Applications of Linear Systems Theory to Koopman Operator Approximation

Steven Dahdah

Department of Mechanical Engineering
McGill University, Montreal

August 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Abstract

This thesis considers the use of linear systems theory in the context of Koopman operator approximation. The Koopman operator allows nonlinear systems to be represented as infinite-dimensional linear systems by viewing their evolution in terms of an infinite set of lifting functions. Data-driven methods are used to identify a finite-dimensional approximation of the Koopman operator using a finite selection of lifting functions. Thanks to its linearity, the approximate Koopman model can be used for analysis, design, and optimal controller or observer synthesis.

Depending on the choice of lifting functions, Koopman operator approximation methods can incorrectly identify unstable models for asymptotically stable systems. Two methods are proposed to guarantee asymptotically stable Koopman models: the first constrains the spectral radius of the Koopman matrix, while the second regularizes the $\mathcal{H}_\infty$ norm of the Koopman system.

Inherently unstable systems often require a stabilizing controller to operate practically. To account for the structure of a closed-loop system, a closed-loop Koopman operator identification method is proposed, in which Koopman models of the closed-loop system and plant are simultaneously identified given prior knowledge of the controller.

The linearity of the Koopman operator allows uncertainty within a population of models to be characterized in the frequency domain. Linear robust control tools can therefore be applied to a population of approximate Koopman models. A robust Koopman observer synthesis method is proposed, based on mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control. Its advantages are demonstrated experimentally using a population of several dozen motor drives.

Selecting lifting functions is a critical part of any Koopman operator approximation method, but there is currently no systematic selection process beyond trial-and-error. The `pykoop` Python library, presented in this thesis, helps address this issue by making it fast and easy to compose, parameterize, and optimize lifting functions.

Résumé

Cette thèse considère l'utilisation de la théorie des systèmes linéaires dans le contexte de l'approximation de l'opérateur de Koopman. L'opérateur de Koopman permet de représenter les systèmes non linéaires comme des systèmes linéaires de dimension infinie en considérant leur évolution en termes d'un ensemble infini de fonctions de levage. Des méthodes basées sur les données sont utilisées pour identifier une approximation finie de l'opérateur de Koopman à l'aide d'une sélection finie de fonctions de levage. Grâce à sa linéarité, le modèle de Koopman approximatif peut être utilisé pour l'analyse, la conception et la synthèse de contrôleurs ou d'observateurs optimaux.

Selon le choix des fonctions de levage, les méthodes d'approximation de l'opérateur de Koopman peuvent identifier incorrectement des modèles instables pour des systèmes asymptotiquement stables. Deux méthodes sont proposées pour garantir des modèles de Koopman asymptotiquement stables : la première contraint le rayon spectral de la matrice de Koopman, tandis que la seconde régularise la norme $\mathcal{H}_\infty$ du système de Koopman.

Les systèmes intrinsèquement instables nécessitent souvent un contrôleur stabilisant pour fonctionner de manière pratique. Pour tenir compte de la structure d'un système en boucle fermée, une méthode d'identification de l'opérateur de Koopman en boucle fermée est proposée, dans laquelle les modèles de Koopman des systèmes en boucle fermée et en boucle ouverte sont identifiés simultanément à partir d'une connaissance préalable du contrôleur.

La linéarité de l'opérateur de Koopman permet de caractériser l'incertitude dans une population de modèles dans le domaine des fréquences. Les outils de contrôle robustes linéaires peuvent donc être appliqués à une population de modèles de Koopman approximatifs. Une méthode de synthèse d'observateur de Koopman robuste est proposée, basée sur un contrôle optimal mixte $\mathcal{H}_2$-$\mathcal{H}_\infty$. Ses avantages sont démontrés expérimentalement à l'aide d'une population de plusieurs dizaines de moteurs.

La sélection des fonctions de levage est une partie critique de toute méthode d'approximation de l'opérateur de Koopman, mais il n'existe actuellement aucun processus de sélection systématique au-delà de l'essai et de l'erreur. Le logiciel Python `pykoop`, présenté dans cette thèse, aide à résoudre ce problème en rendant rapide et facile la composition, le paramétrage et l'optimisation des fonctions de levage.

Acknowledgements

My sincerest thanks go to my supervisor, Professor James Richard Forbes, for always encouraging me to do my best work, listening to my point of view, and guiding me through the highs and lows of the past five years. In 2019, I was adamant that I would never do a Ph.D., but spending a year working with Prof. Forbes and the DECAR group changed my view of research completely. I am tremendously lucky to have been able to study among such kind, brilliant people. They are too numerous to name individually, but they have all donated their time and expertise to me at one point or another, and many have become my closest friends.

I must also thank my mother Joanne Scullion, and my siblings Caroline, Nick, and Christina Dahdah for their unconditional love and support. I am grateful to my partner, Lilian Liu, for always believing in me, but never hesitating to tell me that my writing made no sense. Of course, I can't forget our cat, Galadriel, for never letting me work past (her) dinnertime.

This work was supported financially by the NSERC Discovery Grants program, the FRQNT, IVADO, CIFAR, the CRM, and by Mecademic through the Mitacs Accelerate program. Thanks to Daniel Bruder, Xun Fu, and Ram Vasudevan for graciously providing the soft robot dataset used in Chapter 5, which was funded in part by the Toyota Research Institute, the NSF Career Award (grant no. 1751093), and the ONR (grant no. N00014-18-1-2575). The FASTER dataset in Chapter 5 was funded in part by the NRC. Thanks to Quanser for the use of the *QUBE-Servo* rotary inverted pendulum system in Chapter 6.

Thanks to Eric Boutet, Jonathan Coulombe, Martin Dionne, and everyone else at Mecademic for being excellent mentors, and for showing me how robots are really made. I must also acknowledge Doug Shi-Dong, Robyn Fortune, Liam Paull, Jason Bramburger, Shaowu Pan, Karthik Duraisamy, and Matthew M. Peet for productive discussions about regularization techniques, the Koopman operator, RFFs, and methods for handling BMI constraints.

Table of Contents

List of Figures

Figure

List of Tables

Table

List of Abbreviations

List of Symbols

$\mathbb{R}$	the set of real numbers
$\mathbb{C}$	the set of complex numbers
$\mathbb{Z}$	the set of integers
$\mathbb{R}_{\geq 0}$	the set of nonnegative real numbers; similar for positive real numbers and for nonnegative and positive integers
x	a scalar
$\mathbf{x}$	a column matrix
$\mathbf{X}$	a matrix
$\mathbf{1}$	the identity matrix
$\mathbf{0}$	a zero matrix
$\mathcal{R}(\cdot)$	the column space of a matrix
$\mathrm{rk}(\cdot)$	the rank of a matrix
$\mathrm{tr}(\cdot)$	the trace of a matrix
$(\cdot)^\mathsf{T}$	the transpose of a matrix
$(\cdot)^{-1}$	the inverse of a matrix
$(\cdot)^\dagger$	the Moore-Penrose pseudoinverse of a matrix
$\bar{\sigma}(\cdot)$	the maximum singular value of a matrix
$\bar{\lambda}(\cdot)$	the maximum eigenvalue of a matrix

$\bar{\rho}(\cdot)$	the spectral radius of a matrix
$\otimes$	the Kronecker product
$\mathbf{X} = \mathbf{X}^\mathsf{T} > 0$	a symmetric positive definite matrix; similar for positive semidefinite, negative definite, and negative semidefinite matrices
$*$	the symmetric part of a matrix
$\mathrm{He}\{\cdot\}$	the sum of a matrix and its transpose
$\mathrm{diag}\{\cdot, \ldots, \cdot\}$	a block diagonal matrix
$\mathcal{G} : (\cdot) \to (\cdot)$	an operator
$\circ$	the composition of functions
$\times$	the Cartesian product of sets
$\|\cdot\|$	a norm
$\langle \cdot, \cdot \rangle$	an inner product
$\langle \cdot, \cdot \rangle_T$	a truncated inner product
ℓ_2	the inner product sequence space
ℓ_{2e}	the extended inner product sequence space
$\boldsymbol{\mathcal{G}} : \ell_{2e} \to \ell_{2e}$	a system
$\mathbf{G}(z)$	a transfer matrix
$\overset{\min}{\sim}$	a minimal state-space realization

Preface

This thesis explores applications of linear systems theory to the Koopman operator approximation problem. Chapters 4, 5, and 6 use linear systems theory to enhance Koopman operator approximation methods. Specifically, system structure and system properties like gain and asymptotic stability are taken into account when approximating the Koopman operator. Chapter 7 applies standard linear robust control methods to a population of Koopman models. Specifically, a robust Koopman observer is synthesized using a frequency-domain uncertainty model of a population of identified Koopman systems. Chapter 8 presents an open-source Koopman operator approximation library written, in part, to implement the contributions of Chapters 4–7. The following is a list of original contributions presented in this thesis, which were developed independently by the author under the supervision of Prof. James Richard Forbes.

- Chapter 4 [1–3]
  - A semidefinite program reformulation of extended dynamic mode decomposition (EDMD).
  - A semidefinite program reformulation of dynamic mode decomposition with control (DMDc).
- Chapter 5 [1, 2]
  - EDMD- and DMDc-based Koopman operator approximation methods that constrain the asymptotic stability of the Koopman matrix.
  - EDMD- and DMDc-based Koopman operator approximation methods that regularize the (weighted) $\mathcal{H}_\infty$ norm of the Koopman system.
- Chapter 6 [3]
  - An EDMD-based Koopman operator approximation method for closed-loop systems with known controllers.
- Chapter 7 [4, 5]

  - A robust nonlinear observer synthesis method for a population of Koopman models based on mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control.
  - The frequency-domain uncertainty characterization for a population of Koopman models considering six different uncertainty forms and nontrivial weighting functions.
  - An outlier detection method for Koopman models.
  - A publicly available dataset for quantifying manufacturing uncertainty in a population of several dozen motor drives.

- Chapter 8 [6]
  - An open-source Koopman operator approximation library that considers exogenous inputs and multiple training trajectories, allows the construction of lifting functions through composition, and supports standard cross-validation and hyperparameter optimization tools.

The optimization algorithm used in Chapter 7 to fit bounds to transfer function residuals was originally implemented by Jonathan Eid with enhancements made by Prof. James Richard Forbes.

Ce qui est simple est toujours faux. Ce qui ne l'est pas est inutilisable.

– Paul Valéry (1942), *Mauvaises pensées et autres*

Chapter 1

Introduction

Differential equations are an essential tool for modelling the physical world. Ordinary differential equations (ODEs) can be used to describe electric circuits, rigid-body dynamics, or chemical reaction rates, while the fundamental laws of electromagnetism, fluid dynamics, and heat transfer can be formulated as partial differential equations (PDEs). Despite the prevalence of differential equations in science and engineering, much emphasis in the field of machine learning is placed on learning time-independent functions. Most machine learning models of functions are black boxes, and are used primarily to predict output data given input data. While these methods can be applied to learning differential equations, they are not designed to account for the time series nature of the training data.

All sets of coupled time-invariant ODEs, including spatially discretized PDEs, can be written in first-order form [7, §C.3],

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

where $\mathbf{x}(0) = \mathbf{x}_0$ is the initial state. Equation (1.1) implies that static black-box machine learning tools can be used to learn differential equations if they treat samples of the state, $\mathbf{x}(t)$, as inputs and samples of its time derivative, $\dot{\mathbf{x}}(t)$, as outputs. While, for example, a neural network model of $\mathbf{f}(\cdot)$ could produce accurate trajectory predictions, no other useful information could be inferred from it. Scientists or engineers may be interested in analyzing the system's modes or stability, or they may want to design a controller for the system. For these purposes, a more interpretable model is desirable. Linear models provide all of these features, but they are unable to capture any nonlinearities in the system. A nonlinear black-box model could be numerically linearized, but that approximation may only be valid in a small region of the state space.

As it stands, the choice is between an opaque nonlinear model and an interpretable linear

model. The Koopman operator provides an interesting solution to this dilemma, since it can model nonlinear differential equations while retaining some of the interpretability of a linear model.

1.1 Background and Related Work

Using Koopman operator theory [8–11], a finite-dimensional nonlinear system can be rewritten as an infinite-dimensional linear system. This is accomplished by viewing the dynamics of a nonlinear system in terms of an infinite number of lifting functions, which are scalar functions of the system's state. The Koopman operator itself serves to advance the system's lifting functions in time. Practically working with an infinite-dimensional operator is intractable, so the Koopman operator is approximated as a finite-dimensional matrix, often using data-driven methods. While the Koopman operator was originally proposed almost one hundred years ago, recent theoretical results [9–11] paired with modern computational resources have led to it becoming a popular tool for identifying differential equations from data.

Thanks to its linearity, the Koopman operator yields a highly interpretable model. The spectrum of the Koopman operator can be analyzed, revealing stability properties, along with growth and decay rates. In the frequency domain, gain and phase can be assessed as functions of frequency. The linear Koopman representation of the nonlinear system also allows linear control design methods to be leveraged [12–19]. Koopman operator approximation methods are also closely related to classical system identification methods [12, 20], most notably subspace identification methods [21, §3]. This similarity means that well-established tools from system identification can be adapted to better approximate Koopman operators from data.

To approximate the Koopman operator, a finite set of lifting functions is first selected. In some situations, these lifting functions are inspired by the dynamics of the system [14, 15, 18], while in others, they are taken from a standard set of basis functions, like polynomials, sinusoids, or radial basis functions [16, 22, 23]. Lifting functions can also be chosen to approximate a given kernel [24–26]. Time delays are often incorporated into the lifted state as well [12, 16, 27]. However, finding a practical set of Koopman lifting functions for a given system is a significant challenge, both for theoretical and numerical reasons. The existence of useful sets of Koopman lifting functions is analyzed theoretically in [28, 29]. Once the chosen lifting functions are applied to the data, linear regression is used to approximate the Koopman matrix, typically with some form of regularization [16].

1.2 Outline and Contributions

This thesis explores how linear systems concepts can be leveraged in the Koopman operator approximation process. It also considers how Koopman operator models can be used in tandem with linear optimal control techniques to design nonlinear controllers or observers.

Chapter 2 lays out the mathematical preliminaries required to understand Koopman operator theory and optimal control. The preliminaries continue in Chapter 3, where Koopman operator theory, Koopman operator approximation methods, and related concepts are explained.

The contributions of this thesis begin in Chapter 4, where two Koopman operator approximation methods are reformulated as semidefinite programs. The following chapters use these reformulations as the basis for their contributions.

Chapter 5 considers how system properties, specifically asymptotic stability and system norms, can be used in the Koopman operator approximation process. Two Koopman operator approximation methods are proposed: one that constrains the asymptotic stability of the identified system and another that regularizes the $\mathcal{H}_\infty$ norm of the identified system. The benefits of these approaches are demonstrated experimentally on a soft robotic manipulator.

In Chapter 6, the identification of closed-loop systems is considered. Presented therein is a method to simultaneously identify Koopman models of a closed-loop system and the corresponding open-loop plant given knowledge of the controller. This approach is particularly useful for identifying unstable systems that are stabilized by a controller. Its advantages are demonstrated experimentally using a rotary inverted pendulum system.

Chapter 7 explores the potential of the Koopman operator to be used in robust control and estimation. By leveraging the linearity of the Koopman operator, uncertainty within a population of Koopman models is quantified in the frequency domain and a robust Koopman observer is synthesized using mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control. An experimental dataset collected from several dozen motor drives is used to validate the proposed approach. The resulting uncertainty model is accurate enough to detect outliers in the dataset.

The contributions of the previous chapters are implemented using `pykoop`, an open-source Koopman operator approximation library. This library allows lifting functions to be constructed through composition, parameterized, and optimized using standard software. Its software architecture and unique features are described in Chapter 8.

The thesis concludes in Chapter 9. While Section 1.1 provides a brief literature review of Koopman operator theory, approximation methods, and applications, each chapter includes an additional literature review to contextualize its contributions. The Preface contains a detailed list of the contributions found in this thesis.

Chapter 2

Mathematical Preliminaries

Summary

This chapter outlines the mathematical fundamentals required to understand Koopman operator theory, Koopman operator approximation methods, and optimal controller synthesis methods. In particular, relevant concepts from linear algebra, linear systems theory, and optimization are reviewed.

2.1 Introduction

This thesis considers how linear systems concepts can enhance the Koopman operator approximation process, and how linear optimal control tools can be paired with the Koopman operator to synthesize nonlinear controllers or observers.

Linear algebra concepts that are required throughout the thesis, namely matrix definiteness, matrix orthonormality, matrix decompositions, and matrix norms, are reviewed in Section 2.2. A review of signals and systems is found in Section 2.3, wherein norms of signals and systems, representations of linear systems, and stability are discussed. Optimization, which is the foundation of Koopman operator approximation and optimal controller synthesis, is reviewed in Section 2.4. Least-squares and regularization, which relate to Koopman operator approximation, are first discussed. Semidefinite programs (SDPs) and linear matrix inequalitys (LMIs), which are used to both approximate the Koopman operator and to synthesize optimal controllers, then are reviewed.

Concepts that are only relevant to one chapter of this thesis are reviewed at the beginning of their respective chapters.

2.2 Linear Algebra

This section discusses the linear algebra fundamentals used in the remainder of the preliminaries. Specifically, the definiteness of a matrix is defined, norms of matrices, and decompositions of matrices are reviewed.

2.2.1 Definiteness of a matrix

A symmetric matrix $\mathbf{A} = \mathbf{A}^\mathsf{T} \in \mathbb{R}^{n\times n}$ is called [30, §C.4.2][31, §1.3.1]

- *positive definite* if $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} > 0$, $\forall \mathbf{x} \neq \mathbf{0} \in \mathbb{R}^{n\times 1}$,
- *positive semidefinite* if $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \geq 0$, $\forall \mathbf{x} \neq \mathbf{0} \in \mathbb{R}^{n\times 1}$,
- *negative definite* if $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} < 0$, $\forall \mathbf{x} \neq \mathbf{0} \in \mathbb{R}^{n\times 1}$,
- *negative semidefinite* if $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} \leq 0$, $\forall \mathbf{x} \neq \mathbf{0} \in \mathbb{R}^{n\times 1}$, and
- *indefinite* otherwise.

Positive definite matrices are denoted $\mathbf{A} > 0$ and positive semidefinite matrices are denoted $\mathbf{A} \geq 0$. Negative (semi)definite matrices are denoted similarly. The notation $\mathbf{A} > \mathbf{B}$ expresses relative definiteness [31, §1.3.3], meaning that $\mathbf{A} - \mathbf{B} > 0$.

2.2.2 Orthonormal matrices

A square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is called *orthonormal* if [32, §3.4]

$$\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{1}. \tag{2.1}$$

Thus, for an orthonormal matrix, $\mathbf{A}^{-1} = \mathbf{A}^\mathsf{T}$.

A rectangular matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \neq n$, cannot be orthonormal, but its columns may still be orthonormal. In that case, $\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{1}$, so $\mathbf{A}^\mathsf{T}$ is only a left inverse. The matrix $\mathbf{A}\mathbf{A}^\mathsf{T}$ is the orthogonal projection matrix onto $\mathcal{R}(\mathbf{A})$, the column space of $\mathbf{A}$ [32, §3.4]. Transposing these relationships reveals the properties of a rectangular matrix with orthonormal rows.

2.2.3 Matrix decompositions

The *Cholesky factorization* of a symmetric positive definite matrix $\mathbf{A} = \mathbf{A}^\mathsf{T} > 0 \in \mathbb{R}^{n\times n}$ is [30, §C.3.2]

$$\mathbf{A} = \mathbf{L}\mathbf{L}^\mathsf{T}, \tag{2.2}$$

where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is lower triangular.

The *eigendecomposition* of a diagonalizable square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}, \tag{2.3}$$

where $\mathbf{\Lambda} \in \mathbb{C}^{n \times n}$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$ and $\mathbf{V} \in \mathbb{C}^{n \times n}$ is a square matrix whose columns are the eigenvectors of $\mathbf{A}$ [32, §5.2]. The magnitude of the largest eigenvalue of a matrix is called the *spectral radius* and is denoted $\bar{\rho}(\cdot)$.

The *singular value decomposition* of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is

$$\mathbf{A} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Z}^{\mathsf{T}}, \tag{2.4}$$

where $\mathbf{Q} \in \mathbb{R}^{m \times m}$ and $\mathbf{Z} \in \mathbb{R}^{n \times n}$ are orthonormal matrices with the left and right singular vectors of $\mathbf{A}$ as their columns, and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a rectangular diagonal matrix containing the singular values of $\mathbf{A}$ on its diagonal, which are nonnegative real numbers [30, §A.5.4][33, §1.1]. The maximum singular value of a matrix is denoted $\bar{\sigma}(\cdot)$.

The number of nonzero singular values of a matrix is equal to its rank [32, §6.3]. In the *economy singular value decomposition*, zero singular values and their corresponding left and right singular vectors are removed, resulting in $\mathbf{Q} \in \mathbb{R}^{m \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and $\mathbf{Z} \in \mathbb{R}^{n \times r}$, where $r = \mathrm{rk}(\mathbf{A})$ [33, §1.1]. In the economy SVD, $\mathbf{Q}^{\mathsf{T}}\mathbf{Q} = \mathbf{1}$ and $\mathbf{Z}^{\mathsf{T}}\mathbf{Z} = \mathbf{1}$, but $\mathbf{Q}\mathbf{Q}^{\mathsf{T}}$ and $\mathbf{Z}\mathbf{Z}^{\mathsf{T}}$ are orthogonal projection matrices onto $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{A}^{\mathsf{T}})$ respectively [30, §A.5.4].

In the *truncated singular value decomposition*, only the r largest singular values and vectors are retained, leading to a lower rank approximation of $\mathbf{A}$. Algorithms like optimal hard thresholding [34] are sometimes used to choose r.

2.2.4 Matrix norms

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. The *Frobenius norm* of a matrix is defined as [30, §A.1.1]

$$\|\mathbf{A}\|_{\mathsf{F}} = \sqrt{\mathrm{tr}(\mathbf{A}^{\mathsf{T}}\mathbf{A})}. \tag{2.5}$$

The *spectral norm* of a matrix is defined as [30, §4.6.3]

$$\|\mathbf{A}\|_2 = \sqrt{\bar{\lambda}\left(\mathbf{A}^{\mathsf{T}}\mathbf{A}\right)} \tag{2.6}$$

$$= \bar{\sigma}(\mathbf{A}). \tag{2.7}$$

The *nuclear norm* of a matrix is defined as [30, §A.1.6]

$$\|\mathbf{A}\|_* = \operatorname{tr}\left(\sqrt{\mathbf{A}^\mathsf{T}\mathbf{A}}\right) \tag{2.8}$$

$$= \sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{A}), \tag{2.9}$$

where $\sigma_i(\cdot)$ is the i^{th} singular value of $\mathbf{A}$. The nuclear norm is a convex approximation of the rank function [35].

2.3 Signals and Systems

This section reviews discrete-time signals and systems. Specifically, it covers inner products and norms of signals, along with state-space and transfer matrix representations of systems, stability theory, and system norms.

2.3.1 Signals

A discrete-time signal is a sequence, which can be expressed as a function $\mathbf{y} : \mathbb{Z}_{\geq 0} \to \mathbb{R}^{n\times 1}$ [31, §1.2].

The ℓ_2 inner product of two signals is [36, §4.2]

$$\langle \mathbf{u}, \mathbf{y} \rangle = \sum_{k=0}^{\infty} \mathbf{u}_k^\mathsf{T} \mathbf{y}_k, \tag{2.10}$$

while the truncated ℓ_2 inner product of two signals is [37, §8.2]

$$\langle \mathbf{u}, \mathbf{y} \rangle_T = \sum_{k=0}^{T} \mathbf{u}_k^\mathsf{T} \mathbf{y}_k. \tag{2.11}$$

The ℓ_2 and truncated ℓ_2 inner products induce the ℓ_2 and truncated ℓ_2 norms. The ℓ_2 norm of a signal is [38, §B.1.1]

$$\|\mathbf{y}\|_2 = \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle} = \sqrt{\sum_{k=0}^{\infty} \mathbf{y}_k^\mathsf{T} \mathbf{y}_k}, \tag{2.12}$$

while the truncated ℓ_2 norm of a signal, called the ℓ_{2T} norm for a given $T \in \mathbb{Z}_{\geq 0}$, is [38,

§B.1.1]

$$\|\mathbf{y}\|_{2T} = \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle_T} = \sqrt{\sum_{k=0}^{T} \mathbf{y}_k^\mathsf{T} \mathbf{y}_k}. \tag{2.13}$$

The ℓ_2 norm defines the inner product sequence space [38, §B.1.1]

$$\ell_2 = \left\{ \mathbf{y} \,\middle|\, \|\mathbf{y}\|_2^2 < \infty \right\}, \tag{2.14}$$

which consists of all square summable signals. The truncated ℓ_2 norm defines the extended inner product sequence space [38, §B.1.1]

$$\ell_{2e} = \left\{ \mathbf{y} \,\middle|\, \|\mathbf{y}\|_{2T}^2 < \infty, \forall T \in \mathbb{Z}_{\geq 0} \right\}. \tag{2.15}$$

A notable feature of ℓ_{2e} is that it contains signals like $y_k = \sin(k)$, which are bounded but not square summable.

Vector spaces equipped with an inner product are called *inner product spaces* [36, §4.2]. Inner product spaces that are complete with respect to the norm induced by their inner product are called *Hilbert spaces*. Both ℓ_2 and ℓ_{2e} are examples of Hilbert spaces [38, §B.1.1].

2.3.2 Linear systems

A system is a mapping from signals to signals. In the time-invariant case, a system is denoted $\boldsymbol{\mathcal{G}} : \ell_{2e} \to \ell_{2e}$. For linear time-invariant (LTI) systems, $\boldsymbol{\mathcal{G}}$ is a linear operator. That is,

$$\boldsymbol{\mathcal{G}} \left(\alpha \mathbf{u} + \beta \mathbf{v} \right) = \alpha \boldsymbol{\mathcal{G}} \mathbf{u} + \beta \boldsymbol{\mathcal{G}} \mathbf{v}. \tag{2.16}$$

The *state-space* representation of a causal LTI system is

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{2.17}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k, \tag{2.18}$$

where $\mathbf{x}_0 \in \mathbb{R}^{n \times 1}$ is the initial state. The output of the system given an input and initial condition is [39, §3.4]

$$\mathbf{y}_k = \mathbf{C}\mathbf{A}^k \mathbf{x}_0 + \mathbf{D}\mathbf{u}_k + \sum_{i=0}^{k-1} \mathbf{C}\mathbf{A}^{k-1-i} \mathbf{B}\mathbf{u}_i, \tag{2.19}$$

where

$$\mathbf{y}_k^{\text{free}} = \mathbf{C}\mathbf{A}^k \mathbf{x}_0 \tag{2.20}$$

is called the *free response* and

$$\mathbf{y}_k^{\text{forced}} = \mathbf{D}\mathbf{u}_k + \sum_{i=0}^{k-1} \mathbf{C}\mathbf{A}^{k-1-i}\mathbf{B}\mathbf{u}_i \tag{2.21}$$

is called the *forced response*.

State-space representations of LTI systems are not unique [40, §3.4.3]. Three key properties of state-space realizations, loosely described below, are controllability, observability, and minimality. The pair $(\mathbf{A}, \mathbf{B})$ is *controllable* if, for every initial state, there exists an input that brings the state to zero in finite time [40, §5.2.1]. The pair $(\mathbf{A}, \mathbf{C})$ is *observable* if the initial state can always be uniquely recovered from measured inputs and outputs [40, §5.2.2]. A state-space realization is *minimal* if and only if $(\mathbf{A}, \mathbf{B})$ is controllable and $(\mathbf{A}, \mathbf{C})$ is observable [40, §8.1]. Minimal state-space realizations of LTI systems capture their input-output relations with the minimum number of states [40, §8.3]. There are an infinite number of minimal state-space realizations for a given LTI system, which are related to each other through a similarity transform [40, §8.3]. The linear operator $\boldsymbol{\mathcal{G}}$ is related to the state-space matrices through

$$\boldsymbol{\mathcal{G}} \overset{\text{min}}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{2.22}$$

where $\overset{\text{min}}{\sim}$ denotes a minimal state-space realization [38, §3.2.1].

Consider the situation where $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{u}_k = \delta_k \mathbf{v}$, where

$$\delta_k = \begin{cases} 0, & k \neq 0 \\ 1, & k = 0 \end{cases} \tag{2.23}$$

is the Kronecker delta function and $\mathbf{v}$ is a constant column matrix. In this situation, the forced response is

$$\mathbf{y}_k = \left(\mathbf{D}\delta_k + \mathbf{C}\mathbf{A}^{k-1}\mathbf{B}\right)\mathbf{v}, \tag{2.24}$$

which is called the *impulse response*. The matrix

$$\mathbf{G}_k = \mathbf{D}\delta_k + \mathbf{C}\mathbf{A}^{k-1}\mathbf{B} \tag{2.25}$$

is called the *impulse response function* of the system.

Another representation of an LTI system, which need not be causal, is the *transfer matrix*.

The transfer matrix relates the z-transform of the system's inputs and outputs via

$$\mathbf{y}(z) = \mathbf{G}(z)\mathbf{u}(z), \tag{2.26}$$

where $z \in \mathbb{C}$ [40, §3.5.2]. Evaluating the transfer matrix at $z = e^{j\theta}$, where $\theta = 2\pi\Delta t f$ is the discrete-time frequency, Δt is the sampling period, and f is the continuous-time frequency, yields the frequency response of the system. The transfer matrix is the z-transform of the impulse response function of the system, and can be computed from its state-space matrices via [40, §3.7]

$$\mathbf{G}(z) = \mathbf{D} + \mathbf{C}(z\mathbf{1} - \mathbf{A})^{-1}\mathbf{B}. \tag{2.27}$$

The maximum singular value of the transfer matrix measures the maximum input-output gain at each frequency. When the sampling period is known, discrete-time transfer matrices may be shown in figures as $\mathbf{G}(f)$ for brevity.

2.3.3 Stability

Consider the autonomous nonlinear difference equation

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k), \tag{2.28}$$

and let $\boldsymbol{\phi}_k$ denote its solution for an initial condition $\mathbf{x}_0$. Without loss of generality, consider an equilibrium point $\bar{\mathbf{x}} = \mathbf{0}$. The equilibrium point is called *Lyapunov stable* if, for every $\varepsilon > 0$, there exists a $\delta(\varepsilon) > 0$ such that $\|\boldsymbol{\phi}_k\| < \varepsilon$ for all $k \geq 0$ when $\|\mathbf{x}_0\| < \delta$ [40, §4.8.1]. The equilibrium point is called *asymptotically stable* (AS) if it is Lyapunov stable and there exists an $\eta > 0$ such that

$$\lim_{k\to\infty} \|\boldsymbol{\phi}_k\| = 0 \tag{2.29}$$

when $\|\mathbf{x}_0\| < \eta$ [40, §4.8.1]. The equilibrium point is called *unstable* if it is not Lyapunov stable. Loosely speaking, trajectories stay near a Lyapunov stable equilibrium point, but converge to an AS equilibrium point. Trajectories diverge from an unstable equilibrium point.

Next, consider an autonomous linear difference equation,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k, \tag{2.30}$$

where $\mathbf{x}_0$ is the initial condition. Also consider the discrete-time Lyapunov equation [40, §4.8.3],

$$\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{Q} = \mathbf{0}, \tag{2.31}$$

where $\mathbf{P} = \mathbf{P}^\mathsf{T} > 0$. If there exists a matrix $\mathbf{Q} = \mathbf{Q}^\mathsf{T} \geq 0$ that satisfies (2.31), then the equilibrium point $\bar{\mathbf{x}} = \mathbf{0}$ of (2.30) is Lyapunov stable. If there exists a matrix $\mathbf{Q} = \mathbf{Q}^\mathsf{T} > 0$ that satisfies (2.31), then the equilibrium point is AS. The equilibrium point $\bar{\mathbf{x}} = \mathbf{0}$ is also AS if and only if

$$\lim_{k\to\infty} \|\mathbf{A}^k\| = 0, \tag{2.32}$$

where $\|\cdot\|$ can be any matrix norm [40, §4.8.2]. Another equivalent condition for asymptotic stability is that the eigenvalues of $\mathbf{A}$ are contained strictly within the unit circle [40, §4.8.2]. That is,

$$\bar{\rho}(\mathbf{A}) < 1, \tag{2.33}$$

where $\bar{\rho}(\cdot)$ denotes the spectral radius of a matrix. A matrix whose spectral radius is strictly less than one is called *Schur* [40, §4.8.2].

Note that a linear system is called Lyapunov stable if all its equilibrium points are Lyapunov stable. To be called AS, a linear system can only have one equilibrium point, $\bar{\mathbf{x}} = \mathbf{0}$, which must be AS. When discussing the stability of a nonlinear system, the equilibrium point must be specified.

2.3.4 Norms of linear systems

Like signals, the size of a linear system can be quantified with a norm. Both norms discussed in this section are related to the notion of gain, that is, the amplification of a signal through a system.

The $\mathcal{H}_\infty$ norm of $\boldsymbol{\mathcal{G}}$ is the worst-case gain from $\|\mathbf{u}\|_2$ to $\|\mathbf{y}\|_2 = \|\boldsymbol{\mathcal{G}}\mathbf{u}\|_2$. That is [38, §B.1.1],

$$\|\boldsymbol{\mathcal{G}}\|_\infty = \sup_{\mathbf{u}\in\ell_2, \mathbf{u}\neq\mathbf{0}} \frac{\|\boldsymbol{\mathcal{G}}\mathbf{u}\|_2}{\|\mathbf{u}\|_2}. \tag{2.34}$$

As such, the $\mathcal{H}_\infty$ norm is the system norm induced by the ℓ_2 signal norm [38, §3.3.2]. In the frequency domain, this definition is equivalent to [38, §B.1.1]

$$\|\boldsymbol{\mathcal{G}}\|_\infty = \sup_{\theta\in(-\pi,\pi]} \bar{\sigma}\left(\mathbf{G}(e^{j\theta})\right), \tag{2.35}$$

where $\bar{\sigma}(\cdot)$ denotes the maximum singular value of a matrix. The $\mathcal{H}_\infty$ norm of a system can be viewed as the peak magnitude of its frequency response.

The $\mathcal{H}_2$ norm is

$$\|\boldsymbol{\mathcal{G}}\|_2 = \sqrt{\sum_{k=0}^{\infty} \mathrm{tr}\big(\mathbf{G}_k^\mathsf{T}\mathbf{G}_k\big)} = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{tr}\Big(\mathbf{G}(e^{j\theta})^\mathsf{H}\mathbf{G}(e^{j\theta})\Big)\mathrm{d}\theta}, \tag{2.36}$$

where $\mathbf{G}_k$ is the impulse response function of $\boldsymbol{\mathcal{G}}$ and $\mathbf{G}(e^{j\theta})$ is the transfer matrix representation of $\boldsymbol{\mathcal{G}}$ [41, §4.4][42]. Note that, unlike in the continuous-time case, it is possible to compute the $\mathcal{H}_2$ norm of a discrete-time system where $\mathbf{D} \neq \mathbf{0}$ [31, §4.3.3][42]. The $\mathcal{H}_2$ norm can be viewed as the expected root-mean-squared (RMS) output of a system when the input is unit variance white noise [38, §3.3.3][36, §5.7][42]. In the frequency domain, the square of the $\mathcal{H}_2$ norm can be interpreted as the area under the curve of the sum of the squared singular values of the transfer matrix [43, §2.3.3].

2.4 Optimization

Optimization problems are at the core of the Koopman operator approximation and optimal control techniques discussed in this thesis. In this section, two types of convex optimization problems are reviewed: regularized least-squares problems and SDPs. LMIs that relate to LTI system properties are also reviewed.

An *optimization problem* has the form

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{2.37}$$

$$\text{s.t. } g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, p, \tag{2.38}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, q, \tag{2.39}$$

where $\mathbf{x} \in \mathbb{R}^{n\times 1}$ is the *optimization variable*, $f : \mathbb{R}^{n\times 1} \to \mathbb{R}$ is the *objective function* to be minimized, the functions $g_i : \mathbb{R}^{n\times 1} \to \mathbb{R}$ are the *inequality constraint functions*, and the functions $h_j : \mathbb{R}^{n\times 1} \to \mathbb{R}$ are the *equality constraint functions* [30, §4.1.1]. The value $\mathbf{x}^*$ is *optimal* if it results in the lowest possible objective function value while satisfying the constraints [30, §4.1.1].

In cases where many optimization variables are required, the subscript is omitted and the notation

$$\min \; J(\mathbf{x}_1, \ldots, \mathbf{x}_r; \alpha_1, \ldots, \alpha_s) \tag{2.40}$$

is used, where the cost function is denoted $J(\cdot\,;\cdot)$, optimization variables are denoted $\mathbf{x}_r$, and hyperparameters that are constant with respect to the optimization problem are denoted α_s.

2.4.1 Convex optimization

A function $f : \mathbb{R}^{n\times 1} \to \mathbb{R}$ is called *convex* if it satisfies the inequality

$$f(\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2) \leq \alpha_1 f(\mathbf{x}_1) + \alpha_2 f(\mathbf{x}_2) \tag{2.41}$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{n\times 1}$ and $\alpha_1, \alpha_2 \in \mathbb{R}_{\geq 0}$ where $\alpha_1 + \alpha_2 = 1$ [30, §1.1]. An optimization problem is called convex if its objective function and inequality constraint functions are convex and its equality constraint functions are affine [30, §4.2.1]. Specifically, a *convex optimization problem* has the form

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{2.42}$$

$$\text{s.t. } g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, p, \tag{2.43}$$

$$\mathbf{a}_j^{\mathsf{T}}\mathbf{x} - b_j = 0, \quad j = 1, \ldots, q, \tag{2.44}$$

where $f(\cdot)$ and $g_1(\cdot), \ldots, g_p(\cdot)$ are convex [30, §4.2.1]. Convex optimization problems are significant because, if they are feasible, they have a unique global and local minimum [30, §4.2.2].

2.4.2 Least-squares problems

Least-squares optimization problems are a specific subset of convex optimization problems of the form [30, §1.2.1]

$$\min_{\mathbf{x}} \ f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{2.45}$$

The more general matrix least-squares problem is

$$\min_{\mathbf{X}} \ f(\mathbf{X}) = \|\mathbf{A}\mathbf{X} - \mathbf{B}\|_{\mathsf{F}}^2. \tag{2.46}$$

Least-squares problems have the analytical solution [30, §1.2.1]

$$\mathbf{X} = \mathbf{A}^{\dagger}\mathbf{B}, \tag{2.47}$$

where $(\cdot)^{\dagger}$ denotes the *Moore-Penrose pseudoinverse*, which is defined as [30, §A.5.4]

$$\mathbf{A}^{\dagger} = \mathbf{Z}\boldsymbol{\Sigma}^{\dagger}\mathbf{Q}^{\mathsf{T}}, \tag{2.48}$$

where $\mathbf{A} = \mathbf{Q}\boldsymbol{\Sigma}\mathbf{Z}^{\mathsf{T}}$ is the singular value decomposition (SVD) of $\mathbf{A}$ and $\boldsymbol{\Sigma}^{\dagger}$ is computed by inverting the nonzero entries on the diagonal of $\boldsymbol{\Sigma}$. When $\mathrm{rk}(\mathbf{A}) = n$ [30, §A.5.4],

$$\mathbf{A}^{\dagger} = \left(\mathbf{A}^{\mathsf{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathsf{T}}, \tag{2.49}$$

while when $\mathrm{rk}(\mathbf{A}) = m$ [30, §A.5.4],

$$\mathbf{A}^{\dagger} = \mathbf{A}^{\mathsf{T}}\left(\mathbf{A}\mathbf{A}^{\mathsf{T}}\right)^{-1}. \tag{2.50}$$

When solving least-squares problems, regularization is often required to obtain well-conditioned solutions [30, §6.3.2] and prevent overfitting [33, §4]. Regularization involves penalizing the size of the optimization variable in the objective function of the optimization problem. The extent to which the optimization variable is penalized is controlled by a scalar regularization coefficient. For example, *Tikhonov regularization* penalizes the squared Frobenius norm of the optimization variable, leading to the least-squares problem [30, §6.3.2]

$$\min_{\mathbf{X}} \; f(\mathbf{X};\alpha) = \|\mathbf{A}\mathbf{X}-\mathbf{B}\|_{\mathsf{F}}^2 + \alpha\|\mathbf{X}\|_{\mathsf{F}}^2. \tag{2.51}$$

The Tikhonov-regularized optimization problem has the closed-form solution

$$\mathbf{X} = \left(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \alpha\mathbf{1}\right)^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{B}, \tag{2.52}$$

where the matrix being inverted is always full rank [30, §6.3.2]. Other norms or measures of the optimization variable can be used as regularizers, however they may affect the convexity of the problem.

2.4.3 Linear matrix inequalities and semidefinite programs

A *linear matrix inequality* (LMI) in the variable $\mathbf{x} \in \mathbb{R}^{m\times 1}$ is an expression of the form [31, §1.3.2]

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{m} x_i\mathbf{F}_i \leq 0, \tag{2.53}$$

where x_i is the i^{th} entry of $\mathbf{x}$ and $\mathbf{F}_i = \mathbf{F}_i^{\mathsf{T}}$, $i = 0,\ldots,m$. LMIs can also be rewritten in terms of matrix variables $\mathbf{X}_i \in \mathbb{R}^{p_i\times q_i}$, $i = 1,\ldots,r$. That is [31, §1.3.2],

$$\mathbf{F}(\mathbf{X}_1,\ldots,\mathbf{X}_r) = \mathbf{F}_0 + \sum_{i=1}^{r} \mathrm{He}\{\mathbf{G}_i\mathbf{X}_i\mathbf{H}_i\} \leq 0, \tag{2.54}$$

where $\mathbf{F}_0 = \mathbf{F}_0^{\mathsf{T}} \in \mathbb{R}^{n\times n}$, $\mathbf{G}_i \in \mathbb{R}^{n\times p_i}$, $\mathbf{H}_i \in \mathbb{R}^{q_i\times n}$, and

$$\mathrm{He}\{\cdot\} = (\cdot) + (\cdot)^{\mathsf{T}}. \tag{2.55}$$

The entries of $\mathbf{X}_i$ can be thought of as the m entries of $\mathbf{x}$ in (2.53), where $m = \sum_{i=1}^{r} p_i q_i$. Note that multiple LMIs can always be concatenated into a single LMI by placing them in a block diagonal matrix [31, §1.3.5].

A *semidefinite program* (SDP) is an optimization problem of the form [31, §1.4]

$$\min_{\mathbf{x}} \mathbf{c}^{\mathsf{T}}\mathbf{x} \tag{2.56}$$

$$\text{s.t. } \mathbf{F}(\mathbf{x}) \leq 0, \tag{2.57}$$

where $\mathbf{c} \in \mathbb{R}^{m\times 1}$ is constant. In other words, a semidefinite program consists of a linear objective function with any number of LMI constraints. Since LMIs are convex [31, §1.3.6], SDPs are also convex. In practice, LMI constraints are written in the matrix form of (2.54), and are translated to the standard form of (2.53) by LMI parsing software like `PICOS` [44] or `CVXPY` [45]. The parsed semidefinite programs can then be solved with specialized solvers like `CVXOPT` [46] and `MOSEK` [47]. A comprehensive list of LMI parsers and SDP solvers can be found in [31, §1.5].

A matrix inequality that contains a product of two unknowns is called a *bilinear matrix inequality* (BMI). BMIs are not convex, and optimization problems containing them can be difficult to solve [48].

2.4.4 Linear matrix inequality properties

This section contains useful properties and identities for manipulating LMIs. Namely, the effect of congruence transformations on LMIs and the Schur complement are presented. The symmetric entries of a matrix are denoted with $*$.

2.4.4.1 Strictness

SDP solvers only work with nonstrict LMIs. To specify a strict LMI, a tolerance $0 < \varepsilon \ll 1$ is specified in the form

$$\mathbf{Q} \geq \varepsilon\mathbf{1} \tag{2.58}$$

or

$$\mathbf{Q} \leq -\varepsilon\mathbf{1}. \tag{2.59}$$

This tolerance should be larger than the optimality and feasibility tolerances given to the solver.

2.4.4.2 Congruence transformations

The definiteness of a matrix is unchanged by a congruence transformation. As such, congruence transformations are a useful tool in reformulating optimization problems as SDPs. Specifically, consider the matrices $\mathbf{Q} = \mathbf{Q}^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ and $\mathbf{W} \in \mathbb{R}^{n\times n}$, where $\operatorname{rk}(\mathbf{W}) = n$. The

LMI

$$\mathbf{Q} < 0 \tag{2.60}$$

is satisfied if and only if [31, §2.3]

$$\mathbf{W}\mathbf{Q}\mathbf{W}^{\mathsf{T}} < 0. \tag{2.61}$$

2.4.4.3 Schur complement

The *Schur complement* is also an essential tool when formulating SDPs. Consider the matrices $\mathbf{Q} = \mathbf{Q}^{\mathsf{T}} \in \mathbb{R}^{n\times n}$, $\mathbf{S} \in \mathbb{R}^{n\times m}$, and $\mathbf{R} = \mathbf{R}^{\mathsf{T}} \in \mathbb{R}^{m\times m}$. Through the Schur complement, the following are equivalent [31, §2.4.1].

1. $\begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ * & \mathbf{R} \end{bmatrix} < 0.$
2. $\mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}^{\mathsf{T}} < 0,\ \mathbf{R} < 0.$
3. $\mathbf{R} - \mathbf{S}^{\mathsf{T}}\mathbf{Q}^{-1}\mathbf{S} < 0,\ \mathbf{Q} < 0.$

2.4.5 Linear matrix inequalities in linear systems

LMIs can be used to describe LTI system properties in terms of their state-space representations. As such, abstract optimization problems written in terms of LTI systems can be rewritten as SDPs in terms of state-space matrices using LMIs. In this section, LMIs relating to Lyapunov stability, asymptotic stability, the $\mathcal{H}_\infty$ norm, and the $\mathcal{H}_2$ norm are presented.

2.4.5.1 Lyapunov and asymptotic stability

Lyapunov and asymptotic stability conditions for LTI systems can be expressed as LMIs. These LMIs are derived from (2.31), the Lyapunov equation, which is presented in Section 2.3.3.

Consider the discrete-time LTI system

$$\boldsymbol{\mathcal{G}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{2.62}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$. The system $\boldsymbol{\mathcal{G}}$ is Lyapunov stable if and only if there exists a matrix $\mathbf{P} = \mathbf{P}^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ such that [31, §4.1.3]

$$\mathbf{P} > 0, \tag{2.63}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{P}\mathbf{A} - \mathbf{P} \leq 0. \tag{2.64}$$

The system $\boldsymbol{\mathcal{G}}$ is AS if and only if there exists a matrix $\mathbf{P} = \mathbf{P}^{\mathsf{T}} \in \mathbb{R}^{n \times n}$ such that [31, §4.1.4]

$$\mathbf{P} > 0, \tag{2.65}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{P}\mathbf{A} - \mathbf{P} < 0. \tag{2.66}$$

2.4.5.2 $\mathcal{H}_\infty$ norm

A bound on the $\mathcal{H}_\infty$ norm of a discrete-time LTI system

$$\boldsymbol{\mathcal{G}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{2.67}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, can be computed using LMIs. The inequality $\|\boldsymbol{\mathcal{G}}\|_\infty < \gamma$ holds if and only if there exists a $\mathbf{P} = \mathbf{P}^{\mathsf{T}} \in \mathbb{R}^{n \times n}$ such that [31, §4.2.2]

$$\mathbf{P} > 0, \tag{2.68}$$

$$\begin{bmatrix} \mathbf{P} & \mathbf{A}\mathbf{P} & \mathbf{B} & \mathbf{0} \\ * & \mathbf{P} & \mathbf{0} & \mathbf{P}\mathbf{C}^{\mathsf{T}} \\ * & * & \gamma\mathbf{1} & \mathbf{D}^{\mathsf{T}} \\ * & * & * & \gamma\mathbf{1} \end{bmatrix} > 0. \tag{2.69}$$

A variety of LMIs equivalent to (2.68) and (2.69) can be found in [31, §4.2.2].

2.4.5.3 $\mathcal{H}_2$ norm

A bound on the $\mathcal{H}_2$ norm of a discrete-time LTI system

$$\boldsymbol{\mathcal{G}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{2.70}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, and where $\mathbf{A}$ is Schur, can be computed using LMIs. The inequality $\|\boldsymbol{\mathcal{G}}\|_2 < \mu$ holds if and only if there exists a matrix $\mathbf{P} = \mathbf{P}^{\mathsf{T}} \in \mathbb{R}^{n \times n}$ such that [31, §4.3.3]

$$\mathbf{P} > 0, \tag{2.71}$$

$$\mathbf{A}^{\mathsf{T}}\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{C}^{\mathsf{T}}\mathbf{C} < 0, \tag{2.72}$$

$$\operatorname{tr}\big(\mathbf{B}^{\mathsf{T}}\mathbf{P}\mathbf{B} + \mathbf{D}^{\mathsf{T}}\mathbf{D}\big) < \mu^2. \tag{2.73}$$

A variety of LMIs equivalent to (2.71)–(2.73) are presented in [31, §4.3.3].

Chapter 3

Koopman Operator Theory

Summary

This chapter provides an overview of Koopman operator theory, data-driven Koopman operator approximation methods, and common choices of Koopman lifting functions.

3.1 Introduction

As discussed in Chapter 1, the Koopman operator allows nonlinear systems to be expressed as infinite-dimensional linear systems by viewing their dynamics in terms of an infinite set of nonlinear lifting functions. The linearity of the Koopman operator means that tools from linear systems theory can be applied to Koopman models with little modification, either for analysis or for control design. The Koopman operator can be approximated from data by selecting a finite set of nonlinear lifting functions, applying them to the data, and solving a regression problem in the lifted space.

The Koopman operator is formally introduced in Section 3.2, and its extension to nonautonomous systems is discussed in Section 3.3. Koopman operator approximation techniques, including extended dynamic mode decomposition (EDMD), dynamic mode decomposition (DMD), and dynamic mode decomposition with control (DMDc), are outlined in Section 3.4. A selection of common Koopman lifting functions is presented in Section 3.5. Additionally, the difference between local and global Koopman prediction error is discussed in Section 3.6, and the Koopman system, which is the linear system defined by the Koopman matrix, is formally defined in Section 3.7.

3.2 The Koopman Operator

Consider the nonlinear difference equation

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k), \tag{3.1}$$

where $\mathbf{x}_k \in \mathcal{M}$ evolves on a manifold $\mathcal{M} \subseteq \mathbb{R}^{m\times 1}$. Also consider an infinite number of scalar-valued *lifting functions*, $\psi : \mathcal{M} \to \mathbb{R}$. The *Koopman operator*, $\mathcal{U} : \mathcal{H} \to \mathcal{H}$, is a linear operator that composes all lifting functions $\psi \in \mathcal{H}$ with $\mathbf{f}(\cdot)$, thereby advancing them in time by one step. That is [49, §3.2],

$$(\mathcal{U}\psi)(\cdot) = (\psi \circ \mathbf{f})(\cdot). \tag{3.2}$$

When the set of lifting functions, $\mathcal{H}$, is a linear vector space, the Koopman operator is a linear operator [10, §2]. That is, if $\alpha_1\psi_1 + \alpha_2\psi_2 \in \mathcal{H}$ for $\alpha_1, \alpha_2 \in \mathbb{R}$ and $\psi_1, \psi_2 \in \mathcal{H}$, then [21, §2.1]

$$\begin{aligned}
\mathcal{U}(\alpha_1\psi_1 + \alpha_2\psi_2) &= (\alpha_1\psi_1 + \alpha_2\psi_2) \circ \mathbf{f} && (3.3)\\
&= \alpha_1(\psi_1 \circ \mathbf{f}) + \alpha_2(\psi_2 \circ \mathbf{f}) && (3.4)\\
&= \alpha_1(\mathcal{U}\psi_1) + \alpha_2(\mathcal{U}\psi_2). && (3.5)
\end{aligned}$$

Sometimes the vector space $\mathcal{H}$ is explicitly chosen to be a Banach space, like the space of continuous functions, or a Hilbert space, like the space of square-integrable functions [10, 21]. Other times, $\mathcal{H}$ is implicitly defined by the set of lifting functions chosen when approximating the Koopman operator from data. The vector space $\mathcal{H}$ is often assumed to be a Banach space when discussing the spectrum of the Koopman operator [10, §3.1][21, §4.4].

When evaluated at $\mathbf{x}_k$, (3.2) is

$$\begin{aligned}
(\mathcal{U}\psi)(\mathbf{x}_k) &= (\psi \circ \mathbf{f})(\mathbf{x}_k) && (3.6)\\
&= \psi(\mathbf{f}(\mathbf{x}_k)) && (3.7)\\
&= \psi(\mathbf{x}_{k+1}). && (3.8)
\end{aligned}$$

Using the Koopman operator, the dynamics of (3.1) may then be rewritten linearly in terms of ψ as

$$\psi(\mathbf{x}_{k+1}) = (\mathcal{U}\psi)(\mathbf{x}_k). \tag{3.9}$$

A finite-dimensional approximation of (3.9) is

$$\boldsymbol{\psi}(\mathbf{x}_{k+1}) = \mathbf{U}\boldsymbol{\psi}(\mathbf{x}_k) + \boldsymbol{\epsilon}_k, \tag{3.10}$$

where $\boldsymbol{\psi} : \mathcal{M} \to \mathbb{R}^{p\times 1}$ is the *vector-valued lifting function*, $\mathbf{U} \in \mathbb{R}^{p\times p}$ is the *Koopman matrix*, and $\boldsymbol{\epsilon}_k$ is the residual error. The relationship between the state-space, lifted space, and approximate lifted space is depicted in Figure 3.1.

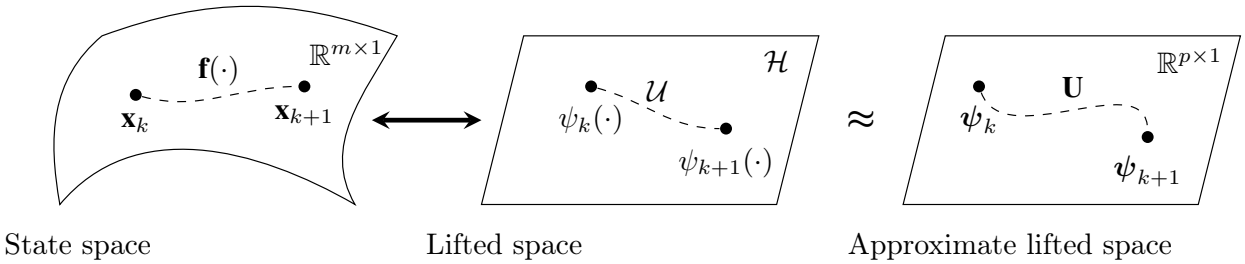


Figure 3.1: The Koopman representation of a system.

3.3 The Koopman Operator with Inputs

The definition of the Koopman operator can be modified to accommodate nonlinear difference equations with exogenous inputs. Consider the difference equation

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{3.11}$$

where the state is $\mathbf{x}_k \in \mathcal{M} \subseteq \mathbb{R}^{m\times 1}$ and the input is $\mathbf{u}_k \in \mathcal{N} \subseteq \mathbb{R}^{n\times 1}$. The lifting functions are now $\psi : \mathcal{M} \times \mathcal{N} \to \mathbb{R}$ and the Koopman operator $\mathcal{U} : \mathcal{H} \to \mathcal{H}$ now satisfies

$$(\mathcal{U}\psi)(\mathbf{x}_k, \mathbf{u}_k) = \psi(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \star), \tag{3.12}$$

where $\star = \mathbf{u}_k$ if the input has state-dependent dynamics, or $\star = \mathbf{0}$ if the input has no dynamics [50][49, §6.5]. Let the vector-valued lifting function $\boldsymbol{\psi} : \mathcal{M} \times \mathcal{N} \to \mathbb{R}^{p\times 1}$ be partitioned as

$$\boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \boldsymbol{\vartheta}(\mathbf{x}_k) \\ \boldsymbol{\upsilon}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix}, \tag{3.13}$$

where the state-dependent lifting functions are $\boldsymbol{\vartheta} : \mathcal{M} \to \mathbb{R}^{p_\vartheta \times 1}$, the input-dependent lifting functions are $\boldsymbol{\upsilon} : \mathcal{M} \times \mathcal{N} \to \mathbb{R}^{p_\upsilon \times 1}$, and the lifting function dimensions satisfy $p_\vartheta + p_\upsilon = p$. With an exogenous input, substituting the state and input into (3.12) results in [49, §6.5.1]

$$\boldsymbol{\vartheta}(\mathbf{x}_{k+1}) = \mathbf{U}\boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\epsilon}_k, \tag{3.14}$$

where $\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$. Expanding (3.14) yields the familiar linear state-space form,

$$\boldsymbol{\vartheta}(\mathbf{x}_{k+1}) = \mathbf{A}\boldsymbol{\vartheta}(\mathbf{x}_k) + \mathbf{B}\boldsymbol{\upsilon}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\epsilon}_k. \tag{3.15}$$

3.4 Approximating the Koopman Operator from Data

In this section, a selection of common Koopman operator approximation methods are presented. Many Koopman operator approximation methods are framed as extensions of DMD. DMD itself was developed in the fluid dynamics community in [51, 52], and its connection to Koopman operator theory was then made in [53]. Later, DMDc [54] was proposed as an extension to DMD that considered exogenous inputs. Approximating the Koopman operator using a dictionary of nonlinear functions as first proposed in [55], wherein the method was called EDMD. Despite its name, EDMD is very similar to the least-squares approach to Koopman operator approximation. The least-squares approach, while simple, is not commonly used, as it scales poorly with the number of lifting functions and data snapshots.

These methods are presented here in order of simplicity, rather than in order of appearance in the literature. First, the least-squares solution is discussed, which, while impractical for large systems, is the easiest method to understand. Then EDMD is discussed, which can be thought of as an extension to the least-squares method. Finally, DMD and DMDc are discussed.

The terms DMD and DMDc typically imply the use of linear lifting functions. However, in this thesis, they are treated as methods to compute the leading eigendecomposition of the Koopman matrix, regardless of the choice of lifting functions. EDMD is sometimes called EDMDc when exogenous inputs are present. Since the method requires no modification when considering inputs, it is referred to as EDMD in this thesis, even when exogenous inputs are included.

3.4.1 Snapshot matrices

To approximate the Koopman matrix from a dataset $\mathcal{D} = \{\mathbf{x}_k, \mathbf{u}_k\}_{k=0}^{q}$, consider the lifted snapshot matrices

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}_0 & \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_{q-1} \end{bmatrix} \in \mathbb{R}^{p \times q}, \tag{3.16}$$

$$\boldsymbol{\Theta}_+ = \begin{bmatrix} \boldsymbol{\vartheta}_1 & \boldsymbol{\vartheta}_2 & \cdots & \boldsymbol{\vartheta}_q \end{bmatrix} \in \mathbb{R}^{p_\vartheta \times q}, \tag{3.17}$$

where $\boldsymbol{\psi}_k = \boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k)$ and $\boldsymbol{\vartheta}_k = \boldsymbol{\vartheta}(\mathbf{x}_k)$. These snapshot matrices will be used in all the following Koopman operator approximation methods.

In the case of DMD, where there are no exogenous inputs, the snapshot matrices are defined as

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}_0 & \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_{q-1} \end{bmatrix} \in \mathbb{R}^{p \times q}, \tag{3.18}$$

$$\boldsymbol{\Psi}_+ = \begin{bmatrix} \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \cdots & \boldsymbol{\psi}_q \end{bmatrix} \in \mathbb{R}^{p \times q}, \tag{3.19}$$

where $\boldsymbol{\psi}_k = \boldsymbol{\psi}(\mathbf{x}_k)$.

3.4.2 Least-squares

Least-squares is the simplest way to approximate the Koopman matrix from data. Recall from (3.14) that

$$\boldsymbol{\vartheta}_{k+1} = \mathbf{U}\boldsymbol{\psi}_k + \boldsymbol{\epsilon}_k, \tag{3.20}$$

Minimizing the residual $\boldsymbol{\epsilon}_k$ for each snapshot in the dataset leads to the optimization problem

$$\min_{\mathbf{U}} \; J(\mathbf{U}) = \frac{1}{q} \left\| \boldsymbol{\Theta}_+ - \mathbf{U}\boldsymbol{\Psi} \right\|_{\mathsf{F}}^2, \tag{3.21}$$

whose solution is [49, §1.2.1]

$$\mathbf{U} = \boldsymbol{\Theta}_+ \boldsymbol{\Psi}^\dagger. \tag{3.22}$$

When the lifted dataset contains many snapshots or many lifting functions, the pseudoinverse required in (3.22) is numerically ill-conditioned. EDMD addresses the case where there are many snapshots, while DMD and DMDc address the case where there are many lifting functions.

3.4.3 Extended dynamic mode decomposition

Extended dynamic mode decomposition (EDMD) [55] improves numerical conditioning in the least-squares approach to Koopman operator approximation when the dataset contains many fewer states than snapshots (*i.e.*, when $p \ll q$) [49, §10.3]. Assuming $\boldsymbol{\Psi}$ is full rank, the EDMD approximation of the Koopman matrix is

$$\mathbf{U} = \boldsymbol{\Theta}_+ \left(\boldsymbol{\Psi}^{\mathsf{T}} \boldsymbol{\Psi}^{\mathsf{T}\dagger} \right) \boldsymbol{\Psi}^\dagger \tag{3.23}$$

$$= \left(\boldsymbol{\Theta}_+ \boldsymbol{\Psi}^{\mathsf{T}} \right) \left(\boldsymbol{\Psi} \boldsymbol{\Psi}^{\mathsf{T}} \right)^\dagger \tag{3.24}$$

$$= \mathbf{G}\mathbf{H}^\dagger, \tag{3.25}$$

where

$$\mathbf{G} = \frac{1}{q}\boldsymbol{\Theta}_+\boldsymbol{\Psi}^\mathsf{T} \in \mathbb{R}^{p_\vartheta \times p}, \tag{3.26}$$

$$\mathbf{H} = \frac{1}{q}\boldsymbol{\Psi}\boldsymbol{\Psi}^\mathsf{T} \in \mathbb{R}^{p \times p}. \tag{3.27}$$

Note that, when the columns of $\boldsymbol{\Psi}$ are linearly independent, $\mathbf{H} = \mathbf{H}^\mathsf{T} > 0$.

EDMD can also be viewed as minimizing

$$J(\mathbf{U}) = \frac{1}{q}\left\|\boldsymbol{\Theta}_+\boldsymbol{\Psi}^\mathsf{T} - \mathbf{U}\boldsymbol{\Psi}\boldsymbol{\Psi}^\mathsf{T}\right\|_\mathsf{F}^2 \tag{3.28}$$

$$= \left\|\mathbf{G} - \mathbf{U}\mathbf{H}\right\|_\mathsf{F}^2. \tag{3.29}$$

In this thesis, methods that minimize (3.29) are referred to as variants of EDMD.

Tikhonov regularization [56, 57], which penalizes the squared Frobenius norm of the unknown matrix in a regression problem, can be easily incorporated into EDMD. Consider the regularized Koopman cost function,

$$J(\mathbf{U};\alpha) = \frac{1}{q}\|\boldsymbol{\Theta}_+ - \mathbf{U}\boldsymbol{\Psi}\|_\mathsf{F}^2 + \frac{\alpha}{q}\|\mathbf{U}\|_\mathsf{F}^2 \tag{3.30}$$

$$= \frac{1}{q}\mathrm{tr}\Big((\boldsymbol{\Theta}_+ - \mathbf{U}\boldsymbol{\Psi})(\boldsymbol{\Theta}_+ - \mathbf{U}\boldsymbol{\Psi})^\mathsf{T}\Big) + \frac{\alpha}{q}\mathrm{tr}\big(\mathbf{U}\mathbf{U}^\mathsf{T}\big) \tag{3.31}$$

$$= \mathrm{tr}\left(\frac{1}{q}\boldsymbol{\Theta}_+\boldsymbol{\Theta}_+^\mathsf{T} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^\mathsf{T}\big\} + \mathbf{U}\left(\mathbf{H} + \frac{\alpha}{q}\mathbf{1}\right)\mathbf{U}^\mathsf{T}\right). \tag{3.32}$$

Comparing (3.32) with the unregularized cost function, given by

$$J(\mathbf{U}) = \mathrm{tr}\left(\frac{1}{q}\boldsymbol{\Theta}_+\boldsymbol{\Theta}_+^\mathsf{T} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^\mathsf{T}\big\} + \mathbf{U}\mathbf{H}\mathbf{U}^\mathsf{T}\right), \tag{3.33}$$

demonstrates that

$$\mathbf{U} = \mathbf{G}\mathbf{H}_\alpha^\dagger \tag{3.34}$$

minimizes the Tikhonov-regularized cost function, where

$$\mathbf{H}_\alpha = \mathbf{H} + \frac{\alpha}{q}\mathbf{1}. \tag{3.35}$$

Including Tikhonov regularization in EDMD therefore amounts to replacing all occurrences of $\mathbf{H}$ with $\mathbf{H}_\alpha$.

3.4.4 Dynamic mode decomposition

Dynamic mode decomposition (DMD) [51, 52, 58] computes the leading eigenvalues and eigenvectors of the Koopman operator when no exogenous inputs are present. This approach is particularly useful when the dimension of the lifted state is so high that the Koopman matrix cannot be formed explicitly (*i.e.*, when $p \gg q$). The eigenvectors of $\mathbf{U}$ approximated with DMD are called *DMD modes*.

Consider the truncated SVD of $\mathbf{\Psi}$,

$$\mathbf{\Psi} \approx \mathbf{Q}\mathbf{\Sigma}\mathbf{Z}^{\mathsf{T}}, \tag{3.36}$$

where $\mathbf{Q} \in \mathbb{R}^{p\times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$, $\mathbf{Z} \in \mathbb{R}^{q\times r}$, and r is the number of singular values to retain. A common approach for choosing r is the hard thresholding algorithm described in [34][49, §8.2]. DMD computes the leading eigenvalues and eigenvectors of $\mathbf{U}$ indirectly through

$$\hat{\mathbf{U}} = \mathbf{Q}^{\mathsf{T}}\mathbf{U}\mathbf{Q} \in \mathbb{R}^{r\times r}, \tag{3.37}$$

which is the projection of $\mathbf{U}$ onto $\mathcal{R}(\mathbf{Q})$. The matrix $\hat{\mathbf{U}}$ can therefore be seen as a low-rank approximation of $\mathbf{U}$. That is,

$$\mathbf{U} \approx \mathbf{Q}\hat{\mathbf{U}}\mathbf{Q}^{\mathsf{T}}. \tag{3.38}$$

Equation (3.38) holds exactly in $\mathcal{R}(\mathbf{Q})$. The projected Koopman matrix $\hat{\mathbf{U}}$ defines a reduced-order version of (3.10),

$$\hat{\boldsymbol{\psi}}_{k+1} = \hat{\mathbf{U}}\hat{\boldsymbol{\psi}}_k + \hat{\boldsymbol{\epsilon}}_k, \tag{3.39}$$

where $\hat{\boldsymbol{\psi}}_k = \mathbf{Q}^{\mathsf{T}}\boldsymbol{\psi}_k$ and $\hat{\boldsymbol{\epsilon}}_k$ is the residual error. Substituting in the least-squares solution to the Koopman matrix results into (3.37) results in

$$\begin{aligned}
\hat{\mathbf{U}} &= \mathbf{Q}^{\mathsf{T}}(\mathbf{\Psi}_{+}\mathbf{\Psi}^{\dagger})\mathbf{Q} && (3.40)\\
&= \mathbf{Q}^{\mathsf{T}}(\mathbf{\Psi}_{+}\mathbf{Z}\mathbf{\Sigma}^{\dagger}\mathbf{Q}^{\mathsf{T}})\mathbf{Q} && (3.41)\\
&= \mathbf{Q}^{\mathsf{T}}\mathbf{\Psi}_{+}\mathbf{Z}\mathbf{\Sigma}^{\dagger}, && (3.42)
\end{aligned}$$

which allows $\hat{\mathbf{U}} \in \mathbb{R}^{r\times r}$ to be computed without ever explicitly forming $\mathbf{U} \in \mathbb{R}^{p\times p}$ or any other intractably large matrix.

The relationship between the eigendecompositions of $\mathbf{U}$ and $\hat{\mathbf{U}}$ is now derived. Consider the eigendecomposition of $\mathbf{U}$,

$$\mathbf{U}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}. \tag{3.43}$$

Next, consider only the eigenvectors that lie in $\mathcal{R}(\mathbf{Q})$. That is, let

$$\mathbf{V} = \mathbf{Q}\hat{\mathbf{V}}, \tag{3.44}$$

which results in

$$\mathbf{U}\mathbf{Q}\hat{\mathbf{V}} = \mathbf{Q}\hat{\mathbf{V}}\mathbf{\Lambda}. \tag{3.45}$$

Premultiplying by $\mathbf{Q}^{\mathsf{T}}$ yields

$$\mathbf{Q}^{\mathsf{T}}\mathbf{U}\mathbf{Q}\hat{\mathbf{V}} = \mathbf{Q}^{\mathsf{T}}\mathbf{Q}\hat{\mathbf{V}}\mathbf{\Lambda} \tag{3.46}$$

$$= \hat{\mathbf{V}}\mathbf{\Lambda}. \tag{3.47}$$

Substituting (3.37) into (3.46) results in the projected eigendecomposition

$$\hat{\mathbf{U}}\hat{\mathbf{V}} = \hat{\mathbf{V}}\mathbf{\Lambda}, \tag{3.48}$$

which implies that the eigenvalues of $\hat{\mathbf{U}}$ are also eigenvalues of $\mathbf{U}$.

One method to compute the eigenvectors of $\mathbf{U}$ is to use (3.44), which yields

$$\mathbf{V}_{\text{proj}} = \mathbf{Q}\hat{\mathbf{V}}, \tag{3.49}$$

where the columns of $\mathbf{V}_{\text{proj}}$ are called the *projected DMD modes*. However, the columns of $\mathbf{V}_{\text{proj}}$ are not the true eigenvectors of $\mathbf{U}$, but rather the eigenvectors of $\mathbf{U}$ projected onto $\mathcal{R}(\mathbf{Q})$ [58, §2.3].

To recover the true eigenvectors of $\mathbf{U}$, consider the eigendecomposition

$$\lambda\mathbf{v}_i = \mathbf{U}\mathbf{v}_i \tag{3.50}$$

$$= (\mathbf{\Psi}_+\mathbf{\Psi}^\dagger)\mathbf{v}_i \tag{3.51}$$

$$= (\mathbf{\Psi}_+\mathbf{Z}\mathbf{\Sigma}^\dagger\mathbf{Q}^{\mathsf{T}})\mathbf{v}_i, \tag{3.52}$$

where $\mathbf{v}_i$ is the i^{th} eigenvector of $\mathbf{U}$. Recall that

$$\hat{\mathbf{v}}_i = \mathbf{Q}^{\mathsf{T}}\mathbf{v}_i. \tag{3.53}$$

Premultiplying by $\mathbf{Q}^\mathsf{T}$ and substituting in (3.53) yields

$$\lambda \mathbf{Q}^\mathsf{T}\mathbf{v}_i = \mathbf{Q}^\mathsf{T}\boldsymbol{\Psi}_+\mathbf{Z}\boldsymbol{\Sigma}^\dagger\mathbf{Q}^\mathsf{T}\mathbf{v}_i, \tag{3.54}$$

$$\lambda \hat{\mathbf{v}}_i = \mathbf{Q}^\mathsf{T}\boldsymbol{\Psi}_+\mathbf{Z}\boldsymbol{\Sigma}^\dagger\hat{\mathbf{v}}_i, \tag{3.55}$$

$$\hat{\mathbf{v}}_i = \frac{1}{\lambda}\mathbf{Q}^\mathsf{T}\boldsymbol{\Psi}_+\mathbf{Z}\boldsymbol{\Sigma}^\dagger\hat{\mathbf{v}}_i. \tag{3.56}$$

Comparing (3.56) with (3.53) implies that

$$\mathbf{v}_i = \frac{1}{\lambda}\boldsymbol{\Psi}_+\mathbf{Z}\boldsymbol{\Sigma}^\dagger\hat{\mathbf{v}}_i. \tag{3.57}$$

Since DMD modes are typically normalized, the scaling factor of $\frac{1}{\lambda}$ can be ignored [58]. The *exact DMD modes* are the columns of

$$\mathbf{V}_{\text{exact}} = \boldsymbol{\Psi}_+\mathbf{Z}\boldsymbol{\Sigma}^\dagger\hat{\mathbf{V}}. \tag{3.58}$$

Note that the columns of $\mathbf{V}_{\text{exact}}$ lie in the column space of $\boldsymbol{\Psi}_+$, which is the column space of $\mathbf{U}$.

The exact DMD modes are recommended for use when implementing DMD, as they are the true eigenvectors of $\mathbf{U}$ [58, §2.3] The projected modes can be used in situations where (3.58) fails. Specifically, this occurs if (3.58) evaluates to zero when computing the mode associated with a zero eigenvalue [49, §1.3].

Equation (3.10) can also be rewritten in terms of the DMD modes,

$$\boldsymbol{\psi}_{k+1} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\dagger\boldsymbol{\psi}_k + \boldsymbol{\epsilon}_k, \tag{3.59}$$

or alternatively,

$$\boldsymbol{\psi}_{k+1} = \mathbf{V}\boldsymbol{\Lambda}^{k+1}(\mathbf{V}^\dagger\boldsymbol{\psi}_0) + \boldsymbol{\epsilon}_k, \tag{3.60}$$

where the elements of $\mathbf{V}^\dagger\boldsymbol{\psi}_0$ are called the *DMD mode amplitudes*.

DMD can be viewed as minimizing

$$J(\mathbf{U}) = \frac{1}{q}\left\|\mathbf{Q}^\mathsf{T}\boldsymbol{\Psi}_+ - \mathbf{Q}^\mathsf{T}\mathbf{U}\boldsymbol{\Psi}\right\|_\mathsf{F}^2, \tag{3.61}$$

which is approximately equivalent to minimizing

$$J(\hat{\mathbf{U}}) = \frac{1}{q}\left\|\mathbf{Q}^{\mathsf{T}}\boldsymbol{\Psi}_{+} - \mathbf{Q}^{\mathsf{T}}\left(\mathbf{Q}\hat{\mathbf{U}}\mathbf{Q}^{\mathsf{T}}\right)\left(\mathbf{Q}\boldsymbol{\Sigma}\mathbf{Z}^{\mathsf{T}}\right)\right\|_{\mathsf{F}}^{2} \tag{3.62}$$

$$= \frac{1}{q}\left\|\mathbf{Q}^{\mathsf{T}}\boldsymbol{\Psi}_{+} - \hat{\mathbf{U}}\boldsymbol{\Sigma}\mathbf{Z}^{\mathsf{T}}\right\|_{\mathsf{F}}^{2}. \tag{3.63}$$

In this thesis, any method that minimizes (3.63) is considered a variation of DMD, regardless of whether or not the modes are computed.

3.4.5 Dynamic mode decomposition with control

Dynamic mode decomposition with control (DMDc) [54] extends DMD to consider exogenous inputs. Like DMD, DMDc computes the leading eigendecomposition of $\mathbf{A}$. The value of $\mathbf{B}$ is either treated as known, or is computed along with the eigendecomposition of $\mathbf{A}$. In this chapter, it is assumed that $\mathbf{B}$ is unknown.

Consider the truncated SVD of $\boldsymbol{\Psi}$,

$$\boldsymbol{\Psi} \approx \tilde{\mathbf{Q}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}, \tag{3.64}$$

where $\tilde{\mathbf{Q}} \in \mathbb{R}^{p\times\tilde{r}}$, $\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$, and $\tilde{\mathbf{Z}} \in \mathbb{R}^{q\times\tilde{r}}$. Unlike in DMD, a projection onto $\tilde{\mathbf{Q}}$ cannot be used to reduce the dimension of $\mathbf{A}$ and $\mathbf{B}$, since $\boldsymbol{\Psi}$ and $\boldsymbol{\Theta}_{+}$ do not have the same dimension [49, §6.1.3]. Instead, consider the SVD of $\boldsymbol{\Theta}_{+}$,

$$\boldsymbol{\Theta}_{+} \approx \hat{\mathbf{Q}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{Z}}^{\mathsf{T}}, \tag{3.65}$$

where $\hat{\mathbf{Q}} \in \mathbb{R}^{p_\vartheta\times\hat{r}}$, $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{\hat{r}\times\hat{r}}$, and $\hat{\mathbf{Z}} \in \mathbb{R}^{q\times\hat{r}}$. Note that, typically, $\hat{r} < \tilde{r}$ [49, §6.1.3].

The projected versions of $\mathbf{A}$ and $\mathbf{B}$ are [54]

$$\hat{\mathbf{A}} = \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{A}\hat{\mathbf{Q}} \tag{3.66}$$

$$= \hat{\mathbf{Q}}^{\mathsf{T}}\boldsymbol{\Theta}_{+}\tilde{\mathbf{Z}}\tilde{\boldsymbol{\Sigma}}^{\dagger}\tilde{\boldsymbol{Q}}_{1}^{\mathsf{T}}\hat{\mathbf{Q}} \tag{3.67}$$

$$= \hat{\boldsymbol{\Sigma}}\hat{\mathbf{Z}}^{\mathsf{T}}\tilde{\mathbf{Z}}\tilde{\boldsymbol{\Sigma}}^{\dagger}\tilde{\boldsymbol{Q}}_{1}^{\mathsf{T}}\hat{\mathbf{Q}}, \tag{3.68}$$

$$\hat{\mathbf{B}} = \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{B} \tag{3.69}$$

$$= \hat{\mathbf{Q}}^{\mathsf{T}}\boldsymbol{\Theta}_{+}\tilde{\mathbf{Z}}\tilde{\boldsymbol{\Sigma}}^{\dagger}\tilde{\boldsymbol{Q}}_{2}^{\mathsf{T}} \tag{3.70}$$

$$= \hat{\boldsymbol{\Sigma}}\hat{\mathbf{Z}}^{\mathsf{T}}\tilde{\mathbf{Z}}\tilde{\boldsymbol{\Sigma}}^{\dagger}\tilde{\boldsymbol{Q}}_{2}^{\mathsf{T}}, \tag{3.71}$$

where $\tilde{\mathbf{Q}} = \begin{bmatrix} \tilde{\mathbf{Q}}_1 & \tilde{\mathbf{Q}}_2 \end{bmatrix}$, $\tilde{\mathbf{Q}}_1 \in \mathbb{R}^{p_\vartheta \times \tilde{r}}$, $\tilde{\mathbf{Q}}_2 \in \mathbb{R}^{p_\upsilon \times \tilde{r}}$, $\hat{\mathbf{A}} \in \mathbb{R}^{\hat{r} \times \hat{r}}$, and $\hat{\mathbf{B}} \in \mathbb{R}^{\hat{r} \times p_\upsilon}$. It follows that

$$\hat{\mathbf{U}} = \hat{\mathbf{Q}}^\mathsf{T} \mathbf{U} \begin{bmatrix} \hat{\mathbf{Q}} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{3.72}$$

Following the same logic as the previous section, the eigenvalues of $\hat{\mathbf{A}}$ are also eigenvalues of $\mathbf{A}$. The exact DMD modes are the columns of [54]

$$\mathbf{V}_{\text{exact}} = \boldsymbol{\Theta}_+ \tilde{\mathbf{Z}} \tilde{\boldsymbol{\Sigma}}^\dagger \tilde{\mathbf{Q}}_1^\mathsf{T} \hat{\mathbf{Q}} \hat{\mathbf{V}}, \tag{3.73}$$

where $\hat{\mathbf{A}}\hat{\mathbf{V}} = \hat{\mathbf{V}}\boldsymbol{\Lambda}$.

DMDc can be viewed as minimizing

$$J(\mathbf{U}) = \frac{1}{q} \left\| \hat{\mathbf{Q}}^\mathsf{T} \boldsymbol{\Theta}_+ - \hat{\mathbf{Q}}^\mathsf{T} \mathbf{U} \boldsymbol{\Psi} \right\|_\mathsf{F}^2, \tag{3.74}$$

which is approximately equivalent to minimizing

$$J(\hat{\mathbf{U}}) = \frac{1}{q} \left\| \hat{\mathbf{Q}}^\mathsf{T} \left(\hat{\mathbf{Q}} \hat{\boldsymbol{\Sigma}} \hat{\mathbf{Z}}^\mathsf{T} \right) - \hat{\mathbf{Q}}^\mathsf{T} \left(\hat{\mathbf{Q}} \hat{\mathbf{U}} \begin{bmatrix} \hat{\mathbf{Q}}^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \right) \boldsymbol{\Psi} \right\|_\mathsf{F}^2 \tag{3.75}$$

$$= \frac{1}{q} \left\| \hat{\boldsymbol{\Sigma}} \hat{\mathbf{Z}}^\mathsf{T} - \hat{\mathbf{U}} \begin{bmatrix} \hat{\mathbf{Q}}^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Psi} \right\|_\mathsf{F}^2. \tag{3.76}$$

In this thesis, any method that minimizes (3.76) is considered a variation of DMDc, regardless of whether or not the modes are computed.

3.5 Koopman Lifting Functions

In this section, a selection of Koopman lifting functions is described. For the sake of brevity, exogenous inputs are not shown and the time index k is omitted unless required.

When designing lifting functions, the first m lifted states are often chosen to be the state of the original difference equation, $\mathbf{x} \in \mathbb{R}^{m \times 1}$. Specifically, [49, §3.3.1]

$$\boldsymbol{\vartheta}(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ \vartheta_{m+1}(\mathbf{x}) \\ \vdots \\ \vartheta_{p_\vartheta}(\mathbf{x}) \end{bmatrix}. \tag{3.77}$$

This choice of lifting functions makes it easier to recover the original state from the lifted state. If $\mathbf{x}$ is not explicitly included in $\boldsymbol{\vartheta}(\mathbf{x})$, it can instead be recovered using least-squares [12, §3.2.1].

3.5.1 Polynomial lifting functions

Polynomial lifting functions typically consist of monomials, Hermite polynomials, or Legendre polynomials. For $\mathbf{x} \in \mathbb{R}^{2\times 1}$, monomial lifting functions have the form

$$\boldsymbol{\psi}(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_2^3 \\ \vdots \end{bmatrix}. \tag{3.78}$$

The main advantage of polynomial lifting functions is their simplicity. Polynomial lifting functions suffer from numerical issues when higher degrees are considered, particularly when the training data is not normalized. High-dimensional systems can also be problematic, as the number of lifting functions can explode even for a low polynomial degree. Examples of polynomial lifting functions can be found in [16, 22].

3.5.2 Time delay lifting functions

Time-delayed measurements can be viewed as Koopman lifting functions [12, §7.1.2][27], and have the form

$$\boldsymbol{\psi}(\mathbf{x}_k) = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \\ \mathbf{x}_{k-2} \\ \vdots \end{bmatrix}. \tag{3.79}$$

When implementing time delay lifting functions, extra care must be taken to not mix state snapshots from different training trajectories. Examples of time delay lifting functions can be found in [12, 16, 27].

3.5.3 Radial basis functions

Consider the state $\mathbf{x} \in \mathbb{R}^{n\times 1}$ and a set of constant *centres* $\mathbf{c}_i \in \mathbb{R}^{n\times 1}$, $i = 1, \ldots, n_{\text{centres}}$. A *radial basis function* (RBF) is a function of the scaled distance

$$r_i(\mathbf{x}) = a\|\mathbf{x} - \mathbf{c}_i\|, \tag{3.80}$$

where $a \in \mathbb{R}_{>0}$ is called the *shape parameter*. Common RBFs include

- the exponential RBF [59], $\psi_i(\mathbf{x}) = e^{-r_i(\mathbf{x})}$,
- the Gaussian RBF [59], $\psi_i(\mathbf{x}) = e^{-r_i(\mathbf{x})^2}$,
- the multiquadric RBF, $\psi_i(\mathbf{x}) = \sqrt{1 + r_i(\mathbf{x})^2}$,
- the inverse quadratic RBF, $\psi_i(\mathbf{x}) = \frac{1}{1+r_i(\mathbf{x})^2}$,
- the inverse multiquadric RBF, $\psi_i(\mathbf{x}) = \frac{1}{\sqrt{1+r_i(\mathbf{x})^2}}$,
- the thin plate spline [60, 61], $\psi_i(\mathbf{x}) = r_i(\mathbf{x})^2 \ln(r_i(\mathbf{x}))$, and
- the bump function, $\psi_i(\mathbf{x}) = \begin{cases} \exp\left(-\frac{1}{1-r_i(\mathbf{x})^2}\right), & r_i(\mathbf{x}) < 1 \\ 0, & r_i(\mathbf{x}) \geq 1 \end{cases}$.

RBF centres are placed in a specified region of $\mathbb{R}^{n\times 1}$. They can be placed randomly, on a uniform grid, or using Quasi–Monte Carlo methods like Latin hypercube sampling [62, 63]. The number of centres used directly determines the dimension of the lifted state.

Consider a set of example centres in $\mathbb{R}^{2\times 1}$,

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{c}_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{3.81}$$

The corresponding Gaussian RBF lifting functions with the original state included are

$$\boldsymbol{\psi}(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ e^{-a\|\mathbf{x}-\mathbf{c}_1\|^2} \\ e^{-a\|\mathbf{x}-\mathbf{c}_2\|^2} \\ e^{-a\|\mathbf{x}-\mathbf{c}_3\|^2} \\ e^{-a\|\mathbf{x}-\mathbf{c}_4\|^2} \end{bmatrix}. \tag{3.82}$$

Examples of RBF lifting functions can be found in [59–61, 64, 65].

3.5.4 Random Fourier features

Koopman lifting functions can also be chosen to approximate a given kernel. The following is a summary of the method of random Fourier features, first proposed in [26, Algorithm 1]. Consider a positive definite shift-invariant kernel [66, §1.4]

$$k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y}), \tag{3.83}$$

whose Fourier transform is

$$p(\boldsymbol{\omega}) = \frac{1}{2\pi} \int_{\mathbb{R}^{n\times 1}} e^{-j\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\Delta}} k(\boldsymbol{\Delta}) \mathrm{d}\boldsymbol{\Delta}, \tag{3.84}$$

where $\boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}$. Bochner's theorem [66, §1.4.3] implies that the shift-invariant kernel $k(\boldsymbol{\Delta})$ is positive definite if and only if it is the (forward or inverse) Fourier transform of a nonnegative measure. This means that, if the kernel is appropriately scaled, its Fourier transform can be treated as a probability distribution. Drawing N samples from $p(\boldsymbol{\omega})$ and arranging them into

$$\mathbf{z}(\mathbf{x}) = \sqrt{\frac{1}{N}} \begin{bmatrix} \cos\left(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}\right) \\ \vdots \\ \cos\left(\boldsymbol{\omega}_N^\mathsf{T}\mathbf{x}\right) \\ \sin\left(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x}\right) \\ \vdots \\ \sin\left(\boldsymbol{\omega}_N^\mathsf{T}\mathbf{x}\right) \end{bmatrix} \tag{3.85}$$

yields $2N$ random Fourier features [26]. For sufficiently large N,

$$\mathbf{z}(\mathbf{x})^\mathsf{T}\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y}). \tag{3.86}$$

An alternative to (3.85) is

$$\mathbf{z}(\mathbf{x}) = \sqrt{2} \begin{bmatrix} \cos\left(\boldsymbol{\omega}_1^\mathsf{T}\mathbf{x} + b_1\right) \\ \vdots \\ \cos\left(\boldsymbol{\omega}_N^\mathsf{T}\mathbf{x} + b_N\right) \end{bmatrix}, \tag{3.87}$$

where $b_1, \ldots, b_N$ are drawn uniformly from $[0, 2\pi)$ [26]. Koopman lifting functions using random Fourier features (RFFs) have the form

$$\boldsymbol{\psi}(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ \mathbf{z}(\mathbf{x}) \end{bmatrix}. \tag{3.88}$$

Common kernels, with shape parameter a, and their corresponding Fourier transforms are [26]

- the Gaussian kernel, $e^{-\|\boldsymbol{\Delta}\|_2^2/2}$, whose Fourier transform is a Gaussian distribution,
- the Laplacian kernel, $e^{-\|\boldsymbol{\Delta}\|_1}$, whose Fourier transform is a Cauchy distribution, and
- the Cauchy kernel, $\prod_{n=1}^{N} \frac{2}{1+\Delta_n^2}$, whose Fourier transform is a Laplace distribution,

where $\boldsymbol{\Delta}$ can be replaced with the rescaled $\boldsymbol{\Delta}_a = \sqrt{2a}\boldsymbol{\Delta}$ to incorporate a shape parameter into each kernel. Samples can be drawn from the above distributions using standard software. RFF lifting functions are useful even when N is not sufficiently large for (3.86) to hold. An alternative to RFFs are random binning features, also presented in [26]. Examples of RFF lifting functions can be found in [24–26].

3.5.5 Lifting control inputs

A common design decision when identifying a Koopman model for control is to leave the input unlifted. That is [12],

$$\boldsymbol{\upsilon}(\mathbf{x}, \mathbf{u}) = \mathbf{u}. \tag{3.89}$$

This ensures that a controller synthesized using the Koopman model will directly output $\mathbf{u}$.

Recent work has also shown that bilinear input-dependent lifting functions are a better choice for control affine systems [67]. An alternative set of input-dependent lifting functions proposed in [67] is

$$\boldsymbol{\upsilon}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{u} \otimes \boldsymbol{\vartheta}(\mathbf{x}) \\ \mathbf{u} \end{bmatrix}, \tag{3.90}$$

where $\otimes$ denotes the Kronecker product. These bilinear input-dependent lifting functions are capable of representing all control affine systems, and therefore present an interesting alternative to leaving the input unlifted [67]. In [67], (3.15) is rewritten as

$$\boldsymbol{\vartheta}(\mathbf{x}_{k+1}) = \mathbf{A}\boldsymbol{\vartheta}(\mathbf{x}_k) + \mathbf{B}_1 \left(\mathbf{u}_k \otimes \boldsymbol{\vartheta}(\mathbf{x}_k)\right) + \mathbf{B}_2\mathbf{u}_k + \boldsymbol{\epsilon}_k, \tag{3.91}$$

where the Koopman matrix is partitioned as $\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2 \end{bmatrix}$. Equation (3.91) can be treated as either a linear system or a bilinear system, depending on the application. In [67], bilinear model predictive control (MPC) methods are used to exploit this particular structure and synthesize a controller for the system.

3.6 Local and Global Prediction Error

When using the identified Koopman matrices for prediction, the state and input are lifted, then multiplied by the Koopman matrix. The state is then recovered and re-lifted with the input for the next timestep. Using this prediction method leads to the local error definition presented in [68], which is used throughout this thesis unless otherwise specified. Lifting the initial state only once and repeatedly multiplying by the Koopman matrix would lead to the global error definition of [68].

3.7 The Koopman System

An output equation,

$$\boldsymbol{\zeta}_k = \mathbf{C}\boldsymbol{\vartheta}_k + \mathbf{D}\boldsymbol{\upsilon}_k, \tag{3.92}$$

where $\boldsymbol{\zeta}_k \in \mathbb{R}^{p_\zeta \times 1}$, can also be considered to incorporate the Koopman operator into a true LTI system with input, output, and state. While $\mathbf{D} = \mathbf{0}$ for all Koopman systems in this thesis, $\mathbf{C}$ can be chosen to recover the original states, or any other desired output. If the original state is not directly included in the lifted state, $\mathbf{C}$ can instead be determined using least-squares [12, §3.2.1]. The *Koopman system* is therefore

$$\boldsymbol{\mathcal{G}} \overset{\text{min}}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right], \tag{3.93}$$

where $\overset{\text{min}}{\sim}$ denotes a minimal state space realization [38, §3.2.1].

The Koopman system $\boldsymbol{\mathcal{G}}$ has a corresponding discrete-time transfer matrix [36, §3.7], $\mathbf{G}(z) = \mathbf{C}(z\mathbf{1} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$.

Chapter 4

Koopman Operator Approximation as Semidefinite Programming

Summary

This chapter explains how EDMD and DMDc, the Koopman operator approximation methods described in Chapter 3, can be reformulated as SDPs. The advantage of rewriting these methods as SDPs is that system properties like asymptotic stability and gain can be incorporated into the optimization problems using the LMI constraints presented at the end of Chapter 2. The effects of including these additional constraints will be discussed in Chapters 5 and 6.

4.1 Introduction

In Chapter 3, closed-form solutions to the EDMD and DMDc problems are presented. However, there is no clear way to modify these methods to account for system properties like asymptotic stability or gain. By treating these methods as optimization problems reformulated as SDPs, additional constraints can be added to enforce desired properties on the identified Koopman systems. For example, in Chapter 5, constraints will be used to enforce asymptotic stability and regularize system gain. In Chapter 6, constraints will be used to enforce the known system structure on the closed-loop system and plant. The modular nature of SDPs makes it possible to add these constraints without modifying the original problems.

4.1.1 Related work

SDPs and LMIs are commonly used tools in system identification [69–75]. In the context of the Koopman operator, LMIs have been used to ensure that the identified Koopman system

is dissipative [76], is negative imaginary [77], or has a known gain [78]. However, they have so far not been used to regularize the Koopman operator regression problem. SDPs are also commonly used to synthesize controllers for Koopman models [17, 79–82]. A related optimization problem is posed in [83], where both the Koopman matrix and lifting functions are treated as unknowns. While this problem is NP-hard [48], a convex relaxation allows both to be found by solving two SDPs.

In [76–78], modified versions of the least-squares approach to Koopman operator approximation are used, as opposed to approaches based on EDMD. These formulations may lead to an excess number of variables or constraints when many lifting functions or data snapshots are considered.

4.1.2 Contribution

The contribution of this chapter is the reformulation of EDMD and DMDc as SDPs. These Koopman operator approximation methods scale better with the number of data snapshots and lifting functions compared to the least-squares solution. LMI constraints for spectral norm regularization and nuclear norm regularization are used as examples to demonstrate the modularity of the SDP formulations.

4.2 EDMD as a Semidefinite Program

Two equivalent SDP reformulations of EDMD are derived in this section. The first method was derived in [1, 2], while the second method was derived in [3]. For completeness, both are shown here. In poorly conditioned problems, one of these methods may outperform the other, but in general, both will yield the same optimal Koopman matrix. In most cases, the second form is preferable due to its simpler cost function.

4.2.1 First method

Recall from (3.21) that the Koopman matrix $\mathbf{U}$ minimizes

$$
\begin{aligned}
J(\mathbf{U}) &= \frac{1}{q}\left\|\mathbf{\Theta}_+ - \mathbf{U}\mathbf{\Psi}\right\|_{\mathsf{F}}^2 && (4.1)\\
&= \frac{1}{q}\mathrm{tr}\Big(\left(\mathbf{\Theta}_+ - \mathbf{U}\mathbf{\Psi}\right)\left(\mathbf{\Theta}_+ - \mathbf{U}\mathbf{\Psi}\right)^{\mathsf{T}}\Big) && (4.2)\\
&= \mathrm{tr}\left(\frac{1}{q}\mathbf{\Theta}_+\mathbf{\Theta}_+^{\mathsf{T}} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^{\mathsf{T}}\big\} + \mathbf{U}\mathbf{H}\mathbf{U}^{\mathsf{T}}\right) && (4.3)\\
&= c - 2\mathrm{tr}\big(\mathbf{U}\mathbf{G}^{\mathsf{T}}\big) + \mathrm{tr}\big(\mathbf{U}\mathbf{H}\mathbf{U}^{\mathsf{T}}\big), && (4.4)
\end{aligned}
$$

where $c = \frac{1}{q}\text{tr}\big(\boldsymbol{\Theta}_+\boldsymbol{\Theta}_+^\mathsf{T}\big)$ is a scalar constant, and $\mathbf{G}$ and $\mathbf{H}$ are defined in (3.26) and (3.27). The minimization of (4.4) is equivalent to the minimization of

$$J(\mathbf{U}, \nu, \mathbf{W}) = c - 2\text{tr}\big(\mathbf{U}\mathbf{G}^\mathsf{T}\big) + \nu \tag{4.5}$$

subject to

$$\text{tr}(\mathbf{W}) < \nu, \tag{4.6}$$

$$\mathbf{W} > 0, \tag{4.7}$$

$$\mathbf{U}\mathbf{H}\mathbf{U}^\mathsf{T} < \mathbf{W}, \tag{4.8}$$

where ν and $\mathbf{W}$ are slack variables that allow the cost function to be rewritten using LMIs [31, §3.7.1]. To rewrite (4.8) as an LMI, consider the matrix decomposition $\mathbf{H} = \mathbf{R}\mathbf{R}^\mathsf{T}$, where $\mathbf{H}$ is assumed to be positive definite. The matrix $\mathbf{R}$ can be found using the Cholesky factorization or eigendecomposition of $\mathbf{H}$, or the SVD of $\boldsymbol{\Psi}$. Assuming the decomposition has been performed, the quadratic term in the optimization problem becomes

$$\mathbf{W} - \mathbf{U}\mathbf{H}\mathbf{U}^\mathsf{T} = \mathbf{W} - \mathbf{U}\mathbf{R}\mathbf{R}^\mathsf{T}\mathbf{U}^\mathsf{T} \tag{4.9}$$

$$= \mathbf{W} - (\mathbf{U}\mathbf{R})\,\mathbf{1}(\mathbf{U}\mathbf{R})^\mathsf{T}, \tag{4.10}$$

where $\mathbf{1}$ is the identity matrix. Applying the Schur complement [31, §2.4.1] to (4.10) yields

$$\begin{bmatrix} \mathbf{W} & \mathbf{U}\mathbf{R} \\ * & \mathbf{1} \end{bmatrix} > 0. \tag{4.11}$$

The LMI form of the optimization problem is therefore

$$\min J(\mathbf{U}, \nu, \mathbf{W}) = c - 2\text{tr}\big(\mathbf{U}\mathbf{G}^\mathsf{T}\big) + \nu \tag{4.12}$$

$$\text{s.t. } \text{tr}(\mathbf{W}) < \nu, \tag{4.13}$$

$$\mathbf{W} > 0, \tag{4.14}$$

$$\begin{bmatrix} \mathbf{W} & \mathbf{U}\mathbf{R} \\ * & \mathbf{1} \end{bmatrix} > 0, \tag{4.15}$$

where $\mathbf{H} = \mathbf{R}\mathbf{R}^\mathsf{T}$.

As previously mentioned, the decomposition $\mathbf{H} = \mathbf{R}\mathbf{R}^\mathsf{T}$ can be realized via the Cholesky factorization, eigendecomposition, or singular value decomposition. Using the Cholesky factorization directly gives $\mathbf{R}$. When using the eigendecomposition, $\mathbf{H} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\mathsf{T}$, it

follows that $\mathbf{R} = \mathbf{V}\sqrt{\mathbf{\Lambda}}$. Alternatively, using the SVD, $\mathbf{\Psi} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Z}^{\mathsf{T}}$, and substituting it into the definition of $\mathbf{H}$ in (3.27) yields $\mathbf{H} = \frac{1}{q}\mathbf{Q}\mathbf{\Sigma}^2\mathbf{Q}^{\mathsf{T}}$. It follows that $\mathbf{R} = \frac{1}{\sqrt{q}}\mathbf{Q}\mathbf{\Sigma}$.

4.2.2 Second method

By expanding the squared Frobenius norm, the cost function in (3.21) can be rewritten as

$$J(\mathbf{U}) = \frac{1}{q}\mathrm{tr}\big(\mathbf{\Theta}_+\mathbf{\Theta}_+^{\mathsf{T}} - \mathrm{He}\big\{\mathbf{U}\mathbf{\Psi}\mathbf{\Theta}_+^{\mathsf{T}}\big\} + \mathbf{U}\mathbf{\Psi}\mathbf{\Psi}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\big). \tag{4.16}$$

Substituting in the definitions of $\mathbf{G}$ and $\mathbf{H}$ from (3.26) and (3.27) yields

$$J(\mathbf{U}) = \mathrm{tr}\big(\mathbf{F} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^{\mathsf{T}}\big\} + \mathbf{U}\mathbf{H}\mathbf{U}^{\mathsf{T}}\big), \tag{4.17}$$

where $\mathbf{F} = \frac{1}{q}\mathbf{\Theta}_+\mathbf{\Theta}_+^{\mathsf{T}}$. Introducing a slack variable [31, §3.7.1] results in an optimization problem equivalent to minimizing (3.21) over $\mathbf{U}$, that being

$$\min\ J(\mathbf{U}, \mathbf{W}) = \mathrm{tr}(\mathbf{W}) \tag{4.18}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{4.19}$$

$$\mathbf{F} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^{\mathsf{T}}\big\} + \mathbf{U}\mathbf{H}\mathbf{U}^{\mathsf{T}} < \mathbf{W}. \tag{4.20}$$

Next, the Schur complement is used to break up the quadratic term in (4.20) [31, §2.4.1]. Again, assuming that $\mathbf{H} = \mathbf{H}^{\mathsf{T}} > 0$, consider its Cholesky factorization, $\mathbf{H} = \mathbf{R}\mathbf{R}^{\mathsf{T}}$. Substituting in the Cholesky factorization and applying the Schur complement yields the SDP

$$\min\ J(\mathbf{U}, \mathbf{W}) = \mathrm{tr}\,(\mathbf{W}) \tag{4.21}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{4.22}$$

$$\begin{bmatrix} -\mathbf{W} + \mathbf{F} - \mathrm{He}\big\{\mathbf{U}\mathbf{G}^{\mathsf{T}}\big\} & \mathbf{U}\mathbf{R} \\ * & -\mathbf{1} \end{bmatrix} < 0. \tag{4.23}$$

4.3 DMDc as a Semidefinite Program

DMDc [54] reduces the dimension of the Koopman matrix regression problem when the dataset contains many more lifted states than time snapshots (i.e. , $p \gg q$) [49, §10.3]. In DMDc, the Koopman matrix is projected onto the left singular vectors of $\mathbf{\Theta}_+$. The size of the problem is then controlled by retaining only the $\hat{r}$ largest singular values in the SVD.

Consider the truncated SVD $\mathbf{\Theta}_+ \approx \hat{\mathbf{Q}}\hat{\mathbf{\Sigma}}\hat{\mathbf{Z}}^{\mathsf{T}}$, where $\hat{\mathbf{Q}} \in \mathbb{R}^{p_\vartheta \times \hat{r}}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{\hat{r}\times\hat{r}}$, and $\hat{\mathbf{Z}} \in \mathbb{R}^{q\times\hat{r}}$. Instead of solving for $\mathbf{U}$ the regression problem is written in terms of its projection onto

$\mathcal{R}(\hat{\mathbf{Q}})$ [54],

$$\hat{\mathbf{U}} = \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}\begin{bmatrix}\hat{\mathbf{Q}} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \tag{4.24}$$

where $\hat{\mathbf{U}} \in \mathbb{R}^{\hat{r}\times\hat{r}+p_\upsilon}$ is significantly smaller than $\mathbf{U} \in \mathbb{R}^{p_\vartheta \times p}$. Also consider the truncated SVD $\boldsymbol{\Psi} \approx \tilde{\mathbf{Q}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}$, where $\tilde{\mathbf{Q}} \in \mathbb{R}^{p\times\tilde{r}}$, $\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$, and $\tilde{\mathbf{Z}} \in \mathbb{R}^{q\times\tilde{r}}$. The SVD dimensions $\hat{r}$ and $\tilde{r}$ are a design choice. A common approach is the hard thresholding algorithm described in [34]. Note that typically $\hat{r} < \tilde{r}$ [49, §6.1.3].

To reformulate DMDc as a convex optimization problem with LMI constraints, the cost function is rewritten in terms of $\hat{\mathbf{U}}$ instead of $\mathbf{U}$ by substituting in the SVDs of $\boldsymbol{\Theta}_+$ and $\boldsymbol{\Psi}$. Recall that the Koopman cost function can be rewritten as (4.16). Minimizing and introducing the slack variable $\mathbf{W}$ results in [31, §3.7.1]

$$\min\ J(\mathbf{U},\mathbf{W}) = \frac{1}{q}\mathrm{tr}(\mathbf{W}) \tag{4.25}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{4.26}$$

$$\boldsymbol{\Theta}_+\boldsymbol{\Theta}_+^{\mathsf{T}} - \mathrm{He}\{\mathbf{U}\boldsymbol{\Psi}\boldsymbol{\Theta}_+^{\mathsf{T}}\} + \mathbf{U}\boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}} < \mathbf{W}. \tag{4.27}$$

Substituting the SVDs of $\boldsymbol{\Theta}_+$ and $\boldsymbol{\Psi}$ into the optimization problem yields

$$\min\ J(\mathbf{U},\mathbf{W}) = \frac{1}{q}\mathrm{tr}(\mathbf{W}) \tag{4.28}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{4.29}$$

$$\hat{\mathbf{Q}}\hat{\boldsymbol{\Sigma}}^2\hat{\mathbf{Q}}^{\mathsf{T}} - \mathrm{He}\Big\{\mathbf{U}\tilde{\mathbf{Q}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}\hat{\mathbf{Z}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{Q}}^{\mathsf{T}}\Big\} + \mathbf{U}\tilde{\mathbf{Q}}\tilde{\boldsymbol{\Sigma}}^2\tilde{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}} < \mathbf{W}. \tag{4.30}$$

Next, consider the projection of (4.28) and (4.30) onto $\mathcal{R}(\hat{\mathbf{Q}})$. Recall that $\mathbf{W} > 0$ is equivalent to

$$\boldsymbol{\vartheta}^{\mathsf{T}}\mathbf{W}\boldsymbol{\vartheta} > 0,\ \forall\boldsymbol{\vartheta} \neq \mathbf{0} \in \mathbb{R}^{p_\vartheta\times 1}. \tag{4.31}$$

Since (4.31) holds over all of $\mathbb{R}^{p_\vartheta\times 1}$, it must also hold over the subspace $\mathcal{R}(\hat{\mathbf{Q}})$. Let the vectors in $\mathcal{R}(\hat{\mathbf{Q}})$ be parameterized by

$$\boldsymbol{\vartheta} = \hat{\mathbf{Q}}\hat{\boldsymbol{\vartheta}}, \tag{4.32}$$

where $\hat{\boldsymbol{\vartheta}} \in \mathbb{R}^{\hat{r}\times 1}$. Substituting (4.32) into (4.31) yields

$$\hat{\boldsymbol{\vartheta}}^{\mathsf{T}}\hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{W}\hat{\mathbf{Q}}\hat{\boldsymbol{\vartheta}} > 0,\ \forall\hat{\boldsymbol{\vartheta}} \neq \mathbf{0} \in \mathbb{R}^{\hat{r}\times 1}, \tag{4.33}$$

which is equivalent to

$$\hat{\mathbf{W}} = \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{W}\hat{\mathbf{Q}} > 0 \tag{4.34}$$

over $\mathcal{R}(\hat{\mathbf{Q}})$. Applying the same logic to (4.30) yields

$$\hat{\mathbf{\Sigma}}^2 - \mathrm{He}\Big\{\hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}\tilde{\mathbf{Q}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}\hat{\mathbf{Z}}\hat{\mathbf{\Sigma}}\Big\} + \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}\tilde{\mathbf{Q}}\tilde{\mathbf{\Sigma}}^2\tilde{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\hat{\mathbf{Q}} < \hat{\mathbf{W}}, \tag{4.35}$$

where the fact that $\hat{\mathbf{Q}}^{\mathsf{T}}\hat{\mathbf{Q}} = \mathbf{1}$ is used.

To further simplify the problem, it is advantageous to rewrite (4.28) in terms of $\hat{\mathbf{W}}$. To accomplish this, first recall that the trace of a matrix is equal to the sum of its eigenvalues. The eigenvalue problem for $\mathbf{W}$ is

$$\mathbf{W}\mathbf{v}_i = \lambda_i\mathbf{v}_i. \tag{4.36}$$

Projecting (4.36) onto $\mathcal{R}(\hat{\mathbf{Q}})$ by substituting $\mathbf{v}_i = \hat{\mathbf{Q}}\hat{\mathbf{v}}_i$, then premultiplying the result by $\hat{\mathbf{Q}}^{\mathsf{T}}$, yields

$$\hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{W}\hat{\mathbf{Q}}\hat{\mathbf{v}}_i = \lambda_i\hat{\mathbf{v}}_i. \tag{4.37}$$

Thus, $\mathbf{W}$ and $\hat{\mathbf{W}}$ share the same eigenvalues for eigenvectors in $\mathcal{R}(\hat{\mathbf{Q}})$, which indicates that minimizing $\mathrm{tr}(\hat{\mathbf{W}})$ is equivalent to minimizing $\mathrm{tr}(\mathbf{W})$ in that subspace.

The full regression problem projected onto $\mathcal{R}(\hat{\mathbf{Q}})$ is therefore

$$\min\ J(\mathbf{U}, \hat{\mathbf{W}}) = \frac{1}{q}\mathrm{tr}(\hat{\mathbf{W}}) \tag{4.38}$$

$$\text{s.t. } \hat{\mathbf{W}} > 0, \tag{4.39}$$

$$\hat{\mathbf{\Sigma}}^2 - \mathrm{He}\Big\{\hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}\tilde{\mathbf{Q}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}\hat{\mathbf{Z}}\hat{\mathbf{\Sigma}}\Big\} + \hat{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}\tilde{\mathbf{Q}}\tilde{\mathbf{\Sigma}}^2\tilde{\mathbf{Q}}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\hat{\mathbf{Q}} < \hat{\mathbf{W}}. \tag{4.40}$$

Substituting $\hat{\mathbf{U}}$ from (4.24) into the optimization problem yields

$$\min\ J(\hat{\mathbf{U}}, \hat{\mathbf{W}}) = \frac{1}{q}\mathrm{tr}(\hat{\mathbf{W}}) \tag{4.41}$$

$$\text{s.t. } \hat{\mathbf{W}} > 0, \tag{4.42}$$

$$\hat{\mathbf{\Sigma}}^2 - \mathrm{He}\Big\{\hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}^{\mathsf{T}}\hat{\mathbf{Z}}\hat{\mathbf{\Sigma}}\Big\} + \hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}}^2\bar{\mathbf{Q}}^{\mathsf{T}}\hat{\mathbf{U}}^{\mathsf{T}} < \hat{\mathbf{W}}, \tag{4.43}$$

where

$$\bar{\mathbf{Q}} = \begin{bmatrix}\hat{\mathbf{Q}} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}^{\mathsf{T}}\tilde{\mathbf{Q}}. \tag{4.44}$$

Applying the Schur complement to (4.43) yields the LMI formulation of DMDc,

$$\min J(\hat{\mathbf{U}}, \hat{\mathbf{W}}) = \frac{1}{q}\mathrm{tr}(\hat{\mathbf{W}}) \tag{4.45}$$

$$\text{s.t. } \hat{\mathbf{W}} > 0, \tag{4.46}$$

$$\begin{bmatrix} -\hat{\mathbf{W}} + \hat{\mathbf{\Sigma}}^2 - \mathrm{He}\left\{\hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}^\intercal\hat{\mathbf{Z}}\hat{\mathbf{\Sigma}}\right\} & \hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}} \\ * & -\mathbf{1} \end{bmatrix} < 0. \tag{4.47}$$

This is now a significantly smaller optimization problem, as its size is controlled by the truncation of the SVD of $\mathbf{\Theta}_+$. Reducing the first dimension of $\hat{\mathbf{U}}$ also reduces the dimension of the slack variable $\hat{\mathbf{W}}$.

4.4 Regularization using Linear Matrix Inequalities

To illustrate how additional constraints can be added to the SDP forms of EDMD and DMDc, two brief examples are shown. The first is DMDc with a spectral norm regularizer, and the second is EDMD with a nuclear norm regularizer. These regularization methods would not be possible with the standard EDMD and DMDc approaches reviewed in Chapter 3. Unlike EDMD with Tikhonov regularization, these optimization problems do not have closed-form solutions.

4.4.1 Spectral norm regularization

To highlight the modularity of the SDP reformulation of DMDc, consider the application of spectral norm regularization to DMDc. Recall from Section 2.2.4 that the spectral norm of a matrix is

$$\|\mathbf{U}\|_2 = \bar{\sigma}(\mathbf{U}). \tag{4.48}$$

The spectral norm can be expressed as an LMI. The inequality $\|\mathbf{U}\|_2 < \gamma$ holds if and only if [31, §3.3.1]

$$\begin{bmatrix} \gamma\mathbf{1} & \mathbf{U} \\ * & \gamma\mathbf{1} \end{bmatrix} > 0. \tag{4.49}$$

Applying this LMI to regularize the spectral norm of the Koopman matrix in the DMDc problem yields

$$\min J(\hat{\mathbf{U}}, \hat{\mathbf{W}}, \gamma; \alpha) = \frac{1}{q}\mathrm{tr}(\hat{\mathbf{W}}) + \frac{\alpha}{q}\gamma \tag{4.50}$$

$$\text{s.t. } \hat{\mathbf{W}} > 0, \tag{4.51}$$

$$\begin{bmatrix} -\hat{\mathbf{W}} + \hat{\mathbf{\Sigma}}^2 - \mathrm{He}\left\{\hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{Z}}^\mathsf{T}\hat{\mathbf{Z}}\hat{\mathbf{\Sigma}}\right\} & \hat{\mathbf{U}}\bar{\mathbf{Q}}\tilde{\mathbf{\Sigma}} \\ * & -\mathbf{1} \end{bmatrix} < 0, \tag{4.52}$$

$$\begin{bmatrix} \gamma\mathbf{1} & \hat{\mathbf{U}} \\ * & \gamma\mathbf{1} \end{bmatrix} > 0. \tag{4.53}$$

4.4.2 Nuclear norm regularization

Next, consider nuclear norm regularization applied to EDMD. Recall from Section 2.2.4 that the nuclear norm is a convex approximation of the rank function. Regularizing using the nuclear norm can result in low-dimensional models, and in the case of the Koopman operator, low-order linear systems [35].

The nuclear norm is defined as

$$\|\mathbf{U}\|_* = \sum_{i=1}^{\min\{p_\vartheta, p\}} \sigma_i(\mathbf{U}), \tag{4.54}$$

where $\sigma_i(\cdot)$ is the i^{th} singular value of $\mathbf{U} \in \mathbb{R}^{p_\vartheta \times p}$. Solving

$$\min_{\mathbf{U}} \ \|\mathbf{U}\|_* \tag{4.55}$$

is equivalent to solving the SDP [35] [31, §3.3.6]

$$\min_{\mathbf{V}_1, \mathbf{V}_2} \ \frac{1}{2}\left(\mathrm{tr}(\mathbf{V}_1) + \mathrm{tr}(\mathbf{V}_2)\right) \tag{4.56}$$

$$\text{s.t. } \begin{bmatrix} \mathbf{V}_1 & \mathbf{U} \\ * & \mathbf{V}_2 \end{bmatrix} \geq 0, \tag{4.57}$$

where $\mathbf{V}_1 = \mathbf{V}_1^\mathsf{T} \in \mathbb{R}^{p_\vartheta \times p_\vartheta}$ and $\mathbf{V}_2 = \mathbf{V}_2^\mathsf{T} \in \mathbb{R}^{p \times p}$. Applying this LMI to regularize the EDMD

problem using the nuclear norm results in

$$\min\ J(\mathbf{U}, \mathbf{W}, \gamma; \alpha) = \operatorname{tr}(\mathbf{W}) + \frac{\alpha}{q}\gamma \tag{4.58}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{4.59}$$

$$\begin{bmatrix} -\mathbf{W} + \mathbf{F} - \operatorname{He}\{\mathbf{U}\mathbf{G}^{\mathsf{T}}\} & \mathbf{U}\mathbf{R} \\ * & -\mathbf{1} \end{bmatrix} < 0, \tag{4.60}$$

$$\operatorname{tr}(\mathbf{V}_1) + \operatorname{tr}(\mathbf{V}_2) \leq 2\gamma, \tag{4.61}$$

$$\begin{bmatrix} \mathbf{V}_1 & \mathbf{U} \\ * & \mathbf{V}_2 \end{bmatrix} \geq 0. \tag{4.62}$$

4.5 Conclusion

SDP reformulations of EDMD and DMDc, two standard Koopman operator approximation methods, are proposed in this chapter. These approaches scale better with the number of data snapshots and lifting functions compared to the least-squares Koopman operator approximation method. These SDP reformulations use more computational resources than the standard implementations of EDMD and DMDc, which require one or two SVDs, but their added flexibility makes this a worthwhile tradeoff in many situations. The SDP reformulations of EDMD and DMDc form the base of subsequent chapters, wherein additional constraints will be included to enforce desired system properties or structure on the identified Koopman system.

Chapter 5

System Norms and Asymptotic Stability in Koopman Operator Approximation

Summary

This chapter proposes two methods to incorporate linear system properties into the Koopman operator regression process. The first method constrains the identified Koopman system to be AS. The second method regularizes the Koopman operator regression problem using the $\mathcal{H}_\infty$ norm of the Koopman system. Both methods are formulated as constraints that can be added to either the EDMD or DMDc problems reformulated as SDPs in Chapter 4. Both methods result in BMIs that are solved through a sequence of convex optimization problems. Two experimental datasets are used to demonstrate the properties of the proposed approaches: one from an aircraft fatigue testing platform, and another from a soft robotic manipulator.

5.1 Introduction

In this chapter, two Koopman operator approximation methods that incorporate desired system properties into the regression process are proposed. Both methods are built upon the SDP reformulations of EDMD and DMDc explained in Chapter 4. The first method proposed constrains the identified Koopman system to be AS, while the second method regularizes the $\mathcal{H}_\infty$ norm of the Koopman system, guaranteeing asymptotic stability along the way.

These methods are motivated by the fact that the regression problem associated with finding an approximate Koopman operator is numerically challenging, as complex lifting function choices can yield unstable or ill-conditioned Koopman models for AS systems [68].

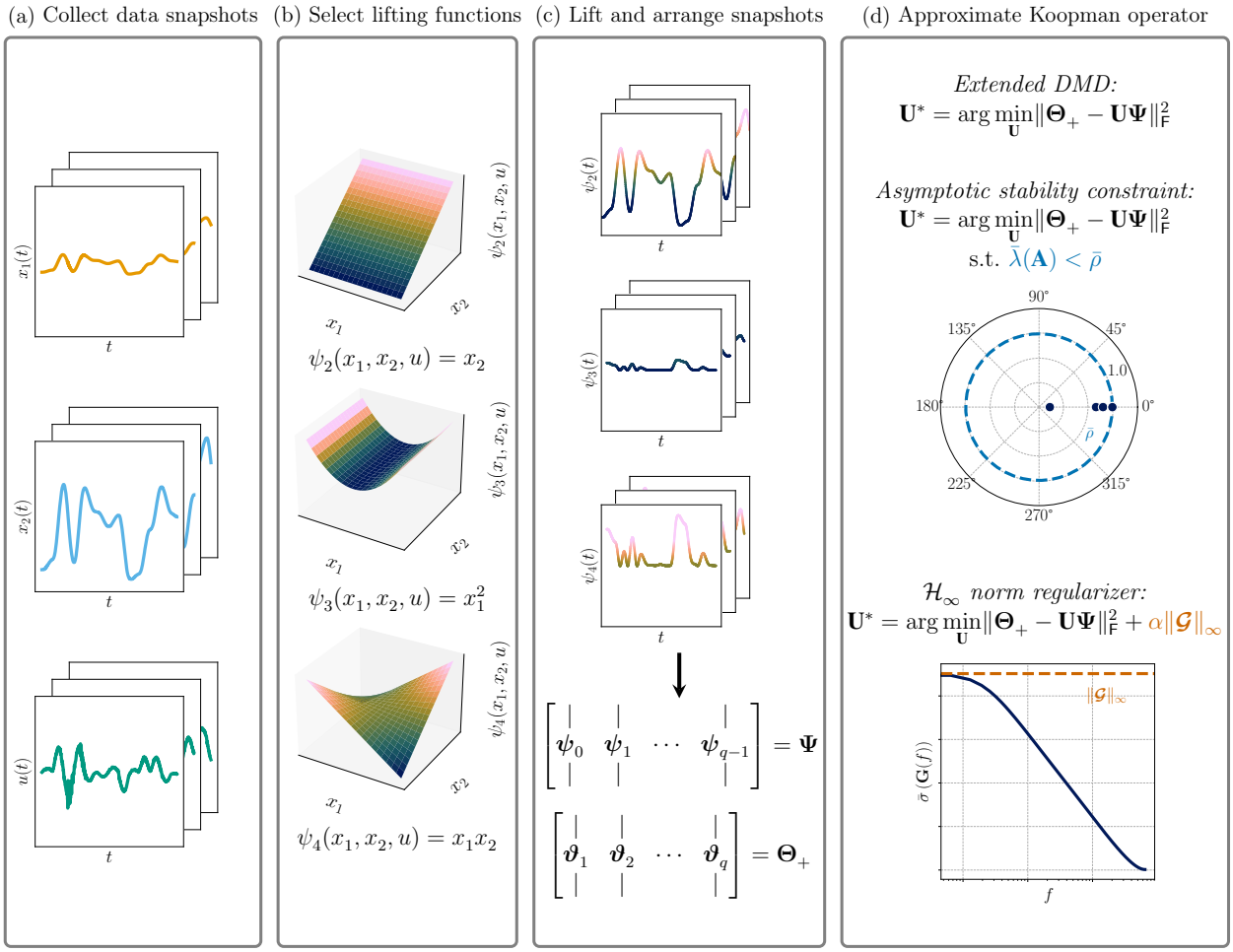


Figure 5.1: Overview of the data-driven Koopman workflow, including the role of the proposed regression methods. (a) First, data is collected from the system to be identified. (b) Next, a set of lifting functions is chosen. (c) The lifting functions are then applied to the data and snapshot matrices are formed. (d) Finally, one of several regression methods is used to approximate the Koopman matrix. The proposed regression methods seek to address the numerical problems often present in this step by viewing it as an optimization over discrete-time linear systems.

Regularization techniques play a crucial role in obtaining usable Koopman models for prediction and control applications. Standard regularization techniques like Tikhonov regularization [56] or the lasso [84] are often used to promote well-conditioned Koopman matrices. These regularization techniques penalize different matrix norms of the Koopman matrix, without considering the fact that the Koopman matrix defines a discrete-time LTI system with input, state, and output. While these methods may indirectly promote asymptotic stability in this Koopman system, their success is highly dependent on the regularization coefficient used. Furthermore, they do not consider the input-output gain of the Koopman system. This chapter takes a systems view of the Koopman matrix regression problem, proposing regression methods that constrain the asymptotic stability of the Koopman system

and regularize the regression problem by penalizing its input-output gain, as represented by its $\mathcal{H}_\infty$ norm. To accomplish this, regularizers and additional constraints are incorporated into the EDMD and DMDc problems as BMI constraints on the reformulations of Chapter 4. The resulting BMIs are solved by solving a sequence of SDPs. The data-driven Koopman workflow and the proposed regression methods are summarized in Figure 5.1.

5.1.1 Related work

The $\mathcal{H}_\infty$ norm of the Koopman operator has previously been considered in [76], however it is in the form of a hard constraint on the system's dissipativity, rather than as a regularizer in the cost function. The identification of Koopman systems with known gain is considered in [78], while the identification of negative imaginary Koopman systems is considered in [77].

In [85], the problem of learning Lyapunov stable and AS linear systems is explored in the context of subspace identification, where Lyapunov inequalities are used to enforce the corresponding stability conditions. The related problem of learning positive real and strictly positive real systems using constrained subspace identification is discussed in [86]. A convex relaxation of the Lyapunov inequality is considered in [87], where linear constraints are added incrementally to enforce the Lyapunov stability of a system. In [68, 88], a gradient-descent method is applied to find locally optimal Lyapunov stable or AS Koopman matrices. The method relies on a parameterization of the Koopman matrix that guarantees Lyapunov or asymptotic stability [89]. While addressing the asymptotic stability problem, this formulation lacks the modularity of the proposed approach.

5.1.2 Contribution

The core contributions of this chapter are the solutions of the EDMD and DMDc problems with asymptotic stability constraints and with system norm regularizers. Of particular focus is the use of the $\mathcal{H}_\infty$ norm as a regularizer, which penalizes the worst-case gain of the Koopman system over all frequencies. Furthermore, weighted $\mathcal{H}_\infty$ norm regularization is explored, which allows the Koopman system's gain to be penalized in a specific frequency band, where experimental measurements may be less reliable, or where system dynamics may be irrelevant. These modifications add BMI constraints to the EDMD and DMDc problems, which must be solved with an iterative method. Finally, the proposed regression methods are evaluated using two experimental datasets, one from a fatigue structural testing platform, and the other from a soft robot arm. The significance of this work is the use of a system norm to regularize the Koopman regression problem, which is viewed as a regression problem over discrete-time linear systems, resulting in a numerically better conditioned data-driven

model.

The software required to fully reproduce the results of this chapter is available at `https://github.com/decargroup/system_norm_koopman`.

5.2 Asymptotic Stability Constraint

Since many systems of interest have AS dynamics, it is desirable to identify Koopman systems that share this property. In [68], it is proven that an AS nonlinear system can only be represented accurately by an AS Koopman system, highlighting the importance of enforcing the asymptotic stability property during the regression process. However, in practice, it is possible to identify an unstable Koopman system from measurements of a Lyapunov stable or AS system [68]. Even if the identified Koopman system is AS in theory, the eigenvalues of its $\mathbf{A}$ matrix may be so close to the unit circle that it is effectively unstable in practice. One solution, presented in this section, is to constrain the largest eigenvalue of $\mathbf{A}$ to be strictly less than one in magnitude, within a desired tolerance. A modified Lyapunov constraint [90, §1.4.4]

$$\mathbf{P} > 0, \tag{5.1}$$

$$\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{A} - \bar{\rho}^2\mathbf{P} < 0, \tag{5.2}$$

can be added to ensure that the magnitude of the largest eigenvalue of $\mathbf{A}$ is no larger than $0 < \bar{\rho} < 1$. Applying the Schur complement to (5.2) yields

$$\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{A} - \bar{\rho}^2\mathbf{P} < 0 \iff \left(\mathbf{A}^\mathsf{T}\mathbf{P}\right)\mathbf{P}^{-1}\left(\mathbf{A}^\mathsf{T}\mathbf{P}\right)^\mathsf{T} - \bar{\rho}^2\mathbf{P} < 0 \tag{5.3}$$

$$\iff -\bar{\rho}\mathbf{P} - \left(-\mathbf{A}^\mathsf{T}\mathbf{P}\right)\left(-\bar{\rho}\mathbf{P}\right)^{-1}\left(-\mathbf{A}^\mathsf{T}\mathbf{P}\right)^\mathsf{T} < 0 \tag{5.4}$$

$$\iff \begin{bmatrix} -\bar{\rho}\,\mathbf{P} & -\mathbf{A}^\mathsf{T}\mathbf{P} \\ * & -\bar{\rho}\,\mathbf{P} \end{bmatrix} < 0,\ -\bar{\rho}\mathbf{P} < 0 \tag{5.5}$$

$$\iff \begin{bmatrix} \bar{\rho}\,\mathbf{P} & \mathbf{A}^\mathsf{T}\mathbf{P} \\ * & \bar{\rho}\,\mathbf{P} \end{bmatrix} > 0,\ \mathbf{P} > 0. \tag{5.6}$$

The full EDMD optimization problem with asymptotic stability constraint is therefore

$$\min\ J(\mathbf{U}, \nu, \mathbf{W}, \mathbf{P}; \bar{\rho}) = c - 2\mathrm{tr}\big(\mathbf{U}\mathbf{G}^{\mathsf{T}}\big) + \nu \tag{5.7}$$

$$\text{s.t. } \mathrm{tr}(\mathbf{W}) < \nu, \tag{5.8}$$

$$\mathbf{W} > 0, \tag{5.9}$$

$$\begin{bmatrix} \mathbf{W} & \mathbf{U}\mathbf{R} \\ * & \mathbf{1} \end{bmatrix} > 0, \tag{5.10}$$

$$\mathbf{P} > 0, \tag{5.11}$$

$$\begin{bmatrix} \bar{\rho}\mathbf{P} & \mathbf{A}^{\mathsf{T}}\mathbf{P} \\ * & \bar{\rho}\mathbf{P} \end{bmatrix} > 0, \tag{5.12}$$

where $\mathbf{H} = \mathbf{R}\mathbf{R}^{\mathsf{T}}$ and $\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$. The SVD is used to compute $\mathbf{R}$ in the experiments presented in this chapter.

Since both $\mathbf{A}$ and $\mathbf{P}$ are unknown, (5.6) includes a BMI constraint. The optimization problem is therefore nonconvex and NP-hard [48]. One method to find a locally optimal solution is outlined in [91]. First, assume $\mathbf{P}$ is constant with an initial guess of $\mathbf{P} = \mathbf{1}$ and solve (5.7)–(5.12), which is now an SDP. Then, hold the remaining variables constant using the solution from that optimization, and solve a feasibility problem for $\mathbf{P}$. Repeat this process until the cost function stops changing significantly. Although this approach is rather simple, it does result in an AS Koopman system with reasonable prediction error.

5.3 System Norm Regularization

While constraining the asymptotic stability of the identified Koopman system helps ensure that the system's predictions are usable, it does not consider the input-output properties of the system. A system norm like the $\mathcal{H}_2$ norm or the $\mathcal{H}_\infty$ norm can be used to regularize the system's gain, while also ensuring asymptotic stability for any regularization coefficient. Specifically, the existence of a finite $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norm guarantees the asymptotic stability of the resulting LTI system [36]. Regularizing using the $\mathcal{H}_2$ norm can be thought of as penalizing the average system gain over all frequencies, while using the $\mathcal{H}_\infty$ norm penalizes the worst-case system gain. As such, the use of the $\mathcal{H}_\infty$ norm as a regularizer is explored next. Weighted $\mathcal{H}_\infty$ norms are also considered as regularizers, which allow the regularization problem to be tuned in the frequency domain with weighting functions.

5.3.1 $\mathcal{H}_\infty$ norm regularization

Since approximating the Koopman matrix amounts to identifying a discrete-time LTI system, it is natural to consider a system norm as a regularizer rather than a matrix norm. With $\mathcal{H}_\infty$ norm regularization, the cost function associated with the regression problem is

$$J(\mathbf{U};\alpha) = \|\mathbf{\Theta}_+ - \mathbf{U}\mathbf{\Psi}\|_\mathsf{F}^2 + \alpha\|\boldsymbol{\mathcal{G}}\|_\infty, \tag{5.13}$$

where α is the regularization coefficient. To integrate the $\mathcal{H}_\infty$ norm into the regression problem, its LMI formulation must be considered. Recall from Section 2.4.5.2 that the inequality $\|\boldsymbol{\mathcal{G}}\|_\infty < \gamma$ holds if and only if there exists a $\mathbf{P} = \mathbf{P}^\mathsf{T} > 0$ such that [31, §4.2.2]

$$\begin{bmatrix} \mathbf{P} & \mathbf{AP} & \mathbf{B} & \mathbf{0} \\ * & \mathbf{P} & \mathbf{0} & \mathbf{PC}^\mathsf{T} \\ * & * & \gamma\mathbf{1} & \mathbf{D}^\mathsf{T} \\ * & * & * & \gamma\mathbf{1} \end{bmatrix} > 0. \tag{5.14}$$

The full EDMD optimization problem with $\mathcal{H}_\infty$ regularization is

$$\min\ J(\mathbf{U},\nu,\mathbf{W},\gamma,\mathbf{P};\alpha) = c - 2\mathrm{tr}\big(\mathbf{UG}^\mathsf{T}\big) + \nu + \frac{\alpha}{q}\gamma \tag{5.15}$$

$$\text{s.t. } \mathrm{tr}(\mathbf{W}) < \nu, \tag{5.16}$$

$$\mathbf{W} > 0, \tag{5.17}$$

$$\begin{bmatrix} \mathbf{W} & \mathbf{UR} \\ * & \mathbf{1} \end{bmatrix} > 0, \tag{5.18}$$

$$\mathbf{P} > 0, \tag{5.19}$$

$$\begin{bmatrix} \mathbf{P} & \mathbf{AP} & \mathbf{B} & \mathbf{0} \\ * & \mathbf{P} & \mathbf{0} & \mathbf{PC}^\mathsf{T} \\ * & * & \gamma\mathbf{1} & \mathbf{D}^\mathsf{T} \\ * & * & * & \gamma\mathbf{1} \end{bmatrix} > 0, \tag{5.20}$$

where $\mathbf{H} = \mathbf{RR}^\mathsf{T}$ and $\mathbf{U} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$.

Like the asymptotic stability constraint proposed in Section 5.2, (5.20) includes a BMI constraint in terms of the unknowns $\mathbf{P}$ and $\mathbf{A}$. As such, the optimization problem in (5.15)–(5.20) is nonconvex and NP-hard [48]. However, it can be solved using the same iterative procedure described in Section 5.2 [91].

5.3.2 Weighted $\mathcal{H}_\infty$ norm regularization

The $\mathcal{H}_\infty$ norm used in (5.15) can be weighted by cascading $\boldsymbol{\mathcal{G}}$ with another LTI system, $\boldsymbol{\mathcal{G}}^{\mathrm{w}}$, as shown in Figure 5.2. For example, choosing $\boldsymbol{\mathcal{G}}^{\mathrm{w}}$ to be a high-pass filter penalizes system gains at high frequencies. Weights can be cascaded before or after $\boldsymbol{\mathcal{G}}$ to weight the inputs, outputs, or both. Recall that multiple-input multiple-output (MIMO) LTI systems do not commute, so introducing the weight before or after the Koopman system results in a different weighted system.

Consider the weight

$$\boldsymbol{\mathcal{G}}^{\mathrm{w}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathrm{w}} & \mathbf{B}^{\mathrm{w}} \\ \hline \mathbf{C}^{\mathrm{w}} & \mathbf{D}^{\mathrm{w}} \end{array}\right], \tag{5.21}$$

with state $\boldsymbol{\vartheta}^{\mathrm{w}}$, input $\boldsymbol{\upsilon}^{\mathrm{w}}$, and output $\boldsymbol{\zeta}^{\mathrm{w}}$. Cascading $\boldsymbol{\mathcal{G}}^{\mathrm{w}}$ after $\boldsymbol{\mathcal{G}}$ yields the augmented state-space system

$$\begin{bmatrix} \boldsymbol{\vartheta}_{k+1} \\ \boldsymbol{\vartheta}^{\mathrm{w}}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B}^{\mathrm{w}}\mathbf{C} & \mathbf{A}^{\mathrm{w}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\vartheta}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{w}}_{k} \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{B}^{\mathrm{w}}\mathbf{D} \end{bmatrix} \boldsymbol{\upsilon}_k, \tag{5.22}$$

$$\boldsymbol{\zeta}^{\mathrm{w}}_k = \begin{bmatrix} \mathbf{D}^{\mathrm{w}}\mathbf{C} & \mathbf{C}^{\mathrm{w}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\vartheta}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{w}}_{k} \end{bmatrix} + \mathbf{D}^{\mathrm{w}}\mathbf{D}\, \boldsymbol{\upsilon}_k. \tag{5.23}$$

Minimizing the $\mathcal{H}_\infty$ norm of the augmented system

$$\boldsymbol{\mathcal{G}}^{\mathrm{w}}\boldsymbol{\mathcal{G}} \overset{\min}{\sim} \left[\begin{array}{cc|c} \mathbf{A} & \mathbf{0} & \mathbf{B} \\ \mathbf{B}^{\mathrm{w}}\mathbf{C} & \mathbf{A}^{\mathrm{w}} & \mathbf{B}^{\mathrm{w}}\mathbf{D} \\ \hline \mathbf{D}^{\mathrm{w}}\mathbf{C} & \mathbf{C}^{\mathrm{w}} & \mathbf{D}^{\mathrm{w}}\mathbf{D} \end{array}\right] \tag{5.24}$$

is equivalent to minimizing the weighted $\mathcal{H}_\infty$ norm of the original system. The choice of weighting function used can be viewed as another hyperparameter in the regression problem.

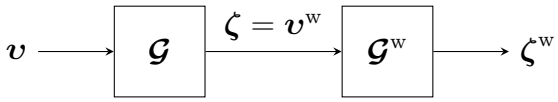


Figure 5.2: Series interconnection of Koopman system $\boldsymbol{\mathcal{G}}$ and weight $\boldsymbol{\mathcal{G}}^{\mathrm{w}}$.

Weighting the regression problem in the frequency domain comes at the cost of increasing the dimension of the optimization problem. When cascading the weight before $\boldsymbol{\mathcal{G}}$, the dimension of $\boldsymbol{\vartheta}^{\mathrm{w}}$ scales with the dimension of $\boldsymbol{\upsilon}$. When cascading after $\boldsymbol{\mathcal{G}}$, the dimension of $\boldsymbol{\vartheta}^{\mathrm{w}}$ scales with the dimension of $\boldsymbol{\zeta}$. In the regression problems considered here, only post-weighting is considered, since $\boldsymbol{\zeta}$ has a much smaller dimension.

5.4 Experimental Example: Aircraft Fatigue Testing Platform

The effectiveness of the proposed asymptotic stability constraint is demonstrated using experimental data collected from the Fatigue Structural Testing Enhancement Research (FASTER) platform at the National Research Council Canada [92], which is used for aircraft fatigue testing research. The purpose of this simple example is to demonstrate that Koopman models with eigenvalues near the unit circle can have diverging predictions due to the accumulation of numerical error.

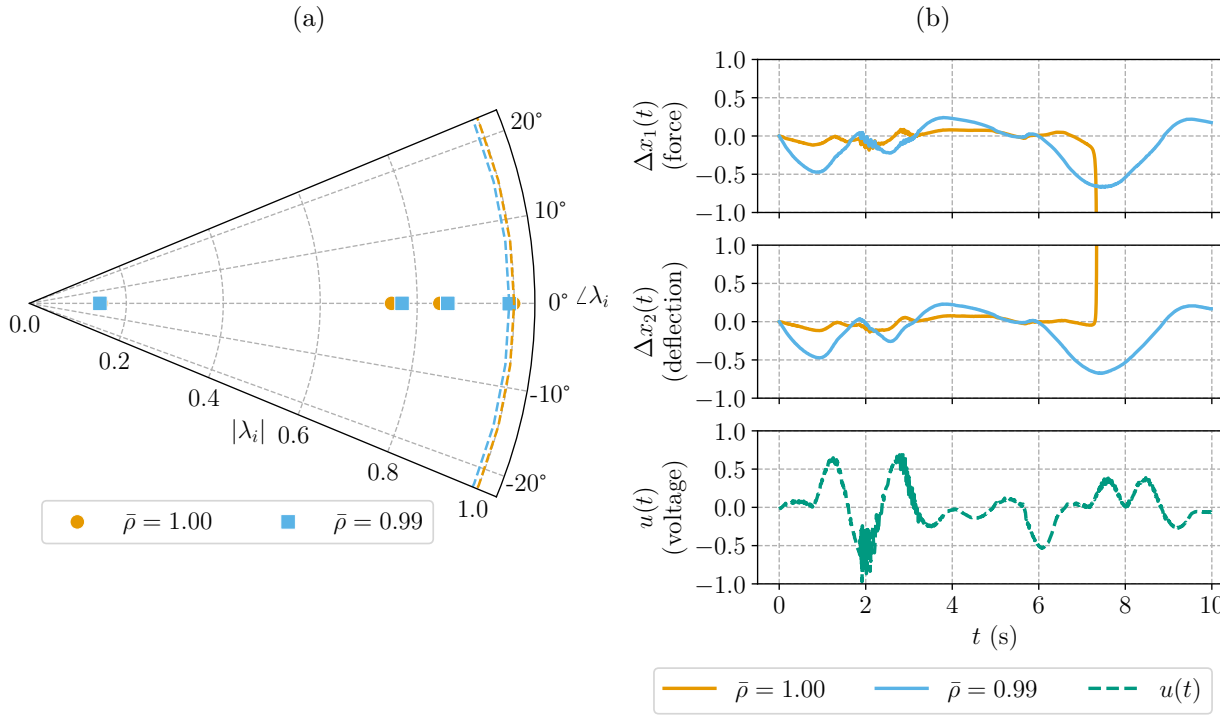


Figure 5.3: (a) Eigenvalues and spectral radius constraints of Koopman **A** matrices approximated from the FASTER dataset. The eigenvalues of **A** satisfy their respective spectral radius constraints. Additionally, lowering the spectral radius constraint from $\bar{\rho} = 1.00$ to $\bar{\rho} = 0.99$ does not significantly alter the eigenvalues. (b) Multi-step prediction error of Koopman systems approximated from the FASTER dataset. All units are normalized. States are recovered and re-lifted between prediction timesteps. Although both systems are AS, only the system with $\bar{\rho} = 0.99$ is usable in practice, as the other system's response diverges due to the accumulation of numerical error.

In the fatigue structural testing dataset used in this section, the structure under test is an aluminum composite beam. During a test, the FASTER platform applies a force to this beam using a hydraulic actuator, which is controlled by a voltage. This input, the applied force, and the structure's deflection are recorded at 128 Hz. The actuator voltage is determined by a linear controller designed to track a reference force profile. In this case, the dynamics of the controller are neglected and the actuator voltage is considered to be

exogenous. Of the 21 275 samples in the dataset, half are used for training and half are used for testing. All states and inputs in the FASTER dataset are normalized. A photograph of this experimental setup can be found in [92].

The Koopman lifting functions chosen for the FASTER platform are first- and second-order monomials. The full lifting procedure consists of several steps to improve numerical conditioning. First, all states and inputs are normalized so that they do not grow when passed through the monomial lifting functions. Then, all first- and second-order monomials of the state and input are computed. Finally, the lifted states are standardized to ensure that they are evenly weighted in the regression. To standardize the lifted states, their means are subtracted and they are rescaled to have unit variance [57, §4.6.6].

Given that the true FASTER system is AS, it is crucial to ensure that the identified Koopman system shares this property. To demonstrate the impact of the asymptotic stability constraint, Koopman matrices with maximum spectral radii of $\bar{\rho} = 1.00$ and $\bar{\rho} = 0.99$ are computed and compared to an unconstrained Koopman matrix computed with standard EDMD. Figure 5.3a shows the eigenvalues of the two constrained Koopman $\mathbf{A}$ matrices. In both cases, the eigenvalues indicate that the systems obey their respective maximum spectral radii. However, the multistep prediction errors of the two Koopman systems in Figure 5.3b show that only the system with the spectral radius constraint of $\bar{\rho} = 0.99$ is usable in practice. The other system produces prediction errors that diverge to infinity due to the accumulation of numerical error. The Koopman matrix without asymptotic stability constraint behaves identically to the matrix with maximum spectral radius $\bar{\rho} = 1.00$, and is therefore not shown in Figure 5.3. By introducing a small amount of conservatism in the spectral radius constraint, the identified Koopman system is rendered AS in the face of accumulated numerical error.

In the following section, a more complex example is used to demonstrate the limitations of the spectral radius constraint and the advantages of the $\mathcal{H}_\infty$ norm regularization approach.

5.5 Experimental Example: Soft Robot

The unique advantages of the $\mathcal{H}_\infty$ norm regularizer are demonstrated using the soft robot dataset published alongside [93] and [16]. Identifying a Koopman representation of a soft robot arm is a particularly interesting problem, as its dynamics are not easily modelled from first principles. In this section, unregularized EDMD and Tikhonov-regularized EDMD [1, 56], two standard Koopman matrix approximation methods, are compared with the EDMD and DMDc versions of the asymptotic stability constraint from Section 5.2 and the $\mathcal{H}_\infty$ norm regularizer presented in Section 5.3. The Koopman systems identified using these regression

methods are analyzed in terms of their prediction errors, system properties, and numerical conditioning. In particular, the limitations of the asymptotic stability constraint compared to the $\mathcal{H}_\infty$ norm regularizer are discussed.

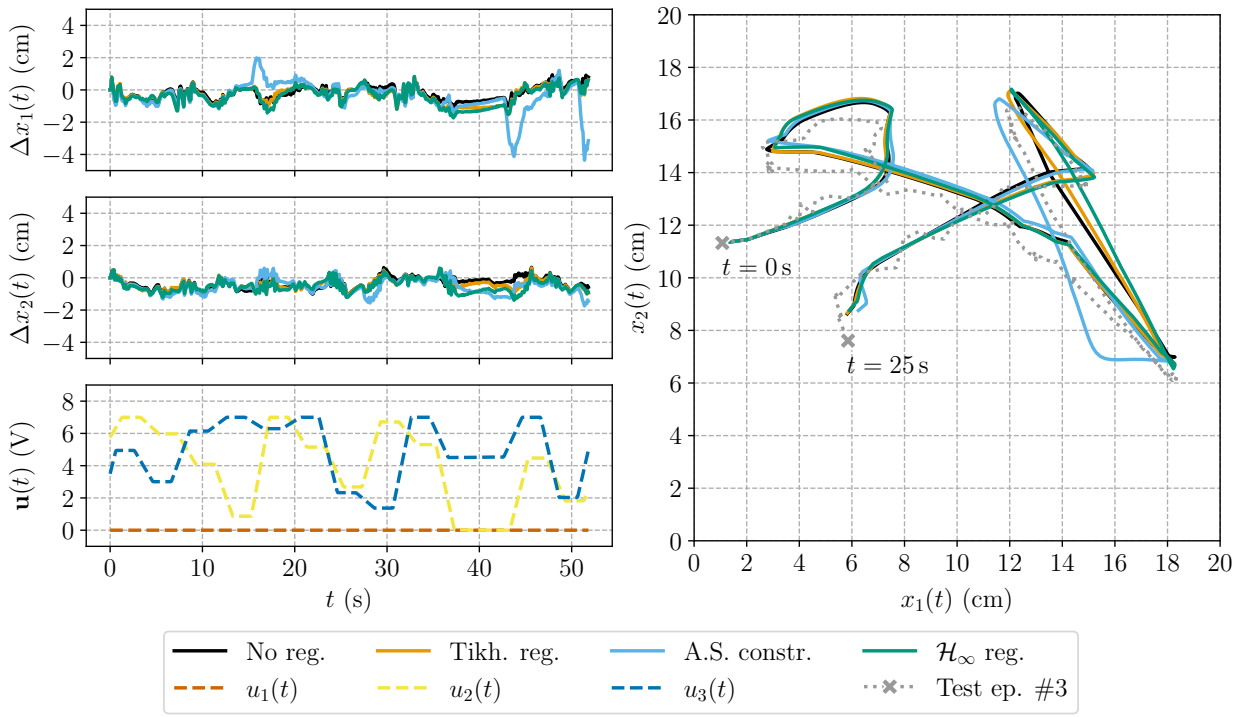


Figure 5.4: Multi-step prediction error and trajectory plot of the third test episode for Koopman systems approximated from the soft robot dataset. States are recovered and re-lifted between prediction timesteps. All Koopman systems have comparable prediction errors, with the exception of two large error spikes in the system with the asymptotic stability constraint.

The soft robot under consideration consists of two flexible segments with a laser pointer mounted at the end. The laser pointer projects a dot onto a board positioned below the robot. The two states of the system are the Cartesian coordinates of the dot on the board, as measured by a camera. The soft robot arm is actuated by three pressure regulators, each controlled by a voltage. The dot position and control voltages are recorded at 12 Hz. Thirteen training episodes and four test episodes were recorded in this manner. The third test episode is shown in Figure 5.4. Photographs of this experimental setup, along with additional details, can be found in [93] and [16].

The lifting functions chosen for the soft robot system consist of a time delay step, followed by a third-order monomial transformation. As with the FASTER dataset in Section 5.2, the states and inputs are first normalized. Then, the states and inputs are augmented with their delayed versions, where the delay period is one timestep. Next, all first-, second-, and third-order monomials are computed. Finally, the lifted states are standardized. Since the

time delay step occurs before the monomial lifting step, cross-terms including delayed and non-delayed states and inputs occur in the lifted state.

Section 5.5.1 discusses the behaviour of the proposed asymptotic stability constraint and $\mathcal{H}_\infty$ norm regularizer in the context of EDMD. Section 5.5.2 discusses the DMDc case, and demonstrates that the DMDc approach attains similar results to EDMD with fewer computational resources.

5.5.1 EDMD results

Unregularized EDMD and Tikhonov-regularized EDMD are used as baselines for comparison with the proposed regression methods. Tikhonov regularization improves the numerical conditioning of $\mathbf{U}$ by penalizing its squared Frobenius norm [1, 56]. The regularizers used in

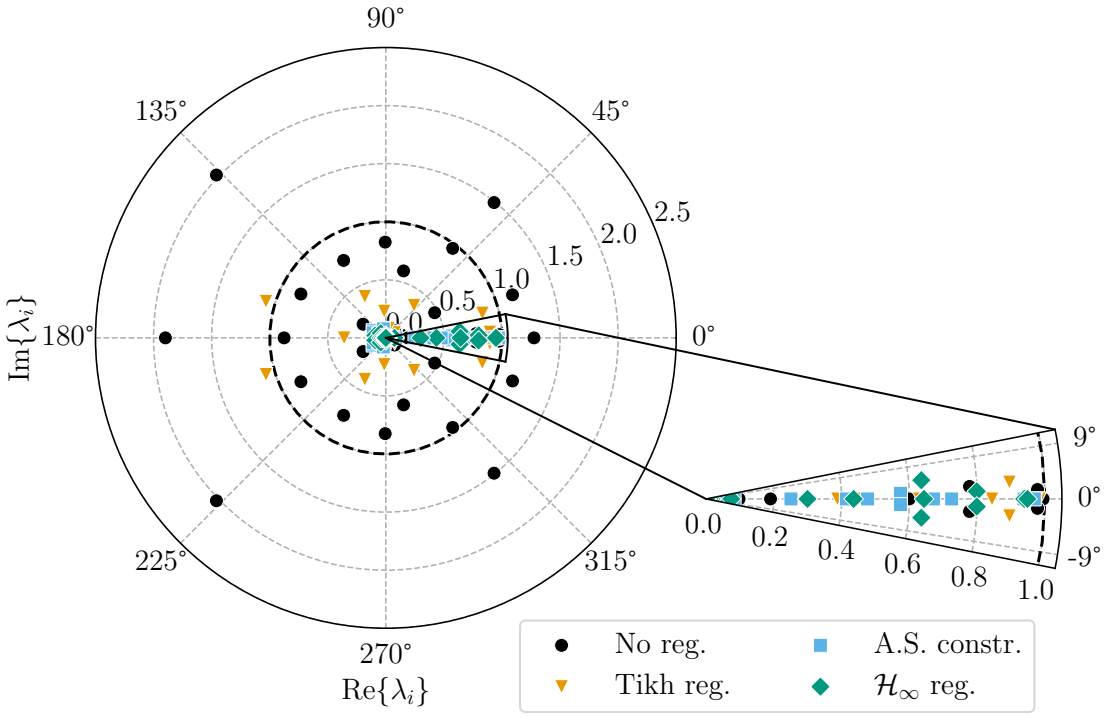


Figure 5.5: Eigenvalues of Koopman $\mathbf{A}$ matrices approximated from the soft robot dataset. EDMD without regularization and EDMD with Tikhonov regularization both identify unstable systems, while the asymptotic stability constraint and $\mathcal{H}_\infty$ norm regularizer yield AS Koopman systems.

this section have coefficients of $\alpha = 7.5 \times 10^{-3}$, while the asymptotic stability constraint has a maximum spectral radius of $\bar{\rho} = 0.999$. Figure 5.4 shows the multistep prediction errors of the four Koopman systems for the third test episode of the dataset. The prediction errors are comparable for all four Koopman systems, aside from two large error spikes produced by the system with the asymptotic stability constraint.

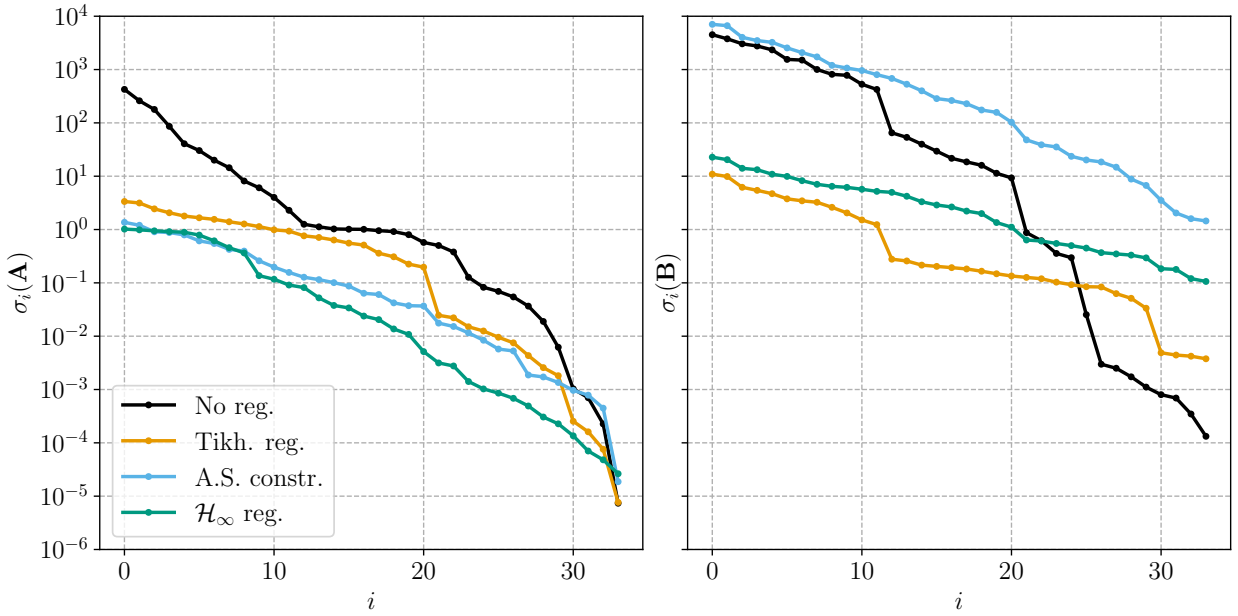


Figure 5.6: Singular values of Koopman **A** and **B** matrices approximated from the soft robot dataset plotted on a logarithmic scale. The **A** and **B** matrices computed using unregularized EDMD both have large singular values. EDMD with Tikhonov regularization decreases the singular values of both matrices, but does not yield an AS system. With an asymptotic stability constraint, the singular values of **A** are reduced, but the singular values of **B** are unaffected and remain large. Using an $\mathcal{H}_\infty$ norm regularizer reduces the singular values of both **A** and **B** significantly, yielding a better-conditioned Koopman matrix.

One way to compare the resulting Koopman systems is to analyze the eigenvalues of their **A** matrices. Figure 5.5 shows that unregularized EDMD and Tikhonov-regularized EDMD produce unstable Koopman systems, even though, as shown in Figure 5.8, the multistep prediction errors do not happen to diverge in any test episodes. As expected, EDMD with an $\mathcal{H}_\infty$ norm regularizer and EDMD with an asymptotic stability constraint both yield AS Koopman systems.

While the Koopman system identified with an asymptotic stability constraint is indeed AS, the system is not well-conditioned. To see this, consider Figure 5.6, which shows the singular values of the Koopman **A** and **B** matrices. These singular values indicate the sizes of the entries in each matrix. With unregularized EDMD, both matrices have singular values on the order of 10^3. Using Tikhonov regularization decreases the singular values of both **A** and **B**, though it still yields an unstable system. Constraining the spectral radius of **A** greatly reduces the singular values of **A** but increases the singular values of **B**. The numerical conditioning of this Koopman system is arguably worse, as the **A** and **B** matrices contain entries of drastically different scales. Regularizing using the $\mathcal{H}_\infty$ norm resolves this problem, as it reduces the singular values in both matrices, yielding a better-conditioned AS

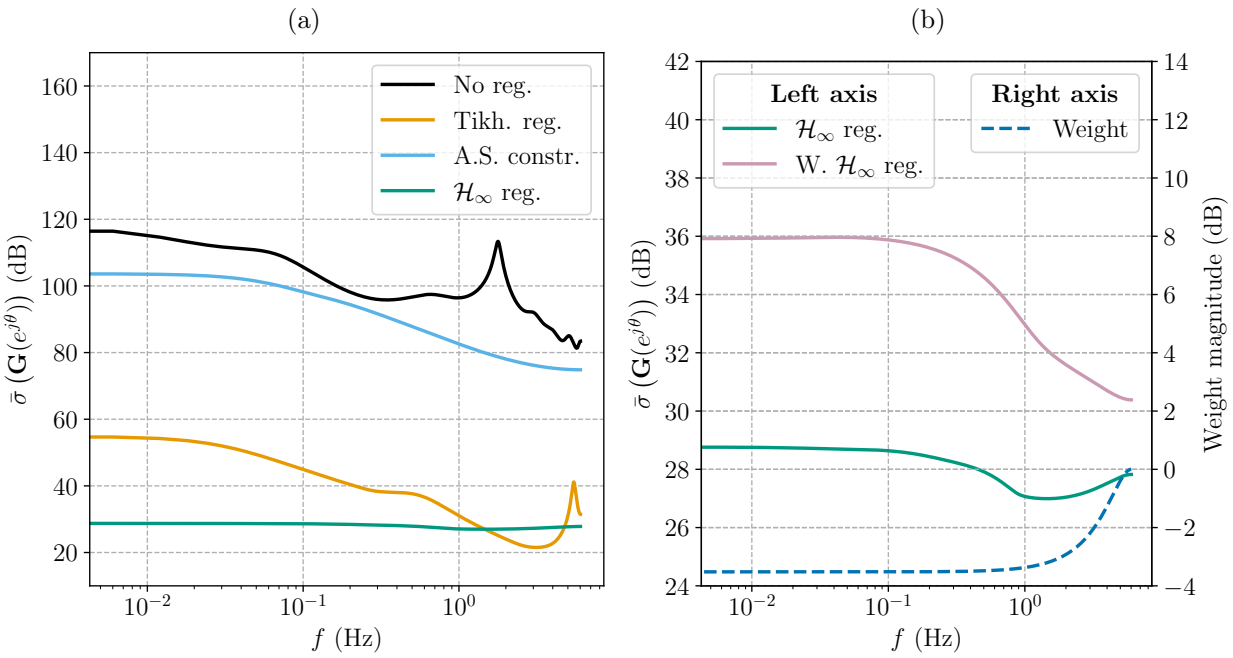


Figure 5.7: (a) Bode plot of Koopman systems approximated from the soft robot dataset. The Koopman system identified with unregularized EDMD has high gain, with a resonant peak at a high frequency. EDMD with Tikhonov regularization reduces the system's gain, but identifies an unstable system and retains an undesirable high-frequency resonant peak in the frequency response. Constraining the asymptotic stability of the system does not significantly reduce the gain compared to the unregularized system. Penalizing the $\mathcal{H}_\infty$ norm of the Koopman system reduces its gain at all frequencies without compromising prediction error. (b) Bode plot of unweighted and weighted Koopman systems approximated from the soft robot dataset, along with weighting function. The dashed line representing the weighting function uses the right axis, while the solid lines use the left axis.

Koopman system with similar prediction error. The key takeaway is that constraining the spectral radius of $\mathbf{A}$ is not sufficient to guarantee a well-conditioned Koopman matrix, as the constraint does not directly impact $\mathbf{B}$. Using the $\mathcal{H}_\infty$ norm as a regularizer considers the system as a whole, thus impacting both $\mathbf{A}$ and $\mathbf{B}$, and reducing their entries to reasonable sizes.

Another way to compare the identified Koopman systems is by looking at their frequency responses, which can be found by plotting the maximum singular value of the transfer matrix at each frequency. Figure 5.7a shows the magnitude responses of the four Koopman systems, and paints a similar picture to Figure 5.6. Unregularized EDMD yields a Koopman system with very high gain and a resonant peak in the upper frequency range. Incorporating Tikhonov regularization reduces the system's gain, but retains the resonant peak at a high frequency. Constraining the asymptotic stability of the system does not significantly impact the system's gain, which, given the large singular values of $\mathbf{B}$ in Figure 5.6, is not surprising.

However, regularizing using the $\mathcal{H}_\infty$ norm directly penalizes the peak of the Bode plot in Figure 5.7a, yielding a system with significantly lower gain and similar prediction error.

The $\mathcal{H}_\infty$ norm regularizer can be expanded upon by weighting the $\mathcal{H}_\infty$ norm using a high-pass filter. This weighting function penalizes high gains at high frequencies while allowing higher gains at low frequencies. Penalizing gain at high frequencies is motivated by the fact that relevant system dynamics typically occupy low frequencies, while high frequencies are corrupted by measurement noise. Furthermore, causal physical systems have frequency responses that roll off as frequency grows, since it is unrealistic for a system to have infinite gain at infinitely high frequencies. Figure 5.7b demonstrates the impact of weighting the $\mathcal{H}_\infty$ norm regularizer with a high-pass filter that has a zero at 4 Hz and a pole slightly below 6 Hz. This weighted regularizer yields a Koopman system with high gain at low frequencies and decreasing gain at high frequencies.

Note that Figure 5.7 shows the frequency response of the Koopman system in the lifted space, which is not the same as the "frequency response of the nonlinear system." Since the ultimate goal is to design linear controllers in the lifted space, only the frequency response of the Koopman system in the lifted space and the corresponding $\mathcal{H}_\infty$ norm are relevant.

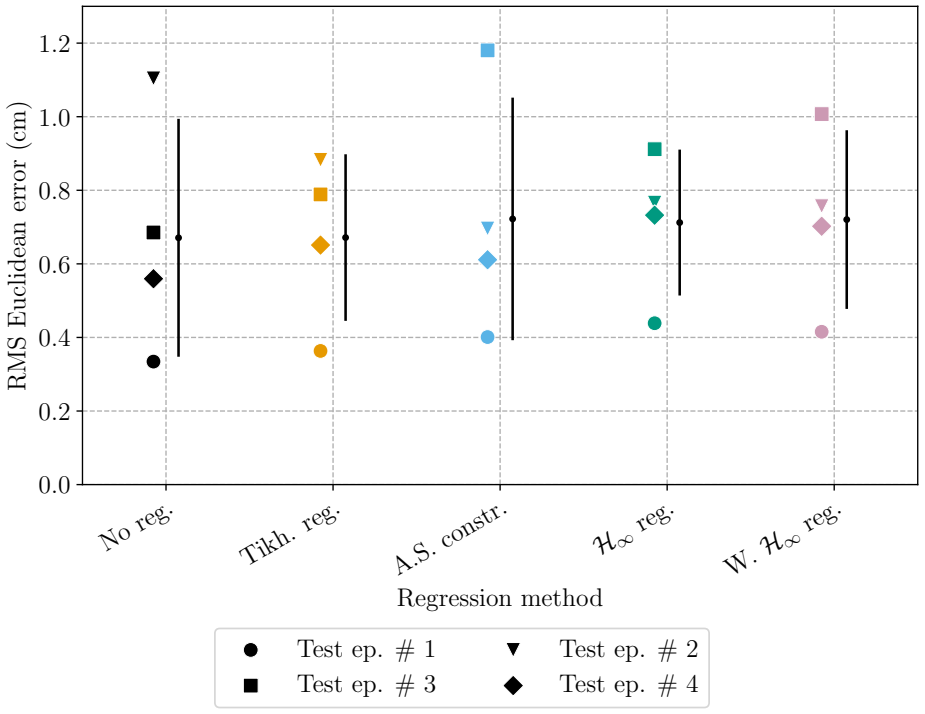


Figure 5.8: RMS Euclidean errors of Koopman systems approximated from the soft robot dataset. Error bars indicate mean and standard deviation of RMS error over the four test episodes. In terms of mean error, all identified Koopman systems perform similarly well. However, the system identified using $\mathcal{H}_\infty$ norm regularizer performed more consistently throughout the test set.

In Figure 5.8, the multistep prediction errors of the five identified Koopman systems are compared across each episode in the test set. Given that the laser pointer dot projected by the soft robot moves within a circle of radius 10 cm, it's practically meaningless to distinguish each method based on mean prediction error alone. However, the distribution of each identified system's prediction errors across the test set clearly highlights the importance of regularization. The Tikhonov regularizer, $\mathcal{H}_\infty$ norm regularizer, and weighted $\mathcal{H}_\infty$ norm regularizer lead to systems with smaller standard deviations in the RMS error over the test set. In contrast, EDMD without regularization and EDMD with an asymptotic stability constraint lead to systems that perform inconsistently over the test set. Comparing standard deviations indicates that the regularized systems generalize better to previously unseen data, with the $\mathcal{H}_\infty$ norm regularizer performing most consistently over the four test episodes.

The results in this section highlight the useful properties of the proposed $\mathcal{H}_\infty$ norm regularizers. While all systems have comparable RMS prediction errors on the test set, unregularized EDMD results in an unstable, poorly conditioned Koopman system with large gain. While Tikhonov regularization improves numerical conditioning, the resulting system is still unstable. Conversely, constraining the asymptotic stability of the system does not improve its numerical properties. Only the $\mathcal{H}_\infty$ norm regularizers guarantee asymptotic stability while improving the numerical conditioning of the system. These key results are summarized in Table 5.1, which also highlights the difference between the unweighted and weighted $\mathcal{H}_\infty$ norm regularizers. Weighting the $\mathcal{H}_\infty$ norm regularizer further improves the condition numbers of $\mathbf{A}$ and $\mathbf{B}$.

Table 5.1: Comparison of regression methods through the condition numbers of their Koopman matrices and asymptotic stability guarantees. Only the $\mathcal{H}_\infty$ regularizers guarantee asymptotic stability while significantly improving the condition number of the Koopman matrices. In this case, weighting the $\mathcal{H}_\infty$ norm further improves cond($\mathbf{A}$) and cond($\mathbf{B}$).

Regression method	cond($\mathbf{A}$)	cond($\mathbf{B}$)	Asymptotic stability
no regularization	5.77×10^7	3.40×10^7	✗
Tikhonov regularization	4.39×10^5	2.90×10^3	✗
asymptotic stability constraint	7.32×10^4	4.87×10^3	✓
$\mathcal{H}_\infty$ regularization	3.87×10^4	2.14×10^2	✓
weighted $\mathcal{H}_\infty$ regularization	1.69×10^3	5.43×10^1	✓

5.5.2 DMDc results

The DMDc projection in (4.32) defines a new Koopman system,

$$\hat{\boldsymbol{\vartheta}}_{k+1} = \hat{\mathbf{A}}\hat{\boldsymbol{\vartheta}}_k + \hat{\mathbf{B}}\boldsymbol{\upsilon}_k, \tag{5.25}$$

$$\boldsymbol{\zeta}_k = \hat{\mathbf{C}}\hat{\boldsymbol{\vartheta}}_k + \mathbf{D}\boldsymbol{\upsilon}_k, \tag{5.26}$$

where $\hat{\mathbf{C}} = \mathbf{C}\hat{\mathbf{Q}}$. The asymptotic stability constraint from Section 5.2 and the $\mathcal{H}_\infty$ norm regularizer from Section 5.3 are now applied to this system, and the resulting Koopman systems are compared with their EDMD counterparts. The same dataset and experimental setup as Section 5.3 is used here.

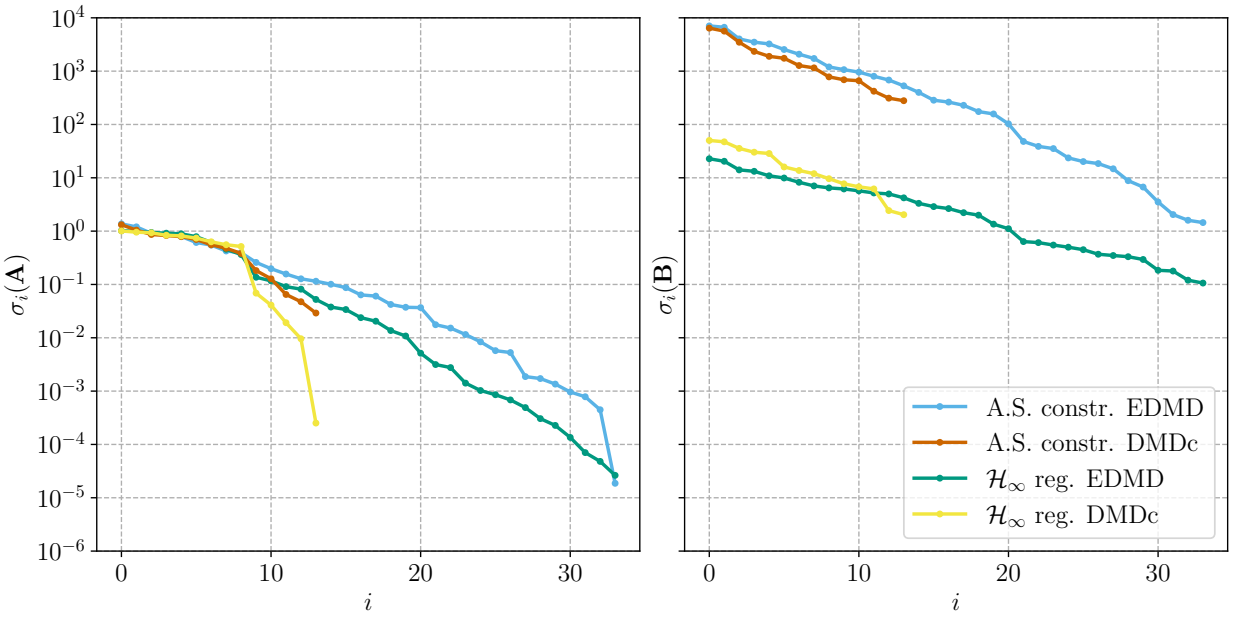


Figure 5.9: Singular values of Koopman **A** and **B** matrices approximated from the soft robot dataset using EDMD and DMDc regressors. Singular values smaller than 10^{-12} are not shown. Note the logarithmic scale. While the EDMD methods retain all 34 singular values, the DMDc methods truncate all but the first 14. The singular values retained by the DMDc methods are close to the corresponding singular values computed by the EDMD methods.

An important decision in the DMDc algorithm is the choice of singular value truncation method. Optimal hard singular value truncation [34] is used to determine $\hat{r}$, while $\tilde{r}$ is left at full rank. For the soft robot dataset, the optimal hard truncation algorithm retains only 14 of the 34 singular values of **A**. Figure 5.9 demonstrates that the DMDc methods indeed reduce the dimension of the problem, while also showing that the remaining singular values are close to their EDMD counterparts. Figure 5.10a shows that the frequency responses of the original Koopman systems are preserved by the DMDc methods. In spite of their reduced dimension,

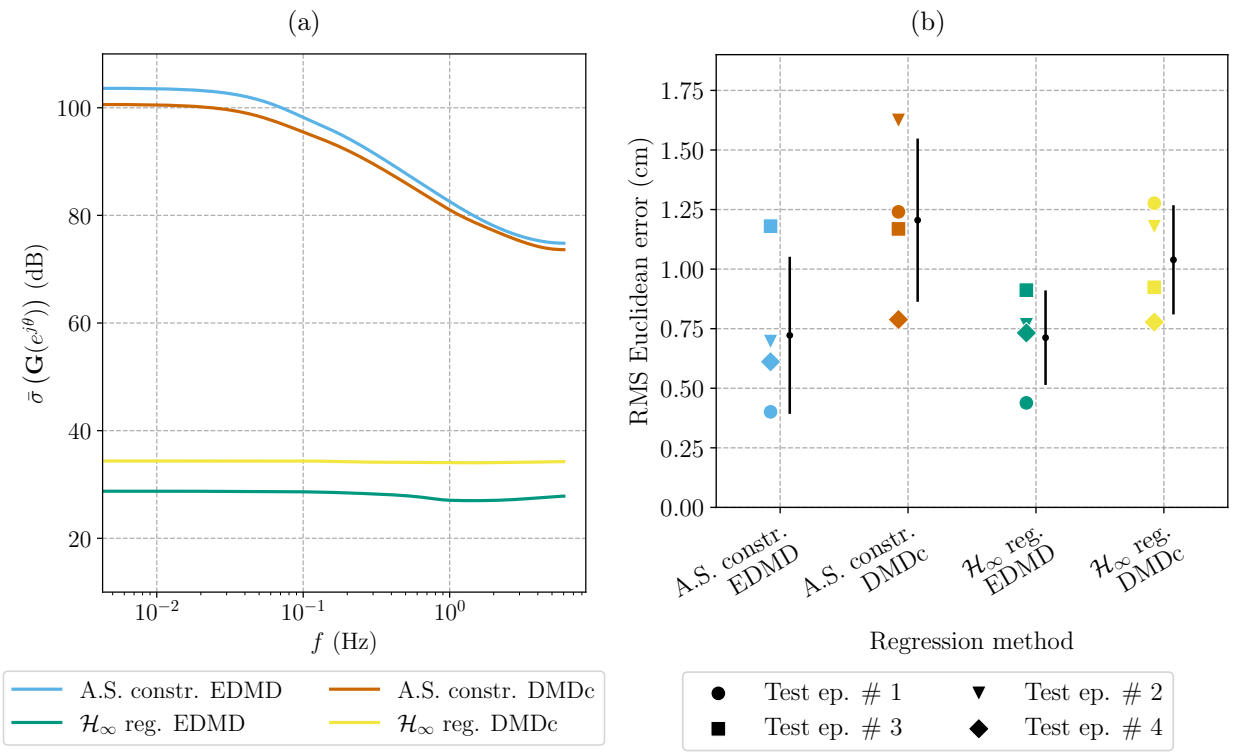


Figure 5.10: (a) Bode plots of Koopman systems approximated from the soft robot dataset using EDMD and DMDc regressors. The DMDc methods preserve the frequency responses of the corresponding systems identified with the EDMD methods. (b) RMS Euclidean errors of Koopman systems approximated from the soft robot dataset using EDMD and DMDc regressors. Error bars indicate standard deviation of RMS error over the four test episodes. Since the DMDc methods identify reduced-order Koopman models of the system, they have larger mean errors. However, the $\mathcal{H}_\infty$ norm regularizer still significantly reduces the standard deviation.

the Koopman systems identified with DMDc retain their frequency domain properties.

In Figure 5.10b, the RMS Euclidean errors of the EDMD and DMDc methods with asymptotic stability constraints and $\mathcal{H}_\infty$ norm regularization are summarized. Since the Koopman systems identified by the DMDc methods are of a lower order, their mean prediction error is higher than that of the EDMD methods. However, the $\mathcal{H}_\infty$ norm regularizer retains its benefit of improving prediction consistency. Furthermore, as demonstrated by Figure 5.10a, the frequency response properties of the EDMD regression methods are preserved by the reduced-order DMDc models.

Note that, in one case, the Koopman system identified using stability-constrained DMDc diverged due to poor numerical conditioning. This short segment of the dataset was omitted throughout the chapter to allow for a more fair comparison between regression methods. This finding highlights the advantages of the $\mathcal{H}_\infty$ regularization method in identifying numerically well-conditioned Koopman matrices.

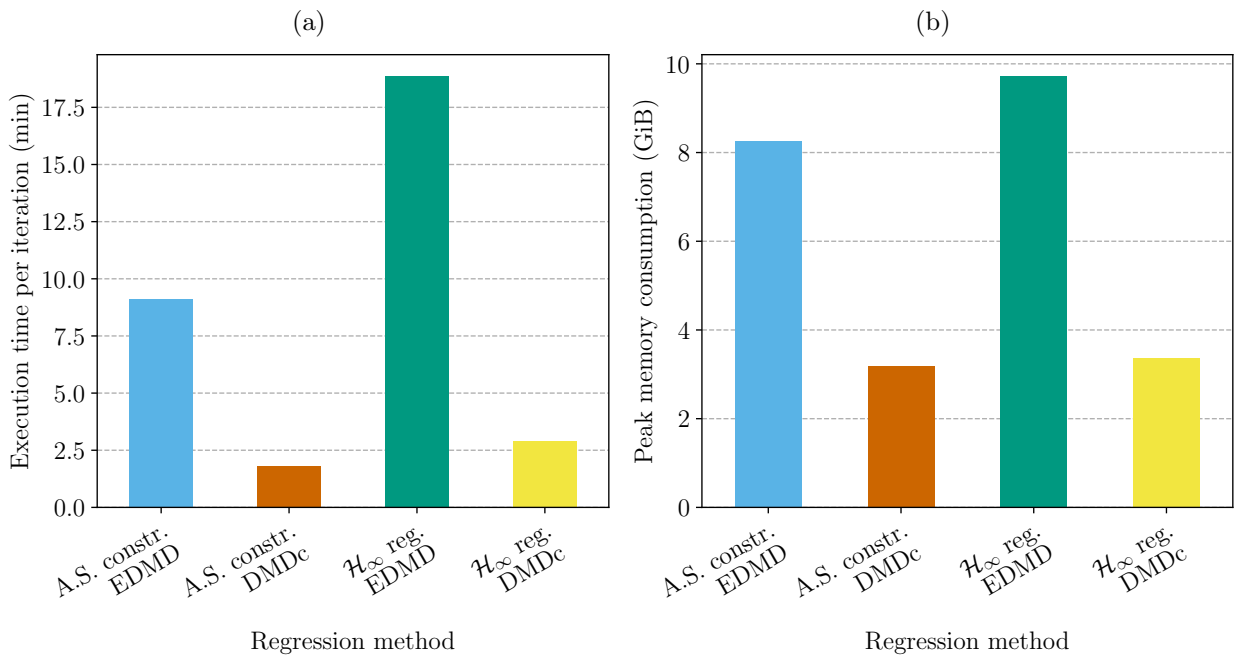


Figure 5.11: (a) Execution time per iteration and (b) peak memory consumption of EDMD and DMDc regression methods using the soft robot dataset. The DMDc methods run significantly faster and consume less memory than the EDMD methods. Tests were run on a desktop computer with an Intel Core i7-10700K processor using the `MOSEK` solver.

The most important advantage of the DMDc regression methods is their computational savings when many lifting functions are required. The long execution times of the EDMD methods, along with their high memory consumption, make cross-validation impractical. Figure 5.11 demonstrates the significant resource savings provided by the DMDc methods in both execution time and peak memory consumption. Memory consumption is of particular importance when running multiple instances of a regressor in a multiprocess cross-validation scheme. The DMDc regression methods presented provide significant computational savings while still retaining the frequency-domain characteristics of their EDMD counterparts. In spite of their higher mean prediction error, it is often worthwhile to leverage them for Koopman operator identification, particularly when hyperparameter optimization is a priority.

5.6 Conclusion

Approximating the Koopman matrix using linear regression proves challenging as lifting function complexity increases. Even small problems can become ill-conditioned when many lifting functions are required for an accurate fit. Viewing the problem from a systems perspective, where system inputs pass through dynamics and lead to outputs, provides multiple

avenues to enforce asymptotic stability and penalize large input-output gains in the system, thus ensuring improved numerical conditioning. In particular, regularizing the regression problem with the $\mathcal{H}_\infty$ norm provides the opportunity to tune the regularization process in the frequency domain using weighting functions. The significant performance savings presented by the DMDc-based regression methods allow the $\mathcal{H}_\infty$ norm regularizer to be applied to much larger systems while still remaining tractable. This chapter proposes two new methods to approximate the Koopman operator while enforcing desired system properties. The first constrains the asymptotic stability of the Koopman system, while the second regularizes the $\mathcal{H}_\infty$ norm of the Koopman system. Both methods ensure asymptotic stability, while the second method also yields Koopman systems with lower gain.

The nonconvex optimization problems required to use the asymptotic stability constraint and $\mathcal{H}_\infty$ norm regularizers limit their applicability to practical problems. Future research will address this limitation by making use of more efficient BMI solution methods, including changes of variables [76–78, 94], Iterative Convex Overbounding [95], and branch-and-bound methods [96]. Although the use of the $\mathcal{H}_\infty$ norm [31, §4.2] as a regularizer is explored in this chapter, any system norm, like the $\mathcal{H}_2$ norm [31, §4.3] or a mixed $\mathcal{H}_2$ norm [31, §4.4], can be used. The unique properties of system norms prove useful in addressing the numerical challenges associated with approximating the Koopman operator from data, and will be explored further in future work.

While this chapter covers the identification of AS systems, the next chapter presents methods to identify Koopman models of potentially unstable systems, which must be operated in closed-loop with a controller.

Chapter 6

Closed-Loop Koopman Operator Approximation

Summary

This chapter proposes a method to identify a Koopman model of a feedback-controlled system given a known controller. Existing Koopman operator approximation methods are designed to identify open-loop systems. However, it is impractical or impossible to run experiments on some systems, such as unstable systems, in an open-loop fashion. The proposed method leverages the linearity of the Koopman operator, along with knowledge of the controller and the structure of the closed-loop system, to simultaneously identify the closed-loop and plant systems. It is realized as a set of equality constraints on the SDP reformulation of EDMD presented in Chapter 4. The advantages of the proposed closed-loop Koopman operator approximation method are demonstrated in simulation using a Duffing oscillator and experimentally using a rotary inverted pendulum system.

6.1 Introduction

Closed-loop (CL) systems with known controllers frequently arise in practice, especially when the plant under consideration is inherently unstable. In this chapter, the identification of Koopman operators for closed-loop systems is considered. Specifically, the proposed method simultaneously identifies the closed-loop system and the open-loop plant given a known controller through the use of constraints on the reformulation of the Koopman operator approximation problem posed in Chapter 4. The proposed approach is summarized in Figure 6.1.

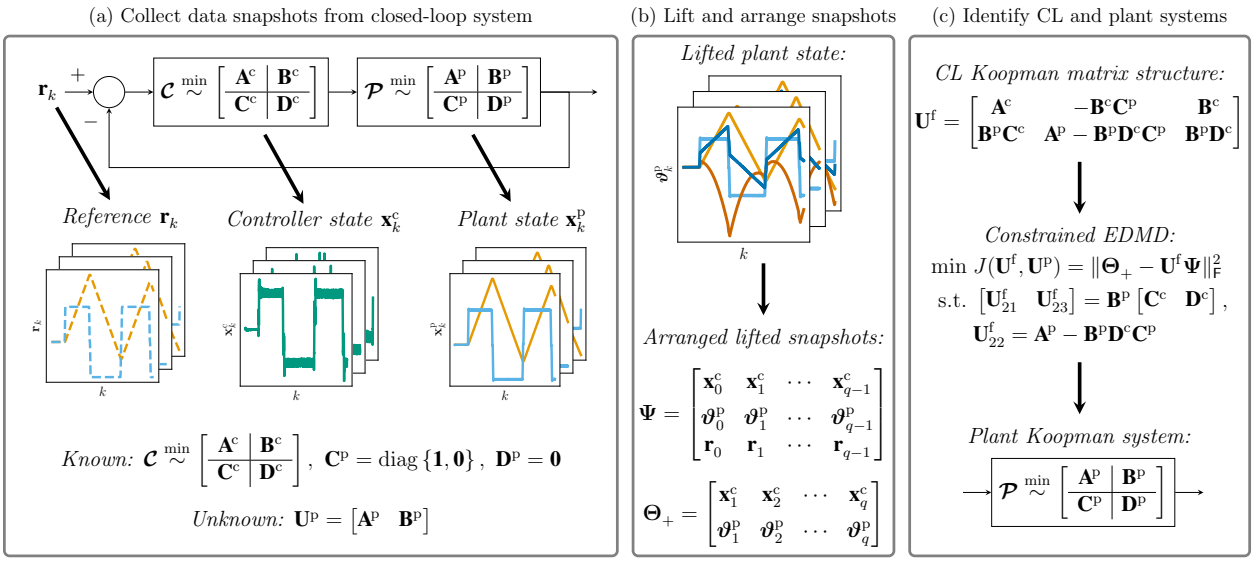


Figure 6.1: Overview of the proposed closed-loop Koopman operator identification method. To simplify the presentation, no feedforward signal is used. (a) First, the controller reference, controller state, and plant state are recorded during a series of experiments. The controller state space matrices are known, and $\mathbf{C}^{\mathrm{p}}$ and $\mathbf{D}^{\mathrm{p}}$ are fixed. If the controller state is not directly available, it can be computed from its input and output. Only the Koopman matrix of the plant $\mathbf{U}^{\mathrm{p}}$ is unknown. (b) The plant state is lifted and augmented with the controller state and reference. (c) The Koopman matrices of the CL system and the plant system are approximated simultaneously by incorporating the known structure of the CL system as a constraint on the EDMD problem.

6.1.1 Related work

In the field of system identification, an area of particular interest is the identification of closed-loop systems [97–100]. It is often unsafe or impractical to run an experiment on a plant without a feedback control system [100, §1.1, §17.3]. In some cases, the control loop is simply ignored and the controller's outputs are treated as exogenous input signals to the plant. However, this simplification, known as the *direct approach* to closed-loop system identification [100, §13.5], is problematic in some situations. The feedback loop introduces correlations between the plant's output and its input, leading to biased estimates of the plant's dynamics [97, 98], [39, §11.1]. In cases where this bias is small, the direct approach may yield acceptable results. In cases where the bias is not small, more sophisticated closed-loop system identification methods must be employed. The *indirect approach* to closed-loop system identification [100, §13.5], which uses the reference signal as the exogenous input, identifies the closed-loop system as a whole. The indirect approach then uses knowledge of the controller to extract a model of the plant system for use on its own [97, 98], [39, §11.1]. While closed-loop system identification can lead to more complex plant models, in many situations it is the appropriate tool to use to avoid biased estimates of model parameters.

Closed-loop identification is particularly practical when identifying unstable plants. Evaluating the performance of an unstable model is challenging, as prediction errors may diverge even if the identified model is accurate. This is problematic in the Koopman framework, where lifting functions are unknown and are often chosen based on the prediction error they achieve. By first identifying a model of the closed-loop system, which is assumed to be AS, the closed-loop prediction error can be used as a more informative goodness-of-fit metric [97, §4.1]. Closed-loop identification methods also allow the closed-loop system's model parameters to be regularized, when regularizing the unstable plant's parameters may incorrectly lead to a stable model.

6.1.2 Contribution

The key contribution of this chapter is a means to identify a Koopman representation of a system using closed-loop data. The proposed method is a variation of the indirect approach to system identification. Given knowledge of the controller, the method simultaneously identifies Koopman models of the closed-loop system and the plant. The significance of this approach is that a system that cannot be operated in open-loop without a feedback controller can now be identified using Koopman operator theory. Even in situations where open-loop or closed-loop methods would identify the same model, it is shown that the closed-loop system is more convenient to work with, as it naturally provides a bounded prediction error and allows the closed-loop Koopman matrix to be included in a regularizer or additional constraint, such as the asymptotic stability constraint presented in Section 5. The advantages of the proposed method are demonstrated on a simulated controlled Duffing oscillator system subject to coloured measurement noise, as well as on a rotary inverted pendulum system which, due to its instability, would be difficult to identify in an open-loop setting.

The software required to fully reproduce the results of this chapter, including the rotary inverted pendulum dataset, is available at `https://github.com/decargroup/closed_loop_koopman`. The software used to gather the dataset, implemented in C, is available at `https://github.com/decargroup/quanser_qube`.

6.2 Formulation of the Closed-Loop Koopman System

Consider the Koopman system modelling the plant,

$$\boldsymbol{\vartheta}^{\mathrm{p}}_{k+1} = \mathbf{A}^{\mathrm{p}}\boldsymbol{\vartheta}^{\mathrm{p}}_{k} + \mathbf{B}^{\mathrm{p}}\boldsymbol{\upsilon}^{\mathrm{p}}_{k}, \tag{6.1}$$

$$\boldsymbol{\zeta}^{\mathrm{p}}_{k} = \mathbf{C}^{\mathrm{p}}\boldsymbol{\vartheta}^{\mathrm{p}}_{k} + \mathbf{D}^{\mathrm{p}}\boldsymbol{\upsilon}^{\mathrm{p}}_{k}, \tag{6.2}$$

along with the linear system modelling the known controller,

$$\mathbf{x}^{\mathrm{c}}_{k+1} = \mathbf{A}^{\mathrm{c}}\mathbf{x}^{\mathrm{c}}_{k} + \mathbf{B}^{\mathrm{c}}\mathbf{u}^{\mathrm{c}}_{k}, \tag{6.3}$$

$$\mathbf{y}^{\mathrm{c}}_{k} = \mathbf{C}^{\mathrm{c}}\mathbf{x}^{\mathrm{c}}_{k} + \mathbf{D}^{\mathrm{c}}\mathbf{u}^{\mathrm{c}}_{k}. \tag{6.4}$$

Let

$$\boldsymbol{\upsilon}^{\mathrm{p}}_{k} = \mathbf{y}^{\mathrm{c}}_{k} + \mathbf{f}_{k}, \tag{6.5}$$

which yields the series interconnection of the controller and the plant with a feedforward signal $\mathbf{f}_k$. This feedforward signal is entirely exogenous, and could be generated by a nonlinear inverse model of the plant. The new Koopman system's input includes the controller input and feedforward input, resulting in

$$\boldsymbol{\upsilon}^{\mathrm{s}}_{k} = \begin{bmatrix} \mathbf{u}^{\mathrm{c}}_{k} \\ \mathbf{f}_{k} \end{bmatrix}, \tag{6.6}$$

while its output is simply the plant output, $\boldsymbol{\zeta}_k = \boldsymbol{\zeta}^{\mathrm{p}}_{k}$. The new system's state includes the controller state and plant state, resulting in

$$\boldsymbol{\vartheta}_{k} = \begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k} \end{bmatrix}. \tag{6.7}$$

The cascaded plant and controller systems are depicted in Figure 6.2.

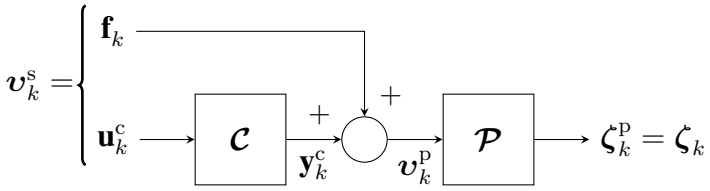


Figure 6.2: Series interconnection of the controller and plant.

The state-space representation of the plant becomes

$$\boldsymbol{\vartheta}^{\mathrm{p}}_{k+1} = \mathbf{A}^{\mathrm{p}}\boldsymbol{\vartheta}^{\mathrm{p}}_{k} + \mathbf{B}^{\mathrm{p}}\left(\mathbf{C}^{\mathrm{c}}\mathbf{x}^{\mathrm{c}}_{k} + \mathbf{D}^{\mathrm{c}}\mathbf{u}^{\mathrm{c}}_{k} + \mathbf{f}_{k}\right) \tag{6.8}$$

$$= \begin{bmatrix} \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k} \end{bmatrix} + \begin{bmatrix} \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\mathrm{c}}_{k} \\ \mathbf{f}_{k} \end{bmatrix}, \tag{6.9}$$

$$\boldsymbol{\zeta}^{\mathrm{p}}_{k} = \mathbf{C}^{\mathrm{p}}\boldsymbol{\vartheta}^{\mathrm{p}}_{k} + \mathbf{D}^{\mathrm{p}}\left(\mathbf{C}^{\mathrm{c}}\mathbf{x}^{\mathrm{c}}_{k} + \mathbf{D}^{\mathrm{c}}\mathbf{u}^{\mathrm{c}}_{k} + \mathbf{f}_{k}\right) \tag{6.10}$$

$$= \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{C}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k} \end{bmatrix} + \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{D}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\mathrm{c}}_{k} \\ \mathbf{f}_{k} \end{bmatrix}. \tag{6.11}$$

The state-space representation of the series-interconnected system is therefore

$$\begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k+1} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k} \end{bmatrix} + \begin{bmatrix} \mathbf{B}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\mathrm{c}}_{k} \\ \mathbf{f}_{k} \end{bmatrix}, \tag{6.12}$$

$$\boldsymbol{\zeta}^{\mathrm{p}}_{k} = \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{C}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{c}}_{k} \\ \boldsymbol{\vartheta}^{\mathrm{p}}_{k} \end{bmatrix} + \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{D}^{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\mathrm{c}}_{k} \\ \mathbf{f}_{k} \end{bmatrix}, \tag{6.13}$$

or equivalently,

$$\boldsymbol{\vartheta}_{k+1} = \begin{bmatrix} \mathbf{A}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\vartheta}_{k} + \begin{bmatrix} \mathbf{B}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\upsilon}^{\mathrm{s}}_{k}, \tag{6.14}$$

$$\boldsymbol{\zeta}_{k} = \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{C}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\vartheta}_{k} + \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{D}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\upsilon}^{\mathrm{s}}_{k}. \tag{6.15}$$

Next, consider the negative feedback interconnection,

$$\mathbf{u}^{\mathrm{c}}_{k} = \mathbf{r}_{k} - \boldsymbol{\zeta}_{k}, \tag{6.16}$$

where $\mathbf{r}_k$ is an exogenous reference signal. The input of this new feedback-interconnected system is defined to be

$$\boldsymbol{\upsilon}_{k} = \begin{bmatrix} \mathbf{r}_{k} \\ \mathbf{f}_{k} \end{bmatrix}, \tag{6.17}$$

as depicted in Figure 6.3. Substituting (6.16) into (6.6) yields

$$\boldsymbol{\upsilon}^{\mathrm{s}}_{k} = \begin{bmatrix} \mathbf{r}_{k} - \boldsymbol{\zeta}_{k} \\ \mathbf{f}_{k} \end{bmatrix} \tag{6.18}$$

$$= \boldsymbol{\upsilon}_{k} - \begin{bmatrix} \boldsymbol{\zeta}_{k} \\ \mathbf{0} \end{bmatrix}. \tag{6.19}$$

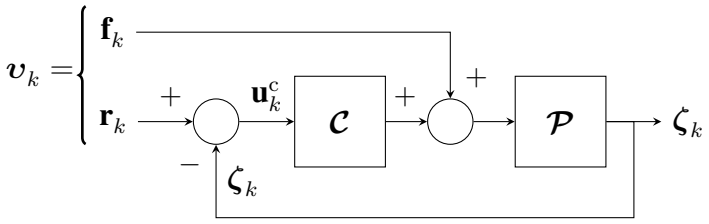


Figure 6.3: Feedback interconnection of the controller and plant.

Substituting (6.19) into (6.15) results in a new output equation,

$$\boldsymbol{\zeta}_k = (\mathbf{1} + \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}})^{-1} \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{C}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\vartheta}_k + (\mathbf{1} + \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}})^{-1} \begin{bmatrix} \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{D}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\upsilon}_k, \tag{6.20}$$

where the feedback interconnection is *well-posed* if and only if $\mathbf{1} + \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}$ is invertible [101, §4.9.1]. Let $\mathbf{Q} = \mathbf{1} + \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}$. Substituting (6.20) into (6.19) yields

$$\boldsymbol{\upsilon}_k^{\mathrm{s}} = \begin{bmatrix} -\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & -\mathbf{Q}^{-1}\mathbf{C}^{\mathrm{p}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{\vartheta}_k + \begin{bmatrix} \mathbf{1} - \mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & -\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\upsilon}_k. \tag{6.21}$$

Substituting (6.21) back into (6.14) and rearranging results in a new state equation,

$$\begin{aligned} \boldsymbol{\vartheta}_{k+1} = &\begin{bmatrix} \mathbf{A}^{\mathrm{c}} - \mathbf{B}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & -\mathbf{B}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{C}^{\mathrm{p}} \\ \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{C}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\vartheta}_k \\ &+ \begin{bmatrix} \mathbf{B}^{\mathrm{c}} - \mathbf{B}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & -\mathbf{B}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}} \\ \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\upsilon}_k, \end{aligned} \tag{6.22}$$

and the output equation,

$$\boldsymbol{\zeta}_k = \begin{bmatrix} \mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{Q}^{-1}\mathbf{C}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\vartheta}_k + \begin{bmatrix} \mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{Q}^{-1}\mathbf{D}^{\mathrm{p}} \end{bmatrix} \boldsymbol{\upsilon}_k. \tag{6.23}$$

In the Koopman system, only $\mathbf{A}^{\mathrm{p}}$ and $\mathbf{B}^{\mathrm{p}}$ are determined from experimental data. The remaining state-space matrices are chosen to be

$$\mathbf{C}^{\mathrm{p}} = \begin{bmatrix} \mathbf{1}_{m\times m} & \mathbf{0}_{m\times(p_\vartheta - m)} \end{bmatrix}, \tag{6.24}$$

$$\mathbf{D}^{\mathrm{p}} = \mathbf{0}, \tag{6.25}$$

such that $\mathbf{C}^{\mathrm{p}}$ recovers the controlled states of the nonlinear system being modelled, which are the first m states in $\boldsymbol{\vartheta}_k^{\mathrm{p}}$. Substituting in (6.25) implies that $\mathbf{Q} = \mathbf{1}$. Thus, the simplified state-space representation of the closed-loop system is

$$\boldsymbol{\vartheta}_{k+1} = \underbrace{\begin{bmatrix} \mathbf{A}^{\mathrm{c}} & -\mathbf{B}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}} \\ \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}} \end{bmatrix}}_{\mathbf{A}^{\mathrm{f}}} \boldsymbol{\vartheta}_k + \underbrace{\begin{bmatrix} \mathbf{B}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix}}_{\mathbf{B}^{\mathrm{f}}} \boldsymbol{\upsilon}_k, \tag{6.26}$$

$$\boldsymbol{\zeta}_k = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{C}^{\mathrm{p}} \end{bmatrix}}_{\mathbf{C}^{\mathrm{f}}} \boldsymbol{\vartheta}_k + \underbrace{\mathbf{0}}_{\mathbf{D}^{\mathrm{f}}} \boldsymbol{\upsilon}_k, \tag{6.27}$$

which is always a well-posed feedback interconnection, since $\mathbf{Q} = \mathbf{1} + \mathbf{D}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} = \mathbf{1}$ is invertible.

6.3 Identification of the Closed-Loop and Plant Systems

Consider the closed-loop lifted dataset $\mathcal{D} = \{\boldsymbol{\vartheta}_k, \boldsymbol{\upsilon}_k\}_{k=0}^{q}$, where the matrix approximation of the Koopman operator is

$$\mathbf{U}^{\mathrm{f}} = \begin{bmatrix} \mathbf{U}_{11}^{\mathrm{f}} & \mathbf{U}_{12}^{\mathrm{f}} & \mathbf{U}_{13}^{\mathrm{f}} & \mathbf{U}_{14}^{\mathrm{f}} \\ \mathbf{U}_{21}^{\mathrm{f}} & \mathbf{U}_{22}^{\mathrm{f}} & \mathbf{U}_{23}^{\mathrm{f}} & \mathbf{U}_{24}^{\mathrm{f}} \end{bmatrix}. \tag{6.28}$$

$$= \begin{bmatrix} \mathbf{A}^{\mathrm{c}} & -\mathbf{B}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}} & \mathbf{B}^{\mathrm{c}} & \mathbf{0} \\ \mathbf{B}^{\mathrm{p}}\mathbf{C}^{\mathrm{c}} & \mathbf{A}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}} & \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix} \tag{6.29}$$

Comparing (6.28) and (6.29) reveals that the matrices, $\mathbf{B}^{\mathrm{p}}$, $\mathbf{U}_{21}^{\mathrm{f}}$, $\mathbf{U}_{23}^{\mathrm{f}}$, and $\mathbf{U}_{24}^{\mathrm{f}}$ are related via the expression

$$\mathbf{B}^{\mathrm{p}} \begin{bmatrix} \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{21}^{\mathrm{f}} & \mathbf{U}_{23}^{\mathrm{f}} & \mathbf{U}_{24}^{\mathrm{f}} \end{bmatrix}. \tag{6.30}$$

Thus, one way to compute $\mathbf{U}^{\mathrm{p}} = \begin{bmatrix} \mathbf{A}^{\mathrm{p}} & \mathbf{B}^{\mathrm{p}} \end{bmatrix}$ is to first compute $\mathbf{U}^{\mathrm{f}}$ using EDMD and then recover the plant's state-space matrices using

$$\mathbf{B}^{\mathrm{p}} = \begin{bmatrix} \mathbf{U}_{21}^{\mathrm{f}} & \mathbf{U}_{23}^{\mathrm{f}} & \mathbf{U}_{24}^{\mathrm{f}} \end{bmatrix} \begin{bmatrix} \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} & \mathbf{1} \end{bmatrix}^{\dagger}, \tag{6.31}$$

$$\mathbf{A}^{\mathrm{p}} = \mathbf{U}_{22}^{\mathrm{f}} + \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}}. \tag{6.32}$$

While this approach is simple, it is not guaranteed to preserve the required relationships between $\mathbf{U}^{\mathrm{f}}$, $\mathbf{A}^{\mathrm{p}}$, and $\mathbf{B}^{\mathrm{c}}$ found in (6.28) and (6.29). As a result, identifying the closed-loop system, extracting the plant model with (6.31) and (6.32), and then closing the loop again with the same controller using (6.29) can result in an entirely different closed-loop system. This pitfall is explored further in Section 6.6.4.

To preserve the structure of the closed-loop system, (6.30) and (6.32) can be treated as constraints on the SDP reformulation of the EDMD problem presented in Chapter 4. Including Tikhonov regularization, the resulting EDMD optimization problem is

$$\min\ J(\mathbf{U}^{\mathrm{f}}, \mathbf{U}^{\mathrm{p}}; \alpha) = \frac{1}{q}\left\|\boldsymbol{\Theta}_{+} - \mathbf{U}^{\mathrm{f}}\boldsymbol{\Psi}\right\|_{\mathsf{F}}^{2} + \frac{\alpha}{q}\left\|\mathbf{U}^{\mathrm{f}}\right\|_{\mathsf{F}}^{2} \tag{6.33}$$

$$\text{s.t. } \begin{bmatrix} \mathbf{U}_{21}^{\mathrm{f}} & \mathbf{U}_{23}^{\mathrm{f}} & \mathbf{U}_{24}^{\mathrm{f}} \end{bmatrix} = \mathbf{B}^{\mathrm{p}} \begin{bmatrix} \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} & \mathbf{1} \end{bmatrix}, \tag{6.34}$$

$$\mathbf{U}_{22}^{\mathrm{f}} = \mathbf{A}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}}. \tag{6.35}$$

Including the structural constraints on the closed-loop system from (6.30) and (6.32) results

in

$$\min J(\mathbf{U}^{\mathrm{f}}, \mathbf{U}^{\mathrm{p}}, \mathbf{W}; \alpha) = \operatorname{tr}(\mathbf{W}) \tag{6.36}$$

$$\text{s.t. } \mathbf{W} > 0, \tag{6.37}$$

$$\begin{bmatrix} -\mathbf{W} + \mathbf{F} - \operatorname{He}\{\mathbf{U}^{\mathrm{f}}\mathbf{G}^{\mathsf{T}}\} & \mathbf{U}^{\mathrm{f}}\mathbf{R}_{\alpha} \\ * & -\mathbf{1} \end{bmatrix} < 0, \tag{6.38}$$

$$\begin{bmatrix} \mathbf{U}^{\mathrm{f}}_{21} & \mathbf{U}^{\mathrm{f}}_{23} & \mathbf{U}^{\mathrm{f}}_{24} \end{bmatrix} = \mathbf{B}^{\mathrm{p}} \begin{bmatrix} \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} & \mathbf{1} \end{bmatrix}, \tag{6.39}$$

$$\mathbf{U}^{\mathrm{f}}_{22} = \mathbf{A}^{\mathrm{p}} - \mathbf{B}^{\mathrm{p}}\mathbf{D}^{\mathrm{c}}\mathbf{C}^{\mathrm{p}}, \tag{6.40}$$

where

$$\mathbf{H}_{\alpha} = \mathbf{H} + \frac{\alpha}{q}\mathbf{1} \tag{6.41}$$

$$= \mathbf{R}_{\alpha}\mathbf{R}_{\alpha}^{\mathsf{T}}, \tag{6.42}$$

which provides a means to simultaneously identify a Koopman model of the closed-loop system and the plant system. While this method applies Tikhonov regularization to the closed-loop system's Koopman matrix, any regularizer or constraint, such as those in Chapter 5, can be applied to either the closed-loop system or the plant system, since both Koopman matrices are optimization variables.

6.4 Bias in EDMD

Even when identifying a plant operating in open-loop, EDMD is known to identify biased Koopman operators when sensor noise is significant [102, §2.2]. However, since the plant's input is exogenous in an open-loop setting, any noise in the input signal is uncorrelated with the measurement noise. In fact, the input signal is often known exactly because it is chosen by the user.

In contrast, consider a scenario where measurements are gathered from a system operating in closed-loop with a controller. Feedback action propagates noise from the plant's measured output throughout the system. If the presence of the controller is ignored and its control signal to the plant is treated as exogenous, correlations between the plant input and measurement noise result in increased bias [97, 98][39, §11.1]. Furthermore, the input signal is no longer known exactly, as it is corrupted by noise that is correlated with the state's measurement noise.

Identifying the Koopman system using a closed-loop procedure sidesteps this issue by

using the exogenous reference and feedforward signals as inputs. If these signals are noisy, the noise is uncorrelated with the state's measurement noise. However, these inputs are typically known exactly. Using a closed-loop approach therefore reduces bias compared to treating the feedback controller's output as an exogenous plant input. The remaining bias, which is due solely to measurement noise, is inherent to EDMD. In practice, this bias is sometimes too small to justify the use of more complex identification techniques [39, §11.1]. Variants of DMD like forward-backward DMD [102, §2.4][94] and total least-squares DMD [102, §2.5] can also be used to compensate for this bias.

6.5 Simulated Example: Duffing Oscillator

Several advantages of the proposed closed-loop Koopman operator approximation method, referred to in this section as closed-loop EDMD (CL EDMD), are demonstrated on a simulated closed-loop Duffing oscillator system, pictured in Figure 6.4.

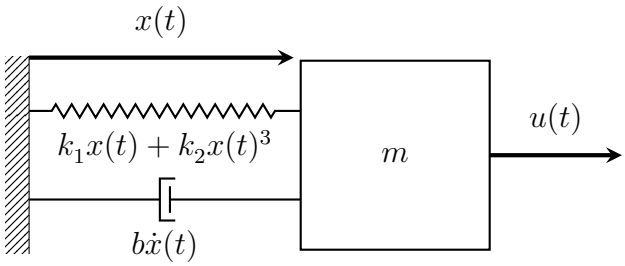


Figure 6.4: Duffing oscillator system with position $x(t)$, force $u(t)$, mass m, viscous damping b, linear stiffness k_1, and nonlinear stiffness k_2.

6.5.1 Simulation setup

The Duffing oscillator [103] can be viewed as a forced mass-spring-damper system with nonlinear spring stiffness. That is,

$$m\ddot{x}(t) + b\dot{x}(t) + k_1x(t) + k_2x(t)^3 = u(t), \tag{6.43}$$

where $x(t)$ is the mass position (m) and $u(t)$ is the external force (N). A simulated dataset is generated using a Duffing oscillator with mass $m = 0.01\,\text{kg}$, viscous damping $b = 0.1\,\text{Ns/m}$, linear stiffness $k_1 = 0.02\,\text{N/m}$, and nonlinear stiffness $k_2 = 0.4\,\text{N/m}^3$. Its position error $e(t) = r(t) - x(t)$ is controlled by a proportional-integral controller,

$$K(s) = K^{\text{p}} + \frac{1}{s}K^{\text{i}}, \tag{6.44}$$

where $s \in \mathbb{C}$ is the Laplace variable, $K^{\mathrm{p}} = 1\,\mathrm{N/m}$ is the proportional gain, and $K^{\mathrm{i}} = 1\,\mathrm{N/(m\,s)}$ is the integral gain. Thus, following the notation of Section 6.2, $x^{\mathrm{p}}(t) = x(t)$, $v^{\mathrm{p}}(t) = u(t)$, and $v(t) = r(t)$.

The simulated dataset consists of 10 training episodes and one test episode. Each episode is 10 s long and begins with zero initial conditions. The episodes are sampled at a frequency of 100 Hz. The position reference is a pseudorandom binary sequence (PRBS) [100, §13.3] with amplitude 1 m. An example reference signal is pictured in Figure 6.5a. The training set position measurements used for both control and identification are subject to additive coloured noise. Noise samples are first drawn from a Gaussian distribution with zero mean and covariance $\sigma^2 = 0.02\,\mathrm{m}^2$. This white noise signal is then filtered using a 12^{th} order Butterworth filter with a cutoff frequency of 5 Hz. The test set does not have any added measurement noise.

6.5.2 Comparison of identified models

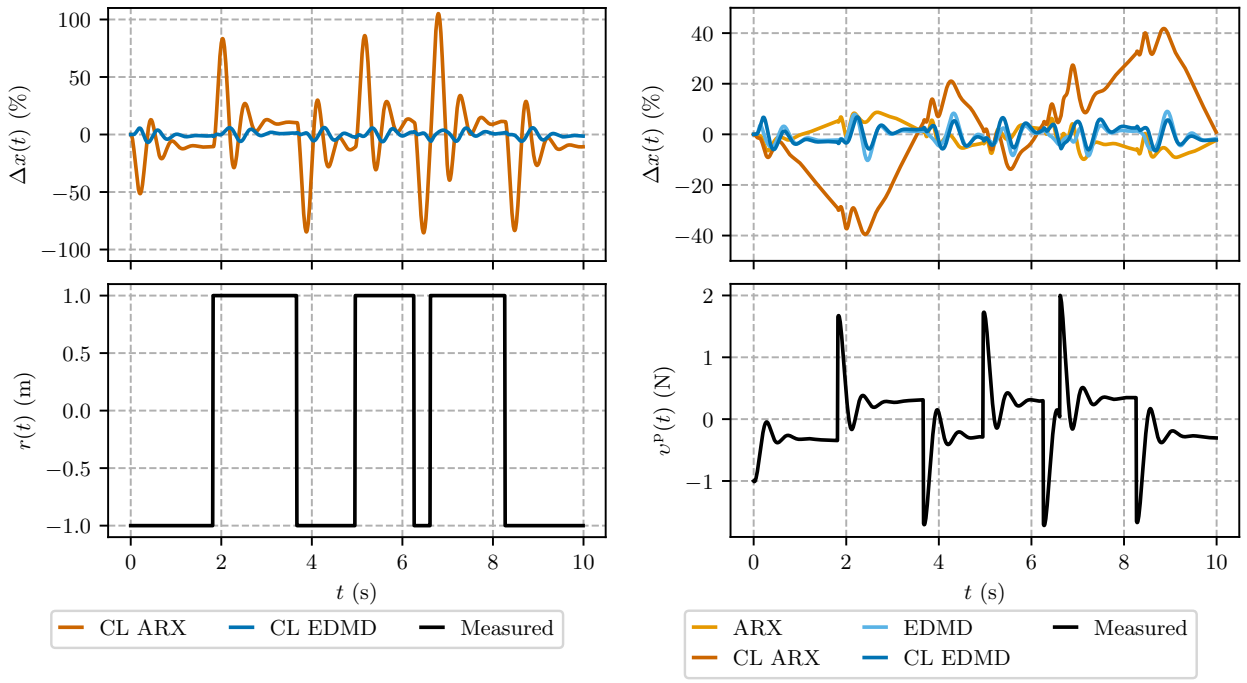
The Koopman lifting functions used to model the Duffing oscillator consist of 6 time delays of the state followed by 50 thin plate RBFs of the form

$$\varphi_i(\mathbf{x}) = (\alpha\|\mathbf{x} - \mathbf{c}_i\| + \delta)^2 \ln(\alpha\|\mathbf{x} - \mathbf{c}_i\| + \delta), \quad i = 1, \ldots, 50, \tag{6.45}$$

where $\alpha = 0.2$ is the shape parameter and $\delta = 0.001$ is a small offset to avoid evaluating the logarithm at zero. The centres $\mathbf{c}_i$ are selected using Latin hypercube sampling [62, 63]. The same set of lifting functions is used to identify two Koopman models of the Duffing oscillator: one using EDMD in open-loop, and another using the proposed CL EDMD method. In both cases, the inputs are not lifted. This results in a 57-dimensional lifted state for the open-loop Koopman model, and a 58-dimensional state for the closed-loop Koopman model, as the controller's state is also included.

To better understand the advantages and disadvantages of the Koopman approaches, two autoregressive exogenous input (ARX) models [100, §4.2][104, §2.2.1.5] are identified using `SIPPY` [105], an open-source system identification library for Python. For comparison with EDMD, an ARX model of the open-loop plant is identified using 6 output lags and one input lag, resulting in a 6^{th} order system. This is an example of the direct approach to closed-loop system identification. For comparison with CL EDMD, an ARX model of the closed-loop system is identified, also using 6 output lags and one input lag. The transfer function of the plant is recovered using the indirect approach discussed in [97, §5.1] and [100, §13.5]. The indirect approach, referred to as CL ARX in this section, increases the order of the plant transfer function to 14. Note that `SIPPY` only supports identification using a single training

episode. Thus, to make full use of the training data, a transfer function is identified for each episode and the coefficients are averaged to obtain a final model.



(a) Prediction errors of the closed-loop systems identified using CL ARX and CL EDMD methods. The Koopman approach better captures the dynamics of the closed-loop system.

(b) Prediction errors of the plant systems identified using closed-loop and open-loop ARX and EDMD methods. The CL ARX model fails to capture the plant dynamics, possibly due to its high order. The Koopman models perform best, with nearly identical prediction errors.

Figure 6.5: Prediction errors the closed-loop and plant systems identified using ARX and EDMD methods.

Figure 6.5 shows the prediction errors of all four models on the test episode. The mean and RMS prediction errors for the plant system, normalized by the peak amplitude of the state, are summarized in Table 6.1. While the CL EDMD method is able to identify accurate closed-loop and plant models, the CL ARX method is not able to accurately predict the Duffing oscillator's position throughout the test episode. Its failure to accurately predict the plant's open-loop dynamics is likely due to its inaccurate identification of the closed-loop system. Furthermore, the classical indirect approach to system identification leads to high-order plant transfer functions [97, §5.1], which may impact the accuracy of the plant model. In contrast, CL EDMD always identifies a plant system with fewer states than the closed-loop system, and does not rely on multiplying or inverting any transfer matrices. While the Koopman models have over 50 states, only 6 of them are time delays, compared to the 14 of the CL ARX plant model. The open-loop ARX model performs better than its

Table 6.1: Normalized mean and RMS open-loop plant errors in test episode.

Method	Normalized mean error	Normalized RMS error
EDMD	$-0.6\,\%$	$3.3\,\%$
CL EDMD	$-0.1\,\%$	$3.2\,\%$
ARX	$-0.9\,\%$	$4.7\,\%$
CL ARX	$2.5\,\%$	$20.4\,\%$

closed-loop version, but not as well as either EDMD or CL EDMD. Both EDMD methods perform nearly identically, with CL EDMD attaining slightly lower mean and RMS errors on the test episode.

While the simulated Duffing oscillator example clearly demonstrates the advantages of Koopman approaches to system identification, it does not highlight the advantages of CL EDMD over standard EDMD. The unique benefits of closed-loop EDMD will be explained in Section 6.6, where a more complex experimental example is considered, and issues such as hyperparameter optimization and regularization are discussed.

6.6 Experimental Example: Rotary Inverted Pendulum

The benefits of the proposed closed-loop EDMD method (CL EDMD) are demonstrated on a dataset collected from the Quanser *QUBE-Servo* [106], a rotary inverted pendulum system. Pictured in Figure 6.6, this system has an unstable equilibrium point when the pendulum is upright.

6.6.1 Experimental setup

The *QUBE-Servo* has one direct current motor with a maximum input voltage of $10\,\mathrm{V}$, as well as two incremental encoders with 2048 counts per revolution. The system is controlled by two parallel proportional-derivative controllers that track references for the motor angle $x_1^{\mathrm{p}}(t)$ and the pendulum angle $x_2^{\mathrm{p}}(t)$. The reference signals are denoted $r_1(t)$ and $r_2(t)$, while the feedforward is denoted $f(t)$. All angles are presented in radians, while the plant inputs are presented in Volts.

The controller's transfer matrix from tracking error $\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{x}^{\mathrm{p}}(t)$ to output $v^{\mathrm{p}}(t)$ is

$$\mathbf{K}(s) = \begin{bmatrix} K_1^{\mathrm{p}} + K_1^{\mathrm{d}} \frac{as}{s+a} & K_2^{\mathrm{p}} + K_2^{\mathrm{d}} \frac{as}{s+a} \end{bmatrix}, \tag{6.46}$$

where $K_1^{\mathrm{p}} = 6\,\mathrm{V/rad}$ and $K_2^{\mathrm{p}} = 30\,\mathrm{V/rad}$ are the proportional gains, $K_1^{\mathrm{d}} = 1.8\,\mathrm{Vs/rad}^2$ and

Figure 6.6: The Quanser *QUBE-Servo* system used to demonstrate the proposed closed-loop Koopman operator approximation method.

$K_2^{\mathrm{d}} = 2.5\,\mathrm{Vs/rad}^2$ are the derivative gains, and $a = 50\,\mathrm{rad/s}$ is the derivative filter cutoff frequency. The controller is discretized using the zero-order hold with a sampling frequency of 500 Hz.

Samples of the exogenous test inputs are pictured in Figure 6.7. The motor angle reference signal is an integrated PRBS, while the pendulum angle reference and feedforward signals are PRBSs [100, §13.3]. The PRBSs are smoothed using a low-pass filter with a cutoff frequency of 200 Hz. The signal amplitudes excite the rotary inverted pendulum as much as possible without causing the pendulum to fall from its upright position.

The dataset presented here consists of 30 training episodes and 20 test episodes. Each episode is 20 s, or 10 000 samples, long. The motor angle, pendulum angle, motor voltage, reference motor angle, reference pendulum angle, and feedforward voltage are recorded at every timestep. The controller's state is computed at each timestep using the recorded tracking error. The first 1 s of each dataset is discarded to remove the transients associated with manually raising the pendulum to the upright position.

The Koopman lifting functions chosen for the rotary inverted pendulum are second-order monomials followed by 10 time delays. As required by the proposed method, the inputs and controller states are not lifted. Further performance improvement can be achieved by including more time delays, with diminishing returns after 20.

As a point of comparison with classical system identification, a MIMO ARX model with 10 output lags and 10 input lags is fit to the closed-loop system using `SIPPY` [105]. However,

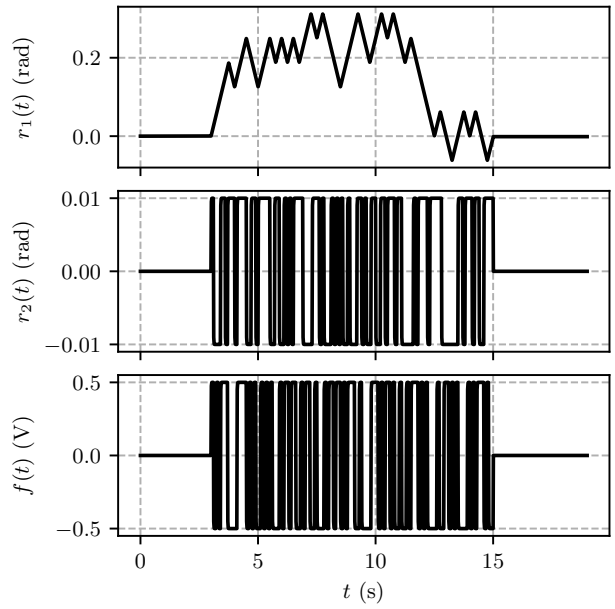


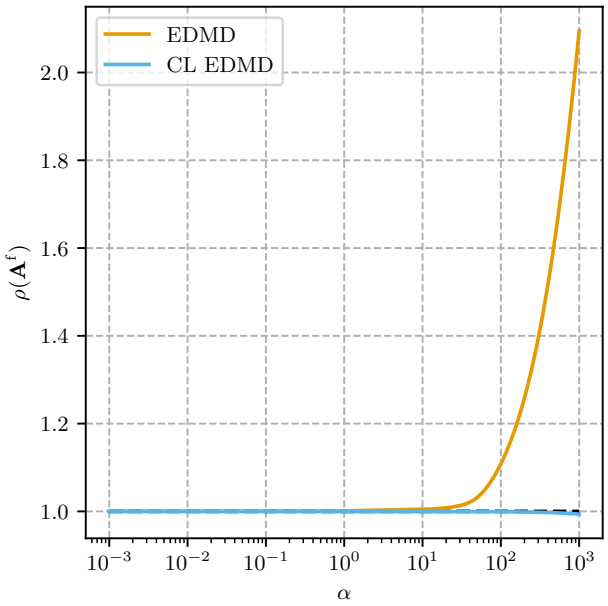
Figure 6.7: Sample of the exogenous test inputs used to identify a Koopman model of the Quanser *QUBE-Servo*.

a difficulty with the indirect approach to closed-loop system identification in the MIMO case is that it requires computing the pseudoinverse of a nonsquare transfer matrix to recover the plant's transfer matrix. Attempting this symbolically leads to a transfer matrix of an intractable order. Since the closed-loop system is strictly proper, computing the pseudoinverse in state-space is also nontrivial. To circumvent this, the 2×1 block of the closed-loop transfer matrix corresponding to the feedforward input is used to recover the plant's transfer matrix. This approach only requires inverting a single-input single-output transfer function.

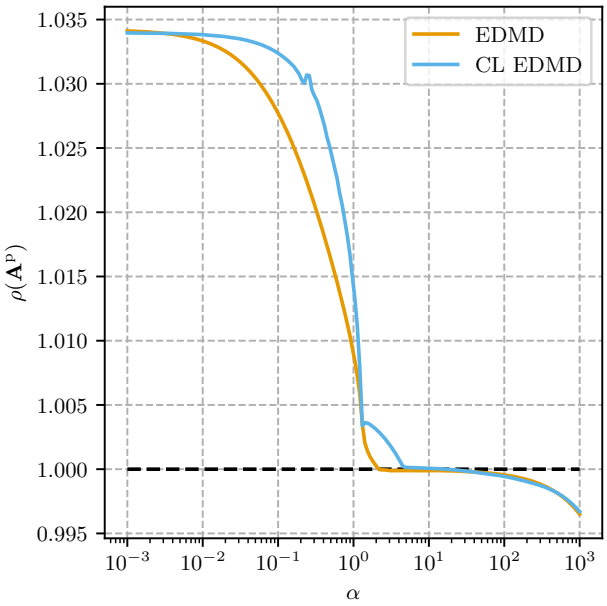
6.6.2 Comparison of regularization methods and scoring metrics

Since the *QUBE-Servo* dataset does not contain significant sensor noise, EDMD does not identify a noticeably biased Koopman model. In fact, with no regularization, both EDMD and the proposed closed-loop method identify the same Koopman system. However, in many situations, regularization is a necessary component of the Koopman identification process. Choosing an appropriate regularization coefficient typically requires a hyperparameter optimization procedure. As such, the advantages of the proposed method are examined through the lens of hyperparameter optimization.

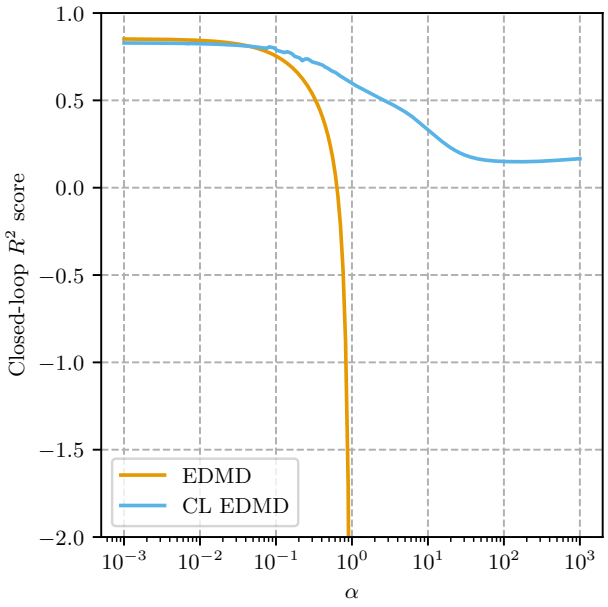
The proposed closed-loop Koopman operator approximation method is first compared to EDMD by varying the regularization coefficient α of each identification method. Specifically, 180 values of α are evaluated, spaced logarithmically between 10^{-3} and 10^{3}. Recall that



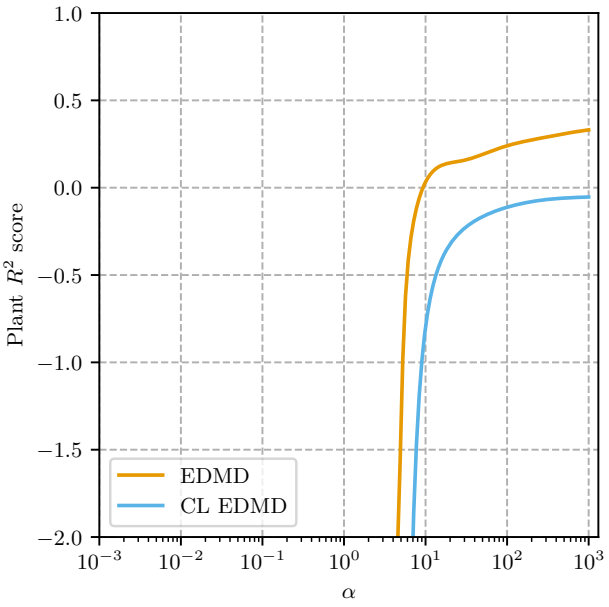
(a) Relationship between the regularization coefficient α and the spectral radii of the CL systems identified with EDMD and CL EDMD. As α increases, the system identified with EDMD becomes unstable, while the one identified with the proposed method remains AS.



(b) Relationship between the regularization coefficient α and the spectral radii of the plant systems identified with EDMD and CL EDMD. As α increases, both systems are incorrectly stabilized by the regularization.



(c) Relationship between the regularization coefficient α and the test set R^2 scores of the CL systems identified with EDMD and CL EDMD. As α increases, both scores decrease. However, the system identified with EDMD becomes unstable around $\alpha = 1$, and its score diverges.



(d) Relationship between the regularization coefficient α and the test set R^2 scores of the plant systems identified with EDMD and CL EDMD. The scores are only finite for large regularization coefficients, indicating that is not a good metric for tuning the regularizer.

Figure 6.8: Effect of regularization on the spectral radii and prediction scores of the CL and plant systems.

EDMD with Tikhonov regularization penalizes the squared Frobenius norm of the plant system, $\|\mathbf{U}^{\mathrm{p}}\|_{\mathsf{F}}^2$, while the proposed closed-loop method penalizes the squared Frobenius norm of the closed-loop system, $\|\mathbf{U}^{\mathrm{f}}\|_{\mathsf{F}}^2$. For the sake of comparison with the closed-loop approach, the open-loop plant models identified by EDMD are wrapped with the known controller. Since the proposed approach simultaneously identifies the closed-loop and plant systems, the plant system identified using the proposed method can be compared directly to the plant model identified using EDMD.

As a first point of comparison between EDMD and the proposed closed-loop method, consider the spectral radii of the closed-loop and plant systems as a function of their regularization coefficients. Given prior knowledge of the inverted pendulum system, both in open-loop and closed-loop contexts, a desirable Koopman model should have AS closed-loop dynamics when controlled using an appropriate controller, but unstable open-loop dynamics. Figure 6.8a shows that regularizing using the Koopman matrix of the closed-loop system always results in an AS closed-loop system. However, regularizing using the plant's Koopman matrix results in an unstable closed-loop system for high enough regularization coefficients. According to Figure 6.8b, the spectral radii of the plant systems identified with both methods share a similar trend. For low regularization coefficients, the systems are correctly identified as unstable, while for very high regularization coefficients, the plant systems are incorrectly stabilized. Figures 6.8a and 6.8b indicate that a low regularization coefficient is appropriate for this system.

As a second point of comparison between the EDMD and the proposed method, consider the three-fold cross-validation score associated with each regularization coefficient. A good cross-validation score should reach its maximum at an appropriate hyperparameter value for the system. Given the spectral radius results in Figures 6.8a and 6.8b, a good scoring metric for the *QUBE-Servo* system should have its peak at a low regularization coefficient. The scoring metric of choice for this system is the R^2 score, also called the coefficient of determination [107]. The R^2 score of a predicted state trajectory $\hat{x}_k$ relative to the true state trajectory x_k is

$$R^2(x_k, \hat{x}_k) = 1 - \frac{\sum_{k=1}^{q} (x_k - \hat{x}_k)^2}{\sum_{k=1}^{q} (x_k - \bar{x})^2}, \tag{6.47}$$

where $\bar{x} = \frac{1}{q}\sum_{k=1}^{q} x_k$ is the mean value of x_k. For multidimensional predicted trajectories, the R^2 score of each state is averaged to obtain a single score. Perfect predictions receive an R^2 score of 1, while predictions that only capture the mean of the data receive an R^2 score of 0. Worse predictions can receive arbitrarily negative scores.

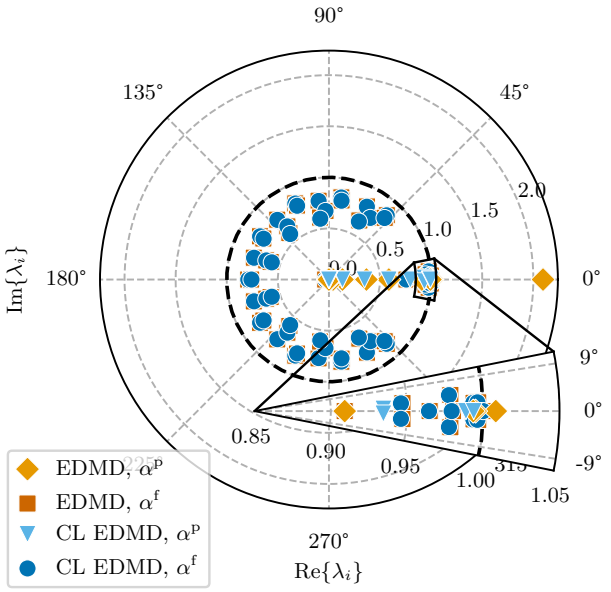
As before, Figure 6.8c shows the closed-loop R^2 scores of models identified with both EDMD and the proposed method, where the plant model identified with EDMD is wrapped

with the known controller. For any regularization coefficient, the R^2 score achieved by the proposed method remains bounded and positive. Thus, the closed-loop regularizer has the expected effect of driving the predicted trajectory towards the mean as the regularization coefficient grows. However, for a large enough regularization coefficient, EDMD identifies an unstable closed-loop system and its R^2 score diverges. The best closed-loop R^2 scores for both methods are attained with a very small regularization, which also leads to the expected stability properties. The use of the plant's open-loop predictions to score the models does not share this behaviour. Since the plant system is inherently unstable, its predictions are highly sensitive to small variations in parameters and initial conditions. Thus, even the predictions of an accurate model can yield unbounded prediction errors. As shown in Figure 6.8d, the only way to achieve a finite plant R^2 score is to select an extremely large regularization coefficient. For EDMD, this results in an unstable closed-loop system and an AS plant, which does not reflect the underlying dynamics of the inverted pendulum system. While the proposed closed-loop methods produces an AS closed-loop system for all regularizers, the open-loop plant it identifies still becomes AS for a large enough regularization coefficient.

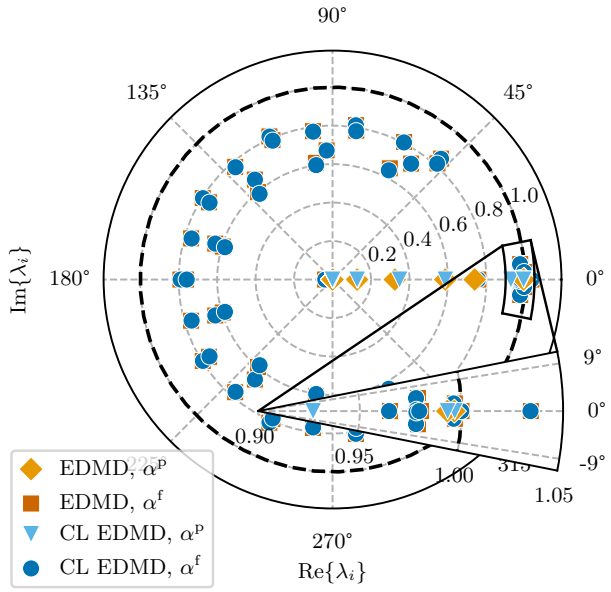
Figure 6.8 demonstrates two key conclusions. First, the closed-loop prediction error should be used for assessing the accuracy of identified models, as the best closed-loop scores correspond to Koopman models with the expected stability properties. Second, regularizing the closed-loop Koopman matrix is preferable to regularizing only the plant's Koopman matrix, as it ensures the closed-loop system will remain AS for high regularization coefficients. Note that closed-loop scoring can be leveraged without the proposed method, simply by identifying the plant using EDMD and wrapping the resulting model with the known controller. However, this approach is susceptible to bias when significant sensor noise is present. Also, only the proposed closed-loop approach has access to the closed-loop Koopman matrix for regularization. Thus, if the controller is already known, it is preferable to use the proposed closed-loop approach from the beginning, rather than leveraging controller knowledge only at the end of the identification procedure.

6.6.3 Comparison of optimized regularization coefficients

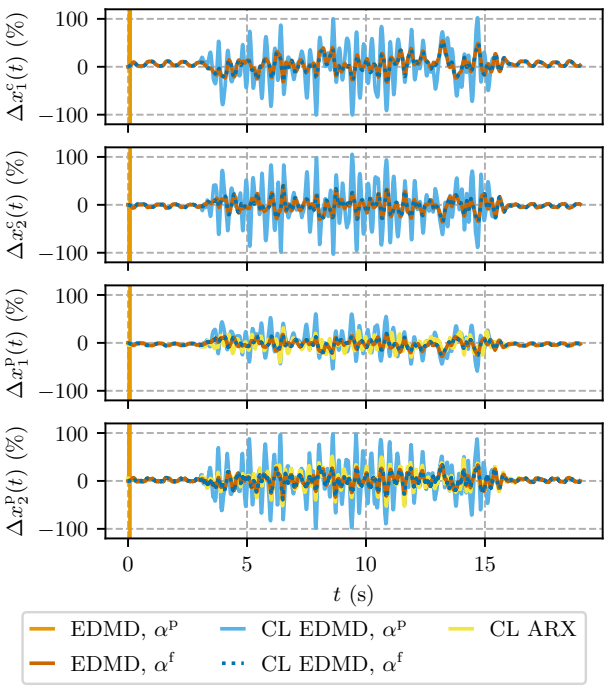
Informed by the results of Section 6.6.2, four approaches to identifying a Koopman model of the *QUBE-Servo* are compared via their eigenvalues and prediction errors. First is the naive open-loop approach, where EDMD is used to identify the plant's Koopman matrix, and the regularization coefficient is selected according to the plant's R^2 score. In this case, the optimal regularization coefficient is $\alpha^{\mathrm{P}} = 10^3$. Second is another open-loop approach, where EDMD is used to identify the plant's Koopman matrix, but the plant is then wrapped



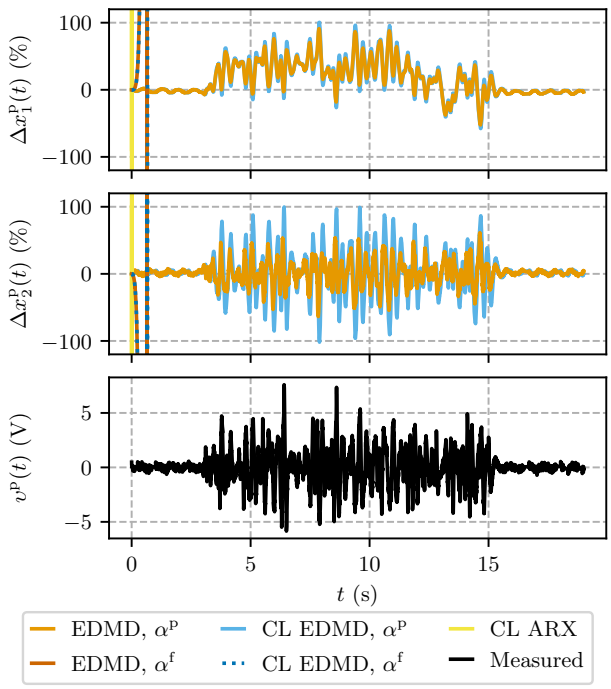
(a) Eigenvalues of the CL systems identified with EDMD and CL EDMD. Unlike EDMD, CL EDMD correctly identifies an AS system for both regularization coefficients.



(b) Eigenvalues of the plant systems identified with EDMD and CL EDMD. For small regularization coefficients, both EDMD and CL EDMD correctly identify unstable systems.



(c) Prediction errors of the CL systems identified with EDMD and CL EDMD. Unlike CL EDMD, EDMD incorrectly identifies an unstable system when using a large regularization coefficient.



(d) Prediction errors and controller output of the plant systems identified with EDMD and CL EDMD. Despite the diverging prediction errors, both methods correctly identify unstable plants for small regularization coefficients.

Figure 6.9: Eigenvalues and prediction errors of closed-loop and plant systems.

with the known controller and the regularization coefficient is selected according to the closed-loop system's R^2 score. In this case, the optimal regularization coefficient is $\alpha^{\mathrm{f}} = 10^{-3}$. Third is a closed-loop approach, where the closed-loop and plant's Koopman matrices are simultaneously identified, but the plant's R^2 score is used to select the regularization coefficient. In this case, the optimal regularization coefficient is $\alpha^{\mathrm{p}} = 10^{3}$. Last is the proposed closed-loop approach, where the closed-loop and plant's Koopman matrices are simultaneously identified, and the closed-loop R^2 score is used to select the regularization coefficient. In this case, the optimal regularization coefficient is $\alpha^{\mathrm{f}} = 10^{-3}$. The prediction errors of these four Koopman models are also compared with those of the closed-loop ARX (CL ARX) approach described in Section 6.6.1 without any regularization.

Consider the closed-loop and plant eigenvalues in Figures 6.9a and 6.9b. The naive approach, which uses the plant for regularization and scoring, identifies an unstable closed-loop system with two eigenvalues clearly outside the unit circle. Due to the large regularization coefficient, the plant system it identifies is AS, which is also inconsistent with the underlying dynamics of the system. The third approach, which uses the closed-loop system for regularization but the plant for scoring, correctly identifies an AS closed-loop system, but incorrectly identifies an AS plant. The remaining two approaches, which use the closed-loop R^2 score to select the regularization coefficient, identify essentially the same eigenvalues. Both methods correctly identify AS closed-loop systems and unstable plant systems. Since measurements from the *QUBE-Servo* system are relatively noiseless, it is expected that the two methods should agree. However, as is well-documented in the system identification literature, the closed-loop method could reduce the bias in the identified model in the presence of significant measurement noise [97, 98][39, §11.1].

Looking at the closed-loop and plant prediction errors of each of these approaches in Figures 6.9c and 6.9d respectively tells the same story as the eigenvalues. Both EDMD and the proposed closed-loop method identify the same Koopman systems at low regularization coefficients. For large regularization coefficients, EDMD incorrectly identifies an unstable closed-loop system and an AS plant. However, the proposed method identifies an AS closed-loop system, regardless of the regularization coefficient. The CL ARX model correctly identifies an AS closed-loop system and an unstable open-loop system. In terms of closed-loop prediction errors, the CL ARX model is slightly worse than the Koopman models that use closed-loop scoring, indicating that it may not be able to capture the nonlinearity of the inverted pendulum system. The R^2 score and normalized root-mean-squared error (NRMSE) of each model on the test set are summarized in Table 6.2. To compute the NRMSE, the RMS error of each state is normalized by the peak amplitude of the true value of that state. The normalized values for each state are then averaged to obtain a single

number summarizing the error of the trajectory prediction.

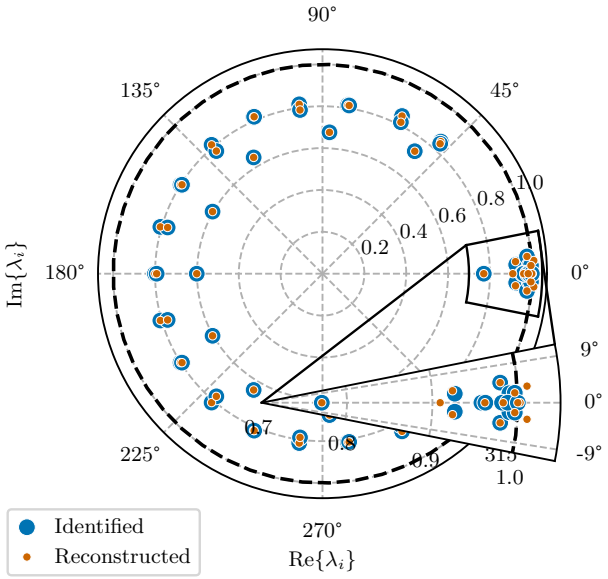
Table 6.2: R^2 score and NRMSE over 20 test episodes.

Method	Regularization	Scoring	R^2 avg.	R^2 std.	NRMSE avg.	NRMSE std.
ARX	—	CL	0.813	0.026	11.7 %	1.3 %
EDMD	Plant	Plant	$-\infty$	—	∞	—
EDMD	Plant	CL	0.897	0.018	8.9 %	1.1 %
EDMD	CL	Plant	0.322	0.061	22.8 %	2.1 %
EDMD	CL	CL	0.893	0.017	9.0 %	1.0 %

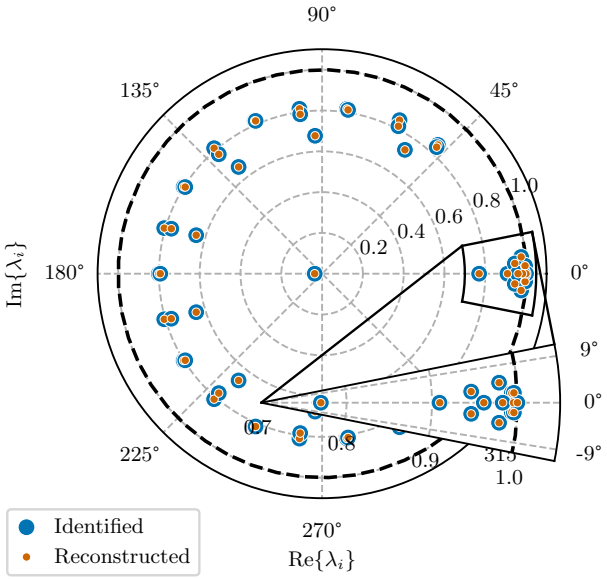
Table 6.2 shows that, among the Koopman models, the most important factor for attaining a high prediction score is the use of the closed-loop prediction error for scoring. For the *QUBE-Servo* system, the choice to use the closed-loop or plant Koopman matrix for regularization is not critical, as long as a sufficiently small regularization coefficient is chosen. However, it must be noted that identifying the open-loop plant directly with EDMD and wrapping the model with the known controller for scoring requires knowledge of the controller and measurements of the controller output, just like the proposed closed-loop approach. If knowledge of the controller is already assumed, then using the proposed closed-loop method is preferable, as it avoids the possibility of bias in the models. Furthermore, in the proposed closed-loop Koopman operator approximation approach both $\mathbf{U}^{\mathrm{f}}$ and $\mathbf{U}^{\mathrm{p}}$ are available as optimization variables. This additional flexibility allows additional knowledge of the closed-loop system and plant to be incorporated into the optimization problem. For example, the closed-loop system could be constrained to be AS using a method from Chapter 5. Alternatively, known properties of the closed-loop system or plant could be incorporated by constraining the poles to a particular region [108]. A distinct advantage of CL EDMD over CL ARX is that it is constraint-based, meaning that it does not require transfer matrix multiplication or pseudoinversion to obtain an estimate of the plant system. Due to the nature of the constraints, the identified plant system will always be of a lower dimension than the closed-loop system. The advantages and disadvantages of each approach are summarized in Table 6.3.

6.6.4 Comparison of least-squares and constrained optimization

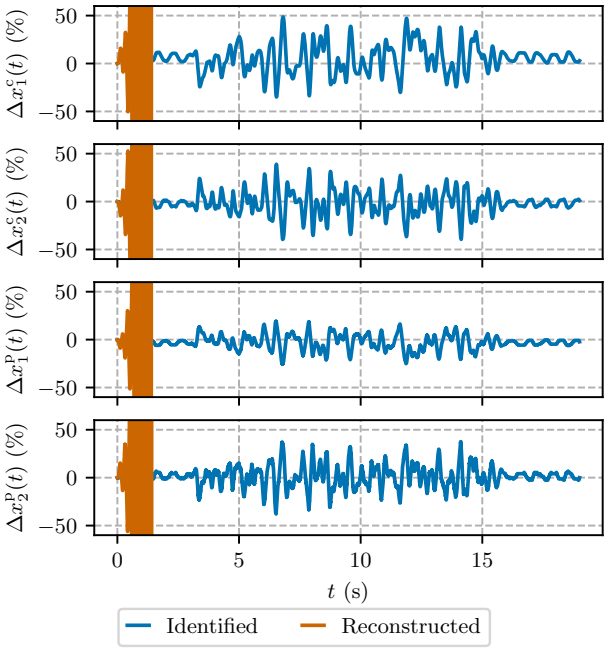
Using least-squares to extract a plant model from an identified closed-loop system seems to provide a simpler alternative to the constraint-based formulation in (6.33)–(6.35). However, using (6.31) and (6.32) to obtain a Koopman model of the plant does not respect the feedback structure of the system. In fact, extracting the plant from the closed-loop system and re-



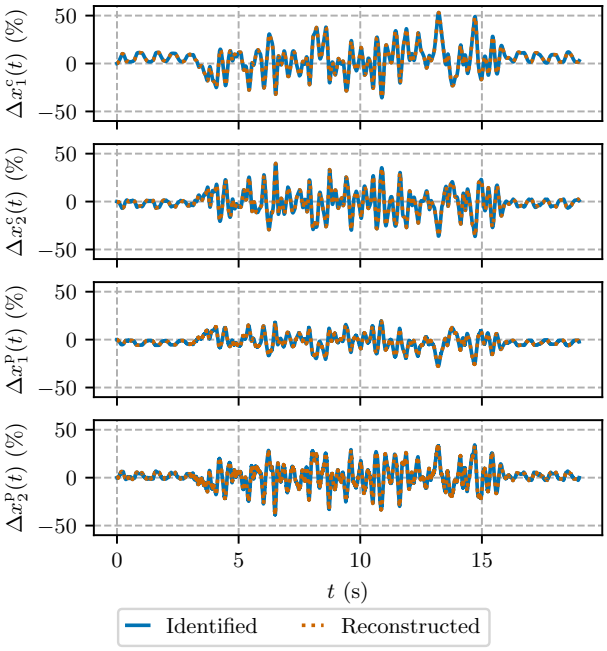
(a) CL eigenvalues identified with the least-squares version of CL EDMD, before and after the plant is extracted and the controller is re-connected. When the CL system is reconstructed from the plant, the eigenvalues move, and some of them leave the unit circle, destabilizing the system.



(b) CL eigenvalues identified with the constrained version of CL EDMD, before and after the plant is extracted and the controller is re-connected. When the CL system is reconstructed from the plant, the eigenvalues do not change.



(c) CL prediction errors of models identified with the least-squares version of CL EDMD, before and after the plant is extracted and the controller is re-connected. Reconstructing the CL system destabilizes it.



(d) CL prediction errors of models identified with the constrained version of CL EDMD, before and after the plant is extracted and the controller is re-connected. Reconstructing the CL system does not change its predictions.

Figure 6.10: CL eigenvalues and prediction errors of models identified with the least-squares and constrained versions of CL EDMD.

Table 6.3: Comparison of system identification methods.

Method	Reg.	Scoring	Bounded score?	CL reg.?	Avoids bias?	Nonlinear?
ARX	—	CL	✓	—	✓	✗
EDMD	Plant	Plant	✗	✗	✗	✓
EDMD	Plant	CL	✓	✗	✗	✓
EDMD	CL	Plant	✗	✓	✓	✓
EDMD	CL	CL	✓	✓	✓	✓

wrapping it with the same controller does not result in the same system. Figure 6.10a shows how the closed-loop eigenvalues change when the plant is extracted and re-wrapped with a controller. In this case, the procedure actually destabilizes the system. When using the constraint-based approach, as shown in Figure 6.10b, extracting the plant and closing the loop again with the same controller does not move the eigenvalues. Figures 6.10c and 6.10d show the prediction errors of each system before and after removing and re-adding the controller. As expected, the predicted trajectories of the destabilized system diverge quickly. The ability to extract the plant from the closed-loop system while respecting the feedback structure of the system is particularly important for control design tasks, where, presumably, the plant is being identified with the end goal of designing an improved controller.

6.7 Conclusion

When identifying closed-loop systems, it is often not possible to neglect the effects of the control loop. This holds true for Koopman operator approximation as well as linear system identification. In this chapter, a closed-loop Koopman operator identification method is presented, where the closed-loop system and plant system are identified simultaneously using knowledge of the controller. The proposed method, built on the SDP of EDMD from Chapter 4, ensures consistency between the closed-loop and plant models while also avoiding having to directly identify an unstable plant. The advantages of this method are demonstrated in simulation using a Duffing oscillator with coloured measurement noise and experimentally using an unstable rotary inverted pendulum system. In particular, the proposed closed-loop method identifies Koopman models that align with the prior knowledge that the closed-loop system is AS, while the plant is unstable. The closed-loop method also naturally allows the closed-loop prediction error to be used as a goodness-of-fit metric, which is crucial for selecting lifting functions and an appropriate regularization coefficient. Furthermore, by identifying the plant system using constraints, the proposed closed-loop

approach avoids inflating the dimension of the plant system.

A limitation of the proposed approach is that including the controller state in the lifted state of the system increases its dimension, especially if complex controllers are used. While high-order controllers are generally undesirable in system identification applications, the assumption that the controller is known exactly does not always hold. Unmodelled effects like actuator saturation may limit a user's ability to correctly calculate the controller state, which could affect the accuracy of the method. The proposed closed-loop Koopman operator approximation method allows both the closed-loop and plant Koopman matrices to be used in regularizers or constraints. Using EDMD, only the plant's Koopman matrix can be used in a regularizer, leading to potentially unstable closed-loop systems for large regularization coefficients. The additional flexibility provided by the proposed closed-loop Koopman operator approximation method will be explored in future work, as the ability to incorporate known information about the closed-loop system or plant into the regression problem could prove useful in identifying more accurate or useful Koopman models. Extensions to address the use of nonlinear controllers, or to address the situation where the controller is also unknown, would widen the applicability of the proposed closed-loop Koopman operator approximation method.

Rather than focusing on Koopman operator identification, the following chapter discusses how Koopman models, potentially identified with the methods presented in Chapter 4, Chapter 5, or this chapter, can be used with standard optimal control techniques to quantify uncertainty and synthesize robust nonlinear observers.

Chapter 7

Uncertainty Modelling and Robust Estimation using the Koopman Operator

Summary

This chapter proposes a robust Koopman observer synthesis method based on mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control. This observer synthesis method is made possible by the Koopman operator's linearity, which allows uncertainty within a population of Koopman models to be quantified in the frequency domain. A population of several dozen motor drives is used to experimentally demonstrate the proposed method. Manufacturing variation is characterized in the frequency domain, and a robust Koopman observer is synthesized. The uncertainty model is shown to be sufficiently accurate to detect an outlier motor drive that was incorrectly installed.

7.1 Introduction

Due to the linearity of the Koopman operator, standard linear control tools can be used to synthesize optimal controllers and observers for nonlinear systems. Furthermore, uncertainty within a population of Koopman models can be expressed in the frequency domain thanks again to their linearity, opening the door for linear robust control. This chapter demonstrates how linear robust control tools can be leveraged to design nonlinear observers that are insensitive to uncertainty within a population of Koopman models. This chapter also shows how a sufficiently accurate frequency-domain uncertainty model can be used to detect outliers or faults in the Koopman operator identification process. The proposed methodology, summarized in Figure 7.1, is demonstrated experimentally through the uncertainty characterization of a population of several dozen motor drives exhibiting nonlinear oscillations. The resulting

uncertainty model is also shown to be accurate enough to identify outlier systems. The resulting nonlinear observer is shown to outperform its linear counterpart due, in part, to its more accurate uncertainty model.

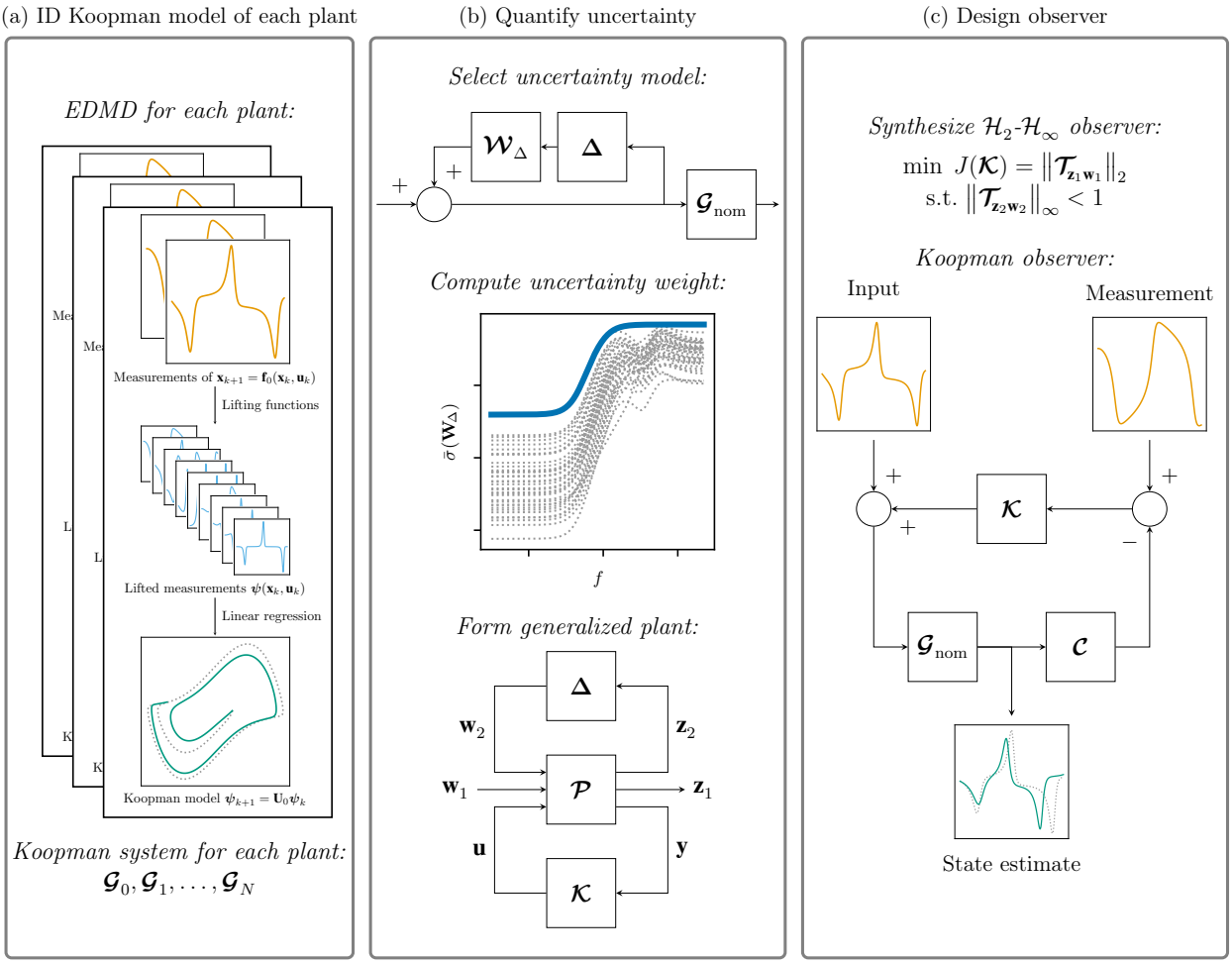


Figure 7.1: Overview of the proposed robust Koopman observer synthesis procedure. (a) First, a Koopman model is identified for each of N distinct plants in a population. (b) Then, an uncertainty model is selected, along with a nominal plant model. A residual for each off-nominal plant is computed, and an uncertainty weight is chosen to bound the residuals. The uncertainty weight, along with performance weights and the nominal plant model, are combined into the generalized plant, $\mathcal{P}$. Finally, the generalized plant is used to synthesize a mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ Koopman observer.

7.1.1 Related work

Ever since exogenous inputs were introduced into the Koopman operator framework in [55], interest in designing linear controllers in the lifted space has only grown. Approaches based on the linear-quadratic regulator (LQR) include [14, 15, 109, 110]. Using a simple example with an exact, finite-dimensional Koopman representation, [109] demonstrates that a

Koopman LQR outperforms a conventional LQR designed using the system's linearized dynamics. In [14], similar results are obtained using an approximate Koopman model of a forced Van der Pol oscillator. A Koopman-based LQR for a tail-actuated robotic fish is designed in [15] using hand-engineered, dynamics-based, lifted states. The controller performs similarly to a nonlinear backstepping controller, however it is much simpler to design. In [110], an architecture called Koopman Reduced Order Nonlinear Identification and Control (KRONIC) is proposed, which identifies Koopman eigenfunctions and synthesizes a Koopman LQR using those functions as the lifted states.

MPC is also a natural fit with the Koopman operator, as shown in [12, 16, 19, 22, 111]. Koopman MPC is most rigorously described in [12], where it is shown that MPC algorithms using Koopman models achieve similar computational complexity to their linear counterparts. Simulated examples are shown for bilinear motor control and nonlinear PDE control. In [22], a Koopman representation of a differential-drive mobile robot is used in open- and closed-loop MPC algorithms. The lifted states are chosen to be products of polynomials in the robot's Cartesian position and velocity. The proposed controller's performance is shown to increase as the dimension of the lifted state increases. Koopman MPC is used in [16] to control an underactuated soft robotic manipulator. Monomials with a single time delay are used as the lifted states. The Koopman-based MPC algorithm performs better than a standard MPC algorithm that uses a linear model. The manipulator's payload is incorporated into the soft robot's Koopman model and an observer is designed to estimate its value online in [111], improving the performance of the controller.

Robust LTI controller synthesis using the Koopman operator is considered in [17, 79–82]. In [17], $\mathcal{H}_2$ optimal control is used to synthesize controllers that perform robustly in the face of uncertainty modelled by polytope sets of Koopman models. Uncertainty in the Koopman representation used in [17] is due to the use of multiple datasets to identify multiple models of the same system. The synthesized controllers are tested in simulation using a Duffing oscillator system and the Kroteweg–De Vries PDE. Uncertainty due to the finite-dimensional approximation of an infinite-dimensional Koopman operator is represented in [79, 80] through the use of incremental quadratic constraints. Controllers are then designed to guarantee stability in the largest possible regions of attraction and are tested on simulated Van der Pol oscillator [79] and inverted pendulum [80] systems. Biased Koopman models identified from noisy data are considered in [81], where uncertainty is modelled based on a known sector bounded model mismatch. A dual-loop $\mathcal{H}_\infty$ control approach is used to attain both robust stability and robust performance guarantees. The approach is demonstrated using a simulated Van der Pol oscillator. In [82], robust $\mathcal{H}_\infty$ control is applied to a linear parameter-varying Koopman representation. Uncertainty due to the linear parameter-

varying nature of the model, along with known model approximation error, is modelled in the frequency domain. The uncertainty is bounded using constant weighting functions in an additive uncertainty representation. A mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ control problem is solved to guarantee the robust stability of the Koopman controller. The performance of the synthesized controller is demonstrated on a simulated bilinear motor model. While robustness is not considered in [112], a sophisticated $\mathcal{H}_\infty$ design procedure with dynamic performance weighting functions is outlined in the lifted space. The synthesized $\mathcal{H}_\infty$ controller is demonstrated on simulated Van der Pol oscillator and two-mass-Duffing-spring systems. Koopman-based MPC algorithms that consider other definitions of robustness include [113–116].

Early work using the Koopman operator for state estimation introduces the Koopman observer form, a state-space form computed from the estimated eigenvalues and eigenfunctions of the Koopman operator [117, 118]. This form is then used to design a Luenberger observer or Kalman filter. In [119], sparsity-promoting DMD is used to estimate the Koopman observer form and design a Koopman Kalman filter to estimate the flow field near an actuated airfoil from pressure measurements. A generalized maximum likelihood variant of the Koopman Kalman filter is introduced in [120], where it is used to estimate the rotor angle and speed of a series of synchronous generators. In [121], a Koopman Luenberger observer is designed to detect faulty actuators in a multirotor system. Koopman observer design for models with bilinear lifting functions is considered in [122]. Koopman state estimation in a batch optimization framework is discussed in [24, 123], while simultaneous localization and mapping is additionally considered in [123].

Outside the Koopman literature, observer synthesis methods for specific types of uncertain nonlinear systems are considered. Gain-scheduled linear observers are designed in [124–126] for linear and bilinear parameter varying systems. $\mathcal{H}_\infty$ optimal observers for systems with Lipschitz nonlinearities are synthesized in [127–130]. A sliding-mode $\mathcal{H}_\infty$ optimal observer for Lipschitz nonlinear systems is proposed in [131].

7.1.2 Contribution

The key contribution of this chapter is the use of linear robust control tools to quantify the uncertainty within a population of Koopman models and to synthesize robust nonlinear observers for uncertain systems modelled with the Koopman operator. Specific contributions include the collection of a dataset including trajectories from 38 individual motor drives in unloaded and loaded conditions, the quantification of manufacturing uncertainty within the dataset, the consideration of multiple frequency-domain uncertainty forms in a Koopman operator identification context, the use of nontrivial uncertainty weighting functions to identify outliers and synthesize a robust Koopman $\mathcal{H}_2$-$\mathcal{H}_\infty$ observer, and the experimental, rather

than simulated, validation of the proposed approach.

The full motor drive dataset is available for download at [5]. A datasheet following the format of [132] is also provided. The software required to fully reproduce the results of this chapter is available at `https://github.com/decargroup/robust_observer_koopman`.

7.2 Robust Control

This section outlines the robust control fundamentals required to quantify uncertainty within a population of LTI models and synthesize a robust controller or observer using that uncertainty model.

7.2.1 Nominal stability

The synthesis of an LTI controller that asymptotically stabilizes a known plant is considered

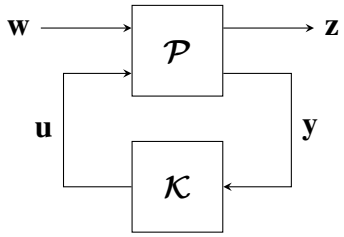


Figure 7.2: The generalized plant $\mathcal{P}$ for a nominal control problem in feedback with a controller $\mathcal{K}$.

in this section.

The concept of the generalized plant forms the basis for modern optimal control [101, §3.8]. The generalized plant $\mathcal{P}$, shown in Figure 7.2, has two sets of inputs and outputs. The controller inputs and outputs, $\mathbf{u}$ and $\mathbf{y}$, represent the signals accessible to a controller $\mathcal{K}$. The performance inputs and outputs, $\mathbf{w}$ and $\mathbf{z}$, encode performance requirements for the optimization problem. It is important to note that the generalized plant is not the plant to be controlled, $\mathcal{G}$, but rather it contains $\mathcal{G}$, along with additional performance inputs and outputs. The performance inputs and outputs exist for the purpose of optimization only, and may not ever be computed when the controller is deployed. Examples of performance inputs include reference signals, disturbances, and measurement noise. Examples of performance outputs include tracking error and control effort.

Consider the LTI system $\mathcal{T}$, which represents the closed-loop system from the performance inputs $\mathbf{w}$ to the performance outputs $\mathbf{y}$. That is, $\mathbf{z} = \mathcal{T}\mathbf{w}$. An optimal controller $\mathcal{K}$ is chosen to minimize a system norm of $\mathcal{T}$. Common choices of system norm include the $\mathcal{H}_2$ norm and the $\mathcal{H}_\infty$ norm. The particular choice of norm, performance input, and

performance output is problem-dependent. For example, if $\mathbf{w}$ contains a disturbance and $\mathbf{z}$ contains tracking error, then the solution to the optimal control problem

$$\min_{\mathcal{K}} \|\mathcal{T}\|_\infty \tag{7.1}$$

can be seen as minimizing the worst-case impact of exogenous disturbances on tracking error. Posing an optimal control problem in this manner guarantees the asymptotic stability of the closed-loop system as long as a system norm of $\mathcal{T}$ is finite [36].

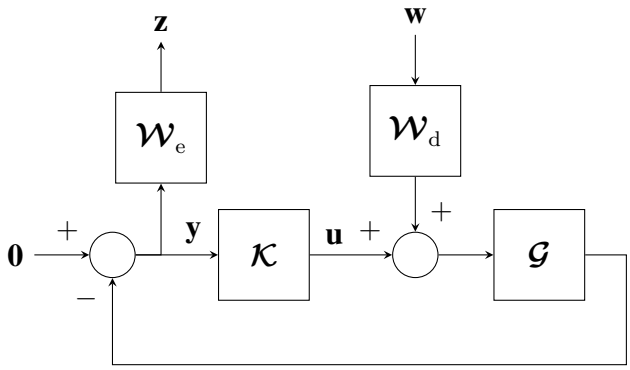


Figure 7.3: Example of a generalized plant for a disturbance rejection problem. The disturbance input is weighted to indicate the range of frequencies occupied by the disturbance signal, while the error output is weighted to specify the frequencies over which errors should be minimized.

Without any further specification, an optimal control problem minimizes $\|\mathcal{T}\|$ equally over all frequencies. However, reference signals, disturbances, and noise occupy known frequency bands. Furthermore, specifications related to tracking error and control effort are only relevant in some frequency bands. As such, performance inputs and outputs are often weighted with LTI systems called *weighting functions*. For example, a noise input would be weighted with a high-pass filter, as the effects of noise are most dominant at high frequencies. A reference input would be weighted with a low-pass filter, as reference signals typically occupy low-frequencies. Weighting functions can encode specific performance requirements that, when paired with the $\mathcal{H}_\infty$ norm, are guaranteed to be met when $\|\mathcal{T}\|_\infty < 1$. An example of a weighted generalized plant for a disturbance rejection problem can be found in Figure 7.3.

7.2.2 Robust stability

Performance inputs and outputs can also be used to enforce robustness requirements on the synthesized controller. Consider Figure 7.4, where $\mathbf{\Delta}$ is an AS system representing a

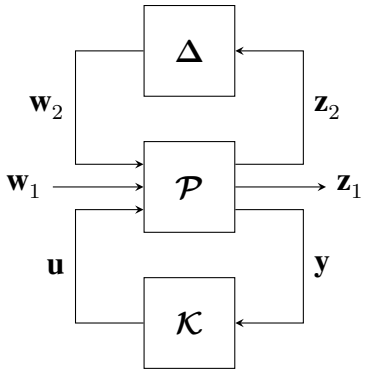


Figure 7.4: The generalized plant $\boldsymbol{\mathcal{P}}$ for the robust optimal control problem in feedback with a controller $\boldsymbol{\mathcal{K}}$ and an uncertainty block $\boldsymbol{\Delta}$.

perturbation. Consider the LTI system $\boldsymbol{\mathcal{T}}_{22}$, where

$$\mathbf{z}_2 = \boldsymbol{\mathcal{T}}_{22}\mathbf{w}_2. \tag{7.2}$$

Assuming that $\boldsymbol{\mathcal{K}}$ asymptotically stabilizes $\boldsymbol{\mathcal{P}}$ for $\boldsymbol{\Delta} = \mathbf{0}$, the small-gain theorem [101, §4.9.4] implies that the closed-loop system in Figure 7.4 is AS for any $\|\boldsymbol{\Delta}\|_\infty \leq 1$ if $\|\boldsymbol{\mathcal{T}}_{22}\|_\infty < 1$. This fact is used in the robust control framework to synthesize controllers that stabilize any plant in a modelled uncertainty set.

To model the uncertainty for a given population of systems, an uncertainty structure must be chosen, often from among those pictured in Figure 7.5. Like with the performance channels, weighting functions are used to characterize uncertainty in the frequency domain. Working concretely with transfer matrices now, consider a nominal plant $\mathbf{G}(z)$, a perturbed plant $\mathbf{G}_\mathrm{p}(z)$, and a perturbation, or residual, $\mathbf{E}(z)$. The perturbation can be expressed as

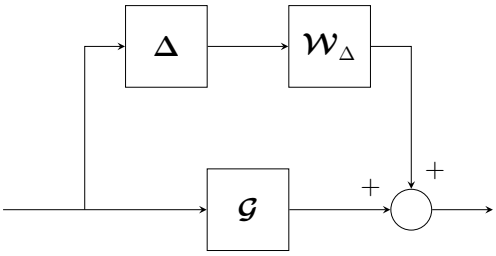
$$\mathbf{E}(z) = \mathbf{W}_2(z)\boldsymbol{\Delta}(z)\mathbf{W}_1(z), \quad \|\boldsymbol{\Delta}(z)\|_\infty \leq 1, \tag{7.3}$$

where often one of the weighting functions is set to identity. The additive uncertainty model corresponds to

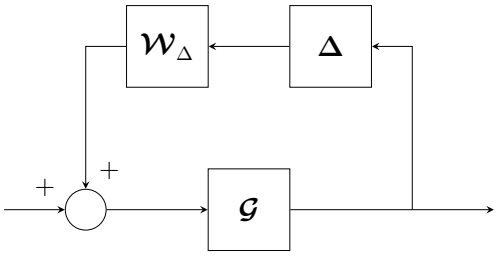
$$\mathbf{G}_\mathrm{p}(z) = \mathbf{G}(z) + \mathbf{E}_\mathrm{a}(z). \tag{7.4}$$

Let $\mathbf{W}_2(z) = \mathbf{1}$. The remaining weighting function $\mathbf{W}_1(z)$ is designed to satisfy

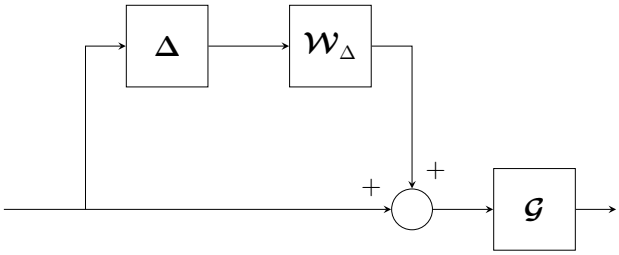
$$\begin{aligned} \|\mathbf{G}_\mathrm{p}(z) - \mathbf{G}(z)\|_\infty &= \|\boldsymbol{\Delta}(z)\mathbf{W}_1(z)\|_\infty && (7.5)\\ &\leq \|\boldsymbol{\Delta}(z)\|_\infty \|\mathbf{W}_1(z)\|_\infty && (7.6)\\ &\leq \|\mathbf{W}_1(z)\|_\infty && (7.7) \end{aligned}$$



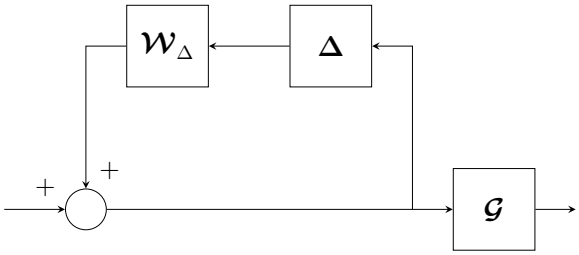
(a) Additive uncertainty.



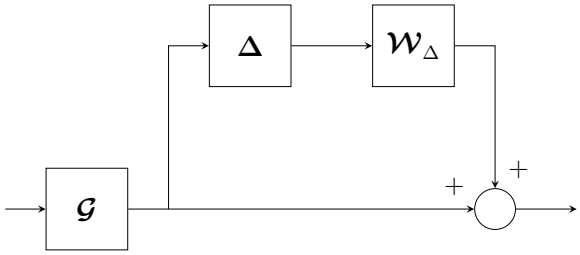
(b) Inverse additive uncertainty.



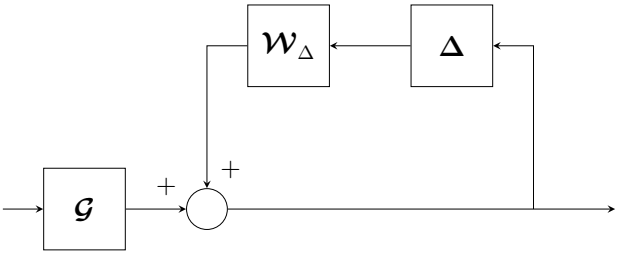
(c) Input multiplicative uncertainty.



(d) Inverse input multiplicative uncertainty.



(e) Output multiplicative uncertainty.



(f) Inverse output multiplicative uncertainty.

Figure 7.5: Six possible unstructured uncertainty forms [101, §8.2.3]

for all perturbed plants. The remaining standard uncertainty forms are

- input multiplicative, where $\mathbf{G}_{\mathrm{p}}(z) = \mathbf{G}(z)\left(\mathbf{1} + \mathbf{E}_{\mathrm{i}}(z)\right)$,
- output multiplicative, where $\mathbf{G}_{\mathrm{p}}(z) = \left(\mathbf{1} + \mathbf{E}_{\mathrm{o}}(z)\right)\mathbf{G}(z)$,
- inverse additive, where $\mathbf{G}_{\mathrm{p}}(z) = \mathbf{G}(z)(\mathbf{1} - \mathbf{E}_{\mathrm{ia}}(z)\mathbf{G}(z))^{-1}$,
- inverse input multiplicative, where $\mathbf{G}_{\mathrm{p}}(z) = \mathbf{G}(z)(\mathbf{1} - \mathbf{E}_{\mathrm{ii}}(z))^{-1}$, and
- inverse output multiplicative, where $\mathbf{G}_{\mathrm{p}}(z) = (\mathbf{1} - \mathbf{E}_{\mathrm{io}}(z))^{-1}\mathbf{G}(z)$.

Given a population of perturbed plants, each perturbation $\mathbf{E}(z)$ is typically calculated frequency-by-frequency. An uncertainty weight $\mathbf{W}(z)$ that bounds all the perturbations is then designed manually or found using an optimization procedure.

For all uncertainty forms, a nominal plant must be chosen. If no nominal model is directly available, an average model can be computed. However, the choice of parameters to average, be they transfer matrix coefficients or canonical state-space matrices, is not obvious. In data-driven applications, a nominal model can often be computed by identifying a model using the combined training data from all the plants. In some situations, one plant out of the population can simply be selected as the nominal plant. Both the choice of uncertainty structure and the choice of nominal model have a significant impact on the magnitude of the uncertainty bound and therefore on the performance of the controller or observer being designed.

Let

$$\mathbf{z}_1 = \boldsymbol{\mathcal{T}}_{11}\mathbf{w}_1, \tag{7.8}$$

$$\begin{bmatrix}\mathbf{z}_1 \\ \mathbf{z}_2\end{bmatrix} = \boldsymbol{\mathcal{T}}\begin{bmatrix}\mathbf{w}_1 \\ \mathbf{w}_2\end{bmatrix}. \tag{7.9}$$

Given the generalized plant in Figure 7.4, multiple controller synthesis methods can synthesize a robustly stabilizing controller, two of which are discussed here. The $\mathcal{H}_\infty$ optimal controller achieves robust stability by solving [101, §9.3]

$$\gamma = \arg\min_{\boldsymbol{\mathcal{K}}} \ \|\boldsymbol{\mathcal{T}}\|_\infty \tag{7.10}$$

and verifying that $\gamma < 1$. Another option is mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ robust control, where the opti-

mization problem [133–135]

$$\min_{\boldsymbol{\kappa}} \|\boldsymbol{\mathcal{T}}_{11}\|_2 \tag{7.11}$$

$$\text{s.t. } \|\boldsymbol{\mathcal{T}}_{22}\|_\infty < 1, \tag{7.12}$$

is solved. In the next section, concrete approaches to solving these optimization problems over LTI systems are outlined.

7.2.3 Controller synthesis as semidefinite programming

In this section, one SDP formulation of the mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control problem is shown. Other optimal control problems, like the $\mathcal{H}_\infty$ optimal control problem, have similar formulations, but with different LMI constraints. A comprehensive list of SDP formulations of optimal control and observer design problems can be found in [31].

Consider the discrete-time generalized plant

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \begin{bmatrix}\mathbf{B}_{1,1} & \mathbf{B}_{1,2}\end{bmatrix} \begin{bmatrix}\mathbf{w}_{1,k} \\ \mathbf{w}_{2,k}\end{bmatrix} + \mathbf{B}_2\mathbf{u}_k, \tag{7.13}$$

$$\begin{bmatrix}\mathbf{z}_{1,k} \\ \mathbf{z}_{2,k}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_{1,1} \\ \mathbf{C}_{1,2}\end{bmatrix} \mathbf{x}_k + \begin{bmatrix}\mathbf{D}_{11,11} & \mathbf{D}_{11,12} \\ \mathbf{D}_{11,21} & \mathbf{D}_{11,22}\end{bmatrix} \begin{bmatrix}\mathbf{w}_{1,k} \\ \mathbf{w}_{2,k}\end{bmatrix} + \begin{bmatrix}\mathbf{D}_{12,1} \\ \mathbf{D}_{12,2}\end{bmatrix} \mathbf{u}_k, \tag{7.14}$$

$$\mathbf{y}_k = \mathbf{C}_2\mathbf{x}_k + \begin{bmatrix}\mathbf{D}_{21,1} & \mathbf{D}_{21,2}\end{bmatrix} \begin{bmatrix}\mathbf{w}_{1,k} \\ \mathbf{w}_{2,k}\end{bmatrix} + \mathbf{D}_{22}\mathbf{u}_k, \tag{7.15}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x \times 1}$, $\mathbf{w}_{1,k} \in \mathbb{R}^{n_{w_1} \times 1}$, $\mathbf{w}_{2,k} \in \mathbb{R}^{n_{w_2} \times 1}$, $\mathbf{u}_k \in \mathbb{R}^{n_u \times 1}$, $\mathbf{y}_k \in \mathbb{R}^{n_y \times 1}$, $\mathbf{z}_{1,k} \in \mathbb{R}^{n_{z_1} \times 1}$, and $\mathbf{z}_{2,k} \in \mathbb{R}^{n_{z_2} \times 1}$. The state-space matrices have the corresponding dimensions. The mixed

$\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal control problem in SDP form is [31, §5.4.4]

$$\min J(\mu, \mathbf{A}^{\mathrm{n}}, \mathbf{B}^{\mathrm{n}}, \mathbf{C}^{\mathrm{n}}, \mathbf{D}^{\mathrm{n}}, \mathbf{X}_1, \mathbf{Y}_1, \mathbf{Z}; \gamma) = \mu \tag{7.16}$$

$$\text{s.t. } \mathbf{X}_1 > 0, \tag{7.17}$$

$$\mathbf{Y}_1 > 0, \tag{7.18}$$

$$\mathbf{Z} > 0, \tag{7.19}$$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{1} & \mathbf{X}_1\mathbf{A} + \mathbf{B}^{\mathrm{n}}\mathbf{C}_2 & \mathbf{A}^{\mathrm{n}} & \mathbf{X}_1\mathbf{B}_{1,1} + \mathbf{B}^{\mathrm{n}}\mathbf{D}_{21,1} \\ * & \mathbf{Y}_1 & \mathbf{A} + \mathbf{B}_2\mathbf{D}^{\mathrm{n}}\mathbf{C}_2 & \mathbf{A}\mathbf{Y}_1 + \mathbf{B}_2\mathbf{C}^{\mathrm{n}} & \mathbf{B}_{1,1} + \mathbf{B}_2\mathbf{D}^{\mathrm{n}}\mathbf{D}_{21,1} \\ * & * & \mathbf{X}_1 & \mathbf{1} & \mathbf{0} \\ * & * & * & \mathbf{Y}_1 & \mathbf{0} \\ * & * & * & * & \mathbf{1} \end{bmatrix} > 0, \tag{7.20}$$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{1} & \mathbf{X}_1\mathbf{A} + \mathbf{B}^{\mathrm{n}}\mathbf{C}_2 & \mathbf{A}^{\mathrm{n}} & \mathbf{X}_1\mathbf{B}_{1,2} + \mathbf{B}^{\mathrm{n}}\mathbf{D}_{21,2} & \mathbf{0} \\ * & \mathbf{Y}_1 & \mathbf{A} + \mathbf{B}_2\mathbf{D}^{\mathrm{n}}\mathbf{C}_2 & \mathbf{A}\mathbf{Y}_1 + \mathbf{B}_2\mathbf{C}^{\mathrm{n}} & \mathbf{B}_{1,2} + \mathbf{B}_2\mathbf{D}^{\mathrm{n}}\mathbf{D}_{21,2} & \mathbf{0} \\ * & * & \mathbf{X}_1 & \mathbf{1} & \mathbf{0} & \mathbf{C}_{1,2}^{\mathsf{T}} + \mathbf{C}_2^{\mathsf{T}}\mathbf{D}^{\mathrm{n}^{\mathsf{T}}}\mathbf{D}_{12,2}^{\mathsf{T}} \\ * & * & * & \mathbf{Y}_1 & \mathbf{0} & \mathbf{Y}_1\mathbf{C}_{1,2}^{\mathsf{T}} + \mathbf{C}^{\mathrm{n}^{\mathsf{T}}}\mathbf{D}_{12,2}^{\mathsf{T}} \\ * & * & * & * & \gamma\mathbf{1} & \mathbf{D}_{11,22}^{\mathsf{T}} + \mathbf{D}_{21,2}^{\mathsf{T}}\mathbf{D}^{\mathrm{n}^{\mathsf{T}}}\mathbf{D}_{12,2}^{\mathsf{T}} \\ * & * & * & * & * & \gamma\mathbf{1} \end{bmatrix} > 0, \tag{7.21}$$

$$\begin{bmatrix} \mathbf{Z} & \mathbf{C}_{1,1} + \mathbf{D}_{12,1}\mathbf{D}^{\mathrm{n}}\mathbf{C}_2 & \mathbf{C}_{1,1}\mathbf{Y}_1 + \mathbf{D}_{12,1}\mathbf{C}^{\mathrm{n}} \\ * & \mathbf{X}_1 & \mathbf{1} \\ * & * & \mathbf{Y}_1 \end{bmatrix} > 0, \tag{7.22}$$

$$\mathbf{D}_{11,11} + \mathbf{D}_{12,1}\mathbf{D}^{\mathrm{n}}\mathbf{D}_{21,1} = \mathbf{0}, \tag{7.23}$$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{1} \\ * & \mathbf{Y}_1 \end{bmatrix} > 0, \tag{7.24}$$

$$\mathrm{tr}(\mathbf{Z}) < \mu, \tag{7.25}$$

where $\mu \in \mathbb{R}_{\geq 0}$, $\gamma \in \mathbb{R}_{\geq 0}$, $\mathbf{A}^{\mathrm{n}} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B}^{\mathrm{n}} \in \mathbb{R}^{n_x \times n_y}$, $\mathbf{C}^{\mathrm{n}} \in \mathbb{R}^{n_u \times n_x}$, $\mathbf{D}^{\mathrm{n}} \in \mathbb{R}^{n_u \times n_y}$, $\mathbf{X}_1 = \mathbf{X}_1^{\mathsf{T}} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{Y}_1 = \mathbf{Y}_1^{\mathsf{T}} \in \mathbb{R}^{n_x \times n_x}$, and $\mathbf{Z} = \mathbf{Z}^{\mathsf{T}} \in \mathbb{R}^{n_{z_1} \times n_{z_1}}$. The state-space matrices of the controller are [31, §5.4.4]

$$\mathbf{D}^{\mathrm{c}} = \left(\mathbf{1} + \hat{\mathbf{D}}^{\mathrm{c}}\mathbf{D}_{22}\right)^{-1}\hat{\mathbf{D}}^{\mathrm{c}}, \tag{7.26}$$

$$\mathbf{C}^{\mathrm{c}} = \left(\mathbf{1} - \mathbf{D}^{\mathrm{c}}\mathbf{D}_{22}\right)\hat{\mathbf{C}}^{\mathrm{c}}, \tag{7.27}$$

$$\mathbf{B}^{\mathrm{c}} = \hat{\mathbf{B}}^{\mathrm{c}}\left(\mathbf{1} - \mathbf{D}_{22}\mathbf{D}^{\mathrm{c}}\right), \tag{7.28}$$

$$\mathbf{A}^{\mathrm{c}} = \hat{\mathbf{A}}^{\mathrm{c}} - \mathbf{B}^{\mathrm{c}}(\mathbf{1} - \mathbf{D}_{22}\mathbf{D}^{\mathrm{c}})^{-1}\mathbf{D}_{22}\mathbf{C}^{\mathrm{c}}, \tag{7.29}$$

where

$$\begin{bmatrix} \hat{\mathbf{A}}^{\mathrm{c}} & \hat{\mathbf{B}}^{\mathrm{c}} \\ \hat{\mathbf{C}}^{\mathrm{c}} & \hat{\mathbf{D}}^{\mathrm{c}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_2 & \mathbf{X}_1\mathbf{B}_2 \\ \mathbf{0} & \mathbf{1} \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{A}^{\mathrm{n}} & \mathbf{B}^{\mathrm{n}} \\ \mathbf{C}^{\mathrm{n}} & \mathbf{D}^{\mathrm{n}} \end{bmatrix} - \begin{bmatrix} \mathbf{X}_1\mathbf{A}\mathbf{Y}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right) \begin{bmatrix} \mathbf{Y}_2^{\mathsf{T}} & \mathbf{0} \\ \mathbf{C}_2\mathbf{Y}_1 & \mathbf{1} \end{bmatrix}^{-1} \tag{7.30}$$

and

$$\mathbf{X}_2\mathbf{Y}_2^{\mathsf{T}} = \mathbf{1} - \mathbf{X}_1\mathbf{Y}_1. \tag{7.31}$$

The value of γ must be chosen in advance. To guarantee robust stability, $\gamma = 1$ is typically used.

To compute the state-space representation of the mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal controller,

1. solve the SDP in (7.16)–(7.25) to get $(\mathbf{A}^{\mathrm{n}}, \mathbf{B}^{\mathrm{n}}, \mathbf{C}^{\mathrm{n}}, \mathbf{D}^{\mathrm{n}})$, $\mathbf{X}_1$, $\mathbf{Y}_1$, and $\mathbf{Z}$,
2. compute $\mathbf{X}_2$ and $\mathbf{Y}_2$ using a matrix decomposition of (7.31),
3. compute $(\hat{\mathbf{A}}^{\mathrm{c}}, \hat{\mathbf{B}}^{\mathrm{c}}, \hat{\mathbf{C}}^{\mathrm{c}}, \hat{\mathbf{D}}^{\mathrm{c}})$ using (7.30), and
4. compute $(\mathbf{A}^{\mathrm{c}}, \mathbf{B}^{\mathrm{c}}, \mathbf{C}^{\mathrm{c}}, \mathbf{D}^{\mathrm{c}})$ using (7.26)–(7.29).

The final controller is

$$\mathcal{K} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathrm{c}} & \mathbf{B}^{\mathrm{c}} \\ \hline \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} \end{array}\right]. \tag{7.32}$$

In this chapter, $\mathbf{X}_2$ and $\mathbf{Y}_2$ are computed using the SVD. That is,

$$\mathbf{X}_2 = \mathbf{Q}\sqrt{\mathbf{\Sigma}} \tag{7.33}$$

$$\mathbf{Y}_2 = \mathbf{Z}\sqrt{\mathbf{\Sigma}} \tag{7.34}$$

where

$$\mathbf{Q}\mathbf{\Sigma}\mathbf{Z}^{\mathsf{T}} = \mathbf{1} - \mathbf{X}_1\mathbf{Y}_1. \tag{7.35}$$

7.2.4 Optimal observer synthesis

Optimal observer design can be viewed as optimal controller design with a particular choice of generalized plant. This section provides a brief overview of linear observers and their relationship to linear controllers. Consider the state-space representation of a strictly proper LTI system,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{7.36}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k, \tag{7.37}$$

where $(\mathbf{A}, \mathbf{C})$ is observable. The structure of the most common type of observer, the *Luenberger observer*, is [125]

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k + \mathbf{L}\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right), \tag{7.38}$$

$$\hat{\mathbf{y}}_k = \mathbf{C}\hat{\mathbf{x}}_k, \tag{7.39}$$

where $\mathbf{L}$ is the observer gain. The observer gain is chosen to asymptotically stabilize the error dynamics [125]

$$\mathbf{e}_{k+1} = (\mathbf{A} - \mathbf{L}\mathbf{C})\,\mathbf{e}_k, \tag{7.40}$$

where $\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k$.

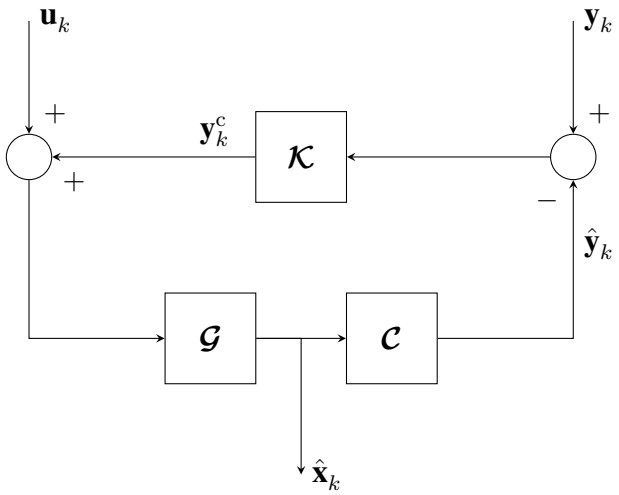


Figure 7.6: The structure of an input-output observer. Note that the output matrix $\mathbf{C}$ is drawn as a separate system from $\mathcal{G}$ to allow access to the state estimate.

However, observers need not be limited to static gains. In [125, 136], robust optimal control techniques are used to synthesize *input-output observers* of the form

$$\mathbf{x}_{k+1}^c = \mathbf{A}^c\mathbf{x}_k^c + \mathbf{B}^c\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right), \tag{7.41}$$

$$\mathbf{y}_k^c = \mathbf{C}^c\mathbf{x}_k^c + \mathbf{D}^c\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right), \tag{7.42}$$

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\left(\mathbf{u}_k + \mathbf{y}_k^c\right), \tag{7.43}$$

$$\hat{\mathbf{y}}_k = \mathbf{C}\hat{\mathbf{x}}_k. \tag{7.44}$$

A block diagram of an input-output observer is shown in Figure 7.6, where

$$\boldsymbol{\mathcal{K}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathrm{c}} & \mathbf{B}^{\mathrm{c}} \\ \hline \mathbf{C}^{\mathrm{c}} & \mathbf{D}^{\mathrm{c}} \end{array}\right], \tag{7.45}$$

$$\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{G}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{0} \end{array}\right]. \tag{7.46}$$

A copy of the nominal plant is used for comparison against measurements. The synthesized controller compares the measurements with the nominal model's output and makes corrections to the nominal system's input accordingly. Note that the output matrix $\mathbf{C}$ is drawn as a separate system $\boldsymbol{\mathcal{C}}$ in Figure 7.6, to allow access to the state estimate $\hat{\mathbf{x}}_k$ in the block diagram. The input-output observer is the structure of choice for the experimental example in this chapter. The generalized plant for that specific estimation problem, which is identical for the linear and Koopman observer problems, is shown in Figure 7.7, demonstrating what the generalized plant for a robust estimation problem looks like. The specific design choices made in Figure 7.7 will be detailed in Section 7.4.

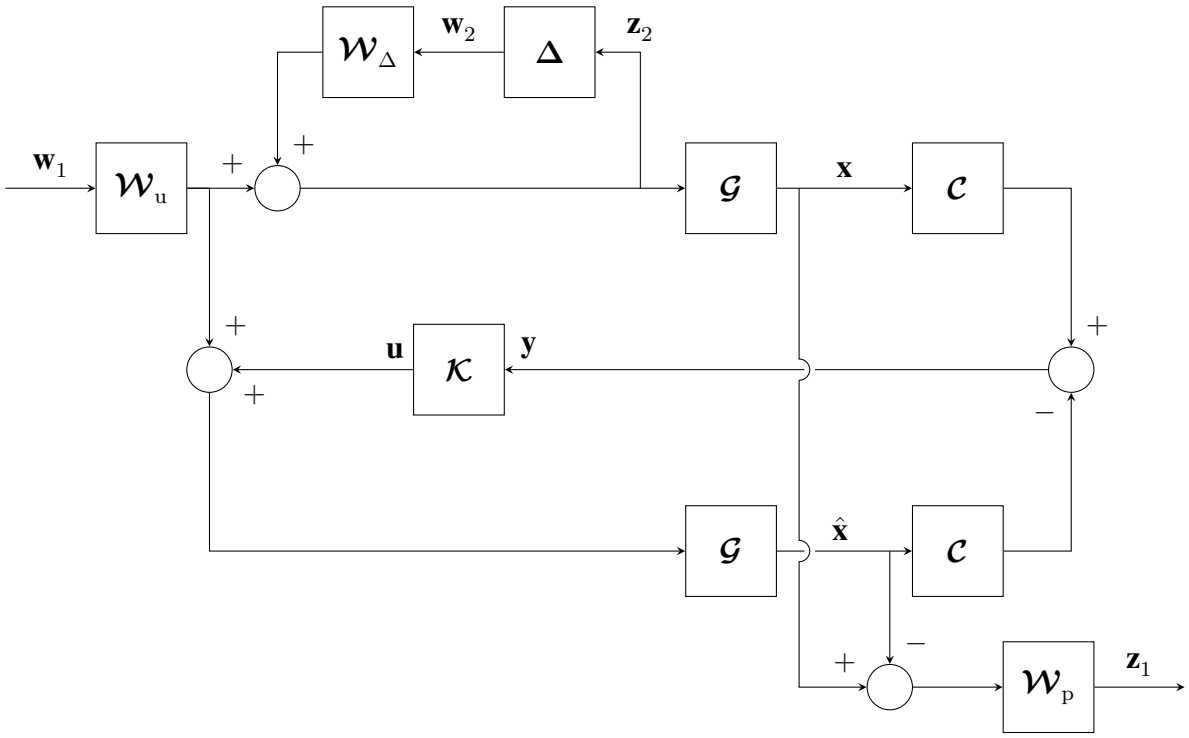


Figure 7.7: The generalized plant for a robust input-output observer problem with inverse input multiplicative uncertainty.

7.3 Optimal Control and Estimation with Koopman Models

Thanks to the linearity of the Koopman operator, the optimal control and estimation techniques outlined in this section can be applied to Koopman models with little modification, as the control design process takes place in the lifted space. One important consideration is that, as mentioned in Section 3.5.5, the control inputs are typically not lifted in a Koopman model intended for control. This is to ensure that the lifted control input can be uniquely retracted and applied to the system. Also, when comparing Koopman controllers to their linear counterparts, weights on lifted states should be zero to ensure that both methods minimize the same cost functions.

7.4 Experimental Example: Robust Observer for a Population of Motor Drives

In this section, the robust control synthesis tools outlined in Section 7.2 are applied to synthesize a robust Koopman observer for a population of 38 motor drives. Each motor drive, pictured in Figure 7.8, consists of a motor and a Harmonic Drive gearbox. The gearbox introduces a periodic oscillation into the system which cannot be captured by a linear model. The motor drives operate in closed-loop, accepting position and velocity reference signals and returning position, velocity, and current measurements. The velocity measurements are computed from position measurements by the drives via a filtered finite difference scheme.

Once a Koopman model for each of the 38 drives is identified, manufacturing uncertainty is characterized in the frequency domain and a robust Koopman observer is designed to estimate motor velocity and current using position measurements. Each motor drive is identified with and without an inertial load. The models used for observer synthesis are identified using unloaded drives, so the motor current estimation errors can be treated as load torque estimates. The uncertainty characterization is shown to be sufficiently accurate to identify an outlier drive that was installed in its mount with incorrect fastener torques.

7.4.1 Dataset

The motor drive dataset consists of 40 episodes per drive, each approximately 20 s long. The drive is loaded with an inertial load, pictured in Figure 7.9, for 20 of the 40 episodes. The loaded and unloaded datasets are split into 18 training episodes and two test episodes. The reference position, reference velocity, measured position, measured velocity, and measured motor current are recorded at 1 kHz. The positions and velocities are recorded in rad and rad/s respectively at the output shaft of the gearbox, while the current is recorded as a

Figure 7.8: Motor drive used to generate the training data, which consists of a motor with a Harmonic Drive gearbox. The gearbox introduces nonlinear oscillations into the system, leading to tracking errors at specific frequencies related to the input velocity. Photo courtesy of Alexandre Coulombe.

Figure 7.9: Rendering of the asymmetric inertial load used when collecting the motor drive dataset.

fraction of the drive's full-scale current. Figure 7.10 shows a portion of one of the test

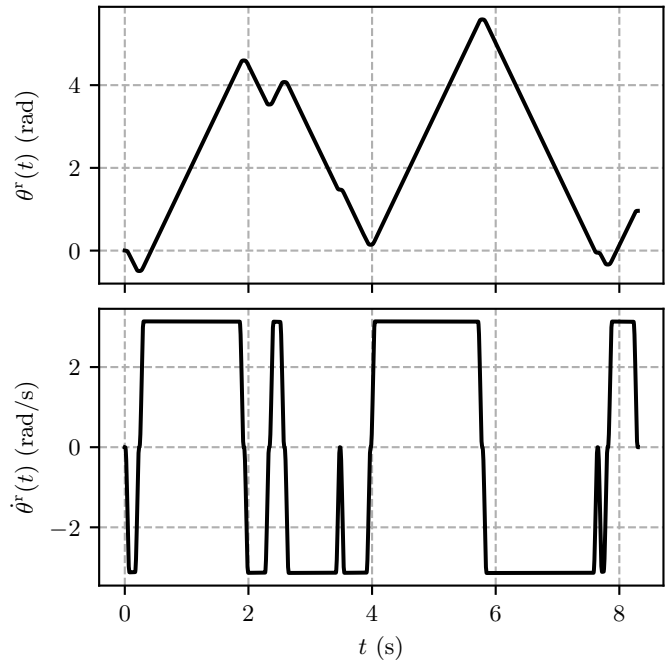


Figure 7.10: Reference position $\theta^{\mathrm{r}}(t)$ and velocity $\dot{\theta}^{\mathrm{r}}(t)$ for the closed-loop motor drive system from the first test episode. The velocity reference is a slightly smoothed PRBS, generated by setting pseudorandom position checkpoints at the maximum allowable velocity and acceleration in the drive's control software.

inputs used to characterize the motor drives. The drive's control software accepts position checkpoints and generates the smoothed trapezoidal velocity trajectories required to reach each checkpoint. Each episode contains 10 pseudorandom position checkpoints set within one revolution of the gearbox output shaft in either direction. The resulting velocity profile resembles a smoothed PRBS [100, §13.3]. For all episodes, the maximum allowed velocity and acceleration settings are set in the drive's control software to generate the most challenging training and test trajectories possible.

7.4.2 Koopman operator approximation

This section considers the Koopman operator identification and calibration procedure for a single motor drive. The lifting functions for the motor drive are chosen based on the nonlinear oscillations that are known to be present in Harmonic Drive gearboxes. The phase of this oscillation is a calibration parameter that must be identified for each motor drive. EDMD is then used to identify the Koopman matrix given the calibrated lifting functions.

7.4.2.1 Lifting function selection

To identify a Koopman model of a motor drive, lifting functions must first be selected. The motor drives to identify are equipped with Harmonic Drive gearboxes, which are known to generate vibrations at a frequency once and twice the input frequency [137, 138]. Let $\theta(t)$ be the gearbox output angle in rad and let $i(t)$ be the measured motor current as a fraction of the full-scale current. The nonlinear vibration can be modelled as an exogenous load torque disturbance, which is proportional to the current disturbance [137]

$$i^{\mathrm{d}}(t) = a_1 \sin\left(r\theta(t) + \varphi_1\right) + a_2 \sin\left(2r\theta(t) + \varphi_2\right), \tag{7.47}$$

where a_1 and a_2 are vibration amplitudes, φ_1 and φ_2 are phase shifts, and r is the reduction ratio of the gearbox. In this case, the gearbox multiplies torque by a factor of 100, so the reduction ratio is $r = 100$. These vibrations appear as tracking errors in the position and

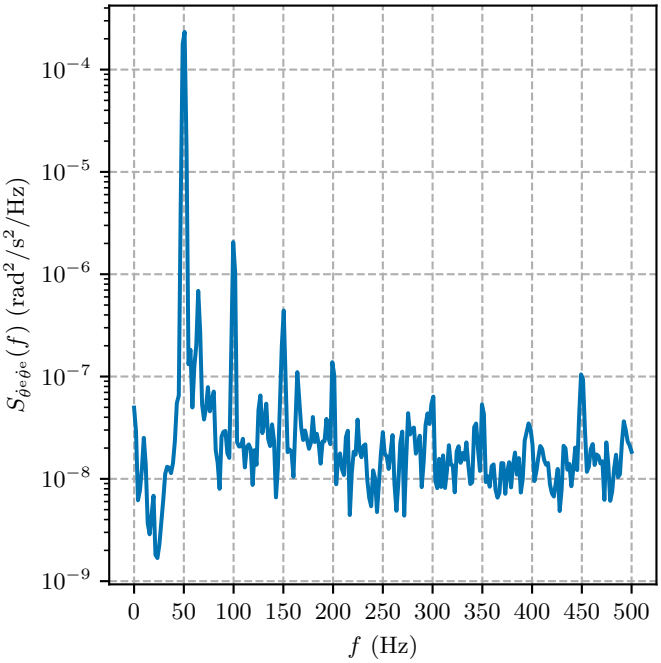


Figure 7.11: Power spectral density of output velocity tracking error during a constant-velocity trajectory segment. The velocity at the gearbox input is 50 rev/s, leading to vibrations at integer multiples of 50 Hz. The most prominent tracking errors occur at the fundamental frequency. A logarithmic scale is used to better show the high-frequency harmonics.

velocity of the drive. They also appear in the current command due to the feedback action of the controller. Figure 7.11 shows the power spectral density of a drive's velocity tracking error during a constant 0.5 rev/s movement, which is the drive's maximum speed. Since the gearbox reduction ratio is known to be $r = 100$, the resulting gearbox input velocity is

50 rev/s. Velocity tracking errors occur at integer multiples of 50 Hz, with decreasing power as frequency increases. By several orders of magnitude, the most significant tracking errors occur at 50 Hz.

Inspired by Figure 7.11 and by (7.47), the Koopman lifting functions for the motor drive system are chosen to be

$$\boldsymbol{\vartheta}_k = \begin{bmatrix} \theta_k \\ \dot{\theta}_k \\ \sin\left(100\,\theta_k + \varphi\right) \end{bmatrix}, \tag{7.48}$$

$$\boldsymbol{\upsilon}_k = i_k. \tag{7.49}$$

Since the 50 Hz error in Figure 7.11 is by far the largest, only that sinusoidal term is included in the lifting functions. It was found experimentally that including higher-frequency harmonics has a minimal effect on the system's overall prediction error. The vibration amplitude a_1 is identified as part of the approximated Koopman matrix.

Note the inclusion of a constant phase offset φ in (7.48). A calibration procedure for this fixed offset is proposed in Section 7.4.2.2.

7.4.2.2 Phase calibration

The Koopman lifting functions in (7.48) include a fixed calibration parameter for the phase of the nonlinear oscillation term, which must be determined before computing the Koopman matrix. Since this parameter is determined when the motor drive is being assembled, it is unknown but constant. Designing lifting functions to include both $\cos\left(100\,\theta_k\right)$ and $\sin\left(100\,\theta_k\right)$ could account for an unknown phase offset, absorbing it into the Koopman matrix. While this may be appropriate when identifying a single motor drive, it significantly and artificially increases uncertainty within a population of motor drives, which negatively impacts observer performance. Including the phase offset in the Koopman matrix also affects the selection of a nominal plant. Choosing a nominal plant with average damping or inertia is reasonable for robust controller or observer synthesis, but choosing a nominal Koopman model with an averaged phase offset does not result in useful predictions for off-nominal plants. Since the phase offset is constant and is easy to determine for each motor drive, removing it from the Koopman matrix by identifying it in advance is the preferred approach. By calibrating for the phase offset in advance, the uncertainty within the population of Koopman models better reflects the variation in the dynamics of each motor drive and gearbox.

In an earlier version of this work, the phase offset is determined through a hyperparameter optimization procedure [139]. However, a simpler and more reliable method is presented in

this chapter. To find the phase offset for a given drive, the position and velocity trajectories

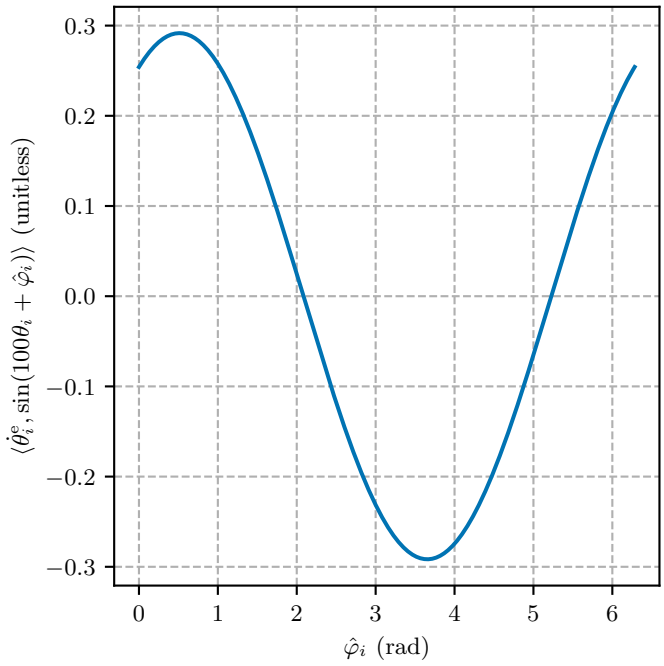


Figure 7.12: Inner product of the output velocity tracking error $\dot{\theta}_i^{\mathrm{e}}$ and the signal $\sin(100\theta_i + \hat{\varphi}_i)$ for phase shifts $\hat{\varphi}_i \in [0, 2\pi)$, where i is the segment index. The signal best aligns with the tracking error at $\varphi_i = 0.52\,\mathrm{rad}$.

are separated into constant-velocity segments. For each segment i, the velocity tracking error $\dot{\theta}_i^{\mathrm{e}}$ is calculated and normalized, and the optimum phase,

$$\varphi_i = \arg\max_{\hat{\varphi}_i} \left\langle \dot{\theta}_i^{\mathrm{e}},\, \sin(100\,\theta_i + \hat{\varphi}_i) \right\rangle, \tag{7.50}$$

is found by evaluating 1000 phase samples in $[0, 2\pi)$. The final phase offset is then calculated by averaging the optimal phase of each segment using the circular mean [140, §2.2.1],

$$\varphi = \operatorname{atan2}\left(\frac{1}{N} \sum_{i=1}^{N} \sin\varphi_i, \frac{1}{N} \sum_{i=1}^{N} \cos\varphi_i \right), \tag{7.51}$$

where N is the number of segments. To avoid this offline calibration procedure, a phase-locked loop could be used to estimate φ online from velocity tracking errors.

7.4.2.3 Koopman regression

EDMD with Tikhonov regularization is used to identify Koopman matrices using linear lifting functions and the lifting functions (7.48) and (7.49), along with the phase offset calibration

parameter φ. A manual bisection procedure is used to find the smallest regularization coefficient that results in stable Koopman systems for each drive, both with linear and nonlinear lifting functions. The Tikhonov regularization coefficient required to stabilize all Koopman models is $\alpha = 90$. More advanced methods like those in Chapter 5 could also be used to guarantee asymptotic stability without such a high regularization coefficient. However, in spite of the large regularization coefficient, the identified models are shown to result in good predictive performance and observer designs.

7.4.2.4 Prediction results

The predicted position, velocity, and current trajectories of one motor drive are shown in

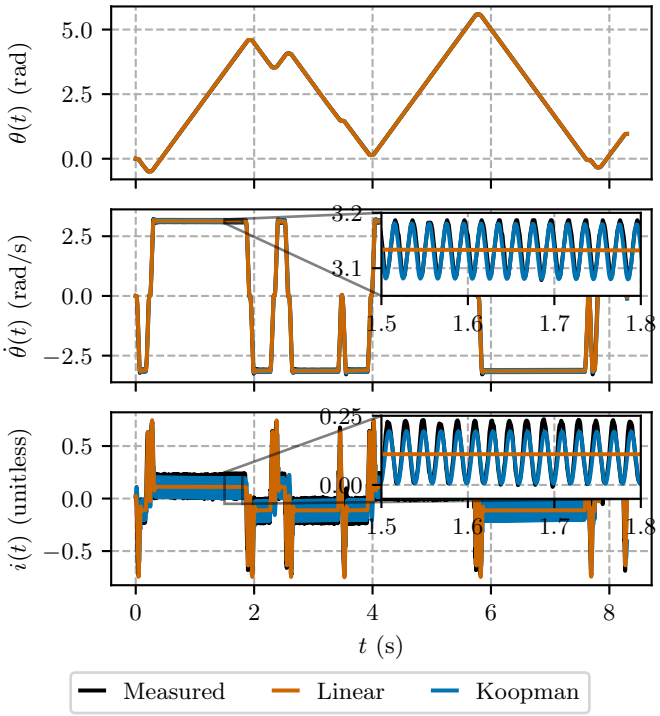
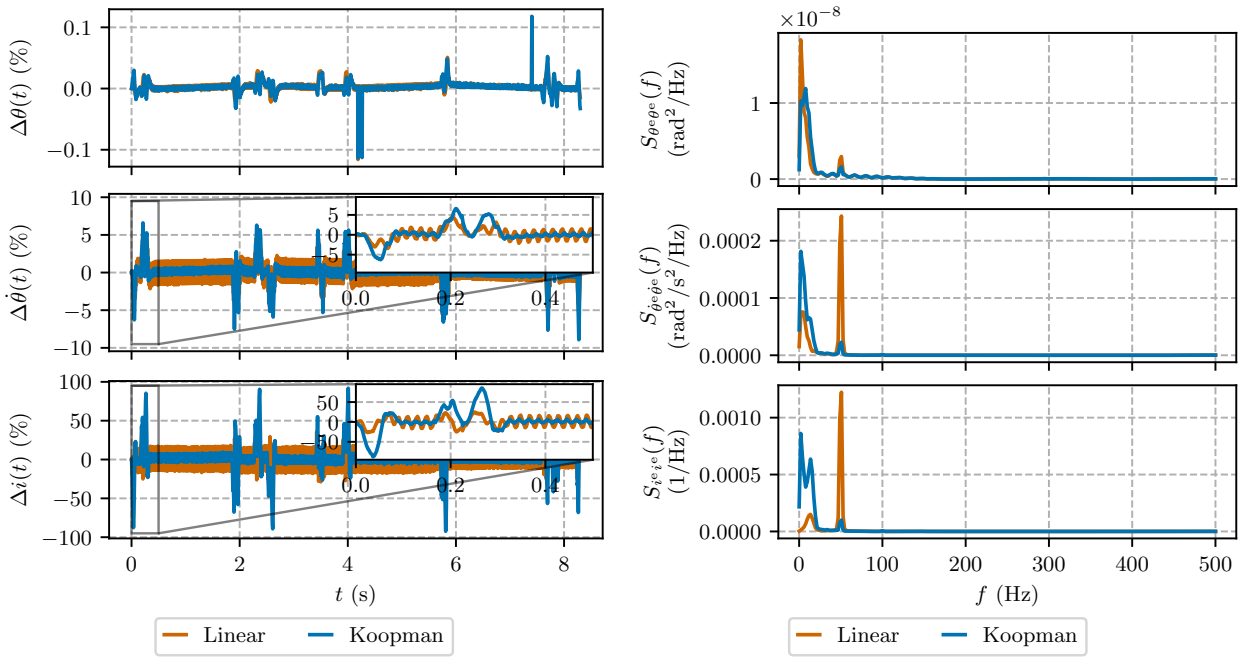


Figure 7.13: Predicted position, velocity, and current trajectories for linear and Koopman drive models using the first test episode. While both models capture the low-frequency behaviour of the drive, only the Koopman model is able to reproduce the Harmonic Drive vibrations.

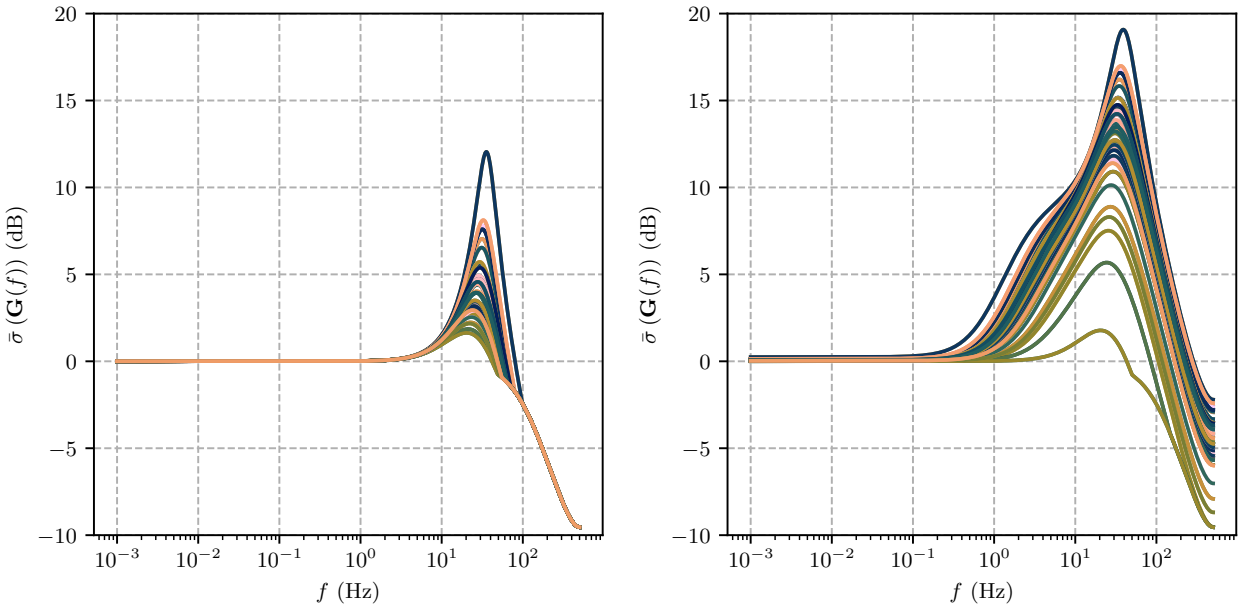
Figure 7.13. The corresponding prediction errors are shown in Figure 7.14. Figure 7.13 shows that the linear model correctly identifies the low-frequency dynamics of the motor drive, but does not predict the vibrations induced by the Harmonic Drive gearbox. In contrast, the Koopman model predicts the oscillations accurately. The position predictions of both models are nearly identical. Figure 7.14b shows the power spectral density of the prediction errors, which demonstrate that the Koopman model significantly reduces prediction errors at 50 Hz,



(a) Prediction error trajectories.

(b) Prediction error power spectral densities. A linear scale is used to emphasize the difference in error.

Figure 7.14: Predicted position, velocity, and current errors for linear and Koopman drive models using the first test episode. The Koopman model better predicts the gearbox oscillations, leading to lower velocity and current errors at 50 Hz. The position prediction errors are effectively identical.



(a) Linear models.

(b) Koopman models.

Figure 7.15: Frequency responses of the linear and Koopman models of the 38 motor drives under test. Uncertainty is highest in the 10 Hz–100 Hz decade, where the gearbox oscillation is located. The Koopman models vary more than the linear models throughout the spectrum.

while introducing more low-frequency current prediction error. While higher accuracy could be obtained by using a more complex Koopman model, the simplicity of the proposed model is attractive.

7.4.3 Uncertainty characterization

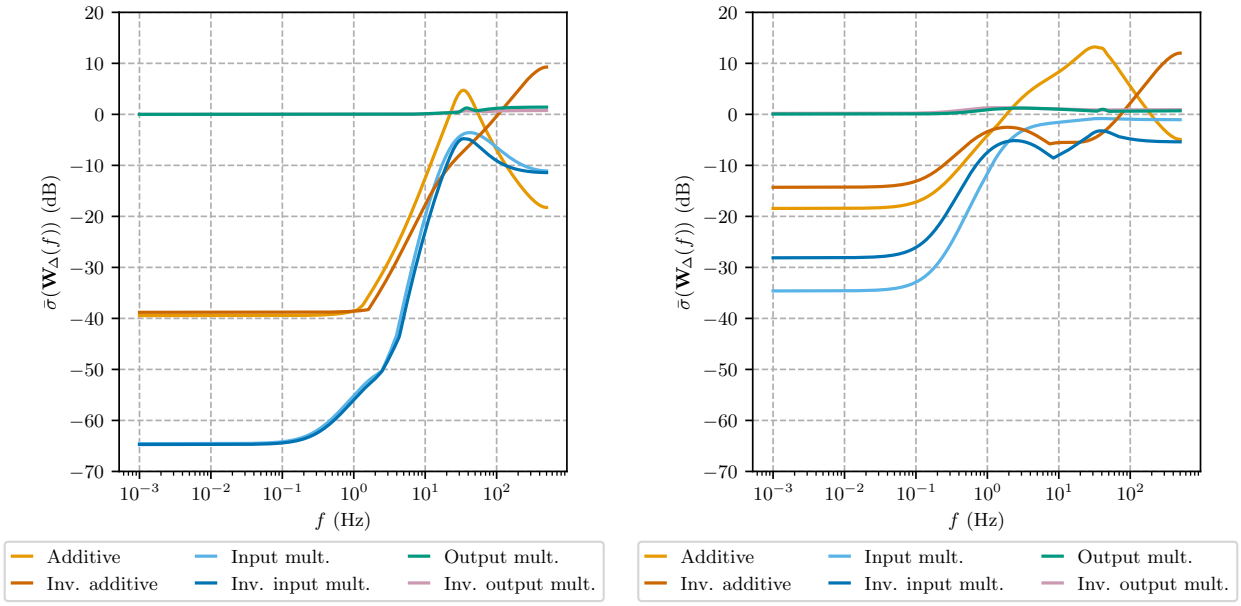
Now that a Koopman model for each motor drive has been identified, uncertainty is quantified within the population. Figure 7.15 shows the maximum singular values of the linear and Koopman transfer matrices identified for each drive. According to Figure 7.15, the frequency responses vary most in the 10 Hz–100 Hz decade, with the Koopman models varying significantly more than the linear models in that region. Figures A.1 and A.2, shown in Appendix A, also show the frequency responses of the linear and Koopman transfer functions but for each input-output pair.

To design a robust observer, an uncertainty form must be selected. To do so, residual transfer matrices are computed given a nominal model. Often, an averaged model is chosen as the nominal model. In this case, lower uncertainty is achieved by selecting one motor drive model from the population as the nominal model. To select this drive, residuals for each uncertainty form in Figure 7.5 are computed while treating each identified drive model as the nominal model. For each uncertainty form, the motor drive yielding the smallest peak residual is chosen as nominal. For the sake of comparison between the linear and Koopman approaches, the nominal model for a given drive is selected using the Koopman model's residuals.

Figure 7.16 shows the upper bounds on the maximum singular values of the residuals for each uncertainty form in Figure 7.5. Both output multiplicative uncertainty forms have high gain over the whole frequency spectrum, and are therefore unsuitable for controller or observer design. The input multiplicative uncertainty forms are more desirable, as they have low uncertainty at low frequency and their gains remain below 0 dB over all frequencies. Between the two, the inverse input multiplicative uncertainty form has the lowest gain at high frequencies, so it is selected for observer design. In Figure 7.16, each residual bound is computed using its best nominal model. Input-to-output residuals for each uncertainty form are shown in Appendix A.2.

The uncertainty bounds in Figure 7.16 are computed frequency-by-frequency. Once an uncertainty model is selected, transfer functions must be designed to bound them. This can be done manually or automatically with software assistance. Here, a nonlinear optimization problem is solved to find the transfer function coefficients that result in a magnitude response that closely bounds the residuals at each frequency.[1] Figure 7.17 shows the uncer-

[1]The optimization algorithm used to fit bounds to transfer function residuals was originally implemented



(a) Linear model.

(b) Koopman model.

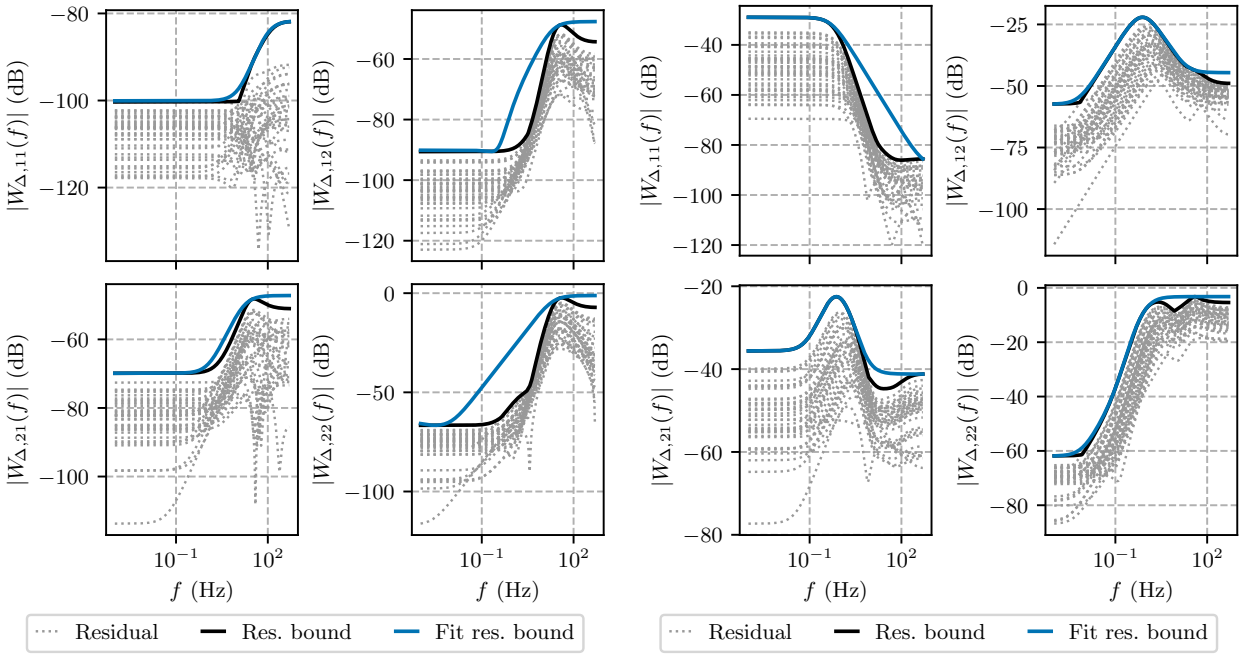
Figure 7.16: Upper bounds on the residuals for each uncertainty form. Both output multiplicative uncertainty forms have high uncertainty everywhere, while both input multiplicative forms have uncertainty below 0 dB everywhere. Inverse input multiplicative uncertainty is preferred, as it remains lower than its feedforward counterpart at higher frequencies.

tainty weights bounding the linear and Koopman residuals. Individual weighting functions are computed for each input-to-output transfer function. For the linear model, first- and second-order transfer functions are used. The Koopman models require third-order transfer functions, except for $W_{\Delta,11}(z)$, which uses a first-order transfer function. In both the linear and Koopman cases, $W_{\Delta,22}(z)$ represents the majority of the uncertainty in the transfer matrix.

7.4.4 Outlier detection

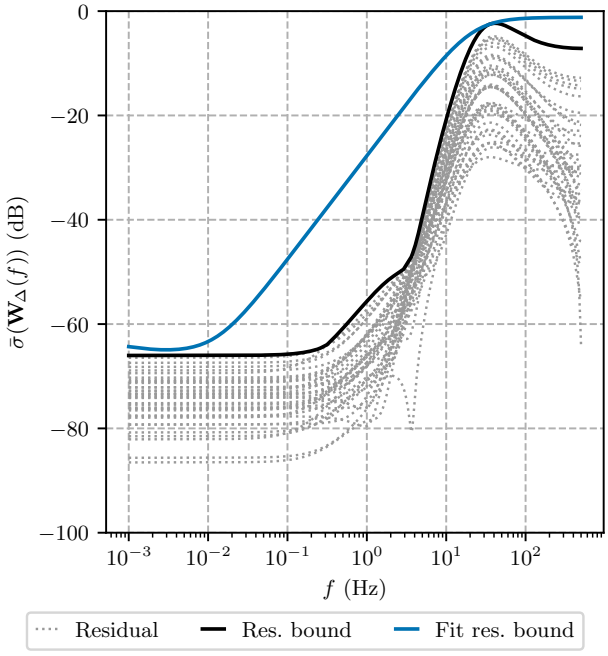
Before using the uncertainty models developed in Section 7.4.3 for robust observer design, an additional, simpler, application of frequency-domain uncertainty characterization is explored. The transfer matrix characterizing the uncertainty within a population can be used to identify drives that do not belong to the population, referred to in this section as outliers. In this example, outliers are motor drives that have been installed incorrectly. When a motor drive is installed in its frame with inappropriate fastener torques, tracking performance is degraded and oscillations induced by the Harmonic Drive gearbox are worsened. This effect is most apparent when the motor drive is loaded.

by Jonathan Eid with enhancements made by Prof. James Richard Forbes.

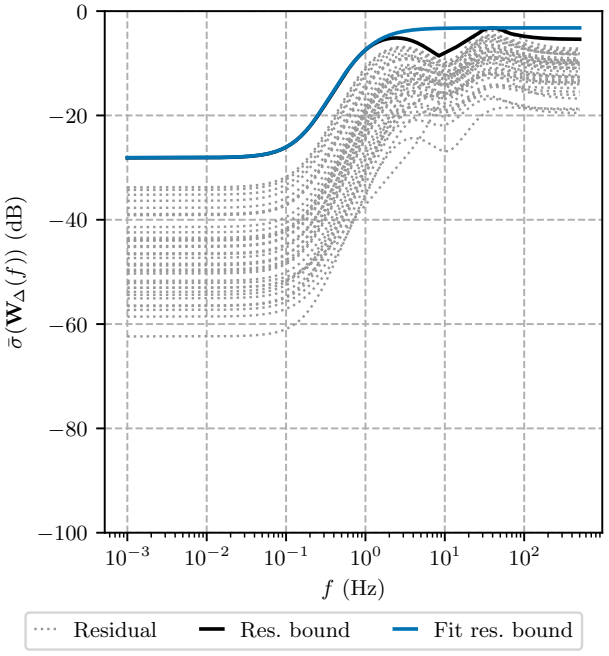


(a) Linear uncertainty bounds, input-to-output.

(b) Koopman uncertainty bounds, input-to-output.

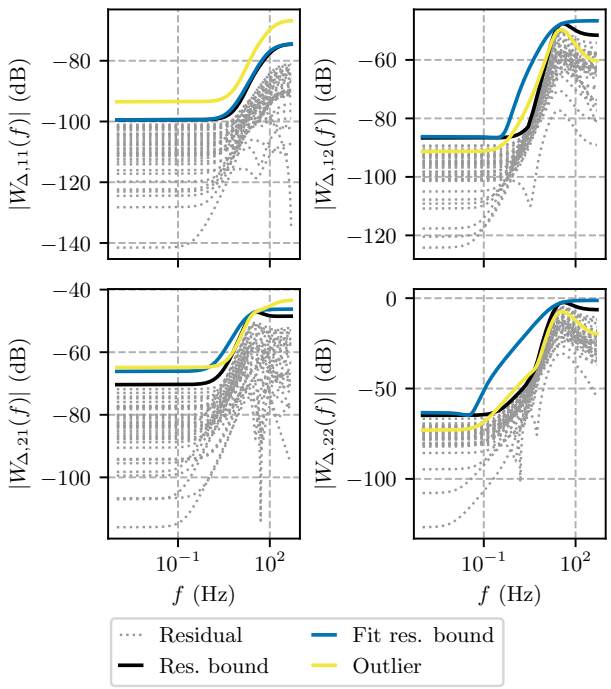


(c) Linear uncertainty bounds, maximum singular value of transfer matrix.

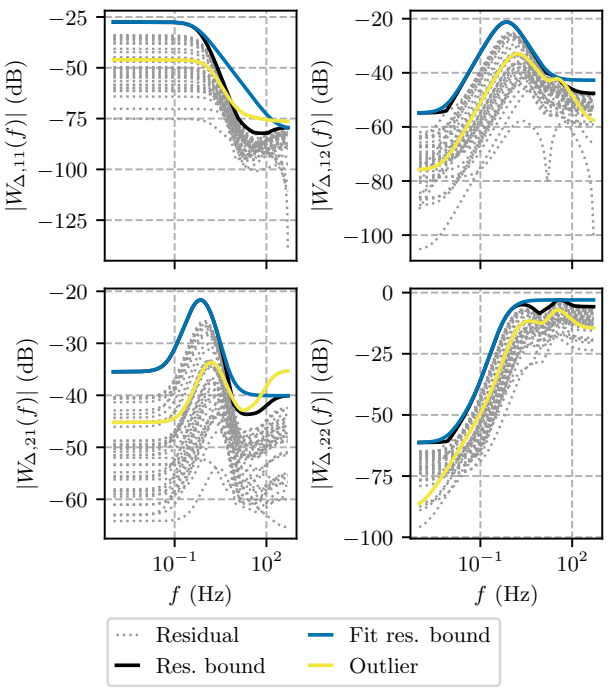


(d) Koopman uncertainty bounds, maximum singular value of transfer matrix.

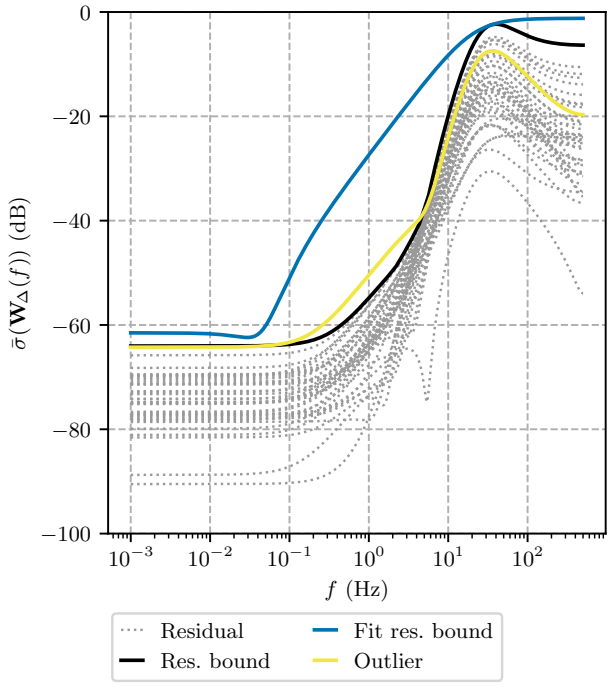
Figure 7.17: Inverse input multiplicative uncertainty bounds and fit transfer functions for linear and Koopman models.



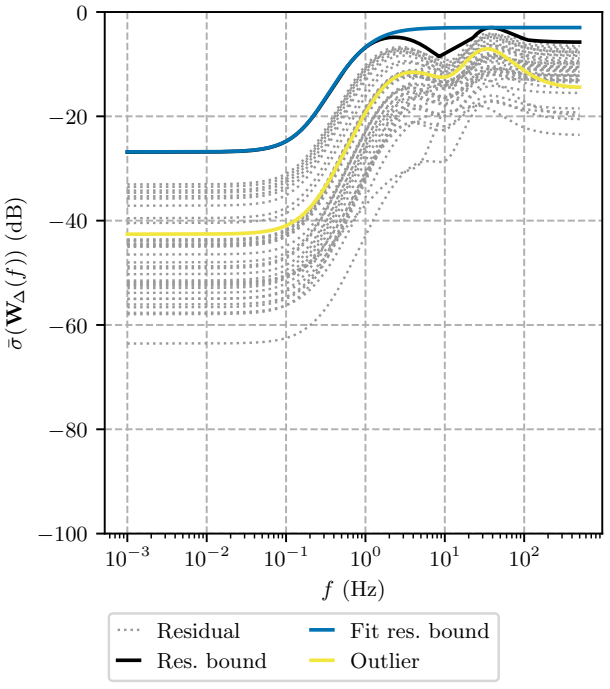
(a) Linear uncertainty bounds and outliers, input-to-output.



(b) Koopman uncertainty bounds and outliers, input-to-output.



(c) Linear uncertainty bounds and outliers, maximum singular value of transfer matrix.



(d) Koopman uncertainty bounds and outliers, maximum singular value of transfer matrix.

Figure 7.18: Inverse input multiplicative uncertainty bounds, fit transfer functions, and outlier residual for linear and Koopman models. The outlier is detectable from the input-to-output Bode plots, but not from the maximum singular value plots.

The uncertainty characterization procedure of Section 7.4.3 is now repeated for the loaded dataset, using the same optimization procedure to determine weighting functions. The inverse input multiplicative residuals corresponding to a motor drive that was deliberately installed with incorrect fastener torques are shown in Figure 7.18. The incorrectly installed motor drive can be identified by looking at $W_{\Delta,11}(z)$ in Figure 7.18a and $W_{\Delta,21}(z)$ in Figure 7.18b, where the outlier residuals exceed the fit residual bound. Note that, while the outliers are easy to identify from the input-to-output plots in Figures 7.18a and 7.18b, they are not detectable in the maximum singular value plots in Figures 7.18c and 7.18d. As such, outlier detection must be performed by looking at each residual transfer function individually, rather than looking only at the maximum singular value of the residual at each frequency.

While inverse input multiplicative uncertainty is shown in this section, the outlier drive is detectable using any uncertainty model, as shown in the residual plots in Appendix A.2. It has been verified that this outlier identification criterion is insensitive to the choice of nominal model. All choices of nominal model except one correctly identify the outlier model. It could be argued that that nominal model should also be considered an outlier. Ultimately, the decision of whether a system should be considered an inlier or outlier depends on the use case. The uncertainty mode could equally be extended to include the incorrectly installed drive if a more conservative uncertainty model is required.

7.4.5 Robust observer design

In this section, the robust observer design problem for a population of linear and Koopman models is solved. Since the motor drive has no true velocity sensor, motor velocity and current are observed given position measurements. Because the observer is synthesized using unloaded models, the motor current can be viewed as a proxy for the motor's electromagnetic torque. When the observer is fed measurements from a loaded drive, it predicts only the electromagnetic torque, not the load torque. Consequently, the current prediction error can be viewed as an estimate of the motor drive's load torque.

Recall that, unlike a controller, an observer cannot destabilize the observed system. As such, robustness in this chapter does not refer to robust stability. If the error dynamics of the synthesized observer are AS, no perturbation of the observed plant can change that property. Instead, as in [125, 136], robustness refers to insensitivity of performance to model uncertainty. In essence, the uncertainty model developed in Section 7.4.3 is used to inform the observer of when it can trust its internal nominal model.

7.4.5.1 Observer generalized plant

The generalized plant used for robust observer synthesis, depicted earlier in Figure 7.7, is inspired by the input-output observer originally proposed in [125, 136]. The controller component of the observer, $\mathcal{K}$ is synthesized by solving a mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ robust control problem. The $\mathcal{H}_2$ norm is a suitable performance metric for an observer due to its interpretation as the expected RMS output of a system subject to unit variance white noise input [38, §3.3.3][36, §5.7][42]. Recall that

$$\mathcal{CG} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{0} \end{array}\right], \tag{7.52}$$

and let

$$\mathcal{W}_{\mathrm{p}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{B}^{\mathcal{W}_{\mathrm{p}}} \\ \hline \mathbf{C}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{D}^{\mathcal{W}_{\mathrm{p}}} \end{array}\right], \quad \mathcal{W}_{\mathrm{u}} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{B}^{\mathcal{W}_{\mathrm{u}}} \\ \hline \mathbf{C}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{D}^{\mathcal{W}_{\mathrm{u}}} \end{array}\right], \quad \mathcal{W}_{\Delta} \overset{\min}{\sim} \left[\begin{array}{c|c} \mathbf{A}^{\mathcal{W}_{\Delta}} & \mathbf{B}^{\mathcal{W}_{\Delta}} \\ \hline \mathbf{C}^{\mathcal{W}_{\Delta}} & \mathbf{D}^{\mathcal{W}_{\Delta}} \end{array}\right], \tag{7.53}$$

where the corresponding states are denoted $\mathbf{x}_k^{\mathcal{W}_{\mathrm{p}}}$, $\mathbf{x}_k^{\mathcal{W}_{\mathrm{u}}}$, and $\mathbf{x}_k^{\mathcal{W}_{\Delta}}$. A state-space realization of the generalized plant is

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{x}}_{k+1} \\ \mathbf{x}_{k+1}^{\mathcal{W}_{\mathrm{p}}} \\ \mathbf{x}_{k+1}^{\mathcal{W}_{\mathrm{u}}} \\ \mathbf{x}_{k+1}^{\mathcal{W}_{\Delta}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{B}\mathbf{C}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{B}\mathbf{C}^{\mathcal{W}_{\Delta}} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & \mathbf{B}\mathbf{C}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{0} \\ \mathbf{B}^{\mathcal{W}_{\mathrm{p}}} & -\mathbf{B}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{A}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}^{\mathcal{W}_{\Delta}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{x}}_k \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{p}}} \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{u}}} \\ \mathbf{x}_k^{\mathcal{W}_{\Delta}} \end{bmatrix} + \begin{bmatrix} \mathbf{B}\mathbf{D}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{B}\mathbf{D}^{\mathcal{W}_{\Delta}} \\ \mathbf{B}\mathbf{D}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{B}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^{\mathcal{W}_{\Delta}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1,k} \\ \mathbf{w}_{2,k} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_k, \tag{7.54}$$

$$\begin{bmatrix} \mathbf{z}_{1,k} \\ \mathbf{z}_{2,k} \end{bmatrix} = \begin{bmatrix} \mathbf{D}^{\mathcal{W}_{\mathrm{p}}} & -\mathbf{D}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{C}^{\mathcal{W}_{\mathrm{p}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{C}^{\mathcal{W}_{\Delta}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{x}}_k \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{p}}} \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{u}}} \\ \mathbf{x}_k^{\mathcal{W}_{\Delta}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{D}^{\mathcal{W}_{\mathrm{u}}} & \mathbf{D}^{\mathcal{W}_{\Delta}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1,k} \\ \mathbf{w}_{2,k} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_k, \tag{7.55}$$

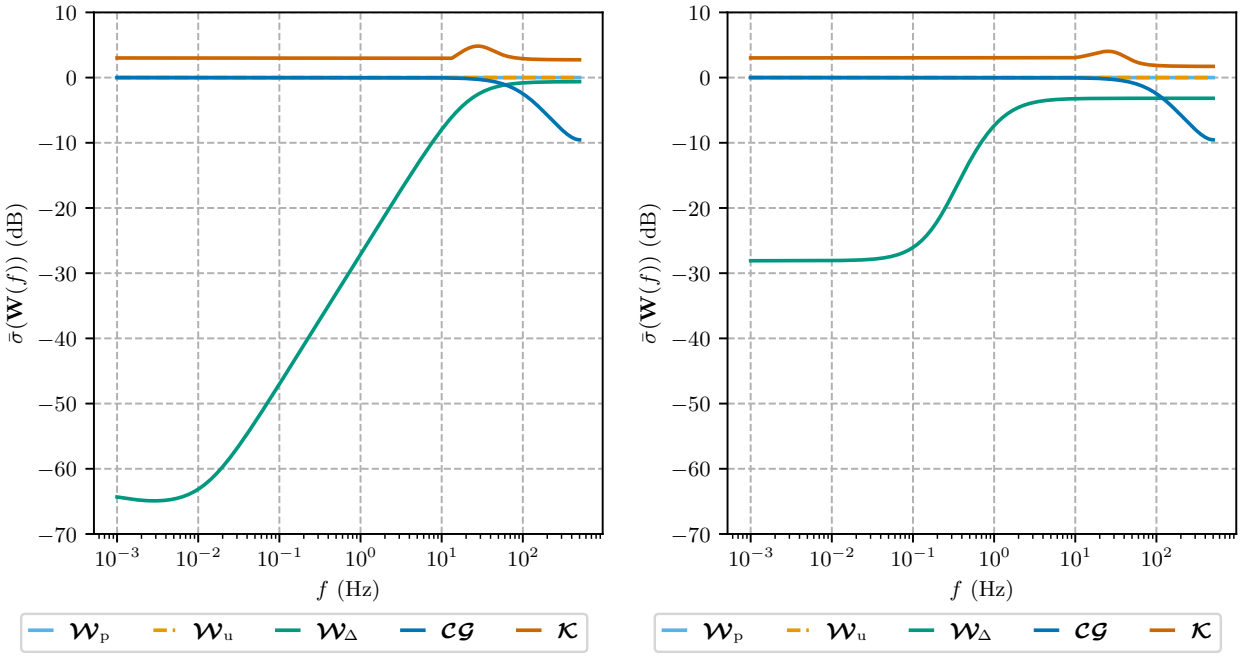
$$\mathbf{y}_k = \begin{bmatrix} \mathbf{C} & -\mathbf{C} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \hat{\mathbf{x}}_k \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{p}}} \\ \mathbf{x}_k^{\mathcal{W}_{\mathrm{u}}} \\ \mathbf{x}_k^{\mathcal{W}_{\Delta}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1,k} \\ \mathbf{w}_{2,k} \end{bmatrix} + \mathbf{0}\,\mathbf{u}_k. \tag{7.56}$$

Using these state-space matrices, an $\mathcal{H}_2$-$\mathcal{H}_\infty$ optimal controller is synthesized using the method outlined in Section 7.2.3. The final observer has the structure shown in Figure 7.6.

The observer synthesis procedure is identical for the linear and Koopman models, as they are both represented by state-space matrices. When using the Koopman model for prediction in the Koopman observer, the original system states are recovered and re-lifted with new exogenous inputs at each timestep, as is done in the rest of this thesis.

7.4.5.2 Observer weighting functions

Figure 7.19 shows the specific weighting functions used in the robust observer synthesis



(a) Linear observer weights.

(b) Koopman observer weights.

Figure 7.19: Performance, input, and uncertainty weights for the observer design problem, along with the plant and controller frequency responses. Note that the linear uncertainty weight is very close to 0 dB above 100 Hz. This leads to higher controller gains at high frequencies, leading to degraded performance compared to the Koopman observer.

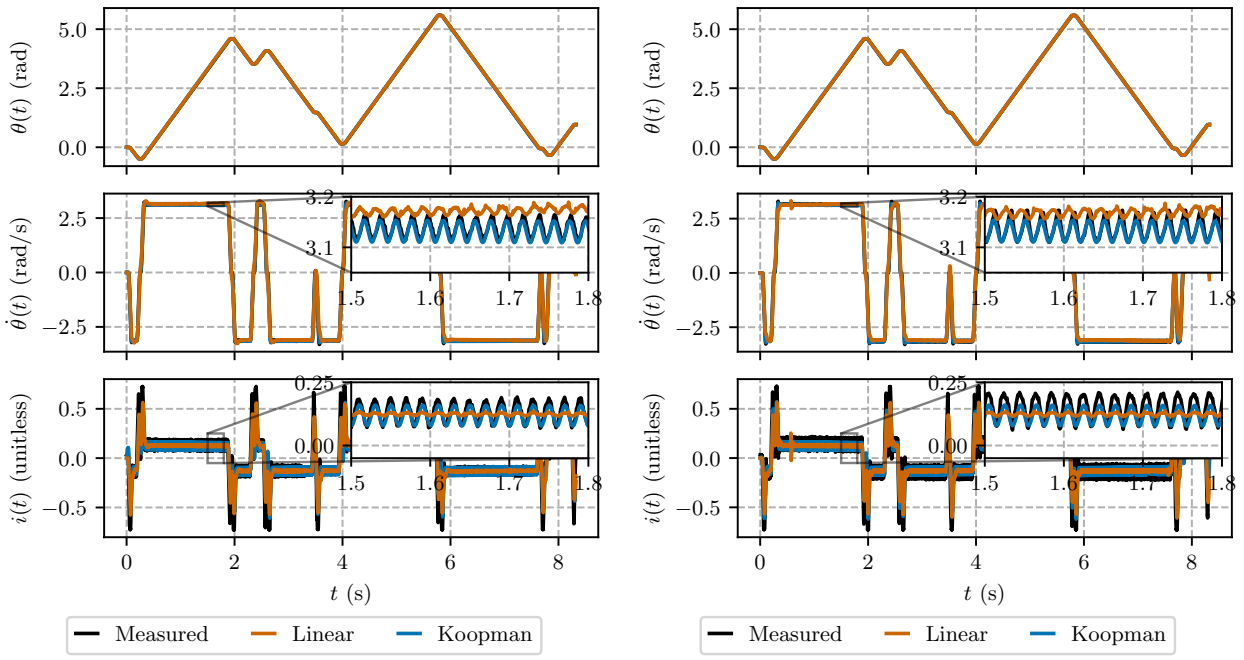
problem. Aside from $\mathbf{w}_2$ and $\mathbf{z}_2$, the input and output associated with the uncertainty block, the generalized plant has one performance input and one performance output: $\mathbf{w}_1$, which is associated with the system input and $\mathbf{z}_1$, which is associated with the state estimation error. For the sake of comparison, the input and performance weights are left as identity. In the Koopman observer, the performance weight for the lifted state estimation error is zero. Only the uncertainty weight differs between the linear and Koopman observer synthesis problems.

The intuition for how the observer weight affects the performance of the observer is

as follows. The controller component of the observer, $\boldsymbol{\mathcal{K}}$, governs the degree to which the observer's internal nominal plant model is corrected. Low uncertainty indicates that the true plant and nominal model match well, so the observer's internal nominal plant model should be trusted without correction. This corresponds to a low controller gain. High uncertainty indicates that the true plant and the nominal model may differ, so measurements should be used to correct the nominal plant model. This corresponds to a high controller gain. In Figure 7.19, the controller gain rises as the model uncertainty rises. As the plant gain rolls off, so does the controller gain. The linear model has higher uncertainty at high frequencies, so the controller gain remains higher than that of the Koopman model. In Section 7.4.6, this effect is shown to degrade state estimation performance.

7.4.6 Experimental results

The performance of the linear and Koopman robust observers is compared in this section.

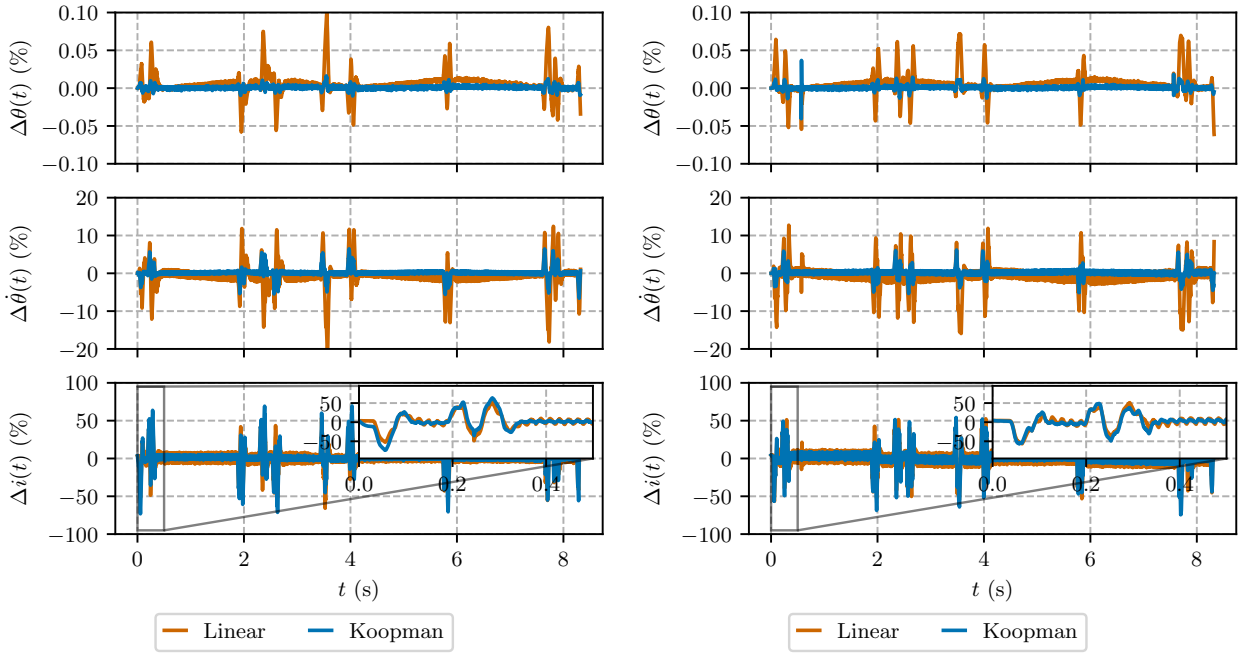


(a) Nominal motor drive.

(b) Off-nominal motor drive.

Figure 7.20: Estimated position, velocity, and current trajectories of the linear and Koopman observers for the nominal and the worst off-nominal systems. Unlike the linear observer, the Koopman observer correctly predicts the gearbox oscillations. The linear model's transient velocity estimates are degraded due to higher model uncertainty at high frequencies. Off-nominal predictions are slightly worse for all observers.

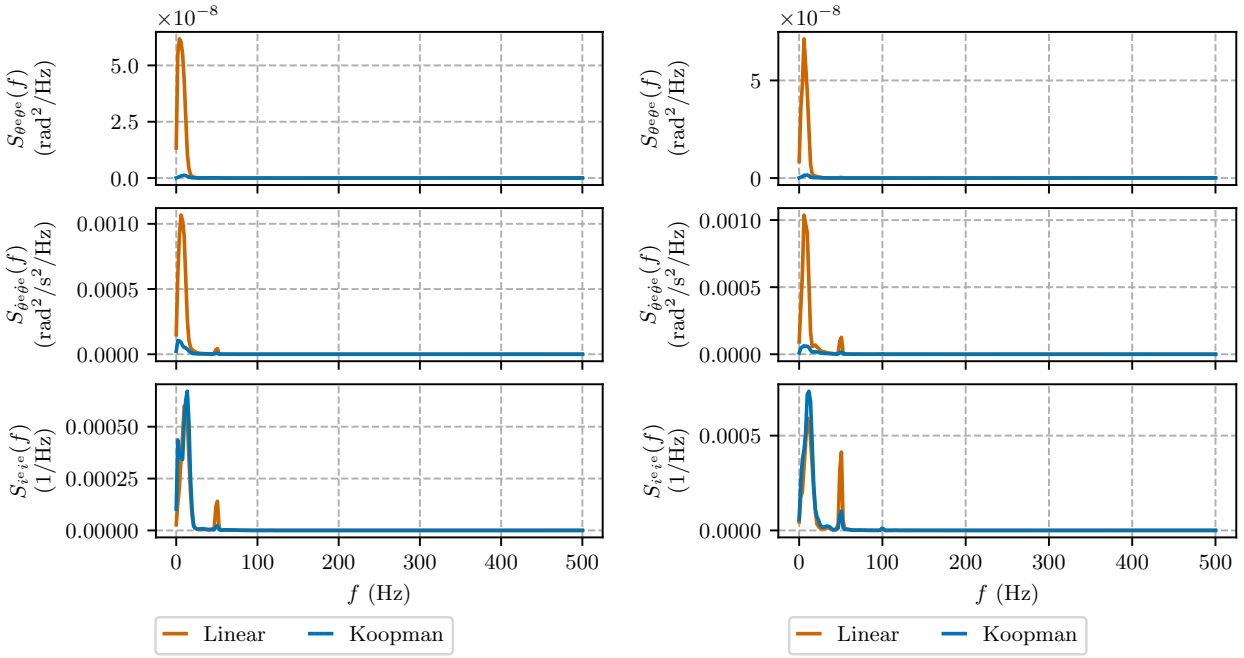
Three test conditions are considered. First, the observers are tested with measurements from their nominal motor drive. Then they are tested with measurements from the furthest



(a) Nominal motor drive.

(b) Off-nominal motor drive.

Figure 7.21: Position, velocity, and current estimation errors of the linear and Koopman observers for the nominal and worst off-nominal systems. The Koopman model captures the nonlinear oscillations and has superior transient performance. Estimation error is slightly worse in the off-nominal case.



(a) Nominal motor drive.

(b) Off-nominal motor drive.

Figure 7.22: Power spectral densities of position, velocity, and current estimation errors of the linear and Koopman observers for the nominal and worst off-nominal systems. In both cases, the Koopman observer reduces state estimation errors at 50 Hz. Note the linear scale.

off-nominal motor drive. These results are shown in Section 7.4.6.1. Finally, as shown in Section 7.4.6.2, the observers are tested with measurements from the loaded nominal motor drive.

7.4.6.1 Performance with nominal and off-nominal drives

The linear and Koopman observers synthesized in Section 7.4.5 are compared in this section for both the nominal drive and the worst off-nominal drive. Figure 7.20 shows the position, velocity, and current state estimates for the nominal and off-nominal drives. The Koopman observer is able to account for the gearbox oscillation while the linear model is not. Due to higher model uncertainty at high frequencies, the linear observer's transient velocity predictions are less accurate than those of the Koopman observer. There is a slight performance decrease going from the nominal to off-nominal plant, but the performance is still acceptable. Figures 7.21 and 7.22 show the corresponding state estimation errors in the time domain and the frequency domain. In both cases, the Koopman observer significantly reduces errors at 50 Hz.

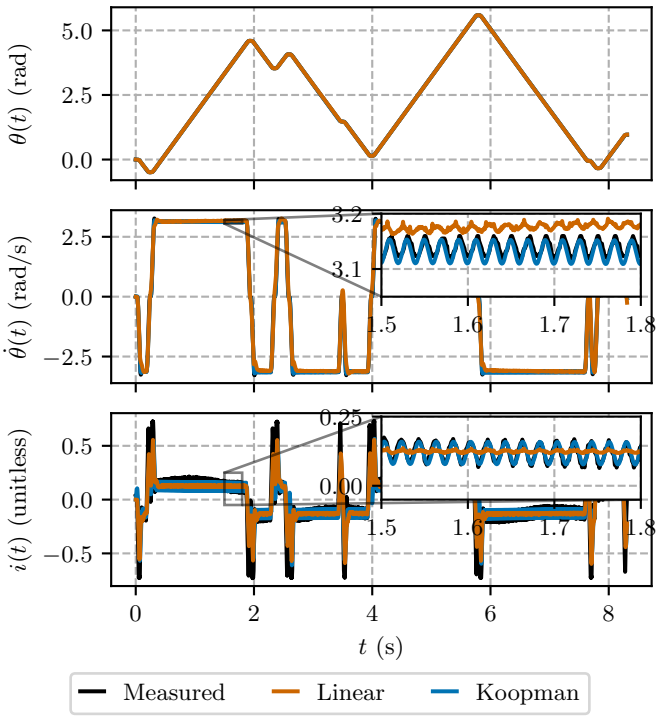
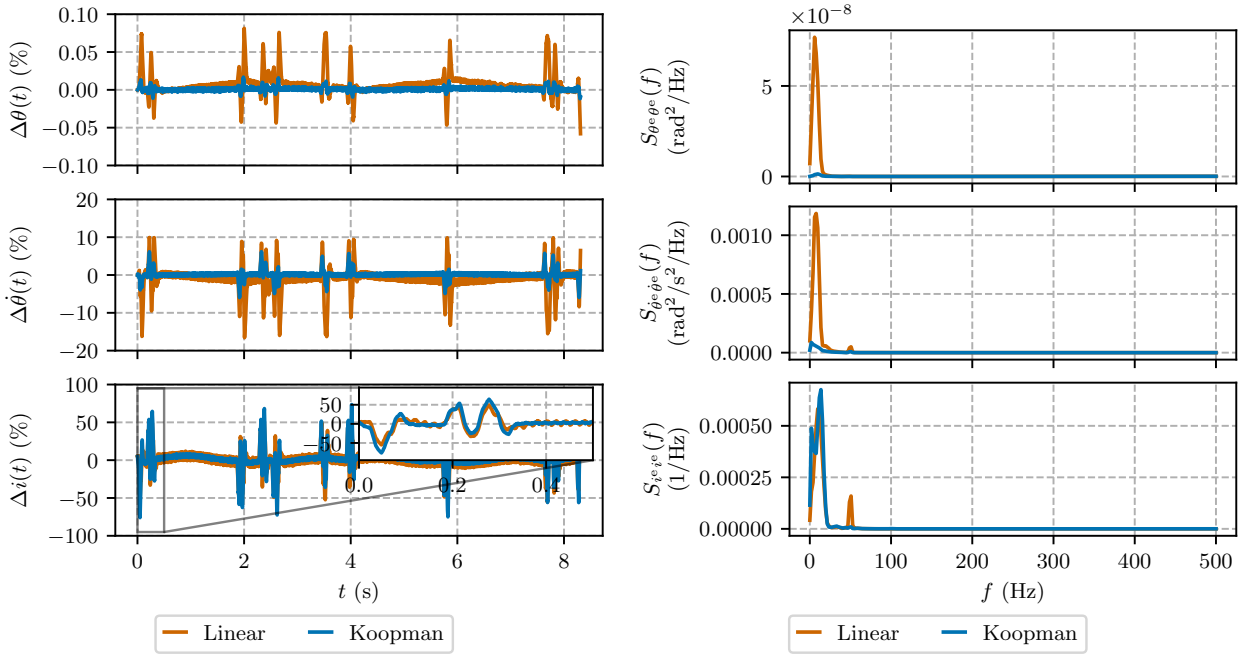


Figure 7.23: Estimated position, velocity, and current trajectories of the linear and Koopman observers for a nominal plant with an asymmetric inertial load. Neither the linear nor the Koopman model account for the sinusoidal torque disturbance introduced by the load.

7.4.6.2 Performance with loaded drive

Observer performance is now assessed for the nominal motor drive with an asymmetric inertial load attached. Due to the force of gravity, the load introduces a low-frequency sinusoidal disturbance to the system. The goal of the observer in this context is to estimate the low-frequency load disturbance while rejecting the high-frequency gearbox disturbance.

The linear and Koopman models, which were not identified using loaded data, do not predict this disturbance. As such, the observers do not include it in the state estimate. The state estimates are shown in Figure 7.23, while the state estimate errors are shown in Figure 7.24. The current tracking error in Figure 7.24a can be treated as an estimate of the external load on the motor drive, particularly when the drive is not accelerating strongly. The Koopman load estimate contains less 50 Hz disturbance than the linear load estimate.



(a) Prediction error trajectory.

(b) Prediction error power spectral density. A linear scale is used to emphasize the difference in error.

Figure 7.24: Position, velocity, and current prediction errors for linear and Koopman observers for the nominal plant with an asymmetric inertial load. Neither the linear nor the Koopman model account for the torque disturbance introduced by the load. The current estimation error Δi can be treated as an estimate of the load torque.

7.5 Conclusion

This chapter explores how the linearity of the Koopman operator can be exploited to quantify model uncertainty in the frequency domain and synthesize robust nonlinear observers using that uncertainty. A detailed experimental example is presented, wherein Koopman models for a population of 38 motor drives are identified and uncertainty within the population is quantified using standard robust control tools. The frequency-domain uncertainty characterization is shown to be sufficiently accurate to identify incorrectly installed motor drives. Using this uncertainty model, a mixed $\mathcal{H}_2$-$\mathcal{H}_\infty$ robust observer is designed to estimate motor velocity and current from position measurements. While much contemporary Koopman control literature focuses on simulated systems, the proposed observer design approach is demonstrated with real data. Furthermore, the state-of-the-art in robust Koopman control considers only very simple uncertainty models compared to those presented here.

As with many Koopman control approaches, one limiting factor in this approach is the inability to lift the control inputs in the Koopman model. Bilinear lifting approaches like [67] show great potential for MPC algorithms, but may be difficult to integrate with LTI controllers. The proposed observer design approach can equally be used to synthesize robust optimal controllers, which is a topic that will be explored in future work.

Chapter 8

pykoop: a Python Library for Koopman Operator Approximation

Summary

This chapter describes the software architecture of `pykoop`, an open-source Koopman operator approximation library written in Python. The contributions of `pykoop` relative to other Koopman operator approximation libraries are its support of control inputs and multiple training episodes, its ability to arbitrarily compose lifting functions, and its compatibility with standard cross-validation and hyperparameter optimization tools. The methods presented in Chapters 4–7 are all implemented using `pykoop`.

8.1 Introduction

Designing Koopman lifting functions for a particular system is largely a trial-and-error process [141]. Rather than addressing theoretical aspects of lifting function design, this chapter presents `pykoop` [6], a Python library that aims to facilitate experimentation with Koopman lifting functions and regressors. The `pykoop` library addresses three limitations in current Koopman operator approximation software packages. First, `pykoop` allows lifting functions to be constructed in a modular fashion, specifically through the composition of a series of lifting functions. Second, the library allows regressor hyperparameters and lifting functions to be selected automatically through full compatibility with `scikit-learn`'s [142] existing cross-validation infrastructure. Third, `pykoop` handles datasets with control inputs and multiple training episodes at all stages of the pipeline.

8.1.1 Related work

Open-source Python libraries for Koopman operator approximation include `PyKoopman` [143], `DLKoopman` [144], and `kooplearn` [145]. While `PyKoopman` provides a similar selection of lifting functions to `pykoop`, it does not allow lifting functions to be composed. Furthermore, the library does not include built-in functionality to handle multiple training episodes. Unlike `pykoop`, `PyKoopman` is able to identify continuous-time Koopman operators. As such, the library includes several built-in numerical differentiation methods. Neural network lifting functions are also supported by `PyKoopman`. The `DLKoopman` library focuses on learning Koopman models using only neural network lifting functions, which are not implemented in `pykoop`. The library supports multiple training episodes but does not support exogenous inputs and does not follow the standard `scikit-learn` interface. The `kooplearn` library includes kernel and neural network approaches to learning Koopman models, but does not include other types of lifting functions. While multiple training episodes are handled by `kooplearn`, exogenous inputs are not.

8.1.2 Contribution

The `pykoop` library tracks training episodes and input-dependent lifting functions throughout the entire pipeline. This allows lifting functions to be composed arbitrarily while guaranteeing that the lifting functions can always be partitioned into input-independent and input-dependent groups. The library's strict adherence to `scikit-learn`'s interface allows lifting functions to be parameterized and tuned using the cross-validation and hyperparameter optimization tools built in to `scikit-learn` and other compatible libraries. These key contributions, which are not fully supported in [143–145], make `pykoop` a flexible and user-friendly library for designing Koopman lifting functions, especially when exogenous inputs are required.

8.2 Software Architecture

The contributions outlined in Section 8.1.2 are direct results of specific software architecture choices in `pykoop`, which are described in this section. The aim of this section is to explain the high-level organization of `pykoop` rather than its implementation details. Up-to-date documentation for `pykoop` can be found at `https://pykoop.readthedocs.io/en/stable/`. The examples found in this chapter use `pykoop` v1.2.3 and Python 3.12.4.

Unified modeling language (UML) class diagrams [146] are used extensively in this section. A brief review of the notation follows [146, §B]. Each box represents a class. Inside each

class, attributes are shown first, followed by methods. Public attributes and methods are denoted with +, while private ones are denoted with −. Concrete methods are shown in roman type and abstract methods are shown in *italic* type. Inheritance is denoted by a solid arrow pointing from the child class to the parent class. Implementation is denoted by a dashed arrow. An arrow with a closed diamond denotes composition, while an arrow with an open diamond denotes aggregation. In both cases, the multiplicity and relevant attribute name are written next to the arrow.

8.2.1 Data format

In the `scikit-learn` framework, an *estimator* is a general term for an object that fits a model from training data [142]. Koopman lifting functions and regressors can be viewed as `scikit-learn` estimators. Estimators adhering to the `scikit-learn` interface typically accept two-dimensional numeric NumPy [147] arrays as inputs. Contrary to the notation in the rest of this thesis, an array compliant with `scikit-learn` has the number of samples as its first dimension and the number of features as its second dimension. To correctly apply lifting functions to training data, a `pykoop` estimator must know which samples belong to which episodes. Furthermore, the estimator must know which features are states and which are inputs.

To specify training episode information, the first feature of a `pykoop`-compatible NumPy array may be the *episode feature*, which contains an integer index that indicates the episode to which a sample belongs. Exogenous input signals are specified as the last features in the data array. The presence of this optional episode feature and the dimension of the exogenous input signal are specified when fitting an estimator. To fit an estimator, the `scikit-learn` interface specifies that the estimator must have the method `fit()`, which accepts training data but may also have additional optional keyword arguments. All fit methods in `pykoop` have the signature `fit(X, n_inputs, episode_feature)`, where `n_inputs` is an integer indicating the dimension of the exogenous input and `episode_feature` is a Boolean variable indicating the presence of the episode feature.

Consider an example where `episode_feature=True` and `n_inputs=1`. The data matrix

$$\mathbf{X} = \begin{array}{c} \begin{matrix} \overset{\mathbf{e}}{\leftrightarrow} & \overset{\mathbf{x}}{\longleftrightarrow} & & \overset{\mathbf{u}}{\longleftrightarrow} \end{matrix} \\ \begin{bmatrix} 0 & 0.1 & -0.1 & 0.2 \\ 0 & 0.2 & -0.2 & 0.3 \\ 0 & 0.3 & -0.3 & 0.4 \\ 1 & -0.1 & 0.1 & 0.3 \\ 1 & -0.2 & 0.2 & 0.4 \\ 1 & -0.3 & 0.3 & 0.5 \\ 2 & 0.3 & -0.1 & 0.3 \\ 2 & 0.2 & -0.2 & 0.4 \end{bmatrix} \end{array} \begin{array}{l} \uparrow \\ \downarrow \end{array} \text{episode } 0 \quad \begin{array}{l} \uparrow \\ \downarrow \end{array} \text{episode } 1 \quad \begin{array}{l} \uparrow \\ \downarrow \end{array} \text{episode } 2 \tag{8.1}$$

is then interpreted by `pykoop` as having three episodes with a two-dimensional state and one-dimensional input.

The `pykoop` library uses this information to keep the original system state at the beginning of the lifted state, keep the input-dependent lifted states at the end of the lifted state, and prevent reordering or sub-sampling that would mix data from different training episodes.

8.2.2 The Koopman pipeline

The `KoopmanPipeline` class, shown in Figure 8.1, is central to `pykoop`'s Koopman operator approximation workflow. A `KoopmanPipeline` object contains a single regression method,

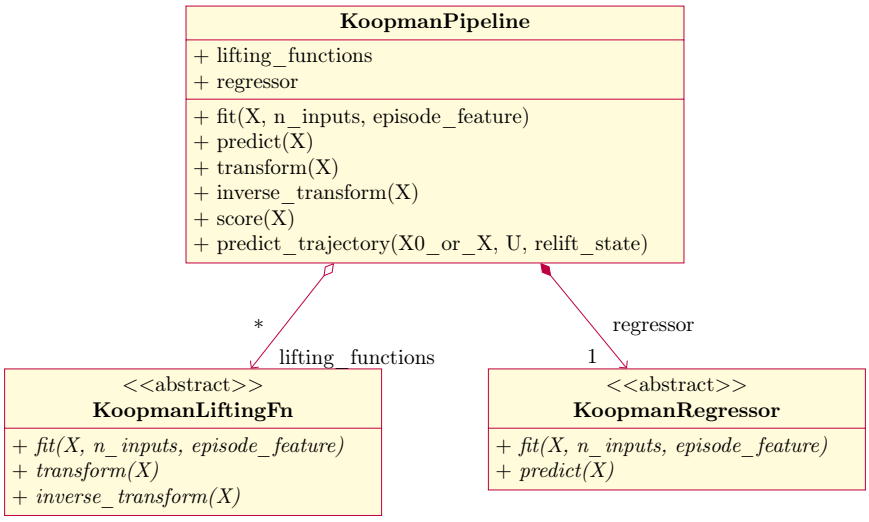


Figure 8.1: Abridged UML class diagram of the `KoopmanPipeline` class, which aggregates any number of `KoopmanLiftingFn` objects and contains exactly one `KoopmanRegressor` object.

represented by a `KoopmanRegressor`, and any number of composed lifting functions, represented by a sequence of `KoopmanLiftingFn` objects. When fit, a `KoopmanPipeline` object lifts the training data. This is done by calling the `fit()` and `transform()` methods of each `KoopmanLiftingFn` object in order. Each lifting function is responsible for ensuring that the input-independent lifting functions are ordered before input-dependent ones. The `KoopmanRegressor` object is then fit. Note that, in keeping with `scikit-learn`, each lifting function in a `KoopmanPipeline` object must be given a name. As such, the `lifting_functions` attribute is a list of tuples, where each tuple contains a user-defined name for the lifting function and the corresponding `KoopmanLiftingFn` object. The Koopman matrix can be extracted from the fit `KoopmanRegressor` object. The `transform()` and `inverse_transform()` methods of `KoopmanPipeline` simply lift and retract data.

The `KoopmanPipeline` object has two prediction methods. The first, `predict()`, advances each data point provided by one timestep. The second, `predict_trajectory()`, accepts an initial condition and input sequence, and performs a running prediction of an entire trajectory. Its `relift_state` parameter determines whether the system's original state is recovered and re-lifted between timesteps. These options correspond to the local and global error definitions of [68] respectively, as discussed in Section 3.6. The `score()` method uses running predictions to evaluate the estimator, rather than one-step-ahead predictions.

A set of convenient plotting methods are also provided by `KoopmanPipeline`. These methods can be used to plot the Koopman system's frequency response, the maximum singular values and eigenvalues of the Koopman matrix, the Koopman matrix itself, and the model's predictions and prediction errors.

A Python example using the `KoopmanPipeline` object can be found in Listing 8.1. In the example, time-delayed polynomials are used as lifting functions to identify a mass-spring-damper system using EDMD with Tikhonov regularization. Once fit, the `KoopmanPipeline` object can be used to predict and plot new trajectories or score the estimator. Note that, if the order of the polynomial and delay lifting functions was swapped, the resulting lifting functions would differ. In this case, the polynomial combinations of states are computed first, and then delayed, so there are no polynomial combinations of delayed states. If the polynomial lifting functions were placed after, then lifted states of the form $\psi_i(\mathbf{x}_k) = x_{1,k}x_{1,k+1}$ would be present.

8.2.3 Koopman regressors

The `KoopmanRegressor` class, shown in Figure 8.2, is used to represent Koopman regression algorithms in `pykoop`. In keeping with the `scikit-learn` interface, the class inherits from `sklearn.base.BaseEstimator` and `sklearn.base.RegressorMixin`. Each implementa-

```python
import pykoop

# Get example mass-spring-damper data
eg = pykoop.example_data_msd()

# Create pipeline
kp = pykoop.KoopmanPipeline(
    lifting_functions=[
        ('polynomial', pykoop.PolynomialLiftingFn(order=2)),
        ('delay', pykoop.DelayLiftingFn(n_delays_state=1, n_delays_input=1)),
    ],
    regressor=pykoop.Edmd(alpha=0.1),
)

# Fit the pipeline
kp.fit(
    eg['X_train'],
    n_inputs=eg['n_inputs'],
    episode_feature=eg['episode_feature'],
)

# Predict using the pipeline
X_pred = kp.predict_trajectory(eg['x0_valid'], eg['u_valid'])

# Score using the pipeline
score = kp.score(eg['X_valid'])

# Plot results using the pipeline
kp.plot_predicted_trajectory(eg['X_valid'], plot_input=True)
```

Listing 8.1: Example usage of the `KoopmanPipeline` class. Lifting functions are time-delayed polynomials.

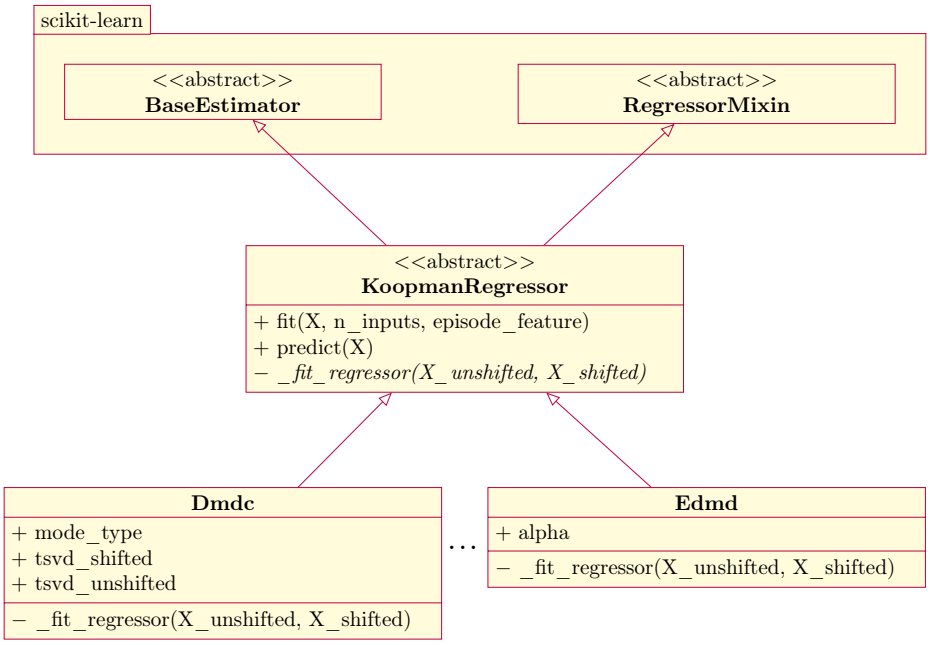


Figure 8.2: Abridged UML class diagram of the `KoopmanRegressor` abstract class, which inherits from standard `scikit-learn` abstract classes. All Koopman regression methods in `pykoop` are implementations of this class.

tion of `KoopmanRegressor` overrides a private method, `_fit_regressor()`, which takes the unshifted and shifted lifted data matrices as inputs with the episode feature removed. These preprocessing operations are done inside `fit()`, and thus do not need to be re-implemented inside each child class.

The `KoopmanRegressor` classes implemented in `pykoop` are

- `Dmd`, which implements DMD (see Section 3.4.4),
- `Dmdc`, which implements DMDc (see Section 3.4.5),
- `Edmd`, which implements EDMD (see Section 3.4.3 and Listing 8.1),
- `EdmdMeta`, which implements EDMD by wrapping `scikit-learn` regressors, and
- `DataRegressor`, which allows a known Koopman matrix to be hard-coded.

LMI-based `KoopmanRegressor` classes implementing the methods from Chapters 4 and 5 are implemented in `pykoop.lmi_regressors`. Selected `KoopmanRegressor` classes are now described in detail in their own sections.

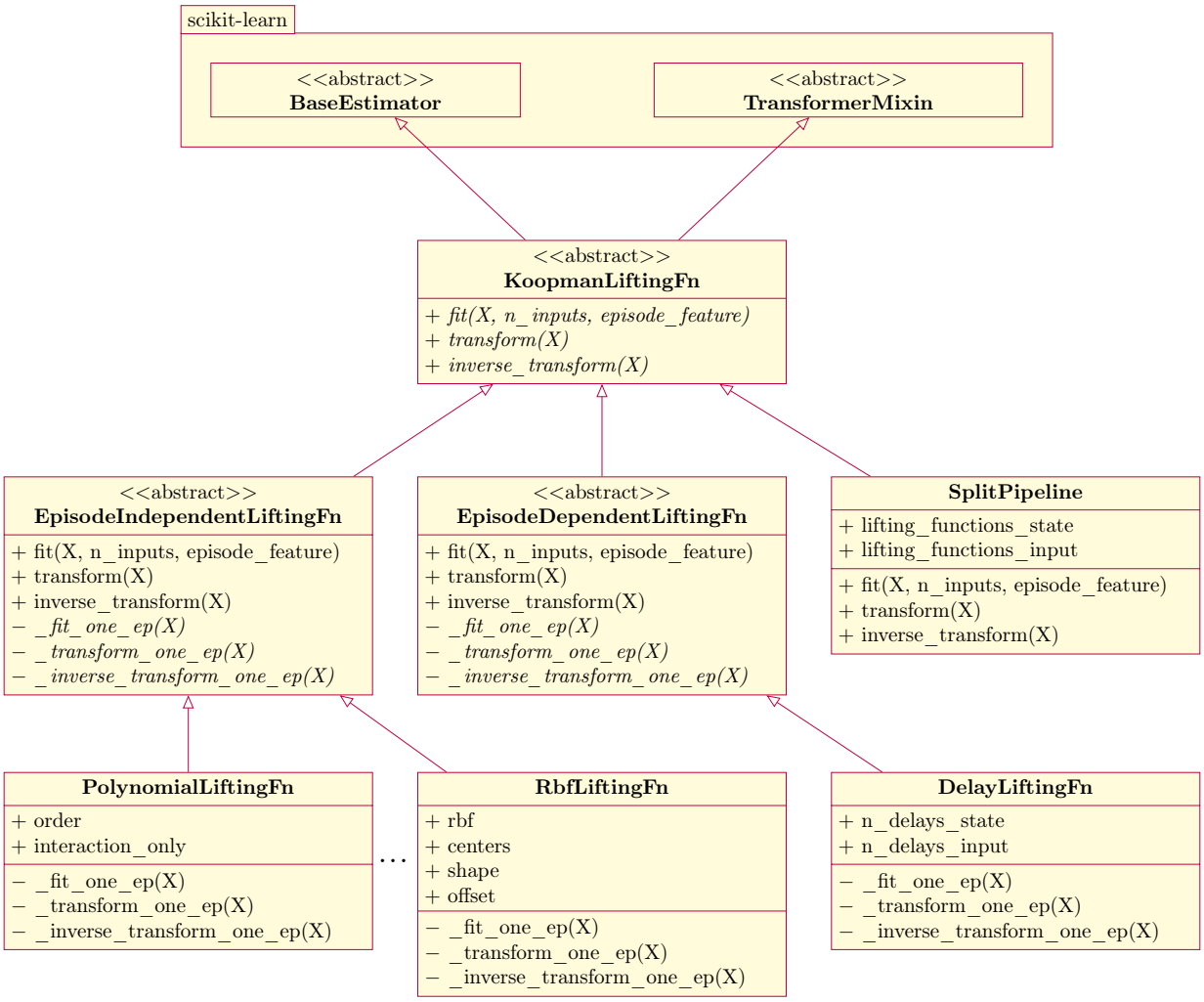


Figure 8.3: Abridged UML class diagram describing the `KoopmanLiftingFn` abstract class, which inherits from standard `scikit-learn` abstract classes. All Koopman lifting functions in `pykoop` are implementations of this class. The `EpisodeIndependentLiftingFn` and `EpisodeDependentLiftingFn` are used to simplify the implementations of common types of lifting functions.

8.2.3.1 `Edmd`

The `Edmd` class implements EDMD with Tikhonov regularization. Its regularization coefficient, `alpha`, is specified in the constructor. Example usage of this class can be found in Listing 8.1.

8.2.3.2 `Dmd` and `Dmdc`

The `Dmd` and `Dmdc` classes implement DMD and DMDc respectively. DMD requires one

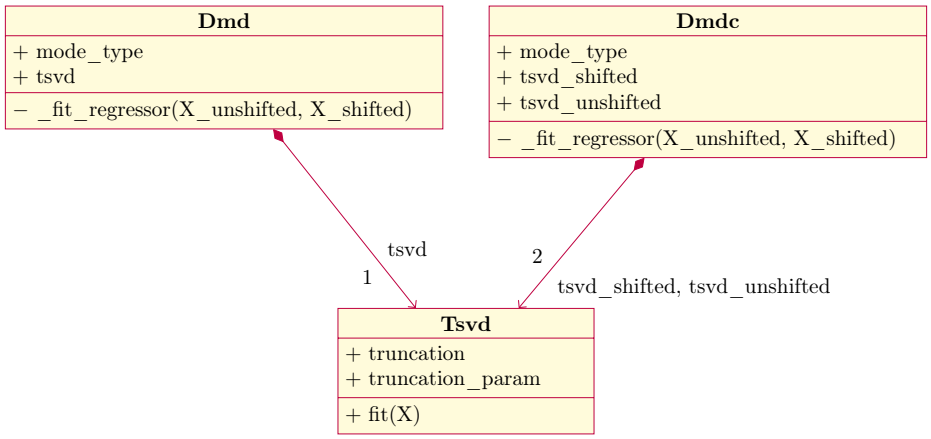


Figure 8.4: Abridged UML class diagram of the `Dmd` and `Dmdc` classes, which use the `Tsvd` class to implement the truncated SVD.

truncated SVD while DMDc requires two. As shown in Figure 8.4, these classes make use of the `Tsvd` class to compute SVDs with different singular value truncation criteria, such as those discussed in [34]. The `Tsvd` class inherits from `sklearn.base.BaseEstimator`. Thanks to the `scikit-learn` interface, its truncation parameters can be explored and optimized using compatible cross-validation and hyperparameter optimization software.

8.2.4 Koopman lifting functions

Koopman lifting functions are implemented using the `KoopmanLiftingFn` class, shown in Figure 8.3. The `KoopmanLiftingFn` class inherits from both `sklearn.base.BaseEstimator` and `sklearn.base.TransformerMixin`. Each child class of `KoopmanLiftingFn` implements `fit()`, `transform()`, and `inverse_transform()`. Beyond bookkeeping, the functionality of `fit()` varies for each lifting function. The `transform()` and `inverse_transform()` methods lift and retract data matrices.

Some classes, like `SplitPipeline`, inherit directly from `KoopmanLiftingFn`. However, in most cases, it is more convenient to inherit from `EpisodeIndependentLiftingFn` or `EpisodeDependentLiftingFn`, as they contain the majority of the shared code

for common types of lifting functions. When inheriting from one of these two abstract classes, the child class need only override `_fit_one_ep()`, `_transform_one_ep()`, and `_inverse_transform_one_ep()`, which consider only a single episode. Any lifting function that can be applied on a sample-by-sample basis inherits from `EpisodeIndependentLiftingFn`. All lifting functions except time delays fall into this category. For these types of lifting functions, the single-episode methods can be applied to the entire data matrix, regardless of episode. Lifting functions that require episode information, like time delays, inherit from `EpisodeDependentLiftingFn`. In these cases, the data matrix must be divided by episode. Episode-dependent lifting functions are applied to each sub-matrix and the resulting lifted sub-matrices are re-combined.

The `KoopmanLiftingFn` classes implemented in `pykoop` are

- `PolynomialLiftingFn`, which implements polynomial lifting functions (see Section 3.5.1 and Listing 8.1),
- `DelayLiftingFn`, which implements time delay lifting functions (see Section 3.5.2 and Listing 8.1),
- `RbfLiftingFn`, which implements RBF lifting functions (see Section 3.5.3),
- `KernelApproxLiftingFn`, which implements RFF lifting functions (see Section 3.5.4),
- `BilinearInputLiftingFn`, which implements bilinear input lifting functions (see Section 3.5.5),
- `ConstantLiftingFn`, which adds a constant term to the lifting functions,
- `SplitPipeline`, which allows different lifting functions to be applied to states and inputs, and
- `SkLearnLiftingFn`, which wraps `scikit-learn` transformer objects into lifting functions.

Selected `KoopmanLiftingFn` classes are now described in detail in their own sections.

8.2.4.1 `PolynomialLiftingFn`

The `PolynomialLiftingFn` class generates monomial lifting functions of a specified order. Its constructor has two parameters, `order`, which determines the maximum monomial order, and `interaction_only`, which removes lifting functions containing only one variable like $\psi_i(\mathbf{x}_k) = x_{1_k}^3$. Example usage of this class can be found in Listing 8.1.

8.2.4.2 `DelayLiftingFn`

The `DelayLiftingFn` class generates time delay lifting functions. Its constructor has two parameters, `n_delays_state` and `n_delays_input`, which control the number of time delays used for the state and input respectively. Note that, if the number of state and input delays differ, `transform()` and `inverse_transform()` will not be exact inverses since some samples from the data matrix are dropped. Example usage of this class can be found in Listing 8.1.

8.2.4.3 `RbfLiftingFn`

The `RbfLiftingFn` class, shown in Figure 8.5, generates RBF lifting functions. The choice

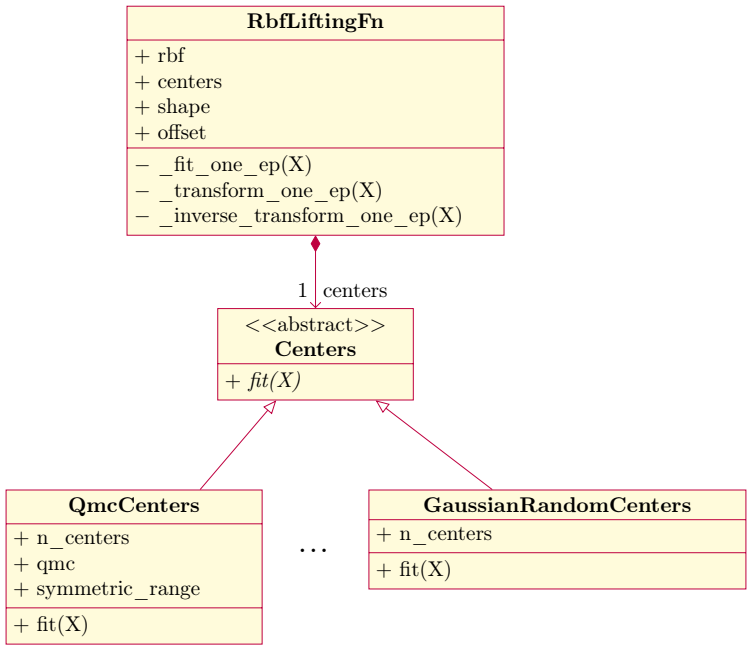


Figure 8.5: Abridged UML class diagram of the `RbfLiftingFn` class, which implements RBF lifting functions. The `RbfLiftingFn` class is composed of one `Centers` object, which controls where the `RBF` centres are placed.

of RBF is specified through the `rbf` parameter, the shape is specified through the `shape` parameter, and, if needed, the offset is specified through the `offset` parameter. The choice of algorithm to generate RBF centres is made through the `centers` parameter, which contains one object implementing the `Centers` abstract class. The `Centers` abstract class inherits from `sklearn.base.BaseEstimator`, so the parameters of any class inheriting from `Centers`, including the number of centres, can be tuned using standard hyperparameter optimization tools.

Some centre generation classes provided by `pykoop` are

- `ClusterCenters`, which chooses centres according to a clustering algorithm,

- `DataCenters`, which uses hard-coded centres provided by the user,
- `GaussianRandomCenters`, which distributes centres according to a Gaussian distribution,
- `GaussianMixtureRandomCenters`, which distributes centres according to a Gaussian mixture distribution,
- `GridCenters`, which distributes centres evenly on a grid,
- `QmcCenters`, which distributes centres using a Quasi–Monte Carlo algorithm like Latin hypercube sampling [62, 63] (see Listing 8.2), and
- `UniformRandomCenters`, which distributes centres according to a uniform distribution.

```
kp = pykoop.KoopmanPipeline(
    lifting_functions=[(
        'rbf',
        pykoop.RbfLiftingFn(
            rbf='thin_plate',
            centers=pykoop.QmcCenters(
                n_centers=100,
                qmc=scipy.stats.qmc.LatinHypercube,
            ),
        ),
    )],
    regressor=pykoop.Edmd(),
)
```

Listing 8.2: Example usage of the `RbfLiftingFn` class with centres generated using Latin hypercube sampling.

An example of a `KoopmanPipeline` with thin plate RBF lifting functions distributed using Latin hypercube sampling [62, 63] can be found in Listing 8.2.

8.2.4.4 `KernelApproxLiftingFn`

The `KernelApproxLiftingFn` class generates lifting functions that approximate a specified kernel. The only argument taken by `KernelApproxLiftingFn`'s constructor is `kernel_approx`, which controls the kernel to approximate, its shape parameter, and the number of components to use. The `kernel_approx` attribute can be an instance of the `KernelApproximation` abstract class, but it can also be one of `scikit-learn`'s built-in kernel approximation methods found in `sklearn.kernel_approximation`.

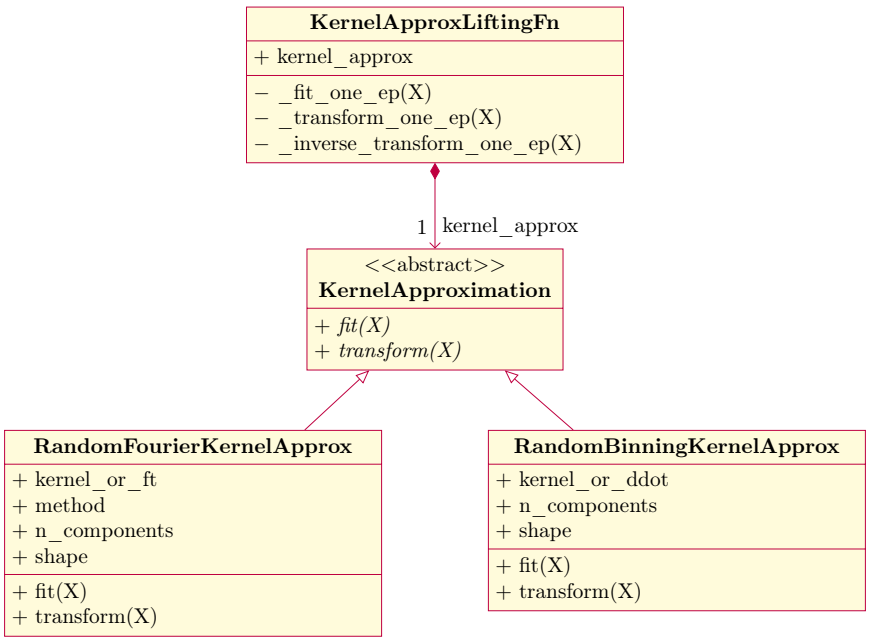


Figure 8.6: Abridged UML class diagram of the `KernelApproxLiftingFn` class, which implements RFF lifting functions. The `KernelApproxLiftingFn` class is composed of one `KernelApproximation` object, which determines the kernel approximation method used.

```
kp = pykoop.KoopmanPipeline(
    lifting_functions=[(
        'rff',
        pykoop.KernelApproxLiftingFn(
            kernel_approx=pykoop.RandomFourierKernelApprox(
                kernel_or_ft='gaussian',
                n_components=100,
                shape=1,
            )),
    )],
    regressor=pykoop.Edmd(),
)
```

Listing 8.3: Example usage of `KernelApproxLiftingFn` to generate lifting functions using RFFs.

The `KernelApproximation` class inherits from `sklearn.base.BaseEstimator` and `sklearn.base.TransformerMixin`, ensuring compatibility with standard cross-validation methods. The two kernel approximation classes provided by `pykoop` are `RandomFourierKernelApprox`, which is discussed in Section 3.5.4 and `RandomBinningKernelApprox`, which is discussed in [26]. An example of a `KoopmanPipeline` with Gaussian RFF lifting functions can be found in Listing 8.3.

8.2.4.5 `SplitPipeline`

The `SplitPipeline` class, shown in Figure 8.7 is different from other lifting functions in that it behaves similarly to `KoopmanPipeline`. This class allows different sets of lifting functions

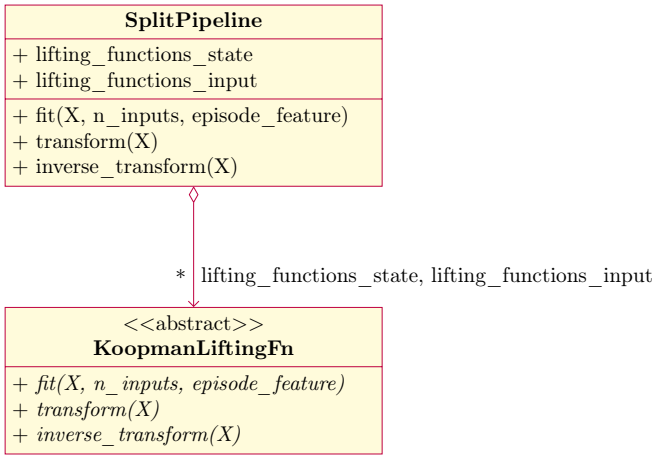


Figure 8.7: Abridged UML class diagram of the `SplitPipeline` class, which aggregates any number of `KoopmanLiftingFn` objects in `lifting_functions_state` and `lifting_functions_input`.

to be applied to states and inputs. As with `KoopmanPipeline`, lifting functions are specified as lists containing tuples, where the first entry of the tuple is a user-defined name and the second entry is the corresponding `KoopmanLiftingFn` object. This class can be used to avoid lifting inputs when identifying Koopman models for control applications, as shown in Listing 8.4.

```
kp = pykoop.SplitPipeline(
    lifting_functions_state=[
        ('polynomial', pykoop.PolynomialLiftingFn(order=2))
    ],
    lifting_functions_input=None,
)
```

Listing 8.4: Example usage of `SplitPipeline` to lift the state with polynomials while leaving the input unlifted.

8.3 Example: Cross-Validation

The regressors and lifting functions discussed in this chapter all depend on hyperparameters. For regressors, they may specify regularization coefficients or SVD truncation criteria. For lifting functions, they may affect the dimension of the lifted state or the shape of a kernel function. Systematically hand-tuning these parameters is intractable for complex Koopman models. Hyperparameters are most often tuned by searching over a grid, searching randomly, or by using Bayesian optimization, among other techniques [148].

A key tool in hyperparameter optimization, regardless of the optimization technique, is *cross-validation* [33, §4.6]. In a cross-validation scheme, a portion of the available data is set aside to be used for final model evaluation. This portion is called the *test set*. In k-fold cross-validation, the remaining data is subdivided into a *training set* and a *validation set* in k different ways, usually randomly. For a given hyperparameter value, k models are trained on the training sets, and are evaluated on the validation sets. The k validation scores are averaged to determine the score of that hyperparameter value. This process is repeated for each hyperparameter value, regardless of whether it originates from a grid search, a random search, or a Bayesian optimization routine. Once the cross-validation procedure is complete, the model with the highest score is re-fit on the combined training and validation sets, and is tested on the test set. By adhering to the `scikit-learn` interface, `pykoop` allows standard cross-validation tools to be used to select lifting functions and regressor hyperparameters. In this section, a simple `scikit-learn` cross-validation example using `pykoop` is presented.

Table 8.1: Cross-validation results for Van der Pol system with RBF lifting functions.

n_centers	alpha	mean_test_score	std_test_score	rank_test_score
20	0	-3.320724	1.170352	9
20	0.1	-3.267083	1.155264	8
20	1	-0.984383	0.612771	3
30	0	-2.084154	0.040298	5
30	0.1	-2.287393	0.093318	6
30	1	-2.356491	0.148787	7
40	0	-1.788004	0.927631	4
40	**0.1**	**-0.436251**	**0.178864**	**1**
40	1	-0.787177	0.433653	2

In this example, Gaussian RBF lifting functions are used to identify a Koopman model of a Van der Pol oscillator [149]. EDMD with Tikhonov regularization is used to approximate the Koopman matrix. This model structure is defined on lines 7–16 of Listing 8.5. The number of RBF centres and the value of the regularization coefficient are left unspeci-

```python
import sklearn.model_selection
import pykoop

# Load example Van der Pol data
eg = pykoop.example_data_vdp()
# Set up Koopman pipeline
kp = pykoop.KoopmanPipeline(
    lifting_functions=[(
        'rbf',
        pykoop.RbfLiftingFn(
            rbf='gaussian',
            centers=pykoop.QmcCenters(random_state=1234),
        ),
    )],
    regressor=pykoop.Edmd(),
)
# Split data episode-by-episode
episode_feature = eg['X_train'][:, 0]
cv = sklearn.model_selection.GroupShuffleSplit(
    random_state=1234,
    n_splits=3,
).split(eg['X_train'], groups=episode_feature)
# Define search parameters
params = {
    'rbf__centers__n_centers': [20, 30, 40],
    'regressor__alpha': [0, 0.1, 1],
}
# Set up grid search
gs = sklearn.model_selection.GridSearchCV(
    estimator=kp,
    param_grid=params,
    cv=cv,
    scoring=pykoop.KoopmanPipeline.make_scorer(),
)
# Fit the pipeline
gs.fit(
    eg['X_train'],
    n_inputs=eg['n_inputs'],
    episode_feature=eg['episode_feature'],
)
# Get results
cv_results = gs.cv_results_
best_estimator = gs.best_estimator_
```

Listing 8.5: Example usage of `pykoop` for cross-validation.

fied. Three-fold cross-validation is implemented using `scikit-learn`'s `GroupShuffleSplit`. This class uses the episode feature to ensure that episodes are not broken up or mixed. The cross-validation specification is found on lines 18–22 of Listing 8.5. A grid search algorithm, `scikit-learn`'s `GridSearchCV`, is used to evaluate every $(\alpha, n_{\text{centres}}) \in \{0, 0.1, 1\} \times \{20, 30, 40\}$. The scoring metric is, by default, the negated mean-squared error. The grid search setup is found on lines 24–34. The resulting `GridSearchCV` object is itself a `scikit-learn` estimator. To perform the cross-validation, this object is fit using the combined training and validation set. The cross-validation results, along with the best estimator, are added as attributes of the `GridSearchCV` object after the fit. A summary of the cross-validation results can be found in Table 8.1.

8.4 Conclusion

This chapter outlines the software architecture of `pykoop`, an open-source Python library for Koopman operator approximation. The `pykoop` library differentiates itself from other offerings through its support of multi-episode training sets and exogenous inputs, through its compatibility with standard cross-validation and hyperparameter optimization tools, and through its ability to arbitrarily compose lifting functions.

Each of these features is enabled by the specific design choices explained throughout the chapter. Support for multi-episode training sets and exogenous inputs is made possible by the data format defined in Section 8.2.1, which all `pykoop` regressors and lifting functions adhere to. The compatibility of `pykoop` with standard cross-validation and hyperparameter tuning is due to the fact that every component of `pykoop` is a `scikit-learn` estimator. The ability to compose lifting functions is thanks to the structure imposed by the `KoopmanPipeline` and `KoopmanLiftingFn` classes defined in Sections 8.2.2 and 8.2.4 respectively.

A limitation of `pykoop`'s architecture is that it is difficult to remove lifting functions after a regressor has been fit. Methods that rely on removing redundant or problematic Koopman modes like [150] would be difficult to implement in `pykoop`. Furthermore, `pykoop` only supports retracting the lifted state by extracting it from the first lifting functions. The ability to recover the lifted state using least-squares as in [12, §3.2.1] would be valuable, as it would remove the restriction that the original system's state must be included in the lifted state.

Given that the contributions outlined in Chapters 4–7 of this thesis are all implemented using `pykoop`, its software architecture has been proven to be user-friendly while remaining flexible enough for research tasks.

Chapter 9

Closing Remarks and Future Work

Learning a Koopman operator amounts to identifying a linear system in a higher-dimensional lifted space. This thesis explores how concepts from linear systems theory can be used to enhance the Koopman operator identification process, and how standard linear optimal control tools can be applied to Koopman models in the lifted space. The Preface contains a comprehensive list of the contributions found in this thesis.

Chapters 4, 5, and 6 consider the Koopman regression problem. In Chapter 4, EDMD and DMDc are reformulated as SDPs. The following two chapters build upon these reformulations by adding additional constraints that incorporate linear systems properties into the regression problem. Chapter 5 considers how two important system properties, asymptotic stability and gain, can be used to shape the Koopman regression problem. In the first part of the chapter, an asymptotic stability constraint is added to the regression problem to ensure that the Koopman model of an AS system shares this property. In the second part, the $\mathcal{H}_\infty$ norm of the Koopman system is used to regularize the regression problem, also guaranteeing asymptotic stability while ensuring that the gain of the Koopman system is reasonable. An appealing feature of $\mathcal{H}_\infty$ norm regularization is the ability to add LTI weighting functions to the regularizer. In Chapter 6, the identification of potentially unstable systems is considered. To practically collect data from an unstable system, a stabilizing controller is required. In the Koopman operator identification process, the effect of this controller must be removed. Chapter 6 proposes a method for simultaneously identifying the closed-loop and open-loop Koopman systems given a known controller, also through the use of constraints on the SDPs reformulation of the Koopman regression problem. Both Chapter 5 and Chapter 6 demonstrate their proposed methods on experimental data.

Chapter 7 discusses how linear robust control tools can be used to synthesize a robust Koopman observer in the lifted space. Due to the linearity of the Koopman operator, a population of Koopman models can be compared through their frequency responses using

standard robust control tools. Uncertainty within a population can be bounded in the frequency domain and used to synthesize controllers or observers that are robust to variation in the population. Chapter 7 specifically considers the robust observer synthesis problem, where the uncertainty model determines the frequency ranges in which the observer's internal model is trustworthy. The proposed approach is tested on an experimental dataset collected from several dozen motor drive systems, where the uncertainty reflects manufacturing variation.

Chapter 8 outlines the software architecture of `pykoop`, an open-source Koopman operator approximation library designed to support the research in Chapters 4–7. The `pykoop` library sets itself apart from other Koopman operator approximation libraries through its compatibility with multi-episode datasets and exogenous inputs, through its ability to construct new lifting functions through composition, and through its compatibility with standard hyperparameter optimization tools.

The limitations of each contribution, along with proposed enhancements, are discussed at the end of each chapter. A high-level discussion of future research directions follows.

9.1 Future Work

Chapter 5 considers properties of the approximate Koopman system from a practical perspective: it is often convenient to work with AS systems, or systems with low gain. However, it is not clear how the properties of an approximated Koopman system relate to the properties of the original nonlinear system. There is a relationship between the asymptotic stability of a Koopman system and the original system [68], but there is no clear relationship for system norms. Future theoretical work might ask what, if anything, the $\mathcal{H}_\infty$ norm of a Koopman system says about the original nonlinear system, which does not have a norm defined.

A pervasive issue in designing Koopman lifting functions in Chapters 6 and 7 is the inability to lift exogenous inputs. Leaving exogenous inputs unlifted has been shown to limit the types of systems that can accurately be represented with a Koopman model [67]. Bilinear lifting functions are shown to represent a much wider class of systems, but are still difficult to work with in control applications.

A final open question at the intersection of control theory and Koopman operator theory is whether Koopman models intended for control might have different properties than Koopman models intended for prediction alone. Koopman models with excellent predictions do not always result in good controllers, and the reasons for this are so far unclear. Properties like the closure of the Koopman model, the number and type of lifting functions, the sensitivity of the lifting functions in the region of interest, and the methods used to recover the original system's state may all have different effects when synthesizing Koopman controllers using

optimal control tools.

The Koopman operator presents a tremendous opportunity to combine tried-and-true linear systems and control techniques with novel machine learning methods, but more fundamental work is required before these techniques can be applied to Koopman systems with the same ease as linear systems.

Appendices

Appendix 1

Motor Drive Uncertainty Characterization

This appendix contains additional figures relating to the uncertainty characterization of the population of 38 motor drives. Specifically, Appendix A.1 contains the input-to-output frequency responses for each unloaded motor drive, Appendix A.2 contains the residuals of the unloaded motor drives for each of the six uncertainty forms, and Appendix A.3 contains the loaded outlier residuals for each of the six uncertainty forms.

A.1 Frequency Responses of Linear and Koopman Models

This appendix contains the input-to-output frequency responses of each unloaded linear and Koopman drive model.

Figure A.1 shows the frequency responses of the linear models. The magnitude of $G_{11}(f)$ is close to 0 dB for every motor drive up to 100 Hz, indicating that the tracking error of the drive's controller is low at low frequencies. The magnitude of $G_{22}(f)$ is similarly 0 dB for all motor drives below 100 Hz, but at higher frequencies, the transfer functions vary significantly more.

Figure A.2 shows the frequency responses of the Koopman models. The magnitudes of $G_{11}(f)$ and $G_{22}(f)$ are similar to those of the linear models. However, the variation between the individual motor drives appears to be lower for $G_{22}(f)$ and higher for all other transfer functions.

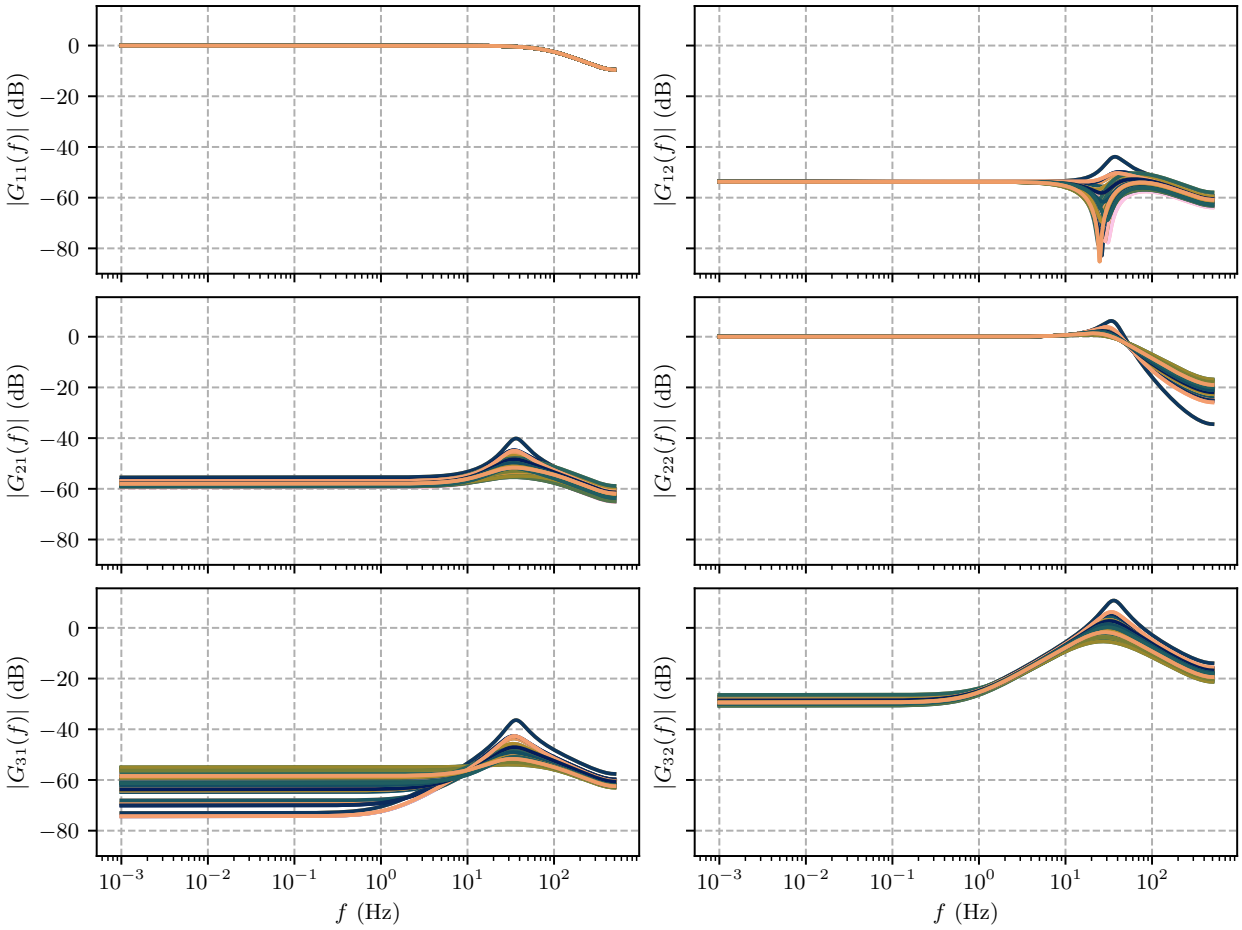


Figure A.1: Frequency responses of linear models of the 38 motor drives under test, from each input to each output. Notably, $G_{11}(f)$ and $G_{22}(f)$ have unity gain throughout the frequency range, indicating that the position and velocity track the reference well, at least below 100 Hz.

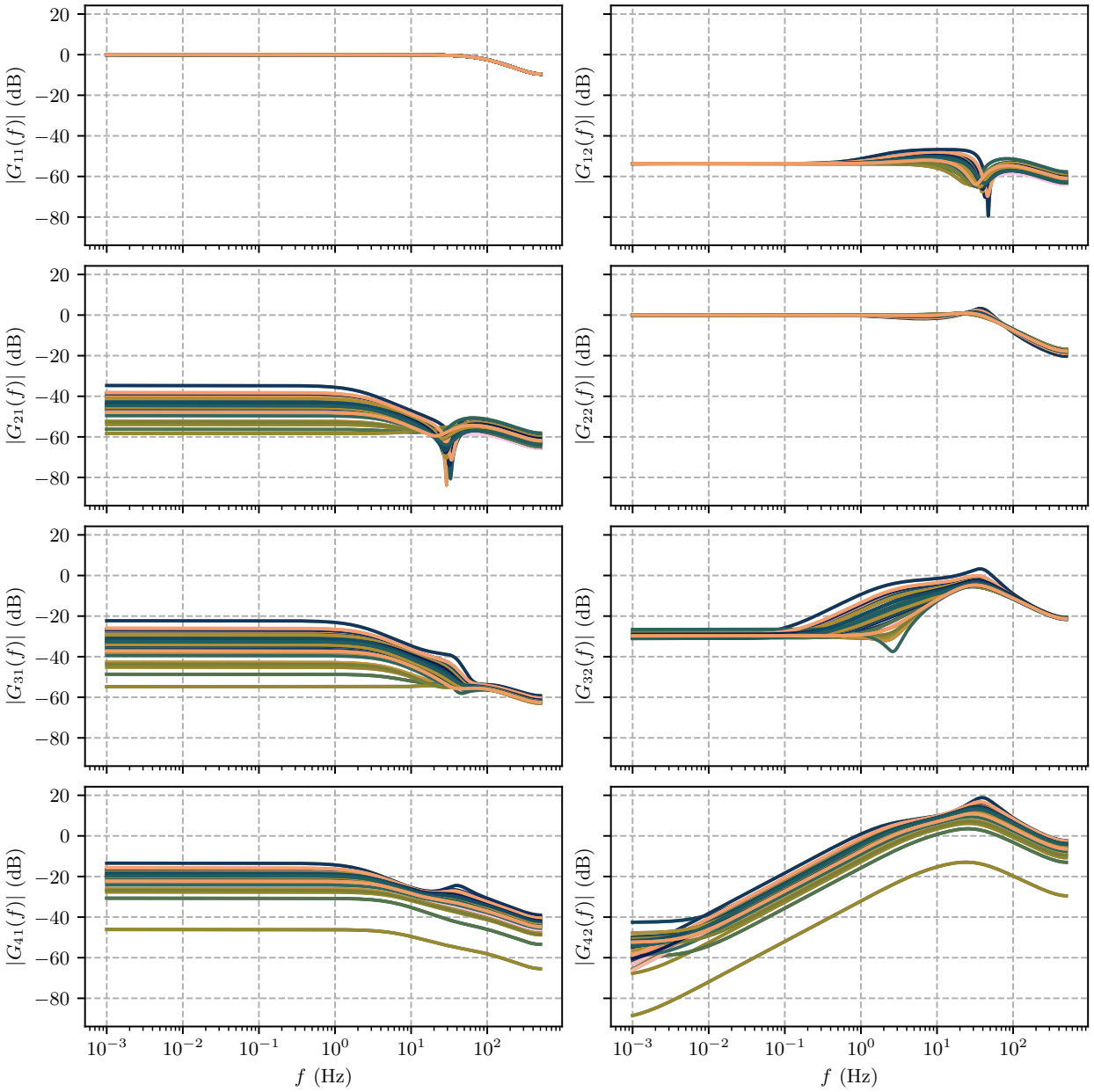
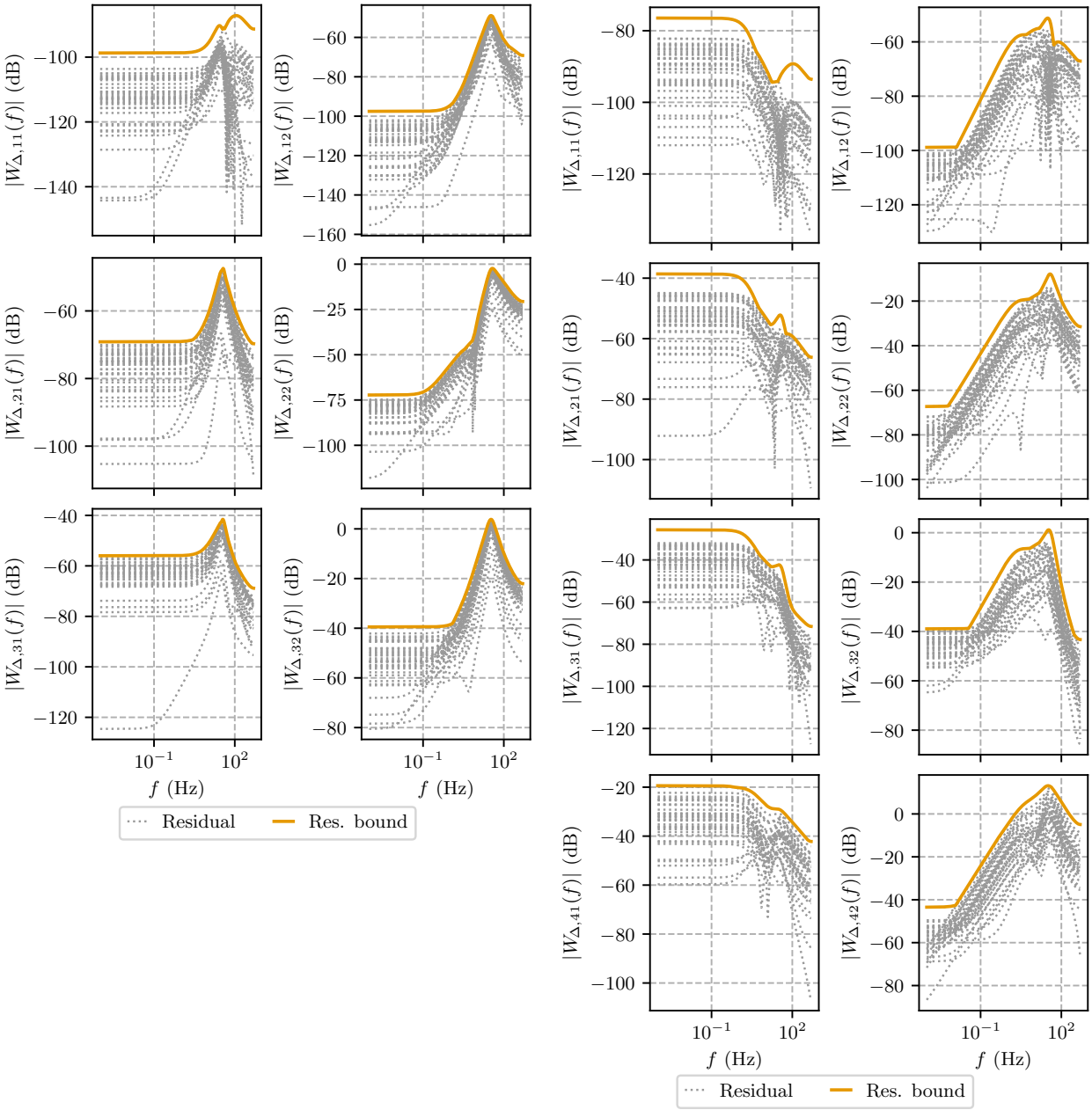


Figure A.2: Frequency responses of Koopman models of the 38 motor drives under test, from each input to each output. Notably, $G_{11}(f)$ and $G_{22}(f)$ have unity gain throughout the frequency range, indicating that the position and velocity track the reference well, at least below 100 Hz. The frequency responses of the Koopman models are significantly more varied than those of the linear models.

A.2 Residuals of Unloaded Linear and Koopman Models

This appendix contains the input-to-output residuals of each unloaded linear and Koopman drive model, for each of the six uncertainty forms.

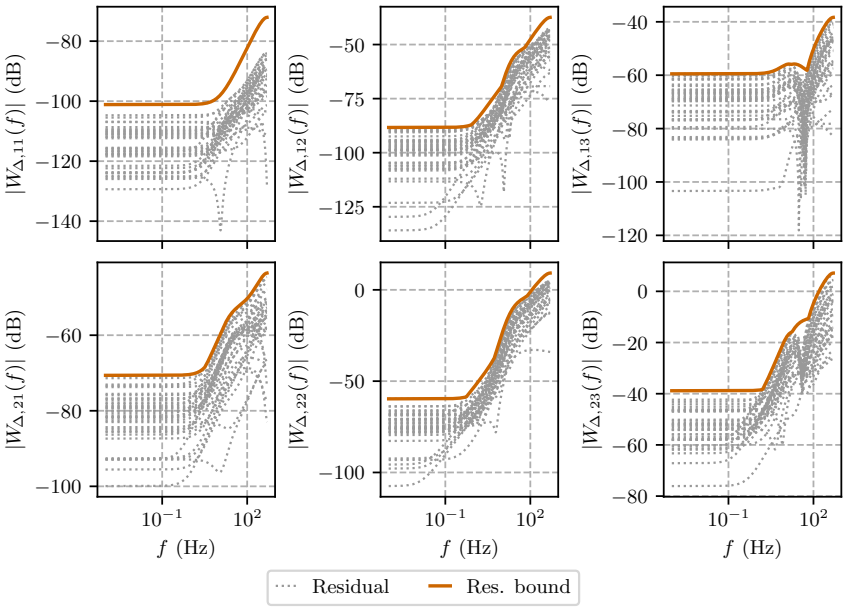
The additive residuals are shown in Figure A.3, the inverse additive residuals are shown in Figure A.4, the input multiplicative residuals are shown in Figure A.5, the inverse input multiplicative residuals are shown in Figure A.6, the output multiplicative residuals are shown in Figures A.7 and A.8, and the inverse output multiplicative residuals are shown in Figures A.9 and A.10.



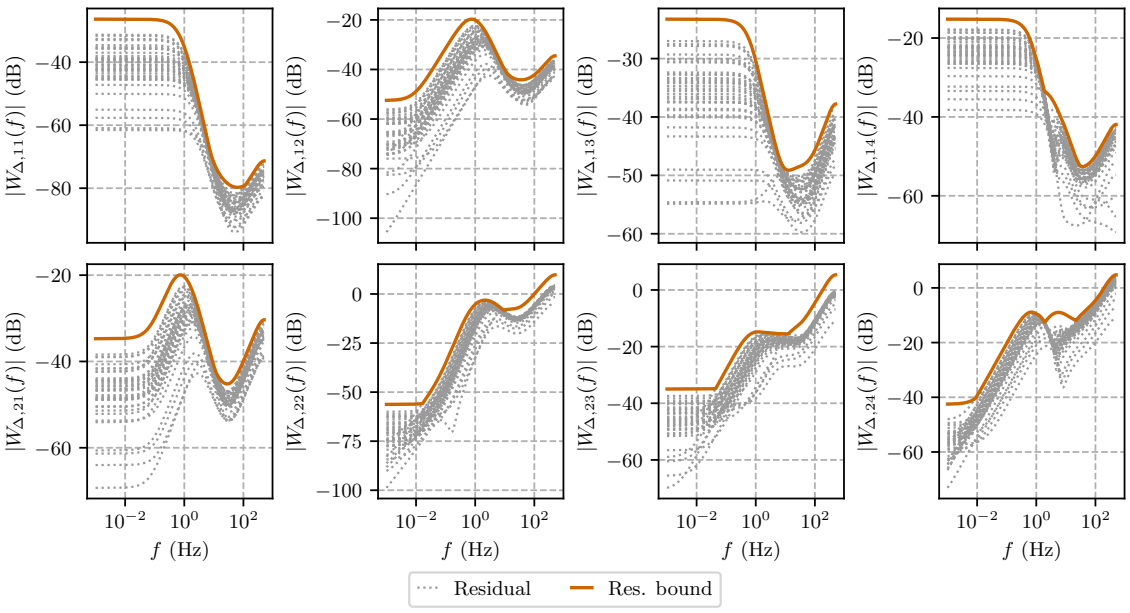
(a) Linear models.

(b) Koopman models.

Figure A.3: Additive residuals.

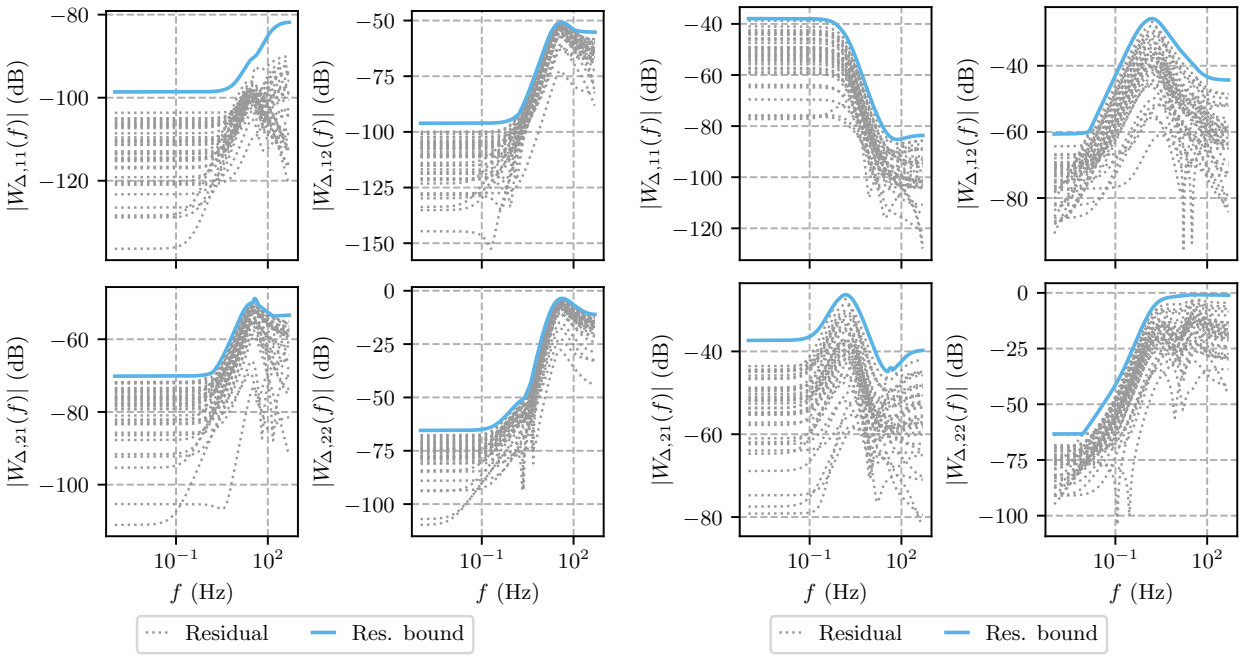


(a) Linear models.



(b) Koopman models.

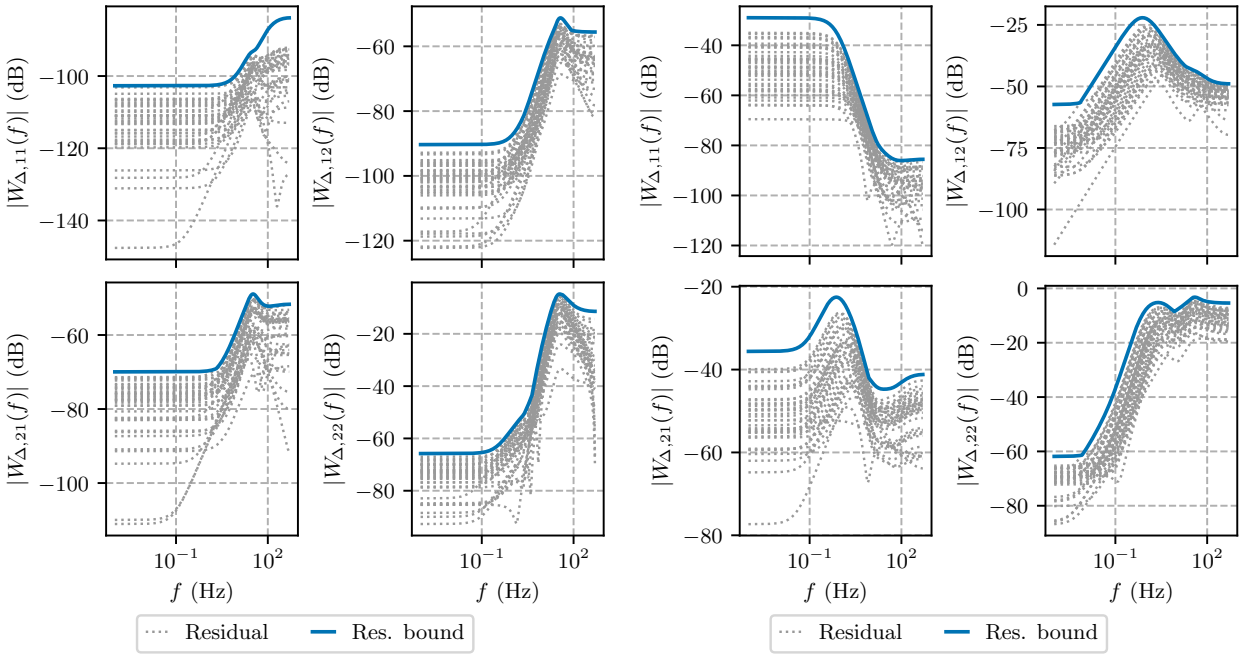
Figure A.4: Inverse additive residuals.



(a) Linear models.

(b) Koopman models.

Figure A.5: Input multiplicative residuals.



(a) Linear models.

(b) Koopman models.

Figure A.6: Inverse input multiplicative residuals.

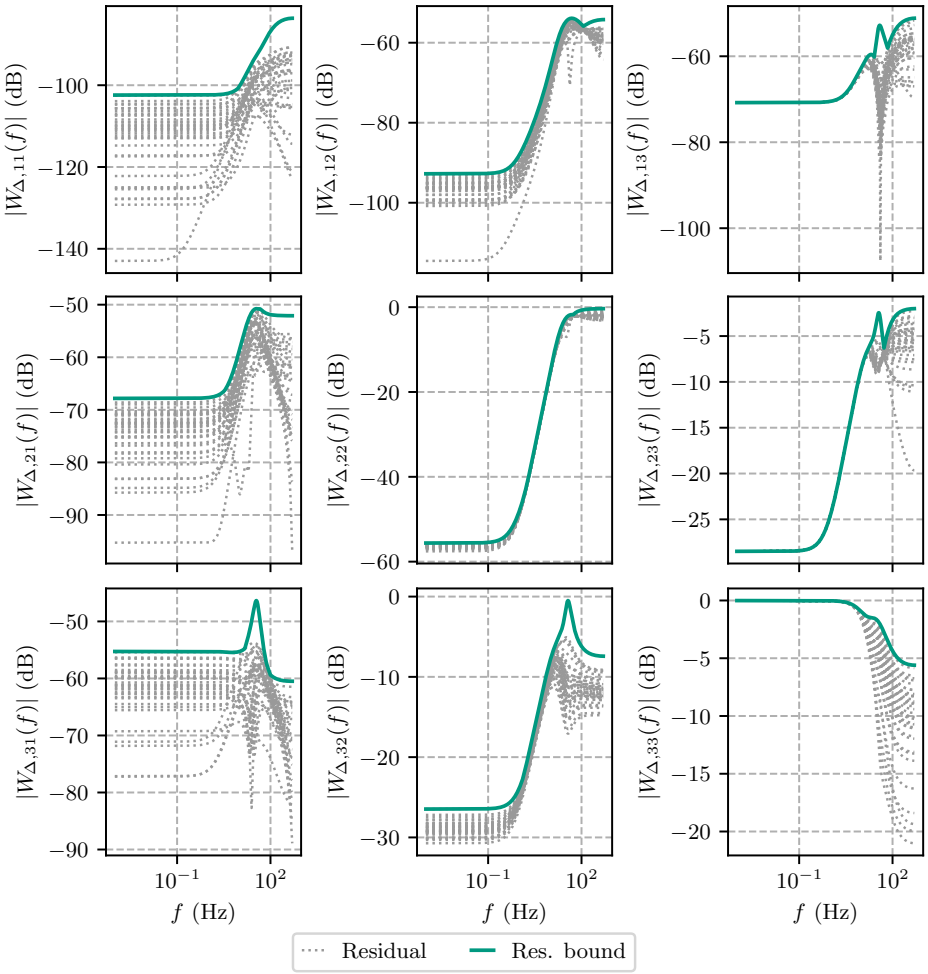


Figure A.7: Linear output multiplicative residuals.

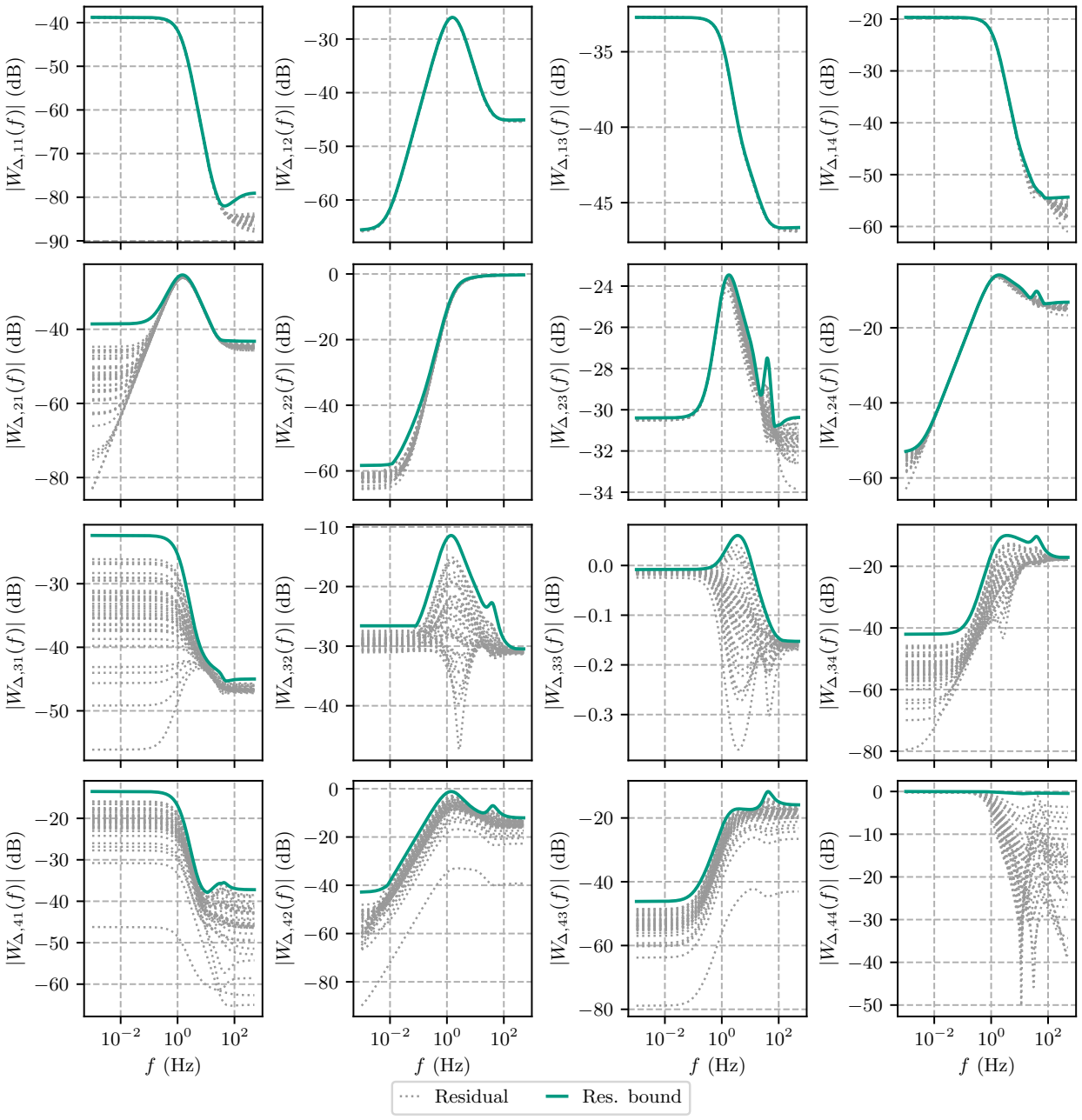


Figure A.8: Koopman output multiplicative residuals.

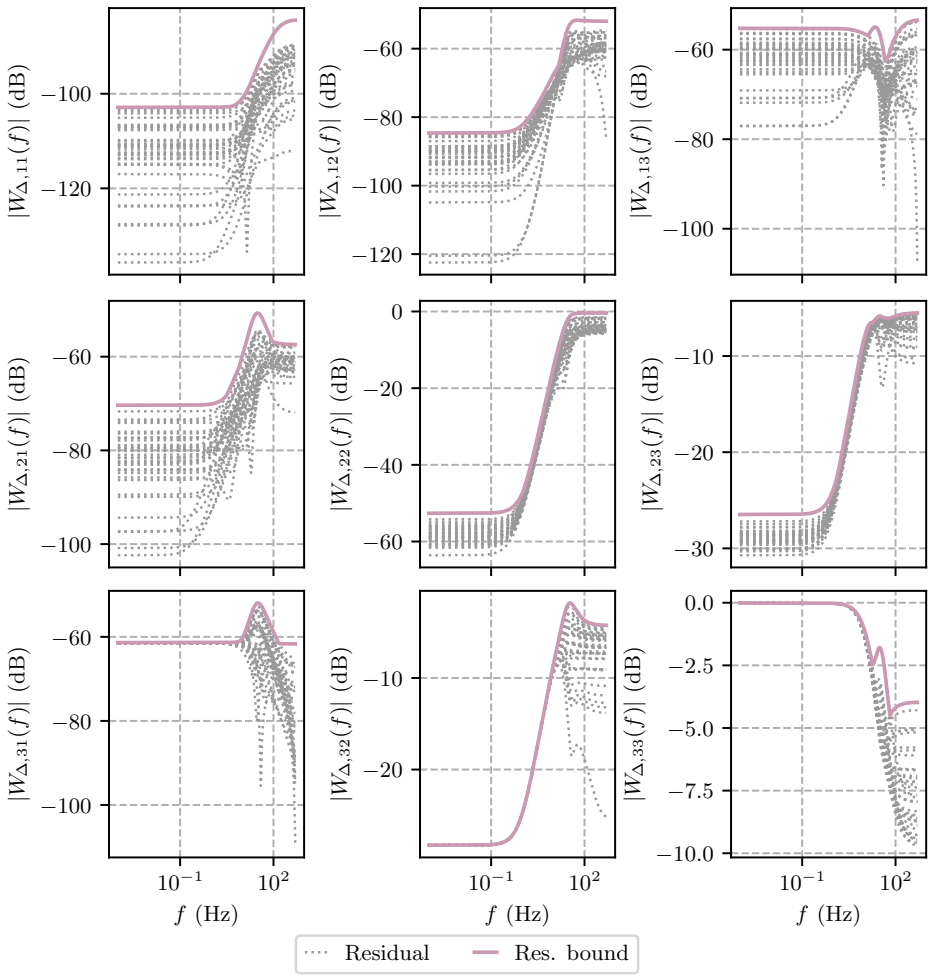


Figure A.9: Linear inverse output multiplicative residuals.

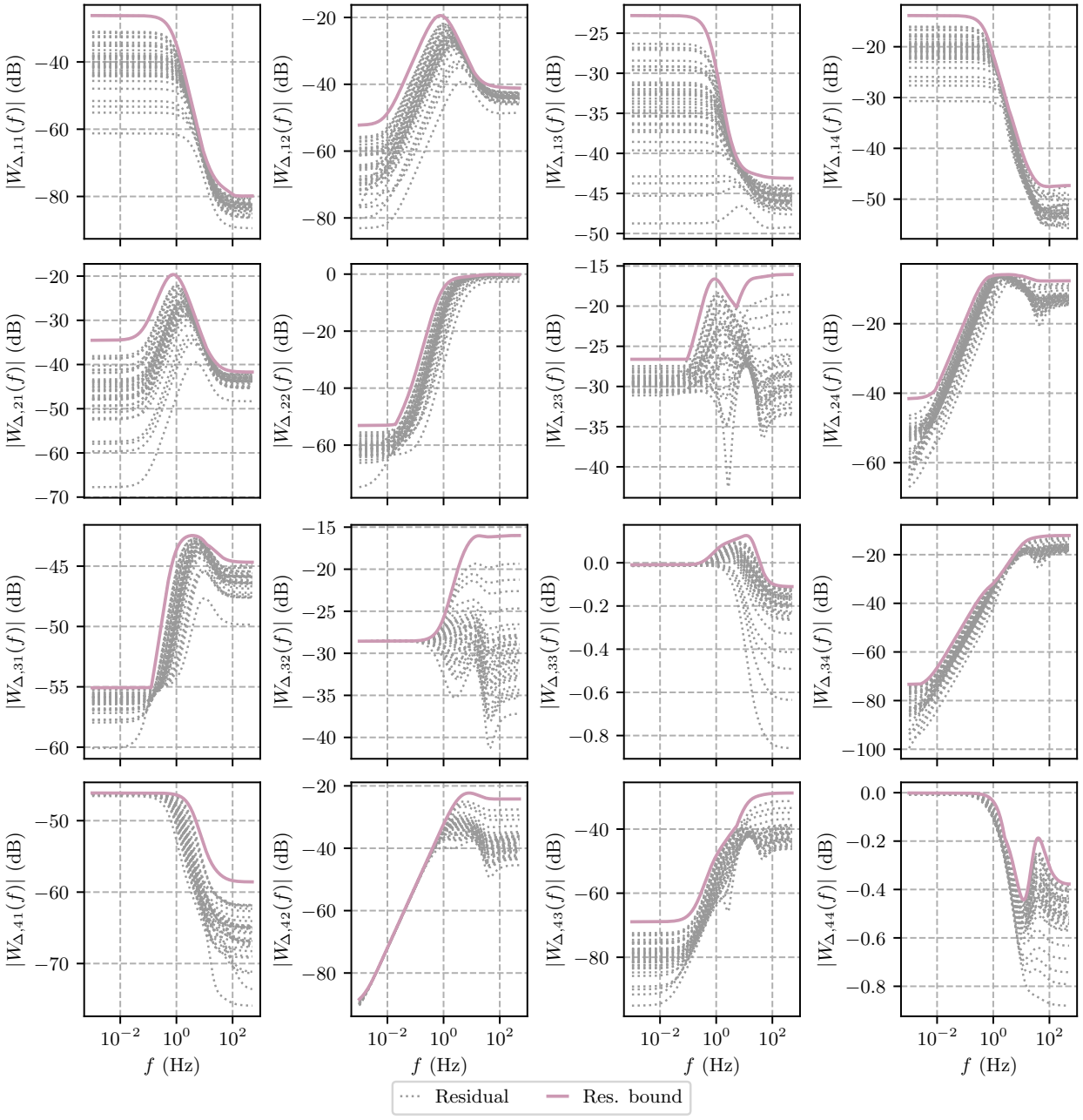
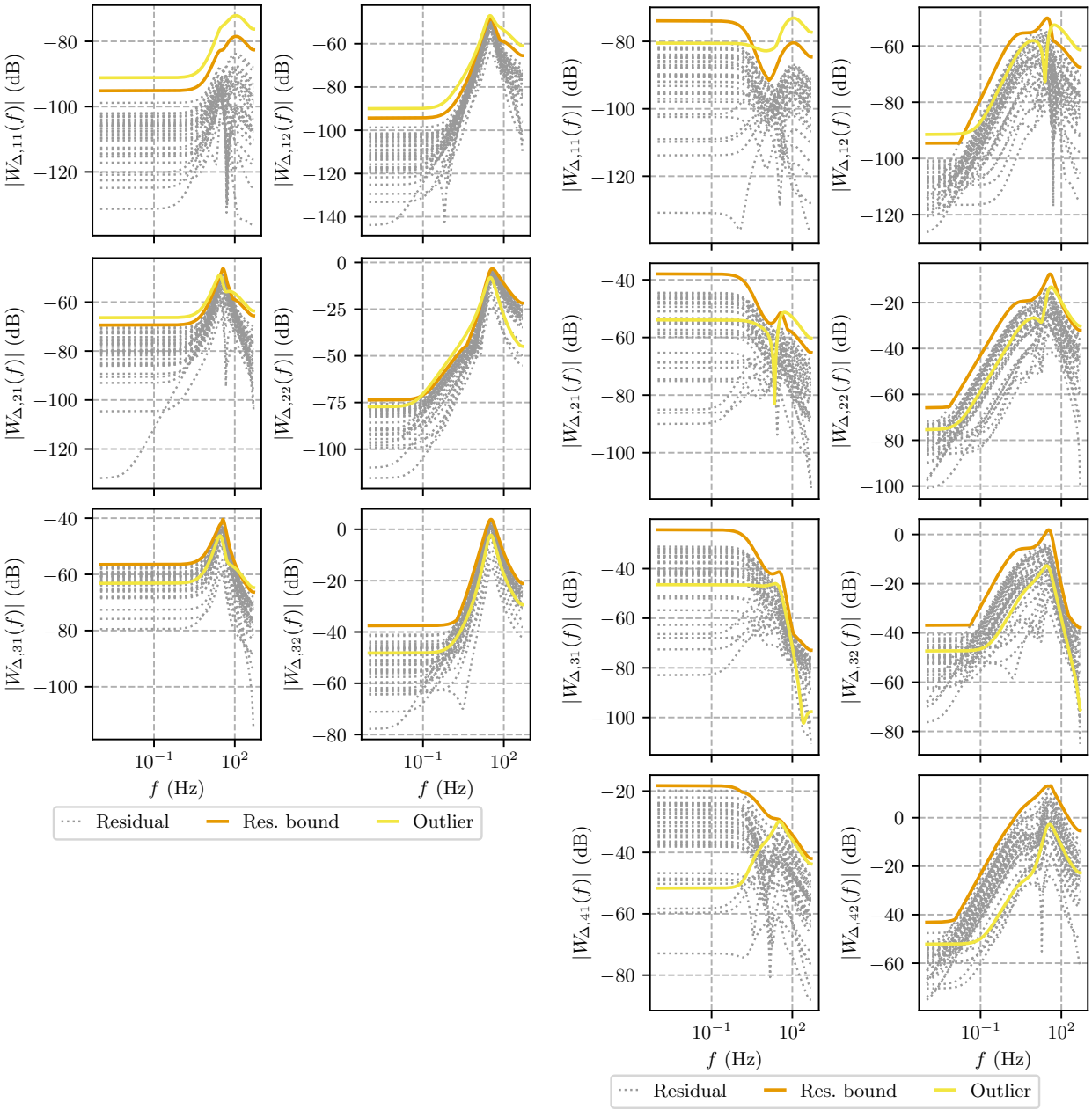


Figure A.10: Koopman inverse output multiplicative residuals.

A.3 Outlier Residual for Loaded Linear and Koopman Models

This appendix contains the input-to-output residuals of each loaded linear and Koopman drive model, for each of the six uncertainty forms, along with the outlier residuals. In each figure, the outlier residual exceeds the residual bound in at least one of the transfer functions, meaning it could be detected with a weighting function that is sufficiently close to the residual bound.

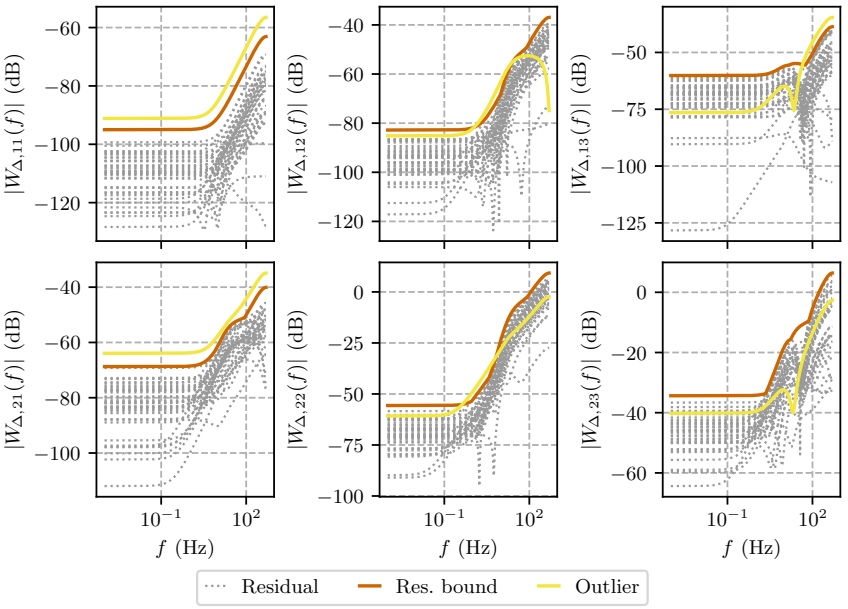
The additive residuals are shown in Figure A.11, the inverse additive residuals are shown in Figure A.12, the input multiplicative residuals are shown in Figure A.13, the inverse input multiplicative residuals are shown in Figure A.14, the output multiplicative residuals are shown in Figures A.15 and A.16, and the inverse output multiplicative residuals are shown in Figures A.17 and A.18.



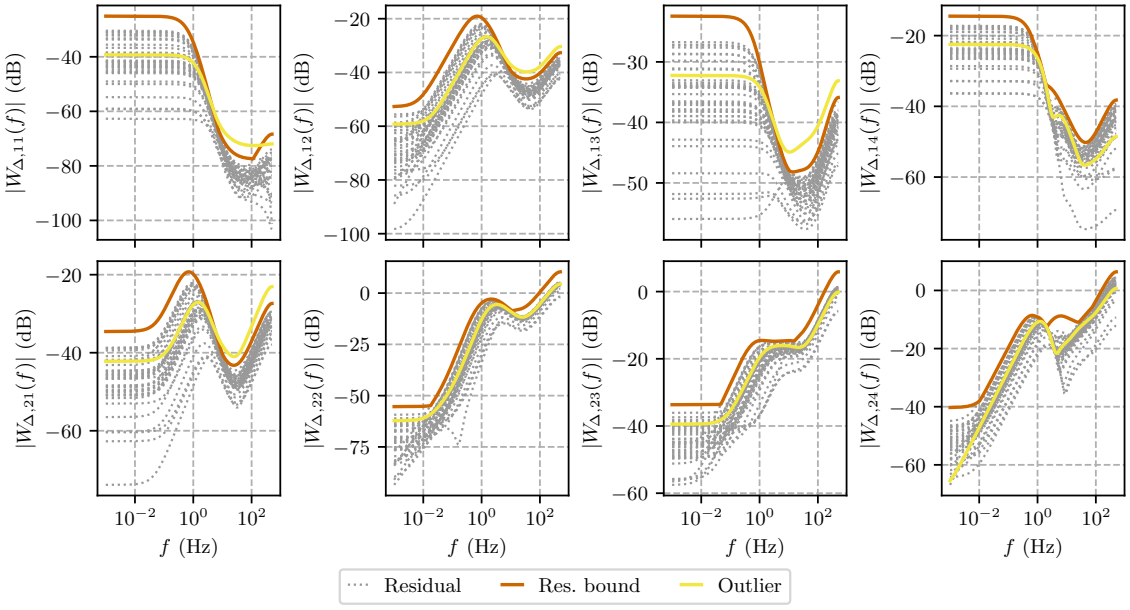
(a) Linear models.

(b) Koopman models.

Figure A.11: Additive residuals and outlier.

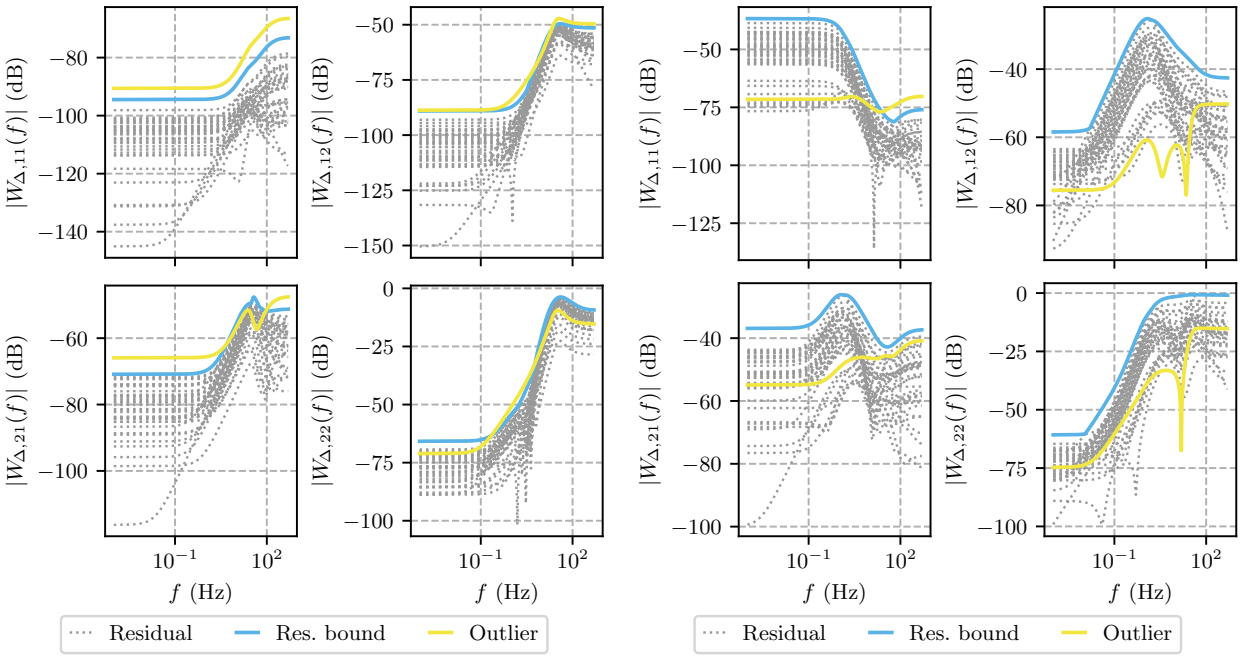


(a) Linear models.



(b) Koopman models.

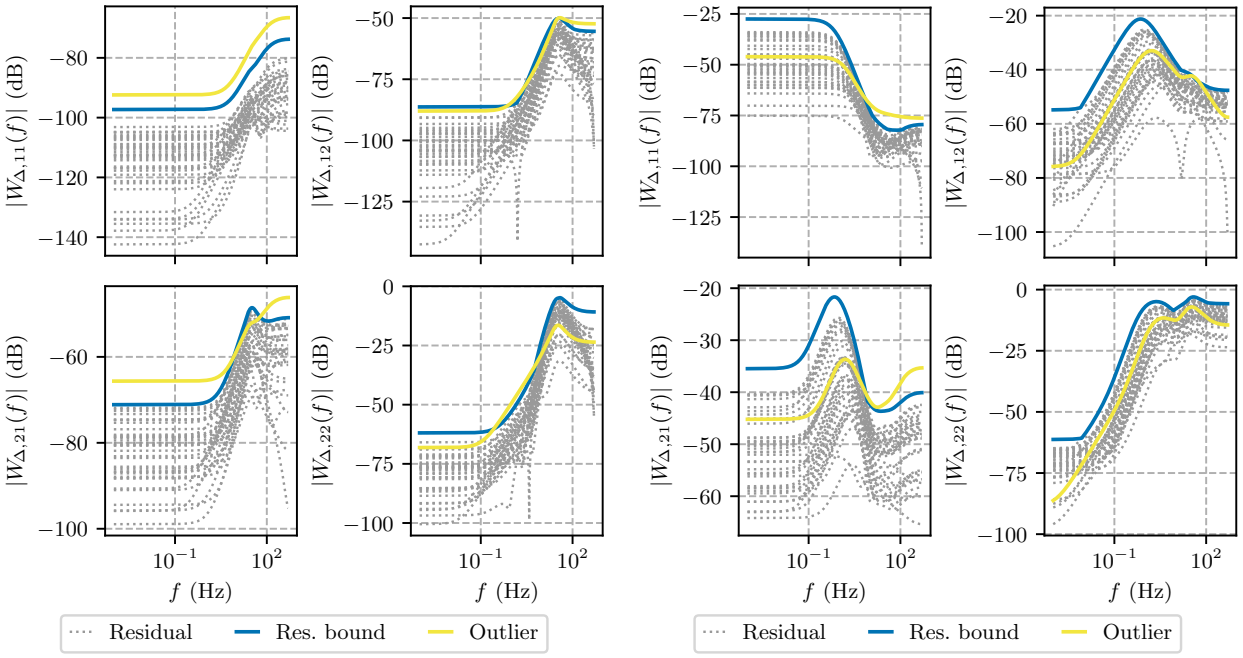
Figure A.12: Inverse additive residuals and outlier.



(a) Linear models.

(b) Koopman models.

Figure A.13: Input multiplicative residuals and outlier.



(a) Linear models.

(b) Koopman models.

Figure A.14: Inverse input multiplicative residuals and outlier.

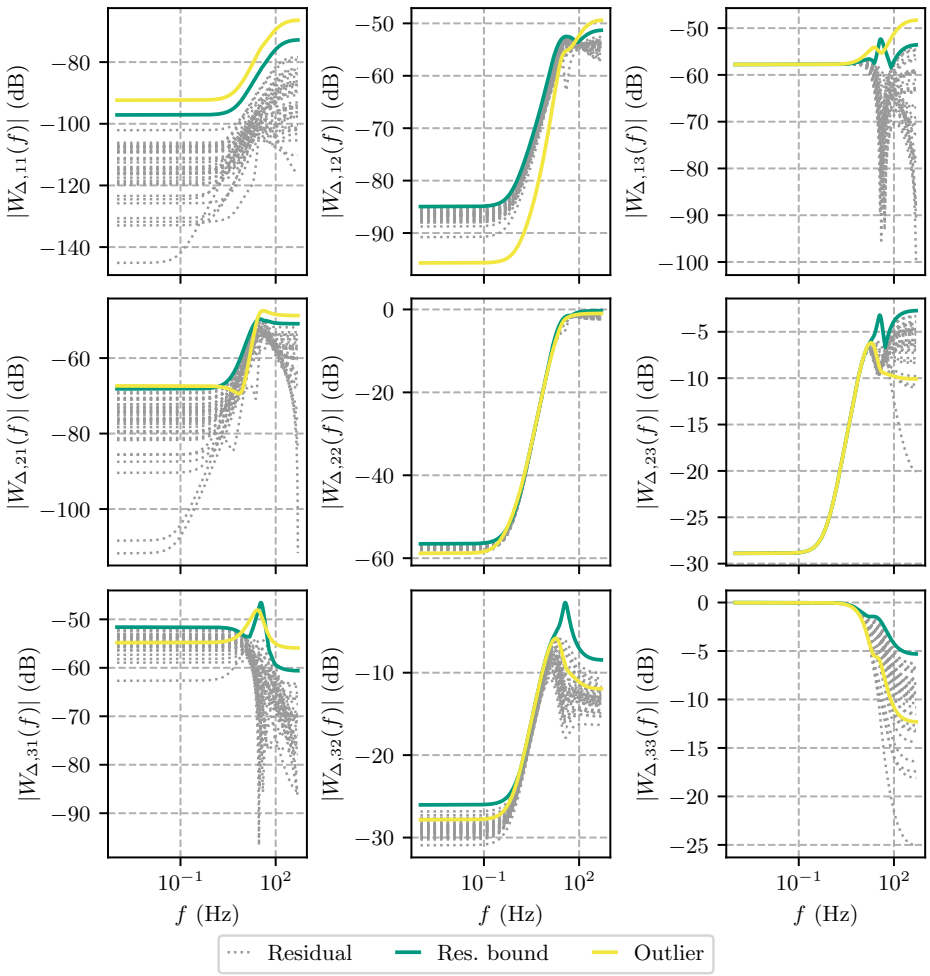


Figure A.15: Linear output multiplicative residuals and outlier.

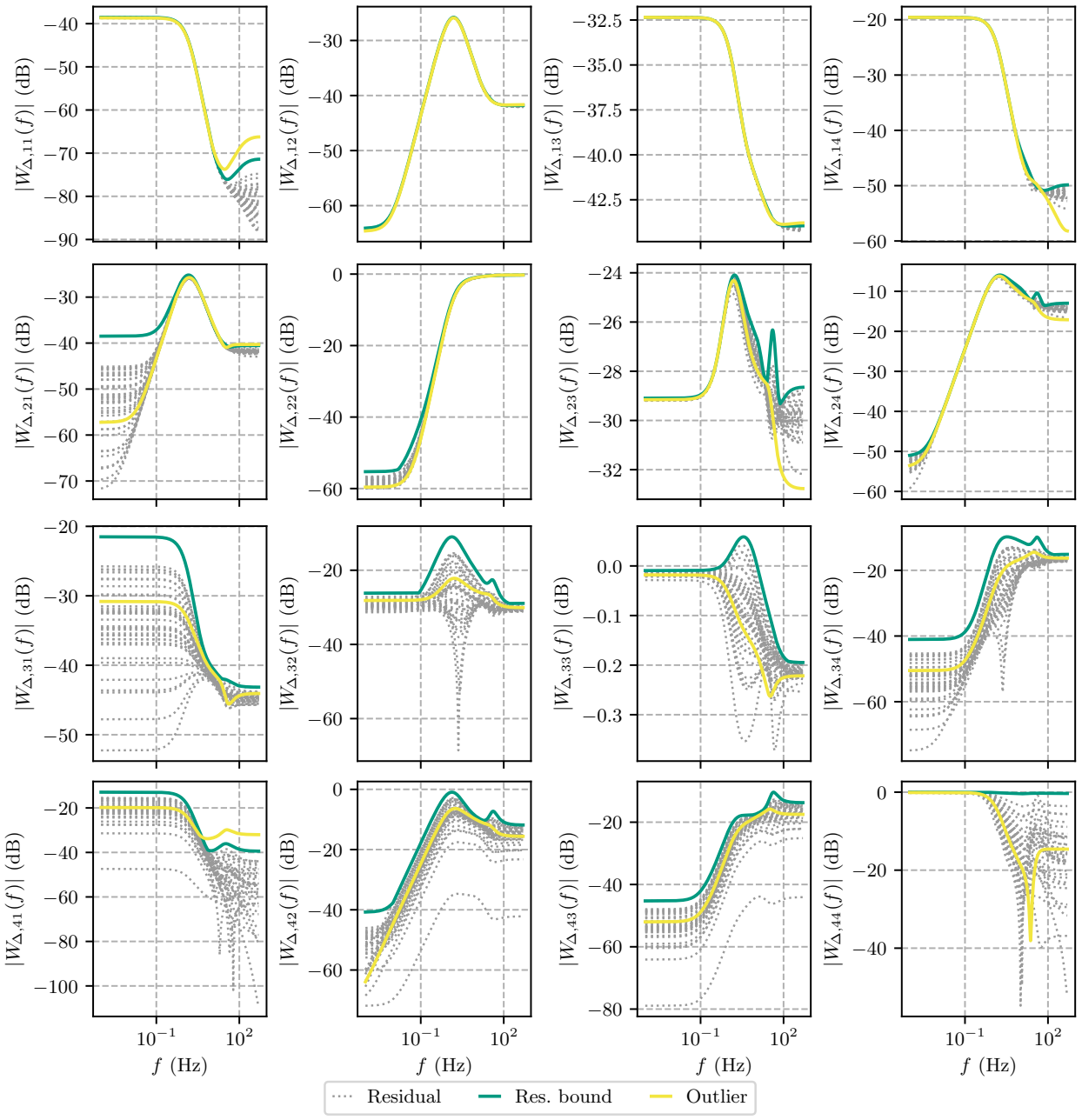


Figure A.16: Koopman output multiplicative residuals and outlier.

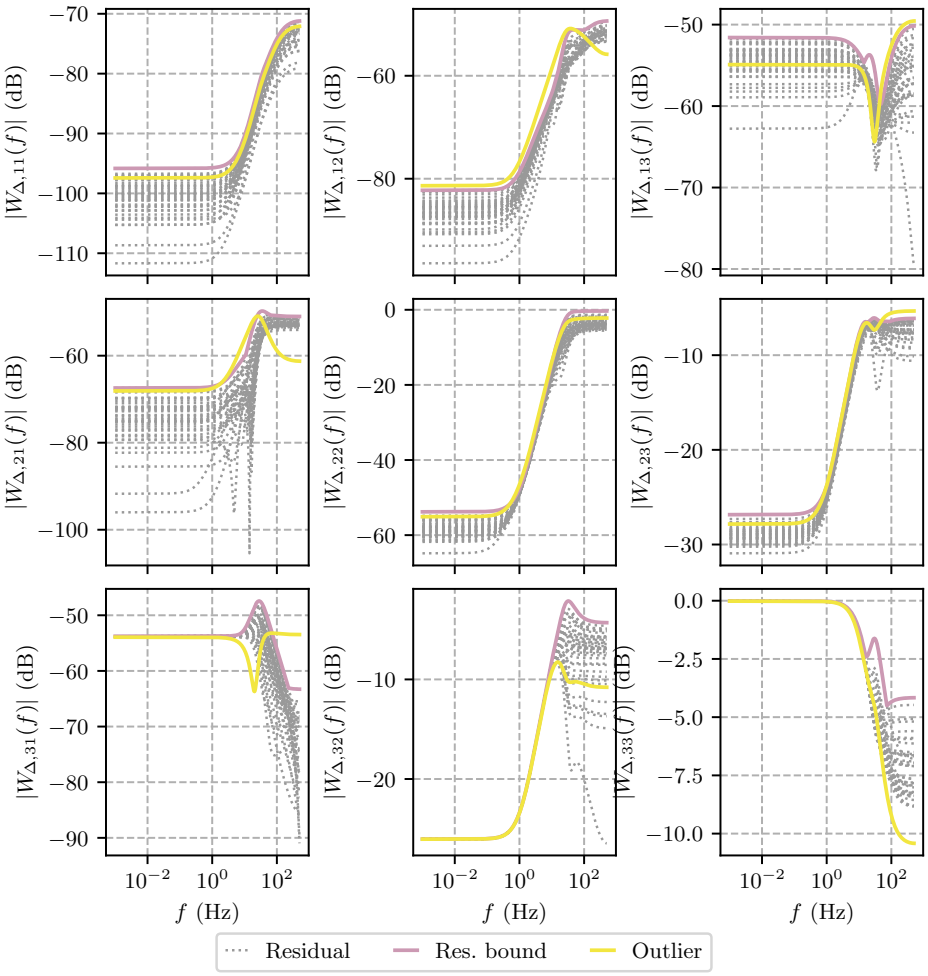


Figure A.17: Linear inverse output multiplicative residuals and outlier.

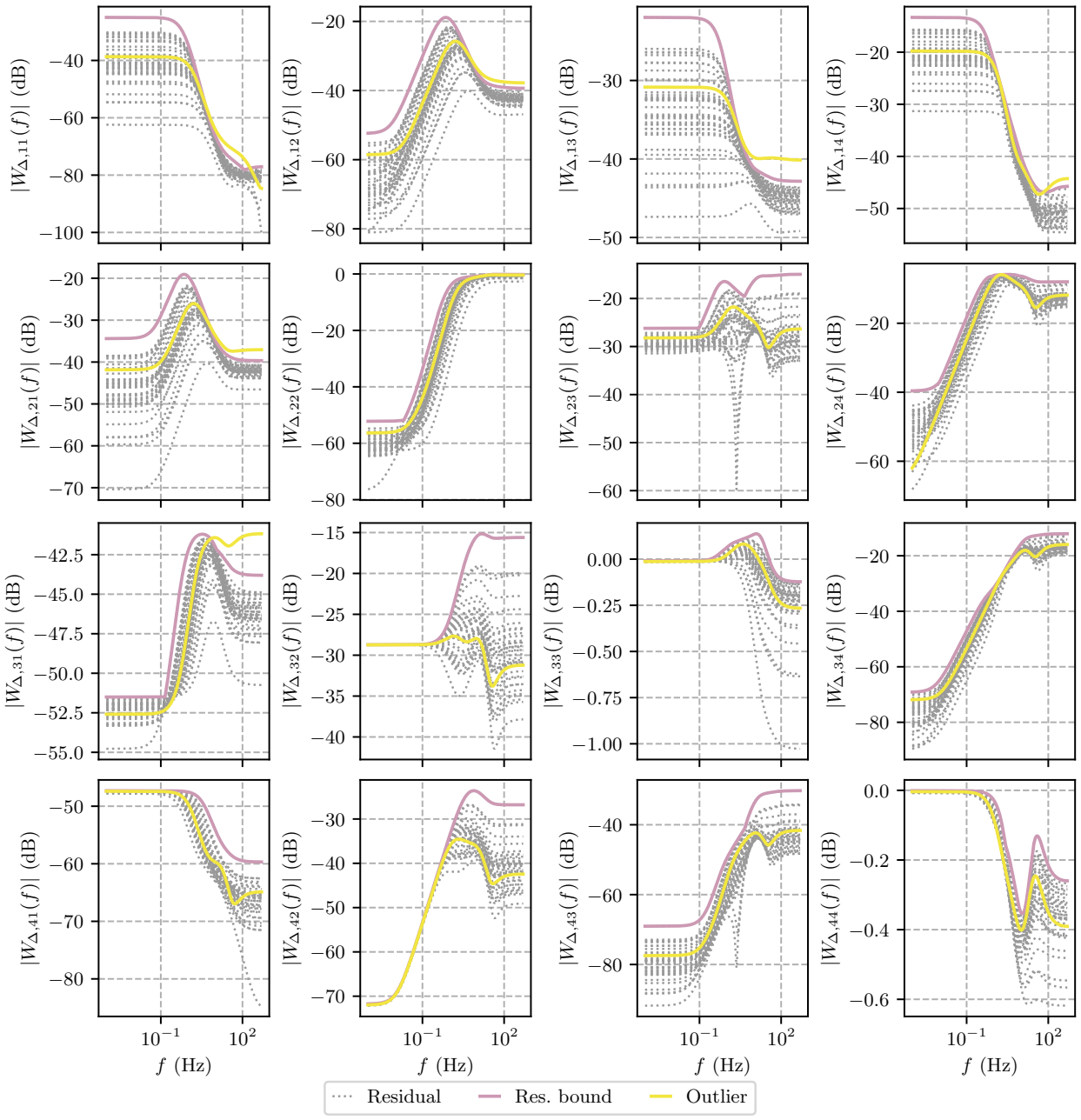


Figure A.18: Koopman inverse output multiplicative residuals and outlier.

Bibliography

[1] S. Dahdah and J. R. Forbes, "Linear matrix inequality approaches to Koopman operator approximation", arXiv:2102.03613v2 [eess.SY], 2021.

[2] S. Dahdah and J. R. Forbes, "System norm regularization methods for Koopman operator approximation", *Proc. R. Soc. A*, vol. 478, no. 2265, 2022. DOI: 10.1098/rspa.2022.0162.

[3] S. Dahdah and J. R. Forbes, "Closed-loop Koopman operator approximation", *Mach. Learn.: Sci. Technol.*, vol. 5, no. 2, p. 025 038, 2024. DOI: 10.1088/2632-2153/ad45b0.

[4] S. Dahdah and J. R. Forbes, "Uncertainty modelling and robust observer synthesis using the Koopman operator", arXiv:2410.01057v1 [eess.SY], 2024.

[5] S. Dahdah and J. R. Forbes, *Quantifying manufacturing variation in motor drives*, FRDR, 2024. DOI: 10.20383/103.01057.

[6] S. Dahdah and J. R. Forbes, *decargroup/pykoop v1.2.3*, Zenodo, 2023. DOI: 10.5281/zenodo.7464660.

[7] N. J. Kasdin and D. A. Paley, *Engineering Dynamics: A Comprehensive Introduction*. Princeton University Press, 2011, ISBN: 9780691135373.

[8] B. O. Koopman, "Hamiltonian systems and transformations in Hilbert space", *Proc. Nat. Acad. Sci.*, vol. 17, no. 5, pp. 315–318, 1931. DOI: 10.1073/pnas.17.5.315.

[9] I. Mezić, "Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry", *J. Nonlinear Sci.*, vol. 30, no. 5, pp. 2091–2145, 2019. DOI: 10.1007/s00332-019-09598-5.

[10] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism", *Chaos*, vol. 22, no. 4, 047510, 2012. DOI: 10.1063/1.4772195.

[11] A. Mauroy, I. Mezić, and Y. Susuki, Eds., *The Koopman Operator in Systems and Control*. Cham, Switzerland: Springer, 2020, ISBN: 9783030357122.

[12] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control", *Automatica*, vol. 93, pp. 149–160, 2018. DOI: 10.1016/j.automatica.2018.03.046.

[13] S. E. Otto and C. W. Rowley, "Koopman operators for estimation and control of dynamical systems", *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, no. 1, pp. 59–87, 2021. DOI: 10.1146/annurev-control-071020-010108.

[14] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using Koopman operators", *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1071–1083, 2019. DOI: 10.1109/tro.2019.2923880.

[15] G. Mamakoukas, M. Castano, X. Tan, and T. Murphey, "Local Koopman operators for data-driven control of robotic systems", in *Proc. Robot.: Sci. Syst. XV*, Freiburg im Breisgau, Germany, 2019. DOI: 10.15607/RSS.2019.XV.054.

[16] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control", in *Proc. Robot.: Sci. Syst. XV*, Freiburg im Breisgau, Germany, 2019. DOI: 10.15607/rss.2019.xv.060.

[17] D. Uchida, A. Yamashita, and H. Asama, "Data-driven Koopman controller synthesis based on the extended $\mathcal{H}_2$ norm characterization", *IEEE Contr. Syst. Lett.*, vol. 5, no. 5, pp. 1795–1800, 2021. DOI: 10.1109/lcsys.2020.3042827.

[18] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control", *Mach. Learn.: Sci. Technol.*, vol. 2, no. 3, p. 035 023, 2021. DOI: 10.1088/2632-2153/abf0f5.

[19] L. Shi *et al.*, "Koopman operators in robot learning", arXiv:2408.04200v1[cs.RO], 2024.

[20] L. Ljung, T. Chen, and B. Mu, "A shift in paradigm for system identification", *Int. J. Control*, vol. 93, no. 2, pp. 173–180, 2019. DOI: 10.1080/00207179.2019.1578407.

[21] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern Koopman theory for dynamical systems", *SIAM Review*, vol. 64, no. 2, pp. 229–340, 2022. DOI: 10.1137/21m1401243.

[22] I. Abraham, G. de la Torre, and T. Murphey, "Model-based control using Koopman operators", in *Proc. Robot.: Sci. Syst. XIII*, Cambridge, MA, 2017. DOI: 10.15607/rss.2017.xiii.052.

[23] A. Mallen, C. A. Keller, and J. N. Kutz, "Koopman-inspired approach for identification of exogenous anomalies in nonstationary time-series data", *Mach.*

Learn.: Sci. Technol., vol. 4, no. 2, p. 025 033, 2023. DOI: 10.1088/2632-2153/acdd50.

[24] Z. C. Guo, V. Korotkine, J. R. Forbes, and T. D. Barfoot, "Koopman linearization for data-driven batch state estimation of control-affine systems", *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 866–873, 2022. DOI: 10.1109/LRA.2021.3133587.

[25] A. M. DeGennaro and N. M. Urban, "Scalable extended dynamic mode decomposition using random kernel approximation", *SIAM J. Sci. Comput.*, vol. 41, no. 3, A1482–A1499, 2019. DOI: 10.1137/17m115414x.

[26] A. Rahimi and B. Recht, "Random features for large-scale kernel machines", in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, Vancouver, Canada: Curran Associates, 2007, pp. 1177–1184. [Online]. Available: https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html (visited on 08/12/2024).

[27] S. Pan and K. Duraisamy, "On the structure of time-delay embedding in linear models of non-linear dynamical systems", *Chaos*, vol. 30, no. 7, 2020. DOI: 10.1063/5.0010886.

[28] M. D. Kvalheim and P. Arathoon, "Linearizability of flows by embeddings", arXiv:2305.18288v5 [math.DS], 2024.

[29] Z. Liu, N. Ozay, and E. D. Sontag, "On the non-existence of immersions for systems with multiple omega-limit sets", *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 60–64, 2023. DOI: 10.1016/j.ifacol.2023.10.1408.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009, ISBN: 9780521833783.

[31] R. J. Caverly and J. R. Forbes, "LMI properties and applications in systems, stability, and control theory", arXiv:1903.08599v4 [cs.SY], 2024.

[32] G. Strang, *Linear Algebra and its Applications*. Cengage Learning, 2006, ISBN: 9780030105678.

[33] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering*. Cambridge University Press, 2019, ISBN: 9781108422093.

[34] M. Gavish and D. L. Donoho, "The optimal hard threshold for singular values is $4/\sqrt{3}$", *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 5040–5053, 2014. DOI: 10.1109/tit.2014.2323359.

[35] B. Recht, M. Fazel, and P. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization", *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010. DOI: 10.1137/070697835.

[36] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Englewood Cliffs, NJ: Prentice Hall, 1995, ISBN: 9780134565675.

[37] H. J. Marquez, *Nonlinear Control Systems: Analysis and Design*. Wiley, 2003, ISBN: 9780471427995.

[38] M. Green and D. J. N. Limebeer, *Linear Robust Control*. London, England: Prentice Hall, 1994, ISBN: 9780131022782.

[39] T. Katayama, *Subspace Methods for System Identification*. Springer, 2005, ISBN: 9781852339814.

[40] P. J. Antsaklis and A. N. Michel, *A Linear Systems Primer*. Birkhäuser, 2007, ISBN: 9780817644604.

[41] T. Chen and B. A. Francis, *Optimal Sampled-Data Control Systems*. Springer, 1995. DOI: 10.1007/978-1-4471-3037-6.

[42] A. Megretski, *Multivariable control systems: Interpretations for standard optimization setup*, 2004. [Online]. Available: https://ocw.mit.edu/courses/6-245-multivariable-control-systems-spring-2004/resources/lec2_6245_2004/ (visited on 07/23/2024).

[43] R. Toscano, *Structured Controllers for Uncertain Systems: A Stochastic Optimization Approach*. Springer, 2013, ISBN: 9781447159155.

[44] G. Sagnol and M. Stahlberg, "PICOS: A Python interface to conic optimization solvers", *J. Open Source Softw.*, vol. 7, no. 70, p. 3915, 2022. DOI: 10.21105/joss.03915.

[45] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization", *J. Mach. Learn. Res.*, vol. 17, pp. 1–5, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-408.html (visited on 08/12/2024).

[46] M. Andersen, J. Dahl, and L. Vandenberghe, *CVXOPT: Python software for convex optimization*, 2020. [Online]. Available: http://cvxopt.org/index.html (visited on 07/25/2024).

[47] MOSEK ApS, *MOSEK optimizer API for Python. version 10.2.3*, 2023. [Online]. Available: https://docs.mosek.com/10.2/pythonapi/index.html (visited on 08/12/2024).

[48] O. Toker and H. Ozbay, "On the NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback", in *Proc. 1995 Am. Control Conf.*, vol. 4, IEEE, 1995, pp. 2525–2526. DOI: 10.1109/ACC.1995.532300.

[49] N. J. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA: SIAM, 2016, ISBN: 9781611974492.

[50] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control", *J. Appl. Dyn. Syst.*, vol. 17, no. 1, pp. 909–930, 2018. DOI: 10.1137/16m1062296.

[51] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data", *J. Fluid Mech.*, vol. 656, pp. 5–28, 2010. DOI: 10.1017/s0022112010001217.

[52] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust, "Applications of the dynamic mode decomposition", *Theor. Comput. Fluid Dyn.*, vol. 25, pp. 249–259, 2011. DOI: 10.1007/s00162-010-0203-9.

[53] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows", *J. Fluid Mech.*, vol. 641, pp. 115–127, 2009. DOI: 10.1017/s0022112009992059.

[54] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control", arXiv:1409.6358v1 [math.OC], 2014.

[55] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition", *J. Nonlinear Sci.*, vol. 25, no. 6, pp. 1307–1346, 2015. DOI: 10.1007/s00332-015-9258-5.

[56] A. N. Tikhonov, A. Goncharsky, V. V. Stepanov, and A. G. Yagola, *Numerical Methods for the Solution of Ill-Posed Problems*. Dordrecht, Netherlands: Springer, 1995, ISBN: 9789048145836.

[57] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY: Springer, 2013, ISBN: 9789048145836.

[58] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications", *J. Comput. Dyn.*, vol. 1, no. 2, pp. 391–421, 2014. DOI: 10.3934/jcd.2014.1.391.

[59] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, "Extending data-driven Koopman analysis to actuated systems", *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 704–709, 2016. DOI: 10.1016/j.ifacol.2016.10.248.

[60] F. Dietrich, T. N. Thiem, and I. G. Kevrekidis, "On the Koopman operator of algorithms", *SIAM J. Appl. Dyn. Syst.*, vol. 19, no. 2, pp. 860–885, 2020. DOI: 10.1137/19m1277059.

[61] V. Cibulka, T. Hanis, and M. Hromcik, "Data-driven identification of vehicle dynamics using Koopman operator", in *Proc. 22nd Int. Conf. Process Control*, IEEE, 2019. DOI: 10.1109/pc.2019.8815104.

[62] V. Eglājs and P. Audze, "New approach to the design of multifactor experiments", *Probl. Dyn. Strengths*, vol. 35, no. 1, pp. 104–107, 1977.

[63] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[64] Y. Susuki, K. Eto, N. Hiramatsu, and A. Ishigame, "Control of in-room temperature field via damping assignment to nonlinear Koopman mode", *IEEE Trans. Control Syst. Technol.*, vol. 32, no. 5, pp. 1569–1578, 2024. DOI: 10.1109/tcst.2023.3345149.

[65] M. Švec, Š. Ileš, and J. Matuško, "Predictive direct yaw moment control based on the Koopman operator", *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 6, pp. 2912–2919, 2023. DOI: 10.1109/tcst.2023.3269921.

[66] W. Rudin, *Fourier analysis on groups*. Wiley, 1991, ISBN: 9780471523642.

[67] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics", *IEEE Trans. Robot. Autom.*, vol. 6, no. 3, pp. 4369–4376, 2021. DOI: 10.1109/lra.2021.3068117.

[68] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning data-driven stable Koopman operators", arXiv:2005.04291v1 [cs.RO], 2020.

[69] J. Umenberger and I. R. Manchester, "Specialized interior-point algorithm for stable nonlinear system identification", *IEEE Trans. Autom. Control*, vol. 64, no. 6, pp. 2442–2456, 2019. DOI: 10.1109/TAC.2018.2867358.

[70] T. Chen, M. S. Andersen, L. Ljung, A. Chiuso, and G. Pillonetto, "System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques", *IEEE Trans. Autom. Control*, vol. 59, no. 11, pp. 2933–2945, 2014. DOI: 10.1109/TAC.2014.2351851.

[71] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification", *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 3, pp. 1235–1256, 2010. DOI: 10.1137/090755436.

[72] C. Feng, C. M. Lagoa, N. Ozay, and M. Sznaier, "Hybrid system identification: An SDP approach", in *Proc. 49th Conf. Decis. Control*, IEEE, 2010, pp. 1546–1552. DOI: 10.1109/CDC.2010.5718082.

[73] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution", *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 60–67, 2018. DOI: 10.1109/LRA.2017.2729659.

[74] J. Hasenauer, S. Waldherr, K. Wagner, and F. Allgöwer, "Parameter identification, experimental design and model falsification for biological network models using semidefinite programming", *IET Systems Biology*, vol. 4, 119–130(11), 2 2010. DOI: 10.1049/iet-syb.2009.0030.

[75] X. Gong, X. Wang, and B. Cao, "N data-driven modeling and control in modern power grids stability: Survey and perspective", *Appl. Energy*, vol. 350, p. 121 740, 2023. DOI: 10.1016/j.apenergy.2023.121740.

[76] K. Hara, M. Inoue, and N. Sebe, "Learning Koopman operator under dissipativity constraints", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1169–1174, 2020. DOI: 10.1016/j.ifacol.2020.12.1327.

[77] M. A. Mabrok, I. Aksikas, and N. Meskin, "Koopman operator approximation under negative imaginary constraints", *IEEE Control Syst. Lett.*, vol. 7, pp. 2767–2772, 2023. DOI: 10.1109/LCSYS.2023.3290195.

[78] K. Hara and M. Inoue, "Gain-preserving data-driven approximation of the Koopman operator and its application in robust controller design", *Mathematics*, vol. 9, no. 9, p. 949, 2021. DOI: 10.3390/math9090949.

[79] R. Strässer, J. Berberich, and F. Allgöwer, "Robust data-driven control for nonlinear systems using the Koopman operator", *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 2257–2262, 2023, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2023.10.1190.

[80] R. Strässer, M. Schaller, K. Worthmann, J. Berberich, and F. Allgöwer, "Koopman-based feedback design with stability guarantees", *IEEE Trans. Autom. Control*, pp. 1–16, 2024, ISSN: 2334-3303. DOI: 10.1109/tac.2024.3425770.

[81] T. He and A. Pal, "Dual-loop robust control of biased Koopman operator model by noisy data of nonlinear systems", arXiv:2401.08536v2[eess.SY], 2024.

[82] M. Eyüboğlu, N. Powell, and A. Karimi, "Data-driven control synthesis using Koopman operator: A robust approach", in *Proc. 2024 Am. Control Conf.*, Toronto, Canada: IEEE, 2024.

[83] M. Sznaier, "A convex optimization approach to learning Koopman operators", arXiv:2102.03934v1 [eess.SY], 2021.

[84] R. Tibshirani, "Regression shrinkage and selection via the lasso", *J. Roy. Statistical Soc.: Ser. B*, vol. 58, no. 1, pp. 267–288, 1996. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[85] S. Lacy and D. Bernstein, "Subspace identification with guaranteed stability using constrained optimization", *IEEE Trans. Autom. Control*, vol. 48, no. 7, pp. 1259–1263, 2003. DOI: 10.1109/tac.2003.814273.

[86] J. B. Hoagg, S. L. Lacy, R. S. Erwin, and D. S. Bernstein, "First-order-hold sampling of positive real systems and subspace identification of positive real models", in *Proc. 2004 Am. Control Conf.*, Boston, MA: IEEE, 2004. DOI: 10.23919/acc.2004.1383714.

[87] S. M. Siddiqi, B. Boots, and G. J. Gordon, "A constraint generation approach to learning stable linear dynamical systems", Defense Technical Information Center, Tech. Rep., 2008. DOI: 10.21236/ada480921.

[88] G. Mamakoukas, O. Xherija, and T. Murphey, "Memory-efficient learning of stable linear dynamical systems for prediction and control", *Advances Neural Inf. Process. Syst.*, vol. 33, pp. 13 527–13 538, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/9cd78264cf2cd821ba651485c111a29a-Abstract.html (visited on 08/12/2024).

[89] N. Gillis, M. Karow, and P. Sharma, "A note on approximating the nearest stable discrete-time descriptor systems with fixed rank", *Appl. Numer. Math.*, vol. 148, pp. 131–139, 2020. DOI: 10.1016/j.apnum.2019.09.004.

[90] L. El Ghaoui and S.-I. Niculescu, *Advances in Linear Matrix Inequality Methods in Control*. Philadelphia, PA: SIAM, 2000, ISBN: 0898714389.

[91] A. Doroudchi *et al.*, "Decentralized control of distributed actuation in a segmented soft robot arm", in *Proc. 57th Conf. Decis. Control*, Miami Beach, FL: IEEE, 2018. DOI: 10.1109/cdc.2018.8619036.

[92] R. Fortune, C. A. Beltempo, and J. R. Forbes, "System identification and feedforward control of a fatigue structural testing rig: The single actuator case", *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 382–387, 2019. DOI: 10.1016/j.ifacol.2019.11.273.

[93] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using Koopman operator theory", *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 948–961, 2021. DOI: 10.1109/tro.2020.3038693.

[94] L. Lortie, S. Dahdah, and J. R. Forbes, "Forward-backward extended DMD with an asymptotic stability constraint", arXiv:2403.10623v1[eess.SY], 2024.

[95] E. Warner and J. Scruggs, "Iterative convex overbounding algorithms for BMI optimization problems", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 449–10 455, 2017. DOI: 10.1016/j.ifacol.2017.08.1974.

[96] J. G. VanAntwerp and R. D. Braatz, "A tutorial on linear and bilinear matrix inequalities", *J. Process Control*, vol. 10, no. 4, pp. 363–385, 2000. DOI: 10.1016/s0959-1524(99)00056-6.

[97] U. Forssell and L. Ljung, "Closed-loop identification revisited", *Automatica*, vol. 35, no. 7, pp. 1215–1241, 1999. DOI: 10.1016/s0005-1098(99)00022-9.

[98] P. V. den Hof, "Closed-loop issues in system identification", *Annu. Rev. Control*, vol. 22, pp. 173–186, 1998. DOI: 10.1016/s1367-5788(98)00016-9.

[99] P. Van Overschee and B. De Moor, "Closed loop subspace system identification", in *Proc. 36th Conf. Decis. Control*, San Diego, California: IEEE, 1997.

[100] L. Ljung, *System Identification: Theory for the User*. Prentice Hall, 1999, ISBN: 9780136566953.

[101] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. West Sussex, England: Wiley, 2006, ISBN: 9780470011683.

[102] S. T. M. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley, "Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition", *Exp. Fluids*, vol. 57, no. 3, 2016. DOI: 10.1007/s00348-016-2127-7.

[103] G. Duffing, "Forced oscillations with variable natural frequency and their technical relevance", *Heft*, vol. 41, no. 42, pp. 1–134, 1918.

[104] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Chichester, UK: Wiley, 2013, ISBN: 9781119943594.

[105] G. Armenise, M. Vaccari, R. B. Di Capaci, and G. Pannocchia, "An open-source system identification package for multivariable processes", in *Proc. 12th UKACC Int. Conf. Control*, IEEE, 2018. DOI: 10.1109/control.2018.8516791.

[106] Quanser, *QUBE-Servo 2*, https://www.quanser.com/products/qube-servo-2/ (accessed 2024-01-30), 2023.

[107] S. Wright, "Correlation and causation", *J. Agricultural Res.*, vol. 20, pp. 557–585, 1921.

[108] M. Chilali, P. Gahinet, and P. Apkarian, "Robust pole placement in LMI regions", *IEEE Trans. Autom. Control*, vol. 44, no. 12, pp. 2257–2270, 1999. DOI: 10.1109/9.811208.

[109] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control", *PLOS One*, vol. 11, no. 2, 2016. DOI: 10.1371/journal.pone.0150171.

[110] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control", arXiv:1707.01146v2 [math.OC], 2018.

[111] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Koopman-based control of a soft continuum manipulator under variable loading conditions", arXiv:2002.01407v1 [cs.RO], 2020.

[112] F. Ganz, A. Datar, P. Gottsch, and H. Werner, "Data-driven $\mathcal{H}_\infty$ optimal controller design using the Koopman operator: Case study", in *Proc. 2021 Eur. Control Conf.*, IEEE, 2021. DOI: 10.23919/ecc54610.2021.9654883.

[113] G. Mamakoukas, S. Di Cairano, and A. P. Vinod, "Robust model predictive control with data-driven Koopman operators", in *Proc. 2022 Am. Control Conf.*, IEEE, 2022. DOI: 10.23919/acc53348.2022.9867811.

[114] X. Zhang, W. Pan, R. Scattolini, S. Yu, and X. Xu, "Robust tube-based model predictive control with Koopman operators", *Automatica*, vol. 137, p. 110 114, 2022, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2021.110114.

[115] T. Gholaminejad and A. Khaki-Sedigh, "Stable data-driven Koopman predictive control: Concentrated solar collector field case study", *IET Control Theory Appl.*, vol. 17, no. 9, pp. 1116–1131, 2023, ISSN: 1751-8652. DOI: 10.1049/cth2.12442.

[116] T. de Jong, V. Breschi, M. Schoukens, and M. Lazar, "Koopman data-driven predictive control with robust stability and recursive feasibility guarantees", arXiv:2405.01292v1[math.OC], 2024.

[117] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems", in *Proc. 55th Conf. Decis. Control*, IEEE, 2016. DOI: 10.1109/cdc.2016.7799268.

[118] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using Koopman operator framework", *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016. DOI: 10.1016/j.ifacol.2016.10.250.

[119] D. F. Gomez, F. D. Lagor, P. B. Kirk, A. H. Lind, A. R. Jones, and D. A. Paley, "Data-driven estimation of the unsteady flowfield near an actuated airfoil", *J. Guid. Control Dyn.*, vol. 42, no. 10, pp. 2279–2287, 2019. DOI: 10.2514/1.g004339.

[120] M. Netto and L. Mili, "A robust data-driven Koopman Kalman filter for power systems dynamic state estimation", *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 7228–7237, 2018. DOI: 10.1109/tpwrs.2018.2846744.

[121] J. D. Lee, S. Im, L. Kim, H. Ahn, and H. Bang, "Data-driven fault detection and isolation for multirotor system using Koopman operator", *J. Intell. Robot. Syst.*, vol. 110, no. 3, 2024. DOI: 10.1007/s10846-024-02142-y.

[122] A. Lambe and S. N. Sharma, "On embedding the Koopmanization into controlled nonlinear systems, its comparison with the Carleman linearisation and concerning results: Beyond the feedback linearisation", *Int. J. Syst. Sci.*, pp. 1–21, 2024. DOI: 10.1080/00207721.2024.2370313.

[123] Z. C. Guo, F. Dümbgen, J. R. Forbes, and T. D. Barfoot, "Data-driven batch localization and SLAM using Koopman linearization", *IEEE Trans. Robot.*, vol. 40, pp. 3964–3983, 2024. DOI: 10.1109/tro.2024.3443674.

[124] Z. Wang and H. Unbehauen, "A class of nonlinear observers for discrete-time systems with parametric uncertainty", *Int. J. Syst. Sci.*, vol. 31, no. 1, pp. 19–26, 2000. DOI: 10.1080/002077200291415.

[125] H. Marquez and M. Riaz, "Robust state observer design with application to an industrial boiler system", *Control Eng. Pract.*, vol. 13, no. 6, pp. 713–728, 2005. DOI: 10.1016/j.conengprac.2004.06.004.

[126] L. Etienne, K. A. Langueh, H. Karkaba, and A. Iovine, "Robust observer synthesis for bilinear parameter varying system", in *Proc. 61st Conf. Decis. Control*, IEEE, 2022, pp. 1900–1905. DOI: 10.1109/cdc51059.2022.9992459.

[127] G. Lu and D. Ho, "Robust $\mathcal{H}_\infty$ observer for nonlinear discrete systems with time delay and parameter uncertainties", *IEE Proc. — Control Theory Appl.*, vol. 151, no. 4, pp. 439–444, 2004. DOI: 10.1049/ip-cta:20040490.

[128] A. M. Pertew, H. J. Marquez, and Q. Zhao, "$\mathcal{H}_\infty$ synthesis of unknown input observers for non-linear Lipschitz systems", *Int. J. Control*, vol. 78, no. 15, pp. 1155–1165, 2005. DOI: 10.1080/00207170500155488.

[129] M. Abbaszadeh and H. J. Marquez, "Robust $\mathcal{H}_\infty$ observer design for sampled-data Lipschitz nonlinear systems with exact and Euler approximate models", *Automatica*, vol. 44, no. 3, pp. 799–806, 2008. DOI: 10.1016/j.automatica.2007.07.021.

[130] M. Abbaszadeh and H. J. Marquez, "LMI optimization approach to robust $\mathcal{H}_\infty$ observer design and static output feedback stabilization for discrete-time nonlinear uncertain systems", *Int. J. Robust Nonlinear Control*, vol. 19, no. 3, pp. 313–340, 2008. DOI: 10.1002/rnc.1310.

[131] R. Raoufi, H. J. Marquez, and A. S. I. Zinober, "$\mathcal{H}_\infty$ sliding mode observers for uncertain nonlinear Lipschitz systems with fault estimation synthesis", *Int. J. Robust Nonlinear Control*, vol. 20, no. 16, pp. 1785–1801, 2010. DOI: 10.1002/rnc.1545.

[132] T. Gebru *et al.*, "Datasheets for datasets", *Commun. ACM*, vol. 64, no. 12, pp. 86–92, 2021. DOI: 10.1145/3458723.

[133] P. Khargonekar and M. Rotea, "Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control: A convex optimization approach", *IEEE Trans. Autom. Control*, vol. 36, no. 7, pp. 824–837, 1991. DOI: 10.1109/9.85062.

[134] I. Kaminer, P. P. Khargonekar, and M. A. Rotea, "Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control for discrete-time systems via convex optimization", *Automatica*, vol. 29, no. 1, pp. 57–70, 1993. DOI: 10.1016/0005-1098(93)90174-r.

[135] P. P. Khargonekar, M. A. Rotea, and E. Baeyens, "Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ filtering", *Int. J. Robust Nonlinear Control*, vol. 6, no. 4, pp. 313–330, 1996. DOI: 10.1002/(sici)1099-1239(199605)6:4<313::aid-rnc235>3.0.co;2-8.

[136] H. Marquez, "A frequency domain approach to state estimation", *J. Frankl. Inst.*, vol. 340, no. 2, pp. 147–157, 2003. DOI: 10.1016/s0016-0032(03)00017-6.

[137] T. Tuttle and W. Seering, "A nonlinear model of a Harmonic Drive gear transmission", *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 368–374, 1996. DOI: 10.1109/70.499819.

[138] F. H. Ghorbel, P. S. Gandhi, and F. Alpeter, "On the kinematic error in Harmonic Drive gears", *J. Mech. Des.*, vol. 123, no. 1, pp. 90–97, 1998. DOI: 10.1115/1.1334379.

[139] S. Dahdah and J. R. Forbes, "Closed-loop Koopman operator approximation", arXiv:2303.15318v1 [eess.SY], 2023.

[140] K. Mardia and P. E. Jupp, *Directional Statistics*. Wiley, 1999, ISBN: 9780471953333.

[141] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems", in *Proc. 2019 Am. Control Conf.*, IEEE, 2019. DOI: 10.23919/acc.2019.8815339.

[142] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python", *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (visited on 08/12/2024).

[143] S. Pan, E. Kaiser, B. M. de Silva, J. N. Kutz, and S. L. Brunton, "PyKoopman: A Python package for data-driven approximation of the Koopman operator", arXiv:2306.12962v1[eess.SY], 2023.

[144] S. Dey and E. W. Davis, "DLKoopman: A deep learning software package for Koopman theory", in *Proc. Mach. Learn. Res.*, vol. 211, PMLR, 2023, pp. 1467–1479.

[145] P. Novelli and V. Kostic, *Machine-Learning-Dynamical-Systems/kooplearn v1.1.0*, 2023. [Online]. Available: https://github.com/Machine-Learning-Dynamical-Systems/kooplearn (visited on 08/06/2024).

[146] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2005, ISBN: 9780321245625.

[147] "Array programming with NumPy", *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2.

[148] M. Feurer and F. Hutter, "Hyperparameter optimization", *Autom. Mach. Learn.: Methods Syst. Chall.*, pp. 3–33, 2019. DOI: 10.1007/978-3-030-05318-5_1.

[149] B. Van der Pol, "On 'relaxation-oscillations'", *Lond. Edinb. Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 978–992, 1926.

[150] S. Pan, N. Arnold-Medabalimi, and K. Duraisamy, "Sparsity-promoting algorithms for the discovery of informative Koopman-invariant subspaces", *J. Fluid Mech.*, vol. 917, 2021. DOI: 10.1017/jfm.2021.271.