nonlinear systems with provable guarantees [59]. As previously mentioned, thanks to the development of learning-based methods, several data-driven approaches have recently shown their effectiveness in identifying unknown (or partly known) systems [27, 112]. However, how to simultaneously find Lyapunov functions and nonlinear controllers for systems with unknown dynamics are still an open and active problem in control and robotics applications [51].

Similar to the unknown dynamics, the Lyapunov function of an autonomous system can be regarded as a nonlinear function as well. A Lyapunov function is well defined in the

following famous theorem:

Theorem 1.3.1 (Lyapunov Stability Theorem)**.** *Let f in (1.1) be a locally Lipschitz function defined over a domain $D \subset \mathbb{R}^n$, which contains the origin, and $f(0) = 0$. Let $V(x)$ be a continuously differentiable function defined over D such that*

$$V(0) = 0 \text{ and } V(x) > 0 \text{ for all } x \in D \text{ with } x \neq 0 \tag{1.2}$$

$$\dot{V}(x) \leq 0 \text{ for all } x \in D \tag{1.3}$$

Then, the origin is a stable equilibrium point of $\dot{x} = f(x)$. Moreover, if

$$\dot{V}(x) < 0 \text{ for all } x \in D \text{ with } x \neq 0, \tag{1.4}$$

then the origin is asymptotically stable. Furthermore, if $D = R^n$, (1.2) and (1.4) hold for all $x \neq 0$, and

$$V(x) \to \infty \quad as \quad \|x\| \to \infty, \tag{1.5}$$

then the origin is globally asymptotically stable.

With that being said, it brings us an insight of using another learning-based approach, for example, neural networks [22], to learn a Lyapunov function for any given nonlinear system. After obtaining a Lyapunov function candidate, one can use an additional verification step to verify if all the equality and inequality Lyapunov conditions are met. Verification of the outputs of learning methods is also an active and open research topic. Generally speaking, taking neural networks as an example, one can use SMT solvers to verify the outputs, or formulating the verification problem as a convex optimization problem or a reachability problem [139], which will be further discussed in detail in the next chapter.

1.4 Organization of the Thesis

As discussed in the previous sections, the goal of this thesis is to construct systematic approaches to perform system identification tasks and find learning-based Lyapunov functions for nonlinear dynamical systems, with some extensions to optimal control.

Chapter 2 provides a literature review on the related topics, including system identification, data-driven control design, and Lyapunov-based stability analysis. For all of them, learning-based approaches are given as the main focus.

Chapter 3 presents a neural network-based method to efficiently learn the Lyapunov functions for nonlinear systems by solving stability-related PDEs, given that the system dynamics are known. On top of that, by solving the well-known HJB equation, apart from the stability certification, we can also obtain the optimal control policy for the system.

Chapter 4 proposes a framework consisting of two neural networks to simultaneously learn the system dynamics and Lyapunov functions for unknown nonlinear systems. The learned Lyapunov function is formally verified using an SMT solver to provide provable stability guarantees for the unknown dynamics.

Chapter 5 introduces a novel Koopman generator-based approach to identify unknown nonlinear dynamical systems. The proposed data-driven method leverages the resolvent operator of the Koopman generator to have an accurate approximation of it, which is then used to learn the system dynamics.

Chapter 6 extends the Koopman generator-based method to learn Lyapunov functions for unknown nonlinear systems. With the learned Koopman generator, solving the stability-related PDEs (Lyapunov's equation and Zubov's equation) can be reduced to a least-squares problem, and the obtained stability certificates are verified using an SMT solver.

Chapter 7 concludes the thesis and discusses future work.

Chapter 2

Literature Review

This chapter contains a brief review of the literature relevant to system identification, data-driven control, and Lyapunov-based stability analysis for nonlinear systems. Section 2.1 provides an essential background of system identification and data-driven control approaches, with an emphasis on the state-of-the-art machine learning approaches to learn the unknown dynamics and controllers. Following that, we provide a more detailed review of Lyapunov-based stability analysis, including the maximal Lyapunov functions, in Section 2.2. On top of that, an emphasis on Lyapunov-based stability guarantees for nonlinear systems via learning-based methods is presented in Section 2.3.

2.1 System Identification and Data-Driven Control

When the notation of modern control theory was proposed with the parametric state-space models by Kalman sixty years ago, numerous beautiful and rigorous theorems were developed for model-based control last century, including zero-pole assignment, linear–quadratic regulator (LQR) design, robust control and Lyapunov-based controller design [47]. But all of them require that we have explicit knowledge of the dynamics of the systems, and most of them can only deal with low-dimensional dynamical systems. However, in practice, there are several challenges that may lead to the failure of the aforementioned model-based control methods. Here, we list a few main issues [17]:

- **Unknown or partly known dynamics.** In some cases, it is extremely hard to accurately model the nonlinear systems with first principles in a mathematical way.

- **Complex nonlinearities.** Many model-based control approaches are only eligible for implementation on linear systems. When it comes to nonlinear systems, one must linearize the system first. However, when complex nonlinearities are involved, linearization may not be able to provide any meaningful results.
- **High-dimensional systems.** A large fraction of classical model-based control techniques handle only relatively low-dimensional systems.

As a result of the rapid development of modern sensory technology and machine learning algorithms, we typically have easy access to a large amount of measurements. The confluence of big data and learning-based approaches contributes to a new paradigm of control theory, data-driven control, which is also known as learning-based control in some literature. In general, data-driven control refers to the methods where the controllers are designed with input and output data directly to provide stability guarantees, convergence, and robustness of the control systems [47]. In a similar manner, learning-based system identification is to find the dynamical models from the data. It is notable that some learning-based control applications identify the system first and then design a controller, while some do not require the explicit information of the model, in which adaptive controllers are well designed to meet the requirements. Here, we briefly review the literature on both system identification and data-driven control.

2.1.1 System Identification for Nonlinear Systems

System identification is one of the most well-developed branches of control theory. There are two typical models: *black box* and *grey box*, where the latter one refers to having some prior knowledge of the dynamical system, while *black box* means that one only has access to the observed input and output signals/data. Plentiful identification approaches have been proposed to complete this regression task, such as least-squares methods as well as transient and frequency analysis [78]. In this subsection, after reviewing the classical system identification techniques, we emphatically review the ones related to the following three dominant learning-based approaches: neural networks, Gaussian process regression and Koopman operator.

Classical System Identification Techniques for Nonlinear Systems

As detailed in [79, Chapter 5], if one has the physical knowledge or understanding about the essential nonlinearities which might be very limited beforehand, then *semi-physical*

modelling techniques can be used to model the system, such as Wiener and Hammerstein models and some linear regression models. On the other hand, if the nonlinear system is completely a *black-box*, then it typically boils down to computing a regression vector β with a (carefully chosen) set of basis functions by solving an optimization problem in the following form [27]:

$$\beta = \arg\min_{w} \frac{1}{N} \sum_{i=1}^{N} \mathcal{M}_i \left(y_i, f(x_i; w)\right) + \gamma \|w\|^2, \tag{2.1}$$

where x_i and y_i are input samples/sequences and output samples/sequences, respectively, up to the N-th sample/sequence. $\mathcal{M}_i$ is a loss function measuring the difference between the predicted output $f(x_i; w)$ and the observed output y_i, and γ is a regularization parameter to avoid overfitting. The function f is typically a linear combination of M basis functions $\phi_j(x_i)$, i.e., $f(x_i; w) = \sum_{j=1}^{M} w_j \phi_j(x_i)$, where w_j are the weights of the basis functions. The common choices of the basis functions include: polynomials, neural networks, and kernel estimators.

To reduce model complexity and avoid overfitting, many scholars nowadays pursue the sparse identification of nonlinear systems from the basis functions. The most well-known work is the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm [19]. In a similar vein, the authors of [103, 143] proposed a method to find the sparse representation from a Bayesian viewpoint. However, these methods typically require that time derivatives of the state can be accurately estimated, an assumption that may not be robustly satisfied due to potential challenges such as low sampling rates, noisy measurements, and short observation periods. Different from these methods approximating the derivatives, [96] integrated the weak formulation into the framework with well-designed test functions to compute a more accurate sparse representation of the dynamics on top of SINDy, by computing the integrations instead. This method is well-known as Weak SINDy, which is initially designed to handle systems governed by ODEs. Given the usage of the weak form, the natural extension of it was to model PDEs in [95] by the same authors.

System Identification via Neural Networks

The inherent ability of approximating any nonlinear function arbitrarily well makes neural networks a perfect match for system identification, and it has been extensively studied and implemented in this area. A systematic study on how to approximate nonlinear systems with classical feedback neural networks is presented in [74]. Moreover, as the trajectories of dynamical systems can be regarded as time series, Wang et al. [50] use a recurrent

neural network to generate the state-space representation of an unknown dynamical system. Inspired by that, [3] proposes a physics-constrained learning framework to learn a more accurate dynamical model with the help of an autoencoder structure.

System Identification via Gaussian Process Regression

Gaussian process (GP) regression, a probabilistic supervised machine learning framework, has also been widely used in function regression tasks [135], and this nonparametric learning-based regression approach has shown its effectiveness and superiority, compared to deep learning, when there is not a large amount of data available. Moreover, the trained model can be updated easily when there are newly observed data points via recalculating the mean and covariance matrices. As one of the earliest applications of GP regression on function approximation of nonlinear dynamical systems, the prediction of the states of a robot arm problem can be found in [137]. Moreover, by taking advantage of GP regression, which can be updated easily, [8] proposes an algorithm to update the learned dynamics with probabilistic stability guarantees provided by a robust controller. Based on that, the same authors explore the ability of GP regression on gradually enlarging the region of attraction of the unknown systems by safely exploring the state spaces [7]. Apart from that, GP regression has been widely utilized with model predictive control (MPC), which also updates its control policies for a given future time horizon based on the new measurements. In this setting, the posterior distribution of possible functions from GP regression contributes to generating the dynamics constraints for the MPC problem [52, 62]. It is notable that GP regression lies in the category of kernel-based machine learning methods for system identification. We direct the reader to [27] for a detailed survey along this research direction.

System Identification via Koopman Operator

Recently, the Koopman operator theory provides a promising alternative learning approach for nonlinear system identification, which shows that a finite-dimensional nonlinear system can be mapped into an infinite-dimensional linear operator space. The theory was first proposed by Koopman in 1931 [64], and then it witnessed a resurgence after Mezic presented that the spectral properties of the linear Koopman operator could be used as Koopman mode decomposition for nonlinear systems in 2005 [97]. Basically speaking, the Koopman operator lifts the states of the nonlinear systems into the infinite-dimensional space of its observable functions (functions of the states), where the evolution of those observable functions is linear. This is vital, as there is a library of well-developed tools for linear systems, including stability analysis, controllability and observability. But in practice, it is impossible to

find some infinite-dimensional space or Koopman invariant subspace spanned by specially chosen observable functions. Hence, we need to find an approximation where the leading observable functions are included [16]. For physical systems, the observable functions can be real measurements, for example, velocity or vorticity for fluid dynamics. Given the measurements, some techniques have been devised to obtain the linear representation, such as, dynamic mode decomposition [125] and extended dynamic mode decomposition [137].

However, how to find the best observable functions for general nonlinear systems with the state measurements is not easy but at the core of the Koopman operator, and it determines the accuracy of the approximation. Owing to the development of deep learning, observable function approximation can be found by training an autoencoder, in which the lifted states are the needed observable functions [82]. Under this framework, [3] proposes a consistent Koopman autoencoder structure to reach better approximation performance. Moreover, the recent work [43] designs an MPC controller for the nonlinear system with the knowledge of the linear relationship of the observable functions by taking both the dynamics and the controller synthesis into consideration when training the deep neural networks for general nonlinear systems.

Apart from using the Koopman operators to directly identify the nonlinear systems, including both discrete-time and continuous-time systems, researchers have also been working on using the infinitesimal generator of the Koopman operator, so-called Koopman generator, to learn the continuous-time dynamics of nonlinear systems. By taking advantage of the exponential relationship between the infinitesimal generator and the Koopman operator, [85] proposed a method to compute the approximation of the infinitesimal generator by taking the logarithm of the learned Koopman operator from data. However, this approach requires the diagonal property of the Koopman operator and poses strict requirements on the choice of basis functions and the underlying dynamics [94]. The authors of [61], from a different perspective, leveraged the definition of the infinitesimal generator to compute the Koopman generator and perform system identification tasks afterwards when the learned Koopman operator is available.

2.1.2 Data-Driven Control

The most well-known and popular data-driven control in the industry must be proportional–integral–derivative (PID) controller, which uses the error information between the reference and the outputs to drive the systems to the desired states [101]. The key in PID control is to tune the three parameters: proportional gain, integral gain and derivative gain. Various methodologies are invented to guarantee a good performance in practice, for instance, the genetic algorithm [17].

Another well-developed branch of learning-based control is neural network control, especially in closed-loop feedback control systems, to guarantee the tracking performance. How to design one-layer and multi-layer neural network controllers for discrete-time nonlinear systems with/without uncertainties are articulated in [122]. Meanwhile, the robustness and stability of neural network controllers with uncertain nonlinear systems are investigated in [140, 70, 71], especially the guaranteed tracking performance with error dynamics.

RL is also a good candidate for data-driven controllers given its nature of updating the action/control policies by minimizing the cost or maximizing the rewards over iterations. RL provides the flexibility on the model information, considering the fact that it can be implemented on both model-based or model-free systems. In the latter case, one typically does not need explicit models about the dynamics, and the 'optimal' control policies are obtained by interacting with the environment and taking more input and output data. One of the most common RL structures is the actor-critic structure, in which the actor performs action or control policies, while the critic evaluates the performance via a value function. By optimizing the value function with Bellman's optimality principle, we have access to the optimal controllers for the feedback control systems [69]. Furthermore, [110] devises the control policies with safety and performance guarantees in the framework of RL by combining it with Lyapunov design principles. Based on that, Brunke et al. [15] give a systematic survey on reinforcement learning-based control with applications on safe learning in robotics.

In this subsection, a brief review on three mainstream data-driven control approaches is presented. We refer the readers to [47] for a detailed survey on both model-based control and data-driven control methods, as well as [69] for a systematic review on RL for feedback control systems.

2.2 Lyapunov-Based Stability Analysis

The stability analysis literature discussed in this section is the ones based on Lyapunov stability theorems, known as Lyapunov-based stability analysis. As stated in Theorem 1.3.1, the stability of an equilibrium point can be characterized by a Lyapunov function $V(x)$, and then one is able to estimate the corresponding ROA for the nonlinear system. In the following, we first introduce the following definitions.

Definition 2.2.1. *A set $S \subseteq \mathbb{R}^n$ is called forward invariant for (1.1) under the flow ϕ if*

$$x_0 \in S \quad \Longrightarrow \quad \phi(t, x_0) \in S \quad \forall\, t \geq 0. \tag{2.2}$$

Definition 2.2.2. *We assume that $x = 0$ is an asymptotically stable equilibrium point of (1.1). The domain of attraction (DOA) of the origin for (1.1) is defined as*

$$\mathcal{D} := \left\{ x \in \mathbb{R}^n : \lim_{t \to \infty} |\phi(t, x)| = 0 \right\}. \tag{2.3}$$

We call any forward invariant subset of $\mathcal{D}$ a region of attraction (ROA).

As a matter of fact, $\mathcal{D}$ is an open and connected set [10].

As characterized in [77], when the origin is an asymptotically stable equilibrium point for (1.1) and f is locally Lipschitz, one can compute a Lyapunov function by solving the following Lyapunov equation:

$$DV \cdot f = -\omega(x), \quad x \in \Omega, \tag{2.4}$$

where $\omega(x)$ is a positive definite function with respect to the origin, i.e., $\omega(0) = 0$ and $\omega(x) > 0$ for all $x \in \Omega \setminus \{0\}$.

2.2.1 Maximal Lyapunov Functions

Traditional approaches introduced in classical textbooks for computing the Lyapunov functions, such as solving the notable Lyapunov equation after linearization [59] and sum-of-squares (SOS) tenchinques [105, 104], typically yield ROAs which are under-approximations of the DOA. The following theorem, proposed in [132], characterizes the maximal Lyapunov function, which provides the possibility of capturing the whole DOA, or so-called maximal ROA estimate.

Theorem 2.2.3 ([132])**.** *Let $D \subset \mathbb{R}^n$ be an open set containing the origin. Suppose that there exists a continuous function $V : D \to \mathbb{R} \cup \{\infty\}$ and the following conditions hold:*

1. *V is positive definite on D with respect to the origin, i.e., $V(0) = 0$ and $V(x) > 0$ for all $x \in D \setminus \{0\}$;*
2. *the derivative of V along solutions of (1.1), given by*

$$\dot{V}(x) := \lim_{t \to 0+} \frac{V(\phi(t, x)) - V(x)}{t}, \tag{2.5}$$

is well defined for all $x \in D$ and satisfies

$$\dot{V}(x) = -\omega(x), \tag{2.6}$$

where $\omega : D \to \mathbb{R}$ continuous and positive definite with respect to the origin;

3. $V(x) \to \infty$ *as* $x \to \partial D$ *or* $\|x\| \to \infty$.

Then $D = \mathcal{D}$.

Correspondingly, we have the following definition:

Definition 2.2.4 (Maximal Lyapunov function [132]). *A function* $V : \mathbb{R}^n \to \mathbb{R}_+ \cup \{\infty\}$ *is called a maximal Lyapunov function for the system* (1.1) *if*

1. V *is positive definite on* $\mathcal{D}$ *with respect to the origin, i.e.,* $V(0) = 0$ *and* $V(x) > 0$ *for all* $x \in \mathcal{D} \setminus \{0\}$;
2. $V < \infty$ *if and only if* $x \in \mathcal{D}$;
3. $V(x) \to \infty$ *as* $x \to \partial\mathcal{D}$ *and/or* $\|x\| \to \infty$;
4. $\dot{V}(x)$ *is well-defined and negative definite over* $\mathcal{D}$.

The maximal Lyapunov function defined above possesses an unbounded nature at the boundary, which imposes unsolvable difficulties in numerical computations in practice. Theorem 2.2.3 is equivalent to the well-known Zubov's theorem [75], as stated below.

Theorem 2.2.5 (Zubov's theorem [152]). *Let* $D \subset \mathbb{R}^n$ *be an open set containing the origin. Then* $D = \mathcal{D}$ *if and only if there exist two continuous functions* $W : D \to \mathbb{R}$ *and* $\Psi : D \to \mathbb{R}$ *such that the following conditions hold:*

1. $0 < W(x) < 1$ *for all* $x \in D \setminus \{0\}$ *and* $W(0) = 0$;
2. Ψ *is positive definite on* D *with respect to the origin;*
3. *for any sufficiently small* $c_3 > 0$, *there exist two positive real numbers* c_1 *and* c_2 *such that* $|x| \geq c_3$ *implies* $W(x) > c_1$ *and* $\Psi(x) > c_2$;
4. $W(x) \to 1$ *as* $x \to y$ *for any* $y \in \partial D$;
5. W *and* Ψ *satisfy*
$$\dot{W}(x) = -\Psi(x)(1 - W(x)), \tag{2.7}$$
where $\dot{W}$ *is the right-hand derivative of* W *along solutions of (1.1) as defined in* (2.5).

We refer to (2.7) as Zubov's equation in the following content. Clearly, the sublevel-1 set of $W(x)$ is the DOA, $\mathcal{D}$. Moreover, there is a connection between V defined in Theorem 2.2.3 and W here. Let $\beta : [0, \infty) \to \mathbb{R}$ be a function, satisfying

$$\dot{\beta} = (1 - \beta)\psi(\beta), \quad \beta(0) = 0, \tag{2.8}$$

where ψ is a locally Lipschitz and $\psi(s) > 0$ for $s \geq 0$. Then, for any function β defined above, we have

$$W(x) = \begin{cases} \beta(V(x)), & \text{if } V(x) < \infty, \\ 1, & \text{otherwise.} \end{cases} \tag{2.9}$$

It is evident that β is continuously differentiable and strictly increasing, with $\beta(0) = 0$ and $\beta(s) \to 1$ as $s \to \infty$. Two special cases that are commonly used in the literature are: $\beta(s) = 1 - \exp(-\alpha s)$ and $\beta = \tanh(\alpha s)$, for some constant $\alpha > 0$.

Originated from [132] where the authors utilized rational functions to compute the Lyapunov functions in an interactive way, researchers have been devoted to computing the maximal Lyapunov functions in the past four decades. Based on this framwork, Matallana et al. formulated the computation as an optimization problem in [84]. More recently, the authors in [56] used neural networks to compute the maximal Lyapunov functions by performing regression tasks with the help of data generated by ODE solvers, while physics-informed neural networks (PINNs) are utilized to solve Zubov's equation (2.7) for maximal ROA estimates. It is also worth noting that Wei Kang extended this idea to controlled nonlinear systems in [55] for optimal control policy in 1995. Apart from the papers directly derived based on Zubov's theorem, Yuan and Li [142] leveraged the famous Hamilton–Jacobi equation and the corresponding reachable sets to estimate the maximal ROA for nonlinear systems.

2.3 Lyapunov-Based Stability Analysis with Learning-based Methods

With the review on Lyapunov-based stability analysis in the previous section, the focus in this section is on how to compute the Lyapunov functions with learning-based approaches, emphatically on two prevalent methods: neural networks and the Koopman operator. At the same time, it is worth mentioning that classic Lyapunov-based methods have been widely used for stability analysis of adaptive control systems, where the Lyapunov functions are designed based on both the states and error/disturbance terms to provide stability

guarantees [106, 121, 147]. Furthermore, the stability analysis of (unknown) nonlinear systems with neural network-based controllers has been well studied with Lyapunov-based approaches under the framework of adaptive control as well [140, 70, 71]. The recent work [107] extends the stability results to deep neural networks with real-time weight adaptation laws for different layers. The detailed review on these two topics is omitted.

2.3.1 Lyapunov-Based Stability Analysis via Neural Networks

Given its capacity to approximate any nonlinear functions, neural networks have been used to construct or approximate Lyapunov functions since the 1990s [80, 114]. Based on the two aforementioned earliest works, [111] proposed an approach to construct valid neural Lyapunov functions by solving an optimization problem with the knowledge that the value at the equilibrium point should be a local optimum. More recently, due to the prevalence of big data and deep learning, various researchers have developed algorithms to learn Lyapunov functions for high-dimensional systems. In [42], the authors showed that compositional Lyapunov functions can be constructed for high-dimensional systems using deep neural networks if the given system satisfies the small-gain condition. In a similar manner, Gaby et al. [38] proposed a different deep neural network architecture, Lyapunov-Net, to approximate Lyapunov functions for high-dimensional systems. Moreover, a Lyapunov neural network is derived in [117] to learn a fixed quadratic format Lyapunov function for finding the largest estimated ROA. However, the learned Lyapunov functions in the aforementioned literature lack formal guarantees.

As a matter of fact, when it comes to finding a Lyapunov function with a typical feedforward neural network to certify the stability for the nonlinear systems, performing this kind of regression task is difficult for shallow neural networks, where only equality and/or inequality Lyapunov conditions, instead of explicit expression or values of the function, are given. Consequently, it is crucial to develop methods to test or verify the outputs of a neural network and show that they are valid Lyapunov functions. There are three main approaches to verify the neural networks [139, 73]:

- Using SMT solvers which aim to decide the satisfiability of a first-order logic formula with unknowns and relations lying in certain theories [98]. Its predicates include equalities, linear inequalities and functions, e.g., $2\sin(x) > 1$ with a given domain of x. There are typically two kinds of results returned by the SMT solver: `SAT`, which stands for satisfiable; `UNSAT`, in which case no satisfying assignment is found. Several SMT solvers have been developed by leading researchers, such as, Z3 [99] and CVC5 [4].

- Formulating the neural networks as a reachability problem to investigate the reachable set of the output of the neural networks with layer-by-layer reachability analysis [119], which has been widely studied on control systems [133, 130].
- Formulating it into an optimization problem, which is utilized to falsify the assertion. The most common approach is to represent the neural network as a constraint and use convex over-optimizations of the activation functions. In this framework, semidefinite programming or mixed-integer programming (ReLU activation functions mainly) can be used to solve optimization problems with constraints [37, 129].

The authors of [57] proposed that the validity of the learned Lyapunov functions can be validated by SMT solvers and the counterexamples generated by the solvers can be used to guide the search of the Lyapunov function, but the Lyapunov function is of polynomial form in this work. On top of that, Chang et al. [22] proposed the popular neural Lyapunov control method, which consists of a learner to learn Lyapunov function candidates and a falsifier to find counterexamples with the help of an SMT solver. When the SMT solver returns `UNSAT`, the learned Lyapunov function is a valid one. A tool of this idea, named FOSSIL, is also developed by Abate et al. [1] to synthesize valid Lyapunov functions with two SMT solvers, Z3 [99] and dReal [40]. Recall that the sub-level sets of the Lyapunov function depict estimates of ROA of dynamical systems. To realize larger ROA estimates with the neural Lyapunov functions, two different approaches were proposed in the following two papers. In the earlier work, Richards et al. [117] proposed a Lyapunov neural network of a quadratic form, and the ROA estimate given by the sub-level set is enlarged by generally exploring the convergent trajectories in the state space. Recent work by Mehrjou et al. [88] designed a different loss function for training the Lyapunov neural network to learn better controllers, which yields larger ROA estimates. More recent work in [31] uses the same methodology to approximate control Lyapunov barrier functions for nonlinear systems to guarantee both stability and safety for reach-avoid robot tasks. Furthermore, by taking advantage of the well-known Zubov's theorem, the authors in [75] utilized a physics-informed neural network (PINN) to learn a maximal neural Lyapunov function which yields the maximal estimate of ROA with formal guarantees.

In the previous paragraph, the papers mainly concern finding a Lyapunov function for systems where the dynamics are known. In the presence of unknown dynamics, finding a valid Lyapunov function is more difficult. There are a few results regarding learning unknown dynamics and Lyapunov functions simultaneously. In [63], an algorithm was proposed to jointly learn the dynamics and a Lyapunov function using input convex neural networks. Moreover, taking advantage of projection from state space to a latent space and stable invariant sets, [127] achieved better results within the same framework. Similar

to the known dynamics case, the foregoing learned Lyapunov functions may not be valid Lyapunov functions that satisfy all the Lyapunov conditions with respect to the true dynamics. To address this issue, the authors of [150] modified the Lyapunov condition on the Lie derivative to accommodate the generalization error of the learned dynamics when verifying the learned Lyapunov function with the SMT solver.

Besides pursuing formal guarantees with SMT solvers, one can also achieve the same target by solving a mixed-integer programming (MIP) if only ReLU neural networks are presented. Given the piecewise linearity of (leaky) ReLU neural networks, converting them into MIP is easy, and then MIP solvers can be used to check the output constraints translated from Lyapunov conditions. With the similar learner-verifier structure as [22], [24] learned piecewise quadratic Lyapunov functions for piecewise affine systems with ReLU neural network controllers by formulating the learning process as a mixed-integer quadratic program (MIQP). Based on that, the same authors devised robust Lyapunov functions for uncertain nonlinear systems and hybrid systems by approximating them with piecewise linear neural networks and then solving the MIQP problem in [25, 26], where the Lyapunov functions were parametrized as a quadratic function of the states on trajectories. To address the same problem for piecewise linear dynamical systems, Dai et al. [29] parametrized the Lyapunov function by leaky ReLU neural networks and obtained a valid one by solving a mixed-integer linear program (MILP). Moreover, the superset of the authors devised an approach to simultaneously learn a neural network controller and a Lyapunov function for general nonlinear systems by approximating the dynamics with a ReLU neural network in [30]. It is worth mentioning that this algorithm can be used to learn Lyapunov functions for high-dimensional systems as well, such as 3D quadrotor. But it also suffers from the lack of formal guarantees for the obtained Lyapunov function against the true dynamics, since the MILP is computed with the learned dynamics. However, it is interesting to notice that none of these works analyzed the closed-loop stability for unknown nonlinear systems in the general setting.

In the big picture, since neural Lyapunov functions are neural networks with Lyapunov conditions as output constraints, the verification of such networks is a branch of the field of neural network verification. We refer the readers to a more detailed survey in [73], where state-of-the-art verifiers of neural networks are discussed and categorized into three groups: reachability analysis-based, optimization-based, and search-based.

2.3.2 Lyapunov-Based Stability Analysis via Koopman Operator

As described in Section 2.1.1, the Koopman operator is an effective methodology to learn the unknown dynamics, while it also provides a promising way to generate the Lyapunov

function for global stability analysis. In [86], Mauroy et al. illustrate that the eigenfunctions of the Koopman operator yield a bunch of Lyapunov functions of the nonlinear systems with global stability. By combining this result and the autoencoder structure proposed in [3], the authors in [33] present an algorithm in which the observable functions are determined by optimizing the autoencoder, and the corresponding Koopman eigenfunctions parameterize several Lyapunov function candidates. Differing from the previous two papers, [151] uses a similar framework to learn the unknown dynamics but learns the Lyapunov function with a neural network proposed by [22], and then designs stabilizing controllers for unknown nonlinear systems with Sontag's formula for the lifted observable functions. Note that the obtained neural Lyapunov function and the corresponding stabilizing controller can only provide stability guarantees for the lifted Koopman dynamics, not the underlying unknown dynamics. Therefore, how to effectively verify the Lyapunov function candidates generated by Koopman eigenfunctions and how to conduct Lyapunov-based stability verification for the unknown dynamics with the Koopman operator have not been studied in any previous literature.

2.3.3 Other Learning-based Methods

GP regression is used in [7] to learn the unknown systems while providing safety and stability guarantees using quadratic Lyapunov functions for the states of partly known systems in the ROA with high probability. At the same time, how to enlarge the estimated ROA is also well studied. Furthermore, an extension work [9] by the same authors proposes a reinforcement learning framework in which the value function is the Lyapunov function while a neural network controller is trained with stability and safety guarantees. On top of that, a method to get a larger ROA by iteratively re-design a control Lyapunov function is presented in [88]. However, we have to contend with the safety-critical nature of certain control and robotics applications. As a result, worst-case bounds and performance guarantees are more desirable. In a similar vein, an upper bound on the infinity-norm bounds was established in the context of training deep residual networks [83]. Except for the above traditional machine learning methodologies, [36] proposed a piecewise learning framework with which both the unknown dynamics and Lyapunov functions with piecewise affine models, and the stability guarantees are verified by solving an optimization problem.

2.4 Summary

In this chapter, the literature on system identification and data-driven control, as well as Lyapunov-based stability analysis, is reviewed, with emphasis on the ones using learning-based methods. In the following chapters, before presenting the proposed technical contributions, we will also include a short literature review on that specific topic at the beginning of each chapter.

Chapter 3

Physics-Informed Extreme Learning Machine Lyapunov Functions

This chapter is an extended version of the work published in IEEE Control Systems Letters [148].

3.1 Introduction

Several longstanding challenges in nonlinear systems and control are tied to the computational difficulty of solving PDEs. For example, the celebrated HJB equation, which characterizes the optimal value function for a controlled system, is notoriously difficult to solve in high dimensions. A closely related topic is the construction of Lyapunov functions. Despite being a century-old idea, a general approach to effectively computing Lyapunov functions remains elusive. To address this challenge, computational methods have been investigated, notably including SOS [104] and radial basis function (RBF) [41] approaches.

More recently, due to the increased popularity of neural networks, a number of researchers have investigated the use of neural networks for computing Lyapunov functions [22, 42, 1, 38, 56, 150, 75] and solving HJB equations [48]. Most relevant to the current work, in [77, 76], the authors have demonstrated that physics-informed learning [116] with Zubov's equation [152] can outperform traditional SOS techniques [104] for computing Lyapunov functions in terms of verifiable ROA estimates. Intuitively, such improvements come from the universal approximation properties of neural networks and the characterization of the problem in terms of PDEs. Often, training neural networks involves embracing non-convex optimization.

However, a convex optimization formulation can significantly speed up training and leave no optimality gap because a global minimum can be easily achieved.

An extreme learning machine (ELM) [49] contrasts with traditional neural networks by randomly assigning weights and biases to hidden nodes and adjusting only the weights of the output layer. This approach transforms the optimization into a linear least squares problem that can be efficiently solved, exhibiting fast learning speed and strong generalization performance. In this paper, we explore the application of ELM in obtaining physics-informed neural networks to solve PDEs for computing Lyapunov functions, ROA estimates, and optimal control.

The contributions of the method proposed in this chapter are threefold. First, we theoretically analyze ELM for providing guarantees in practical stability analysis and ROA estimates. Second, we discuss using ELM to estimate the domain of attraction for nonlinear systems via solving Zubov's equation. Lastly, to the best of the authors' knowledge, this is the first work to thoroughly demonstrate the computational advantages of ELM in solving neural Lyapunov certificates and optimal control problems compared to existing approaches.

3.2 Preliminaries and Problem Formulation

Consider an autonomous system of form

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

and a control-affine system of the form

$$\dot{x} = f(x) + g(x)u, \tag{3.2}$$

$f : \mathbb{R}^d \to \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}^{d\times k}$. The state lies in $\mathbb{R}^d$ and the control input u lies in $\mathbb{R}^k$. For simplicity, we assume f and g are both continuously differentiable. We also assume $f(0) = 0$, the linearization of the system (3.1) is asymptotically stable, i.e., $A = Df(0)$ is Hurwitz, and the linearization of the control system (3.2) is stabilizable, i.e., (A, B) is stabilizable, where $B = g(0)$.

We are primarily concerned with the following computational tasks for systems (3.1) and (3.2) in this paper.

Computing Lyapunov functions: Given that $x = 0$ is an asymptotically stable equilibrium point for (3.1), how can we compute Lyapunov functions that not only characterize

asymptotic stability of the origin but also certify as large a region of attraction estimate as possible?

Computing domain of attraction: Related to the point above, how can we estimate the regions of attraction that can be verified to be contained in the true domain of attraction

$$\mathcal{D} := \left\{ x_0 \in \mathbb{R}^d : \lim_{t \to \infty} \|\phi(t, x_0)\| = 0 \right\},$$

where $\phi(t, x_0)$ is the solution to (3.1) starting from $x(0) = x_0$?

Computing optimal value and control: Define a cost

$$J(x_0, u) = \int_0^\infty Q(\phi(t, x_0, u)) + u^T(t) R(\phi(t, x_0, u)) u(t) dt, \tag{3.3}$$

where $\phi(t, x_0, u)$ is the solution to (3.2) under control signal u, $Q(x)$ and $R(x)$ are assumed to be positive definite functions of x. For simplicity, one can consider $Q(x) = x^T Q x$ and $R \in \mathbb{R}^{k \times k}$ for some positive definite matrices Q and R. Let $\mathcal{U}$ denote the set of admissible controls that can asymptotically stabilize the system with a finite cost. The objective is to find u^* that achieves the optimal value $V^*(x) = J(x, u^*) = \inf_{u \in \mathcal{U}} J(x, u)$. The function V^* is the optimal value function. In practice, we hope to compute accurate approximations to u^* and V^* and obtain a close-to-optimal controller that is also stabilizing.

3.3 Physics-Informed ELM Lyapunov Functions for Stability Analysis and Control

3.3.1 Neural Network Architecture

We consider an ELM that is essentially a single-hidden-layer feedforward neural network:

$$V(x; \beta) = \beta^T \sigma(Wx + b), \tag{3.4}$$

where $W \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, $\beta \in \mathbb{R}^m$, and σ is an activation function applied elementwise. While any activation function[1] is theoretically viable, we shall exclusively use the hyperbolic tangent function $\tanh(\cdot)$ in all our examples.

[1]To achieve universal approximation guarantees, a non-polynomial activation is often required. However, for practical purposes, polynomial activation functions can still be used, e.g., to obtain a polynomial network.

For the purpose of solving PDEs using ELM, we need to compute derivatives of V with respect to x, denoted as DV. It can be easily verified that the gradient can be computed as

$$DV(x;\beta) = \beta^T \mathrm{diag}(\sigma'(Wx+b))W, \tag{3.5}$$

where $\mathrm{diag}(\cdot)$ transforms the m-vector $\sigma'(Wx+b)$ into an $m \times m$ diagonal matrix, with $\sigma'(\cdot)$ denoting the derivative of σ, applied elementwise to $Wx+b$.

3.3.2 Physics-Informed Neural Network and Convex Optimization

A PINN [67, 116] is essentially a neural network for solving PDEs. It minimizes errors deemed important for forcing the network to approximate the true solution of the partial differential equation (PDE).

Consider a generic first-order PDE of the form:

$$F(x, V, DV) = 0, \quad x \in \Omega, \tag{3.6}$$

with the boundary condition $V = h$ on $\partial\Omega$. PINNs seek a neural network solution $V(x;\beta)$ to (3.6) by minimizing a loss function that encompasses the PDE and any supplementary information pertinent to the problem. The loss function is

$$\mathrm{Loss}(\beta) = \mathcal{L}_{\mathrm{residual}}(\beta) + \mathcal{L}_{\mathrm{boundary}}(\beta) + \mathcal{L}_{\mathrm{data}}(\beta), \tag{3.7}$$

where $\mathcal{L}_{\mathrm{residual}}$ targets the PDE's residual error, $\mathcal{L}_{\mathrm{boundary}}$ addresses the boundary condition error, and $\mathcal{L}_{\mathrm{data}}$ pertains to the error in data or any auxiliary information.

In this paper, we focus on utilizing the architecture (3.4) to formulate a convex optimization problem for optimizing $V(x;\beta)$. This is feasible if the loss function $\mathrm{Loss}(\beta)$ is convex with respect to β. Specifically, if the PDE given in (3.6) is linear in V and DV, and we employ the mean square error for $\mathcal{L}_{\mathrm{residual}}$, along with $\mathcal{L}_{\mathrm{boundary}}$ and $\mathcal{L}_{\mathrm{data}}$, then we are led to a linear least squares problem for optimizing β.

3.3.3 ELM Neural Lyapunov Functions via Lyapunov's Equation

Let $\Omega \subset \mathbb{R}^d$ be any open set containing the origin and contained in the domain of attraction $\mathcal{D}$. A Lyapunov function for (3.1) can be characterized by the following PDE:

$$\dot{V}(x) := DV(x) \cdot f(x) = -\omega(x), \tag{3.8}$$

where ω is a positive definite function with respect to the origin, i.e., $\omega(0) = 0$ and $\omega(x) > 0$ for all $x \in \Omega \setminus \{0\}$. We consider a single boundary condition $V(0) = 0$. We refer interested readers to [77] for technical results on well-posedness and solution properties for this PDE, both in the viscosity and classical solution senses.

Clearly, the PDE given by (3.8) is linear in DV, as we refer to it as Lyapunov's equation. Hence, we can formulate a convex optimization problem to solve it using an ELM neural network. For clarity, we explicitly present a mean square loss function as follows:

$$\text{Loss}(\beta) = \frac{1}{N}\sum_{s=1}^{N} \|DV(x_s;\beta)\cdot f(x_s) + \omega(x_s)\|^2 + \lambda_0\|\beta^T\sigma(b)\|^2, \tag{3.9}$$

where $\{x_s\}_{s=1}^N \subset \Omega$ is a set of collocation points and $\lambda_0 > 0$ is a weight parameter for the condition $V(0) = 0$. The loss function loss(β) is clearly a quadratic function of β. Solving this to determine β provides an approximate solution to (3.8) in the form of (3.4). This is summarized in Algorithm 1.

Algorithm 1: ELM Neural Lyapunov Function via Lyapunov's equation

Require: f, Ω, N, m

1: Generate random W and b
2: Generate random collocation points $\{x_s\}_{s=1}^N \subset \Omega$
3: Finding β that minimizes (3.9) to form V from (3.4)

We next state a technical result that says if we solve (3.8) well in an approximate sense, then the function obtained as an approximate solution to (3.8) is a Lyapunov function for (3.1) in a practical sense, to be made precise below.

Proposition 3.3.1. *Suppose that there exist two sequences $\{\varepsilon_R^n\} \downarrow 0$, $\{\varepsilon_0^n\} \downarrow 0$ (approaching 0 from above as $n \to \infty$), and a sequence of continuously differentiable functions $\{V_n\}$, where each $V_n : \mathbb{R}^d \to \mathbb{R}$ such that*

$$|DV_n(x)\cdot f(x) + \omega(x)| \le \varepsilon_R^n, \quad |V_n(0)| \le \varepsilon_0^n,$$

for all $x \in \Omega$. Furthermore, suppose that the sequence V_n is equicontinuous[2] at $x = 0$. Then, there exists a sequence $\mu_n \to 0$ such that, for all n sufficiently large, V_n is a practical Lyapunov function in the following sense:

$$DV_n(x)\cdot f(x) < 0, \quad V_n(x) > 0, \tag{3.10}$$

[2]This holds, in particular, when all V_n share a uniform Lipschitz constant or modulus of continuity around zero.

for all $x \in \Omega$ such that $\|x\| \geq \mu_n$.

Proof. According to [58, Theorem 4.18], V_n satisfying the stated condition (3.10) leads to solutions being ultimately bounded, with the ultimate bound arbitrarily small, as $\mu_n \to 0$. This can be interpreted as a practical notion of stability.

Since ω is positive definite, there exists a class $\mathcal{K}$ function that $\omega(x) \geq \eta(\|x\|)$ for all $x \in \Omega$. Pick $r_n > 0$ such that $\eta(r_n) > \varepsilon_R^n$. Then

$$DV_n(x) \cdot f(x) \leq \varepsilon_R^n - \omega(x) \leq \varepsilon_R^n - \eta(r_n) < 0 \tag{3.11}$$

for all $x \in \Omega$ such that $\|x\| > r_n$. Note that we can choose $r_n \downarrow 0$ as $\varepsilon_R^n \downarrow 0$. Let $m_{r_n} = \min_{\|x\|=r_n} V_n(x)$. By the equicontinuity of the V_n's at 0, the fact that $r_n \downarrow 0$ and $|V_n(0)| \leq \varepsilon_0^n \downarrow 0$, we have $m_{r_n} \to 0$.

Pick any $\mu_0 > 0$ such that the closed Euclidean ball of radius μ_0, $\bar{B}_{\mu_0}$, has $\bar{B}_{\mu_0} \subset \Omega \subset \mathcal{D}$. Let $M_f = \max_{x \in \mathcal{R}(\bar{B}_{\mu_0})} \|f(x)\|$, where $\mathcal{R}(\bar{B}_{\mu_0}) := \phi([0, \infty) \times \bar{B}_{\mu_0})$ stands for the forward reachable set of (3.1) from $\bar{B}_{\mu_0}$, which is known to be compact [72]. Pick $\mu_n \in (r_n, \mu_0]$ such that

$$-\frac{\mu_n - r_n}{M_f}(\varepsilon_R^n - \eta(r_n)) + m_{r_n} > 0. \tag{3.12}$$

Clearly, for n sufficiently large, this is always possible, and we can choose $\mu_n \to 0$. We claim that $V_n(x) > 0$ for all $x \in \Omega$ such that $\|x\| > \mu_n$. To prove this, first note that it takes at least $T \geq \frac{\mu_n - r_n}{M_f}$ units of time to reach from x to the sphere $\|x\| = r_n$. Let T be the first hit time of $\phi(t, x)$ reaching the sphere $\left\{y \in \mathbb{R}^d : \|y\| = r_n\right\}$. Then, we have $\|\phi(T, x)\| = r_n$ and $T \geq \frac{\mu_n - r_n}{M_f}$. By (3.11) and (3.12),

$$\begin{aligned} V_n(x) &= V_n(\phi(T, x)) - \int_0^T DV_n(\phi(s, x)) \cdot f(\phi(s, x)) \\ &\geq m_{r_n} - T(\varepsilon_R^n - \eta(r_n)) \\ &\geq m_{r_n} - \frac{\mu_n - r_n}{M_f}(\varepsilon_R^n - \eta(r_n)) > 0, \end{aligned}$$

for all $x \in \Omega$ such that $\|x\| > \mu_n$. □

Remark 3.3.2. *The proposition above describes: (i) a situation where a sequence of $\{V_n\}$ can be found to solve the Lyapunov's PDE (3.8) with increasing accuracy, e.g., by increasing the number of hidden units m and the number of collocation points N; (ii) what stability guarantees can be obtained by one of these functions as a Lyapunov function.*

3.3.4 ELM Neural Lyapunov Functions via Zubov's Equation

The result in the previous section provided a systematic way to construct a Lyapunov function on an a priori specified set *within* the domain of attraction. In this section, we use ELM to find Lyapunov functions that can provide ROA estimate close to the true domain of attraction.

The domain of attraction of the equilibrium point $x = 0$ of (3.1) is captured by the Zubov's equation of the form

$$DV \cdot f = -\omega\psi(V)(1 - V), \tag{3.13}$$

where ω is defined above and ψ is a function satisfying some technical assumptions [152] (see also [20, 56, 77]). Two special cases of $\psi(s)$ are given by (i) $\psi(s) = \alpha$ or (ii) $\psi(s) = \alpha(1 + s)$ for some constant $\alpha > 0$. In order to obtain a linear PDE, we choose $\psi(s) = \alpha$ for some constant $\alpha > 0$, which gives a linear Zubov's equation:

$$DV(x) \cdot f(x) = -\alpha\omega(x)(1 - V(x)). \tag{3.14}$$

We refer interested readers to [77] for the well-posedness of the PDE and properties of its solutions in both the viscosity and classical senses. Let V be the solution to (3.14). By Zubov's theorem, the domain of attraction $\mathcal{D}$ is given by the sub-level set $\mathcal{D} = \{x \in \mathbb{R}^d : V(x) < 1\}$. Hence, obtaining an accurate neural solution of (3.14) implies an accurate approximation of the domain of attraction [77, 76].

We next state an ELM algorithm for solving (3.14). For the sake of completeness, we write down an explicit loss similar to the loss (3.9), but for Zubov's equation (3.14), as follows:

$$\begin{aligned}\text{Loss}(\beta) =& \frac{1}{N}\sum_{s=1}^{N} \|DV(x_s;\beta) \cdot f(x_s) - \alpha\omega(x_s)V(x_s;\beta) + \alpha\omega(x_s)\|^2 \\ &+ \lambda_b \frac{1}{N_b}\sum_{s=1}^{N_b} \|V(y_s;\beta) - 1\|^2 + \lambda_0\|\beta^T\sigma(b)\|^2,\end{aligned} \tag{3.15}$$

where both $DV(\cdot;\beta)$ and $V(\cdot;\beta)$, detailed expressions of which can be found in (3.5) and (3.4), are linear in β. We added a new term $\lambda_b \frac{1}{N_b}\sum_{s=1}^{N_b} \|V(y_s;\beta) - 1\|^2$, $\lambda_b > 0$, to describe the boundary condition. In this formulation, we assume $\mathcal{D} \subset \Omega$, i.e., the domain of attraction is entirely contained within Ω. Hence, at the boundary points $\{y_s\}_{s=1}^{N_b} \subset \partial\Omega$, we should have $V(y_s;\beta) = 1$. Thus, optimizing β remains a linear least squares problem. We state the process of solving (3.14) with an ELM neural network in Algorithm 2.

Remark 3.3.3. *A data loss of the form* $\frac{\lambda_d}{N_d}\sum_{s=1}^{N_d}\|V(z_s;\beta)-\hat{V}(z_s)\|^2$ *can potentially be added to the loss function, which does not affect the linearity of the loss with respect to* β*. A loss like this can be beneficial when the domain of attraction is not entirely contained in* Ω*. In that case, specifying values of* V *to be equal to 1 on the boundary of* Ω *does not conform to the solution of Zubov's equation. We note that in [56], a purely data-driven approach is taken to compute neural network Lyapunov functions, without taking into account any PDE loss or formal verification, whereas in [75, 77], both data and PDE losses are combined to achieve better approximation in certain numerical examples, as verified by SMT solvers. In this paper, we show that the ELM encoding without any data loss can already achieve good results, as shall be demonstrated with numerical examples.*

Algorithm 2: ELM Neural Lyapunov Function via Zubov's equation

Require: f, Ω, N, N_b, m

1: Generate random W and b
2: Generate random collocation points $\{x_s\}_{s=1}^{N}\subset\Omega$
3: Generate random boundary points $\{y_s\}_{s=1}^{N_b}\subset\partial\Omega$
4: Find β that minimizes (3.15) to form V from (3.4)

The next technical result states the theoretical guarantees of solving Zubov's equation well using approximations.

Proposition 3.3.4. *Let* Ω *be bounded. Suppose that* $\mathcal{D}\subset\Omega$ *and there exist sequences of numbers* $\{\varepsilon_R^n\}\downarrow 0$, $\{\varepsilon_b^n\}\downarrow 0$, $\{\varepsilon_0^n\}\downarrow 0$, *and a sequence of continuously differentiable functions* $V_n:\mathbb{R}^d\to\mathbb{R}$ *such that*

$$|DV_n(x)\cdot f(x)-\alpha\omega(x)V_n(x)+\alpha\omega(x)|\le\varepsilon_R^n,\forall x\in\Omega,$$
$$|V_n(0)|\le\varepsilon_0^n,\quad |V_n(x)-1|\le\varepsilon_b^n,\forall x\in\partial\Omega,$$

where $\alpha>0$ *is a constant. Furthermore, suppose that* $\{V_n\}$ *is equicontinuous at* $x=0$*. Then there exist sequences* $c_n\uparrow 1$ *and* $\mu_n\to 0$ *such that, for all* n *sufficiently large,* V_n *uniformly converges to the true solution* V *to (3.14) on* $\bar{\Omega}$ *and satisfies (3.10) for all* $x\in\Omega$ *such that* $V_n(x)\le c_n$ *and* $\|x\|\ge\mu_n$*.*

Proof. The proof relies on the convergence result established in [77, Proposition 4] for approximate solutions of (3.14) in a more general setting. By the assumptions on V_n, Proposition 4 in [77] guarantees that V_n converges uniformly to the true solution V to (3.14) on $\bar{\Omega}$. We know that V is a Lyapunov function for (3.1), with $V(0)=0$ and $0<V(x)\le 1$ for $x\in\Omega$. Furthermore, we have $\mathcal{D}=\{x\in\mathbb{R}^d:V(x)<1\}$.

We first apply analysis in the proof of Proposition 3.3.1 to the set $\Omega' = \left\{x \in \mathbb{R}^d : V(x) < \frac{1}{2}\right\} \subset \Omega$. On this set, we have $V_n \to V$ uniformly. Hence, for n sufficiently large, we have

$$DV_n(x) \cdot f(x) \leq -\alpha\omega(x)(1 - V(x)) + \varepsilon_R^n \leq -\frac{\alpha}{4}\omega(x) + \varepsilon_R^n.$$

Similar to the proof of Proposition 3.3.1, there exists a sequence $\mu_n \to 0$ such that, for all n sufficiently large,

$$DV_n(x) \cdot f(x) < 0, \quad V_n(x) > 0, \tag{3.16}$$

for all $x \in \Omega'$ such that $\|x\| \geq \mu_n$. Furthermore, the choice of μ_n can be guaranteed to satisfy $\eta(\mu_n) > e_R^n$, where η is a class $\mathcal{K}$ function such that $\frac{\alpha}{4}\omega(x) \geq \eta(\|x\|)$ for all $x \in \bar{\Omega}$.

Now choose $c_n \in (0, 1)$ such that $-\alpha w(x)(1 - c_n) + e_R^n < 0$ for all $x \in \bar{\Omega} \setminus \Omega'$. This is always possible for n sufficiently large by setting $c_n < 1 - \frac{e_R^n}{\min_{x \in \bar{\Omega} \setminus \Omega'} \alpha\omega(x)}$. Furthermore, we can choose $c_n \uparrow 1$ as $e_R^n \downarrow 0$. It follows that, for all $x \in \bar{\Omega} \setminus \Omega'$ such that $V_n(x) \leq c_n$, we have

$$\begin{aligned} DV_n(x) \cdot f(x) &\leq -\alpha\omega(x)(1 - V_n(x)) + \varepsilon_R^n \\ &\leq -\alpha\omega(x)(1 - c_n) + \varepsilon_R^n < 0. \end{aligned} \tag{3.17}$$

Combining (3.16) and (3.17), the proof is complete. □

Remark 3.3.5. *A simple Lyapunov argument indicates the existence of a sequence ν_n, such that $B_{\mu_n} \subset \{x \in \Omega : V_n(x) < \nu_n\}$. For large n, $DV_n(x) \cdot f(x) < 0$ whenever $\nu_n \leq V_n(x) \leq \mu_n$. An SMT solver can confirm this, ensuring that solutions starting within $\{x \in \Omega : \nu_n \leq V_n(x) \leq \mu_n\}$ eventually enter $\{x \in \Omega : V_n(x) < \nu_n\}$, a subset of the region of attraction for equation (3.1), further validated by local Lyapunov analysis (see [76]). As $c_n \to 1$ and V_n uniformly converges to V on $\bar{\Omega}$, the regions of attraction certified by V_n approach the actual domain of attraction $\mathcal{D}$.*

3.3.5 ELM Neural policy iteration (PI) for Optimal Control

In this section, we demonstrate that effectively solving Lyapunov's equation with ELM, as detailed in Section 3.3.3, can be leveraged for controller design. We examine the control-affine system (3.2) and the cost function (3.3).

The optimal value function for (3.1) with the associated cost (3.3) is determined by the HJB equation:

$$\inf_{u \in \mathbb{R}^k} \left\{Q(x) + u^T R(x) u + DV(x) \cdot (f(x) + g(x)u)\right\} = 0. \tag{3.18}$$

Since the above is quadratic in u, the minimization is achieved by

$$u = \kappa(x) = -\frac{1}{2}R^{-1}(x)g^T(x)DV^T(x), \tag{3.19}$$

where $R^{-1}(x)$ denotes the inverse of $R(x)$, and $DV(x)$ represents the gradient of V in row vector format. With (3.19), the HJB equation reduces to

$$0 = Q(x) + DV(x) \cdot f(x) - \frac{1}{4}DV(x)g(x)R^{-1}(x)g^T(x)DV^T(x). \tag{3.20}$$

Despite the simplification, the HJB equation (3.20) remains nonlinear in V, making the task of finding its solution particularly difficult, especially for systems with high dimensions. PI offers a solution to this challenge by iteratively solving a simpler, linear PDE instead.

PI imitates the challenge of directly solving the HJB equation by iteratively solving a simpler, linear PDE instead. The PI algorithm starts with an initial controller $u = \kappa_0(x)$, which is assumed to be an asymptotically stabilizing controller. For each $i \geq 0$, it repeats

1. (**policy evaluation**) Compute a value function $V_i(x)$ for the controller κ_i by solving the Lyapunov-type PDE

$$DV_i(x) \cdot (f(x) + g(x)\kappa_i(x)) = -Q(x) - \kappa_i(x)^T R(x)\kappa_i(x) \tag{3.21}$$

   subject to $V_i(0) = 0$.

2. (**policy improvement**) Update the controller using

$$\kappa_{i+1}(x) = -\frac{1}{2}R^{-1}(x)g^T(x)DV_i^T(x). \tag{3.22}$$

Clearly, the challenge lies in solving (3.21). Note that, by

$$\begin{aligned} f_i(x) &:= f(x) + g(x)\kappa_i(x), \\ \omega_i(x) &:= Q(x) + \kappa_i(x)^T R(x)\kappa_i(x), \end{aligned}$$

the PDE (3.21) involved in the PI algorithm reduces exactly to (3.8) with f replaced by the closed-loop dynamics f_i under k_i. We can use ELM to solve it by convex optimization at each iteration step i. We summarize this in Algorithm 3.

Algorithm 3: Extreme Learning Machine Policy Iteration (ELM-PI)

Require: f, g, Q, R, k_0, Ω, N, m

1: **repeat**
2: Generate random W and b
3: Generate random $\{x_s\}_{s=1}^{N} \subset \Omega$
4: Finding β that minimizes (3.9) with $f = f_i$ to form V_i from (3.4)
5: Update κ_{i+1} according to (3.22)
6: $i = i + 1$
7: **until** desired accuracy or max iterations reached

Remark 3.3.6. *Physics-informed neural networks were proposed for PI in [92]. Here, we highlight the efficient use of ELM for PI as a result of efficiently solving Lyapunov's PDE (3.8) (see comparison in Section 3.4.3). Moreover, as demonstrated in [92], exact-PI converges uniformly (on compact subsets) to the unique viscosity solution of the HJB equation (3.18) [6], while in each policy-evaluation step, the value function in ELM-PI converges to the one in exact-PI in a uniform sense. Consequently, by the triangle inequality and the convergence of exact-PI, the ELM-PI value function converges uniformly to the viscosity solution of (3.18), and the associated optimal control policy converges uniformly to the one corresponding to the HJB equation.*

3.4 Numerical Experiments

In this section, we present a set of examples to illustrate the potential advantages of utilizing the ELM approach for computing neural Lyapunov functions and optimal control, including fast learning speed and strong generalization performance. All examples are solved using the tool LyZNet [76], a Python tool that facilitates physics-informed learning of Lyapunov functions [77] and nonlinear optimal control [92], incorporating formal verification through SMT solvers [40]. Computations were mostly performed on a 2020 MacBook Pro with a 2 GHz Quad-Core Intel Core i5 and, for GPU-required comparisons, on an NVIDIA Hopper H100 SXM. Code, model equations, and parameters are available at `https://git.uwaterloo.ca/hybrid-systems-lab/lyznet/`. The test error for numerical examples is the maximum residual error over $2 \times N$ random points, where N is the number of collocation points.

3.4.1 Local Stability Analysis

We use a low-dimensional system, the inverted pendulum with linear control (see Example 3.4.1), to study the impact of increasing the size of the training domain and a high-dimensional system, Example 3.4.2 [42], to investigate the impact of increasing the system's dimension on the difficulty of computing an ELM Lyapunov function. In all cases, we compute neural Lyapunov functions by solving Lyapunov's equation (3.8) using Algorithm 1 with $\omega(x) = \|x\|^2$.

In Table 3.1, it is evident that ELM can solve Lyapunov's PDE with significant accuracy. We note that the accuracy is measured in terms of residual error for satisfying the characterizing PDE (3.8), in contrast with existing neural network Lyapunov functions in the literature [42, 38, 40, 150], which use Lyapunov inequalities as loss. This crucial difference is important when it comes to ROA estimates and optimality. As the domain expands, maintaining the same accuracy level becomes increasingly difficult. Despite this, it consistently attains accuracy comparable to that of multi-layer PINNs trained via gradient descent methods, typically around 1E-3. Table 3.2 demonstrates ELM's effectiveness in solving high-dimensional problems, achieving accuracies from 1E-3 to 1E-5 for dimensions between 10 and 30.

Example 3.4.1 (Inverted pendulum with linear control)**.** *Consider an inverted pendulum*

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \sin(x_1) - x_2 - (k_1 x_1 + k_2 x_2), \end{aligned} \tag{3.23}$$

where the linear gains are given by $k_1 = 4.4142$ *and* $k_2 = 2.3163$.

Example 3.4.2 (A simple high-dimensional system)**.** *Consider the 10-dimensional nonlinear system from [42]. We construct 20- and 30-dimensional systems by 2 and 3 copies of the*

same 10-dimensional system, respectively.

$$
\begin{aligned}
\dot{x}_1 &= -x_1 + 0.5x_2 - 0.1x_9^2,\\
\dot{x}_2 &= -0.5x_1 - x_2,\\
\dot{x}_3 &= -x_3 + 0.5x_4 - 0.1x_1^2,\\
\dot{x}_4 &= -0.5x_3 - x_4,\\
\dot{x}_5 &= -x_5 + 0.5x_6 + 0.1x_7^2,\\
\dot{x}_6 &= -0.5x_5 - x_6,\\
\dot{x}_7 &= -x_7 + 0.5x_8,\\
\dot{x}_8 &= -0.5x_7 - x_8,\\
\dot{x}_9 &= -x_9 + 0.5x_{10},\\
\dot{x}_{10} &= -0.5x_9 - x_{10} + 0.1x_2^2.
\end{aligned}
$$

Domain	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
$[-1, 1]$	100	3,000	1.32E-5	0.02
$[-1, 1]$	200	3,000	8.62E-9	0.05
$[-2, 2]$	200	3,000	2.78E-4	0.05
$[-2, 2]$	400	3,000	2.81E-6	0.1
$[-2, 2]$	800	6,000	8.52E-8	0.5
$[-4, 4]$	800	6,000	5.03E-5	0.5
$[-4, 4]$	1,600	6,000	2.76E-6	1.7
$[-8, 8]$	3,200	6,000	6.49E-3	8.8
$[-8, 8]$	3,200	12,000	1.46E-3	12

Table 3.1: Training results for the inverted pendulum.

A Discussion on Learning Stability Certificates using PINNs and ELM

As demonstrated in [92, Remark 3.1], ELM is more computationally efficient than PINNs when it comes to learning Lyapunov functions for low-dimensional systems, while PINNs are able to handle high-dimensional cases more effectively. In the previous example, we demonstrated the efficiency of ELM for a two-dimensional system, with results verifiable using dReal [40]. However, in the cases where deeper and wider neural networks are needed

Dim (d)	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
10	1,600	6,000	1.59E-4	2
10	3,200	6,000	3.59E-5	9.3
10	3,200	12,000	3.61E-5	13
10	6,400	12,000	1.13E-5	72
20	1,600	6,000	5.58E-3	2.3
20	3,200	6,000	3.17E-3	10.6
20	3,200	12,000	3.34E-3	16.4
20	6,400	12,000	1.20E-3	127
30	1,600	6,000	1.61E-2	2.8
30	3,200	6,000	1.38E-2	11.2
30	3,200	12,000	6.67E-3	22.5
30	6,400	12,000	4.99E-3	172

Table 3.2: Training results for toy high-dimensional systems [42]. We restrict the domain to $[-0.1, 0.1]^d$.

and/or one needs to solve the stability certifications for high-dimensional systems, the limited scalability of dReal necessitates the use of more efficient verification tools. As a next step, we show that stability certificates for high-dimensional systems can also be learned using PINNs, with formal guarantees provided by the state-of-the-art neural network verifier α, β-CROWN [146, 138].

In the following 4-dimensional power system example in Example 3.4.3, we use a PINN which has only 1-hidden layer with $m = 400$ hidden units to learn a Lyapunov function by solving Zubov's equation. The loss function here is the same as the one in Section 3.3.4. As depicted in Fig. 3.1, the learned Lyapunov function captures a much larger ROA estimate than the quadratic one. It is worth mentioning that the results are formally verified by α, β-CROWN with the Lyapunov conditions.

We would like to emphasize that, with the same number of hidden units, ELM cannot return satisfactory results on this example. In our experiments, a width of $m = 3200$ is required to match the PINN solution, but such a wide network poses challenges for formal verification, even with state-of-the-art neural network verifiers. While multiple factors may be involved, two main plausible contributors are as follows. First, ELM essentially solves a least squares problem with a fixed set of hidden-layer features. Once the system is strongly overdetermined, adding more collocation points to minimize (3.7) yields diminishing returns.

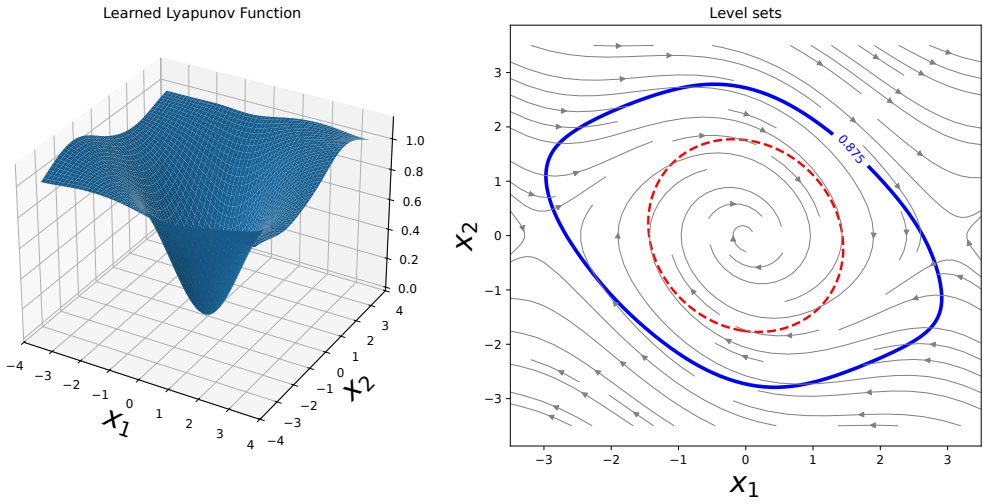


Figure 3.1: Neural Lyapunov function and the corresponding safe ROA (blue curve on the right) for the 4D power system, where the red dashed line represents the safe ROA obtained by the quadratic Lyapunov function.

In contrast, PINNs can take advantage of the additional collocation points to attain a better optimum via gradient descent. In this example, we set $N = 2e7$ for training PINNs. Second, the bias term in the output layer in the PINN solution helps to construct a better Lyapunov function in this case.

Example 3.4.3 (4D power system)**.** *Consider the following 4-dimensional power system consisting of two generators:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \alpha_1 \sin(x_{1,k}) - \beta_1 \sin(x_{1,k} - x_{3,k}) - d_1 x_{2,k} \\ x_4 \\ -\alpha_2 \sin(x_{3,k}) - \beta_2 \sin(x_{3,k} - x_{1,k}) - d_2 x_{4,k} \end{bmatrix},$$

where $\alpha_1 = \alpha_2 = 1$, $\beta_1 = \beta_2 = 0.5$, $d_1 = 0.4$, $d_2 = 0.5$.

3.4.2 Region-of-Attraction Estimates

We use the simple pendulum example to demonstrate the efficiency and advantages of using ELM Lyapunov functions for ROA estimates by solving Zubov's equation (3.14) using

Algorithm 2 in Section 3.3.4. The dynamics of the system are as follows.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{l}\sin(x_1) - bx_2, \end{aligned} \tag{3.24}$$

where $g = l = 9.8$, and $b = 0.1$. We verify the regions of attraction certified by the ELM Lyapunov function using dReal [40] and compare the result with the state-of-the-art approach, SOS semidefinite programming.

We train ELM Lyapunov functions by solving Zubov's equation (3.14) with $m = 100$ and $m = 200$ hidden units, which takes 20 to 50 milliseconds, respectively. We summarize the verified regions of attraction in Table 3.3, compared with that obtained with a sixth degree SOS Lyapunov function and a Taylor approximation to the vector field. ELM significantly outperforms SOS on this example.

ELM ($m = 100$)	ELM ($m = 200$)	SOS
49.06%	**81.01%**	25.62%

Table 3.3: Volume percentage of the verified region of attraction for a simple pendulum.

3.4.3 Optimal Control

We demonstrate with benchmark nonlinear control problems—including inverted pendulum, cartpole, 2D quadrotor, and 3D quadrotor—that ELM-PI effectively solves nonlinear optimal control problems across low- to high-dimensional systems. The dynamics of these models are listed in Examples 3.4.4 - 3.4.7. We solve all examples with ELM-PI as outlined in Algorithm 3 of Section 3.3.5 with 10 iterations. The numbers of hidden units and collocation points are detailed below.

We compare the ELM approach with the traditional SGA method [5] and the PINN [92] on the inverted pendulum example over the domain $[-1, 1]^2$. For PINN-PI, we train networks with one or two hidden layers of m units for 10,000 epochs to ensure an accurate solution to the linear PDE (3.21). For ELM-PI and one-layer PINN-PI, we set $N = d \times m$, where $d = 2$ is the system's dimension, and $N = (d + m) \times m$ for two-layer PINN-PI. PINN-PI was evaluated with and without GPUs. Table 3.4 shows that ELM-PI (on CPUs) provides a significant speedup, being at least 200 times faster than PINN-PI on advanced GPUs. It also demonstrates greater data efficiency and improved generalization. Additionally, SGA seems to suffer from divergence with higher-order bases.

successive Galerkin approximation (SGA)			ELM-PI			PINN-PI		PINN-PI with GPU		PINN-PI (2-hidden) with GPU	
Order	Test error	Time (s)	m	Test error	Time (s)	Test error	Time (s)	Test error	Time (s)	Test error	Time (s)
2	0.15	4.80	50	**0.03**	**0.02**	0.10	79.17	0.10	113.02	0.04	143.80
4	**3.4E-3**	19.37	100	0.01	**0.05**	0.02	93.54	0.03	113.99	0.10	143.56
6	0.01	66.52	200	**2.18E-5**	**0.16**	0.04	150.98	0.04	113.22	0.13	205.47
8	2.79	212.42	400	**2.3E-6**	**0.52**	0.07	330.48	0.03	114.03	0.24	1582.15

Table 3.4: Comparison of training/computational time for ELM-PI, PINN-PI [92], and SGA [5] on the inverted pendulum example: we run ELM-PI and PINN-PI with $m = 50, 100, 200, 400$ and SGA [5] with polynomial bases of order 2, 4, 6, 8.

Example 3.4.4 (Inverted pendulum)**.** *Consider the inverted pendulum model:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 19.6 \sin x_1 - 4.0 x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 40 \end{bmatrix} u. \tag{3.25}$$

Example 3.4.5 (Cartpole)**.** *Consider the Cartpole model:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{-mLx_2^2 \sin(x_1)\cos(x_1) + (M+m) g \sin(x_1)}{L(M + m\sin^2(x_1))} \\ x_4 \\ \frac{m \sin(x_1)(Lx_2^2 - g\cos(x_1))}{M + m\sin^2(x_1)} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{-\cos(x_1)}{L(M+m\sin^2(x_1))} \\ 0 \\ \frac{1}{M+m\sin^2(x_1)} \end{bmatrix} u.$$

Example 3.4.6 (2D quadrotor)**.** *Consider the 2D quadrotor model:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{g}{m}\sin(x_3) \\ -\frac{g}{m} + \frac{g}{m}\cos(x_3) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\sin(x_3)}{m} & 0 \\ \frac{\cos(x_3)}{m} & 0 \\ 0 & \frac{1}{I_m} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Example 3.4.7 (3D quadrotor)**.** *Consider the 3D quadrotor model:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ -g\sin(x_8) \\ g\cos(x_8)\sin(x_7) \\ g\cos(x_8)\cos(x_7) - g \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\sin(x_8) & 0 & 0 & 0 \\ \cos(x_8)\sin(x_7) & 0 & 0 & 0 \\ \cos(x_8)\cos(x_7) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}.$$

The values of the parameters of the models above can be found in the repository of the code for this chapter. Table 3.5 summarizes the performance of ELM-PI in solving benchmark nonlinear control problems, where the accuracy of the solution increases as m and/or N increase. It is noted that the traditional SGA approach can only solve the cartpole problem within a very small region [-0.05,0.05] and requires considerably more time. Therefore, a comparison is not provided. Readers are referred to [92] for additional experiments on PINN-PI, which, as demonstrated by the simple pendulum example, take significantly longer to train.

System	Domain	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
cartpole	$[-0.2, 0.2]^4$	400	3,000	3.95E-2	1.6
cartpole	$[-0.2, 0.2]^4$	800	3,000	3.98E-3	3.99
cartpole	$[-0.2, 0.2]^4$	1,600	3,000	1.63E-3	14.95
cartpole	$[-0.2, 0.2]^4$	3,200	6,000	6.43E-4	148.58
cartpole	$[-0.2, 0.2]^4$	6,400	9,000	1.51E-4	237.41
2D quad	$[-0.2, 0.2]^6$	800	6,000	4.00E-2	9.19
2D quad	$[-0.2, 0.2]^6$	1,600	9,000	1.57E-2	35.05
2D quad	$[-0.2, 0.2]^6$	3,200	9,000	1.17E-3	142.28
2D quad	$[-0.2, 0.2]^6$	6,400	9,000	3.59E-4	1010.33
3D quad	$[-0.2, 0.2]^9$	3,200	12,000	5.33E-2	189.06
3D quad	$[-0.2, 0.2]^9$	6,400	12,000	3.64E-2	1008.57

Table 3.5: Training results for ELM-PI.

3.5 Summary

In this chapter, we propose a physics-informed ELM framework for computing neural network Lyapunov functions in the stability analysis and control of nonlinear systems by solving the characterizing PDEs. This framework leads to a straightforward convex optimization problem solvable as a linear least squares problem. We theoretically analyze the guarantees of approximation solutions as practical Lyapunov functions that can provide arbitrarily precise ultimate bounds and ROA estimates, which are close to the true domain of attraction. We also demonstrate the effectiveness and computational efficiency of the proposed framework through a range of examples.

Chapter 4

Neural Lyapunov Control of Unknown Nonlinear Systems with Stability Guarantees

This chapter is based on the paper: Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. "Neural Lyapunov control of unknown nonlinear systems with stability guarantees". In Advances in Neural Information Processing Systems, 2022 [150].

4.1 Introduction

As discussed in Chapter 1, artificial neural networks are capable of estimating any nonlinear function with arbitrarily desired accuracy [46]. As both the right-hand side of a differential equation and the Lyapunov function can be regarded as nonlinear (or linear) functions, mapping from the state spaces to some vector spaces, a natural question is: can we use neural networks to approximate the dynamics and find a Lyapunov function to attain a larger estimate of the ROA, compared with widely used LQR and SOS methods [104, 60]?

In this chapter, by exploiting the expressiveness of neural networks to tackle the complex nonlinearities of dynamical systems, we propose a learning framework for simultaneously learning control functions and neural Lyapunov functions. We build upon the framework in [22], but address the more challenging problem of stabilizing a nonlinear system with unknown dynamics with formal stability guarantees. In addition, we complement the computational framework in [22], by proving the existence of a neural network satisfying

the Lyapunov conditions, except on an arbitrarily small neighborhood of the origin, which offers a theoretical basis for the learning framework here and in [22].

Our proposed learning framework consists of two neural networks. The first neural network learns the unknown dynamics, while the second one aims to identify a valid Lyapunov function and a stabilizing nonlinear controller. After training the neural networks, we derive error bounds that can be verified directly by the SMT solver. We then use Lipschitz continuity of the nonlinear dynamics and their neural approximation to prove rigorous theoretical guarantees on the closed-loop stability of the unknown system under the learned nonlinear controller. To the best knowledge of the authors, no prior work has provided closed-loop stability analysis in such a general setting of simultaneously learning the dynamics, a nonlinear controller, and a Lyapunov function.

Contributions. We summarize the main contributions of the method proposed in this chapter as follows.

- We propose a framework for simultaneously learning a nonlinear system, a stabilizing nonlinear controller, and a Lyapunov function for verifying the closed-loop stability of the unknown system.
- We provide theoretical analysis on the existence of a neural Lyapunov function that verifies the Lyapunov conditions, except on an arbitrarily small neighborhood of the origin, provided that the origin is asymptotically stable. This provides theoretical support for the proposed method as well as other neural Lyapunov frameworks in the literature.
- Combining classic Lyapunov analysis with rigorous error estimates for neural networks, we establish closed-loop stability for the unknown system under the learned controller. This is accomplished with the help of SMT solvers and recent tools for estimating Lipschitz constants for neural networks.

4.2 Preliminaries

We consider a nonlinear control system of the form

$$\dot{x} = f(x, u), \quad x(0) = x_0, \tag{4.1}$$

where $x \in \Omega$ is the state of the system, and $\Omega \subseteq \mathbb{R}^n$ is an open set containing the origin; $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the feedback control input given by $u = \kappa(x)$, where $\kappa(x)$ is a continuous function of x. Without loss of generality, we assume the origin is an equilibrium point of the closed-loop system

$$\dot{x} = f(x, \kappa(x)), \quad x(0) = x_0. \tag{4.2}$$

When there is no ambiguity, we also refer to the right-hand side of (4.2) by f.

We assume that we do not have explicit knowledge of the right-hand side of the system (4.1). The main objective is to stabilize the unknown dynamical system by designing a feedback control function κ. Stability guarantees are established using Lyapunov functions. We next present some preliminaries on model assumptions and Lyapunov stability analysis.

Assumption 4.2.1 (Lipschitz Continuity)**.** *The right-hand side of the nonlinear system (4.1) is assumed to be Lipschitz continuous, i.e.,*

$$\|f(x,u_1)-f(y,u_2)\| \leq L\|(x,u_1)-(y,u_2)\| \quad \forall x,y \in \Omega \quad \text{and} \quad \forall u_1,u_2 \in \mathcal{U},$$

where L is called the Lipschitz constant; (x,u_1) and (y,u_2) denote the concatenation of the corresponding two vectors. We assume the Lipschitz constant L is known.

Assumption 4.2.2 (Partly Known Dynamics)**.** *The linearized model about the origin in the form of $\dot{x} = \mathbf{A}x + Bu$, where $\mathbf{A}$ and B are constant matrices, is known for the nonlinear system (4.1).*

To analyze the stability properties of the closed-loop system for (4.1) in the presence of uncertainty (due to the need to approximate the unknown dynamics), we introduce a more general notion of stability about a closed set A (see, e.g., [72], and the supplementary material for a definition). When $A = \{0\}$, this coincides with the standard notion of stability about an equilibrium point. Intuitively, set stability w.r.t. to A is measured by the closeness and convergence of solutions to the set A. To this end, define the distance from x to A by $\|x\|_A = \inf_{y\in A}\|x-y\|$.

Set invariance [12] plays an important role in our analysis.

Definition 4.2.3 (Region of Attraction)**.** *For a closed forward invariant set A that is uniformly asymptotically stable (UAS), the region of attraction is the set of initial conditions in Ω such that the solution for the closed-loop system (4.2) is defined for all $t \geq 0$ and $\|x(t)\|_A \to 0$ as $t \to \infty$.*

Remark 4.2.4. *Any set satisfying Definition 4.2.3 is called a region of attraction. The DOA is the largest set contained in Ω satisfying Definition 4.2.3.*

Definition 4.2.5 (Lie Derivatives)**.** *The Lie derivative of a continuously differentiable scalar function $V : \Omega \to \mathbb{R}$ over a vector field f and a nonlinear controller u is defined as*

$$\nabla_f V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i}\dot{x}_i = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x,u). \tag{4.3}$$

The Lie derivative measures the rate of change along the system dynamics.

The following is a standard Lyapunov theorem for the UAS property of a compact invariant set.

Theorem 4.2.6 (Sufficient Condition for UAS property)**.** *Consider the closed-loop nonlinear system (4.2). Let $A \subset \Omega$ be a compact invariant set of this system. Suppose there exists a continuously differentiable function $V : \Omega \to \mathbb{R}$ that is positive definite with respect to A, i.e.,*

$$V(x) = 0 \ \forall x \in A \text{ and } V(x) > 0 \ \forall x \in \Omega \setminus A, \tag{4.4}$$

and the Lie derivative is negative definite with respect to A, i.e.

$$\nabla_f V(x) < 0 \ \forall x \in \Omega \setminus A. \tag{4.5}$$

Then, A is UAS for the system.

See [72, 128] for the sufficiency and necessity of Lyapunov conditions for set stability under more general settings. The function V satisfying (4.4) and (4.5) is called a Lyapunov function with respect to A.

From the forward invariance of level sets of a Lyapunov function, it follows that these sets provide an estimate of the ROA (see [59]).

Lemma 4.2.7 (Region of Attraction with Lyapunov Functions)**.** *Suppose that V satisfies the conditions in Theorem 4.2.6. Denote*

$$V^c := \{x \in \Omega \mid V(x) \leq c\}.$$

For every $c > 0$, V^c is a region of attraction for the closed-loop system (4.2).

4.3 Learn and Stabilize Dynamics with Neural Lyapunov Functions

In this subsection, we show how to use two shallow neural networks to learn the dynamics and find a valid neural Lyapunov function. The first network is designed to learn the dynamics as shown in (4.1), and the second network aims to identify a valid neural Lyapunov function and its corresponding nonlinear controller. The algorithmic structure can be found in Fig. 4.1 and the pseudocode in Algorithm 4.

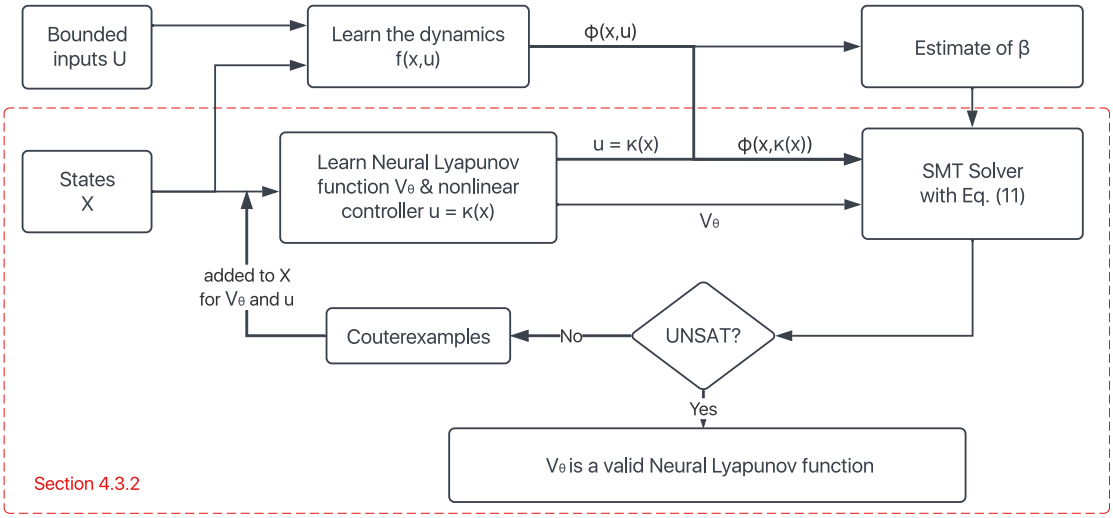


Figure 4.1: The schematic diagram of the proposed algorithm.

4.3.1 Learn the Unknown Dynamics

According to the well-known universal approximation theorem [28], shallow neural networks are able to approximate the right-hand side $f(x, u)$ of the nonlinear system (4.1) arbitrarily well. Here we use a one-hidden-layer feedforward neural network, denoted as $\phi(x, u)$, to conduct this regression task with a mean square loss. For notational convenience, we slightly overuse the notation of ϕ to denote both the learned dynamics and the parameters that define the function. Note that if we use the tanh or sigmoid function as activation functions, the output layer should not have such activation functions, as f is not bounded in most cases.

When learning the dynamics, f is regarded as a function of two variables, x and u. The input dimension of the neural network is $(n + m)$, and the output dimension is n. The training data of x and u are sampled uniformly and independently over their respective spaces. In most robotic systems, the input is provided by motors which are typically saturated in practice. To reflect this, we use the activation function tanh to simulate this property. The format of the controller is

$$u = \kappa(x) = C\sigma(kx + b), \tag{4.6}$$

where C is a constant matrix, determined by the saturation property of the controller in real systems, which defines the bounds of $\mathcal{U}$, k and b are the weight matrix and bias vector, respectively, which are initialized with an LQR controller based on the linearized

system $\dot{x} = \mathbf{A}x + Bu$. The bias b is chosen such that $f(0, C\sigma(b)) = 0$, i.e., the origin is an equilibrium point for the closed-loop system (4.2). Note that the bias vector b is not updated in the learning process.

4.3.2 Neural Lyapunov Function and Nonlinear Controller

Following learning the dynamics using ϕ, the other neural network is designed to learn a nonlinear controller and a neural Lyapunov function simultaneously during the same training process. The detailed structure can be found in Fig. 4.2. It is worth mentioning that since we train the Lyapunov function with data generated by the learned dynamics, which is an approximation of the actual dynamics, we rewrite (4.4) and (4.5) as follows to accommodate for approximation errors:

$$V(0) = 0, \text{ and } , \forall x \in \Omega \backslash \{0\}, V(x) > 0 \text{ and } \nabla_\phi V(x) < -\beta. \tag{4.7}$$

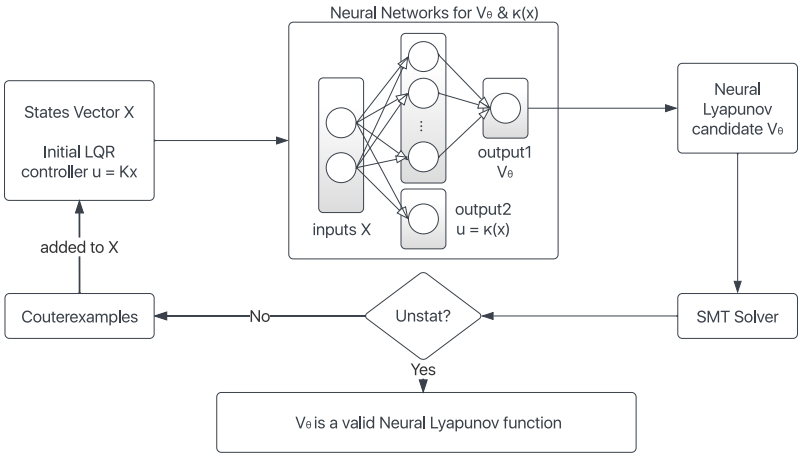


Figure 4.2: Algorithmic structure of learning a neural Lyapunov function and the corresponding nonlinear controller with a 1-hidden layer neural network and an SMT solver.

Here β is a positive real number, which can be determined as follows.

Assume that (x, u) is a pair of unsampled states and inputs, and (y, v) is its nearest sample used in training the neural network for learning the dynamics. Let $\delta > 0$ be such that $\|(x, u) - (y, v)\| \leq \delta$ holds for all such (x, u) and (y, v); and let $M > 0$ be such that $\|\frac{\partial V}{\partial x}\| < M$ for all $x \in \Omega$. Denote α as the maximum of the 2-norm loss among all training

data samples. Then, the bound on the testing error in terms of the training error can be calculated as

$$\begin{aligned}\|f(x,u)-\phi(x,u)\| &\leq \|f(x,u)-f(y,v)\|+\|f(y,v)-\phi(y,v)\|+\|\phi(y,v)-\phi(x,u)\| \\ &\leq K_f\delta+\alpha+K_\phi\delta<\frac{\beta}{M},\end{aligned} \tag{4.8}$$

where f and ϕ are Lipschitz continuous with Lipschitz constants K_f, K_ϕ, respectively, and $\beta>0$ is some sufficiently large constant. Then, choosing β to satisfy (4.8) implies that

$$\nabla_f V(x)-\nabla_\phi V(x)\leq \|\frac{\partial V}{\partial x}\|\|f(x,\kappa(x))-\phi(x,\kappa(x))\|<M\frac{\beta}{M}=\beta. \tag{4.9}$$

In view of (4.7) and (4.9), we have

$$\nabla_f V(x)<\nabla_\phi V(x)+\beta<-\beta+\beta<0,\quad \forall x\in\Omega\backslash\{0\}, \tag{4.10}$$

which implies that the actual dynamics is also stable with the obtained neural Lyapunov function.

Here M can be determined by checking the inequality with an SMT solver. An initial guess of β is needed according to some prior knowledge, and it will be re-computed in each epoch with (4.8).

A valid neural Lyapunov function can be guaranteed by the SMT solver with all the Lyapunov conditions written as falsification constraints in the form of first-order logic formula over reals [22]:

$$\Phi_\varepsilon(x):=\left(\sum_{i=1}^n x_i^2\geq\varepsilon\right)\wedge\left(V(x)\leq 0\vee\nabla_\phi V(x)\geq-\beta\right), \tag{4.11}$$

where ε is a numerical error parameter, which is explicitly introduced for controlling numerical sensitivity around the origin. If the SMT solver returns either `UNSAT`, this means that the falsification constraint is guaranteed not to have any solutions and confirms all the Lyapunov conditions are met. If the SMT solver returns δ-`SAT`, there exists at least one counterexample under the δ-weakening condition [39] that satisfies the falsification conditions.

We use θ to denote the parameter vectors for a neural Lyapunov function candidate V_θ. The parameters θ and k are found by minimizing the following cost function, which is

a modification of the so-called *empirical Lyapunov risk* in [22] by adding one more term $\|\frac{\partial V}{\partial x}\|$, as we need β to be small as well:

$$L(\theta, k) = \frac{1}{N}\sum_{i=1}^{N}\left(C_1 \max\left(-V_\theta\left(x_i\right), 0\right) + C_2 \max\left(0, \nabla_\phi V_\theta\left(x_i\right)\right)\right) + C_3 V_\theta^2(0) + C_4\|\frac{\partial V}{\partial x}\| \tag{4.12}$$

where C_1, C_2, C_3 and C_4 are tunable constants. The cost function can be regarded as the positive penalty of any violation of the Lyapunov conditions in (4.4) and (4.5). Note that the ROA can also be maximized by adding an L_2-like cost term to the *Lyapunov risk* with

$$L(\theta, k) + \frac{1}{N}\sum_{i=1}^{N}\|x_i\|_2 - \alpha V_\theta\left(x_i\right),$$

where α is a tunable parameter, as shown in the original paper [23].

4.4 Theoretical Guarantees

In this section, we analyze the theoretical guarantees of using neural networks to learn to control an unknown nonlinear system and a Lyapunov function for certifying the closed-loop stability.

4.4.1 Approximation Guarantee of Unknown Dynamics

Our analysis provides stability guarantees for the unknown system by rigorously quantifying the errors using Lipschitz constants of the unknown dynamics and their neural approximation. To this end, we need a theoretical guarantee that extends the universal approximation theorem, stating that we can approximate the Lipschitz constants and function values by a neural network to an arbitrary precision.

Theorem 4.4.1. *(Approximation of Lipschitz constants). Suppose that $K \subset \mathbb{R}^n$ is a compact set.*
(a) If $f : K \to \mathbb{R}^m$ is L-Lipschitz in the uniform norm, i.e.

$$\|f(x) - f(y)\|_\infty \leq L\|x - y\|_\infty, \tag{4.13}$$

Algorithm 4: Neural Lyapunov Control with Unknown Dynamics

Function `LearningDynamics(`$X_{dyn}, U_{(bdd)}$`):`
  Set learning rate γ, input dimension $(n+m)$, output dimension n;
  repeat
    $f \leftarrow \mathrm{NN}_\phi(x)$; `// Output of NN forward pass`
    Compute MSE $L(f, \phi)$;
    $\phi \leftarrow \phi - \gamma \nabla_\phi L(f, \phi)$; `// SGD weight update`
  until *convergence*;
  return ϕ;

Function `LearningLyapunov(`$X_{Lyp}, f_\phi, \mathcal{K}^{lqr}$`):`
  Set learning rate α, input dimension n, output dimension 1;
  Initialize feedback controller u to LQR solution $\mathcal{K}^{lqr}$;
  repeat
    $V_\theta(x), u(x) \leftarrow \mathrm{NN}_{\theta,u}(x)$; `// NN forward pass`
    $\nabla_\phi V(x) \leftarrow \sum_{i=1}^{D_{in}} \frac{\partial V}{\partial x_i}[\phi]_i(x)$;
    Compute Lyapunov risk $L(\theta, k)$;
    $\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta, k)$;
    $k \leftarrow k - \alpha \nabla_k L(\theta, k)$; `// SGD updates`
  until *convergence*;
  return V_ϕ, u;

Function `Falsification(`$f_\phi, u, V_\theta, \varepsilon, \delta, \beta$`):`
  Encode conditions from (4.11);
  Use SMT solver with δ to verify the conditions;
  return satisfiability;

Function `Main():`
  input: initial guess of bound (β), parameters of LQR $(\mathcal{K}^{lqr})$, radius (ε), precision (δ), sampled states X, sampled inputs U;
  $\phi \leftarrow$ `LearningDynamics(`$X_{dyn}, U_{(bdd)}$`)`;
  while *Satisfiable* **do**
    Add counterexamples to X;
    $V_\phi, u \leftarrow$ `LearningLyapunov(`$X_{Lyp}, \phi, \mathcal{K}^{lqr}$`)`;
    update β according to (4.8);
    CE $\leftarrow$ `Falsification(`$\phi, u, V_\theta, \varepsilon, \delta, \beta$`)`;

then for every $\epsilon > 0$ *there exists a neural network of the form* $\phi(x) = C(\sigma \circ (\omega x + b))$ *for* $\sigma \in C^1(\mathbb{R})$ *and not a polynomial,* $\omega \in \mathbb{R}^{k\times m}, b \in \mathbb{R}^k$ *and* $C \in \mathbb{R}^{k\times n}$ *for some* $k \in \mathbb{N}$ *such that* $\sup_{x\in K} |f(x) - \phi(x)| < \epsilon$ *and* ϕ *has a Lipschitz constant of* $L + \epsilon$ *in the same norms as (4.13).*
(b) If $f : K \to \mathbb{R}^m$ *is* L*-Lipschitz in the two norm, i.e.*

$$\|f(x) - f(y)\|_\infty \leq L\|x - y\|_2, \tag{4.14}$$

then for every $\epsilon > 0$ *there exists a neural network* ϕ *of the same form such that* $\sup_{x\in K} |f(x) - \phi(x)| < \epsilon$ *and* ϕ *has a Lipschitz constant of* $L + \epsilon\left(\frac{\sqrt{n+n/\epsilon}}{2} + L\right)$ *in the same norms as* (4.14).

The idea of the proof is to first approximate f by a smooth function F and then approximate F by a neural network ϕ, and the details can be found in the Appendix of [150].

Remark 4.4.2. *The equivalence of norms gives an upper bound on the Lipschitz constant for all norms.*

4.4.2 Existence of Neural Lyapunov Functions

As a theoretical guarantee, we show that it is possible to train a neural network as a Lyapunov function, provided that a Lyapunov function exists. According to converse Lyapunov theorems [72, 128], Lyapunov functions do exist when the origin is UAS. More specifically, we show that the learned neural network satisfies the Lyapunov conditions outside some neighborhood of the origin that can be chosen to be arbitrarily small in measure. The idea will be to perform an under-approximation of the domain $\mathcal{D}$ in a controlled way. Let $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n), \mu)$ denote the standard measure space where $\mathcal{B}(\mathbb{R}^n)$ is the Borel σ-algebra and μ is the Lebesgue measure. The following lemma from [118] states that it is possible to arbitrarily under-approximate open sets by compact sets in measure.

Lemma 4.4.3. *For every open set* $O \in \mathcal{B}(\mathbb{R}^n)$ *such that* $\mu(O) < \infty$ *and every* $\epsilon > 0$*, there exists a compact set* K *such that* $\mu(O \setminus K) < \epsilon$.

By the pointwise approximation of the universal approximation theorem, it is not possible to satisfy the Lyapunov conditions on $\mathcal{D}$ as this set contains the origin. However, if there is a Lyapunov function for (4.2) that a neural network could learn and if practical

stability (i.e., set stability w.r.t. a sufficiently small neighborhood of the origin) is sufficient, Theorem 4.4.4 states that there exists a neural network satisfying the Lyapunov conditions on a compact set K except on a closed neighborhood B of the origin that is UAS. Moreover, this approximation can be controlled in measure. The details of the proof can be found in the Appendix of [150].

Theorem 4.4.4. *Suppose that the origin is UAS for system (4.2) and $\mathcal{I}$ is a forward invariant set contained in the ROA of the origin. Fix any $\gamma_1, \gamma_2 > 0$. There exists a forward invariant and compact set $K \subset \mathcal{I}$ satisfying the under approximation $\mu(\mathcal{I} \setminus K) < \gamma_1$. On K there exists a neural network V_ϕ that satisfies the Lyapunov conditions on $K \setminus \mathcal{A}$, where $\mathcal{A}$ is a closed neighborhood of the origin. The neural Lyapunov function V_ϕ can certify that a closed invariant set $\mathcal{B}$ containing $\mathcal{A}$ and satisfying $\mu(\mathcal{B} \setminus \mathcal{A}) < \gamma_2$ is UAS. Furthermore, the set K is contained in the ROA of $\mathcal{B}$.*

4.4.3 Asymptotic Stability Guarantees of Unknown Nonlinear Systems

With the theoretical guarantee of learning a neural Lyapunov function established above, we show that the neural network trained with the learned dynamics is robust, that is this neural network also satisfies the Lyapunov conditions with respect to the actual dynamics f. Since the SMT solver verifies the Lyapunov conditions outside of some ϵ-ball in (4.11), which is not necessarily forward invariant, the following technical assumption helps bridge this gap. A linear system $\dot{x} = \mathbf{A}x + Bu$ is said to be *stabilizable* if there exists a matrix $\mathcal{K}$ such that $\mathbf{A} + B\mathcal{K}$ is Hurwitz, i.e., all eigenvalues of $\mathbf{A} + B\mathcal{K}$ have negative real part. If $(\mathbf{A}, B)$ is stabilizable, then the gain matrix $\mathcal{K}$ can be obtained by an LQR controller.

Assumption 4.4.5 (ROA of LQR Controller)**.** *Suppose that the linearized model $\dot{x} = \mathbf{A}x + Bu$ from Assumption 4.2.2 is stabilizable. Consequently, an LQR controller and a quadratic Lyapunov function can be found such that the origin is UAS for the closed-loop system (4.2). Furthermore, we assume that the set B_ε which is not verified by an SMT solver lies in the interior of an ROA of the closed-loop system, provided by the quadratic Lyapunov function.*

Since this ε-ball is small, we further assume that the level sets of the Lyapunov function are contained in the ROA of the LQR controller.

Assumption 4.4.6 (Controlled Level Sets)**.** *Denote $B_\varepsilon := \{x : \|x\| \leq \varepsilon\}$. Let V be a continuously differentiable function satisfying the Lyapunov conditions on $\Omega \setminus B_\varepsilon$. Suppose*

that there exist constants $0 < c_1 < c_2$ *such that the following chain of inclusions holds*

$$\{x \in \Omega : V(x) \leq c_1\} \subset B_\varepsilon \subset \{x \in \Omega : V(x) \leq c_2\}, \tag{4.15}$$

and $\{x \in \Omega : V(x) \leq c_2\}$ *lies in the interior of the ROA of the closed loop system provided by the quadratic Lyapunov function.*

Theorem 4.4.7. *(Stability Guarantees for the Unknown System) Let* ϕ *be the approximated dynamics of right-hand side of the closed-loop system (4.2) trained by the first neural network. There exists a neural Lyapunov function* V *which is learned using* ϕ *and verified by an SMT solver that satisfies the Lyapunov conditions with respect to the actual dynamics* f*. Furthermore, if the system satisfies Assumption 4.4.5 and* V *satisfies Assumption 4.4.6, then the origin is UAS for the closed-loop system* (4.2)*.*

The details of this proof can be found in the Appendix of [150]. This theorem proves that if the dynamics are approximated to sufficient precision then the neural Lyapunov function satisfies the Lyapunov conditions on $\Omega \setminus B_\varepsilon$ for the actual dynamics. Furthermore, if the level sets of the neural Lyapunov function are sufficiently well behaved and the set B_ε excluded from SMT verification is small then this learning framework certifies that the origin is UAS for the actual system (4.2).

4.5 Numerical Experiments

In this section, we demonstrate the effectiveness of the proposed algorithm on learning the unknown dynamics and finding provably stable controllers as well as neural Lyapunov functions for various classic nonlinear systems. In all the following problems, we use a two-layer neural network with one hidden layer for both learning the dynamics and learning the neural Lyapunov function. For learning the dynamics, the number of neurons in the hidden layer varies from 100 to 200 without an output layer activation function as stated before, and we call this neural network FNN for convenience. However, for learning the neural Lyapunov function, there are six neurons in the hidden layer for all the experiments, and we name this neural network VNN in short. Regarding other parameters, we use the Adam optimizer for both FNN and VNN, and we use dReal as the SMT solver, setting δ as 0.01 for all experiments. All the training of FNN is performed on Google Colab with a 16GB GPU, and VNN training is done on a 3 GHz 6-Core Intel Core i5.

4.5.1 Reversed Van der Pol Oscillator

As a starting point, we test the proposed algorithm on the nonlinear system without input u first to show its effectiveness in learning unknown dynamical systems and finding a valid neural Lyapunov function. The Van der Pol oscillator is a well-known nonlinear system that exhibits stable oscillations in the form of a limit cycle, whereas its reversed version yields a locally asymptotically stable system, with the former limit cycle delineating the actual ROA of the origin. The state equations of the reversed Van der Pol oscillator are:

$$\begin{aligned} \dot{x}_1 &= -x_2 \\ \dot{x}_2 &= x_1 + \left(x_1^2 - 1\right) x_2. \end{aligned} \tag{4.16}$$

The area within the limit cycle is the non-convex ROA, as shown in Figure 4.3 [59].

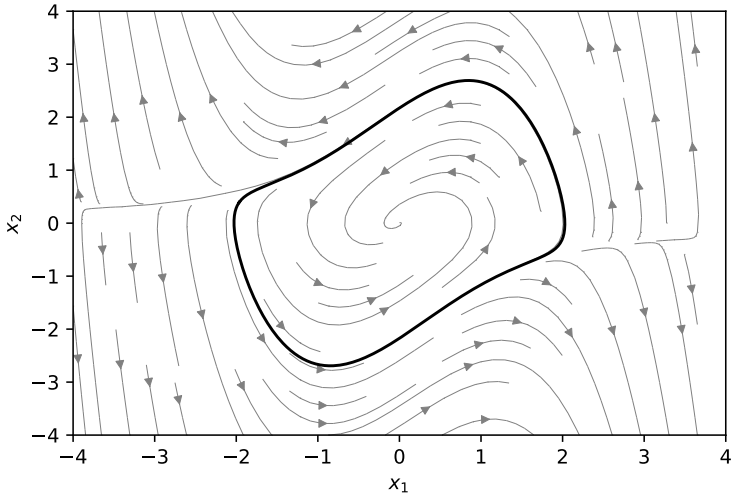


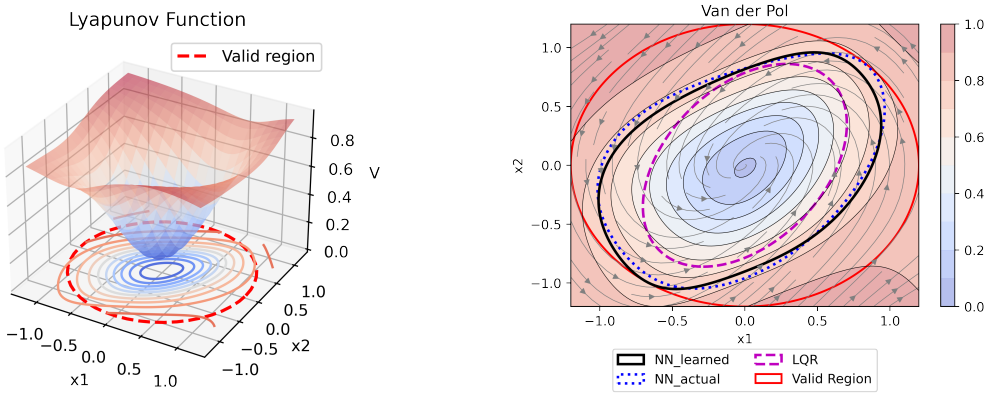
Figure 4.3: Phase space plot and the limit cycle (bold black line) of the reversed Van der Pol oscillator without controller, where the area within the bold black curve forms the actual ROA.

According to the algorithm described in Section 4.3, we can write $x = [x_1, \quad x_2]^\intercal$ and learn the two nonlinear dynamical equations with 100 hidden neurons. To ensure an accurate model, we use 9 million data points sampled on $(-1.5, 1.5) \times (-1.5, 1.5)$, and the learning rate of the training process varies from 0.1 to 1e-5 to acquire a small enough training error α. With the learned dynamics, we aim to find a valid Lyapunov function over the domain $\|x\|_2 < 1.2$, and the obtained neural Lyapunov function is shown in Fig. 4.4a.

The corresponding ROA estimate can be found in Fig. 4.4b. In comparison, the ROAs found by the neural Lyapunov function and the classic LQR techniques in [59] using actual dynamics are also plotted, as the blue and magenta ellipses respectively. The phase portrait of the system is given as the grey curves with small arrows. It is obvious that the neural Lyapunov function after tuning obtains a larger ROA. We also have a comparable verified ROA for the system with the one obtained with actual dynamics based on the same neural Lyapunov approach, both larger than the LQR case. The values of parameters can be found in Table 4.1.

Table 4.1: Parameters in the reversed Van der Pol oscillator case

K_f	K_ϕ	δ	α	$\|\frac{\partial V}{\partial x}\|$	β	ε
3.4599	5.197	5e-4	8.5e-3	1.249	0.02	0.2



(a) Neural Lyapunov function

(b) ROA comparison with LQR

Figure 4.4: Neural Lyapunov function and the corresponding estimated ROA for the reversed Van der Pol oscillator.

4.5.2 Unicycle Path Following

The path following task is a typical stabilization problem for a nonlinear system. Here, we consider path tracking control using a kinematic unicycle [21] with error dynamics. In this case, we have two state variables, the angle error θ_e and the distance error d_e, and the

dynamics of this system can be written as:

$$\begin{aligned} \dot{s} &= \frac{v\cos(\theta_e)}{1-d_e\kappa(s)}, \\ \dot{d}_e &= v\sin(\theta_e), \\ \dot{\theta}_e &= \omega - \frac{v\kappa(s)\cos(\theta_e)}{1-d_e\kappa(s)}. \end{aligned} \tag{4.17}$$

Here we assume the target path is a unit circle $\kappa(s) = 1$ and take ω as the input u with $x = [d_e \quad \theta_e]^\mathsf{T}$, consequently the dynamical system is of the format $\dot{x} = f(x, u)$. Similarly, after obtaining the learned dynamics $\phi(x, u)$ with 200 hidden neurons, a neural Lyapunov function can be learned on the valid region $\|x\|_2 \leq 0.8$, and the neural controller is set as $u = 5\tanh(kx + b)$. The parameters for this valid neural Lyapunov function in this case is listed in Table 4.2. The Lipschitz constant K_f is computed by using the bound $\|J_f\|_2 \leq \sqrt{m}\,\|J_f\|_\infty$, where J_f is the Jacobian matrix of f and m is the number of rows of J_f. Note that α can be the maximum of the 2-norm loss over a testing dataset, as what we need here is the discrepancies between the actual value and the approximated value of some known samples. In this regard, we can train FNN with fewer data samples, which is more computationally efficient. On top of that, a much larger dataset uniformly sampled over state and input spaces is used to calculate α, which contributes to a smaller δ.

Table 4.2: Parameters in unicycle path following case

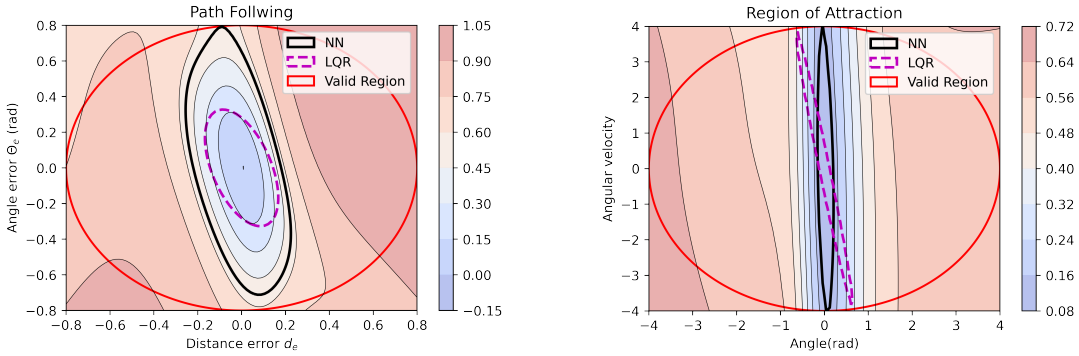
K_f	K_ϕ	δ	α	$\|\frac{\partial V}{\partial x}\|$	β	ε
< 45	108	1e-4	7e-3	4.43	0.1	0.1

The ROA comparison with the LQR method can be found in Fig. 4.5a. Apparently, the neural network method yields a larger estimated ROA, compared to the classic LQR approach in which the level set is determined by considering some relaxation of the largest reasonable range of linearization for practical systems under a small angles assumption ($< \frac{\pi}{9}$), given the fact that the actual dynamics is unknown.

4.5.3 Inverted Pendulum

The inverted pendulum is another well-known nonlinear control problem. The system dynamics of that can be described as

$$\ddot{\theta} = \frac{mg\ell\sin(\theta) + u - 0.1\dot{\theta}}{m\ell^2}. \tag{4.18}$$



(a) ROA comparison for path following

(b) ROA comparison for inverted pendulum

Figure 4.5: Comparison of obtained ROAs for path following and inverted pendulum.

This system has two state variables $\theta, \dot{\theta}$ and one control input u. θ and $\dot{\theta}$ represent the angular position from the inverted position and angular velocity. The only nonlinear function we need to learn here for FNN is (4.18). Therefore, the input $[x, \quad u]^\mathsf{T}$ is 3-dimensional and the output is 1-dimensional. The valid domain is $\|x\|_2 \leq 4$. Using constants $g = 9.81, m = 0.15$ and $\ell = 0.5$, by the same process as the previous experiment, the similar ROA comparison is shown in Fig. 4.5b, where the LQR approach uses the same function as given in [22]. The corresponding parameters can be found in Table 4.3.

Table 4.3: Parameters in inverted pendulum case

K_f	K_ϕ	δ	α	$\|\frac{\partial V}{\partial x}\|$	β	ε
$<$33.214	633.806	5e-5	5e-3	0.51	0.02	0.4

4.6 Summary

This chapter explores the ability of neural networks to approximate an unknown nonlinear system with a sufficient precision and find a neural Lyapunov function as well as a feedback controller to stabilize the unknown dynamics. Provable guarantees are also provided with the help of bounds on the approximation error and partial derivatives of the Lyapunov function.

Moreover, we experiment on three typical nonlinear system examples: the reversed Van der Pol oscillator, the inverted pendulum, and the unicycle path following. To align with the theoretical results, we carefully establish bounds for the Lipschitz constants, keep track of the training losses, and adjust the Lyapunov conditions to be verified by the SMT solver accordingly so that the verified ROA is valid for the unknown system. The numerical experiments show that the ROA learned this way are comparable with the ones computed with actual dynamics, and improve those obtained from the LQR controllers with quadratic Lyapunov functions.

Chapter 5

Resolvent-Type Data-Driven Learning of Generators for Unknown Dynamical Systems

This chapter extends the work reported in two papers: the conference version in [93] and a journal version currently under review at the IEEE Transactions on Automatic Control, for which a preprint is available in [94].

5.1 Introduction

The development of Koopman operator theory [64] for dynamical systems provides a promising alternative learning approach for nonlinear system identification and stability analysis [87], using data from snapshots of the flow map (trajectories). By leveraging the spectral properties [97] and mode decomposition capabilities [124, 137] of Koopman operators, nonlinear dynamics can be converted into a discrete family of observable functions (functions of the states), which evolve linearly in the infinite-dimensional function space driven by the Koopman operator. This nonlinear-to-linear conversion, seemingly friendly at first, is later found to require the study of continuous functional calculus. It suffers from restrictive assumptions on the function properties [82] to ensure correct usage. Consequently, many applications are limited to discrete-time nonlinear systems.

In identifying the continuous-time dynamics of an unknown system, the equivalence between the vector field and the infinitesimal generator of the system semigroup has

been shown in [64, 97], indicating an equivalence in terms of conventional system (model) identification and generator (or semigroup) characterization. In this chapter, we summarize two widely used Koopman-based characterization methods in detail before introducing our proposed method.

Finite-difference method (FDM) is a numerical technique used to solve differential equations by approximating derivatives with finite differences. In the context of Koopman-based learning, the generator on any observable function is approximated by the rate of its evolution along the trajectory, using a forward difference method with a small discretized time scale. Similar to direct methods, this approximation scheme also relies on the notion of evolution speed or time derivatives. As anticipated, its precision heavily depends on the size of the temporal discretization [13, 61, 100, 134, 102].

Recent studies [85, 61, 34, 11] have facilitated another indirect learning of the generator using the logarithm of the learned Koopman operator. This approach can potentially circumvent the need for high observation rates and longer observation periods, as it does not require the estimation of time derivatives. Heuristically, researchers tend to represent the Koopman operator $\mathcal{K}_t$ by an exponential form of its generator $\mathcal{L}$ as $\mathcal{K}_t = e^{t\mathcal{L}}$, leading to the converse representation $\mathcal{L} = \frac{1}{t}\log(\mathcal{K}_t)$ for any $t > 0$. However, problems arise given the following concerns: 1) representing Koopman operators in exponential form requires the boundedness of the generator $\mathcal{L}$; 2) the operator logarithm is a single-valued mapping only within a specific sector of the spectrum; 3) for general systems that fall short of the aforementioned restrictions, it is unclear how the data-driven approximation of the logarithm of Koopman operators converges to the true generator.

As a complement to the work in [85, 34], recent studies [145, 144] have rigorously investigated the sufficient and necessary conditions under which the Koopman-logarithm-based generator learning method can guarantee learning accuracy. To provide the sampling rate, the theorem relies on the concept of a "generator-bounded space", which remains invariant under the Koopman operator, and where the generator is bounded when restricted to it. However, the mentioned concept is less likely to be verifiable for unknown systems.

To address the aforementioned issues, we aim to propose an operator logarithm-free generator learning scheme, named the resolvent-type method (RTM), which is robust to the choice of the dictionary of observable functions and does not require knowledge of spectral properties of the Koopman operators. A brief discussion on the resolvent of the Koopman generator can be found in [126]. However, to the best of the authors' knowledge, the current work is the first to utilize a resolvent-based method to identify the generator and, consequently, the vector fields. In summary, the main contributions of the method proposed in this chapter are:

1. We demonstrate the converse relationship between the Koopman operators and the generator in more general cases where the generator may be unbounded. Specifically, we draw upon the rich literature to propose a finite-dimensional approximation based on Yosida approximation for these cases.

2. We provide the analytical convergence rate of this approximation and demonstrate the theoretical feasibility of a data-driven adaptation.

3. We adapt the Koopman operator learning structure for generator learning based on 1, and demonstrate that a modification is needed to accommodate a low observation rate constraint.

4. We provide numerical experiments and demonstrate the effectiveness of the proposed approach by comparing it with the FDM and logarithm-based methods.

5.2 Special Notations for This Chapter

For any complete normed function spaces $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$ and $(\mathcal{W}, \|\cdot\|_{\mathcal{W}})$ of the real-valued observable functions, and for any bounded linear operator $\mathcal{B} : \mathcal{V} \to \mathcal{W}$, we define the operator (uniform) norm as $\|\mathcal{B}\| := \sup_{\|h\|_{\mathcal{V}}=1} \|\mathcal{B}h\|_{\mathcal{W}}$. Let $\mathcal{C}(\Omega)$ be the set of continuous functions with domain Ω, endowed with the uniform norm $\|\cdot\| := \sup_{x\in\Omega} |\cdot(x)|$. We denote the set of i-times continuously differentiable functions by $\mathcal{C}^i(\Omega)$. For convergence analysis, we also write $a \lesssim b$ if there exists a $C > 0$ (independent of a and b) such that $a \leq Cb$, particularly when C is lengthy or its exact value is not necessarily required. From Section 5.5 to 5.7, we will frequently use notations that represent approximations of certain quantities. We use $Q^{\text{sup}}_{\text{sub}}$ to denote an approximation of quantity Q that depends specifically on parameters appearing in the subscripts and superscripts, and use $\widetilde{Q}$ to denote an approximation of quantity Q without emphasizing its dependent parameters.

5.3 Preliminaries

This chapter builds upon the theoretical framework of linear operator theory and strongly continuous semigroups that underlie the Koopman operator approach to dynamical systems. To maintain readability, the essential background results on bounded, closed, and inverse operators are collected in Appendix A, while the basic properties of $\mathcal{C}_0$-semigroups and their infinitesimal generators are summarized in Appendix B.

5.3.1 Dynamical Systems

Given a pre-compact state space $\Omega \subseteq \mathbb{R}^d$, we consider a continuous-time nonlinear dynamical system of the form

$$\dot{x} = f(x), \tag{5.1}$$

where the vector field $f : \Omega \to \Omega$ is assumed to be locally Lipschitz continuous.

Given an initial condition x_0, on the maximal interval of existence $I \subseteq [0, \infty)$, the unique forward flow map (solution map) $\phi : I \times \Omega \to \Omega$ should satisfy

1. $\partial_t(\phi(t, x_0)) = f(\phi(t, x_0))$,
2. $\phi(0, x_0) = x_0$, and
3. $\phi(s, \phi(t, x_0)) = \phi(t + s, x_0)$ for all $t, s \in I$ such that $t + s \in I$.

Throughout this chapter, we will assume that the maximal interval of existence of the flow map to the initial value problem (5.1) is $I = [0, \infty)$.

Remark 5.3.1. *The above assumption is equivalent to assuming that the system exhibits forward invariance w.r.t. the set Ω. However, this is usually not the case for general nonlinear systems. In this paper, if the system dynamics violate the above assumption, we can adopt the approach outlined in [91, Section III.B] to recast the dynamics within the set $\overline{\Omega}$. In other words, we constrain the vector field f such that $f(x) = 0$ for any $x \in \partial\Omega$, while f remains unchanged within the open domain Ω. This modification ensures that the system data is always collectible within $\overline{\Omega}$.* ◇

5.3.2 Koopman Operators and the Infinitesimal Generator

Let us now consider a complete normed function space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ of the real-valued observable functions[1] $h : \Omega \to \mathbb{R}$.

Definition 5.3.2. *A one-parameter family $\{\mathcal{S}_t\}_{t\geq 0}$ of bounded linear operators from $\mathcal{F}$ into $\mathcal{F}$ is a semigroup if*

[1]We clarify that in the context of operator learning, the term "observable functions," or simply "observables," commonly refers to "test functions" for operators, rather than to the concept of "observability" in control systems.

1. $\mathcal{S}_0 = \mathrm{Id}$;

2. $\mathcal{S}_t \circ \mathcal{S}_s = \mathcal{S}_{t+s}$ *for every* $t, s \geq 0$.

In addition, a semigroup $\{\mathcal{S}_t\}_{t\geq 0}$ *is a* $\mathcal{C}_0$*-semigroup if* $\lim_{t\downarrow 0} \mathcal{S}_t h = h$ *for all* $h \in \mathcal{F}$*, and moreover a* $\mathcal{C}_0$*-semigroup of contractions if* $\|\mathcal{S}_t\| \leq 1$ *for* $t \geq 0$.

The (infinitesimal) generator $\mathcal{A}$ *of* $\{\mathcal{S}_t\}_{t\geq 0}$ *is defined by*

$$\mathcal{A}h(x) := \lim_{t\downarrow 0} \frac{\mathcal{S}_t h(x) - h(x)}{t}, \tag{5.2}$$

where the observable functions are within the domain of $\mathcal{A}$*, defined as*

$$\mathrm{dom}(\mathcal{A}) = \left\{ h \in \mathcal{F} : \lim_{t\downarrow 0} \frac{\mathcal{S}_t h - h}{t} \text{ exists} \right\}.$$

◇

It is a well-known result that $\mathrm{dom}(\mathcal{A})$ is dense in $\mathcal{F}$, i.e. $\overline{\mathrm{dom}(\mathcal{A})} = \mathcal{F}$.

The evolution of observables of system (5.1) restricted to $\mathcal{F}$ is governed by the family of Koopman operators, as defined below. Koopman operators also form a linear $\mathcal{C}_0$-semigroup, allowing us to study nonlinear dynamics through the infinite-dimensional lifted space of observable functions, which exhibit linear dynamics.

Definition 5.3.3. *The Koopman operator family* $\{\mathcal{K}_t\}_{t\geq 0}$ *of system* (5.1) *is a collection of maps* $\mathcal{K}_t : \mathcal{F} \to \mathcal{F}$ *defined by*

$$\mathcal{K}_t h = h \circ \phi(t, \cdot), \quad h \in \mathcal{F} \tag{5.3}$$

for each $t \geq 0$*, where* $\circ$ *is the composition operator. The (infinitesimal) generator* $\mathcal{L}$ *of* $\{\mathcal{K}_t\}_{t\geq 0}$ *is defined accordingly as in* (5.2). ◇

Due to the (local) Lipschitz continuity of f in (5.1), and considering that observable functions are usually continuous, we will focus on $\mathcal{K}_t : \mathcal{C}(\Omega) \to \mathcal{C}(\Omega)$ for the rest of the paper. For (5.1), there exist constants $\omega \geq 0$ and $C \geq 1$ such that $\|\mathcal{K}_t\| \leq Ce^{\omega t}$ for all $t \geq 0$ [109, Theorem 2.2, Chapter 1].

It can also be verified that $\mathcal{C}^1(\Omega) \subseteq \mathrm{dom}(\mathcal{L}) \subseteq \mathcal{C}(\Omega)$. In general, $\mathrm{dom}(\mathcal{L})$ depends on the regularity and degeneracy of f, as well as the geometry of the flow, which determines whether $f \cdot \nabla h$ ($h \in \mathrm{dom}(\mathcal{L})$) exists in a classical, weak, or viscosity sense. For instance,

$h \in \mathrm{dom}(\mathcal{L})$ need not be differentiable on $E := \{x \in \mathbb{R}^d : f(x) = 0\}$ but it may be $\mathcal{C}^1$ on $\Omega \setminus E$ with $f \cdot \nabla h$ extending continuously to E. In the extreme case $f(x) \equiv 0$, we have $\mathrm{dom}(\mathcal{L}) = \mathcal{C}(\Omega)$.

For $\mathcal{F} = \mathcal{C}(\Omega)$ endowed with the uniform norm $\|\cdot\| := \sup_{x\in\Omega} |\cdot(x)|$, the graph norm of $\mathcal{L}$ is naturally defined as

$$\|\cdot\|_{\mathcal{L}} := \|\cdot\| + \|\mathcal{L}(\cdot)\|. \tag{5.4}$$

For an unknown f, it is impossible to recover $\mathcal{L}$ on the entire domain $\mathrm{dom}(\mathcal{L})$. We therefore restrict our attention to a *core* of $\mathrm{dom}(\mathcal{L})$, based on the following concept.

Definition 5.3.4. *Consider $\mathcal{A} : \mathrm{dom}(\mathcal{A}) \subseteq \mathcal{F} \to \mathcal{F}$ and $\mathcal{B} : \mathcal{D} \subseteq \mathcal{F} \to \mathcal{F}$. We say that $\mathcal{B}$ admits a closure w.r.t. $\mathcal{A}$, denoted as $\overline{\mathcal{B}} = \mathcal{A}$, if $\mathcal{A}|_{\mathcal{D}} = \mathcal{B}$ and $\overline{\mathcal{D}} = \mathrm{dom}(\mathcal{A})$ w.r.t. the graph norm of $\mathcal{A}$. In this case, we also say that $\mathcal{D}$ is a core of $\mathcal{A}$.* ◇

Note that the flow $\phi(t, x_0)$ is absolutely continuous. For learning the generator, we usually work with a (hypothetical) smoother surrogate of the flow. In particular, there exists a $\mathcal{C}^1(\Omega)$ surrogate that approximates the true flow uniformly on any finite time window. For discrete snapshot data, the approximation can be chosen to match the observations to arbitrary accuracy, and, if desired, exactly at finitely many sampled states via an additional small $\mathcal{C}^1$ perturbation. By abuse of notation, we continue to write $\phi(t, x_0)$ for this $\mathcal{C}^1$ surrogate[2]. Additionally, in this setting, for any $h \in \mathcal{C}^1(\Omega)$, $\nabla(\mathcal{K}_t h)(x)$ exists and $\mathcal{K}_t : \mathcal{C}^1(\Omega) \to \mathcal{C}^1(\Omega)$ for all $t > 0$.

Proposition 5.3.5. *Working with a sufficiently smooth surrogate $\phi(t, x)$, for any $h \in \mathrm{dom}(\mathcal{L}) \subseteq \mathcal{C}(\Omega)$, there exists a sequence of $\{h_n\} \subseteq \mathcal{C}^1(\Omega)$ such that $\|h_n - h\|_{\mathcal{L}} \to 0$.*

Proof: For any $h \in \mathrm{dom}(\mathcal{L})$, let $\{g_n\} \subseteq \mathcal{C}^1(\Omega)$ be a sequence such that $\|g_n - h\| \to 0$. Let $h_{n,t}(x) = \frac{1}{t}\int_0^t \mathcal{K}_t h_n(x) ds$ for any $t > 0$. Then, $h_{n,t} \in \mathcal{C}^1(\Omega)$ for any n and any fixed $t > 0$. Consider $\{t_n\}$ such that $t_n \to 0$ and $\|g_n - h\|/t_n \to 0$ as $n \to \infty$, and set $h_n := h_{n,t_n}$. Then,

$$\begin{aligned} &\|h_n - h\| \\ \leq &\left\| \frac{1}{t_n}\int_0^{t_n} \mathcal{K}_s(g_n - h)\mathrm{d}s \right\| + \left\| \frac{1}{t_n}\int_0^{t_n} \mathcal{K}_s h \,\mathrm{d}s - h \right\| \\ \leq &\frac{C(e^{\omega t_n} - 1)}{\omega t_n}\|g_n - h\| + \left\| \frac{1}{t_n}\int_0^{t_n} \mathcal{K}_s h \,\mathrm{d}s - h \right\|, \end{aligned} \tag{5.5}$$

[2]We do not need to construct this surrogate explicitly. It is used solely to simplify exposition, and since the learned system is approximate in any case, this convention does not affect our conclusions.

and $\|h_n - h\| \to 0$ as $n \to \infty$.

Furthermore, by [109, Theorem 2.4, Chapter 1], we have $\mathcal{L}\int_0^t \mathcal{K}_s g_n ds = \mathcal{K}_s g_n - g_n$. Therefore,

$$\begin{aligned}
&\|\mathcal{L}h_n - \mathcal{L}h\| \\
=&\left\|\frac{\mathcal{K}_{t_n} g_n - g_n}{t_n} - \mathcal{L}h\right\| \\
\leq&\left\|\frac{(\mathcal{K}_{t_n} - \mathrm{Id})(g_n - h)}{t_n}\right\| + \left\|\frac{\mathcal{K}_{t_n} h - h}{t_n} - \mathcal{L}h\right\| \\
\leq&\frac{Ce^{\omega t_n} + 1}{t_n}\|g_n - h\| + \left\|\frac{\mathcal{K}_{t_n} h - h}{t_n} - \mathcal{L}h\right\|,
\end{aligned} \tag{5.6}$$

and $\|\mathcal{L}h_n - \mathcal{L}h\| \to 0$ as $n \to \infty$. The conclusion follows immediately. ■

Then, by Proposition 5.3.5, we have $\overline{\mathcal{L}|_{\mathcal{C}^1(\Omega)}} = \mathcal{L}$, and $\mathcal{C}^1(\Omega)$ is the core of $\mathcal{L}$. For the remainder of the paper, we work with $\mathcal{C}^1(\Omega)$-class dictionary of observables, where approximation on this core is effectively equivalent to approximation on the full domain for the purpose of learning $\mathcal{L}$ with both approximation guarantees and numerical tractability.

Moreover, for all continuously differentiable observable functions, i.e., $h \in \mathcal{C}^1(\Omega)$, the generator of the Koopman operators is given by

$$\mathcal{L}h(x) = \nabla h(x) \cdot f(x). \tag{5.7}$$

The space $\mathcal{F}$ can be extended to $L^\infty(\Omega)$, which allows us to study the L^2-adjoint operator of $\mathcal{L}$ with domain $L^1(\Omega)$, leading to more interesting applications such as the Liouville equation and ergodic measure [108]. We will pursue generator learning on this extended domain in future work.

Definition 5.3.6. *For operator learning, we define*

$$Z_N(x) := [\mathfrak{z}_0(x), \mathfrak{z}_1(x), \cdots, \mathfrak{z}_{N-1}(x)]^\mathsf{T},\ N \in \mathbb{N}, \tag{5.8}$$

as the vector of dictionary functions, where $\{\mathfrak{z}_i\} \subseteq \mathcal{C}^1(\Omega)$. *We denote by* $\mathcal{Z}_N := \mathrm{span}\{\mathfrak{z}_i : i = 1, 2, \cdots, N\}$ *the span of dictionary functions.*

For any linear operator $\mathcal{B} : \mathcal{C}(\Omega) \to \mathcal{C}(\Omega)$, *we adopt the notational conventions* $\mathcal{B}(\mathcal{D}) := \{\mathcal{B}h : h \in \mathcal{D}\}$ *for any* $\mathcal{D} \subseteq \mathcal{C}(\Omega)$ *and* $\mathcal{B}Z_N(x) = [\mathcal{B}\mathfrak{z}_0(x), \mathcal{B}\mathfrak{z}_1(x), \cdots, \mathcal{B}\mathfrak{z}_{N-1}(x)]^\mathsf{T}$. ◇

We make the standing assumption on the dictionary that $\mathcal{Z}_1 \subseteq \mathcal{Z}_2 \subseteq \cdots$ and $\overline{\cup_{n\geq 1}\mathcal{Z}_N} = \mathcal{C}(\Omega)$ to ensure density. Denoting $E_N(h) := \inf_{v\in\mathcal{Z}_N} \|h - v\|$ as the best-approximation error, then $E_N(h) \xrightarrow{N\to\infty} 0$ for any $h \in \mathcal{C}(\Omega)$.

Remark 5.3.7. *In view of Koopman-based indirect approximation of the generator, for each $h \in \mathcal{C}^1(\Omega)$, we have $\mathcal{K}_\tau h(x) - h(x) = \int_0^\tau \mathcal{L}h(\phi(s,x))ds \approx \int_0^\tau \mathcal{L}h(\phi(0,x))ds = \mathcal{L}h(x)\tau$, where the approximation is achieved through a small terminal time τ. Then, the derivative (of any h) along the trajectory is approximated by $\mathcal{L}h(x) \approx \frac{\mathcal{K}_\tau h(x)-h(x)}{\tau}$. Note that the FDM $\mathcal{L} \approx \frac{\mathcal{K}_\tau - \mathrm{Id}}{\tau}$ simply follows (5.2) for sufficiently small $\tau > 0$. Through this approximation scheme of the time derivative, it can be anticipated that the precision heavily depends on the size of τ [13, 100, 134]. We will revisit the FDM in the numerical examples to compare it with the method proposed in this paper.* ◇

5.3.3 Representation of Semigroups

In this subsection, we introduce basic operator topologies and explore how a semigroup $\{\mathcal{S}_t\}_{t\geq 0}$ can be represented through its generator $\mathcal{A}$.

Definition 5.3.8 (Operator Topologies)**.** *Consider Banach spaces $(\mathcal{V}, \|\cdot\|_\mathcal{V})$ and $(\mathcal{W}, \|\cdot\|_\mathcal{W})$ Let $\mathcal{B} : \mathcal{V} \to \mathcal{W}$ and $\mathcal{B}_n : \mathcal{V} \to \mathcal{W}$, for each $n \in \mathbb{N}$, be linear operators.*

1. *The sequence $\{\mathcal{B}_n\}_{n\in\mathbb{N}}$ is said to converge to $\mathcal{B}$* uniformly*, denoted by $\mathcal{B}_n \to \mathcal{B}$, if $\lim_{n\to\infty} \|\mathcal{B}_n - \mathcal{B}\| = 0$. We also write $\mathcal{B} = \lim_{n\to\infty} \mathcal{B}_n$.*
2. *The sequence $\{\mathcal{B}_n\}_{n\in\mathbb{N}}$ is said to converge to $\mathcal{B}$* strongly*, denoted by $\mathcal{B}_n \rightharpoonup \mathcal{B}$, if $\lim_{n\to\infty} \|\mathcal{B}_n h - \mathcal{B}h\|_\mathcal{W} = 0$ for each $h \in \mathcal{V}$. We also write $\mathcal{B} = \text{s-}\lim_{n\to\infty} \mathcal{B}_n$.* ◇

Remark 5.3.9. *In analogy to the pointwise convergence of functions, the strong topology is the coarsest topology such that $\mathcal{B} \mapsto \mathcal{B}h$ is continuous in $\mathcal{B}$ for each fixed $h \in \mathcal{F}$.* ◇

If $\mathcal{A}$ is a bounded linear operator that generates $\{\mathcal{S}_t\}$, then $\mathcal{S}_t = e^{t\mathcal{A}}$ for each t in the uniform operator topology. Otherwise, [109, Theorem 5.5, Chapter 1] (also rephrased as Theorem 5.3.11 in this paper) still provides an interpretation for the sense in which $\mathcal{S}_t$ "equals" $e^{t\mathcal{A}}$.

We revisit some facts to show the above concepts of equivalence, particularly in the context where $\mathcal{A}$ is unbounded.

Definition 5.3.10 (Resolvents)**.** *Let $\mathcal{A} : \mathrm{dom}(\mathcal{A}) \subseteq \mathcal{F} \to \mathcal{F}$ be a linear, not necessarily bounded, operator. Then the resolvent set is defined as*

$$\rho(\mathcal{A}) := \{\lambda \in \mathbb{C} : \lambda \,\mathrm{Id} - \mathcal{A} \text{ is invertible}\}\,. \tag{5.9}$$

For any $\lambda \in \rho(\mathcal{A})$, the resolvent operator is defined as

$$\mathcal{R}(\lambda; \mathcal{A}) := (\lambda \,\mathrm{Id} - \mathcal{A})^{-1}, \tag{5.10}$$

and it is a bounded linear operator by [109, Theorem 4.3, Chapter 1]. ◇

For the basic notions of invertibility and the resolvent of linear operators, we refer the reader to Appendix A.

Following the standard construction (see Appendix B.3), we further define the *Yosida approximation* of $\mathcal{A}$ as

$$\mathcal{A}_\lambda := \lambda \mathcal{A} \mathcal{R}(\lambda; \mathcal{A}) = \lambda^2 \mathcal{R}(\lambda; \mathcal{A}) - \lambda \,\mathrm{Id}\,. \tag{5.11}$$

Note that $\{\mathcal{A}_\lambda\}_{\lambda \in \rho(\mathcal{A})}$ is a family of bounded linear operators, and $e^{t\mathcal{A}_\lambda}$ is well-defined for each $\lambda \in \rho(\mathcal{A})$.

The characterization of $\mathcal{A}$ as the infinitesimal generator of a $\mathcal{C}_0$-semigroup is typically formulated in terms of conditions on the resolvent of $\mathcal{A}$, where $\mathcal{A}$ must be closed and $\rho(\mathcal{A}) \neq \emptyset$. We also have the following theorem for semigroup approximation.

Theorem 5.3.11. *[109, Theorem 5.5, Chapter 1] Suppose that $\{\mathcal{S}_t\}_{t \geq 0}$ is a $\mathcal{C}_0$-semigroup on $\mathcal{F}$ and $\mathcal{A}$ is the generator. Then, $\mathcal{S}_t = \text{s-}\lim_{\lambda \to \infty} e^{t\mathcal{A}_\lambda}$ for all $t \geq 0$.*

5.4 The Characterization of the Infinitesimal Generators

For system (5.1), we are able to represent $\mathcal{K}_t$ as $e^{t\mathcal{L}}$ for bounded $\mathcal{L}$, or, by Theorem 5.3.11, as the strong limit of $e^{t\mathcal{L}_\lambda}$, where $\mathcal{L}_\lambda$ is the Yosida approximation of the unbounded $\mathcal{L}$.

For the converse representation of $\mathcal{L}$ based on $\{\mathcal{K}_t\}_{t \geq 0}$, it is intuitive to take the operator logarithm such that $\mathcal{L} = \frac{1}{t} \log \mathcal{K}_t$. When $\mathcal{L}$ is bounded, its spectrum's sector should be confined to make the logarithm a single-valued mapping [145, 144]. However, for an unbounded $\mathcal{L}$, there is no direct connection $\mathcal{L}$ and $\{\mathcal{K}_t\}_{t \geq 0}$. In this subsection, we review how $\mathcal{L}$ can be properly approximated based on $\{\mathcal{K}_t\}_{t \geq 0}$.

5.4.1 Asymptotic Approximations of Generators

First, we examine the base case where $\{\mathcal{K}_t\}_{t\geq 0}$ is a contraction semigroup. In this situation, by the famous Hille-Yosida theorem [109, Theorem 3.1, Chapter 1], the resolvent set is such that $\rho(\mathcal{L}) \supseteq \mathbb{R}^+$, and for each $\lambda \in \rho(\mathcal{L})$, we have $\|R(\lambda;\mathcal{L})\| \leq 1/\operatorname{Re}(\lambda)$. Consequently, by [109, Lemma 3.3, Chapter 1], $\{\mathcal{L}_\lambda\}_{\lambda>0}$ converges in a strong sense to $\mathcal{L}$ (w.r.t. $\|\cdot\|_{\mathcal{L}}$), i.e.,

$$\operatorname*{s-lim}_{\lambda\to\infty} \mathcal{L}_\lambda = \operatorname*{s-lim}_{\lambda\to\infty} \lambda^2 R(\lambda;\mathcal{A}) - \lambda \operatorname{Id} = \mathcal{L}. \tag{5.12}$$

Remark 5.4.1. *Due to the contraction restriction of $\{\mathcal{K}_t\}_{t\geq 0}$, the system (5.1) is necessarily* dissipative *[109, Theorem 4.3, Chapter 1]. As characterized in [136], it is equivalent to verifying the existence of a value (storage) function $v : \Omega \to \mathbb{R}^+$ and a (supply) function $\eta : \Omega \to \mathbb{R}$ satisfying $\int_s^t |\mathcal{K}_\tau \eta(x)| d\tau < \infty$, for all $s \leq t$ and all $x \in \Omega$, such that $\mathcal{K}_{t-s} v(x) - v(x) \leq \int_s^t \mathcal{K}_\tau \eta(x) d\tau$. In other words, the energy storage rate of the system cannot exceed η. This assumption is somewhat restrictive when considering unknown systems.* ◇

To accommodate more general cases, we propose the following extension of the Yosida approximation on $\mathbb{R}^+$. To achieve this, we first present the following facts.

Proposition 5.4.2. *For system (5.1), there exist constants $\omega \geq 0$ and $C \geq 1$ such that $\|\mathcal{K}_t\| \leq Ce^{\omega t}$ for all $t \geq 0$. In addition, for any $\lambda \in \mathbb{C}$, the family $\{\mathcal{K}_{t,\lambda}\}_{t\geq 0}$, where*

$$\mathcal{K}_{t,\lambda} := e^{\lambda t}\mathcal{K}_t \tag{5.13}$$

is a $\mathcal{C}_0$-semigroup with generator $\mathcal{L} + \lambda \operatorname{Id} : \operatorname{dom}(\mathcal{L}) \to \mathcal{C}(\Omega)$.

Proof: The first part follows directly by [109, Theorem 1.2.2]. The semigroup property of $\{\mathcal{K}_{t,\lambda}\}_{t\geq 0}$ follows easily by the definition. To verify its generator, we have

$$\frac{\mathcal{K}_{t,\lambda}h - h}{t} = \frac{e^{\lambda t}\mathcal{K}_t h - e^{\lambda t}h}{t} + \frac{e^{\lambda t}h - h}{t}. \tag{5.14}$$

For all $h \in \operatorname{dom}(\mathcal{L})$, the limit exists as $t \downarrow 0$ for each r.h.s. term. It then follows that $\lim_{t\downarrow 0} \frac{\mathcal{K}_{t,\lambda}h - h}{t} = \mathcal{L} + \lambda \operatorname{Id}$. ■

Although it has been demonstrated in extensive literature that $\text{s-}\lim_{\lambda\to\infty} \mathcal{L}_\lambda = \mathcal{L}$ for $\{\mathcal{L}_\lambda\}_{\lambda>\omega}$, to better understand how data-driven approaches can be integrated into the approximation scheme, we prove the following theorem and demonstrate an explicit convergence rate.

Theorem 5.4.3. *For system (5.1), consider* $\{\mathcal{L}_\lambda\}_{\lambda\geq\tilde{\omega}}$, *where* $\tilde{\omega}\geq\omega>0$. *Then,* s-$\lim_{\lambda\to\infty}\mathcal{L}_\lambda=\mathcal{L}$. *Moreover, for any* $h\in\mathcal{C}^2(\Omega)$, *there exists an* $\widetilde{C}>0$ *such that*

$$\|\mathcal{L}_\lambda h-\mathcal{L}h\|\leq\widetilde{C}(\lambda-\tilde{\omega})^{-1}(\|h\|_{\mathcal{L}}+\|\mathcal{L}h\|_{\mathcal{L}}).$$

Proof: For any $\tilde{\omega}\geq\omega$, we consider $\{\mathcal{K}_{t,-\tilde{\omega}}\}_{t\geq0}$, where $\mathcal{K}_{t,-\tilde{\omega}}$ is defined as (5.13). It is clear that $\|\mathcal{K}_{t,-\tilde{\omega}}\|\leq C$ for all $t\geq0$. By Proposition 5.4.2, $\{\mathcal{K}_{t,-\tilde{\omega}}\}_{t\geq0}$ also admits the generator $\mathcal{L}-\tilde{\omega}\,\mathrm{Id}$. Within $C^1(\Omega)$, for any $\lambda>\tilde{\omega}$, we have

$$\begin{aligned}\mathcal{L}-\mathcal{L}_\lambda=&\mathcal{L}-\tilde{\omega}\,\mathrm{Id}+\tilde{\omega}\,\mathrm{Id}-\mathcal{L}_\lambda\\=&(\mathcal{L}-\tilde{\omega}\,\mathrm{Id}-(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}})+((\mathcal{L}-\tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}}+\tilde{\omega}\,\mathrm{Id}-\mathcal{L}_\lambda)\\=:&\mathcal{O}_1+\mathcal{O}_2,\end{aligned}\tag{5.15}$$

where $(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}}$ is the Yosida approximation of $\mathcal{L}-\tilde{\omega}\,\mathrm{Id}$. It suffices to show the bound for each of the two terms on the r.h.s. of (5.15).

To show the bound for $\mathcal{O}_1$, we consider an alternative norm for $h\in\mathcal{C}^1(\Omega)$, defined as $\|h\|_\infty=\sup_{t\geq0}\|\mathcal{K}_{t,-\tilde{\omega}}h\|$. It can be verified that $\|h\|\leq\|h\|_\infty\leq C\|h\|$. For $\|h\|_{\mathcal{L}}$, we define the alternative norm $\|h\|_{\mathcal{L},\mathrm{alt}}=\|h\|_\infty+\|\mathcal{L}h\|_\infty$. In addition, $\|\mathcal{K}_{t,-\tilde{\omega}}h\|_\infty=\sup_{s\geq0}\|\mathcal{K}_{s,-\tilde{\omega}}\mathcal{K}_{t,-\tilde{\omega}}h\|\leq\sup_{s\geq0}\|\mathcal{K}_{s,-\tilde{\omega}}h\|=\|h\|_\infty$, which demonstrates the contraction property w.r.t. $\|\cdot\|_\infty$. Then, by the definition of the resolvent operator, we have $((\lambda-\tilde{\omega})I-(\mathcal{L}-\tilde{\omega}\,\mathrm{Id}))\mathcal{R}(\lambda-\tilde{\omega};\mathcal{L}-\tilde{\omega}\,\mathrm{Id})=\mathrm{Id}$, which implies

$$((\lambda-\tilde{\omega})I-(\mathcal{L}-\tilde{\omega}\,\mathrm{Id}))\mathcal{R}(\lambda-\tilde{\omega};\mathcal{L}-\tilde{\omega}\,\mathrm{Id})(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})=\mathcal{L}-\tilde{\omega}\,\mathrm{Id}\,.\tag{5.16}$$

Recall the definition of the Yosida approximation for $(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}}$. For any $h\in\mathcal{C}^2(\Omega)$, by expanding (5.16), we can obtain $\mathcal{O}_1h=-(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})\mathcal{R}(\lambda-\tilde{\omega};\mathcal{L}-\tilde{\omega}\,\mathrm{Id})(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})h$. Consequently,

$$\begin{aligned}|\mathcal{O}_1h|_\infty\leq\|O_1h\|_\infty&\leq(\lambda-\tilde{\omega})^{-1}\|(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})h\|_{\mathcal{L}-\tilde{\omega}\,\mathrm{Id},\mathrm{alt}}\\&\leq C(\lambda-\tilde{\omega})^{-1}\|(\mathcal{L}-\tilde{\omega}\,\mathrm{Id})h\|_{\mathcal{L}-\tilde{\omega}\,\mathrm{Id}}\\&\leq C(\lambda-\tilde{\omega})^{-1}(\|\mathcal{L}h\|_{\mathcal{L}}+\tilde{\omega}\|h\|_{\mathcal{L}}+\tilde{\omega}\|h\|)\\&\leq C(\lambda-\tilde{\omega})^{-1}(\|\mathcal{L}h\|_{\mathcal{L}}+(2\tilde{\omega}+1)\|h\|_{\mathcal{L}}),\end{aligned}$$

where the second inequality can be proved in the same way as in [109, Lemma 3.2, Chapter 1], and the last two inequalities are based on the definitions of the norms $\|\cdot\|_{\mathcal{L}-\tilde{\omega}\,\mathrm{Id}}$ and

$\|\cdot\|_{\mathcal{L}}$, obtained by expanding them. Since $\mathcal{C}^2(\Omega)$ is dense in $\mathcal{C}^1(\Omega)$, and the operator $\mathcal{O}_1$ is uniformly bounded [109, Theorem 3.1, Chapter 1] and hence continuous on $\mathcal{C}^1(\Omega)$, we have

$$\mathcal{L} - \tilde{\omega}\,\mathrm{Id} = \operatorname*{s-lim}_{\lambda\to\infty}(\mathcal{L} - \tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}} \tag{5.17}$$

on $(\mathcal{C}^1(\Omega), \|\cdot\|_{\mathcal{L},\mathrm{alt}})$. Due to the norm equivalence between $\|\cdot\|_{\mathcal{L},\mathrm{alt}}$ and $\|\cdot\|_{\mathcal{L}}$, the convergence (5.17) also holds on $(\mathcal{C}^1(\Omega), \|\cdot\|_{\mathcal{L}})$.

We now work on the bound for $\mathcal{O}_2$. One can show by a direct calculation that, for any $h \in \mathcal{C}^1(\Omega)$,

$$\begin{aligned}\|\mathcal{O}_2 h\| &= \|2\tilde{\omega}h - \tilde{\omega}(2\lambda - \tilde{\omega})R(\lambda;\mathcal{L})h\| \\ &= \|\tilde{\omega}(\tilde{\omega}R(\lambda;\mathcal{L}) - 2\mathcal{L}R(\lambda;\mathcal{L}))h\| \\ &\le C(\lambda-\tilde{\omega})^{-1}(\tilde{\omega}^2\|h\| + 2\tilde{\omega}\|\mathcal{L}h\|) \\ &\le C_0(\lambda-\tilde{\omega})^{-1}\|h\|_{\mathcal{L}},\end{aligned} \tag{5.18}$$

where $C_0 = C \cdot \max\{\tilde{\omega}^2, 2\tilde{\omega}\}$.

The conclusion follows by combining both parts and considering $\lambda \to \infty$. ■

Remark 5.4.4. *In the proof of Theorem 5.4.3, we have implicitly demonstrated the effects of C and ω in the semigroup estimation $\|\mathcal{K}_t\| \le Ce^{\omega t}$. Intuitively, C represents the uniform scaling of the magnitude of the Koopman operator, while ω indicates the dominant exponential growth or decay rate of the flow on $\mathcal{C}(\Omega)$.*

Specifically, one can shift the original generator $\mathcal{L}$ to a stable generator $\mathcal{L} - \tilde{\omega}\,\mathrm{Id}$, which generates a contraction semigroup w.r.t. a norm equivalent to $\|\cdot\|$, and then shift back to $\mathcal{L}$ by adding $\tilde{\omega}\,\mathrm{Id}$. The analysis in the proof determines the error ($\mathcal{O}_1$) between the stable generator $\mathcal{L} - \tilde{\omega}\,\mathrm{Id}$ and its Yosida approximation, as well as the error ($\mathcal{O}_2$) between the approximation $\mathcal{L}_\lambda$ of the original generator and the approximation $(\mathcal{L} - \tilde{\omega}\,\mathrm{Id})_{\lambda-\tilde{\omega}} + \tilde{\omega}\,\mathrm{Id}$ of the "shift-back" operator.

Note that this shifting strategy used in the proof is advantageous for applying the analysis of contraction semigroups, but the shift inevitably increases the cost. Nonetheless, the overall error of using the Yosida approximation on $\rho(\mathcal{L})$ is always dominated by a reciprocal term.

5.4.2 Representation of Resolvent Operators

Motivated by representing $\mathcal{L}$ by $\{\mathcal{K}_t\}_{t\ge0}$ and the Yosida approximation for $\mathcal{L}$ on $\{\lambda > \omega\}$, we establish a connection between $\mathcal{R}(\lambda;\mathcal{L})$ and $\{\mathcal{K}_t\}_{t\ge0}$.

Proposition 5.4.5. *Let $\mathcal{R}(\lambda)$ on $\mathcal{C}(\Omega)$ be defined by*

$$\mathcal{R}(\lambda)h := \int_0^\infty e^{-\lambda t}(\mathcal{K}_t h)dt. \tag{5.19}$$

Then, for all $\lambda > \omega$,

1. *$\mathcal{R}(\lambda)(\lambda \operatorname{Id} - \mathcal{L})h = h$ for all $h \in \operatorname{dom}(\mathcal{L})$;*
2. *$(\lambda \operatorname{Id} - \mathcal{L})\mathcal{R}(\lambda)h = h$ for all $h \in \mathcal{C}(\Omega)$.*

Proof: For all $h \in \operatorname{dom}(\mathcal{L})$ and $x \in \Omega$, we have

$$\begin{aligned}
&\mathcal{R}(\lambda)(\lambda \operatorname{Id} - \mathcal{L})h(x)\\
=&\lambda\mathcal{R}(\lambda)h(x) - \mathcal{R}(\lambda)\mathcal{L}h(x)\\
=&\lambda \int_0^\infty e^{-\lambda t}h(\phi(t,x))dt - \int_0^\infty e^{-\lambda t}\mathcal{L}h(\phi(t,x)dt\\
=&-h(\phi(t,x))e^{-\lambda t}\big|_0^\infty + \int_0^\infty e^{-\lambda t}d(h(\phi(t,x)) - \int_0^\infty e^{-\lambda t}\mathcal{L}h(\phi(t,x)dt\\
=&h(\phi(0,x)) = h(x),
\end{aligned}$$

where we have used the fact that the time derivative along the trajectories of (5.1) is such that $dh(\phi(t,x))/dt = \mathcal{L}h(\phi(t,x))$.

Recall (5.13). For all $h \in \mathcal{C}(\Omega)$, all $x \in \Omega$, and all $t \geq 0$,

$$\begin{aligned}
&\mathcal{R}(\lambda)h(x)\\
=&\int_0^t \mathcal{K}_{s,-\lambda}h(x)ds + \int_t^\infty \mathcal{K}_{s,-\lambda}h(x)ds\\
=&\int_0^t \mathcal{K}_{s,-\lambda}h(x)ds + \int_0^\infty \mathcal{K}_{t+s,-\lambda}h(x)ds\\
=&\int_0^t e^{-\lambda s}\mathcal{K}_s h(x)ds + e^{-\lambda t}\mathcal{R}(\lambda)h(\phi(t,x)).
\end{aligned}$$

However,

$$\mathcal{K}_t\mathcal{R}(\lambda)h(x) = \int_0^\infty e^{-\lambda s}\mathcal{K}_s h(\phi(t,x))ds = \mathcal{R}(\lambda)\mathcal{K}_t h(x).$$

Therefore,

$$\begin{aligned}
&\frac{\mathcal{K}_t\mathcal{R}(\lambda)h(x)-\mathcal{R}(\lambda)h(x)}{t}\\
=&\frac{\mathcal{K}_tR(\lambda)h(x)-e^{-\lambda t}\mathcal{R}(\lambda)h(\phi(t,x))}{t}\\
&+\frac{e^{-\lambda t}\mathcal{R}(\lambda)h(\phi(t,x))-\mathcal{R}(\lambda)h(x)}{t}\\
=&\frac{\mathcal{K}_t\mathcal{R}(\lambda)h(x)-e^{-\lambda t}\mathcal{K}_t\mathcal{R}(\lambda)h(x)}{t}-\frac{1}{t}\int_0^t e^{-\lambda s}\mathcal{K}_sh(x)ds.
\end{aligned}$$

With $t\downarrow 0$ on both sides, we have $\mathcal{L}\mathcal{R}(\lambda)h(x)=\lambda\mathcal{R}(\lambda)h(x)-h(x)$, which is equivalent as $(\lambda\,\mathrm{Id}-\mathcal{L})\mathcal{R}(\lambda)h(x)=h(x)$. ■

The family of $\{\mathcal{R}(\lambda)\}$ serves as pseudo-resolvent operators. It becomes a true resolvent family, i.e., satisfies the commutative property between $\mathcal{R}(\lambda)$ and $\lambda\,\mathrm{Id}-\mathcal{L}$, only when the inverse mapping of $\mathcal{R}(\lambda)$ is defined on its range $\mathrm{dom}(\mathcal{L})$ rather than on the entire $\mathcal{C}(\Omega)$.

Note that the pseudo-resolvent need not be a resolvent for the restriction of $\mathcal{L}$ to a subdomain, not even for the *pre-generator* $\mathcal{L}|_{\mathcal{C}^1(\Omega)}$. In particular, if $\mathcal{L}|_{\mathcal{C}^1(\Omega)}$ is not closed, Proposition 5.4.5-2) (asserting surjectivity of $\lambda\,\mathrm{Id}-\mathcal{L}|_{\mathcal{C}^1(\Omega)}$) may fail; in that case one may only have $\overline{(\lambda\,\mathrm{Id}-\mathcal{L})(\mathcal{C}^1(\Omega))}=\mathcal{C}(\Omega)$. We illustrate this fact in Example 5.4.6.

Example 5.4.6. *Consider system $\dot{x}=-x^3$ on $\Omega=(-a,a)$ for any $a>0$, whose solution is given by $\phi(t,x)=\frac{x}{\sqrt{1+2tx^2}}$. It can be verified that $\mathcal{C}^1(\Omega)\subsetneq\mathrm{dom}(\mathcal{L})$, with a counterexample given by the continuous function $h(x)=|x|^{1/2}$, which lies in $\mathrm{dom}(\mathcal{L})$ but not in $\mathcal{C}^1(\Omega)$. Indeed, $\mathcal{L}h(x)=\lim_{t\downarrow 0}\sqrt{|x|}\cdot\frac{(1+2tx^2)^{-1/4}-1}{t}=-\frac{|x|^{5/2}}{2}$ for all $x\in\Omega$.*

Now, let $g:=(\lambda\,\mathrm{Id}-\mathcal{L})h\in\mathcal{C}(\Omega)$. Then, by Proposition 5.4.5-1), we have $\mathcal{R}(\lambda)g=h\notin\mathcal{C}^1(\Omega)$. Hence $(\lambda\,\mathrm{Id}-\mathcal{L}|_{\mathcal{C}^1(\Omega)})\mathcal{R}(\lambda)g$ is not well defined, and Proposition 5.4.5-2) does not hold for $\mathcal{L}|_{\mathcal{C}^1(\Omega)}$. This fact implies that $\mathcal{L}|_{\mathcal{C}^1(\Omega)}$ is not closed on $\mathcal{C}^1(\Omega)$, and one cannot characterize $(\lambda I-\mathcal{L}|_{\mathcal{C}^1(\Omega)})^{-1}$ via $\mathcal{R}(\lambda)$. Accordingly, $\rho(\mathcal{L}|_{\mathcal{C}^1(\Omega)})=\emptyset$; since if it is not, $(\lambda\,\mathrm{Id}-\mathcal{L}|_{\mathcal{C}^1(\Omega)})^{-1}$ is continuous for some λ, which implies that $\mathcal{L}|_{\mathcal{C}^1(\Omega)}$ is closed and contradicts the discussion above. ◇

Remark 5.4.7. *Even though $\mathcal{R}(\lambda)$ is well defined on $\mathcal{C}(\Omega)$, the commutative property between $\mathcal{R}(\lambda)$ and $\lambda\,\mathrm{Id}-\mathcal{L}$ only holds on $\mathcal{C}^1(\Omega)$. Consequently, $R(\lambda;\mathcal{L})$ and $R(\lambda)$ are equivalent only on $\mathcal{C}^1(\Omega)$. This domain aligns with the valid domain where $\mathcal{L}=\text{s-}\lim_{\lambda\to\infty}\mathcal{L}_\lambda$.* ◇

As a strong-sense approximation of $\mathcal{L}$ may not be feasible from the information in $\mathcal{R}(\lambda)$ via direct inversion, we do not pursue this inverse strategy and instead adopt the Yosida-type approximation, whose existence and boundedness are always guaranteed. To use the approximation, we replace $R(\lambda;\mathcal{L})$ with $R(\lambda)$. We can then immediately conclude the following representation.

Corollary 5.4.8. *For each $\lambda > \omega$,*

$$\mathcal{L}_\lambda = \lambda^2 \int_0^\infty e^{-\lambda t}\mathcal{K}_t dt - \lambda \,\mathrm{Id} \tag{5.20}$$

and $\mathcal{L}_\lambda \rightharpoonup \mathcal{L}$ on $\mathrm{dom}(\mathcal{L})$ as $\lambda \to \infty$.

In the case where $\rho(\mathcal{L})$ is empty, we establish the following results.

Proposition 5.4.9. *Consider $\hat{\mathcal{L}}_\lambda = \lambda^2\mathcal{R}(\lambda) - \lambda\,\mathrm{Id}$. Suppose $\rho(\mathcal{L}) = \emptyset$. Then, for all $h \in C^1(\Omega)$, $\hat{\mathcal{L}}_\lambda h \to \mathcal{L}h$.*

Proof: Let $\mathcal{L}^{(t)}h = \frac{\mathcal{K}_t h - h}{t}$. Then, $\mathcal{L}^{(t)}h \to \mathcal{L}h$ by definition. Define $\psi_\lambda(t) := \lambda e^{-\lambda t}$, and then we have the property $\int_0^\infty \psi_\lambda(t)dt = 1$ for all $\lambda > 0$. Then,

$$\begin{aligned}
\hat{\mathcal{L}}_\lambda h(x) &= \lambda^2 \int_0^\infty e^{-\lambda t} h(\phi(t;x))dt - \lambda h(x) \\
&= \lambda \left(\int_0^\infty \lambda e^{-\lambda t} h(\phi(t;x))dt - h(x) \right) \\
&= \lambda \left(\int_0^\infty \psi_\lambda(t)(h(\phi(t;x) - h(x))dt \right) \\
&= \int_0^\infty \lambda t \psi_\lambda(t) \mathcal{L}^{(t)}h(x)dt.
\end{aligned} \tag{5.21}$$

Define $k_\lambda(t) := \lambda t \psi_\lambda(t)$. We can show that $\int_0^\infty k_\lambda(t)dt = 1$ for all $\lambda > 0$. Furthermore, we have

$$\int_\varepsilon^\infty k_\lambda(t)dt = \lambda\varepsilon e^{-\lambda\varepsilon} + e^{-\lambda\varepsilon}, \tag{5.22}$$

which converges to 0 as $\lambda \to \infty$ for each fixed $\varepsilon > 0$. By (5.21),

$$|\hat{\mathcal{L}}_\lambda h(x) - \mathcal{L}h(x)| = \lim_{\varepsilon \downarrow 0} \underbrace{\int_\varepsilon^\infty k_\lambda(t)|\mathcal{L}^{(t)}h(x) - \mathcal{L}h(x)|dt}_{I_{\lambda,\varepsilon}}. \tag{5.23}$$

However, due to boundedness of $|\hat{\mathcal{L}}^{(t)}h(x) - \mathcal{L}h(x)|$ on Ω, we have $I_{\lambda,\varepsilon} \leq C\int_{\varepsilon}^{\infty} k_\lambda(t)dt \to 0$ as $\lambda \to \infty$ for each $\varepsilon > 0$, for some constant $C > 0$. Sending $\varepsilon \to 0$, the conclusion follows. ■

Remark 5.4.10. *A mapping $\mathcal{R} : \Delta \subseteq \mathbb{C} \to L(H)$, where H is a function space, is called pseudo-resolvent if $\mathcal{R}(\lambda) - \mathcal{R}(\mu) = (\mu - \lambda)\mathcal{R}(\mu)\mathcal{R}(\lambda)$ on Δ. A pseudo-resolvent $\mathcal{R}$ is called of L_∞-type $\lambda\mathcal{R}(\lambda) \rightharpoonup \mathrm{Id}$. Clearly, the integral operator (5.19) is an L_∞-type pseudo resolvent. It can be verified that $\overline{\mathcal{R}(\lambda)(H)} = H$, and if $\ker \mathcal{R}(\lambda) = \{0\}$ for all $\lambda \in \Delta$, then there exists a closed operator $V : D \to H$ where $\overline{D} = H$ such that $\mathcal{R}(\lambda) = (\lambda\,\mathrm{Id} - V)^{-1}$. The tricky part is that $\mathcal{L}$ may not be closed on D, and we may not have $\overline{D} = H$; hence, $\mathcal{L} \neq V$ in general.*

The remainder of this chapter builds on this Yosida-type approximation (5.20) and the evaluation of $\mathcal{R}(\lambda)$ using Koopman-related information in the integral to learn $\mathcal{L}_\lambda$ from data.

5.5 Koopman-Based Finite-Dimensional Approximation of Generators

The rest of this chapter focuses on learning a bounded operator $\mathcal{R}(\lambda)$ — and hence $\mathcal{L}_\lambda$— using the dictionary of observable functions Z_N introduced in Definition 5.3.6. This process is conceptually similar to the conventional Koopman learning framework, and also resembles many recent works on learning Koopman-related bounded operators, if input data from the domain and output data from the range are accessible. The current form of $\int_0^\infty e^{-\lambda t}(\mathcal{K}_t\cdot)dt$ in (5.20) is not advantageous for data-driven approximation. In this section, we derive a finite-horizon, finite-dimensional approximation, demonstrating the feasibility of using observable data to learn the generator.

5.5.1 Finite Time Horizon Approximation

Observing the form of (5.20), we first define the following truncation integral operator.

Definition 5.5.1. *For any $h \in \mathcal{C}(\Omega)$ and $\tau_s \geq 0$, we define $\mathcal{R}_{\lambda,\tau_s} : \mathcal{C}(\Omega) \to \mathcal{C}(\Omega)$ as*

$$\mathcal{R}_{\lambda,\tau_s}h(x) := \int_0^{\tau_s} e^{-\lambda s}\mathcal{K}_s h(x)ds. \tag{5.24}$$

We aim to demonstrate that for any sufficiently large $\lambda \in \mathbb{R}$, the aforementioned truncation of the integral will not significantly "hurt" the accuracy of the approximation (5.20).

Theorem 5.5.2. *Let $\tau_s \geq 0$ and $\lambda > \omega$ be fixed. Then, $\|\lambda^2 \mathcal{R}_{\lambda,\tau_s} - \lambda \,\mathrm{Id} - \mathcal{L}_\lambda\| \leq \frac{C\lambda^2}{\lambda-\omega} e^{-\lambda\tau_s}$ on $\mathrm{dom}(\mathcal{L})$.*

Proof: Note that, for any $\lambda > \omega$,

$$\begin{aligned} \|\mathcal{R}(\lambda)\| &\leq \int_0^\infty e^{-\lambda t}\|\mathcal{K}_t\| dt \\ &\leq \int_0^\infty C e^{-\lambda t} e^{\omega t} dt = \frac{C}{\lambda - \omega}. \end{aligned} \tag{5.25}$$

Therefore, for any $h \in C(\Omega)$,

$$\begin{aligned} \|\mathcal{R}_{\lambda,\tau_s} h - \mathcal{R}(\lambda;\mathcal{L})h\| &= \sup_{x\in\Omega} |e^{-\lambda\tau_s}\mathcal{R}(\lambda)h(\phi(\tau_s,x))| \\ \leq e^{-\lambda\tau_s}\|\mathcal{R}(\lambda)\|\|h\| &\leq e^{-\lambda\tau_s}\frac{C}{\lambda-\omega}\|h\| \end{aligned} \tag{5.26}$$

and $\sup_{\|h\|=1} \|\lambda^2\mathcal{R}_{\lambda,\tau_s} h - \lambda h - \mathcal{L}_\lambda h\| \leq \frac{C\lambda^2}{\lambda-\omega} e^{-\lambda\tau_s}$, which completes the proof. ■

We notice that, for any fixed τ_s, the error bound $\frac{C\lambda^2}{\lambda-\omega} e^{-\lambda\tau_s}$ in Theorem 5.5.2 demonstrates an exponential decaying rate in the uniform sense as $\lambda \to \infty$ any fixed $\tau_s > 0$. Recalling the convergence rate of $\mathcal{L}_\lambda \rightharpoonup \mathcal{L}$ in Theorem 5.4.3, we note that the convergence in Theorem 5.5.2 does not dominate for any fixed $\tau_s > 0$. This allows us to use

$$\mathcal{L}_{\lambda,\tau_s} := \lambda^2 \mathcal{R}_{\lambda,\tau_s} - \lambda \,\mathrm{Id} \tag{5.27}$$

to approximate $\mathcal{L}$ within a small time horizon. We use the following example to illustrate this approximation. Furthermore, we also attempt to use a relatively small τ_s to reduce the data size for the integral evaluation, as seen in Section 5.6.3.

Example 5.5.3. *Consider the simple dynamical system $\dot{x} = x$ and the observable function $v(x) = x^n$ for any $n \geq 1$. Then, analytically, $\phi(s,x) = xe^s$ for all $s \geq 0$ and $\mathcal{L}v(x) = nx^n$. We test the validity of using Eq. (5.27). Note that, for sufficiently large λ, we have $\lambda^2 \int_0^{\tau_s} e^{-\lambda s}(\mathcal{K}_s v(x)) ds = \lambda^2 \int_0^{\tau_s} e^{-\lambda s} e^{sn} x^n ds = \lambda^2 x^n \int_0^{\tau_s} e^{-(\lambda-n)s} ds = \frac{\lambda^2 x^n}{\lambda-n}(1 - e^{-(\lambda-n)\tau_s}) \approx \frac{\lambda^2 x^n}{\lambda-n}$, and $\lambda^2\mathcal{R}_{\lambda,\tau_s} v(x) - \lambda v(x) \approx \frac{\lambda^2 x^n}{\lambda-n} - \lambda x^n = \frac{n\lambda}{\lambda-n} x^n \approx nx^n = \mathcal{L}v(x)$. With high-accuracy evaluation of the truncated integral, we can achieve a reasonably good approximation.* ◇

5.5.2 Finite-Rank Approximation

Based on (5.27), it suffices to learn the operator $\mathcal{R}_{\lambda,\tau_s}$ for any fixed $\tau_s > 0$ and to predict the image of the operator when acting on some $\mathcal{C}(\Omega)$ function. In favor of a learning-based approach based on a dictionary of a finite number of observable functions serving as basis functions, we verify if $\mathcal{R}_{\lambda,\tau_s}$ is representable as a finite-rank operator.

We first look at the following property of $\mathcal{R}_{\lambda,\tau_s}$.

Proposition 5.5.4. *For any $\lambda > \omega$, the operator $\mathcal{R}_{\lambda,\tau_s}$ is compact if and only if $\mathcal{K}_s$ is compact for any $s \in (0, \tau_s]$.*

Proof: We assume that $\mathcal{R}_{\lambda,\tau_s}$ is compact for $\lambda > \omega$. By [109, Theorem 2.2, Chapter 1], $\mathcal{K}_s$ is continuous w.r.t. the uniform operator norm $\|\cdot\|$. We can also easily verify the compactness of $\lambda\mathcal{R}_{\lambda,\tau_s}\mathcal{K}_s$, for any $s \in (0, \tau_s]$, by Definition 5.5.1. In addition, for any arbitrarily small $t > 0$,

$$
\begin{aligned}
&\|\lambda\mathcal{R}_{\lambda,\tau_s}\mathcal{K}_s - \mathcal{K}_s\| \\
\leq &\int_0^t \lambda e^{-\lambda\sigma}\|\mathcal{K}_{s+\sigma} - \mathcal{K}_s\|d\sigma + \int_t^{\tau_s} \lambda e^{-\lambda\sigma}\|\mathcal{K}_{s+\sigma} - \mathcal{K}_s\|d\sigma \\
\leq &\sup_{\sigma\in(0,t]} \|\mathcal{K}_{s+\sigma} - \mathcal{K}_s\|(1 - e^{-\lambda t}) + 2\int_t^{\tau_s} \lambda e^{-\lambda\sigma}\|\mathcal{K}_{\tau_s}\|d\sigma \\
\leq &\sup_{\sigma\in(0,t]} \|\mathcal{K}_{s+\sigma} - \mathcal{K}_s\| + \frac{2C\lambda}{\lambda - \omega} e^{\omega(t+\tau_s)-\lambda t}.
\end{aligned}
$$

By the continuity of $\mathcal{K}_s$, letting $\lambda \to \infty$ and $t \to 0$, we have $\mathcal{K}_s \to \lambda\mathcal{R}_{\lambda,\tau_s}\mathcal{K}_s$ in the uniform sense, which shows the compactness.

To show the converse side, we notice that the operator $\mathcal{R}^t_{\lambda,\tau_s} := \int_t^\tau e^{-\lambda s}\mathcal{K}_s ds$ is always compact for any $t \in (0, \tau_s]$ given the compactness of $\mathcal{K}_s$ with $s \in (0, \tau_s]$. However,

$$
\begin{aligned}
&\|\mathcal{R}_{\lambda,\tau_s} - \mathcal{R}^t_{\lambda,\tau_s}\| \\
\leq &\int_0^t e^{-\lambda s}\|\mathcal{K}_s\|ds \leq \int_0^t \|\mathcal{K}_s\|ds \\
\leq &\int_0^t Ce^{\omega s}ds \leq tCe^{\omega t}.
\end{aligned}
$$

Letting $t \to 0$, we see the uniform convergence of $\mathcal{R}^t_{\lambda,\tau_s}$ to $\mathcal{R}_{\lambda,\tau_s}$, which shows compactness of $\mathcal{R}_{\lambda,\tau_s}$. ■

It is worth noting that $\mathcal{K}_t$ of (5.1) is not necessarily compact for each $t > 0$. To show that $\mathcal{K}_t(B_r) \subseteq \mathcal{C}(\Omega)$ is relatively compact, where $B_r = \{h \in \mathcal{C}(\Omega) : \|h\| \leq r\}$ for some $r > 0$, one needs to verify the equicontinuity within $\mathcal{K}_t(B_r)$. However, this is not guaranteed. As a counterexample, we set $\Omega := (-1, 1)$, $h_n(x) = \sin(nx) \in B_1$ (or similarly, for the Fourier basis), and define $\phi(t, x) = e^{-t}x$ for all $x \in \Omega$. Then, the sequence $h_n \circ \phi(t, \cdot)$ for each t does not exhibit equicontinuity due to the rapid oscillation as n increases.

For the purpose of approximating $\mathcal{L}_\lambda$ strongly, we aim to find a compact approximation of $\{\mathcal{K}_t\}_{t>0}$ that enables a finite-rank representation of $\mathcal{R}_{\lambda,\tau_s}$ in the same sense.

Proposition 5.5.5. *Consider a smooth mollifier $\eta \in C^\infty_0(\Omega)$ with compact support $\mathbb{B}(0; 1)$. For each $\varepsilon > 0$, let $\eta_\varepsilon(x) := \frac{1}{\varepsilon^n}\eta\left(\frac{x}{\varepsilon}\right)$ with $\int_\Omega \eta(y)dy = 1$. For all $h \in \mathcal{C}(\Omega)$ and $\varepsilon > 0$, set $\mathcal{J}_\varepsilon h(x) = \int_\Omega \eta_\varepsilon(x-y)h(y)dy$. Define $\mathcal{K}^\varepsilon_t := \mathcal{J}_\varepsilon\mathcal{K}_t$. Then, for each $t > 0$, $\{\mathcal{K}^\varepsilon_t\}_{\varepsilon>0}$ is a family of compact linear operator and satisfies $\mathcal{K}^\varepsilon_t \rightharpoonup \mathcal{K}_t$ on $\mathcal{C}(\Omega)$ as $\varepsilon \to 0$. In addition, for each $t, s > 0$, there exists a family of compact linear operator $\{\mathcal{K}^\varepsilon_t\}_{\varepsilon>0}$, such that $\mathcal{K}^\varepsilon_t \circ \mathcal{K}^\varepsilon_s \rightharpoonup \mathcal{K}_t \circ \mathcal{K}_s$ on $h \in \mathcal{C}(\Omega)$ as $\varepsilon \to 0$.*

We omit the proof as it follows similar arguments presented in [91, Section V.A]. Heuristically, the bump functions η_ε converge weakly to Dirac measures centered at their respective flow locations, distributing point masses that $\{\mathcal{K}_t\}$ transports along the trajectories. Taking advantage of the compactness approximation, the following statement demonstrates the feasibility of approximating $\mathcal{R}_{\lambda,\tau_s}$ by a finite-rank operator.

Corollary 5.5.6. *For any fixed $T > 0$, for any arbitrarily small $\delta > 0$, there exists a sufficiently large N and a finite-dimensional approximation $\mathcal{R}^N_{\lambda,\tau_s}$ such that $\|\mathcal{R}^N_{\lambda,\tau_s}h - \mathcal{R}_{\lambda,\tau_s}h\| < \delta$, $h \in \mathcal{C}(\Omega)$.*

Proof: We show the sketch of the proof. Working on the compact family of $\{\mathcal{K}^\varepsilon_s\}_{s\in(0,\tau_s]}$ for some small $\varepsilon > 0$, one can find a sufficiently large N such that

$$\|\mathcal{K}^N_s h - \mathcal{K}_s h\| < \delta, \quad h \in C(\Omega),\ s \in (0, \tau_s],$$

where K^N_s is the finite-dimensional representation of K^ε_s. Let $\mathcal{R}^N_{\lambda,\tau_s} := \int_0^{\tau_s} e^{-\lambda s}\mathcal{K}^N_s ds$. Then,

for any $t \in (0, \tau_s)$,

$$\begin{aligned}
&\|\mathcal{R}^N_{\lambda,\tau_s} h - \mathcal{R}_{\lambda,\tau_s} h\| \\
\leq &\int_0^t e^{-\lambda s} \|\mathcal{K}^N_s h - \mathcal{K}_s h\| ds + \int_t^{\tau_s} e^{-\lambda s} \|\mathcal{K}^N_s h - \mathcal{K}_s h\| ds \\
\leq &C\|h\| t + \sup_{s\in(0,\tau_s]} \|\mathcal{K}^N_s h - \mathcal{K}_s h\| \cdot \frac{e^{-\lambda t} - e^{-\lambda \tau_s}}{\lambda} \\
< &C\|h\| t + \delta,
\end{aligned}$$

where $C := \sup_{s\in(0,t]} \|K^N_s\| + Ce^{\omega t}$. The conclusion follows by sending $t \to 0$. ■

Remark 5.5.7. *Similar to [91, Section V.B], one can consider a Hilbert space $\mathcal{H} := L^2(\Omega) \supseteq \mathcal{C}(\Omega)$ as a special case. Let $\{\varphi_i\}_{i\in\mathbb{Z}} \subseteq \mathcal{H}$ be the possibly complex-valued eigenfunctions of $\mathcal{K}^\varepsilon_t$. Then, $\mathcal{K}^\varepsilon_t \cdot = \sum_{i\in\mathbb{Z}} \rho^\varepsilon_i \langle \cdot, \bar{\varphi}_i \rangle \varphi_i$ can be approximated by a finite-sum $\mathcal{K}^N_t \cdot = \sum_{i=-N}^N \rho^\varepsilon_i \langle \cdot, \bar{\varphi}_i \rangle \varphi_i$, where $\{\rho^\varepsilon_i\}_{i\in\mathbb{Z}}$ represents the set of eigenvalues of $\{\mathcal{K}^\varepsilon_t\}_{t\geq 0}$ and $\langle \bar{\varphi}_i, \varphi_i \rangle = 1$ for all i. Then, $\mathcal{R}^N_{\lambda,\tau_s}$ can be defined accordingly as shown in the proof of Corollary 5.5.6.* ◇

When dealing with linear operators in finite-dimensional spaces, the concept of basis equivalence ensures that different representations corresponding to different bases are related through similarity transformations. Thus, the operator's inherent properties remain invariant under a change of basis. Building on the feasibility of finite-dimensional approximation, the next section will investigate how to train $\mathcal{L}_{\lambda,\tau_s}$ using a finite dictionary of test functions.

5.6 Data-Driven Algorithm

We continue to discuss the data-driven learning based on the approximations discussed in Section 5.5 Similar to the learning of Koopman operators [137, 85, 91], obtaining a fully discretized version L of the bounded linear operator $\lambda^2 \mathcal{R}^N_{\lambda,\tau_s} - \lambda \,\mathrm{Id}$ based on the training data typically relies on the selection of a discrete dictionary of continuously differentiable observable functions defined in (5.8). Then, the following are expected to hold: 1) Let $(\mu_i, \xi_i)_{i=0}^{N-1}$ be the eigenvalues and eigenvectors of L. Let $(\rho_i, \varphi_i)_{i=0}^{N-1}$ be the eigenvalues and eigenfunctions of $\mathcal{L}$. Then, for each i,

$$\mu_i \approx \rho_i, \quad \varphi_i(x) \approx \mathcal{Z}_N(x)^\mathsf{T} \xi_i. \tag{5.28}$$

2) For any $h \in \text{span}\{\mathfrak{z}_0, \mathfrak{z}_1, \cdots, \mathfrak{z}_{N-1}\}$ such that $h(x) = \mathcal{Z}_N(x)^\mathsf{T}\theta$ for some column vector θ, we have that

$$\mathcal{L}h(\cdot) \approx \mathcal{L}_{\lambda,\tau_s}h(\cdot) \approx \lambda^2\mathcal{R}^N_{\lambda,\tau_s}h(\cdot) - \lambda h(\cdot) \approx \mathcal{Z}_N(\cdot)^\mathsf{T}(L\theta). \tag{5.29}$$

We can let $\widetilde{\mathcal{L}}h(x) := \mathcal{Z}_N(x)(L\theta)$ for simplicity. In this section, we modify the existing Koopman learning technique to obtain L such that (5.28) and (5.29) hold.

Remark 5.6.1. *Denoting $\widetilde{\mathcal{L}}h_\theta(x) := Z_N(x)^\mathsf{T}(L_\lambda\theta)$, the approximation in (5.29) also guarantee the convergence of $\{e^{t\widetilde{\mathcal{L}}}\}$ to the original semigroup $\{\mathcal{K}_t\}_{t\geq 0}$ for any $h_\theta(x) = Z_N(x)^\mathsf{T}\theta$. Indeed, letting $\mathcal{O} = \widetilde{\mathcal{L}} - \mathcal{L}_\lambda$, we have*

$$\begin{aligned} &\|e^{t\widetilde{\mathcal{L}}}h_\theta - \mathcal{K}_t h_\theta\| \\ \leq &\|e^{t\mathcal{O}}\|\|e^{t\mathcal{L}_\lambda}h_\theta - \mathcal{K}_t h_\theta\| + \|\mathcal{K}_t\|\|e^{t\mathcal{O}}h_\theta - h_\theta\| \\ \leq &\|e^{t\mathcal{O}}\|\|e^{t\mathcal{L}_\lambda}h_\theta - \mathcal{K}_t h_\theta\| + t\|\mathcal{K}_t\|e^{t\|\mathcal{O}\|}\|\mathcal{O}h_\theta\|. \end{aligned} \tag{5.30}$$

As $\lambda \to \infty$, the first part goes to 0 by Theorem 5.3.11. The second part goes to 0 by the sequence of approximations (in a strong sense) in (5.29). ◇

5.6.1 Generating Training Data

The training data is obtained in the following way. By randomly sampling M initial conditions $\{x^{(m)}\}_{m=0}^{M-1} \subseteq \Omega$ and fixing a τ_s, we stack the *features* into X, such that:

$$X = [\mathcal{Z}_N(x^{(0)}), \mathcal{Z}_N(x^{(1)}), \cdots, \mathcal{Z}_N(x^{(M-1)})]^\mathsf{T}. \tag{5.31}$$

For any fixed $\lambda > 0$, given a dictionary $\mathcal{Z}_N$ of the form (5.8), for each $\mathfrak{z}_i \in \mathcal{Z}_N$ and each $x^{(m)}$, we consider $\lambda^2\mathcal{R}_{\lambda,\tau_s}\mathfrak{z}_i(x^{(m)}) - \lambda\mathfrak{z}_i(x^{(m)}) = \lambda^2\int_0^{\tau_s} e^{-\lambda s}\mathfrak{z}_i(\phi(s, x^{(m)}))ds - \lambda\mathfrak{z}_i(x^{(m)})$ as the labels. To compute the integral, we employ numerical quadrature techniques for approximation. This approach inevitably requires discrete-time observations (snapshots) with an observation/sampling rate of γ Hz within the interval $[0, \tau_s]$ of each flow map $\phi(\tau_s, x^{(m)})$. We denote the number of snapshots as $\Gamma := \gamma\tau_s$. We stack the snapshot data in the following intermediate matrix:

$$\begin{aligned} U^{(m)}\ (= U^{(m)}_\Gamma) = &\lambda^2[\mathcal{Z}_N(\phi(0, x^{(m)})), \cdots, e^{-\frac{\lambda k\tau_s}{\Gamma}}\mathcal{Z}_N(\phi(\frac{k\tau_s}{\Gamma}, x^{(m)})), \\ &\cdots e^{-\frac{\lambda\tau_s}{\Gamma}}\mathcal{Z}_N(\phi(\tau, x^{(m)}))]^T,\ \ m \in \{0, 1, \cdots, M-1\}. \end{aligned} \tag{5.32}$$

Denote by $\mathcal{G}^{\lambda}_{[0,\tau_s]}(v)$, or simply $\mathcal{G}(v)$ for brevity, the Gauss–Legendre quadrature[3] based on the vector of points v within $[0, \tau_s]$, and denote by $U^{(m)}[:, j]$ the j^{th} column of $U^{(m)}$. The stack of labels is given by[4]

$$Y \ (= Y^{\Gamma}_{\lambda,\tau_s}) := \mathcal{I}^{\Gamma}_{\lambda,\tau_s} - \lambda X, \tag{5.33}$$

where

$$\mathcal{I}^{\Gamma}_{\lambda,\tau_s} = \begin{bmatrix} \mathcal{G}(U^{(0)}[:,0]) & \cdots & \mathcal{G}(U^{(0)}[:,N-1]) \\ \vdots & \ddots & \vdots \\ \mathcal{G}(U^{(m)}[:,0]) & \cdots & \mathcal{G}(U^{(m)}[:,N-1]) \\ \vdots & \ddots & \vdots \\ \mathcal{G}(U^{(M-1)}[:,0]) & \cdots & \mathcal{G}(U^{(M-1)}[:,N-1]) \end{bmatrix}. \tag{5.34}$$

We omit the algorithm for generating the training data (X, Y), as it follows straightforwardly from the data stacking shown in (5.31) and (5.33).

5.6.2 Extended Dynamic Mode Decomposition Algorithm

After obtaining (X, Y), we can find L by $L = \text{argmin}_{A\in\mathbb{C}^{N\times N}} \|Y - XA\|_F$. The L is given in closed-form as $L = \left(X^{\mathsf{T}}X\right)^{\dagger} X^{\mathsf{T}}Y$. Similar to Extended Dynamic Mode Decomposition (EDMD) [137] for learning Koopman operators, the approximations (5.28) and (5.29) can be guaranteed.

5.6.3 Modification Under Low Sampling Rate Constraints

Recall that $Y \ (= Y^{\Gamma}_{\lambda,\tau_s})$, or equivalently, $\mathcal{I}^{\Gamma}_{\lambda,\tau_s}$, is obtained through both analytical and numerical approximations. Among the three parameters, λ, τ_s, and Γ, τ_s is less important than the others according to Theorem 5.5.2, so we can set τ_s to be a relatively small fixed value. On the other hand, λ and Γ need to be large to reduce the error.

Empirically, for larger λ under any fixed $\tau_s > 0$, the numerical integration requires a larger Γ, which corresponds to a higher sampling frequency. Conversely, if Γ is restrictive

[3]Given the Laplace transform-type integral, we choose to use the Gauss–Legendre quadrature rule to improve accuracy for relatively smaller values of Γ. We omit the details of implementing the Gauss–Legendre quadrature numerically, as the algorithms are well-established in this field.

[4]The superscripts and subscripts in the notation indicate the dependence of the parameters, reflecting that the image function values of the generator are only achieved approximately. The λ and τ_s determine the analytical error, while the Γ determines the numerical error. When the dependence on any parameter is not emphasized, we can use shorthand notation without the corresponding superscript and subscript.

in practice (for example, in automatic vehicles or intelligent transportation systems, where sensors can collect state data at no more than a 100 Hz sensing rate), λ cannot be large to ensure accuracy when using numerical integration. Making λ and Γ both sufficiently large cannot be achieved without modifications based on (5.33).

To resolve this conflict, we employ the first resolvent identity

$$[(\lambda-\mu)\mathcal{R}(\mu)+\mathrm{Id}]\mathcal{R}(\lambda)=\mathcal{R}(\mu) \tag{5.35}$$

to connect two resolvent operators with different values λ and μ in the resolvent set. This identity allows us to learn a resolvent operator with a small μ (thus enabling the use of a small Γ for numerical integration) and then infer the resolvent operator with a large λ. To adapt this inference step into generator learning based on (5.33), we provide the formula for the modified L as

$$L_{\mathrm{mod}}=A^{\dagger}B, \tag{5.36}$$

where

$$A=\frac{\lambda-\mu}{\mu^2}\mathcal{I}_{\mu}+X,\ \text{and}\ B=\frac{\lambda}{\mu}\mathcal{I}_{\mu}-\lambda X, \tag{5.37}$$

and $\mathcal{I}_{\mu}$ $(=\mathcal{I}^{\Gamma}_{\mu,\tau_s})$ is evaluated for a small μ. The derivation of this formula is given below.

For any $h\in\mathcal{C}^1(\Omega)$, by the first resolvent identity (5.35), we have

$$[(\lambda-\mu)\mathcal{R}(\mu)+\mathrm{Id}](\lambda^2\mathcal{R}(\lambda)h(x)-\lambda h(x)+\lambda h(x))=\lambda^2\mathcal{R}(\mu)h(x). \tag{5.38}$$

Then,

$$[(\lambda-\mu)\mathcal{R}(\mu)+\mathrm{Id}](\mathcal{L}_{\lambda}h(x)+\lambda h(x))=\lambda^2\mathcal{R}(\mu)h(x),$$

and

$$\begin{aligned}&\mathcal{L}_{\lambda}[(\lambda-\mu)\mathcal{R}(\mu)+\mathrm{Id}]h(x)\\ =&\lambda^2\mathcal{R}(\mu)h(x)-\lambda(\lambda-\mu)\mathcal{R}(\mu)h(x)-\lambda h(x)\\ =&\lambda\mu\mathcal{R}(\mu)h(x)-\lambda h(x),\end{aligned}$$

where we have used the commutativity of $\mathcal{L}$ (or $\mathcal{L}_{\lambda}$) and its resolvent. Letting $\widetilde{\mathcal{I}}:=\mu^2\mathcal{R}(\mu)$, then,

$$\mathcal{L}_{\lambda}\left[\frac{\lambda-\mu}{\mu^2}\widetilde{\mathcal{I}}+\mathrm{Id}\right]h(x)=\left[\frac{\lambda}{\mu}\widetilde{\mathcal{I}}-\lambda\,\mathrm{Id}\right]h(x). \tag{5.39}$$

It is also clear that, in a strong sense, $\mathcal{L}\left[\frac{\lambda-\mu}{\mu^2}\widetilde{\mathcal{I}}+\mathrm{Id}\right]h(x) \approx \left[\frac{\lambda}{\mu}\widetilde{\mathcal{I}}-\lambda\,\mathrm{Id}\right]h(x)$. We then solve this equation using a data-fitting method.

One can obtain the fully discretized version of the operator $\frac{\lambda-\mu}{\mu^2}\widetilde{\mathcal{I}}+\mathrm{Id}$ in a similar way as $\left(X^\intercal X\right)^\dagger X^\intercal A$. Similarly, the fully discretized version of the operator $\frac{\lambda}{\mu}\widetilde{\mathcal{I}}-\lambda\,\mathrm{Id}$ is $\left(X^\intercal X\right)^\dagger X^\intercal B$. It can be shown that the data-driven version of (5.39) is such that $\mathcal{Z}_N(\cdot)\left(X^\intercal X\right)^\dagger X^\intercal A(L_{\mathrm{mod}}\theta) \approx \mathcal{Z}_N(\cdot)\left(X^\intercal X\right)^\dagger X^\intercal B\theta$. Solving it in the least squares sense gives the formula (5.37).

However, there are two main issues in approximating the Koopman generator using (5.36): 1) One needs a large τ_s to have an accurate finite time horizon approximation as defined in Definition 5.5.1; 2) The accuracy of the approximation is highly sensitive to the choice of μ, which we will further explain in detail with numerical results in the next section. Therefore, we have the following proposition to address the aforementioned issues.

Proposition 5.6.2. *For any $h \in \mathcal{C}(\Omega)$ and $\tau_s \geq 0$, $\mathcal{R}_{\mu,\tau_s}h(x) = \int_0^{\tau_s} e^{-\mu s}\mathcal{K}_s h(x)ds$. Then*

$$\mathcal{R}_\mu h(x) = \mathcal{R}_{\mu,\tau_s}h(x) + e^{-\mu\tau_s}\mathcal{K}_{\tau_s}\mathcal{R}_\mu h(x). \tag{5.40}$$

Proof: Note that $\mathcal{R}_\mu h(x) = \int_0^\infty e^{-\mu t}\mathcal{K}_t h(x)dt$. By change of variable,

$$\begin{aligned}
\int_{\tau_s}^\infty e^{-\mu t}\mathcal{K}_t h(x)dt &= \int_0^\infty e^{-\mu(s+\tau_s)}\mathcal{K}_{s+\tau_s}h(x)ds && (5.41)\\
&= e^{-\mu\tau_s}\int_0^\infty e^{-\mu s}\mathcal{K}_{s+\tau_s}h(x)ds && (5.42)\\
&= e^{-\mu\tau_s}\mathcal{K}_{\tau_s}\int_0^\infty e^{-\mu s}\mathcal{K}_s h(x)ds && (5.43)\\
&= e^{-\mu\tau_s}\mathcal{K}_{\tau_s}\mathcal{R}_\mu h(x), && (5.44)
\end{aligned}$$

which proves the statement. ■

Back to the data-driven algorithm, by (5.40), for some column vector $\Upsilon \in \mathbb{R}^{N\times N}$, we have

$$X\Upsilon \approx \frac{\mathcal{I}^\Gamma_{\mu,\tau_s}}{\mu^2} + e^{-\mu\tau_s}\Phi_{\tau_s}\Upsilon, \tag{5.45}$$

where

$$\Phi_{\tau_s} := [\mathcal{Z}_N(\phi(\tau_s, x^{(0)})), \mathcal{Z}_N(\phi(\tau_s, x^{(1)})), \cdots, \mathcal{Z}_N(\phi(\tau_s, x^{(M-1)}))]^\intercal. \tag{5.46}$$

By solving the following least-squares problem:

$$\Upsilon = \underset{\upsilon \in \mathbb{R}^N}{\arg\min} \left\| (X - e^{-\mu\tau_s}\Phi_{\tau_s})\upsilon - \frac{\mathcal{I}^{\Gamma}_{\mu,\tau_s}}{\mu^2} \right\|, \tag{5.47}$$

we have an approximation of $\mathcal{R}_\mu h(x) \approx X\Upsilon$. With that, by applying the first resolvent identity, we provide the following updated version of the data-driven approximation for L:

$$L_{\text{update}} = A_1^{\dagger} B_1, \tag{5.48}$$

where

$$A_1 = (\lambda - \mu)X\Upsilon + X, \text{ and } B_1 = \lambda\mu X\Upsilon - \lambda X. \tag{5.49}$$

We summarize the data-driven algorithm in Algorithm 5.

5.6.4 Discussion of Other Existing Methods

Recall Remark 5.3.7 on the FDM with an expression $\frac{\mathcal{K}_\tau - \mathrm{Id}}{\tau} \rightharpoonup \mathcal{L}$, where the convergence is well studied in [13]. We also revisit the benchmark Koopman Logarithm Method (KLM) based on the expression $\mathcal{L} = \frac{1}{\tau}\log(\mathcal{K}_\tau)$, as described in [85]. In this subsection, we discuss the data-driven algorithms for FDM and KLM, in preparation for the comparison in the case studies in Section 5.7.

Observing that the expressions for $\mathcal{L}$ in FDM and KLM rely on just one moment of the Koopman operator $\mathcal{K}_\tau$, the data-driven versions of these methods are divided into two steps: 1) learning $\mathcal{K}_\tau$; 2) transforming the learned $\mathcal{K}_\tau$ to $\mathcal{L}$, respectively. To make the sampling rate consistent with the RTM, we set $\tau := \tau_s/\Gamma$.

The corresponding data-driven approximations also rely on the selection of a discrete dictionary $\mathcal{Z}_N$, and similarly, the approximation is to achieve $\mathcal{L}h(\cdot) \approx \mathcal{Z}_N(\cdot)(L_i\theta)$ for $h(x) = \mathcal{Z}_N(x)\theta$ and for any $i \in \{\text{FDM}, \text{KLM}\}$. To do this, we fix a τ, and stack the *features* into X the same way as (5.31). The *labels* are stacked into Z:

$$Z = [\mathcal{Z}_N(\phi(\tau, x^{(0)}), \cdots, \mathcal{Z}_N(\phi(\tau, x^{(M-1)})]^{\mathsf{T}}. \tag{5.50}$$

After obtaining the training data (X, Z), we can find K (the fully discretized version of $\mathcal{K}_\tau$) by $K = \text{argmin}_{A \in \mathbb{C}^{N\times N}} \|Z - XA\|_F$. The K is given in closed-form as $K = \left(X^{\mathsf{T}}X\right)^{\dagger} X^{\mathsf{T}}Z$ [137]. The data-driven approximation for $\mathcal{L}$ based on FDM and KLM can be immediately obtained using K.

Algorithm 5: Resolvent-Type Koopman Generator Learning

Require: Dictionary $\mathcal{Z}_N$, user-defined μ and λ, initial conditions $\{x^{(m)}\}_{m=0}^{M-1} \subseteq \Omega$, τ_s, and snapshots $\phi(\frac{k\tau_s}{\Gamma}, x^{(m)})$ for $k = 0, 1, \ldots, \Gamma$.

1: Compute and stack X using (5.31):

$$X = [\mathcal{Z}_N(x^{(0)}), \mathcal{Z}_N(x^{(1)}), \cdots, \mathcal{Z}_N(x^{(M-1)})]^{\mathsf{T}}$$

2: Compute and stack $U^{(m)}$ with respect to μ using (5.32)

3: Compute and stack numerical integration matrix $\mathcal{I}^{\Gamma}_{\mu,\tau_s}$ using (5.34)

4: Stack Φ_{τ_s} using (5.46):

$$\Phi_{\tau_s} = [\mathcal{Z}_N(\phi(\tau_s, x^{(0)})), \mathcal{Z}_N(\phi(\tau_s, x^{(1)})), \cdots, \mathcal{Z}_N(\phi(\tau_s, x^{(M-1)}))]^{\mathsf{T}}$$

5: Solve for resolvent approximation Υ using (5.47):

$$\Upsilon = \underset{\upsilon \in \mathbb{R}^N}{\arg\min} \left\| (X - e^{-\mu\tau_s}\Phi_{\tau_s})\upsilon - \frac{\mathcal{I}^{\Gamma}_{\mu,\tau_s}}{\mu^2} \right\|$$

6: Compute matrices A_1 and B_1 using (5.49):

$$A_1 = (\lambda - \mu)X\Upsilon + X, \quad B_1 = \lambda\mu X\Upsilon - \lambda X$$

7: Compute $L_{\text{update}} = A_1^{\dagger} B_1$

8: **return** L_{update}

1. FDM: $L_{\text{FDM}} = (K - \text{Id})/\tau$, where $\text{Id} = \text{Id}_{N\times N}$ is the identity matrix in this expression.

2. KLM: $L_{\text{KLM}} = \log(K)/\tau$.

Remark 5.6.3. *Let $(\mu_i^K, \xi_i^K)_{i=0}^{N-1}$ be the eigenvalues and eigenvectors of K. Let $(\rho_i^K, \varphi_i^K)_{i=0}^{N-1}$ be the eigenvalues and eigenfunctions of $\mathcal{K}_\tau$. Similar to (5.28) and (5.29), for each i, we have $\varphi_i^K(x) \approx \mathcal{Z}_N(x)\xi_i^K$ and $\mathcal{K}_s h(\cdot) \approx \mathcal{Z}_N(\cdot)(K\theta)$ for $h(x) = \mathcal{Z}_N(x)\theta$.* ◇

It is worth noting that, even when $\mathcal{L}$ can be represented by $(\log \mathcal{K}_\tau)/\tau$, we cannot guarantee that $\frac{\log \mathcal{K}_\tau}{\tau} h(\cdot) \approx \mathcal{Z}_N(\cdot)(\frac{\log(K)}{\tau}\theta)$ as in KLM, not to mention the case where the above logarithm representation does not hold. Denoting $\Phi(\cdot) = [\varphi_0^K(\cdot), \varphi_1^K(\cdot), \cdots, \varphi_{N-1}^K(\cdot)]$ and $\Xi = [\xi_0^K, \xi_1^K, \cdots, \xi_{N-1}^K]$, then it is clear that $\Phi(\cdot) = \mathcal{Z}_N(\cdot)\Xi$. The (possibly complex-valued) rotation matrix Ξ establishes the connection between finite-dimensional eigenfunctions and dictionary functions through data-fitting, ensuring that any linear combination within $\mathcal{Z}_N$ can be equivalently represented using Φ with a cancellation of the imaginary parts.

This imaginary-part cancellation effect does not generally hold when applying the matrix logarithm. Suppose the imaginary parts account for a significantly large value, the mutual representation of Φ and $\mathcal{Z}_N$ does not match in the logarithmic scale. An exception holds unless the chosen dictionary is inherently rotation-free with respect to the true eigenfunctions [145], or there is direct access to the data for $\log(\mathcal{K}_\tau)$ allowing for direct training of the matrix. However, such conditions contravene our objective of leveraging Koopman data to conversely find the generator.

In comparison, the FDM and the resolvent-type approach approximate $\mathcal{L}$ regardless of its boundedness. These two methods enable learning of L without computing the logarithm, thus avoiding the potential occurrence of imaginary parts caused by basis rotation. However, as we will see in Section 5.7, the performance of the FDM degrades as the sampling frequency decreases.

Apart from these Koopman-based methods, to illustrate the efficacy of the proposed method, we also compare it with the well-known system identification method, SINDy [19]. However, sparsity analysis within the Koopman framework typically involves the spectral analysis of the operators, which is highly relevant to the choice of the dictionary of the observable functions. We plan to explore this promising research direction in future work. Meanwhile, to ensure a fair comparison, we also present a sparse variant of RTM, referred to as Sparse RTM (SRTM), by enforcing sparsity in the same manner as in SINDy, namely, using sequential thresholded least squares on a given validation dataset to promote sparsity in the learned weights.

5.7 Case Studies

In this section, we test the effectiveness of the RTM and its sparse variant SRTM and compare their performance to the two Koopman-based benchmark methods, FDM and KLM, along with SINDy. Particularly, we perform numerical examples for system identification of the following representative systems [85, 144]: the reversed Van der Pol oscillator (polynomial), two-machine power system (nonpolynomial), a system with rational vector fields (nonpolynomial), Lorenz-63 system (chaotic), Lorenz-96 system and a 7-dim biochemical system (high-dimensional chaotic), and a system at the bifurcation (empty resolvent set). We also show that the RTM is able to solve transport-related PDEs effectively. The research code can be found at https://github.com/RuikunZhou/Resolvent-Type-Operator-Learning.

We provide two measurements to demonstrate the error in system identification.

(1) The root mean squared error (RMSE) of flow

$$\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}} = \frac{1}{M}\sum_{m=0}^{M-1}\sqrt{\frac{1}{\Gamma_s}\sum_{k=1}^{\Gamma_s}|\phi(t_k, x^{(m)}) - \hat{\phi}(t_k, x^{(m)})|^2} \tag{5.51}$$

measures the root mean square difference between the actual time-series data $\{\phi(t_k, x^{(m)})\}_{k=0}^{\Gamma_s}$ and the estimated data $\{\hat{\phi}(t_k, x^{(m)})\}_{k=0}^{\Gamma_s}$ using the learned generator up to time T_s. Here, Γ_s represents the number of snapshots used to verify the performance (which is independent of Γ, used for the data collection procedure), and the time $t_k = kT_s/\Gamma_s$ corresponds to the sampling instances.

(2) For polynomial models, and if we use monomial dictionary functions, we can use the RMSE of the weights assigned to each monomial

$$\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}} := \sqrt{\frac{1}{dN}\sum_{j=1}^{d}\sum_{k=0}^{N-1}|\theta_k^j - \hat{\theta}_k^j|^2}. \tag{5.52}$$

Here, N represents the size the dictionary; d is the dimension of the system; θ_k^j is the weight for f_j at the k^{th} dictionary function; $\hat{\theta}_k^j$ is similarly defined but for the estimated vector field.

5.7.1 Reversed Van der Pol Oscillator

We use this example to thoroughly explain how the proposed method works and to highlight the phenomena discovered in the numerical example.

Consider the reversed Van der Pol oscillator

$$\dot{x}_1 = -x_2, \quad \dot{x}_2 = x_1 - (1 - x_1^2)x_2,$$

with $x := [x_1, x_2]^\mathsf{T}$. We assume the system dynamics are unknown to us, and our information is limited to the system dimension, $d = 2$, and observations of sampled trajectories. To generate training data, we simply select $\Omega = (-1, 1)^2$ and obtain a total of $M = 10^2$ uniformly sampled initial conditions $\left\{x^{(m)}\right\}_{m=0}^{M-1}$ in Ω. We choose the dictionary as

$$\mathcal{Z}_N(x) = [\mathfrak{z}_0(x), \mathfrak{z}_1(x), \cdots, \mathfrak{z}_{N-1}(x)], \ N = P \times Q, \tag{5.53}$$

where $\mathfrak{z}_i(x) = x_1^p x_2^q$, $p = \mod (i/P)$, and $q = \lfloor i/Q \rfloor$.

The actual vector field is $f(x) := [f_1(x), f_2(x)]^\mathsf{T} = [-x_2, x_1 - (1 - x_1^2)x_2]^\mathsf{T}$. As we can analytically establish that $\mathcal{L}_{\mathfrak{z}_1}(x) = [1, 0] \cdot f(x) = f_1(x)$ and $\mathcal{L}_{\mathfrak{z}_P}(x) = [0, 1] \cdot f(x) = f_2(x)$, by (5.29), we use the approximation $[\widetilde{\mathcal{L}}_{\mathfrak{z}_1}, \widetilde{\mathcal{L}}_{\mathfrak{z}_P}]$ to conversely obtain f.

We first show the results with the modified L_{mod} (5.36) for the general learning. To use the formula in Section 5.6.3, we also need to determine the values of μ, λ, τ_s, and Γ. The discrete form L ($= L_{\text{mod}}$) can be then obtained according to Section 5.6.3. We apply the learned L to identify the system vector fields. Note that $\mathfrak{z}_1(x) = \mathcal{Z}_N(x)\chi_1$ and $\mathfrak{z}_P(x) = \mathcal{Z}_N(x)\chi_P$, where each χ_i for $i \in \{0, 1, \cdots, N-1\}$ is a column unit vector with all components being 0 except for the i-th component, which is 1. To apply L, we have that $\widetilde{\mathcal{L}}_{\mathfrak{z}_1}(x) = \mathcal{Z}_N(x)(L\chi_1)$ and $\widetilde{\mathcal{L}}_{\mathfrak{z}_P}(x) = \mathcal{Z}_N(x)(L\chi_P)$. In other words, we approximate $\widetilde{\mathcal{L}}_{\mathfrak{z}_1}$ and $\widetilde{\mathcal{L}}_{\mathfrak{z}_P}$ (and hence f_1 and f_2) using a linear combination of functions within $\mathcal{Z}_N$.

Test of Parameters of the RTM

We first test how the parameters μ, λ, τ_s and Γ impact the precision of the RTM in terms of the modified L in (5.36). As the impact of λ and τ_s has been shown analytically in Theorem 5.4.3 and 5.5.2, we can set λ to be a very large number while fixing τ_s to any reasonable value. The main focus of this part of the experiment is to see how tuning μ can impact performance under the constraint of Γ.

In the experiments, we set $\lambda = 10^8$ and $\tau_s = 5$, with $N = 3 \times 3$ monomials as in (5.53). We first use μ to evaluate the integral $\mathcal{I}_\mu$ under the sampling frequency 100 Hz (equivalently,

$\Gamma = 500$), and then use (5.36) and (5.29) to learn the generator. Similarly, we perform the experiments for the cases of 50 Hz and 10 Hz. In Fig. 5.1, the relationship between τ_s, μ, and $\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}}$ is illustrated.

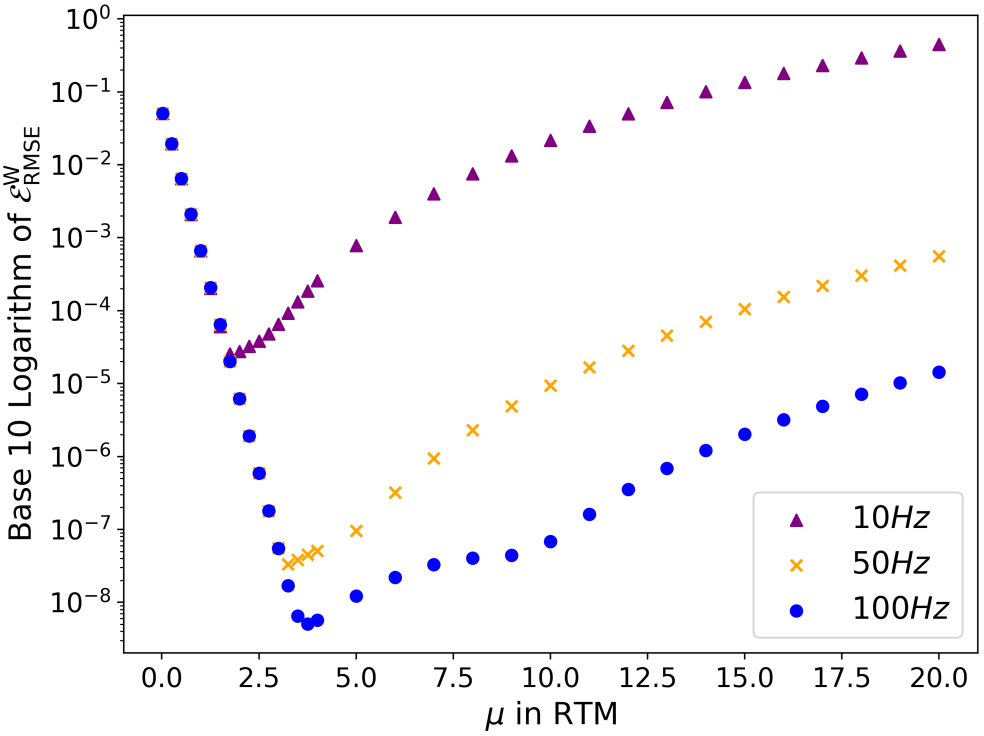


Figure 5.1: RMSE of weights ($\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}}$) using RTM for the reversed Van der Pol oscillator with different sampling frequencies using the modified L_{mod} obtained with (5.36).

Under a sampling rate of 100 Hz, the highest accuracy ($\approx 10^{-8}$) is achieved at $\mu \approx 3.75$. A similar pattern is observed for the cases of 50 Hz and 10 Hz. However, the optimal accuracy decreases as the sampling frequency lowers, as expected. For a fixed tuple $(\tau_s, \lambda, \Gamma)$, it is clear that the accuracy of the numerical integral $\mathcal{I}_\mu$ increases as μ decreases, since the integrand $e^{-\mu s}\mathcal{K}_s\mathfrak{z}_i$ becomes less steep. By observing the form of the inference formula in (5.36), the non-monotonic trend of the learning accuracy with respect to changes in μ can be heuristically attributed to the combined effects of λ, μ, and the errors of the numerical integral.

Remark 5.7.1. *A similar pattern has also been found in other numerical experiments, but we omit presenting them to avoid repetitive expression of the same idea. The optimal tuning of parameters can be roughly determined by the local Lipschitz constant of the system. It is of the authors' interest to pursue further numerical analysis to obtain a reasonable estimate of the tuning under mild assumptions about the system in future work.*

The method has also been applied in real-world scenarios, such as identifying the car-following model [89], particularly in applications such as identifying the car-following model, where the acceleration/deceleration (which directly determines the Lipschitz constant of the model) is known to be bounded. Numerical results provide a range for the choice of μ that leads to acceptable accuracy under other fixed parameters. ◇

As demonstrated above, one still needs to tune the parameter μ and a large τ_s to get desirable approximation accuracy with L_{mod}. We refer to it as 'Modified' in what follows. To tackle these two issues, we thus employ the updated version of the data-driven approximation L_{update} derived in (5.48), as detailed in Algorithm 5, hereafter referred to as 'Updated' in the following comparison. For this specific example, we use the same $\mathcal{Z}_N(x)$ but with $\tau_s = 1$, and the results for a sampling rate of 100 Hz are shown in Fig. 5.2. It is evident that with L_{update}, its performance is less sensitive to the choice of μ, as long as the value falls into a wide range of μ from 0.02 to 5. Moreover, for all the μ values in the range with fewer data samples required ($\tau_s = 1$), we can achieve comparable results as the best one in Fig. 5.1 corresponding to the same sampling frequency but with $\tau_s = 5$. As illustrated in Fig. 5.3, numerical results for other sampling frequencies have a similar trend and the same optimal choice of μ values. Similar behavior is observed for other numerical examples, though these results are omitted here to avoid redundancy. Consequently, we adopt the 'Updated' version to learn the generator with $\tau_s = 1$ in all subsequent numerical experiments.

Comparisons Among All Methods

We compare the RTM with the FDM and KLM. We set $\mu = 2.5, \tau_s = 1$ for RTM, while the other parameters remain the same as in the experiment in Section 5.7.1-1). The weights for these approximations are given by L_{χ_1} and L_{χ_3}, respectively. For better illustration, we report the results for all three methods under sampling rate $\gamma = 50$ Hz in Tables 5.1 and 5.2. The RMSE of weights ($\mathcal{E}^{\text{W}}_{\text{RMSE}}$) and the RMSE of flow ($\mathcal{E}^{\text{F}}_{\text{RMSE}}$) are reported in Table 5.3 for sampling rates of 10, 50, and 100 Hz. Since SINDy typically relies on one single trajectory, we set the sampling period as $50s$ to ensure a comparable number of data samples for a fair comparison. Unless otherwise specified, the same setting is applied to all

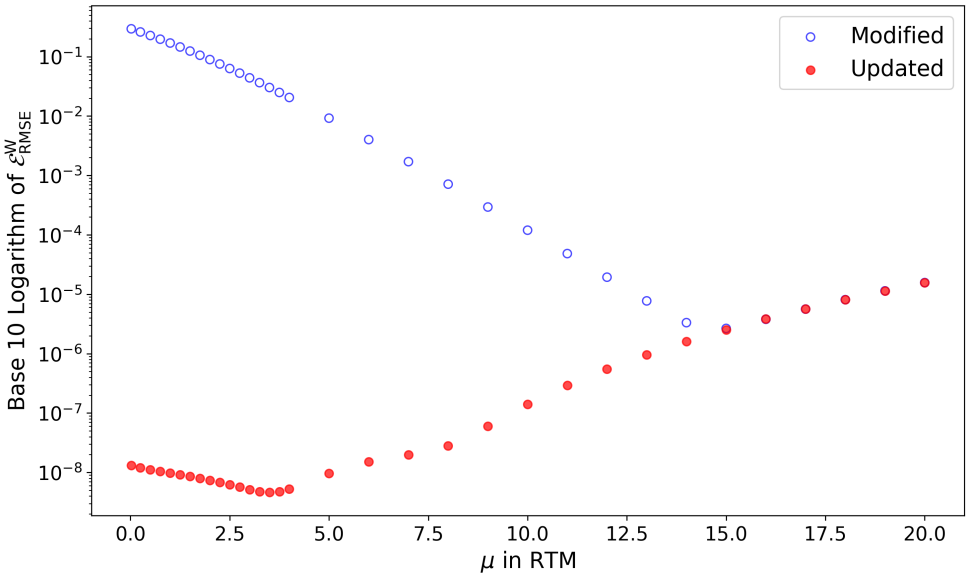


Figure 5.2: The comparison of performances between using L_{mod} and the updated version L_{update}.

subsequent numerical examples that involve SINDy and Weak SINDy (WSINDy). SRTM in general shows the best performance across all sampling frequencies, while RTM shows slightly better performance than SINDy overall.

Table 5.1: Comparisons of the weights in f_1 with the ground truth for the reversed Van der Pol oscillator with $\gamma = 50$

	1	x_1	x_1^2	x_2	x_1x_2	$x_1^2x_2$	x_2^2	$x_1x_2^2$	$x_1^2x_2^2$
Ground Truth	0	0	0	-1	0	0	0	0	0
SINDy	0	0	0	**-1 + 6.72e-9**	0	0	0	0	0
FDM	-4.06e-8	-9.97e-3	1.01e-5	-1 + 1.00e-2	-8.48e-7	-9.91e-3	-3.63e-6	1.33e-4	-1.01e-5
KLM	2.87e-9	-1.93e-8	-1.21e-8	-1 + 1.93e-8	2.46e-8	1.95e-7	-4.85e-8	-1.98e-7	8.98e-8
RTM	**-5.67e-11**	**-1.04e-8**	**7.06e-11**	-1 + -1.67e-8	**-5.52e-10**	**1.17e-8**	**4.44e-10**	**1.84e-9**	**1.80e-9**
SRTM	0	0	0	-1 + -1.67e-8	0	0	0	0	0

If the sparsity is not taken into consideration, the results demonstrate that the proposed resolvent-type method outperforms the other two Koopman-based methods. Particularly, the errors of FDM are at least 100 times higher than those of KLM and RTM at all tested sampling rates. The improvement in accuracy with increasing sampling rates for FDM is also not significant. A potential drawback of using the offline-learned dynamics by FDM is the requirement for more frequent resampling to ensure real-time operational safety.

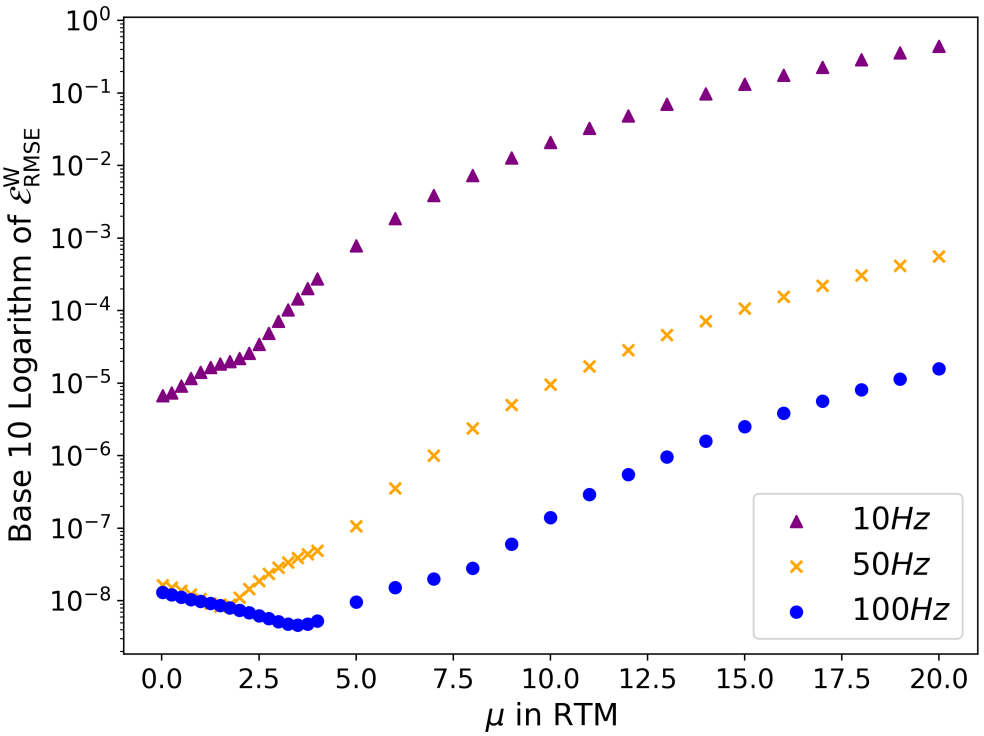


Figure 5.3: RMSE of weights ($\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}}$) using RTM for the reversed Van der Pol oscillator with different sampling frequencies using the updated version L_{update} obtained with (5.49).

On the other hand, KLM demonstrates a medium-level error compared to the others. One potential drawback of KLM is its sensitivity to the choice of dictionary functions, including the number and type of basis. In the preliminary version [93] of this paper, we showed that deliberately choosing different numbers for i and j (e.g. $N = 4 \times 3$) can lead to non-negligible imaginary parts after taking the matrix logarithm of K. A heuristic explanation can be found in Section 5.6.4. The presence of these imaginary parts is significant and cannot be overlooked when aiming to minimize human intervention in identifying unknown systems in practice.

Table 5.2: Comparisons of the weights in f_2 with the ground truth for the reversed Van der Pol oscillator with $\gamma = 50$

	1	x_1	x_1^2	x_2	x_1x_2	$x_1^2x_2$	x_2^2	$x_1x_2^2$	$x_1^2x_2^2$
Ground Truth	0	1	0	-1	0	1	0	0	0
SINDy	0	1 + 3.74e-8	0	**-1 + -2.97e-10**	0	1 + 4.88e-7	0	0	0
FDM	7.54e-6	1 + -4.58e-3	-1.52e-3	-1 + -3.73e-4	1.32e-4	1 + -1.35e-2	5.30e-4	-1.99e-2	1.51e-3
KLM	-1.04e-6	1 + -2.53e-6	1.77e-6	-1 + -1.58e-5	1.04e-5	1 + 8.53e-5	1.39e-6	3.50e-5	-4.81e-7
RTM	**-4.49e-10**	**1 + 6.12e-9**	**2.76e-9**	-1 + -3.28e-8	**9.23e-10**	**1 + 3.58e-8**	**4.17e-9**	**5.49e-8**	**-1.93e-8**
SRTM	0	**1 + 6.12e-9**	0	-1 + -3.28e-8	0	**1 + 3.58e-8**	0	0	0

Table 5.3: Comparisons of RMSE of weights and flow over 100 trajectories for the three polynomial systems

System	M	N	γ	RMSE of weights ($\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}}$)					RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)				
				SINDy	FDM	KLM	RTM	SRTM	SINDy	FDM	KLM	RTM	SRTM
Reversed Van der Pol	10^2	9	10	1.26e-4	3.52e-2	5.53e-4	**3.43e-5**	**3.20e-5**	2.47e-5	1.96e-2	9.51e-5	**2.04e-5**	**1.85e-5**
			50	1.15e-7	7.08e-3	2.22e-5	**1.88e-8**	**1.22e-8**	2.56e-8	4.01e-3	3.65e-6	**7.27e-9**	**5.51e-9**
			100	6.53e-9	3.54e-3	6.41e-6	**6.23e-9**	**5.54e-9**	**1.54e-9**	1.83e-3	1.07e-6	3.37e-9	2.34e-9
Scaled Lorenz-63 (3-dimensional)	10^3	8	10	**2.45e-4**	1.82e-1	6.99e-3	1.74e-3	1.43e-3	**1.05e-3**	2.32e-1	1.12e-2	1.47e-3	1.18e-3
			50	3.92e-7	3.81e-2	2.85e-4	**3.67e-7**	**1.76e-7**	1.43e-6	4.32e-2	4.78e-4	**3.12e-7**	**2.32e-7**
			100	**2.49e-8**	1.92e-2	7.11e-5	**2.49e-8**	**1.52e-8**	8.62e-8	2.31e-2	1.18e-4	**4.42e-8**	**2.48e-8**
Lorenz-96 (6-dimensional)	5^6	64	10	3.03e-1	2.18e-2	3.03e-4	**7.51e-5**	**4.01e-5**	3.31e-1	2.96e-2	3.29e-4	**6.40e-5**	**5.63e-5**
			50	2.74e-1	4.63e-3	1.17e-5	**1.84e-8**	**6.63e-9**	3.60e-1	6.08e-3	1.42e-5	**1.71e-8**	**1.58e-8**
			100	2.89e-1	2.33e-3	2.92e-6	**3.64e-9**	**3.41e-9**	3.74e-1	3.05e-3	3.60e-6	**4.69e-9**	**4.68e-9**

5.7.2 Scaled Lorenz-63 System

We consider the scaled Lorenz-63 system [2] by scaling it with a factor of ϵ, which yields

$$\begin{aligned}\dot{x}_1 &= \sigma(x_2 - \epsilon x_1),\\ \dot{x}_2 &= x_1(\gamma - x_3) - \epsilon x_2,\\ \dot{x}_3 &= x_1 x_2 - \epsilon\beta x_3,\end{aligned}$$

where $\sigma = 10$, $\gamma = 0.28$, $\beta = \frac{8}{3}$, and $\epsilon = 0.1$. With this so-called $\epsilon-$Lorenz system, the system possesses an attractor on $\Omega = (-1, 1)^3$. We choose the dictionary of monomials with total number of $N = P \times Q \times J$ and set $\mathfrak{z}_i(x) = x_1^p x_2^q x_3^j$, where $p = i - PQj - Pq$, $q = \lfloor \frac{i-PQj}{P} \rfloor$, and $j = \lfloor \frac{i}{PQ} \rfloor$. Similar to Section 5.7.1, to identify the vector field, one can apply $\mathcal{L}_{\mathfrak{z}_1}$, $\mathcal{L}_{\mathfrak{z}_P}$, and $\mathcal{L}_{\mathfrak{z}_{P+Q}}$ after learning the generator.

The above scaled system is considered to facilitate easier observation within a smaller region of interest and to maintain a lower growth rate, ensuring that the tuning parameters remain within a reasonable range similar to the previous case study. Specifically, the original system (as studied in [85]) has the vector field $\hat{f}(x) = [\sigma(x_2 - x_1), x_1(\gamma - x_3) - x_2, x_1 x_2 - \beta x_3]$, compared to the scaled version $f(x) = [\sigma(x_2 - \epsilon x_1), x_1(\gamma - x_3) - \epsilon x_2, x_1 x_2 - \epsilon\beta x_3]$.

Remark 5.7.2. *Denoting the solution to the original Lorenz model as $\hat{x}$, it can be verified that $\hat{x}_1(t) = (1/\epsilon)x_1(\epsilon t)$, $\hat{x}_2(t) = (1/\epsilon^2)x_2(\epsilon t)$, and $\hat{x}_3(t) = (1/\epsilon^2)x_3(\epsilon t)$. Let $\widehat{\mathcal{L}}$ be the generator of the original system. One can also verify that $\widehat{\mathcal{L}} = \epsilon\mathcal{L}$. It is worth noting that the scaling with $\epsilon > 0$ does not affect the topological stability.* ◇

In this case study, we set $P = Q = J = 2$ (or $N = 2^3$). We also set $\mu = 2.5$, $\lambda = 1e8$, and $\tau_s = 1$ for RTM. The comparisons for RMSE of weights and flow are summarized in Table 5.3. Fig. 5.4 depicts the comparison of the trajectory for $T_s = 100$ using the approximated dynamics with RTM ($\gamma = 100$ Hz) and ground truth for an initial condition sampled in Ω, while the comparisons for the ones with KLM and FDM are included in Fig. 5.5. It is evident that the flow prediction using the dynamics approximated by RTM is highly accurate, successfully exhibiting the attractor (a long-term behavior) for this chaotic system, while the other two methods struggle to predict the trajectory effectively.

We choose to use the scaled system to facilitate the prediction of long-term errors in the original unscaled chaotic system within a smaller region of interest and over a short dimensionless observation horizon. Specifically, due to the nature of chaotic systems, two trajectories starting very close together will rapidly diverge from each other, resulting in completely different long-term behaviors. The practical implication is that long-term

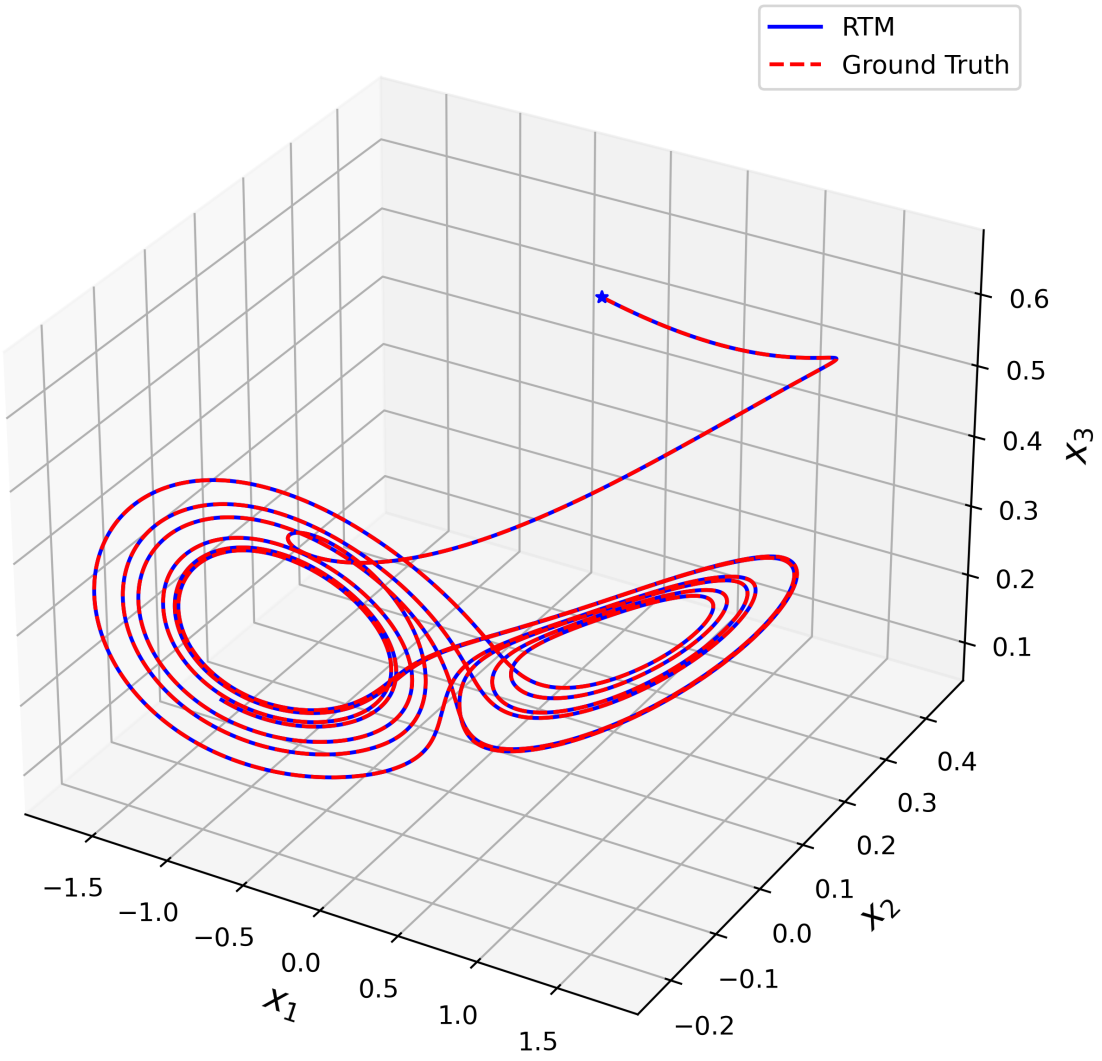


Figure 5.4: Comparison of the trajectory using RTM with the ground truth for the scaled Lorenz-63 system, where the blue star denotes the initial condition.

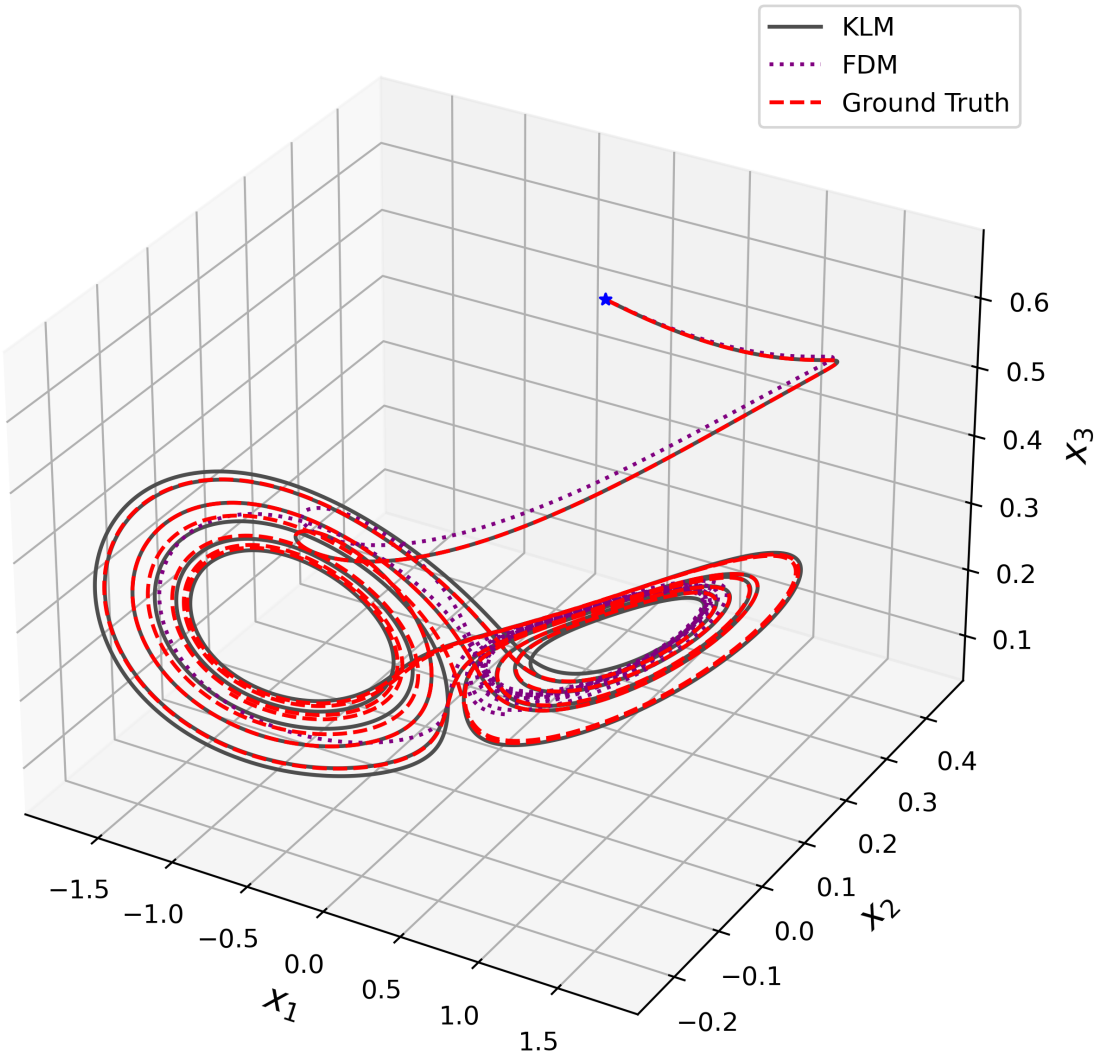


Figure 5.5: Comparison of the trajectories using KLM and FDM with the ground truth for the scaled Lorenz-63 system, where the blue star denotes the initial condition.

prediction becomes impossible in such a system, where small errors are amplified extremely quickly. Such a two-point motion divergence phenomenon can be roughly characterized by the maximal Lyapunov exponent $\varrho > 0$ [123] and quantitatively expressed by $\mathcal{E}(t) \approx \mathcal{E}(0)e^{\varrho t}$, where the process $\{\mathcal{E}(t)\}_{t\geq 0}$ represents the evolution of error w.r.t. any initial condition. Suppose we now consider the original unscaled system (as studied in [85]), and denote $\widehat{\mathcal{E}}(t)$ as the unscaled error. Then, in view of Remark 5.7.2 on the relation between the solutions of the original and scaled systems, one can obtain $|\widehat{\mathcal{E}}(t/\epsilon)| \geq |\mathcal{E}(t)|/\epsilon$. Hence, whatever error is reflected in Fig. 5.4 and 5.5, the error for the unscaled version will be at least $1/\epsilon$ times larger at the extended time scale t/ϵ. For the prediction of a chaotic system, controlling the error within an even smaller tolerance level is necessary, and in this regard, RTM significantly outperforms the other two Koopman-based methods, and shows competitive or better performance with SINDy, particularly at higher sampling frequencies.

5.7.3 Lorenz-96 System

In this case study, we work on a high-dimensional chaotic system, the monoscale Lorenz-96 system[81], given as

$$\dot{x}_j = -x_{j-2}x_{j-1} + x_{j-1}x_{j+1} - x_j + F$$

for all $j \in \{1, 2, \cdots, d\}$. Here, we set $d = 6$ and $F = 0.1$ with $\Omega = (-1, 1)^d$, such that the forward invariance w.r.t. Ω can be guaranteed. Considering that this is a high-dimensional system, we simply set the number of monomials in the dictionary to $N = 2^d$, using a similar indexing method as in the previous case studies.

For RTM, we set $\tau_s = 1$, $\lambda = 1e8$ and set $\mu = 2.5$ as well for all three sampling rates, respectively. The detailed comparisons with the other two methods are listed in Table 5.3.

As shown in the table, for all three polynomial systems (Section 5.7.1 to 5.7.3), compared to KLM and FDM, RTM and SRTM reach better accuracy in terms of the errors for both the weights and the flow. In addition, all three Koopman-based methods show better performance as the sampling rate increases when using the monomial dictionary functions, with FDM exhibiting the slowest rate of improvement. In terms of the performance of SINDy, the results demonstrate that while SINDy may work well for lower-dimensional polynomial systems, Koopman-based methods excel in high-dimensional nonlinear systems like Lorenz-96, providing both superior weight estimation and flow prediction accuracy. This highlights the strength of the Koopman-based approaches for complex dynamical systems. The poor performance of SINDy on the high-dimensional example can be attributed to its reliance on a single trajectory for computation. As the trajectory approaches steady state,

the combination of a large number of basis functions in this high-dimensional system results in a rank-deficient least-squares problem, thereby limiting the method's effectiveness.

For non-polynomial systems, selecting monomial dictionary functions may be ineffective due to their limited expressiveness. To enhance adaptability in characterizing the generator for more general dynamical systems, we will continue to investigate the impact of using non-polynomial dictionary functions for all three methods in the upcoming experiments.

5.7.4 Two-Machine Power System

Consider the two-machine power system [132] modelled by

$$\dot{x}_1 = x_2,\ \dot{x}_2 = -0.5x_2 - (\sin(x_1 + \zeta) - \sin(\zeta)),$$

where $\zeta = \frac{\pi}{3}$. Since the vector field consists of polynomial and periodic terms, using monomial approximations appears to be local and may not be satisfactory. Instead, we utilize the less biased hidden layers of random feature neural networks as the dictionary functions.

Remark 5.7.3 (Related Work)**.** *In the context of Koopman operator learning, the authors of [3] proposed a consistent Koopman autoencoder neural network structure aimed at achieving better approximation performance. Essentially, this autoencoder seeks to find the* $K = \operatorname{argmin}_{A\in\mathbb{C}^{N\times N}} \|Z - XA\|_F$ *by optimally matching the input and output data. However, for deep neural network-based dictionary functions, a closed-form expression cannot be obtained, as they are not linear combinations. Additionally, the training process tends to be computationally expensive.*

Comparatively, the work in [32] demonstrates that shallow tanh *neural networks suffice to approximate functions at comparable or better rates than much deeper ReLU neural networks. The approximation error decreases when the number of hidden neurons increases [32]. Numerous applications have also been explored in solving first-order linear PDEs, highlighting their expressive power.*

In the upcoming experiments, we leverage this expressive feature by using shallow tanh*-activated neural networks as the dictionary functions, achieving the linear combination effect of the test and image functions of the learned operator.* ◇

By choosing tanh as the activation function, the dictionary of observable functions mainly consists of $\tanh(Wx + b)$, where $W \in \mathbb{R}^{\sigma\times d}$ and $b \in \mathbb{R}^{\sigma}$ are the randomly generated weight and bias of the first layer, respectively, and tanh is applied elementwise. To obtain

the expressions of the vector fields, we also need to add $\{x_j\}_{j=1}^d$ to the dictionary (recall that $\mathcal{L}x_j = f_j$). All together, the dictionary $\mathcal{Z}_N(x)$ consists of σ random feature neural networks appended by $\{x_j\}_{j=1}^d$, with a total of $N = \sigma + d$ elements.

We sampled $M = 10^2$ initial points randomly across $\Omega = (-1, 1)^2$ with $\tau_s = 1$. For RTM, we set $\lambda = 1e8$ and $\mu = 2.5$ for all sampling frequencies. As we cannot compare the RMSE of weights in this case, the comparisons of RMSE of flow with three different frequencies are summarized in Table 5.4. Like polynomial systems, RTM outperforms the other two methods in most cases. The results clearly show that RTM consistently outperforms both FDM and KLM across almost all configurations for the two-machine power system, with performance generally improving as the number of basis functions increases and sampling frequency increases. Particularly, as with the tanh neural networks, the learned Koopman matrix K does not preserve the diagonal property, leading to non-negligible imaginary parts after taking the matrix logarithm of K. The RMSE of the imaginary parts of flow using KLM is reported in Table 5.5. Note that even though KLM performs slightly better in the case of $N = 22$, the RMSE of the imaginary parts of the results obtained by this approach is not small, which might result in large oscillations in practice.

The trajectories generated with the approximated dynamics for all three methods are close to the ground truth when $\sigma = 50$ with $\tau = 0.01$. However, when σ increases to 100, both the performances of FDM and KLM deteriorate, especially KLM, as shown in Fig. 5.6. These results validate the statement that when one lacks prior knowledge of the underlying system, which makes it difficult to determine the number and/or types of dictionary functions, both FDM and KLM may struggle to produce accurate results in system identification.

We also present the RMSE of the imaginary parts of flow using KLM for the two-machine power system here. The results are reported in Table 5.5.

5.7.5 Nonpolynomial Vector Fields

Consider a system with nonpolynomial vector fields:

$$\dot{x}_1 = -x_1 + \frac{4x_2}{1 + x_2^2}, \quad \dot{x}_2 = -x_2 - \frac{4x_1}{1 + x_2^2}.$$

For this nonpolynomial system, as shown in [145], it is necessary to include $\{\frac{x_i}{1+x_j^p}\}$ in the dictionary functions for the exact identification of the model. However, this prior knowledge is typically not accessible for most nonlinear systems in practice. To reduce the bias in

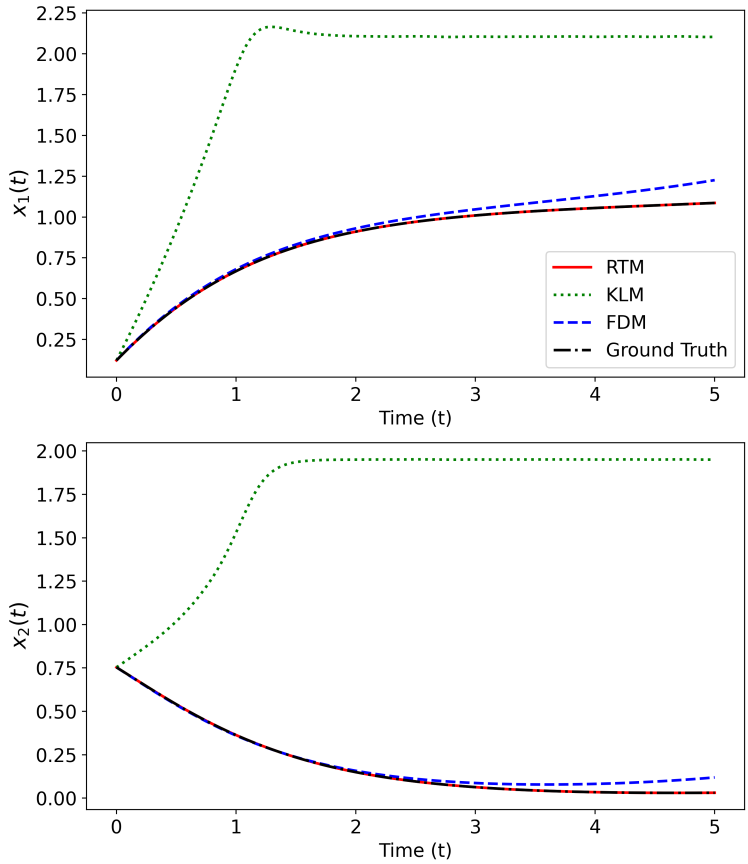


Figure 5.6: Comparisons of the trajectories with the approximated dynamics using three different methods and ground truth for the two-machine power system.

selecting dictionary functions, we continue to use random feature NNs as the dictionary functions. Regarding the training data, $M = 10^2$ initial points are randomly sampled within $\Omega = (-1, 1)^2$. We also set $\tau_s = 1$, $\mu = 3.5$, and $\lambda = 1e8$ for RTM. The comparisons are detailed in Table 5.6, and the RMSE of the imaginary parts of flow using KLM is reported in Table 5.7. It is worth noting that when $\sigma = 50$ and $\tau = 0.1$, KLM fails to return meaningful results, as the trajectory of the approximated dynamics blows up. This is because the matrix approximation of the Koopman operator, K, does not possess the diagonal property, while numerical errors occurring while taking the logarithm lead to worse results, as discussed in Section 5.6.4.

Table 5.4: Comparisons of RMSE of flow over 100 trajectories for the two-machine power system with tanh-activated neural networks

System	N	Sampling Rate (Hz)	RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)		
			FDM	KLM	RTM
Two-machine ($\mu = 4$)	22	10	1.13e-2	**1.00e-3**	1.06e-3
		50	2.56e-3	**1.01e-3**	1.09e-3
		100	1.70e-3	**1.03e-3**	1.05e-3
Two-machine ($\mu = 4$)	52	10	1.10e-2	3.77e-3	**1.47e-4**
		50	4.60e-3	2.32e-2	**1.07e-5**
		100	7.75e-3	1.11e-2	**1.07e-5**
Two-machine ($\mu = 4$)	102	10	1.18e-2	1.16e-2	**1.56e-4**
		50	5.13e-3	8.36e-2	**8.08e-6**
		100	2.12e-2	1.83e-1	**6.64e-6**

To make a more comprehensive comparison with the existing methods. We again choose monomials as the dictionary $\mathcal{Z}_N(x)$ and compare the results with the previous three benchmark methods and a more recent method based on SINDy, WSINDy [96]. Since this non-polynomial system does not have a sparse representation with monomial basis functions, we do not enforce the sparisty for the Koopman-based methods, and thus the results with SRTM are not included. We use the same values for the parameters as the ones in the previous para-

Table 5.5: RMSE of the imaginary parts of flow over 100 trajectories for the two-machine power system using KLM

System	N	Sampling Rate (Hz)	RMSE of imaginary part
Two-machine	22	10	7.80e-1
		50	7.84e-1
		100	7.82e-1
Two-machine	52	10	7.76e-1
		50	7.63e-1
		100	7.63e-1
Two-machine	102	10	7.73e-1
		50	8.11e-1
		100	8.04e-1

Table 5.6: Comparisons of RMSE of flow over 100 trajectories for the nonpolynomial system using tanh-activated neural networks

System	N	Sampling Rate (Hz)	RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)		
			FDM	KLM	RTM
Nonpoly ($\mu = 3.5$)	22	10	7.27e-2	3.08e-3	**2.64e-3**
		50	1.81e-2	2.47e-3	**2.22e-3**
		100	8.90e-3	2.85e-3	**2.52e-3**
Nonpoly ($\mu = 3.5$)	52	10	1.08e-1	-	**6.72e-2**
		50	2.42e-2	3.96e-2	**9.37e-4**
		100	1.74e-2	1.10e-1	**9.19e-4**
Nonpoly ($\mu = 3.5$)	102	10	1.01e-1	1.43	**5.48e-3**
		50	2.64e-2	9.05e-2	**9.33e-5**
		100	1.68e-2	2.12e-1	**6.18e-4**

Table 5.7: RMSE of the imaginary parts of flow over 100 trajectories for the nonpolynomial system using tanh-activated neural networks with KLM

System	N	Sampling Rate (Hz)	RMSE of imaginary part
Nonpoly	22	10	7.78e-1
		50	7.78e-1
		100	7.80e-1
Nonpoly	52	10	1.78e36
		50	7.90e-1
		100	8.32e-1
Nonpoly	102	10	2.60e39
		50	7.40e-1
		100	7.75e-1

graph. In order to mitigate the numerical instability associated with high-order polynomial regression, which can lead to computational blow-up, we employ two conservative monomial sets that differ from those used in Section 5.7.1. The first set contains $N = 10$ monomials up to order 3: $\{1, x_1, x_1^2, x_1^3, x_2, x_1x_2, x_1^2x_2, x_2^2, x_1x_2^2, x_2^3\}$, while the second set includes $N = 15$ monomials up to order 4: $\{1, x_1, x_1^2, x_1^3, x_1^4, x_2, x_1x_2, x_1^2x_2, x_1^3x_2, x_2^2, x_1x_2^2, x_1^2x_2^2, x_2^3, x_1x_2^3, x_2^4\}$. The results are summarized in Table 5.8. The RMSE of the imaginary parts of flow using KLM is reported in Table 5.9. It is evident that RTM overall outperforms the other four

methods in terms of RMSE of flow. Even so, KLM performs slightly better than RTM for $N = 10$ and $\tau = 0.01$, the RMSE of the imaginary parts of flow is not small, which may lead to large oscillations in practice.

Table 5.8: Comparisons of RMSE of flow over 100 trajectories for the nonpolynomial system using monomials

System	N	Sampling Rate (Hz)	RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)				
			SINDy	WSINDy	FDM	KLM	RTM
Nonpoly ($\mu = 3.5$)	10	10	1.22e-1	3.14e-2	1.12e-1	1.41e-2	**1.39e-2**
		50	2.76e-1	1.85e-2	3.00e-2	1.41e-2	**1.39e-2**
		100	3.28e-1	2.05e-2	2.01e-2	**1.43e-2**	1.57e-2
Nonpoly ($\mu = 3.5$)	15	10	1.05e-1	4.73e-2	1.08e-1	1.37e-2	**1.36e-2**
		50	4.45e-1	6.78e-2	3.10e-2	1.49e-2	**1.46e-2**
		100	4.90e-1	1.35e-1	2.02e-2	1.44e-2	**1.36e-2**

Table 5.9: RMSE of the imaginary parts of flow over 100 trajectories for the nonpolynomial system using monomials with KLM

System	N	Sampling Rate (Hz)	RMSE of imaginary part
Nonpoly	10	10	7.86e-1
		50	7.77e-1
		100	7.77e-1
Nonpoly	15	10	7.43e-1
		50	7.93e-1
		100	7.43e-1

5.7.6 7D Biochemical System

In this subsection, we consider a 7-dimensional biochemical system, which is a model of yeast glycolytic oscillations [120]. This model nowadays is a standard benchmark for system identification and model prediction tasks, and it has been studied in [19, Appendix B]. Here, we modify the $\dot{S}_1$ term a bit to allow the invariance of the state space to satisfy the

assumptions within this Koopman framework:

$$
\begin{aligned}
\dot{S}_1 &= k_{\text{ex}}\,(G_{\text{ex}} - S_1) \;-\; v_1,\\
\dot{S}_2 &= 2\,v_1 \;-\; k_2 S_2(N - S_5) \;-\; k_6 S_2 S_5,\\
\dot{S}_3 &= k_2 S_2(N - S_5) \;-\; k_3 S_3(A - S_6),\\
\dot{S}_4 &= k_3 S_3(A - S_6) \;-\; k_4 S_4 S_5 \;-\; \kappa\,(S_4 - S_7),\\
\dot{S}_5 &= k_2 S_2(N - S_5) \;-\; k_4 S_4 S_5 \;-\; k_6 S_2 S_5,\\
\dot{S}_6 &= -2\,v_1 \;+\; 2\,k_2 S_3(A - S_6) \;-\; k_5 S_6,\\
\dot{S}_7 &= \psi\,\kappa\,(S_4 - S_7) \;-\; k\,S_7,\\
\text{with}\quad v_1 &= \frac{k_1\,S_1\,S_6}{1 + \left(S_6/K_1\right)^q}.
\end{aligned}
\tag{5.54}
$$

The values of the parameters are reported in Table 5.10.

Table 5.10: Yeast glycolysis model parameters for the modified system

Parameter	Value	Parameter	Value	Parameter	Value
k_{ex}	0.5	G_{ex}	0.5	k_1	100
k_2	6	k_3	16	k_4	100
k_5	1.28	k_6	12	k	1.8
κ	13	q	4	K_1	0.52
ψ	0.1	N	1.0	A	4.0

We again compare RTM with the four benchmark methods. In this case, we use monomials up to order 2, resulting in $N = 36$ dictionary functions. The list of monomials can be found in [19, Appendix B]. For this example, we sample $M = 7^7$ initial points randomly across $\Omega = (0, 0.5)^7$. The parameter μ is set to 2, and $\lambda = 1e8$ for RTM. The results are summarized in Table 5.11. The RMSE of the imaginary parts of flow using KLM is reported in Table 5.12. Similar to the previous nonpolynomial system, RTM overall outperforms the other four methods in terms of RMSE of flow. KLM performs slightly better than RTM for $\tau = 0.1$, but the RMSE of the imaginary parts of flow is large, which may lead to large oscillations in practice. Furthermore, SINDy and WSINDy fail to produce meaningful regression results for this high-dimensional system. The learned weights from SINDy lead to numerical instability and computational blow-up, a finding that aligns with the limitations reported in the original literature.

Table 5.11: Comparisons of RMSE of flow over 100 trajectories for the 7-dimensional biochemical system using monomials

System	N	Sampling Rate (Hz)	RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)				
			SINDy	WSINDy	FDM	KLM	RTM
Biochemical ($\mu = 2.5$)	36	10	-	-	8.69e-2	**2.63e-2**	3.35e-2
		50	-	-	3.78e-2	3.31e-2	**2.13e-2**
		100	1.16e4	2.02e2	3.35e-2	4.18e-2	**2.33e-2**

Table 5.12: RMSE of the imaginary parts of flow over 100 trajectories for the nonpolynomial system using monomials with KLM

System	N	Sampling Rate (Hz)	RMSE of imaginary part
Biochemical	36	10	9.39e2
		50	7.58e-1
		100	7.58e-1

5.7.7 System at the Bifurcation Point

In this example, we demonstrate that the proposed method can also be applied to the cases where $\rho(\mathcal{L})$ is empty over $\mathcal{C}^1(\Omega)$. We first consider the following scalar system that undergoes a pitchfork bifurcation [66]:

$$\dot{x} = \alpha x - x^3, \quad x \in \mathbb{R}^1, \ \alpha \in \mathbb{R}^1.$$

Apparently, the origin is an equilibrium point for all α, and it is stable for $\alpha < 0$ and unstable for $\alpha > 0$. When $\alpha > 0$, we have two extra stable equilibria branching from the origin i.e., $x_{1,2} = \pm\sqrt{\alpha}$. At the bifurcation point $\alpha = 0$, the origin is a non-hyperbolic equilibrium.

We then consider the system at the bifurcation point $\alpha = 0$:

$$\dot{x} = -x^3, \quad x(0) = x_0. \tag{5.55}$$

Then, $x(t) = \frac{x_0}{\sqrt{1+2x_0^2 t}}$. For all $h \in \mathcal{C}^1(\Omega)$, $\mathcal{L}h(x) = -h'(x) \cdot x^3$. Specifically, $\mathcal{L}h(0) = 0$. Now we try to find the resolvent set and resolvent operator.

Suppose $\lambda \in \rho(\mathcal{L})$, then $\lambda \,\mathrm{Id} - \mathcal{L}$ is invertible and for any $h \in \mathcal{C}^1(\Omega)$, $\lambda u + x^3 u'(x) = h(x)$ has a solution. Solving this, one has $u(x) = \exp\left(\frac{\lambda}{2x^2}\right)\left(\int_\Omega \frac{h(x)}{x^3} \exp\left(\frac{\lambda}{2x^2}\right) dx\right)$. However, it is

only valid on $\Omega \setminus \{0\}$, with a singular value at 0. Recall the discussion in Remark 5.4.10, in this case, for $D = \mathcal{C}^1(\Omega)$ and $H = \mathcal{C}(\Omega)$, the resolvent set $\rho(\mathcal{L}) = \emptyset$. We can also verify that $\mathcal{L}$ is not closed on D, whence the pseudo-resolvent $\mathcal{R}(\lambda) \neq (\lambda \operatorname{Id} - \mathcal{L})^{-1}$.

However, for $D = \mathcal{C}^1(\Omega \setminus \{0\})$ and $H = \mathcal{C}(\Omega) \setminus \{0\}$, the resolvent set is non-empty. One can also consider $D = \{h \in \mathcal{C}(\Omega) : h \in \mathcal{C}^1(\Omega \setminus \{0\}), \mathcal{L}h \in \mathcal{C}(\Omega), \lim_{x \to 0} \mathcal{L}h(x) = 0\}$. Then $\lambda \operatorname{Id} - \mathcal{L}$ is injective on this D, $\overline{\operatorname{Range}(\lambda \operatorname{Id} - \mathcal{L})} = \mathcal{C}(\Omega)$ (in fact, $\operatorname{Range}(\lambda \operatorname{Id} - \mathcal{L}) = \mathcal{C}(\Omega)$, which implies surjectivity), and $\ker(\mathcal{R}(\lambda)) = \{0\}$. The closedness of $\mathcal{L}$ also holds on this D. Therefore, $\mathcal{R}(\lambda) = (\lambda \operatorname{Id} - \mathcal{L})^{-1}$, meaning that the pseudo equals the true resolvent operator.

For the purpose of system identification, we generally do not know and therefore do not require knowledge of the singular point. Consequently, we may not find a function space in which $\mathcal{L}$ has the desired properties as stated above (e.g., $D = \{h \in \mathcal{C}(\Omega) : h \in \mathcal{C}^1(\Omega \setminus \{0\}), \mathcal{L}h \in \mathcal{C}(\Omega), \lim_{x \to 0} \mathcal{L}h(x) = 0\}$). Nonetheless, we always have $\lambda^2 \mathcal{R}(\lambda) - \lambda \operatorname{Id} \rightharpoonup \mathcal{L}$ on regular function space.

We then use monomials up to order 4 as the dictionary functions, i.e., $\{1, x, x^2, x^3, x^4\}$, and $M = 10$ initial points randomly sampled within $\Omega = (-1, 1)$. We set $\tau_s = 1$, $\mu = 0.02$ and $\lambda = 1e8$ for RTM. The comparisons with SINDy, WSINDy, FDM, and KLM are presented in Table 5.13. It is evident that RTM outperforms the other two Koopman-based methods and reaches comparable performance as SINDy.

Table 5.13: Comparisons of RMSE of weights and flow over 100 trajectories for the system at the bifurcation point

System	M	N	γ	RMSE of weights ($\mathcal{E}^{\mathrm{W}}_{\mathrm{RMSE}}$)				RMSE of flow ($\mathcal{E}^{\mathrm{F}}_{\mathrm{RMSE}}$)			
				SINDy	FDM	KLM	RTM	SINDy	FDM	KLM	RTM
1D Cubic	10	5	10	**3.98e-5**	5.94e-2	1.61e-3	1.05e-4	5.18e-6	5.11e-3	1.23e-4	**3.18e-6**
			50	1.09e-7	1.36e-2	7.37e-5	**7.78e-8**	1.41e-8	1.10e-3	5.70e-6	**6.89e-9**
			100	**7.31e-9**	6.92e-3	1.90e-5	6.03e-8	**9.43e-10**	5.53e-4	1.52e-6	6.40e-9

Remark 5.7.4. *For bifurcation systems at a bifurcation point (e.g., pitchfork, Hopf, etc.), the system does not have an exponential evolution rate. Therefore, the solution to the resolvent equation $\lambda u - \mathcal{L}u = h$ may exhibit a singular point, as shown in this example. However, we do not aim to summarize whether such regularity generally applies to systems with bifurcations at the bifurcation point. Our discussion is not concerned with locating the bifurcation point; rather, it focuses on whether the resolvent set exists. Proposition 5.4.9 extends the previous result where $\rho(\mathcal{L}) = \emptyset$, and the system identification/generator learning procedure remains unchanged.*

5.7.8 Prediction of Region of Attraction

In this subsection, we demonstrate that both the dataset and dictionary functions can be reused to predict the ROA $\mathcal{D}$ of an equilibrium point x_{eq} for the system[5]. For simplicity, we can assume that the system has an equilibrium point at the origin.

The methodology involves using the learned generator to solve a Zubov's equation of the form

$$\mathcal{L}u(x) = -\alpha|x - x_{\text{eq}}|^2(1 - u(x)), \quad \alpha > 0. \tag{5.56}$$

The solution should have the following properties: 1) $0 < u(x) < 1$ for all $x \in \mathcal{D} \setminus \{x_{\text{eq}}\}$, and 2) $u(x) \to 1$ as $x \to y$ for any $y \in \partial\mathcal{D}$. Therefore, the set $\{x \in \mathbb{R}^n : u(x) < 1\}$ can readily represent the ROA. Moreover, for any approximator $\widetilde{u}$ of u such that $|\widetilde{u} - u|_\infty \leq \varepsilon$, one can show that the set $\{x : \widetilde{u}(x) \leq 1 - \varepsilon\}$ is a tight inner approximation of the ROA.

Remark 5.7.5. *Note that the ROA can also be characterized by the function domain D of a maximal Lyapunov function v if the following conditions hold: 1) $v(x_{\text{eq}}) = 0$ and $v(x) > 0$ for all $x \in D \setminus \{x_{\text{eq}}\}$; 2) the derivative of v along solutions of (5.1) is well-defined for all $x \in D$ and satisfies the Lyapunov equation*

$$\mathcal{L}v(x) = -|x - x_{\text{eq}}|^2; \tag{5.57}$$

and 3) $v(x) \to \infty$ as $x \to \partial D$ or $|x| \to \infty$.

It can be verified that $u(x) = 1 - \exp(-\alpha v(x))$ for all $x \in \mathcal{D}(\mathcal{A})$, and $u(x) = 1$ elsewhere, is the unique bounded viscosity solution that satisfies the required properties [77]. We prefer to use u instead of v for characterizing the ROA due to its global boundedness, which allows the extension of the function domain to the entire state space or any desired set where computations take place. Additional features of the above PDEs can be found in [20, 77]. $\diamond$

To illustrate the accuracy of the learned generator, as well as the effectiveness of using it to solve Eq. (5.56), we revisit the reversed Van der Pol oscillator system. The difference is that we choose the dictionary $\mathcal{Z}_N(x)$ the same as in Sections 5.7.4 and 5.7.5, which consists of the σ-dimensional random feature neural network $\tanh(Wx + b)$ appended with $\{x_1, x_2\}$. In the experiment, we set $\sigma = 100$, $\mu = 10$, and $M = 100^2$.

With the above settings, we can achieve a good approximation of the dynamics with the RMSE of flow $\mathcal{E}^{\text{F}}_{\text{RMSE}} = 1.08e{-}5$. We find an approximate equilibrium point at $[6.81e{-}8, -1.25e{-}8]$, which is sufficiently close to the origin with negligible error. In addition, we

[5]Supposing that x_{eq} is asymptotically stable, the ROA of x_{eq} is a set defined as $\mathcal{D} := \{x \in \Omega : \lim_{t\to\infty} |\phi(t, x) - x_{\text{eq}}| = 0\}$.

observe that the average approximation error of $\mathcal{L}\tanh(Wx+b)$ is $1.66e{-}4$, indicating an overall good performance of the least-squares regression L.

We aim to find a single-hidden-layer feedforward neural network of the form $u(x;\theta) = \mathcal{Z}_N(x)\theta$ that approximates the unique bounded viscosity solution of $\mathcal{L}u(x) = -\alpha|x|^2(1-u(x)$ with $\alpha = 0.1$ and the boundary condition $V(0) = 0$. Due to the fact that $\nabla u(x;\theta)\cdot\widetilde{f}(x) \approx \mathcal{L}u(x;\theta) \approx \mathcal{Z}_N(x)L\theta$ for all $u(x;\theta) = \mathcal{Z}_N(x)\theta$, where $\widetilde{f}(x)$ is the approximated vector field using the learned generator, it is equivalent to finding the weights θ that approximately satisfy

$$\mathcal{Z}_N(x)L\theta = -\alpha|x|^2(1-u(x;\theta)) \tag{5.58}$$

or

$$\nabla u(x;\theta)\cdot\widetilde{f}(x) = -\alpha|x|^2(1-u(x;\theta)), \tag{5.59}$$

ensuring that $u(x;\theta)$ is bounded and a viscosity solution at the same time.

It is worth noting that, without the boundary condition and the constraints on its viscosity property, the problem can be reduced to finding the θ that minimizes the least-squares error (residual loss) between the l.h.s. and r.h.s. of Eq. (5.58) (or (5.59)) at those sampled initial conditions. However, such a solution may not possess the correct physical meaning, with a simple counterexample being $u(x;\theta) \approx 1$ for all x.

Inspired by recent research on physics-informed neural networks, it is also necessary to introduce and minimize additional loss terms, beyond the residual loss, that encompass the PDE and any supplementary information pertinent to the problem [76, 77, 148]. In this case, we specifically need to consider loss terms to match the condition $u(0;\theta) = 0$ and valued on $\partial\Omega$. Clearly, finding θ that minimizes the weighted sum of these loss terms does not have a closed-form solution. That said, an ELM, which seeks solutions of the form $\mathcal{Z}_N(x)\theta$ for PDEs that are linear in both u and ∇u, transforms the optimization into a linear least-squares problem that can be efficiently solved, exhibiting fast learning speed and strong generalization performance [148].

Based on the above feature, and for the purpose of obtaining a valid physics-informed approximation of $u(x)$ with efficient computation, we can leverage the well-tuned toolbox LyZNet [76] to implement the physics-informed ELM algorithm. The learned Lyapunov function and the corresponding region of attraction estimate can be found in Fig. 5.7, where the blue curve represents the region of attraction estimate, which is sufficiently close to the true region of attraction. It is worth noting that the Lyapunov function and the corresponding ROA presented here are computed with the identified system and thus are valid for the identified system, which can be verified using SMT solvers. The natural question is how we can compute a valid Lyapunov function and get the responding ROA estimate for the unknown system. We will tackle this issue in the next chapter.

Remark 5.7.6. *As investigated in [148], one can expect good performance from using random feature neural networks to solve transport-related PDEs that are linear in u and ∇u. The key lies in the ability of using weighted $\tanh(Wx+b)$ to approximate $u(x)$, $\mathcal{L}u(x)$, and $f(x)$, as well as the accuracy of the learned L that guarantees $\nabla u(x;\theta)\cdot\widetilde{f}(x)\approx\mathcal{L}u(x;\theta)\approx\mathcal{Z}_N(x)L\theta$.*

Other dictionary functions may have the above features. However, the choice may depend on side information and the solution space properties, which is not conducive to data-driven approximation for unknown systems. ◇

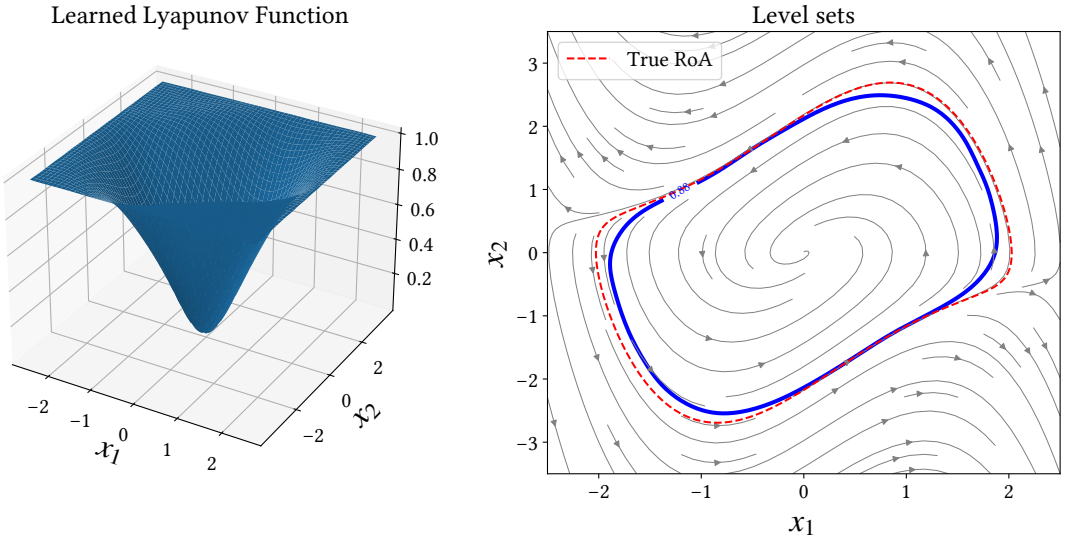


Figure 5.7: Learned Lyapunov function and the corresponding region of attraction estimate using the random feature neural network dictionary functions for the identified reversed Van der Pol oscillator.

5.8 Summary

In this chapter, we propose a novel resolvent-type Koopman operator-based learning framework for the generator of unknown systems. The proposed method demonstrates both theoretical and numerical improvements over the Koopman logarithm method [85] and the finite difference method [13], offering greater adaptability in handling low observation rate

scenarios and utilizing less biased random feature neural networks as dictionary functions. Specifically, in cases where the logarithm of the Koopman operator is invalid for representing the generator, we draw upon the rich literature to propose data-driven approximations based on Yosida approximation and the properties of the resolvent operator. The analytical convergence rate and a modified data-driven learning algorithm are provided to assist users in tuning the parameters.

Chapter 6

Learning Koopman-Based Stability Certificates for Unknown Nonlinear Systems

This chapter is based on work that has been accepted for publication in the Proceedings of the 2025 IEEE 64th Conference on Decision and Control (CDC). A preprint version is available in [149].

6.1 Introduction

Koopman operators also offer a powerful tool for constructing Lyapunov functions and stability analysis [86, 141]. In [86], the authors demonstrated that a set of Lyapunov functions for nonlinear systems with global stability can be constructed from the eigenfunctions of the Koopman operator. Building on this result, and incorporating the autoencoder structure [3] in Koopman operator learning, the authors of [33] introduced an algorithm to identify Koopman eigenfunctions that parameterize a set of Lyapunov function candidates. More recently, [131] used the spectrum of the Koopman operator to directly identify the stability boundary of nonlinear systems. However, methods that use Koopman eigenfunctions to build Lyapunov functions have two main weaknesses. First, the eigenfunctions themselves are only approximations because they depend on how well the Koopman operator is learned within a limited set of functions. This can lead to errors and no formal guarantees of correctness. Second, even when many valid eigenfunctions exist, one must manually choose

which one (or combination) gives the "best" Lyapunov function—usually the one that yields the largest ROA. Since any eigenfunction with a negative real part of its eigenvalue technically works, and weighted sums or products of them can also qualify, finding the optimal candidate becomes cumbersome and uncertain. As a result, all the aforementioned stability verification methods face the limitation of conservative estimation of the ROA. In contrast, [91] proposed a modified Zubov-Koopman operator approach to approximate a maximal Lyapunov function, which is related to the solution of stability-related PDEs (the Lyapunov's or Zubov's equation [77]), where the non-trivial domain corresponds to the ROA. Regardless, all existing Koopman-based constructions of Lyapunov functions cannot be easily integrated into the system identification structure, and therefore lack the ability to be formally verified using the predicted system vector field, leading to a loss of the true predictability of the certifiable region of attraction.

In this paper, we propose an integrated algorithmic framework to simultaneously learn the vector field and Lyapunov functions with formal guarantees for unknown nonlinear systems, thereby achieving full predictability of stability. Our methodology leverages the strengths of the Koopman generator learning framework and utilizes the learned generator to approximately solve stability-related PDEs, with the aim of enhancing the formally verified ROA. In light of the recent development of PINN solutions for PDEs, the proposed method in this paper is not merely a straightforward combination of Koopman generator learning and PDE solving using a parameterized ansatz to match the data and the equation. The input of human knowledge on the physics-informed conditions and the corresponding design of the loss function to train the solution are essential components.

It is worth noting that the technique proposed in [91] can learn a near-maximal Lyapunov candidate for unknown systems, but requires a computationally intensive deep neural network smoothing process. Moreover, the Lyapunov derivative of the candidate cannot be verified due to the lack of information about the system's vector field. On the other hand, a non-Koopman neural system learning and Lyapunov construction framework [150] can provide formal guarantees to ensure closed-loop stability. However, this approach does not offer insights into the quality of the resulting ROA. We address these shortcomings and, to the best of the authors' knowledge, this is the first work to develop a systematic approach for finding formally verified Lyapunov functions using the Koopman generator.

The detailed contributions of this chapter are as follows.

1. We propose a streamlined framework to simultaneously learn the vector field and Lyapunov functions by reusing the same observable test functions and a subset of the data samples.

2. Beyond formulating the least squares problem, we provide additional conditions for solving the stability-related PDEs using the Koopman generator.

3. We demonstrate that the learned Lyapunov function is valid for unknown systems, with formal guarantees achieved through verification using SMT solvers.

It is remarkable that, unlike the Koopman eigenfunction-based approaches, the proposed method avoids guessing or tuning over combinations of the Koopman eigenfunctions for learning the (maximal) Lyapunov functions. It directly solves the stability-related PDEs, ensuring the Lyapunov conditions are satisfied by construction. This approach is more general, as it does not rely on specially chosen observable functions or handcrafted dictionaries, which are sometimes needed for eigenfunction methods. As a result, it can handle a broader range of nonlinear systems, including those with unknown dynamics, while still achieving comparable or better performance without heavy customization or manual parameter tuning.

6.2 Preliminaries

In this chapter, we also consider a continuous-time nonlinear dynamical system of the following form:

$$\dot{x} = f(x), \quad x(0) = x_0, \tag{6.1}$$

and the vector field $f : \Omega \to \mathbb{R}^n$ is assumed to be locally Lipschitz continuous but unknown, where $\Omega \subseteq \mathbb{R}^n$ is a pre-compact state space. We assume that the origin, $x = 0$, is a locally asymptotically equilibrium point of (6.1). For the rest, we use the same notations and assumptions as those in the previous chapter.

Given the definition of the infinitesimal generator of the Koopman operator $\mathcal{L}$ in Definition 5.3.3, we can compute the Lyapunov functions in Theorem 1.3.1 by solving the following PDE:

$$\mathcal{L}V(x) = -\eta(x), \tag{6.2}$$

which we refer to as Lyapunov's equation. Here η is a positive definite function over Ω with respect to the origin. However, due to the unbounded nature of the Lyapunov function, it remains challenging to approximate the true domain of attraction using typical data-driven learning methods, given the compactness of the sampling subspace. The following theorem states that the domain of attraction can be characterized by a transformed maximal Lyapunov function with a bounded range from 0 to 1, where the 1-sublevel set represents the

domain of attraction. We recap Theorem 2.2.5 here from a differential operator perspective as follows.

Theorem 6.2.1 (Zubov's Theorem [152]). *Let $D \subset \mathbb{R}^n$ be an open set containing the origin. Then $D = \mathcal{D}$ if and only if there exist two continuous functions $W : D \to \mathbb{R}$ and $\eta : D \to \mathbb{R}$ such that the following conditions hold: 1) $0 < W(x) < 1$ for all $x \in D \setminus \{0\}$ and $W(0) = 0$; 2) η is positive definite on D with respect to 0; 3) for any sufficiently small $c_3 > 0$, there exist $c_1, c_2 > 0$ such that $|x| \geq c_3$ implies $W(x) > c_1$ and $\eta(x) > c_2$; 4) $W(x) \to 1$ as $x \to y$ for any $y \in \partial D$; 5) W and η satisfy*

$$\mathcal{L}W(x) + \eta(x)(1 - W(x)) = 0. \tag{6.3}$$

We refer to (6.3) as Zubov's equation in the following context.

6.3 Recap of System Identification with the Koopman Generator and Problem Formulation

Following the derivation of the Koopman generator in Section 5.3.2, based on (5.7), with $x := [x_1, x_2, \ldots, x_n]^\intercal \in \mathbb{R}^n$, we have $\mathcal{L}x_i = f_i(x)$, for $i = 1, 2, \ldots n$, where f_i is defined as $\dot{x}_i = f_i(x)$, a component of f. As a result of Theorem 5.5.2, when $x_1, x_2, \ldots, x_n$ are included in the dictionary of the observable functions, the vector field can be readily approximated:

$$\hat{f}_i(x) = \mathcal{L}_{\lambda,\tau_s} x_i \approx \mathcal{L}x_i = f_i(x), \quad \forall i = 1, 2 \cdots, n. \tag{6.4}$$

The idea of this chapter is to identify the Koopman generator and then solve the stability-related PDEs outlined above, with the goal of enlarging the estimation of the ROA. In previous work, [77, 148] demonstrated excellent performance in learning accuracy. However, when solving PDEs with learned generators for unknown systems, a simple data-fitting strategy by minimizing the residual is typically insufficient, as additional physical information—such as boundary conditions—must also be incorporated. In the following section, we first recap the resolvent-type method for learning the Koopman generator in [94]. In this chapter, we focus on developing the physics-informed Koopman-generator-based construction of the Lyapunov function to address the technical concerns mentioned above.

6.4 Stability Certificates for the Unknown Systems

In this section, we show that the Koopman generator can be used not only to approximate the vector field but also to establish stability certificates for unknown systems, i.e., constructing Lyapunov functions by solving linear PDEs. Furthermore, we show that the derived Lyapunov functions provide stability guarantees for the unknown systems.

We assume that the origin is a stable equilibrium point of (5.1). Then (5.1) can be rewritten as $\dot{x} = Ax + g(x)$, where $g(x) = f(x) - Ax$ satisfies $\lim_{x\to 0} \frac{\|g(x)\|}{\|x\|} = 0$. If A is known, a local quadratic Lyapunov function $V_P(x) = x^\mathsf{T} P x$ can be computed by solving $PA + A^\mathsf{T} P = -Q$, where Q is any symmetric positive definite matrix. It can be easily shown that there exists a sufficiently small $c > 0$ such that $\mathcal{O} = \{x \in \Omega | V_P(x) \leq c\}$, where the linearization dominates, that is an ROA [77, Section 5.1]. We make the following assumption.

Assumption 6.4.1. *Let the unknown system be identified using (6.4) and denote the linearization of the approximated vector field about the origin as $\dot{x} = \hat{A}x$. With $\hat{A}$ being Hurwitz (which can be checked numerically), let P solve the Lyapunov's equation $P\hat{A} + \hat{A}^\mathsf{T} P = -Q$ for some positive definite Q. We assume that identification is sufficiently accurate such that $\mathcal{O} = \{x \in \Omega | V_P(x) \leq c\}$ is an ROA for (5.1) for some $c > 0$.*

Note that this assumption can typically be satisfied in most cases by collecting a sufficient amount of data around the origin and ensuring $f(0) = \hat{f}(0)$ in practice. Now, consider an unsampled state z, and y represents the nearest known sample for which $f(y)$ and $\hat{f}(y)$ are accessible. The following theorem establishes the stability conditions for the unknown systems.

Proposition 6.4.2. *Suppose that Assumption 6.4.1 holds. Let $V : \Omega \to \mathbb{R}$ be a candidate Lyapunov function for $\dot{x} = \hat{f}(x)$, with $\hat{f}(0) = 0$, and let K_f and $K_{\hat{f}}$ be the Lipschitz constants of f and $\hat{f}$ respectively, on Ω. Let $\mathcal{Y} \subset \Omega$ be a finite set of samples. Let δ, ν, and α be positive constants such that 1) for every $z \in \Omega$, there exists $y \in \mathcal{Y}$ such that $\|z - y\| < \delta$, 2) $\sup_{x\in\Omega} \|\nabla V(x)\| \leq \nu$, and 3) $\max_{y\in\mathcal{Y}} \|f(y) - \hat{f}(y)\| = \alpha$. Let β be such that $((K_f + K_{\hat{f}})\delta + \alpha)\nu < \beta$ and*

$$\nabla V(x) \cdot \hat{f}(x) \leq -\beta \tag{6.5}$$

holds on $\Omega_{c_1,c_2} = \{x \in \Omega : c_1 \leq V(x) \leq c_2\}$, for some $0 < c_1 < c_2$, such that $\Omega_{c_1} = \{x \in \Omega : V(x) \leq c_1\} \subset \mathcal{O}$, then $\Omega_{c_2} = \{x \in \Omega : V(x) \leq c_2\}$ is an ROA for the unknown system (6.1), provided that Ω_{c_2} does not intersect with the boundary of Ω.

Proof. For any $x \in \Omega$ and $y \in \mathcal{Y}$, we have

$$\begin{aligned} &\|f(x) - \hat{f}(x)\| \\ \leq &\|f(x) - f(y)\| + \|f(y) - \hat{f}(y)\| + \|\hat{f}(y) - \hat{f}(x)\| \\ \leq &K_f\delta + \alpha + K_{\hat{f}}\delta. \end{aligned} \tag{6.6}$$

It follows that

$$\begin{aligned} &\nabla V(x) \cdot f(x) - \nabla V(x) \cdot \hat{f}(x) \\ \leq &\|\nabla V\|\|f(x) - \hat{f}(x)\| \\ \leq &\nu((K_f + K_{\hat{f}})\delta + \alpha) \\ < &\beta. \end{aligned} \tag{6.7}$$

By (6.5) and (6.7), we have $\nabla V(x) \cdot f(x) < \nabla V(x) \cdot \hat{f}(x) + \beta < 0$ on Ω_{c_1,c_2}. Hence all the trajectories in Ω_{c_2} converge to $\Omega_{c_1} \subset \mathcal{O}$. By Assumption 6.4.1, solutions of (6.1) will converge to the origin from $\mathcal{O}$. □

6.5 Data-Driven Algorithm for Deriving Stability Certificates

In this section, we continue to discuss the data-driven learning algorithm based on the derivation for learning the vector field and Lyapunov function in the previous two sections.

6.5.1 Recap of Learning the Koopman Generator

First, we select a finite dictionary of continuously differentiable observable functions, denoted by $\mathcal{Z}_N(x) := [\mathfrak{z}_1(x), \mathfrak{z}_2(x), \cdots, \mathfrak{z}_N(x)]$, $N \in \mathbb{N}$. Suppose that there exists a data-driven finite-dimensional approximation of $\mathcal{L}_{\lambda,\tau_s}$, denoted as L, such that for any $h \in \text{span}\{\mathfrak{z}_1, \mathfrak{z}_2, \cdots, \mathfrak{z}_N\}$, i.e., $h(x) = \mathcal{Z}_N(x)\zeta$ where ζ is a column vector in $\mathbb{R}^N$, we have that $\mathcal{L}h(\cdot) \approx \mathcal{L}_{\lambda,\tau_s}h(\cdot) \approx \mathcal{Z}_N(\cdot)(L\zeta)$.

By randomly sampling M initial conditions $\{x^{(m)}\}_{m=1}^M \subseteq \Omega$ and fixing a τ_s with a sampling rate of γ Hz along the trajectories, we construct the observable functions into X, as $\mathbf{B} = [\mathcal{Z}_N(x^{(1)})^\mathsf{T}, \mathcal{Z}_N(x^{(2)})^\mathsf{T}, \cdots, \mathcal{Z}_N(x^{(M)})^\mathsf{T}]^\mathsf{T}$. Under the specified sampling rate, to avoid inaccurate computation of the truncated integral of the Koopman resolvent (5.24) for large

λ, we first apply the Gauss–Legendre quadrature method [113] using trajectory samples to obtain the numerical approximation $\hat{\mathcal{R}}_{\mu,\tau_s}$ for $\mathcal{R}_{\mu,\tau_s}$ with a relatively small constant $\mu > 0$. Subsequently, the generator L[1] can be inferred as

$$L = X^{\dagger} Y \tag{6.8}$$

where $X = (\lambda - \mu)\hat{\mathcal{R}}_{\mu,\tau_s} + \mathbf{B}$, and $Y = \lambda\mu\hat{\mathcal{R}}_{\mu,\tau_s} - \lambda\mathbf{B}$.

By including x in the dictionary, the approximated vector field $\hat{f}(x) \approx \Xi\mathcal{Z}_N(x)^{\mathsf{T}}$, for some $\Xi \in \mathbb{R}^{n\times N}$. To ensure that the approximated dynamics preserve the same equilibrium point at the origin, we can perform $\tilde{f}(x) = \hat{f}(x) - \hat{f}(0)$, as demonstrated in [115]. In doing so, Assumption 6.4.1 is easier to be satisfied with $\tilde{f}(0) = 0$. Again, we would like to emphasize that with this proposed method, the vector field is learned via learning the Koopman generator using the discrete sampled points along the trajectories. Namely, one does not need to approximate the time derivatives and then identify the continuous-time systems. Moreover, it contributes to a more accurate identification, in comparison with finite-difference based methods, for instance, SINDy[94, 61].

6.5.2 Learning the Lyapunov Function

We consider a Lyapunov function candidate of the form $V(x) = \mathcal{Z}_N(x)\theta$ with $\theta \in \mathbb{R}^N$. Then, to solve the Lyapunov's equation (6.2) and the Zubov's equation (6.3) respectively, we need the Lyapunov function candidate to satisfy:

$$\mathcal{Z}_N(x)L\theta = -\eta(x), \tag{6.9}$$

and

$$[\mathcal{Z}_N(x)L - \eta(x)\mathcal{Z}_N(x)]\theta = -\eta(x), \tag{6.10}$$

respectively. Then, it appears that the problem is reduced to finding the vector θ such that the least-squares error between the l.h.s. and r.h.s. of (6.9) or (6.10) is minimized at the sampled initial conditions. However, directly solving the linear PDEs in such a way may not return the true solutions, especially for the Zubov's equation (6.10). For instance, it is evident that $W(x) = 1$ is a trivial solution to (6.3).

[1]Note that we employ the modified version of the data-driven approximation L_{mod} obtained with (5.36) here, whereas the updated version algorithm for L_{update} may yield more accurate results.

Next, we select a set of boundary points $\{y^{(p)}\}_{p=0}^{P-1}$ and formulate the least-squares problem for solving the Zubov's equation as:

$$\theta = \underset{K\in\mathbb{R}^N}{\arg\min} \frac{1}{M}\sum_{i=1}^{M} \left\| \left[\mathcal{Z}_N(x_i)L - \eta(x_i)\mathcal{Z}_N(x_i)\right] K + \eta(x_i)\right\|^2 + \lambda_b \frac{1}{P}\sum_{i=1}^{P} \left\|\mathcal{Z}_N(y_i)K - b(y_i)\right\|^2, \tag{6.11}$$

where $\lambda_b > 0$ is a weight parameter, and $b(y_i)$ is boundary values for the sampled y_i. In a similar vein, solving the Lyapunov function (6.9) can be formulated as:

$$\theta = \underset{K\in\mathbb{R}^N}{\arg\min} \frac{1}{M}\sum_{i=1}^{M} \left\|\mathcal{Z}_N(x_i)LK + \eta(x_i)\right\|^2 + \lambda_b \frac{1}{P}\sum_{i=1}^{P} \left\|\mathcal{Z}_N(y_i)K - b(y_i)\right\|^2. \tag{6.12}$$

For the positive definite function η, one common choice is $\eta(x) = r|x|^2$, where r is a positive constant. Unless otherwise specified, we will use it with $r = 0.1$ in all the following numerical experiments. The boundary condition for the Zubov's equation consists of both $W(0) = 0$ and $W(y) = 1$ for $y \notin \mathcal{D}$ (and $y \in \partial\mathcal{D}$ if the knowledge is available), whereas the one for the Lyapunov's equation is a single point $V(0) = 0$. Note that the positive definiteness of the Lyapunov function V can be ensured by solving the PDEs accurately with the chosen η [77].

Remark 6.5.1. *Note that with the identified vector field $\tilde{f}(x)$, the stability certificates can also be attained by directly solving the Lyapunov's equation $\nabla V(x;\theta)\cdot\tilde{f}(x) = -\eta(x)$ and the Zubov's equation $\nabla W(x;\theta)\cdot\tilde{f}(x) = -\eta(x)(1 - W(x;\theta))$, using the method proposed in [77]. However, the proposed method provides a more efficient approach, requiring only the solution of a least-squares problem by repeatedly using the same (observable) test functions.*

The process of identifying the dynamics and computing the Lyapunov functions using the Zubov's equation is summarized in Algorithm 6.

6.5.3 The Choice of the Dictionary

In this work, we primarily employ two types of observable functions widely recognized in the literature [94, 85]: monomials and shallow neural networks, featuring randomly initialized weights and biases.

Algorithm 6: Koopman generator-based Lyapunov function and system identification

Input: X, Y

Output: V or $\hat{f}\&V$

Learn L using (6.8) ;

if *Identifying the dynamics is needed* **then**

  Identify the dynamics $\hat{f}$ with L using (6.4) ;

  Learn the Lyapunov function V with (6.11) or (6.12) ;

end

else

  Learn the Lyapunov function V with (6.11) or (6.12);

end

It is well-known that one-hidden layer neural networks are able to approximate any continuous function on a compact set to any desired degree of accuracy [28]. More importantly, as illustrated in [49, 148], the ELM algorithm can solve the linear stability-related PDEs well using single-hidden layer feedforward neural networks. It returns the Lyapunov functions of form $V(x;\theta) = \sigma(\omega x + b)\theta$, where ω and b are randomly generated weights and bias, while θ is learned by performing a least-squares task. In this paper, we mainly consider the hyperbolic tangent function tanh as the activation function σ. That said, in order to simultaneously learn the vector field and the Lyapunov function via the Koopman generator, we set $\mathcal{Z}_N = [\tanh(\omega x + b)^\mathsf{T}, x^\mathsf{T}]$, where $\omega \in \mathbb{R}^{(N-n)\times n}, b \in \mathbb{R}^{N-n}$. Consequently, we have an approximation of the vector field: $\hat{f}(x) = \Xi[\tanh(\omega x + b)^\mathsf{T}, x^\mathsf{T}]^\mathsf{T}$ with $\Xi = L[(N-n) : end, :]$, and $V(x) = [\tanh(\omega x + b)^\mathsf{T}, x^\mathsf{T}]\theta$.

The above selection of observable functions applies to general nonlinear systems. However, with specific information about the systems, tailored test functions can improve approximation accuracy. For example, in polynomial systems, monomials can be used for more accurate results. Note that with the information, we potentially can get a better approximation of the Koopman generator, and thus better ROA estimates can be attained. In general, the more accurate the resulting generator and corresponding identified vector field are, the closer the ROA estimate can be to the actual ROA.

6.6 Numerical Experiments

We present two numerical examples to demonstrate the effectiveness of the proposed method. For both examples, we assume that the systems are unknown and set $Q = I$ when solving for a quadratic Lyapunov function V_P, as shown in Section 6.4. Recalling equations (6.9) and (6.10), the Zubov's equation is in general more challenging to solve and yields larger ROA estimates. Hence we focus exclusively on its results in this section. The Lyapunov's equation is a simpler case in terms of solving the linear PDE and can be readily computed by simplifying the process, mainly the residual term and the boundary conditions, with the provided code. The code is available at https://github.com/RuikunZhou/Unknown_Zubov_Koopman.

6.6.1 Reversed Van der Pol Oscillator

Consider the reversed Van der Pol oscillator $\dot{x}_1 = -x_2, \quad \dot{x}_2 = x_1 - (1 - x_1^2)x_2$, with $x := [x_1, x_2]$. Since this is a polynomial system, we utilize the monomials as the $N = J \times K$ basis functions with $\mathfrak{z}_i(x) = x_1^p x_2^q$, $p = \mod(i/J)$, and $q = \lfloor i/K \rfloor$. In order to solve the Zubov's equation (6.10) sufficiently well using polynomials, we choose $J = K = 8$. In this case, for generator learning, we only need to randomly sample $M = 100$ initial conditions on $\Omega = [-1.2, 1.2]^2$ and the data points along the trajectories with a sampling frequency $\gamma = 50Hz$ for $\tau_s = 5s$. In comparison, the feedforward neural network-based method in [150] needs 9 million samples with the exact values of the corresponding time derivative for each data point. As a natural result of the generator approximation L, we have the identified vector field $\tilde{f}$. Also, we verified that Assumption 6.4.1 indeed holds in this case.

Next, we randomly sample 3000 data points across $[-2.5, 2.5] \times [-3.5, 3.5]$ with 100 boundary points. Note that the boundary conditions given here do not need to be exactly on $\partial\mathcal{D}$, and the samples for generator learning can be reused here. For this example, we directly use the boundary points on the box which contains $\mathcal{D}$ as the boundary points for solving the PDE. We then solve Zubov's equation (6.10) using the learned L with the help of the well-developed toolbox for finding and verifying Lyapunov functions, LyZNet [76], resulting in a polynomial Lyapunov function. It is remarkable that, differing from the regular Lyapunov conditions on the Lie derivative $\dot{V}(x) < 0$ defined in Theorem 1.3.1, we verify the modified Lyapunov conditions for the Lie derivative as (6.5), using the integrated SMT solvers in the toolbox. The resulting Lyapunov function and the corresponding ROA are included in Fig. 6.1.

The parameters for learning the generators and verification for stability are detailed in

Table 6.1, where μ and λ are used to compute (6.8). We assume that the Lipschitz constant K_f is known. For $\hat{f}$ and V, we have the exact expressions, given that they are linear combinations of the observable functions. We thus can leverage symbolic computations and then compute $K_{\hat{f}}$ and ν with the help of the SMT solver, dReal [40] by performing interval analysis with the expressions. Note that when the 2-norm needs to be evaluated, it is over-approximated by the Frobenius norm for the expressions. It is noteworthy that while data is sampled on Ω to learn the dynamics within the domain of attraction, the certified ROA extends beyond this so-called valid region described in [150]. This extension is made possible by incorporating physical insights—specifically, leveraging the knowledge that the system is polynomial. With that, we have a more accurate approximation with improved generalization performance outside Ω. More importantly, both the system identification and stability certification processes require significantly less computational time compared to existing methods. The accuracy and efficiency also result in a substantially smaller β in (6.5), reduced by a factor of $\frac{1}{10}$ for verification. This reduction is critically important for practical applications involving real-world systems.

Table 6.1: Parameters in the Van der Pol Oscillator case, where the first four pertain to learning the Koopman generator, while the last six correspond to those in Proposition 6.4.2

$\tau_s(s)$	γ	μ	λ	λ_b	K_f	$K_{\hat{f}}$	δ	α	ν	β
5	50	2.5	1e8	100	4.90	4.90	3e-4	4.16e-6	7.07e-1	2.08e-3

6.6.2 Two-Machine Power System

Consider the two-machine power system in [132] with the following governing equation: $\dot{x}_1 = x_2,\ \dot{x}_2 = -0.5x_2 - (\sin(x_1 + a) - \sin(a))$, where $a = \frac{\pi}{3}$. In this example, the system is non-polynomial with a trigonometric function, and we assume no prior knowledge of its dynamics. The dictionary is constructed using 100 tanh-activated neural networks and the state variable x. We sample $M = 50^2$ initial conditions within $\Omega = [-1, 1]^2$, which lies entirely within the true domain of attraction. Following the same procedure as in the previous example, we solve the Zubov's equation over $[-2, 3] \times [-3, 1.5]$. The parameter settings are summarized in Table 6.2, and the resulting Lyapunov function and corresponding ROA estimate can be found in Fig. 6.2.

We emphasize that the Koopman generator-based method proposed in this work serves as a modular framework for identifying continuous-time systems and finding Lyapunov functions. After identifying the system, one can also directly solve the PDEs to find a

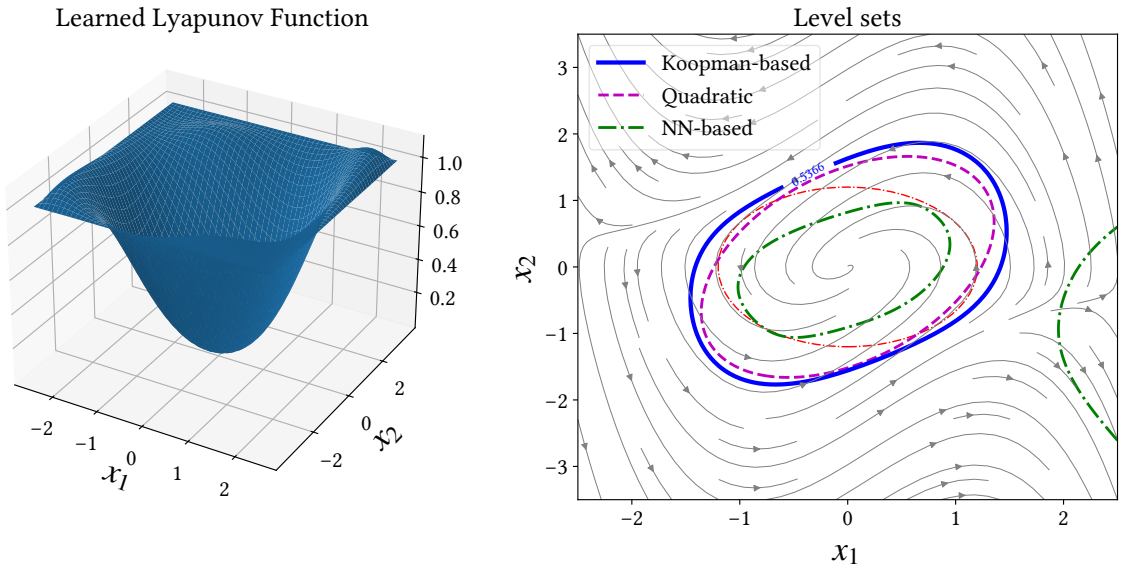


Figure 6.1: The learned Lyapunov function and corresponding certified ROA estimates, where the blue curve is the one using the proposed method. The magenta dashed one is the verified largest ROA estimate with $V_P(x)$, while the green dot-dashed line is with the neural network-based approach proposed in [150]. The red dot-dashed circle denotes Ω on which we collect the data for computing the Koopman generator.

Table 6.2: Parameters for the two-machine power system using tanh-activated neural networks

$\tau_s(s)$	γ	μ	λ	λ_b	K_f	$K_{\hat{f}}$	δ	α	ν	β
5	10	3	1e8	100	1.52	1.52	1e-4	2.72e-4	1.45	8.34e-4

Lyapunov function, as discussed in Remark 6.5.1. On the other hand, if the vector field is identified using other techniques, such as SINDy, the learned generator L is exclusively used for solving the stability-related PDEs. Notably, leveraging the generator for both system identification and stability analysis significantly reduces computational overhead, underscoring the efficiency of the proposed approach.

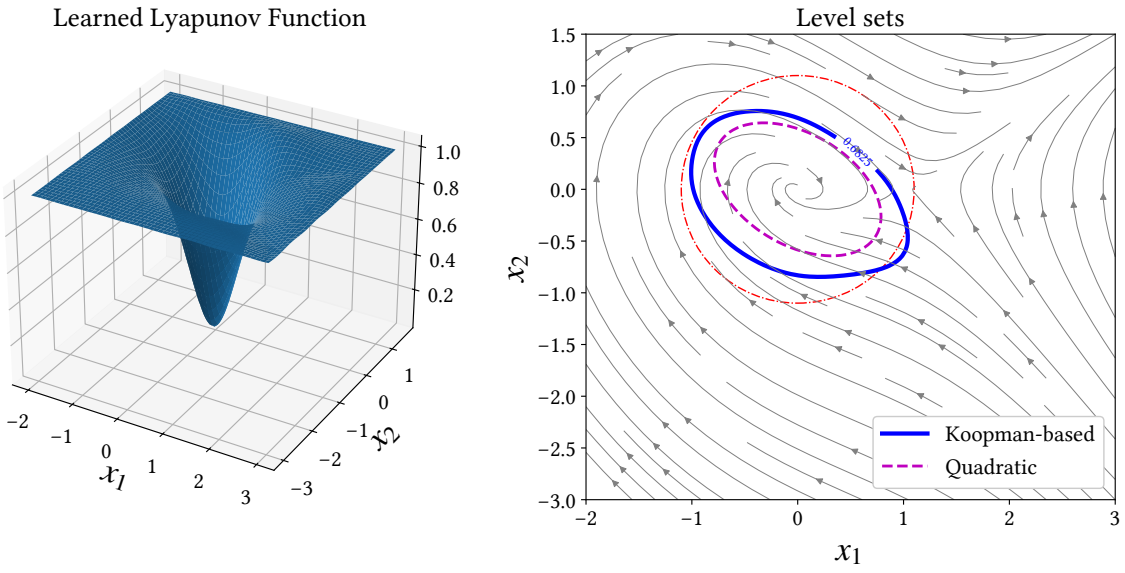


Figure 6.2: The learned Lyapunov function and corresponding certified ROA estimates, where the red dot-dashed circle denotes Ω on which we collect the data for computing the Koopman generator. The blue curve is the certified ROA estimate using the proposed method, while the magenta dashed line is the one with $V_P(x)$.

6.7 Summary

In this chapter, we propose a framework for simultaneously identifying the vector field and finding a Lyapunov function for an unknown nonlinear system by computing the Koopman generator. Using a shared dictionary of observable functions for both computing the Koopman generators and solving stability-related PDEs, we establish a more efficient way for constructing stability certificates from data. Leveraging SMT solvers further allows for less conservative ROA estimates, compared to the quadratic Lyapunov functions and an existing neural network-based method.

Chapter 7

Conclusions and Future Work

The objective of this thesis is to explore the utility of learning-based approaches to tackle the challenges in system identification and stability analysis of nonlinear dynamical systems. In this final chapter, we summarize the main contributions and bring up some future research directions.

Model-Based Stability Certification Computation

For nonlinear systems with known dynamics, we have shown that by taking advantage of ELM and linearity in the stability-related PDEs, we can efficiently compute the Lyapunov functions and corresponding optimal control policies. When it came to high-dimensional systems or the scenario where more complex neural networks were needed, PINN-based solutions could be employed and verified with more scalable SMT solvers or neural network verifiers.

One potential limitation of the framework is the increase in computational difficulty as the computation domain expands. Future work could investigate domain decomposition techniques to alleviate this challenge. Additionally, exploring advanced tools for solving large-scale linear least squares problems could maximize the framework's potential. Another promising area for future research is developing verification techniques specifically designed for one-hidden-layer neural networks. Our studies indicate that existing SMT solvers [40] are not optimized yet for this purpose.

Feedforward Neural Network-Based SysID and Stability Certification

A learning framework to simultaneously stabilize an unknown nonlinear system with a neural controller and learn a neural Lyapunov function to certify an ROA for the closed-loop system was proposed in Chapter 4. The algorithmic structure consists of two neural networks

and an SMT solver. The first neural network was responsible for learning the unknown dynamics. The second neural network aimed to identify a valid Lyapunov function and a provably stabilizing nonlinear controller, while the training was performed by imposing positive penalties on the violation of the Lyapunov conditions. The SMT solver then verified that the candidate Lyapunov function indeed satisfies the Lyapunov conditions. We also provided theoretical guarantees of the proposed learning framework in terms of the closed-loop stability for the unknown nonlinear system, and a set of numerical experiments was given to illustrate the effectiveness of the proposed approach.

However, the current approach suffers from high computational costs and inefficiencies in learning the system dynamics, requiring substantial amounts of training data and test data. Future work could focus on exploring more data-efficient approaches by designing better neural network architectures that can capture system dynamics with fewer parameters and training samples. Additionally, the computational cost of computing generalization error remains prohibitively high. Taking advantage of PAC (Probably Approximately Correct) learning theory to provide probabilistic bounds on the learning performance could be a promising direction to address this challenge. Further research could also investigate the scalability of the approach to high-dimensional systems and explore the robustness of the learned Lyapunov functions and controllers under uncertainties and disturbances.

Koopman Generator-Based System Identification and Stability Certification

In Chapters 5 and 6, we presented a novel resolvent operator-based learning framework for learning the system generator to relax the requirements on the observation frequency and overcome the constraints of taking operator logarithms. By leveraging the Yosida approximation and the first resolvent identity, this high-accuracy Koopman generator learning method captures transient system transitions, providing an alternative approach to identify continuous-time dynamics without approximating time derivatives. Moreover, the infinitesimal generator is a differential operator with respect to time. If the approximation of the Koopman generator was accurate enough and accessible, the generator could be used to construct a Lyapunov-based stability certificate for the unknown nonlinear system in the same function space. By formulating the linear PDEs as a linear least squares problem, Lyapunov functions can be computed efficiently with the dictionary of observable test functions. The learned Lyapunov functions can also be formally verified using an SMT solver and provide less conservative estimates of the region of attraction, compared to existing methods.

Although we have developed an updated version of the algorithm for approximating the Koopman generator, one of the current drawbacks of the proposed method still lies in tuning the parameter μ under the constraints of the observation rate, as well as in the lack

of understanding regarding the sampling efficiency of the initial conditions. We will pursue these analyses in future work and provide a more systematic and automated parameter tuning strategy, with the ultimate goal of developing a software toolbox for data-driven system verification of unknown systems. We also expect that the current research and the analysis of data efficiency will lay the foundations for studying control systems, shed light on dimension reduction in generator learning, and be beneficial for developing an online learning adaptation of this method with a provable error bound. Another possible extension direction based on the current research is robustness analysis under noisy and partial observations.

Furthermore, for computing Lyapunov functions within this framework, current numerical results focus on low-dimensional systems. A promising direction for future research is to extend the approach to high-dimensional dynamics by leveraging advanced verification tools.

References

[1] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. FOSSIL: a software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks. In *Proc. of HSCC*, pages 1–11, 2021.

[2] Minos Axenides and Emmanuel Floratos. Scaling properties of the Lorenz system and dissipative Nambu mechanics. In *Chaos, Information Processing And Paradoxical Games: The Legacy Of John S Nicolis*, pages 27–41. World Scientific, 2015.

[3] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020.

[4] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. cvc5: A versatile and industrial-strength SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Springer, 2022.

[5] Randal W Beard, George N Saridis, and John T Wen. Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. *Automatica*, 33(12):2159–2177, 1997.

[6] Randal Winston Beard. *Improving the closed-loop performance of nonlinear systems*. PhD thesis, Rensselaer Polytechnic Institute, 1995.

[7] Felix Berkenkamp, Riccardo Moriconi, Angela P Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4661–4666. IEEE, 2016.

[8] Felix Berkenkamp and Angela P Schoellig. Safe and robust learning control with Gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501. IEEE, 2015.

[9] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

[10] Nam P Bhatia and George P Szegö. *Dynamical Systems: Stability Theory and Applications*, volume 35. Springer, 1967.

[11] Mitchell Black and Dimitra Panagou. Safe control design for unknown nonlinear systems with Koopman-based fixed-time identification. *IFAC-PapersOnLine*, 56(2):11369–11376, 2023.

[12] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[13] Jason J Bramburger and Giovanni Fantuzzi. Auxiliary functions as Koopman observables: Data-driven analysis of dynamical systems via polynomial optimization. *Journal of Nonlinear Science*, 34(1):8, 2024.

[14] Alberto Bressan. *Lecture notes on functional analysis.* American Mathematical Society, 2012.

[15] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. *arXiv:2108.06266 [cs, eess]*, December 2021. arXiv: 2108.06266.

[16] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. *PLOS ONE*, 11(2):e0150171, February 2016.

[17] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 1 edition, January 2019.

[18] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, 2022.

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

[20] Fabio Camilli, Lars Grüne, and Fabian Wirth. A generalization of Zubov's method to perturbed systems. *SIAM Journal on Control and Optimization*, 40(2):496–515, 2001.

[21] Ricardo Carona, A Pedro Aguiar, and José Gaspar. Control of unicycle type robots tracking, path following and point stabilization. In *International Proceeding of IV Electronics and Telecommunications*. Citeseer, November 2008.

[22] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov control. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov Control. In *arXiv:2005.00611 [cs, eess, stat]*, December 2020. arXiv: 2005.00611.

[24] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. Learning lyapunov functions for piecewise affine systems with neural network controllers. *arXiv preprint arXiv:2008.06546*, 2020.

[25] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. Learning lyapunov functions for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[26] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. Learning region of attraction for nonlinear systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6477–6484. IEEE, 2021.

[27] Alessandro Chiuso and Gianluigi Pillonetto. System identification: A machine learning perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:281–304, 2019.

[28] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[29] Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1274–1281, 2020.

[30] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, 2021.

[31] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions. *arXiv:2109.06697 [cs, eess]*, October 2021. arXiv: 2109.06697.

[32] Tim De Ryck, Samuel Lanthaler, and Siddhartha Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

[33] Shankar A Deka, Alonso M Valle, and Claire J Tomlin. Koopman-based neural Lyapunov functions for general attractors. In *61st Conference on Decision and Control (CDC)*, pages 5123–5128. IEEE, 2022.

[34] Zlatko Drmač, Igor Mezić, and Ryan Mohr. Identification of nonlinear systems using the infinitesimal generator of the Koopman semigroup—a numerical implementation of the Mauroy–Goncalves method. *Mathematics*, 9(17):2075, 2021.

[35] Lawrence C Evans. *Partial Differential Equations*, volume 19. American Mathematical Society, 2022.

[36] Milad Farsi, Yinan Li, Ye Yuan, and Jun Liu. A piecewise learning framework for control of unknown nonlinear systems with stability guarantees. In *Learning for Dynamics and Control Conference*, pages 830–843. PMLR, 2022.

[37] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

[38] Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for Lyapunov function approximation. In *Proc. of CDC*, pages 2091–2096. IEEE, 2022.

[39] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. δ-complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, pages 286–300. Springer, 2012.

[40] Sicun Gao, Soonho Kong, and Edmund M Clarke. dReal: An SMT Solver for Nonlinear Theories over the Reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.

[41] Peter Giesl. *Construction of Global Lyapunov Functions Using Radial Basis Functions*, volume 1904. Springer, 2007.

[42] Lars Grüne. Computing Lyapunov functions using deep neural networks. *Journal of Computational Dynamics*, 8(2):131–152, 2021.

[43] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[44] Simon Haykin and N Network. A comprehensive foundation. *Neural networks*, 2(2004):41, 2004.

[45] Peter D Hislop and Israel Michael Sigal. *Introduction to spectral theory: With applications to Schrödinger operators*, volume 113. Springer Science & Business Media, 2012.

[46] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[47] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, June 2013.

[48] Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *arXiv preprint arXiv:2307.12306*, 2023.

[49] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[50] Jeen-Shing Wang and Yen-Ping Chen. A fully automated recurrent neural network for unknown dynamic system identification and control. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(6):1363–1372, June 2006.

[51] Yu Jiang and Zhong-Ping Jiang. *Robust adaptive dynamic programming*. John Wiley & Sons, 2017.

[52] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[53] Rudolph E. Kalman and Richard S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(1):95–108, 1961.

[54] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[55] Wei Kang. Zubov theorem and domain of attraction for controlled dynamic systems. In *Nonlinear Control Systems Design 1995*, pages 143–146. Elsevier, 1995.

[56] Wei Kang, Kai Sun, and Liang Xu. Data-driven computational methods for the domain of attraction and Zubov's equation. *IEEE Transactions on Automatic Control*, 2023.

[57] James Kapinski, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142, 2014.

[58] Hassan K Khalil. *Nonlinear Systems*. Pearson, 2001.

[59] H.K. Khalil. *Nonlinear Control*. Always Learning. Pearson, 2015.

[60] Larissa Khodadadi, Behzad Samadi, and Hamid Khaloozadeh. Estimation of region of attraction for polynomial nonlinear systems: A numerical method. *ISA transactions*, 53(1):25–32, 2014.

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

[62] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

[63] J Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32, 2019.

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

[65] Erwin Kreyszig. *Introductory functional analysis with applications*. John Wiley & Sons, 1991.

[66] Yuri A Kuznetsov. *Elements of applied bifurcation theory*. Springer, 1998.

[67] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[68] F W Lewis, S. Jagannathan, and A Yesildirak. *Neural Network Control Of Robot Manipulators And Non-Linear Systems*. CRC Press, 0 edition, August 2020.

[69] Frank L Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE circuits and systems magazine*, 9(3):32–50, 2009.

[70] Frank L Lewis, A Yeşildirek, and Kai Liu. Neural net robot controller: Structure and stability proofs. *Journal of Intelligent and Robotic Systems*, 12(3):277–299, 1995.

[71] Frank L Lewis, Aydin Yesildirek, and Kai Liu. Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Transactions on neural networks*, 7(2):388–399, 1996.

[72] Yuandan Lin, Eduardo D Sontag, and Yuan Wang. A smooth converse Lyapunov theorem for robust stability. *SIAM Journal on Control and Optimization*, 34(1):124–160, 1996.

[73] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404, 2021.

[74] Guo Ping Liu. *Nonlinear identification and control: a neural network approach*. Springer Science & Business Media, 2001.

[75] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Towards learning and verifying maximal neural Lyapunov functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 8012–8019. IEEE, 2023.

[76] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Tool LyZNet: A lightweight Python tool for learning and verifying neural Lyapunov functions and regions of attraction. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–8, 2024.

[77] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification. *Automatica*, 175:112193, 2025.

[78] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[79] Lennart Ljung. *System Identification: Theory for the User*. Pearson Education, 1998.

[80] Y. Long and M.M. Bayoumi. Feedback stabilization: control Lyapunov functions modelled by neural networks. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2812–2814 vol.3, 1993.

[81] Edward N. Lorenz. Predictability: A problem partly solved. In *Seminar on Predictability, 4–8 September 1995*, pages 1–18, Reading, Berkshire, UK, 1996. European Centre for Medium-Range Weather Forecasts (ECMWF).

[82] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, December 2018.

[83] Matteo Marchi, Bahman Gharesifard, and Paulo Tabuada. Training deep residual networks for uniform approximation guarantees. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 677–688. PMLR, 07 – 08 June 2021.

[84] Luis G. Matallana, Aníbal M. Blanco, and J. Alberto Bandoni. Estimation of domains of attraction: A global optimization approach. *Mathematical and Computer Modelling*, 52(3-4):574–585, August 2010.

[85] Alexandre Mauroy and Jorge Goncalves. Koopman-based lifting techniques for nonlinear systems identification. *IEEE Transactions on Automatic Control*, 65(6):2550–2565, 2019.

[86] Alexandre Mauroy and Igor Mezic. Global Stability Analysis Using the Eigenfunctions of the Koopman Operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, November 2016.

[87] Alexandre Mauroy, Y Susuki, and Igor Mezic. *Koopman operator in systems and control*, volume 484. Springer, 2020.

[88] Arash Mehrjou, Mohammad Ghavamzadeh, and Bernhard Scholkopf. Neural Lyapunov redesign. In *Conference on Learning for Dynamics & Control*, 2020.

[89] Yiming Meng, Hangyu Li, Melkior Ornik, and Xiaopeng Li. Koopman-based data-driven techniques for adaptive cruise control system identification. In *27th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2024.

[90] Yiming Meng, Yinan Li, Maxwell Fitzsimmons, and Jun Liu. Smooth converse Lyapunov-barrier theorems for asymptotic stability with safety constraints and reach-avoid-stay specifications. *Automatica*, 144:110478, 2022.

[91] Yiming Meng, Ruikun Zhou, and Jun Liu. Learning regions of attraction in unknown dynamical systems via Zubov-Koopman lifting: Regularities and convergence. *IEEE Transactions on Automatic Control*, 2025.

[92] Yiming Meng, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. Physics-informed neural network policy iteration: Algorithms, convergence, and verification. In *International Conference on Machine Learning*, pages 35378–35403. PMLR, 2024.

[93] Yiming Meng, Ruikun Zhou, Melkior Ornik, and Jun Liu. Koopman-based learning of infinitesimal generators without operator logarithm. In *The 63rd IEEE Conference on Decision and Control (CDC)*. IEEE, 2024.

[94] Yiming Meng, Ruikun Zhou, Melkior Ornik, and Jun Liu. Resolvent-type data-driven learning of generators for unknown continuous-time dynamical systems. *arXiv preprint arXiv:2411.00923*, 2024.

[95] Daniel A Messenger and David M Bortz. Weak sindy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

[96] Daniel A Messenger and David M Bortz. Weak sindy: Galerkin-based data-driven model selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, 2021.

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

[98] David Monniaux. A survey of satisfiability modulo theory. In *International Workshop on Computer Algebra in Scientific Computing*, pages 401–425. Springer, 2016.

[99] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[100] Ameneh Nejati, Abolfazl Lavaei, Sadegh Soudjani, and Majid Zamani. Data-driven estimation of infinitesimal generators of stochastic systems. *IFAC-PapersOnLine*, 54(5):277–282, 2021.

[101] Katsuhiko Ogata et al. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[102] Samuel Otto, Sebastian Peitz, and Clarence Rowley. Learning bilinear models of actuated Koopman generators from partially observed trajectories. *SIAM Journal on Applied Dynamical Systems*, 23(1):885–923, 2024.

[103] Wei Pan, Ye Yuan, Jorge Gonçalves, and Guy-Bart Stan. A sparse bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control*, 61(1):182–187, 2015.

[104] A. Papachristodoulou and S. Prajna. A tutorial on sum of squares techniques for systems analysis. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2686–2700, Portland, OR, USA, 2005. IEEE.

[105] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew M Peet, and Declan Jagt. SOSTOOLS version 4.00 sum of squares optimization toolbox for MATLAB. *arXiv preprint arXiv:1310.4716*, 2013.

[106] Patric Parks. Liapunov redesign of model reference adaptive control systems. *IEEE Transactions on Automatic Control*, 11(3):362–367, 1966.

[107] Omkar Sudhir Patil, Duc M Le, Max L Greene, and Warren E Dixon. Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network. *IEEE Control Systems Letters*, 6:1855–1860, 2021.

[108] Grigorios A Pavliotis and Andrew Stuart. *Multiscale Methods: Averaging and Homogenization*, volume 53. Springer Science & Business Media, 2008.

[109] Amnon Pazy. *Semigroups of Linear Operators and Applications to Partial Differential Equations*, volume 44. Springer Science & Business Media, 2012.

[110] Theodore J Perkins and Andrew G Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3(Dec):803–832, 2002.

[111] Vassilios Petridis and Stavros Petridis. Construction of neural network based Lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065. IEEE, 2006.

[112] Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao, and Lennart Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682, 2014.

[113] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[114] Danil V Prokhorov. A Lyapunov machine for stability analysis of nonlinear systems. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 2, pages 1028–1031. IEEE, 1994.

[115] Thanin Quartz, Ruikun Zhou, Hans De Sterck, and Jun Liu. Stochastic reinforcement learning with stability guarantees for control of unknown nonlinear systems. *arXiv preprint arXiv:2409.08382*, 2024.

[116] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[117] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.

[118] H.L. Royden and P. Fitzpatrick. *Real Analysis*. Prentice Hall, 2010.

[119] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2651–2659. AAAI Press, 2018.

[120] Peter Ruoff, Melinda K Christensen, Jana Wolf, and Reinhart Heinrich. Temperature dependency and temperature compensation in a model of yeast glycolytic oscillations. *Biophysical chemistry*, 106(2):179–192, 2003.

[121] Nader Sadegh and Roberto Horowitz. Stability analysis of an adaptive controller for robotic manipulators. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1223–1229. IEEE, 1987.

[122] Jagannathan Sarangapani. *Neural Network Control of Nonlinear Discrete-Time Systems*. CRC Press, 0 edition, October 2018.

[123] Shinichi Sato, Masaki Sano, and Yasuji Sawada. Practical methods of measuring the generalized dimension and the largest Lyapunov exponent in high dimensional chaotic systems. *Progress of theoretical physics*, 77(1):1–5, 1987.

[124] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[125] Peter J Schmid and J Sesterhenn. Dynamic mode decomposition of experimental data. In *8th International Symposium on Particle Image Velocimetry, Melbourne, Victoria, Australia*, 2009.

[126] Yoshihiko Susuki, Alexandre Mauroy, and Igor Mezic. Koopman resolvent: A Laplace-domain analysis of nonlinear autonomous dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 20(4):2013–2036, 2021.

[127] Naoya Takeishi and Yoshinobu Kawahara. Learning dynamics models with stable invariant sets. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*, pages 9782–9790, 2021.

[128] Andrew R. Teel and Laurent Praly. A smooth Lyapunov function from a class-$\mathcal{KL}$ estimate involving two positive semidefinite functions. *ESAIM: Control, Optimisation and Calculus of Variations*, 5:313–367, 2000.

[129] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[130] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T Johnson. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pages 3–17. Springer, 2020.

[131] Bhagyashree Umathe and Umesh Vaidya. Spectral Koopman method for identifying stability boundary. *IEEE Control Systems Letters*, 2023.

[132] Anthony Vannelli and Mathukumalli Vidyasagar. Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

[133] Joseph A. Vincent and Mac Schwager. Reachable Polyhedral Marching (RPM): A Safety Verification Algorithm for Robotic Systems with Deep Neural Network Components. *arXiv:2011.11609 [cs]*, April 2021. arXiv: 2011.11609.

[134] Chuanzheng Wang, Yiming Meng, Stephen L Smith, and Jun Liu. Data-driven learning of safety-critical control with stochastic control barrier functions. In *61st Conference on Decision and Control (CDC)*, pages 5309–5315. IEEE, 2022.

[135] Jie Wang. An intuitive tutorial to Gaussian processes regression. *arXiv preprint arXiv:2009.10862*, 2020.

[136] Jan C Willems. Dissipative dynamical systems part I: General theory. *Archive for Rational Mechanics and Analysis*, 45(5):321–351, 1972.

[137] 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*, 25:1307–1346, 2015.

[138] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

[139] Anton Xue, Lars Lindemann, and Rajeev Alur. Chordal sparsity for SDP-based neural network verification. *Automatica*, 161:111487, 2024.

[140] A Yeşildirek and Frank L Lewis. Feedback linearization using neural networks. *Automatica*, 31(11):1659–1664, 1995.

[141] Bowen Yi and Ian R Manchester. On the equivalence of contraction and Koopman approaches for nonlinear stability and control. *IEEE Transactions on Automatic Control*, 69(7):4336–4351, 2023.

[142] Guoqiang Yuan and Yinghui Li. Estimation of the regions of attraction for autonomous nonlinear systems. *Transactions of the Institute of Measurement and Control*, 41(1):97–106, January 2019.

[143] Ye Yuan, Xiuchuan Tang, Wei Zhou, Wei Pan, Xiuting Li, Hai-Tao Zhang, Han Ding, and Jorge Goncalves. Data driven discovery of cyber physical systems. *Nature communications*, 10(1):4894, 2019.

[144] Zhexuan Zeng, Jun Liu, and Ye Yuan. A generalized Nyquist-Shannon sampling theorem using the Koopman operator. *IEEE Transactions on Signal Processing*, 2024.

[145] Zhexuan Zeng, Zuogong Yue, Alexandre Mauroy, Jorge Gonçalves, and Ye Yuan. A sampling theorem for exact identification of continuous-time nonlinear dynamical systems. *IEEE Transactions on Automatic Control*, 2024.

[146] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31, 2018.

[147] Tao Zhang, Shuzhi Sam Ge, and Chang Chieh Hang. Stable adaptive control for a class of nonlinear systems using a modified Lyapunov function. *IEEE Transactions on Automatic Control*, 45(1):129–132, 2000.

[148] Ruikun Zhou, Maxwell Fitzsimmons, Yiming Meng, and Jun Liu. Physics-informed extreme learning machine Lyapunov functions. *IEEE Control Systems Letters*, 2024.

[149] Ruikun Zhou, Yiming Meng, Zhexuan Zeng, and Jun Liu. Learning Koopman-based stability certificates for unknown nonlinear systems. *arXiv preprint arXiv:2412.02807*, 2024.

[150] Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, 2022.

[151] Vrushabh Zinage and Efstathios Bakolas. Neural Koopman Lyapunov control. *Neurocomputing*, 527:174–183, 2023.

[152] V. I. Zubov. *Methods of A. M. Lyapunov and Their Application*. Noordhoff, 1964.

APPENDICES

Appendix A

Operator Theory

In this appendix, we collect some standard results on operator theory that are used in Chapters 5 and 6. The presentation follows classical references such as [14, 35, 45, 65].

A.1 Bounded Linear Operators

Let X and Y be Banach spaces over the same field $\mathbb{K}$ of scalars.

Definition A.1.1. *A **linear operator** is a mapping A from a subspace $\mathrm{dom}(A) \subseteq X$ into Y such that*

$$A(c_1u + c_2v) = c_1Au + c_2Av$$

for all $u, v \in \mathrm{dom}(A), c_1, c_2 \in \mathbb{K}$. Here, $\mathrm{dom}(A)$ is the domain of A, while range of A is the subspace

$$\mathrm{ran}(A) := \{Au \mid u \in \mathrm{dom}(A)\} \subset Y,$$

*and the **null space** or **kernel** of A is the subspace*

$$\ker(A) := \{u \in \mathrm{dom}(A) \mid Au = 0\}.$$

Definition A.1.2. *A linear operator $A : X \to Y$ is **bounded** if*

$$\|A\| := \sup\{\|Au\|_Y \mid \|u\|_X \leq 1\} < \infty.$$

One can easily check that a bounded linear operator $A : X \to Y$ is continuous, and we have the following fundamental result in functional analysis.

Theorem A.1.3. *A linear operator $A : X \to Y$ is bounded if and only if it is continuous.*

Definition A.1.4. *A linear operator $A : \mathrm{dom}(A) \subset X \to Y$ is **densely defined** if $\mathrm{dom}(A)$ is dense in X, i.e., $\overline{\mathrm{dom}(A)} = X$.*

Definition A.1.5. *A linear operator $A : X \to Y$ is called **closed** if whenever $u_k \to u$ in X and $Au_k \to v$ in Y, then*

$$Au = v.$$

Theorem A.1.6 (Closed Graph Theorem)**.** *Let $A : X \to Y$ be a closed, linear operator with $\mathrm{dom}(A) = X$. Then A is bounded.*

Theorem A.1.7. *We denote by $\mathcal{B}(X, Y)$ the family of all bounded linear operators from X to Y. If Y is a Banach space, then $\mathcal{B}(X, Y)$ is a Banach space.*

A.2 Inverse Operators

In this section, we summarize some standard results on inverse operators of bounded linear operators.

Definition A.2.1. *An operator $A^{-1} : \mathrm{ran}(A) \to \mathrm{dom}(A)$ is called the **inverse** of A if $A^{-1}A = \mathrm{Id}_{\mathrm{dom}(A)}$ (the identity map on $\mathrm{dom}(A)$) and $AA^{-1} = \mathrm{Id}_{\mathrm{ran}(A)}$ (the identity map on $\mathrm{ran}(A)$).*

Theorem A.2.2. *Let $A : \mathrm{dom}(A) \subset X \to \mathrm{ran}(A) \subset Y$ be a linear operator. Then:*

*(1) An operator A has an **inverse** if and only if $\ker(A) = \{0\}$.*

(2) If A^{-1} exists, it is a linear operator.

Proof: (1) "$\Longleftarrow$" Suppose $\ker A = \{0\}$. For $u_1, u_2 \in \mathrm{dom}(A)$, we have $Au_1 = Au_2 \Rightarrow u_1 = u_2$. Thus, the preimage is unique, and we can define an operator

$$A^{-1} : \mathrm{ran}(A) \to \mathrm{dom}(A), \qquad A^{-1}v := u \text{ where } Au = v.$$

This operator is well-defined and linear. Moreover, for $x \in \mathrm{dom}(A)$ we have $A^{-1}Au = A^{-1}(Au) = u$, so $A^{-1}A = \mathrm{Id}_{\mathrm{dom}(A)}$, and for $v \in \mathrm{ran}(A)$ we have $AA^{-1}v = Au = v$, so $AA^{-1} = \mathrm{Id}_{\mathrm{ran}(A)}$. Therefore, A has an inverse.

"$\Longrightarrow$" Conversely, suppose there exists $A^{-1} : \mathrm{ran}(A) \to \mathrm{dom}(A)$ with $A^{-1}A = \mathrm{Id}_{\mathrm{dom}(A)}$ and $AA^{-1} = \mathrm{Id}_{\mathrm{ran}(A)}$. If $u \in \ker A$, then

$$u = A^{-1}Au = A^{-1}0 = 0,$$

so $\ker A = \{0\}$.

(2) Suppose exists A^{-1} exists. We consider any $u_1, u_2 \in \mathrm{dom}(A)$ with images $v_1 = Au_1$, $v_2 = Au_2$, and $c_1, c_2 \in \mathbb{K}$. Then $u_1 = A^{-1}v_1$, $u_2 = A^{-1}v_2$. Since A is linear, we have

$$A(c_1u_1 + c_2u_2) = c_1Au_1 + c_2Au_2 = c_1v_1 + c_2v_2.$$

Applying A^{-1} to both sides gives

$$A^{-1}A(c_1u_1 + c_2u_2) = A^{-1}(c_1v_1 + c_2v_2).$$

Since $u_i = A^{-1}v_i$, this implies

$$c_1A^{-1}v_1 + c_2A^{-1}v_i = A^{-1}(c_1v_1 + c_2v_2).$$

Thus, A^{-1} is linear. ■

Definition A.2.3. *A linear operator A is **invertible** if it has a bounded inverse defined on all of Y. In this case,* $\mathrm{ran}(A) = Y$*, and A is called **bijective** from* $\mathrm{dom}(A)$ *to Y.*

Note that we distinguish the invertibility of the operator A from the existence of inverses for A, which may be unbounded and not defined on all of Y.

Theorem A.2.4 (Bounded Inverse Theorem)**.** *Let $A : X \to Y$ be a bounded, linear operator. If A is bijective, then $A^{-1} : Y \to X$ is bounded.*

A.3 Spectral Theory

In this section, we summarize some standard results on the spectral theory of linear operators.

Definition A.3.1. *Let A be a linear operator on X with domain* $\mathrm{dom}(A)$.

*(1) The **spectrum** of A is the set*

$$\sigma(A) \;=\; \{\lambda \in \mathbb{C} \mid A - \lambda \,\mathrm{Id} \;\; \textit{is not invertible}\}.$$

*(2) The **resolvent set** of A is the set*

$$\rho(A) \;=\; \{\lambda \in \mathbb{C} \mid A - \lambda \,\mathrm{Id} \;\; \textit{is invertible}\}.$$

*(3) If $\lambda \in \rho(A)$, the inverse of $A - \lambda \,\mathrm{Id}$ is called the **resolvent** of A at λ and is denoted*

$$R_A(\lambda) \;:=\; (A - \lambda \,\mathrm{Id})^{-1}.$$

By definition, we have $\sigma(A) \cup \rho(A) \;=\; \mathbb{C}$ and $\sigma(A) \cap \rho(A) \;=\; \emptyset$.

Theorem A.3.2. *The resolvent set $\rho(A)$ of a bounded linear operator is an open subset of $\mathbb{C}$; hence $\sigma(A)$ is closed. Moreover, $R_A(\lambda)$ is an analytic operator-valued function of λ on $\rho(A)$.*

Remark A.3.3. *A map $\lambda \in P \subset \mathbb{C} \mapsto A(\lambda) \in \mathcal{B}(X, X)$ is said to be* (norm) analytic *at $\lambda_0 \in P$ if $A(\lambda)$ has a power series expansion in $(\lambda - \lambda_0)$ (with coefficients in $\mathcal{B}(X, X)$) that converges in the operator norm with nonzero radius of convergence; that is, there exist bounded operators A_n such that*

$$A(\lambda) = \sum_{n=0}^{\infty} (\lambda - \lambda_0)^n A_n.$$

There are basically three reasons why $A - \lambda \,\mathrm{Id}$ fails to be invertible:

1. $\ker(A - \lambda \,\mathrm{Id}) \neq \{0\}$;
2. $\ker(A - \lambda \,\mathrm{Id}) = \{0\}$, and $\mathrm{ran}(A - \lambda \,\mathrm{Id})$ is dense so that $(A - \lambda \,\mathrm{Id})$ has a densely defined inverse but it is unbounded;
3. $\ker(A - \lambda \,\mathrm{Id}) = \{0\}$, but $\mathrm{ran}(A - \lambda \,\mathrm{Id})$ is not dense; in this case $(A - \lambda \,\mathrm{Id})^{-1}$ exists and may be bounded on $\mathrm{ran}(A - \lambda \,\mathrm{Id})$, but it is not densely defined and therefore cannot be uniquely extended to a bounded operator on X.

According to these three situations, we classify $\sigma(A)$ as follows.

Definition A.3.4.

(1) If $\lambda \in \sigma(A)$ is such that $\ker(A - \lambda \,\mathrm{Id}) \neq \{0\}$*, then λ is an* eigenvalue *of A, and any $u \in \ker(A - \lambda \,\mathrm{Id})$, $u \neq 0$, is an* eigenvector *of A for λ and satisfies $Au = \lambda u$. Moreover,* $\dim \ker(A - \lambda \,\mathrm{Id})$ *is called the (geometric)* multiplicity *of λ, and* $\ker(A - \lambda \,\mathrm{Id})$ *is the (geometric)* eigenspace *of A at λ.*

(2) The ***discrete spectrum*** *of A, $\sigma_d(A)$, is the set of all eigenvalues of A with finite (algebraic) multiplicity and which are isolated points of $\sigma(A)$.*

(3) The ***essential spectrum*** *of A is the complement of $\sigma_d(A)$ in $\sigma(A)$:*

$$\sigma_{\mathrm{ess}}(A) \;:=\; \sigma(A) \setminus \sigma_d(A).$$

A.4 Properties of the Resolvent

This section collects some interesting and basic properties of the resolvent of A, $R_A(\lambda) = (A - \lambda \,\mathrm{Id})^{-1}$, $\lambda \in \rho(A)$.

Proposition A.4.1. *Let A be a linear operator on X. Then for $\mu, \lambda \in \rho(A)$,*

(1) $R_A(\lambda)R_A(\mu) = R_A(\mu)R_A(\lambda)$*;*

(2) (First Resolvent Identity) $R_A(\lambda) - R_A(\mu) = (\lambda - \mu)\, R_A(\lambda)\, R_A(\mu)$.

Proof: We first prove (2):

$$\begin{aligned} R_A(\lambda) - R_A(\mu) &= R_A(\lambda)(A - \mu \,\mathrm{Id})R_A(\mu) - R_A(\lambda)(A - \lambda \,\mathrm{Id})R_A(\mu) \\ &= -\mu R_A(\lambda)R_A(\mu) + R_A(\lambda)R_A(\mu)\lambda \\ &= (\lambda - \mu)R_A(\lambda)R_A(\mu), \end{aligned}$$

which holds on all X. By interchanging μ and λ in the previous identity, we have:

$$R_A(\mu) - R_A(\lambda) = (\mu - \lambda)R_A(\mu)R_A(\lambda).$$

Comparing the results, we get the commutativity of $R_A(\mu)$ and $R_A(\lambda)$ in (1). ■

With the first resolvent identity in (2), we have the generalization of the resolvent operator, so-called pseudo resolvents, defined as follows.

Definition A.4.2 (Pseudo Resolvents)**.** *A family of bounded operators* $\{J(\lambda)\}_{\lambda\in\Delta}$ *on a Banach space* X *is called a* ***pseudo resolvent*** *on* $\Delta \subset \mathbb{C}$ *if*

$$J(\lambda) - J(\mu) = (\lambda - \mu)J(\lambda)J(\mu)$$

for all $\lambda, \mu \in \Delta$*.*

Note that although we have defined $\sigma(A)$ and $\rho(A)$, either of them could be empty, as there are examples of unbounded operators with empty spectrum and some with empty resolvent sets. However, if A is bounded, one can show that $\sigma(A)$ is nonempty using the resolvent and some elementary complex analysis.

Theorem A.4.3. *Let* $A \in \mathcal{B}(X, Y)$*. Then* $\sigma(A)$ *is nonempty and is a closed subset of* $\{z \in \mathbb{C} : |z| \leq \|A\|\}$*.*

The detailed proof of this theorem can be found in [45, Theorem 1.7].

Whereas Proposition A.4.1 compares the resolvent of a fixed operator at two points in its resolvent set, the second resolvent identity below compares the resolvents of two operators at a common point in their resolvent sets.

Proposition A.4.4 (Second Resolvent Identity)**.** *Let* A *and* B *be two closed operators with* $z \in \rho(A) \cap \rho(B)$*. Then*

$$\begin{aligned} R_A(z) - R_B(z) &= R_A(z)(A - B)R_B(z) \\ &= R_B(z)(B - A)R_A(z). \end{aligned} \tag{A.1}$$

One can easily show that the identities in this proposition hold by applying the following identity: $a^{-1} - b^{-1} = a^{-1}(b - a)b^{-1}$. Moreover, the right-hand side in (A.1) defines a bounded operator.

Appendix B

Semigroups of Linear Operators

This appendix provides a brief introduction to the semigroups of linear operators, which provides the background knowledge on the semigroups of the Koopman operator family used in Chapters 5 and 6. We follow the references from [109, 14] for succinct definitions.

B.1 Strongly Continuous Semigroups

Definition B.1.1. *Let X be a Banach space. A* ***strongly continuous semigroup of linear operators*** *on X is a family of linear maps $\{\mathcal{S}_t\}_{t\geq 0}$ with the following properties:*

(1) Each $\mathcal{S}_t : X \to X$ is a bounded linear operator.

(2) For every $s, t \geq 0$, the composition satisfies $\mathcal{S}_t\mathcal{S}_s = \mathcal{S}_{t+s}$ (semigroup property). Also, $\mathcal{S}_0 = I$ (the identity operator).

(3) For every $u \in X$, the map $t \mapsto \mathcal{S}_t u$ is continuous from $[0, \infty)$ into X.

A strongly continuous semigroup of bounded linear operators on X is often called **a semigroup of class $\mathcal{C}_0$** or simply a **$\mathcal{C}_0$-semigroup**. In the meantime, we say that $\{\mathcal{S}_t\}_{t\geq 0}$ is a **semigroup of type** ω if the linear operators $\mathcal{S}_t$ satisfy the bounds

$$\|\mathcal{S}_t\| \leq e^{t\omega}, \quad \text{for all } t \geq 0. \tag{B.1}$$

A semigroup of type $\omega = 0$ is called a **contractive semigroup**. In this case, $\|\mathcal{S}_t\| \leq 1$ for every $t \geq 0$, hence

$$\|\mathcal{S}_t u - \mathcal{S}_t v\| \leq \|u - v\|, \quad \forall\, u, v \in X,\, t \geq 0.$$

If moreover $\{\mathcal{S}_t\}_{t\geq 0}$ is a $\mathcal{C}_0$-semigroup, we call it a $\mathcal{C}_0$-semigroup of contractions.

Definition B.1.2. *The **(infinitesimal) generator** $\mathcal{A}$ of $\{\mathcal{S}_t\}_{t\geq 0}$ is defined by*

$$\mathcal{A}u := \lim_{t\downarrow 0} \frac{\mathcal{S}_t u - u}{t}. \tag{B.2}$$

Its domain $\mathrm{dom}(\mathcal{A})$ *is the set of all $u \in X$ for which the limit in (B.2) exists.*

From a differential equation's perspective, one may consider a linear evolution equation in X:

$$\frac{d}{dt}u(t) = \mathcal{A}u(t), \qquad u(0) = \bar{u} \in X. \tag{B.3}$$

For $t \geq 0$, we would like to express the solution as $u(t) = e^{t\mathcal{A}}\bar{u}$, for some family of linear operators $\{e^{t\mathcal{A}};\, t \geq 0\}$.

B.2 Properties of the Semigroup and the Generator

In the following, we summarize some important properties of $\mathcal{C}_0$-semigroups $\{\mathcal{S}_t\}_{t\geq 0}$ and the generator $\mathcal{A}$ on X.

Theorem B.2.1 (Properties of Semigroups)**.** *Let $\{\mathcal{S}_t\}_{t\geq 0}$ be a strongly continuous semigroup and let $\mathcal{A}$ be its generator. Assume $\bar{u} \in \mathrm{dom}(\mathcal{A})$. Then:*

(i) For every $t \geq 0$, $\mathcal{S}_t\bar{u} \in \mathrm{dom}(\mathcal{A})$ and $\mathcal{A}\mathcal{S}_t\bar{u} = \mathcal{S}_t\mathcal{A}\bar{u}$.

(ii) The map $t \mapsto u(t) := \mathcal{S}_t\bar{u}$ is continuously differentiable and provides a solution to the Cauchy problem (B.3).

Theorem B.2.2 (Properties of Generators)**.** *Let $\{\mathcal{S}_t\}_{t\geq 0}$ be a strongly continuous semigroup on the Banach space X, and let $\mathcal{A}$ be its generator. Then:*

(i) The domain of $\mathcal{A}$ is dense in X.

(ii) The operator $\mathcal{A}$ is closed.

The next theorem concerns the existence of a semigroup with a given $\mathcal{A} : \mathrm{dom}(\mathcal{A}) \subset X \to X$ as its generator.

Theorem B.2.3 (Existence of the Semigroup Generated by a Linear Operator)**.** *Let $\mathcal{A}$ be a linear operator on a Banach space X. Then the following are equivalent:*

(i) *$\mathcal{A}$ is the generator of a semigroup of linear operators $\{\mathcal{S}_t\}_{t\geq 0}$ of type ω.*

(ii) *$\mathcal{A}$ is a closed, densely defined operator. Moreover, every real number $\lambda > \omega$ is in the resolvent set of $\mathcal{A}$, and*

$$\|R_{\mathcal{A}}(\lambda) := (\lambda \,\mathrm{Id} - \mathcal{A})^{-1}\| \leq \frac{1}{\lambda - \omega}, \qquad \text{for all } \lambda > \omega. \tag{B.4}$$

In the case $\omega = 0$, the semigroup is contractive, and we have the well-known Hille-Yosida theorem for the $\mathcal{C}_0$-semigroup of contractions, as stated below.

Theorem B.2.4. *A linear (unbounded) operator $\mathcal{A}$ on X is the infinitesimal generator of a $\mathcal{C}_0$-semigroup of contractions $\{\mathcal{S}_t\}_{t\geq 0}$ if and only if*

(i) *$\mathcal{A}$ is closed and densely defined.*

(ii) *For every real number $\lambda > 0$, λ is in the resolvent set of $\mathcal{A}$, and*

$$\|R_{\mathcal{A}}(\lambda)\| \leq \frac{1}{\lambda}, \qquad \text{for all } \lambda > 0. \tag{B.5}$$

Given a semigroup of linear operators $\{\mathcal{S}_t\}_{t\geq 0}$, the generator is uniquely defined by the limit in (B.2). Conversely, the following theorem states that the semigroup generated by $\mathcal{A}$ is also uniquely determined.

Theorem B.2.5 (Uniqueness of the Semigroup)**.** *Let $\{\mathcal{S}_t\}_{t\geq 0}$ and $\{\tilde{\mathcal{S}}_t\}_{t\geq 0}$ be two strongly continuous semigroups of linear operators having the same generator $\mathcal{A}$. Then $\mathcal{S}_t = \tilde{\mathcal{S}}_t$ for every $t \geq 0$.*

We refer the reader to [14, Chapter 7] for the detailed proofs of the foregoing theorems.

B.3 Yosida Approximation

In this section, we introduce the Yosida approximation, which is a useful tool for approximating an unbounded operator by a sequence of bounded linear operators based on their resolvent families. First, we need the following lemma.

Lemma B.3.1. *Let $\mathcal{A}$ satisfy the conditions (i) and (ii) of Theorem B.2.4. Then*

$$\lim_{\lambda\to\infty} \lambda R_{\mathcal{A}}(\lambda)u = u, \quad \text{for all } u \in X. \tag{B.6}$$

Proof: Suppose that $u \in \mathrm{dom}(\mathcal{A})$. Then

$$\|\lambda R_{\mathcal{A}}(\lambda)u - u\| = \|\mathcal{A}R_{\mathcal{A}}(\lambda)u\| = \|R_{\mathcal{A}}(\lambda)\mathcal{A}u\| \leq \frac{1}{\lambda}\|\mathcal{A}u\| \to 0, \quad \text{as} \quad \lambda \to \infty.$$

But $\mathrm{dom}(\mathcal{A})$ is dense in X and $\|\lambda R_{\mathcal{A}}(\lambda)\| \leq 1$ for all $\lambda > 0$. Hence, (B.6) holds for all $u \in X$. ■

Based on the above lemma, we can define the Yosida approximation of $\mathcal{A}$ as follows.

Definition B.3.2. *For every $\lambda > 0$, the **Yosida approximation** of $\mathcal{A}$ is defined by*

$$\mathcal{A}_\lambda := \lambda\mathcal{A}R_{\mathcal{A}}(\lambda) = \lambda^2 R_{\mathcal{A}}(\lambda) - \lambda\,\mathrm{Id}\,. \tag{B.7}$$

$\mathcal{A}_\lambda$ is a bounded linear operator on X for every $\lambda > 0$, and it is an approximation of $\mathcal{A}$ in the following sense:

Theorem B.3.3 (Approximation of the Generator)**.** *Let $\mathcal{A}$ satisfy the conditions (i) and (ii) of Theorem B.2.4. Then, for every $u \in \mathrm{dom}(\mathcal{A})$,*

$$\lim_{\lambda\to\infty} \mathcal{A}_\lambda u = \mathcal{A}u. \tag{B.8}$$

Proof: For every $u \in \mathrm{dom}(\mathcal{A})$, by Lemma B.3.1 and Definition B.3.2 we have

$$\lim_{\lambda\to\infty} \mathcal{A}_\lambda u = \lim_{\lambda\to\infty} \lambda\mathcal{A}R_{\mathcal{A}}(\lambda)u = \mathcal{A}u$$

■

Corollary B.3.4. *Let $\mathcal{A}$ be the infinitesimal generator of a $\mathcal{C}_0$-semigroup of contractions $\{\mathcal{S}_t\}_{t\geq 0}$. If $\mathcal{A}_\lambda$ is the Yosida approximation of $\mathcal{A}$, then for every $u \in X$,*

$$\mathcal{S}_t u = \lim_{\lambda\to\infty} e^{t\mathcal{A}_\lambda}u, \quad \text{for all } t \geq 0. \tag{B.9}$$

Corollary B.3.5. *Let $\mathcal{A}$ be the infinitesimal generator of a $\mathcal{C}_0$-semigroup of contractions $\{\mathcal{S}_t\}_{t\geq 0}$. Then every complex number λ with positive real part is in the resolvent set of $\mathcal{A}$, i.e., $\{\lambda \mid \mathrm{Re}(\lambda) > 0\} \subseteq \rho(\mathcal{A})$, and for such λ,*

$$\|R_{\mathcal{A}}(\lambda)\| \leq \frac{1}{\mathrm{Re}(\lambda)}. \tag{B.10}$$

The proofs of these two corollaries are omitted here, and we refer the reader to [109, Section 1.3, Chapter 1] for details.