Learning Stable Koopman Models for Identification and Control of Dynamical Systems

Fletcher Fan BE (Hons 1)

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

Submitted December 2022; revised August 2023

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Fletcher Fan

6 August 2023

Abstract

Learning models of dynamical systems from data is a widely-studied problem in control theory and machine learning. One recent approach for modelling nonlinear systems considers the class of Koopman models, which embeds the nonlinear dynamics in a higher-dimensional *linear* subspace. Learning a Koopman embedding would allow for the analysis and control of nonlinear systems using tools from linear systems theory. Many recent methods have been proposed for data-driven learning of such Koopman embeddings, but most of these methods do not consider the stability of the Koopman model.

Stability is an important and desirable property for models of dynamical systems. Unstable models tend to be non-robust to input perturbations and can produce unbounded outputs, which are both undesirable when the model is used for prediction and control. In addition, recent work has shown that stability guarantees may act as a regularizer for model fitting. As such, a natural direction would be to construct Koopman models with inherent stability guarantees.

Two new classes of Koopman models are proposed that bridge the gap between Koopman-based methods and learning stable nonlinear models. The first model class is guaranteed to be stable, while the second is guaranteed to be stabilizable with an explicit stabilizing controller that renders the model stable in closed-loop. Furthermore, these models are unconstrained in their parameter sets, thereby enabling efficient optimization via gradient-based methods. Theoretical connections between the stability of Koopman models and forms of nonlinear stability such as contraction are established. To demonstrate the effect of the stability guarantees, the stable Koopman model is applied to a system identification problem, while the stabilizable model is applied to an imitation learning problem. Experimental results show empirically that the proposed models achieve better performance over prior methods without stability guarantees.

Acknowledgements

First and foremost, thanks to David for your unwavering support and reliable advice. There were many challenges throughout my research and they wouldn't have been overcome without your constant support and encouragement.

Thanks also to Ian, Guodong and Bowen. This thesis wouldn't have been possible without all of your technical input and research advice. I've learnt a lot from all of you during all the fruitful and insightful discussions in our meetings.

Last but not least, thanks to everyone at ACFR and beyond who were there during this long journey. In no particular order, Felix, Jack, Jacob, James, Jasper, Jen, Johnny, Max, Tara, Vera, Wei, thanks to all of you for all the good times.

'Real stupidity beats artificial intelligence every time.'
- Terry Pratchett, Hogfather

Contents

Declaration **i**

Abstract **ii**

Acknowledgements **iii**

Contents **v**

List of Figures **ix**

Nomenclature **x**

1 Introduction **1**

1.1 Motivation 2

1.2 Problem Statement 3

1.3 Contributions 4

1.3.1 Publications 5

1.4 Structure of the Thesis 5

2 Background **6**

2.1 Learning from Data 6

2.1.1 Regularization 8

2.1.2 Numerical Optimization 9

2.1.3 Model Classes 14

2.1.4 Direct Parameterizations 16

2.2 Stability of Dynamical Systems 16
2.2.1 Stability of Linear Systems 18
2.2.2 Contraction 20
2.2.3 Control Contraction Metrics 21
2.3 Koopman Theory 23
2.3.1 The Koopman Operator 23
2.3.2 Equivalence of Contraction and Koopman Stability 26
2.3.3 Dynamic Mode Decomposition 29
2.3.4 Koopman Operator with Control 30
2.4 Summary 32

3 Related Work 33
3.1 Learning Koopman Models 33
3.1.1 Stable Koopman Models 35
3.1.2 Koopman Models for Control 35
3.2 Imitation Learning 36
3.2.1 Behavioural Cloning 37
3.2.2 Inverse Reinforcement Learning 39
3.2.3 Stable Imitation Learning 40
3.3 Summary 41

4 Stable Koopman Models 42
4.1 Introduction 42
4.2 Stability Criterion for Discrete-time Koopman Models 45
4.3 Model Set 49
4.3.1 Parametrization of Koopman Matrix 49
4.3.2 Parametrization of Koopman Embedding 51
4.3.3 Overall Koopman Model 52
4.4 Learning Framework 52

4.4.1 Optimization Formulation 52
4.4.2 Implementation Details 54
4.5 Continuous-time Case 55
4.5.1 Continuous-time Koopman Model 55
4.5.2 Learning Framework 56
4.6 Experiments 57
4.6.1 Comparison to Other Koopman Matrix Parameterizations . . 58
4.6.2 Robustness to Perturbations in Initial Conditions 59
4.7 Summary 60

5 Imitation Learning with Koopman Models 64
5.1 Problem Statement 65
5.2 Stabilizable Koopman Models 67
5.2.1 Koopman Stabilizability 68
5.3 Model Set 70
5.3.1 Parameterization of Stabilizable Triples 71
5.3.2 Parameterization of ϕ 72
5.3.3 Parameterization of α 72
5.3.4 CCM Controller 73
5.3.5 Connection to Inverse Optimal Control 74
5.4 Learning Framework 75
5.4.1 Linear Case 76
5.5 Experiments 77
5.5.1 Linear Example 77
5.5.2 Nonlinear Example 83
5.6 Summary 89

6 Conclusion 92
6.1 Summary 92
6.2 Future Work 93

List of References **94**

A Additional Imitation Learning Trajectories **106**

List of Figures

4.1 Koopman model 45
4.2 NSE comparison 60
4.3 Training loss boxplot 61
4.4 SKEL simulations 62
4.5 LKIS simulation 63

5.1 Eigenvalue histogram 80
5.2 NSE boxplot 81
5.3 Controller fit boxplot 82
5.4 Time per iteration 83
5.5 Total time to convergence 84
5.6 Diagram of robot arm 85
5.7 NSE of imitation controllers 87
5.8 Training error of controller output 88
5.9 Test error of controller output 89
5.10 Trajectories induced by learned CCM controller 90
5.11 Trajectories induced by behavioural cloning controller 91

Nomenclature

List of Symbols

$\mathbb{R}$	The set of real numbers
$\mathbb{R}^n$	The set of real-valued n-dimensional vectors
$\mathbb{R}^{n\times m}$	The set of real-valued $n \times m$ matrices
$A \succ 0$	The matrix A is positive-definite
$A \succeq 0$	The matrix A is positive-semidefinite
$A \prec 0$	The matrix A is negative-definite
$A \preceq 0$	The matrix A is negative-semidefinite
$A \succ B$	The matrix $A - B$ is positive-definite
$\|x\|_p$	The p-norm of the vector x
$\|M\|_p$	The matrix p-norm of the matrix M
$M^\dagger$	The pseudoinverse of the matrix M

List of Acronyms

ADMM	alternating direction method of multipliers
BC	behavioural cloning
CCM	control contraction metric
CT	continuous-time
CL	closed-loop
DMD	dynamic mode decomposition
DOF	degree(s) of freedom
DT	discrete-time
EDMD	extended dynamic mode decomposition
FCNN	fully-connected neural network
GAIL	generative adversarial imitation learning
IL	imitation learning
IRL	inverse reinforcement learning
LMI	linear matrix inequality
LPV	linear parameter-varying
LTI	linear time-invariant

LTV	linear time-varying
MLP	multilayer perceptron
NMPC	nonlinear model predictive control
NN	neural network
NSE	normalized simulation error
ODE	ordinary differential equation
OL	open-loop
PGD	projected gradient descent
ReLU	rectified linear unit
SVD	singular value decomposition
SOS	sum-of-squares
UES	universally exponentially stabilizable

Chapter 1

Introduction

Modelling dynamical systems is a ubiquitous problem across many domains in engineering and science. Many important physical phenomena can be modelled as a dynamical system to predict future behaviour, and many actuated systems can benefit from a model of the dynamics of the system for control design. Deriving a model of a dynamical system from first principles, such as physical laws, may be challenging or even intractable for cases like human behaviour. This is where learning approaches that produce a model from data are useful.

A central consideration for a learning algorithm is the structure of the model to be learned. For modelling memoryless input-output mappings, deep neural networks have achieved state-of-the-art results in many problem domains, including image classification (Krizhevsky et al., 2012), playing strategy games such as Go (Silver et al., 2016) and robot control (Levine et al., 2016). However, choosing a suitable model structure to model a dynamical system remains an open question.

In this work, the focus will be on Koopman models, a recently emerging class of models that is both flexible and interpretable.

1.1 Motivation

Koopman models are based on Koopman operator theory (Koopman, 1931), which provides for the existence of an infinite-dimensional linear operator that describes the dynamics of any nonlinear system. Through Koopman theory, nonlinear systems can be studied via a spectral decomposition of the Koopman operator (Mezić, 2005), akin to linear systems analysis. When learning a Koopman model, one attempts to find a finite-dimensional representation of the Koopman operator, which amounts to a linear matrix, along with a mapping that transforms the original state space of the system to a so-called Koopman-invariant subspace on which the dynamics of the system become linear. Koopman models hold the promise of linearising a nonlinear system *globally*. This has huge potential in applying tools from linear systems theory to nonlinear systems, including global stability analysis (Mauroy and Mezić, 2016; Yi and Manchester, 2022) and linear control design methods such as the linear quadratic regulator (Bevanda et al., 2022a).

An important consideration for models of dynamical systems is stability, which has mostly been neglected in prior work on Koopman models. Unstable models tend to be non-robust to input perturbations and can produce unbounded outputs, which are both undesirable when the model is used for prediction and control. In addition, there has been a significant amount of recent work on learning stable nonlinear models (Manek and Kolter, 2019; Neumann et al., 2013; Sindhwani et al., 2018; Singh et al., 2021), where stability was used as a control-theoretic regularizer for model learning. Note that the scope of the present work does not extend to considering Koopman models in the context of general nonlinear models. Koopman models are not necessarily superior in terms of numerical performance to other nonlinear models, but have qualitative benefits like ease of control design and stability analysis as mentioned above.

1.2 Problem Statement

The problems considered in this work are posed as questions that will be answered using mathematical tools and simulated experimental results. The problems can be divided into two categories: system identification and control.

For system identification problems, one is concerned with finding a model of the system that accurately predicts its outputs. Some pertinent questions when identifying a system using a Koopman model are:

- How can a Koopman model be parameterized such that it is guaranteed to be stable?
- What does stability of the Koopman model imply about the stability of the original nonlinear system being modelled?
- How can the model be optimized for a given dataset to identify the dynamics of the underlying system?
- Does the stability guarantee improve the performance of the model for identification problems?

In control problems, one is concerned with designing a controller that stabilizes a given system. In particular, imitation learning is concerned with learning a controller that replicates behaviour from a set of demonstrations. To apply Koopman model learning to the imitation learning problem, some questions to be addressed are:

- How can the Koopman model be extended to incorporate control inputs?
- How can the controlled model be guaranteed to be stabilizable?
- How can the controller be optimized in an imitation learning framework?
- Does guaranteeing stabilizability of the model improve the performance of the controller for imitation learning?

1.3 Contributions

The main contributions of each chapter are stated in the following.

In Chapter 4:

1. A new model class is proposed for representing stable dynamical systems. Importantly, this model class is unconstrained in its parameters, allowing for efficient optimization by leveraging software tools for automatic differentiation.

2. A theoretical equivalence between stability of the Koopman model and contraction is proven, extending a result of Yi and Manchester (2022) to discrete-time systems.

3. A learning framework is proposed to fit this model parameterization to data for a system identification problem.

4. Numerical experiments on a real-world handwriting dataset show empirically that the stable model parameterization improves performance compared to models without stability guarantees, while not significantly increasing the computational burden.

In Chapter 5:

1. A new model class is proposed for imitation learning. This model class comprises of a open-loop dynamics model that is guaranteed to be stabilizable, and a stabilizing controller that is guaranteed to induce a stable closed-loop model. Similar to the stable Koopman model proposed in Chapter 5, this model is also unconstrained in its parameters.

2. A theoretical equivalence between the stabilizability of the Koopman model and the existence of a control contraction metric is proven.

3. A new imitation learning framework is proposed for the new model class. Unlike many imitation learning algorithms, this framework only requires snapshots of state transitions and control inputs as data.

4. Numerical experimental results on a simulated linear system demonstrate the proposed model outperforms a prior stability-constrained imitation learning algorithm as well as a baseline least-squares approach.

5. Further numerical experiments on a handwriting imitation problem with a simulated robotic manipulator show that the learned controller outperforms a baseline imitation learning method.

1.3.1 Publications

Some of the work presented in this thesis was previously published in:

- Fletcher Fan, Bowen Yi, David Rye, Guodong Shi, and Ian R. Manchester. Learning stable Koopman embeddings. In *2022 American Control Conference (ACC 2022)*, pages 2742—2747, 2022.

1.4 Structure of the Thesis

The rest of the thesis is structured as follows. In Chapter 2, the mathematical background on learning from data, stability of dynamical systems and Koopman theory is presented. Chapter 3 reviews related work on learning Koopman models and imitation learning, in particular works that consider stability constraints. In Chapter 4, the first new parameterization of Koopman models is presented, along with theoretical and experimental results. In Chapter 5, the second Koopman model class is presented and applied to the problem of imitation learning. Finally, Chapter 6 provides a summary and concluding remarks, along with suggestions for future directions of research.

Chapter 2

Background

In this chapter, the theoretical machinery that the rest of the thesis builds upon is presented. The aim of this chapter is to define the mathematical tools required to establish the arguments of the thesis. Review and discussion of prior works that consider similar problems are deferred to Chapter 3.

To begin, the problem of statistical learning is discussed, which broadly covers the problems that will be considered in Chapters 4 and 5. The focus is to present the canonical optimization problems, together with their solution methods and solution sets. In Section 2.2, the concept of stability is then defined for both linear and nonlinear dynamical systems, which will enable establishing of theoretical properties of the model classes that are proposed. Finally in Section 2.3, the fundamentals of Koopman theory, which forms the theoretical basis of the proposed models, are briefly presented,

2.1 Learning from Data

The problems considered in this work can be broadly described as learning models from data, and specifically as learning models of dynamical systems given sequences of measurements of those systems. Such problems have been widely studied across

many fields and have a long history of research, with perhaps the most famous early success being the method of least squares dating back to the 19th century. More recently, fields such as machine learning, deep learning (Goodfellow et al., 2016) and system identification (Ljung, 1998) all consider the model learning problem, with differences in the assumptions made about the data and model class.

The learning problem can be broken down into three main components:

1. a set of pairs of input-output data $\{x_i, y_i\}_{i=1}^N$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ can be arbitrary data types including signals, sequences, categorical data and images,
2. a loss function $l : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that measures the discrepancy between the observed and predicted outputs, and
3. a model $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ parameterized by a set of parameters θ belonging to the parameter set Θ (often called the feasible set). The dependence of f on θ is usually denoted by subscripting f_θ while only x is treated as an input to the model.

The learning problem can be written succintly as:

$$\theta^\star = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i). \tag{2.1}$$

Roughly speaking, the aim of learning a model is to predict output values from input values, given a set of input-output data. A model is said to be accurate if its output predictions match the data. Often, the data are split into a training set and a test set. The training set, as the name implies, are the data that the model is trained on, while the test set is used to evaluate the model after training. Evaluating the model on inputs unseen during training reveals the *generalizability* of the model. If the model is able to make accurate predictions on the training inputs but not the test inputs, then the model is said to *overfit* to the training set.

A fundamental assumption made when learning a model is that there is an underlying but unknown relationship between the observed inputs and outputs in the dataset.

This assumption is often represented as a joint probability distribution $p(x, y)$ of the inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$ [1]. In order for the learned model to make accurate predictions, it is important that the model is able to represent this input-output relationship: that is, the model set contains the true model. One way to ensure this is to use a model that is sufficiently expressive or complex to represent a wide range of functions.

The expressivity of the model is a key factor in its generalizability and accuracy. These two criteria usually cannot be achieved simultaneously, and this is commonly known as the bias-variance trade-off (Hastie et al., 2009). A model is said to have high variance if it overfits to the training set, while high bias refers to the model not achieving sufficient accuracy on the training set (underfitting).

2.1.1 Regularization

A general method for reducing the variance of models is known as regularization (Hastie et al., 2009). The main mechanism of regularization is to reduce the complexity of the model, either explicitly by shaping the solution of Problem (2.1) via constraints and additional losses, or implicitly by modifying the optimization procedure. Reducing model variance through regularization usually comes at the cost of increasing the bias of the model.

Common explicit regularization methods include:

- $\mathcal{L}_2$ regularization (sometimes called ridge regression) (Hastie et al., 2009), which adds a term of the form $\|\theta\|_2^2$, where $\|\cdot\|_2^2$ is the squared $\mathcal{L}_2$ norm. This encourages the model parameters to be small, and thus increases the smoothness of the model.

- $\mathcal{L}_1$ regularization (sometimes called lasso regression) (Hastie et al., 2009), which adds a term of the form $\|\theta\|_1$, where $\|\cdot\|_1$ is the $\mathcal{L}_1$ norm, which is simply the

[1] In this case, Problem (2.1) can be interpreted as a sampled estimate of $\mathbb{E}_{x,y\sim p(x,y)}\left[l(f(x), y)\right]$, the expectation of the loss over the joint distribution. This problem is known as empirical risk minimization in statistical learning theory (Vapnik, 1991).

absolute value of the elements of θ if they are real numbers. This encourages sparsity of the model parameters.

One can also regularize model learning by constraining the space of solutions that are optimized over. This can be effective when one has prior knowledge of the true model. By encoding this prior in the optimization problem, models that are known to be 'bad' can be avoided during optimization. However, introducing constraints often comes at the cost of making the optimization problem more difficult to solve. One particularly important constraint for models of dynamical systems is stability, which forms a cornerstone of the thesis and will be discussed in Section 2.2.

Another way to regularize the learning problem is by using heuristics to modify the optimization process itself. These implicit regularization techniques include early stopping, gradient clipping (Pascanu et al., 2013), and dropout (Srivastava et al., 2014). These are widely used in deep learning for training large models with up to millions of parameters.

2.1.2 Numerical Optimization

Given an optimization problem (2.1), there exist many numerical algorithms for finding a solution to it. The appropriate choice of algorithm depends on the loss function and the parameter set Θ. For a smooth and continuously-differentiable nonlinear loss function, gradient-based methods, also known as gradient descent, are commonly used. Furthermore, if the loss function is convex, then gradient-based optimization exhibits linear convergence to the global minimum (Boyd and Vandenberghe, 2004). However, even for nonconvex loss functions, gradient methods can still find a locally optimal solution. Additionally, with the recent proliferation of automatic differentation software packages, gradient methods are easy to implement and use on any problem with a differentiable loss function.

At their core, gradient-based methods iteratively update the parameter vector θ via the update rule:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta l(\theta_k, X, Y), \tag{2.2}$$

where k denotes the iteration, α is the step size or *learning rate*, and the gradient of the loss function $\nabla_\theta l$ is computed over the entire set of training data X and Y. The gradient indicates the direction that the parameters should be adjusted to reduce the loss. When only the gradient, or first-order derivative, of the loss is used for the descent direction, it is called a first-order method.

Second-order methods that also estimate the second-order derivative, or the Hessian, can be used when the loss is locally well-approximated by a quadratic function at its optimum. These methods, such as Newton's method and the BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm (Fletcher, 1987), tend to converge in fewer iterations than first-order methods, at the cost of increased computational complexity and memory usage. For the large neural network models that are used in this work, first-order methods are preferred for computational efficiency.

When the parameter set Θ is the space of real numbers, i.e. $\Theta = \mathbb{R}^M$, then the model set is said to be *unconstrained*. In this case, the optimization problem is immediately amenable to being solved by gradient-based methods. When the parameter set is instead constrained to a subset of the real number space, i.e. $\Theta \subset \mathbb{R}^M$, then in order to solve the optimization via gradient methods, one has to either convert the constrained problem to an unconstrained one, or project the parameters onto the feasible set Θ after every update step. In the following, variants of these gradient methods will be discussed.

Extensions of Gradient Methods

There have been many extensions proposed to the update rule (2.2) to improve the performance of the optimizer. One significant development is stochastic gradient descent (SGD) (Bottou, 2004), which computes the gradient over batches of the data rather than over the entire dataset. This significantly reduces the memory requirements of the optimizer at the small cost of increasing the variance of the gradient estimates. In deep learning, batching the gradient computations has enabled tractable training of models on extremely large datasets (Bottou, 2010).

Recently, there has been a proliferation of software tools (e.g. PyTorch[2], Tensorflow[3]) that perform automatic differentiation (Griewank and Walther, 2008), which has enabled gradient-based methods to be easily implemented for optimizing arbitrary differentiable loss functions and differentiable models.

The optimizer used for training all of the models in the numerical experiments reported here is the Adam optimizer proposed by Kingma and Ba (2014), described in Algorithm 2.1. It uses an adaptive learning rate based on estimates of the first and second moments of the gradient. It has been shown to be effective at optimizing large-scale learning problems involving high-dimensional neural networks, and is relatively robust to hyperparameter choice.

When the model set is constrained, one can still optimize the constrained problem using gradient descent via a modification to the update rule in Equation (2.2):

$$\theta_{k+1} = \mathcal{P}(\theta_k - \alpha \nabla_\theta l(\theta_k, X, Y)), \tag{2.3}$$

where $\mathcal{P}$ is a projection operator that finds the closest θ on the feasible set Θ in the descent direction $\nabla_\theta l$. This is known as projected gradient descent (PGD). Depending on the shape of the feasible set, evaluating the projection operator may be intractable. For convex feasible sets, projection would usually involve solving a semidefinite program (SDP), which may only be tractable for moderately-sized problems. Compared to an unconstrained problem, solving a constrained problem using PGD is more computationally expensive and may be more susceptible to local minima.

Penalty Methods

Another approach to solving a constrained optimization problem is to replace the constraints with additional terms in the loss function that penalize violations of the constraints. One example is the quadratic penalty method, where the penalty terms are positive when the constraint is violated and zero otherwise.

[2]https://pytorch.org/
[3]https://www.tensorflow.org/

Algorithm 2.1: Adam optimizer (Kingma and Ba, 2014). Nominal values of hyperparameters used were $\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Input: α: Learning rate
Input: $\beta_1, \beta_2 \in [0, 1)$: Decay rates for moment estimates
Input: $l(\theta)$: Loss function with parameters θ
Input: θ_0: Initial parameter vector

```
m_0 ← 0                                  // Initialize first moment vector
v_0 ← 0                                  // Initialize second moment vector
k ← 0                                    // Initialize iteration counter
while θ_k not converged do
    k ← k + 1
    g_k ← ∇_θ l_k(θ_k)                   // Compute gradient w.r.t. loss
    m_k ← β_1 m_{k-1} + (1 − β_1) g_k    // Update first moment estimate
    v_k ← β_2 v_{k-1} + (1 − β_2) g_k^2  // Update second moment estimate
    m̂_k ← m_k / (1 − β_1^k)              // Correct for bias
    v̂_k ← v_k / (1 − β_2^k)              // Correct for bias
    θ_k ← θ_{k-1} − α m̂_k / (√v̂_k + ε)   // Update parameters
return θ_k
```

For example, for a problem of the form:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} l(\theta) \quad \text{subject to } c_e(\theta) = 0,\ c_i(\theta) \geq 0, \tag{2.4}$$

with one equality constraint and one inequality constraint, the quadratic penalty loss becomes:

$$l_Q(\theta) = l(\theta) + \mu_e c_e^2(\theta) + \mu_i (\max(-c_i(\theta), 0))^2, \tag{2.5}$$

where μ_e and μ_i are the penalty coefficients. At each iteration, an approximate minimizer of $l_Q(\theta)$ is found and the penalty coefficients are increased for the next iteration, until convergence. It has been shown that the optimal solution of (2.4) is attained in the limit as the coefficients approach infinity, if the exact minimizer of $l_Q(\theta)$ is found at each iteration (see (Nocedal and Wright, 1999, Chapter 17)). These are strong conditions that are not attainable in general, and even convergence is not guaranteed. Despite this, the quadratic penalty method is still widely used when only an approximate solution is required, due to its ease of implementation. In fact, it can be implemented to perform just one iteration of minimizing l_Q with large values for μ, for the sake of computational speed at the cost of the accuracy of the solution.

More sophisticated methods can be used to alleviate the disadvantages of the quadratic penalty method. One example is the augmented Lagrangian method (Hestenes, 1969), which introduces additional terms to explicitly estimate the Lagrange multiplier. Compared to the quadratic penalty method, it better preserves the smoothness of the problem and can achieve convergence without having the penalty coefficients approach infinity.

Often it may be desirable for all of the solutions at every iteration to be feasible with respect to the constraints. The penalty methods described above do not possess this property as constraints may be violated when only an approximate minimizer is found. Interior point methods address this issue through the use of barrier functions, which become infinite when constraints are violated. A backstepping line search algorithm can then be used to find the nearest feasible solution along the search direction. However, line searches can lead to poor convergence when the solution is

near the boundary of the feasible set (see for example (Nocedal and Wright, 1999, Chapter 19)).

2.1.3 Model Classes

A model class defines the space of models that are searched over during optimization, in contrast to the feasible set which defines the space of parameters of the model. In the following, the main model classes that are considered in this thesis are presented.

Neural Networks

Artificial neural networks have recently become the model class of choice for many machine learning applications, from image classification (Krizhevsky et al., 2012) to natural language processing (Brown et al., 2020). Their popularity has been spurred on by developments in computing power and availability of large datasets in these problem domains. On the theoretical side, neural networks have provable error bounds for approximating any continuous function by the universal approximation theorem (Hornik et al., 1989).

A neural network is composed of 'neurons' which contain an affine transformation of the input followed by a nonlinear *activation function*. A standard fully-connected neural network (FCNN), also known as a multilayer perceptron (MLP), can be written as:

$$\begin{aligned} x_{i+1} &= \sigma(W_i x_i + b_i), \quad i = 1, 2, \ldots, L-2, \\ x_L &= W_{L-1} x_{L-1} + b_{L-1}, \end{aligned} \tag{2.6}$$

where x_1 is the input to the network and x_L is the output. The network has L layers of neurons, with the last layer being an affine function. The parameters of the neural network are the weight matrices $W_i \in \mathbb{R}^{n \times n}$ and bias vectors $b_i \in \mathbb{R}^{n \times 1}$.

Common choices for the activation function $\sigma(x)$ include

- the rectified linear unit (ReLU): $\max(x, 0)$,

- hyperbolic tangent function: $(e^x - e^{-x})/(e^x + e^{-x})$, and
- sigmoid function: $1/(1 + e^{-x})$.

More sophisticated layer and network structures have been developed to suit specific problems. These include convolutional neural networks (CNNs) for modelling spatial relationships in the data, commonly used in image classification problems (Krizhevsky et al., 2012), or recurrent neural networks (RNNs) which are often used for modelling sequence-to-sequence mappings (Lipton et al., 2015). In the present work, the functions to be modelled are all memoryless mappings, hence only FCNNs are used.

State Space Models

State space models are widely used to describe the behaviour of dynamical systems. They can be written in the form:

$$\begin{aligned} x_{t+1} &= f_\theta(x_t, u_t), \\ y_t &= g_\theta(x_t, u_t), \end{aligned} \tag{2.7}$$

where $x_t \in \mathbb{R}^n$ is the internal state of the model, and $u_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^{n_y}$ are the model input and output respectively. The two parameterized functions of the model are the dynamics function $f_\theta : \mathbb{R}^{n \times m} \to \mathbb{R}^n$, and the output function $g_\theta : \mathbb{R}^{n \times m} \to \mathbb{R}^{n_y}$. A more general model can also have the timestep t as an input, making it a time-varying model, however in this work, only time-invariant state space models are considered.

When the model has no external input u, it is referred to as an *autonomous* model. In this case, the only 'input' to the model is the initial condition of the state x.

State space models are a very general class of model that can represent many subsets of models depending on the parameterizations of f_θ and g_θ. For example, if f_θ is set to zero—that is, the model has no internal dynamics—then the memoryless mapping $y_t = g_\theta(x_0, u_t)$ is recovered. If f_θ and g_θ are linear functions, then (2.7) becomes a

linear time-invariant (LTI) state space model. When f_θ is a neural network, then the model becomes a recurrent neural network.

2.1.4 Direct Parameterizations

To conclude the discussions of learning from data, some motivations for the model class developed in this thesis are presented. As noted before, there are broadly two approaches to solving constrained optimization problems, both with their shortcomings.

Projected gradient descent methods ensure feasibility of the solution with respect to the constraints, but are more computationally expensive than unconstrained gradient methods, and may be more susceptible to local minima. On the other hand, penalty methods do not introduce significant computational costs, but the constraints are usually only approximately satisfied.

In this work, a different perspective is employed, where instead of enforcing constraints during the optimization process, the constraints are directly included in the model structure. In other words, the model is parameterized in a way that smoothly maps from an unconstrained parameter set $\Theta = \mathbb{R}^M$ to the set of functions that satisfy the constraints. This idea is referred to as a *direct parameterization*. It was used in Revay et al. (2021b) to construct a general class of neural network models with various desirable properties such as robustness in terms of a Lipschitz bound. In this work, the focus is on models with stability and stabilizability properties, which will be defined in the following section.

2.2 Stability of Dynamical Systems

As mentioned in Section 2.1, stability is an important and desirable property for models of dynamical systems. Unstable models tend to be sensitive to perturbations in the input, i.e. small changes in the input can produce large changes in the output

(or internal state if there is no output function). This makes them unsuitable for prediction tasks since small perturbations to the initial condition and input signal can lead to diverging predictions.

In the following, the notion of stability is made precise. The focus will be on discrete-time (DT) systems as DT models are more commonly used in system identification problems than continuous-time (CT) models, since data used in learning usually comes in the form of discrete sequences with a uniform time interval.

Consider a general nonlinear DT dynamical system of the form

$$x(t+1) = f(x(t), u(t)). \tag{2.8}$$

There are many forms of stability for a system (2.8). One can consider the stability of the homogeneous response of the system, i.e. stability of the system in the absence of inputs, also known as Lyapunov stability. One can also consider input-to-state stability for a uniformly bounded input signal (Sontag, 2008), or incremental stability (Angeli, 2002) between pairs of trajectories.

In the following, the standard definitions of stability are first presented for the general case, before discussing the special cases that are considered in this thesis.

Definition 2.1 (Lyapunov stability)**.** *For an autonomous system $x_{t+1} = f(x_t)$ with equilibrium $x^\star$ such that $f(x^\star) = x^\star$, the equilibrium is said to be*

1. Lyapunov stable *if for each $\epsilon > 0$, there exists $\delta > 0$ such that if $\|x(0) - x^\star\| < \delta$, then $\|x(t) - x^\star\| < \epsilon$ for all $t \geq 0$ (alternatively, one can say that the signal $x(t)$ is uniformly bounded),*

2. locally asymptotically stable *if in addition, there exists $\delta > 0$ such that $x(t) \to x^\star$ as $t \to \infty$ for all trajectories satisfying $\|x(0) - x^\star\| < \delta$,*

3. locally exponentially stable *if in addition, there exist $c_1, c_2 > 0$ such that*

$$\|x(t) - x^\star\| \leq c_1 \|x(0) - x^\star\| e^{-c_2 t},$$

for all trajectories satisfying $\|x(0) - x^\star\| < \delta$*, and*

4. unstable *if it is not stable.*

The set of initial conditions x_0 whose trajectory $x(t)$ converges to $x^\star$ is called the *region of attraction* of $x^\star$. An equilibrium is said to be *globally* (asymptotically or exponentially) stable if its region of attraction is the whole state space of x.

Stability analysis of general nonlinear systems is often limited to local convergence analysis, e.g. via local linearization, and can be complicated by the presence of multiple equilibria or other asymptotic behaviour such as limit cycles. For certain classes of systems, namely LTI systems and contracting systems, stability can be verified globally and all unforced solutions converge to a single equilibrium.

2.2.1 Stability of Linear Systems

Consider a DT linear time-invariant (LTI) system of the form:

$$x_{t+1} = Ax_t + Bu_t. \tag{2.9}$$

The stability of System (2.9) is completely characterized by the transition matrix A. A square matrix M can therefore also be referred to as stable, where it is understood that the stability of the matrix M refers to the stability of the linear system with M as the transition matrix.

For a DT transition matrix M, stability is defined as follows:

Definition 2.2 (Schur stability)**.** *A square matrix M is Schur stable if and only if all of its eigenvalues lie within the unit circle in the complex plane.*

This definition of stability is easy to verify given a matrix but can be difficult to enforce as a constraint in a learning problem, as the set of Schur stable matrices is not convex. To find a convex stability condition, we can consider Lyapunov stability for linear systems, which can be stated as follows.

Definition 2.3 (Lyapunov stability for linear systems)**.** *For a DT LTI system* $x_{t+1} = Ax_t$*, the following statements are equivalent:*

1. *The matrix* A *is Schur stable,*
2. (Lyapunov equation) *There exist* $P, Q \succ 0$ *such that*

$$P - A^\top PA = Q, \tag{2.10}$$

3. (Lyapunov inequality) *There exists* $P \succ 0$ *such that*

$$P - A^\top PA \succ 0, \tag{2.11}$$

4. *The equilibrium* $x = 0$ *is globally exponentially stable.*

For a controlled system of the form (2.9), Schur stability of A also implies bounded-input bounded-output (BIBO) stability of (2.9).

The Lyapunov inequality (2.11) is a linear matrix inequality (LMI), and can be easily verified by solving a convex feasibility problem when A is known. However, this LMI is not jointly convex in A and P.

One can also consider an implicit model $Ex_{t+1} = Fx_t + Gu_t$ for some invertible E. This is equivalent to the LTI system (2.9) by writing $A = E^{-1}F$ and $B = E^{-1}G$. It can be shown (Manchester et al., 2021; Tobenkin et al., 2017) that this implicit model is stable if and only if the following LMI is satisfied:

$$\begin{bmatrix} E + E^\top - P & F^\top \\ F & P \end{bmatrix} \succ 0. \tag{2.12}$$

This LMI is jointly convex and also sum-separable in E, F and P. The sum-separability of this matrix will enable a direct parameterization of stable matrices presented in Section 4.3.1. It is also worth noting that feasibility of this LMI guarantees that E is invertible.

Continuous-time case

In continuous-time (CT), the analogous definition for a stable matrix is stated as follows:

Definition 2.4 (Hurwitz matrix)**.** *A square matrix M is Hurwitz if and only if all of its eigenvalues have strictly negative real parts.*

The Lyapunov inequality in CT is given by:

$$A^\top P + PA + Q = 0, \quad P, Q \succ 0. \tag{2.13}$$

2.2.2 Contraction

Contraction analysis (Lohmiller and Slotine, 1998) provides another way to study nonlinear systems by means of linear systems theory *exactly* and *globally*. In contraction analysis, one is concerned with the differential dynamics of a given autonomous system:

$$x(t+1) = f(x(t), t), \tag{2.14}$$

where $x \in \mathbb{R}^n$.

The differential dynamics of the model (2.14) are given by a linear time-varying (LTV) system of the form

$$\delta x(t+1) = \frac{\partial f}{\partial x}(x(t))\delta x(t), \tag{2.15}$$

with $\delta x \in \mathbb{R}^n$ representing the infinitesimal displacement. Informally, if the LTV system (2.15) is exponentially stable along any feasible trajectories $x(t)$, we can say the system (2.14) is contracting. Its formal definition is given as follows.

Definition 2.5. Given the DT system (2.14), if there exists a uniformly bounded metric $M(x)$, i.e. $a_1 I_n \preceq M(x) \preceq a_2 I_n$ for some $a_2 \geq a_1 > 0$, guaranteeing

$$\frac{\partial f}{\partial x}(x(t))^\top M(x(t+1))\frac{\partial f}{\partial x}(x(t)) - M(x(t)) \preceq -\beta M(x(t)), \tag{2.16}$$

with $0 < \beta < 1$, then the given system is contracting. $M(x)$ is a Riemannian metric defined on the tangent space of the state manifold. If the left-hand side of (2.16) is strictly negative definite, then the system is asymptotically contracting.

A central result of contraction analysis is that, for contracting systems, all trajectories converge exponentially to a single trajectory, i.e., for any two trajectories x_a and x_b,

$$|x_a(t) - x_b(t)| \leq a_0 \beta^t |x_a(0) - x_b(0)| \tag{2.17}$$

for some $a_0 > 0$.

Continuous-time case

Contraction analysis can also be applied to continuous-time systems of the form:

$$\dot{x} = f(x, t). \tag{2.18}$$

The definition of contraction is analogous to that in the discrete-time case.

Definition 2.6. If there exists a uniformly bounded metric $M(x, t)$ such that

$$\dot{M} + \frac{\partial f}{\partial x}^\top M + M \frac{\partial f}{\partial x} \preceq -2\lambda M, \tag{2.19}$$

where $\dot{M} = \frac{\partial M}{\partial t} + \sum_i \frac{\partial M}{\partial x_i} f_i(x)$, then the system (2.18) is contracting with rate λ.

2.2.3 Control Contraction Metrics

The notion of contraction can be extended to the stabilizability of control-affine nonlinear systems of the form

$$x(t+1) = f(x(t)) + g(x(t))u(t). \tag{2.20}$$

For System (2.20), one can ask the question: when can this system be globally stabilized? To answer this question, the definition of stabilizability first needs to be clarified.

Definition 2.7 (Global exponential stabilizability (Manchester and Slotine, 2017))**.** *A target trajectory* $(x^\star, u^\star)$*, where* $x^\star$ *and* $u^\star$ *are the desired states and inputs respectively, is globally exponentially stabilizable if there exists a feedback controller such that for any initial condition* $x(0)$*, a unique solution* $x(t)$ *of* (2.20) *exists for all* t *and satisfies*

$$\|x(t) - x^\star\| \leq e^{-\lambda t} R \|x(0) - x^\star(0)\|, \tag{2.21}$$

with rate $\lambda > 0$ *and overshoot* $R > 0$*.*

A system is said to be *universally exponentially stabilizable* (UES) if every forward-complete solution is globally exponentially stabilizable. One can verify that a system is UES if there exists a control contraction metric (CCM) for that system (Manchester and Slotine, 2017; Wei et al., 2021).

Definition 2.8 (Discrete Control Contraction Metric (Wei et al., 2021))**.** *For system* (2.20) *with differential dynamics*

$$\delta x_{t+1} = F(x_t)\delta x_t + G(x_t)\delta u_t, \tag{2.22}$$

where $F(x_t) = \frac{\partial(f(x_t)+g(x_t)u_t)}{\partial x}$ *and* $G(x_t) = \frac{\partial(f(x_t)+g(x_t)u_t)}{\partial u}$*, if there exists a uniformly bounded metric* $M(x)$ *and a differential feedback controller* $\delta u_t = K(x_t)\delta x_t$ *that satisfy*

$$M(x_t) - (F(x_t) + G(x_t)K(x_t))^\top M(x_{t+1})(F(x_t) + G(x_t)K(x_t)) \succ \beta M(x_t), \tag{2.23}$$

then $M(x_t)$ *is called a discrete control contraction metric for system* (2.20)*.*

Definition 2.8 provides a verifiable condition for a system to be UES, but does not immediately provide a construction for a stabilizing controller. In Wei et al. (2021), a sum-of-squares (SOS) problem is formulated to jointly search for a CCM and a corresponding feedback controller, when the differential dynamics are known. In

Chapter 5, a framework is proposed to jointly learn a CCM controller and dynamics model.

2.3 Koopman Theory

From the previous discussions, it is clear that stability analysis is significantly simpler for linear systems than nonlinear systems. In particular, stability of a linear system can be verified globally via its spectral decomposition (i.e. eigendecomposition). Koopman spectral analysis (Koopman, 1931; Mezić, 2005) is an operator-theoretic framework that studies nonlinear systems via an infinite-dimensional linear operator that describes the evolution of measurement functions of the system states. The Koopman operator enables the use of linear systems theory to study the behaviour of nonlinear systems, including stability (Mauroy and Mezić, 2016). In the following, the Koopman operator and its associated spectral properties are defined for continuous and discrete time systems, along with numerical approximations of the Koopman operator and extensions to controlled systems.

2.3.1 The Koopman Operator

The Koopman operator was first proposed in Koopman (1931) to study the flow of continuous-time nonlinear systems from an operator-theoretic perspective. In this work, the focus is on the discrete-time Koopman operator, since it is more suitable for identification problems where data are usually discrete and uniformly-sampled. The definition of the DT Koopman operator is given in the following.

Definition 2.9. (*Discrete-time Koopman operator*) Let $\mathcal{F}$ be a Hilbert space of smooth real-valued scalar functions $\mathbb{R}^n \to \mathbb{R}$. For the DT dynamical model (2.8), the Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ is defined by

$$\mathcal{K}[\varphi(x)] := \varphi \circ f(x) \tag{2.24}$$

for $\varphi \in \mathcal{F}$, assuming that the system has a unique solution $\forall t \in \mathbb{N}$. The scalar real-valued function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is termed an *observable*.

Since the Koopman operator is defined on the functional space, it is infinite-dimensional. It is also easy to verify that the Koopman operator is linear, i.e. $\mathcal{K}[k_1\varphi_1 + k_2\varphi_2] = k_1\mathcal{K}[\varphi_1] + k_2\mathcal{K}[\varphi_2]$ for any $k_1, k_2 \in \mathbb{R}$ and $\varphi_1, \varphi_2 \in \mathcal{F}$. This property makes Koopman methods widely popular in the analysis of dynamical models. Despite the infinite dimension of the Koopman operator, some key properties of a given nonlinear dynamical model — e.g. stability (Mauroy and Mezić, 2016) and dynamical behaviours — can be captured by a few particular functions, i.e. the Koopman eigenfunctions.

Definition 2.10 (Koopman eigenfunction)**.** *A Koopman eigenfunction is a non-zero observable $\varphi_\lambda \in \mathcal{F}/\{0\}$ satisfying*

$$\mathcal{K}[\varphi_\lambda(x)] = \lambda\varphi_\lambda(x) \tag{2.25}$$

for some $\lambda \in \mathbb{C}$, which is the associated Koopman eigenvalue.

A Koopman eigenfunction defines a coordinate in which the system trajectories behave as a linear system. To be precise, define a coordinate change $z_\lambda = \varphi_\lambda(x)$, the dynamics of which are given by $z_\lambda(t+1) = \lambda z_\lambda(t)$, with the initial condition $z_\lambda(0) = \varphi_\lambda(x(0))$. Indeed, the definition (2.25) is equivalent to solving the algebraic equation $\varphi_\lambda(f(x)) = \lambda\varphi_\lambda(x),\ \forall x \in \mathbb{R}^n$ if the DT dynamical model (2.8) is prior.

A Koopman operator will have infinitely many eigenfunctions, as a set of eigenfunctions can be used to construct more eigenfunctions. For example, given two eigenfunctions $\varphi_{\lambda_1}(x)$ and $\varphi_{\lambda_2}(x)$, their product is also an eigenfunction:

$$\mathcal{K}[\varphi_{\lambda_1}(x)\varphi_{\lambda_2}(x)] = \lambda_1\lambda_2\varphi_{\lambda_1}(x)\varphi_{\lambda_2}(x),$$

where $\lambda_1\lambda_2$ is the corresponding eigenvalue.

To obtain a tractable representation of the Koopman operator, one can consider a finite set of Koopman eigenfunctions, which spans a Koopman invariant subspace.

Definition 2.11 (Koopman-invariant subspace)**.** *A Koopman-invariant subspace is defined as $\mathcal{G} \subset \mathcal{F}$ such that for all observables $\varphi \in \mathcal{G}$, $\mathcal{K}\varphi \in \mathcal{G}$.*

If $\mathcal{G}$ is spanned by a finite set of observables $\{\varphi_k\}_{k=1}^{K}$, then all linear combinations of φ_k:

$$g(x) = \sum_k a_k \varphi_k$$

remain in the subspace under the Koopman operator:

$$\mathcal{K}[g(x)] = \sum_k b_k \varphi_k,$$

for some $a_k, b_k \in \mathbb{R}$.

By restricting the Koopman operator to such a subspace, a finite-dimensional matrix representation $\mathbf{K}$ of the Koopman operator can be obtained. This matrix $\mathbf{K}$ is referred to as the *Koopman matrix*. The set of observables that span a Koopman-invariant subspace can be written as a vector-valued function:

$$\phi(x) = [\varphi_1, \ldots, \varphi_K]^\top . \tag{2.26}$$

The dynamics of $\phi(x)$ are linear and given by

$$\phi(x_{t+1}) = \mathbf{K}\phi(x_t). \tag{2.27}$$

Furthermore, if $\phi(x)$ is a homeomorphism (bijective, continuous and has a continuous inverse), then the system (2.27) is topologically conjugate (Brunton et al., 2021) to the original nonlinear system (2.8); that is, the two systems have equivalent dynamics. More generally, the system (2.27) is topologically semi-conjugate to the system (2.8) (the dynamics of (2.27) contain the dynamics of (2.8)) if $\phi(x)$ is injective (Bevanda et al., 2021). As such, $\phi(x)$ is referred to as a *Koopman embedding*, in the topologic sense.

To illustrate the idea of a Koopman embedding, consider the following example from Brunton et al. (2016):

Example 2.1 (Koopman embedding)**.** *Consider the system:*

$$\dot{x} = \begin{bmatrix} -x_1 \\ -x_2 + x_1^2 \end{bmatrix}. \tag{2.28}$$

By augmenting the state vector $x = [x_1, x_2]^\top$ *with the nonlinear observable* x_1^2*, the new coordinates* $y = [x_1, x_2, x_1^2]^\top$ *define a Koopman-invariant subspace with linear dynamics:*

$$\frac{dy}{dt} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -2 \end{bmatrix} y \tag{2.29}$$

Continuous-time Koopman Operator

In this section, trhe definition of the continuous-time Koopman operator will be briefly presented for completeness. For an autonomous system governed by an ordinary differential equation (ODE)

$$\dot{x} = f(x), \tag{2.30}$$

there exists a semigroup of Koopman operator $\mathcal{K}^t$ associated with the flow map $X(x,t)$ of the system, defined as:

$$\mathcal{K}^t \phi(x(t)) = \phi(X(x,t)). \tag{2.31}$$

The infinitesimal generator of this semigroup is referred to as the continuous-time Koopman operator $\tilde{\mathcal{K}}$ (Williams et al., 2015):

$$\tilde{\mathcal{K}}\phi(x(t)) = \frac{d}{dt}\phi(x(t)) = \nabla\phi \cdot f(x(t)). \tag{2.32}$$

2.3.2 Equivalence of Contraction and Koopman Stability

In Koopman analysis, a central question is whether a finite-dimensional Koopman embedding exists for a given system that yields an exact linear representation of its dynamics. Yi and Manchester (2022) showed that for continuous-time systems, such

an embedding is guaranteed to exist if the system is contracting. In Section 4.2, this result is extended to discrete-time systems.

To illustrate the construction of this embedding, let us revisit the example system (2.28). The following example from Yi and Manchester (2022) demonstrates how the Koopman embedding can be analytically derived for a contracting system given its dynamics.

Example 2.2 (Koopman embedding for contracting system)**.** *Consider again the system:*

$$\dot{x} = \begin{bmatrix} -x_1 \\ -x_2 + x_1^2 \end{bmatrix}. \tag{2.33}$$

The Jacobian matrix $F(x)$ is given by

$$F(x) := \frac{\partial f}{\partial x}(x) = \begin{bmatrix} -1 & 0 \\ 2x_1 & -1 \end{bmatrix}.$$

By choosing the metric $M(x) = \left(\begin{smallmatrix} 1+4x^2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, we can verify that this system is contracting by the CT contraction condition:

$$\dot{M}(x) + M(x)F(x) + F(x)^\top M(x) = \begin{bmatrix} -2 - 16x_1^2 & 2x_1 \\ 2x_1 & -2 \end{bmatrix} \prec 0.$$

From (Yi and Manchester, 2022, Theorem 2), there exists a Koopman embedding for this system given by:

$$\phi(x) = x + \int_0^{+\infty} \exp(F(x_\star)s) H(\breve{X}(x, -s)) ds, \tag{2.34}$$

where $H(x) = F(x_\star)x - f(x)$ and $\breve{X}(x,t)$ is the solution of modified dynamics defined as:

$$\dot{x} = \rho(x) f(x), \quad \rho(x) = \begin{cases} 1, & \text{if } \ x \in \mathtt{cl}(\mathcal{X}) \\ 0, & \text{if } \ x \notin \mathcal{X}' \end{cases} \tag{2.35}$$

for some $\mathtt{cl}(\mathcal{X}) \subset \mathcal{X}' \subset \mathbb{R}^n$.

For System (2.33), $F(x_\star) = F(0) = diag(-1,-1)$. *The flow* $X(x,t)$ *is given by:*

$$X(x,t) = \begin{bmatrix} e^{-t}x_1 \\ e^{-t}x_1^2 + e^{-t}x_2 - e^{-2t}x_1^2 \end{bmatrix},$$

and $H(x) = [0, -x_1^2]^\top$. *The system can be modified in the open set* $\mathcal{X} := \{0 < x_1 < 1\}$ *to yield the backward flow* $\breve{X}_1$ *as:*

$$\breve{X}_1(x,t) = \begin{cases} e^{-t}x_1, & \ln x_1 \leq t \leq 0 \\ 1, & t < \ln x_1. \end{cases}$$

The Koopman embedding (2.34) *is then given by:*

$$\begin{aligned} \phi(x) &= x + \int_{-\infty}^{0} \begin{bmatrix} e^s & 0 \\ 0 & e^s \end{bmatrix} \begin{bmatrix} 0 \\ -\breve{X}_1(x,s)^2 \end{bmatrix} ds \\ &= x - \int_{\ln(x_1)}^{0} \begin{bmatrix} 0 \\ e^s(e^{-s}x_1)^2 \end{bmatrix} ds - \int_{-\infty}^{\ln(x_1)} \begin{bmatrix} 0 \\ e^s 1^2 \end{bmatrix} ds \\ &= \begin{bmatrix} x_1 \\ -2x_1 + x_1^2 + x_2 \end{bmatrix}, \end{aligned}$$

which satisfies

$$\dot{\phi}(x) = F(x^\star)\phi(x) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \phi(x).$$

The result from Yi and Manchester (2022) shows that for CT contracting systems, there always exists a Koopman embedding of the same dimensionality as the original state space. However, analytically deriving such an embedding as in the example above is not always possible, especially when the dynamics are unknown. In this work, learning-based methods that approximate the Koopman embedding are considered instead.

2.3.3 Dynamic Mode Decomposition

In the problem of system identification, we are interested in finding the Koopman eigenfunctions and eigenvalues only from the collected data set $\{\tilde{x}_t\}_{t=0}^T$, for which dynamic mode decomposition (DMD) provides an efficient data-driven approach to approximate the Koopman operator (Schmid, 2010).

In DMD, usually some heuristically *predetermined*, sufficiently rich observables $\varphi_1, \dots \varphi_N$ ($N \gg n$)—rather than Koopman eigenfunctions—are involved to learn the nonlinearity in the dynamical model. The task in the DMD method is to seek a matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$ in order to obtain a finite-dimensional approximation of $\mathcal{K}$, which minimizes

$$\sum_{j=0}^{T} |\phi(x(t+1)) - \mathcal{A}\phi(x(t)))|_2^2, \tag{2.36}$$

where $\phi := [\varphi_1, \dots, \varphi_N]^\top$. The least square problem (2.36) has a unique solution

$$\mathcal{A} = Y_1 Y_2^\dagger \tag{2.37}$$

with $Y_1 := [\phi(x(1)), \dots, \phi(x(T))]$, $Y_2 := [\phi(x(0)), \dots, \phi(x(T-1))]$ if Y_2 is full row rank. DMD is a simple, efficient method to approximate the Koopman operator, but two issues arise:

1) In the DMD method, the observables φ are predetermined, which significantly affects the learning accuracy, but in the literature the selection of observables usually done in a heuristic manner Williams et al. (2015). Since these observables are closely connected to the Koopman eigenfunctions for a given dynamical model, a natural question is: can the observables and the matrix $\mathcal{A}$ be learnt concurrently to improve accuracy?

2) For a stable dynamical model, the above least square solution may yield an unstable model due to various kinds of perturbations in the data set $\{\tilde{x}_t\}_{t=0}^T$, which would be unacceptable in many applications. Hence, imposing stability constraints is an important consideration in learning algorithms.

2.3.4 Koopman Operator with Control

There have been many frameworks proposed to extend Koopman operator theory to include control.

Consider the discrete-time system

$$x_{t+1} = f(x_t, u_t) \tag{2.38}$$

with $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$.

The Koopman with Inputs and Control (KIC) framework (Proctor et al., 2018) generalizes the definition of the Koopman operator by defining a Hilbert space $\mathcal{H}$ containing observables that are functions of both the state and input, $\phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$. The KIC operator $\mathcal{K} : \mathcal{H} \to \mathcal{H}$ is defined as:

$$\mathcal{K}\phi(x_t, u_t) = \phi(f(x_t, u_t), u_{t+1}). \tag{2.39}$$

This definition of $\mathcal{K}$ reduces to the autonomous case if the state vector is augmented with the control input to form a new state $y = [x^\top, u^\top]^\top$. Furthermore, if u is generated by a state feedback controller as $u_t = k(x_t)$, then $\mathcal{K}$ is again equivalent to the autonomous Koopman operator if $k(x)$ is in the space of observables. However, this formulation is not well-suited for control design due to the coupling between the state and input in the augmented state y.

An alternative formulation transforms the system into a linear parameter-varying (LPV) system via a lifting function composed of a set of observables of the state only, that span a Koopman-invariant subspace (Iacob et al., 2022a,b). In order to separate the state and input in the lifted space, first decompose the function $f(x_t, u_t)$ into its autonomous and forced dynamics:

$$f(x_t, u_t) = f(x_t, 0) + g(x_t, u_t). \tag{2.40}$$

Then given a lifting function $\phi : \mathbb{R}^n \to \mathbb{R}^{n_K}$, the LPV Koopman system is given by:

$$\phi(x_{t+1}) = A\phi(x_t) + B(x_t, u_t)u_t, \tag{2.41}$$

where

$$B(x_t, u_t) = \int_0^1 \frac{\partial}{\partial u} \left(\left(\int_0^1 \frac{\partial \phi}{\partial x} \left(f(x_t, 0) + \lambda g(x_t, \mu u_t) \right) d\lambda \right) g(x_t, \mu u_t) \right) d\mu \tag{2.42}$$

Continuous-time Case

A analogous formulation can be derived for a continuous-time system

$$\dot{x} = f(x, u) = f(x, 0) + g(x, u). \tag{2.43}$$

Given a Koopman mapping $\phi(x)$, the dynamics of the lifted system is given by (Iacob et al., 2022a):

$$\dot{\phi}(x) = A\phi(x) + B(x, u)u, \tag{2.44}$$

where

$$B(x, u) = \int_0^1 \frac{\partial}{\partial u} \left(\frac{\partial \phi}{\partial x}(x) g(x, \lambda u) \right) d\lambda. \tag{2.45}$$

For a control-affine system of the form $\dot{x} = f(x) + g(x)u$, the Koopman system (2.44) is also a control-affine system (Surana, 2016):

$$\dot{\phi}(x) = A\phi(x) + \frac{\partial \phi}{\partial x}(x) g(x) u. \tag{2.46}$$

Many prior works, e.g. (Folkestad et al., 2022; Goswami and Paley, 2017; Huang et al., 2018; Surana, 2016), have also proposed a further simplifying assumption that there exists a matrix $B_i \in \mathbb{R}^{n_K \times n_K}$ for each element u_i in u, such that

$$\frac{\partial \phi}{\partial x}(x) g_i(x) = B_i \phi(x). \tag{2.47}$$

Then the Koopman system takes on a bilinear form:

$$\dot{\phi}(x) = A\phi(x) + \sum_{i=1}^{m} B_i \phi(x) u_i. \tag{2.48}$$

By identifying a system of this form, one can then apply bilinear control methods to control the original nonlinear system.

This bilinear form may be further simplified to a linear system (Yi and Manchester, 2022) by finding a transformation $u = \alpha(x, v)$ that satisfies:

$$B_v v = \sum_{i=1}^{m} B_i \phi(x) \alpha_i(x, v), \tag{2.49}$$

where α_i are the components of the vector-valued function α. An LTI system is then obtained:

$$\dot{\phi}(x) = A\phi(x) + B_v v. \tag{2.50}$$

2.4 Summary

To summarise, this chapter first discussed the problem of statistical learning in terms of choice of models and optimization methods. Next, the notion of stability was defined for linear systems and contracting systems. Finally, the underlying theory of the Koopman operator was presented, and the relevant terminology was defined.

Chapter 3

Related Work

In this chapter, related work on the problems considered in this thesis, namely, the problem of learning stable Koopman models in Chapter 4 and the problem of stable imitation learning in Chapter 5, are reviewed.

3.1 Learning Koopman Models

The underlying theory of the Koopman operator was presented in Section 2.3. In this section, methods for learning the Koopman matrix and/or embedding from data will be presented. The literature on Koopman-based methods is vast, and readers are referred to recent reviews by Bevanda et al. (2021); Brunton et al. (2021); Otto and Rowley (2021) for more details.

As discussed in Section 2.3.3, dynamic mode decomposition (DMD) is a method for estimating the Koopman operator for a pre-determined set of observables given snapshots of the state transitions. The original DMD algorithm (Schmid, 2010) considered linear observables $\phi(x) = x$ to approximate the Koopman operator as a finite-dimensional matrix. Linear observables were shown to be effective for analysing the spectrum of high-dimensional fluid flows via a truncated singular value decomposition (SVD) of the Koopman matrix. Williams et al. (2015) proposed extended DMD

(EDMD) which used a dictionary of basis functions as nonlinear observables to better model nonlinearities. Proctor et al. (2016) extended DMD to identify controlled systems using the DMD with control (DMDc) framework.

For DMD methods, a central question is the selection of appropriate basis functions that capture the nonlinear behaviour of the system being studied. In the case of EDMD, the basis functions are usually chosen heuristically, such as monomials or radial basis functions. Other bases such as time-delay coordinates (Brunton et al., 2017) or an implicit basis defined by a kernel function (Williams et al., 2016) have also been proposed.

Recently, many works have investigated data-driven methods for automatically acquiring a set of observables, by modelling the Koopman embeddings as neural networks (Azencot et al., 2020; Erichson et al., 2019; Li et al., 2017a; Lusch et al., 2018; Mardt et al., 2018; Otto and Rowley, 2019; Pan and Duraisamy, 2020; Takeishi et al., 2017; Yeung et al., 2019). The neural networks typically have an encoder-decoder architecture, with the encoder representing the Koopman embedding and the decoder being an inverse mapping to map back to the original state space. Additionally, there may be a linear model between the encoder and decoder to represent the linear dynamics of the Koopman subspace.

One consideration in learning Koopman embeddings is the dimensionality of the Koopman-invariant subspace compared to that of the original state space. If the original state space is of very high-dimensional, for example as seen in fluid dynamics problems (Erichson et al., 2019), then typically a lower dimensional Koopman subspace is desired, in order to obtain a reduced-order model of the system and identify the dominant eigenfunctions of the Koopman operator. On the other hand, for lower-dimensional, highly nonlinear systems such as the classic inverted pendulum, the goal is to learn a higher-dimensional Koopman embedding that lifts the nonlinear system onto a linear manifold. This high-dimensional lifting approach is the one taken in this work as it is appropriate for the problems considered.

3.1.1 Stable Koopman Models

Relatively few works have considered stability of the Koopman model. Mamakoukas et al. (2020) proposed to use a constrained parameterization of stable matrices to represent the Koopman matrix, and optimized the model using a projected gradient descent (PGD) algorithm. However, as discussed in Section 2.1.2, PGD optimization methods suffer from scalability and convergence issues. Pan and Duraisamy (2020) proposed a diagonal matrix parameterization for guaranteeing stability of the continuous-time Koopman matrix while jointly learning an embedding. However, the diagonal parameterization restricts the embedding to represent only eigenfunctions and not generalized observables.

Following the publication of the work here in Fan et al. (2022), the stable parameterization proposed in Section 4.5 was used in Bevanda et al. (2022b) to learn a Koopman model with a diffeomorphic embedding.

3.1.2 Koopman Models for Control

A strong motivation for identifying Koopman models is the benefit of applying linear control design methods to nonlinear systems. To this end, many works have proposed parameterizations of Koopman models for controlled systems (Bevanda et al., 2022a; Folkestad et al., 2022; Han et al., 2020; Huang et al., 2018; Iacob et al., 2021; Kaiser et al., 2021; Sinha et al., 2022; Zinage and Bakolas, 2022). Most methods identify a latent Koopman space with dynamics that are linear in the latent state and either 1) linear in the control, 2) bilinear in the control, or 3) control-affine. Models that are linear in the control can be restrictive in terms of the classes of the systems they can represent, but are still attractive as they are amenable to linear control methods such as LQR (Bevanda et al., 2022a). Bilinear models are more flexible than linear models but require more complex control methods like nonlinear model predictive control (NMPC) (Folkestad et al., 2022) or control design via control Lyapunov functions (Huang et al., 2018; Zinage and Bakolas, 2022) such as Sontag's formula (Sontag,

1989). Finally, control-affine models most closely match the theoretical form (2.41), but require linear parameter-varying control methods (Iacob et al., 2021).

The identification of Koopman models for control usually involves applying random inputs to the system and recording the state trajectories. A controller is then designed for the learned model after training. As such, these models are not immediately applicable to the problem of imitation learning considered in this work, since the model only represents the open-loop dynamics and does not explicitly contain the controller. To the best of the writer's knowledge, this is the first work to consider applying Koopman models to the problem of imitation learning.

3.2 Imitation Learning

Imitation learning (IL), also known as learning from demonstrations, is a rapidly growing field at the intersection of many disciplines including machine learning, control theory and human-robot interaction. Imitation learning considers the problem of learning a control policy of an autonomous agent given demonstrations of the desired actions or trajectories. These demonstrations can come from an unknown optimal controller or an expert human operator. A comprehensive review of imitation learning methods is beyond the scope of the thesis. See Osa et al. (2018) for a recent review of IL algorithms from a machine learning perspective. The focus for this section will be on methods most similar to the one proposed in Chapter 5.

Imitation learning algorithms can be broadly classified by the function being learnt. One class of IL algorithms known as behavioural cloning (BC) (Bain and Sammut, 1995) explicitly learns a control policy, which can be seen as a form of supervised learning. Another approach is to assume that the observed controller is optimizing some unknown reward (or negative cost) function:

$$u^\star = \arg\max_{u_k} \mathbb{E}\left[\sum_{k=t}^{T} r_\theta(x_k, u_k)\right], \tag{3.1}$$

where $u^\star$ is the optimal control/action, and $\mathbb{E}\left[\sum_{k=t}^{T} r_\theta(x_k, u_k)\right]$ is the expected return

of the policy from the current time t until the final time T. The imitation learning problem then becomes a problem of learning the reward function $r_\theta(x, u)$. This paradigm is referred to as inverse reinforcement learning (IRL) (Russell, 1998), and is also closely related to inverse optimal control (IOC) (Kalman, 1964), with the two terms sometimes used interchangeably. In this thesis, the behavioural cloning approach is considered, but the proposed method also has connections to IOC which will be discussed in Section 5.3.5.

Within these two paradigms, algorithms can be further divided by whether a forward dynamics model is learned (model-based) or not (model-free). Model-free methods are favoured when the dynamics may be difficult to model due to discontinuities or contact forces. On the other hand, model-based methods can be useful when the system is difficult to control, e.g. if it is underactuated. In this case, learning a dynamics model can help ensure that the learned policy will produce feasible trajectories.

The learned policy can be either stochastic or deterministic. Deterministic policies are a sensible choice when modelling an optimal policy of a deterministic dynamical system. On the other hand, stochastic policies can be useful for modelling uncertainty in observed demonstrations, or suboptimality of the demonstrations through the principle of maximum entropy (Jaynes, 1957; Ziebart et al., 2008).

In the following, algorithms for each paradigm will be discussed separately, along with a focused discussion on imitation learning algorithms that consider stability of the learned controller.

3.2.1 Behavioural Cloning

One of the early successes of behavioural cloning was in autonomous driving (Pomerleau, 1991), where a neural network was trained to map from road images to steering inputs of a car. Since then, BC has been successfully applied to many other problems including autonomous helicopter flight (Abbeel et al., 2010) and robotic surgery (Osa et al., 2017).

The policy representation is a central consideration for BC. In this work, the control policy learned is a state-to-action mapping parameterized by θ:

$$u = \pi_\theta(x). \tag{3.2}$$

Learning such a state-feedback controller enables the use of control-theoretic tools such as contraction analysis to describe the interaction between the controller and the system. An alternative policy representation is to map from initial conditions to entire trajectories, assuming there is a pre-existing low-level controller for trajectory tracking. Many trajectory-based models have been proposed, using Gaussian mixture models (Calinon et al., 2007; Khansari-Zadeh and Billard, 2011), Gaussian processes (Osa et al., 2017) or dynamic movement primitives (Ijspeert et al., 2013; Schaal et al., 2005). Trajectory-based methods have the benefit of guaranteeing feasible and stable trajectories, however, they tend to be less generalizable than methods that learn state-feedback controllers.

One significant shortcoming of BC is the problem of covariate shift (Ross and Bagnell, 2010; Ross et al., 2011). Essentially, any errors in the learned policy's predictions will compound as the system deviates further from the states in the training data. In reinforcement learning terms, this can be seen as an effect of off-policy learning, where the policy that generates the training data is different to the policy executed on the real system. From a supervised learning perspective, this can be described as the policy overfitting to the training data.

Ross et al. (2011) proposed the DAgger algorithm to address covariate shift. DAgger is an online learning algorithm that queries the expert for the correct controls at states encountered by the learner. However, this approach may not always be practical for physical systems.

Behavioural cloning is still widely used as a baseline for benchmarking IL algorithms, due to its ease of implementation and relatively good performance on benchmark problems such as the MuJoCo suite (Fu et al., 2020). One important advantage of vanilla BC is that it only requires state-action pairs as data, and learning is done

entirely offline.

3.2.2 Inverse Reinforcement Learning

Early work on inverse reinforcement learning (IRL) focused on problems with discrete action spaces (Abbeel and Ng, 2004; Ng and Russell, 2000), as the maximization in Equation (3.1) can be easily solved for a discrete set of u to yield a tractable policy representation.

For continuous action spaces, most recent methods are based on the framework of maximum entropy IRL (MaxEntIRL) (Ziebart, 2010; Ziebart et al., 2008). It models the observed policy as an exponential distribution with the reward function as the exponent, which admits a tractable likelihood maximization for discrete state spaces. Guided cost learning (GCL) (Finn et al., 2016) extended MaxEntIRL to continuous domains using a sampling-based approximation to model the distribution. A closely-related method proposed by Ho and Ermon (2016) is called generative adversarial imitation learning (GAIL), named after generative adversarial networks (GANs) (Goodfellow et al., 2014). In GAIL, the reward function acts as a discriminator for the policy, which serves as the generator, and the two functions are jointly trained in an adversarial procedure. GAIL has been widely adopted as a benchmark IRL method, and inspired many extensions (Dadashi et al., 2021; Duan et al., 2017; Fu et al., 2017; Kostrikov et al., 2019; Li et al., 2017b; Wang et al., 2017). However, GAIL-based methods suffer from the same drawback that GANs have, which is difficulty of adversarial training.

Ghasemipour et al. (2020) presented a unifying view of IL methods as minimizing the probabilistic distance (known as f-divergence) between the demonstrator and learner policies. They showed that different IL methods including BC, DAgger and GAIL can be reformulated as minimizations of different types of f-divergences.

One disadvantage of IRL methods compared to BC methods is the fact that most IRL methods are online algorithms, which means the policy is evaluated on the system

during learning to collect additional data. As such, although IRL methods are sample-efficient in terms of expert data, they may require millions of timesteps of interaction with the environment to reach a desired level of performance (Ho and Ermon, 2016). Additionally, during online learning, the policy may exhibit unstable behaviour, which would be a significant safety risk for a physical system.

3.2.3 Stable Imitation Learning

There have been relatively few works that consider the stability of the learned policy from a control-theoretic perspective (Havens and Hu, 2021; Palan et al., 2020; Pfrommer et al., 2022; Tu et al., 2022; Yin et al., 2021).

Early work on stable IL considered the IL problem with known dynamics (Havens and Hu, 2021; Palan et al., 2020; Tu et al., 2022; Yin et al., 2021), and proposed methods to guarantee that the learned policy will be stabilizing for the real system.

Palan et al. (2020) and Havens and Hu (2021) only considered LTI systems with linear controllers, and proposed to constrain the stability of the controller via LMIs. The resulting constrained optimization problem is then be solved using the alternating direction method of multipliers (ADMM) or projected gradient descent (PGD). Both ADMM and PGD are iterative methods that have to solve semidefinite programs at each iteration, hence may not be scalable to high-dimensional problems.

Yin et al. (2021) consider IL with neural network (NN) controllers applied to known LTI systems. They derive an LMI condition for stability of the closed-loop system, by bounding the nonlinearities of the NN using sector quadratic constraints. The constrained optimization problem is then solved using ADMM. However, in order to obtain a convex constraint, all of the bias terms in the NN are set to zero, which severely reduces the expressivity of the NN.

Tu et al. (2022) consider discrete-time control-affine systems of the form (2.20) with known dynamics. They show that the trajectory imitation error can be upper-bounded by the cumulative controller prediction error, if the system is incrementally-

gain-stable, which includes contracting systems. They then propose an iterative online learning algorithm similar to that of Ross and Bagnell (2010) to minimize the controller error. Interestingly, they note in their experiments that hard stability constraints are not required to learn a stabilizing controller, and that optimal controllers may be naturally stabilizing.

The stable IL methods discussed so far all assume knowledge of the dynamics, which allows stability of the closed-loop system to be guaranteed. However, the assumption of known dynamics limits their application to general problem settings where a model of the dynamics may be difficult to derive analytically.

The recent method of Taylor series imitation learning (TaSIL) (Pfrommer et al., 2022) proposed to upper-bound the trajectory imitation error using a Taylor series expansion of the controller prediction error. Notably, TaSIL does not require knowledge of the dynamics, but does require queries to the expert policy to estimate higher-order derivatives of the Taylor series expansion. In many IL problems, the expert policy would not be available for querying and only trajectory data is provided.

As can be seen, existing stable IL methods consider problems where prior knowledge of the system/controller is used to guarantee stability of the learned policy. In this work, the standard BC problem setting is considered, where only trajectory data is available and no prior knowledge of the system is assumed. In such a setting, the learned controller can only be guaranteed to be stable for a *learned* dynamics model. However, such a stability guarantee may act as a regularizer to encourage the controller to be stabilizing for the true system. This hypothesis is supported by the observations made by Tu et al. (2022).

3.3 Summary

To summarise this chapter, prior work on learning Koopman models and imitation learning were discussed, in particular methods that consider stability. Gaps in the current literature were also identified and will be addressed in the following chapters.

Chapter 4

Stable Koopman Models

In this chapter, a new data-driven method for learning stable models of nonlinear systems is presented. The proposed model lifts the original state space to a higher-dimensional linear manifold using Koopman embeddings. It will be shown that every discrete-time nonlinear contracting model can be learnt in our framework. Another significant benefit of the proposed approach is that it allows for *unconstrained* optimization over the Koopman embedding and operator jointly while enforcing stability of the model, via a direct parameterization of stable linear systems, greatly simplifying the computations involved. The proposed method is validated on a simulated system and the advantages of the parameterization compared to alternatives are analyzed.

4.1 Introduction

Let us consider the problem of fitting models to data generated from dynamical systems, known as system identification. One important consideration in system identification is the stability of the model. In many applications, the fitted model is used for prediction of future behaviors of the system, and an unstable model would erroneously produce unbounded predictions.

There are many different forms of stability for nonlinear systems. Contraction, also known as incremental stability, can be viewed as a "strong" type of stability for

dynamical systems that studies the convergence between *any* two trajectories of the given system (Lohmiller and Slotine, 1998). There has been much prior work on learning contracting models in system identification for various model classes, including polynomial models (Tobenkin et al., 2017; Umenberger and Manchester, 2019), Gaussian mixture models (Ravichandar et al., 2017) and neural network models (Manchester et al., 2021; Revay et al., 2021a,b). In this chapter, a new class of contracting nonlinear model is proposed that combines the expressiveness of neural networks with the strong stability guarantees associated with linear systems. Furthermore, a learning framework is proposed that fits this class of models to data via an *unconstrained* optimization problem.

The present work bridges the gap between learning stable nonlinear models and approximating the Koopman operator (Koopman, 1931), an infinite-dimensional *linear* operator that can describe the dynamics of any nonlinear system by embedding it in a higher-dimensional space. There has been growing interest in data-driven methods that estimate finite-dimensional approximations of the Koopman operator and its eigenfunctions (Haseli and Cortes, 2021; Schmid, 2010; Williams et al., 2015), motivated by the appeal of being able to apply linear systems analysis to complex nonlinear systems. Due to this benefit, Koopman-based methods have been developed for system identification (Mauroy and Goncalves, 2020), state observation and control (Korda and Mezić, 2018) of nonlinear systems.

In Koopman identification approaches, a central problem is how to learn the Koopman embedding from data. Recently many methods (Li et al., 2017a; Lusch et al., 2018; Mardt et al., 2018; Otto and Rowley, 2019; Pan and Duraisamy, 2020; Takeishi et al., 2017; Yeung et al., 2019) have been proposed to address this problem, however most of them do not consider the stability of the learned model. Unstable learned models may have serious robustness issues, particularly when applied to dissipative physical systems, making them unsuitable for practical use. The approach taken here to solve this problem is to impose stability constraints on the Koopman model.

The proposed model class is motivated by recent work (Yi and Manchester, 2022) showing that, for continuous-time (CT) nonlinear systems, there is an equivalence

between the Koopman and contraction approaches for stability analysis under some mild technical assumptions.This equivalence result is extended to discrete-time (DT) systems in this chapter, and an algorithmic framework is provided for learning Koopman models with contracting properties.

Consider the identification of a Koopman embedding and operator for a discrete-time (DT) autonomous state-space system:

$$x(t+1) = f(x(t)), \tag{4.1}$$

where $x \in \mathbb{R}^n$ and t is the timestep. The system (4.1) is assumed to have a single equilibrium at $x^\star$, i.e. $f(x^\star) = x^\star$. Further, the function $f(x)$ is assumed to be unknown, but the learning algorithm has access to full-state[1] trajectory data $\{\tilde{x}_t\}_{t=0}^T$ generated by system (4.1), where $\tilde{x}$ denotes measured data. The problem considered is to learn a function $\phi(x)$ (i.e. the Koopman embedding) that smoothly maps from the original state space $\mathbb{R}^n$ to a possibly higher-dimensional space $\mathbb{R}^N$ ($N \geq n$), as well as a linear matrix $\mathcal{A} \in \mathbb{R}^{N\times N}$ (i.e. a finite-dimensional approximation of the Koopman operator) that describes the evolution of $\phi(x)$ over time.

The Koopman embedding and the matrix $\mathcal{A}$ in fact form a predictive model of the system (4.1), which is defined as a Koopman model.

Definition 4.1 (DT Koopman model)**.** *Given a Koopman embedding $\phi(x)$ and matrix $\mathcal{A}$, the corresponding Koopman model is:*

$$x(t) = a(x_0, t) = \phi^L(\mathcal{A}^t \phi(x_0)), \tag{4.2}$$

where $\phi^L : \mathbb{R}^N \to \mathbb{R}^n$ is a left-inverse of $\phi(x)$ such that $\phi^L(\phi(x)) = x$, and x_0 is an

[1]In the case when full-state measurements are not available and states are only partially observed from some output function $y = h(x)$, one can use time-delayed measurements as the state representation, i.e. stack a sequence of past measurements to form the state vector:

$$\tilde{x}_t = [\tilde{y}_t^\top, \tilde{y}_{t-1}^\top \ldots, \tilde{y}_{t-\tau}^\top],$$

where $\tilde{y}_t$ is the measured output and τ is a hyperparameter. This idea is well-known in system identification (Ljung, 1998) and commonly used in Koopman learning (Korda and Mezić, 2018; Takeishi et al., 2017; Tu et al., 2014).

initial condition.

A diagrammatic representation of the discrete-time Koopman model is shown in Figure 4.1.

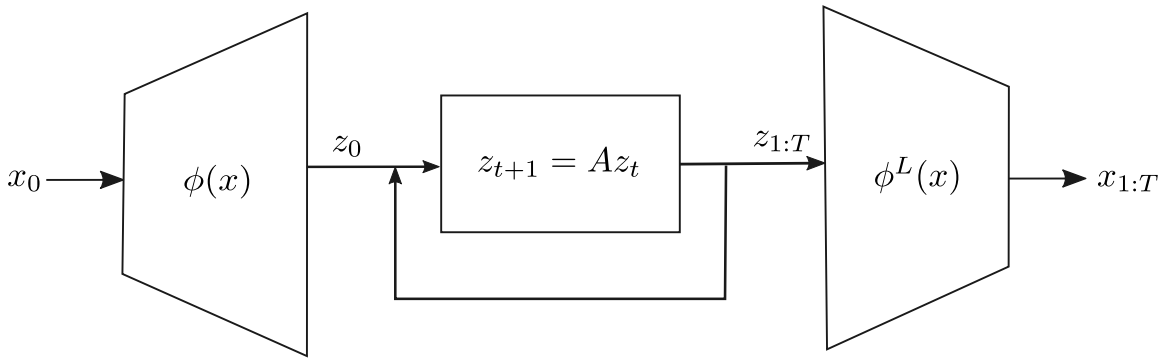


Figure 4.1 – Diagram of discrete-time Koopman model. The notation $x_{1:T}$ denotes the state sequence $\{x_1, \dots x_T\}$, which is the output of the model.

The problem of jointly learning $\phi(x)$ and $\mathcal{A}$ can be treated as a minimization of the prediction error of the Koopman model on the given data $\{\tilde{x}_t\}_{t=0}^{T}$.

4.2 Stability Criterion for Discrete-time Koopman Models

In this section, the main result in Yi and Manchester (2022) is extended to DT systems. It will be proven that the Koopman and contraction approaches are equivalent for nonlinear stability analysis. It will be shown that the model set proposed here can provide sufficient degrees of freedom for learning nonlinear DT models.

Theorem 4.1. *Consider the system* (4.1)*. Suppose that there exists a mapping* $\phi : \mathbb{R}^n \to \mathbb{R}^N$ *with* $N \geq n$ *such that*

D1 *There exists a Schur stable matrix* $\mathcal{A} \in \mathbb{R}^{N \times N}$ *satisfying*

$$\phi(f(x)) = \mathcal{A}\phi(x). \tag{4.3}$$

D2 $\Phi(x) := \nabla\phi(x)^\top$ *has full column rank, and* $\Phi(x)^\top\Phi(x)$ *is uniformly bounded.*

Then system (4.1) *is contracting with the contraction metric* $\Phi(x)^\top P\Phi(x)$*, where* P *is any positive-definite matrix satisfying* $P - \mathcal{A}^\top P\mathcal{A} \succ 0$*. Conversely, if the system* (4.1) *is contracting with the metric* $M(x) \in \mathbb{R}^{n\times n}_{\succ 0}$*, and assuming that* f *is invertible and its inverse* f^{-1} *is continuous. Then, in any invariant compact set* $\mathcal{X} \subset \mathbb{R}^n$*, there exists a continuous Koopman mapping* $\phi : \mathbb{R}^n \to \mathbb{R}^n$ *verifying* **D1** *and* **D2**.

Proof. ($\Rightarrow$) From **D2** there exists a matrix $P = P^\top \succ 0$ satisfying the Lyapunov condition

$$P - \mathcal{A}^\top P\mathcal{A} \succ Q, \tag{4.4}$$

for some constant positive definite matrix $Q \succ 0$ without loss of generality. We define a new coordinate $z := \phi(x)$ in which the infinitesimal displacement $\delta z_t \in \mathbb{R}^N$ at time t is given by

$$\delta z_t = \Phi(x_t)\delta x_t, \tag{4.5}$$

where δx_t is an infinitesimal displacement in the x-coordinate.

The DT differential dynamics of x can be written as

$$\delta x_{t+1} = F(x_t)\delta x_t, \tag{4.6}$$

where $F(x) := \nabla f(x)^\top$.

Similarly, for z we have

$$\delta z_{t+1} = \mathcal{A}\Phi(x_t)\delta x_t = \Phi(x_{t+1})F(x_t)\delta x_t, \tag{4.7}$$

where we have used the relations $z_{t+1} = Az_t$ and $z_{t+1} = \phi(f(x_t))$ in their differential forms. Hence, we obtain

$$\Phi(x_{t+1})F(x_t) = \mathcal{A}\Phi(x_t).$$

Due to the full column rank of $\Phi(x)$ and (4.4), it follows that

$$\Phi(x_t)^\top(P - \mathcal{A}^\top P\mathcal{A})\Phi(x_t) \succ \Phi(x_t)^\top Q\Phi(x_t). \tag{4.8}$$

Then, by substituting (4.7), we have

$$\begin{aligned} &\Phi(x_t)^\top P\Phi(x_t) - F(x_t)^\top \Phi(x_{t+1})^\top P\Phi(x_{t+1})F(x_t) \\ \overset{(4.7)}{\succ}\ &\Phi^\top Q\Phi \succeq \frac{\lambda_{\mathtt{min}}(Q)}{\lambda_{\mathtt{max}}(P)}\Phi^\top P\Phi. \end{aligned} \tag{4.9}$$

Now since Φ has full column rank and $P \succ 0$, we have $M(x) := \Phi^\top P\Phi \succ 0$. Substituting into (4.9):

$$M(x_t) - F(x_t)^\top M(x_{t+1})F(x_t) \succ \beta M(x_t), \tag{4.10}$$

with $\beta := \lambda_{\mathtt{min}}(Q)/\lambda_{\mathtt{max}}(P)$. By selecting $Q = \rho P$ with $\rho \in (0,1)$, we have $\beta \in (0,1)$. This is exactly the contraction condition for the system (4.1) with respect to the metric M.

($\Leftarrow$) For the given DT system, from directly applying the Banach fixed-point theorem we conclude that there exists a unique fixed-point $x^\star \in \mathcal{X}$, i.e. $f(x^\star) = x^\star$.

First, we parameterise the unknown mapping $\phi(x)$ as $\phi(x) := x + T(x)$, with a new mapping $T(x)$ to be searched for. Then, the algebraic equation (4.3) becomes $T(f(x)) + f(x) = \mathcal{A}x + \mathcal{A}T(x)$. By fixing $\mathcal{A} = \nabla f(x^\star)^\top$, from the contraction assumption, we have $M(x^\star) - \mathcal{A}^\top M(x^\star)\mathcal{A} \succeq \beta M(x^\star)$, thus $\mathcal{A}$ is Schur stable. We have that

$$T(f(x)) = \mathcal{A}T(x) + H(x), \tag{4.11}$$

in which we have defined $H(x) := \mathcal{A}x - f(x)$. We make the key observation that the algebraic equation (4.11) exactly coincides with the one in the formulation of the Kazantzis-Kravaris-Luenberger observer for nonlinear DT systems (Brivadis et al., 2019, Eq. (7)). In our case, the function $H(x)$ is continuous and, following (Brivadis

et al., 2019, Theorem 2), we have a feasible solution[2] to (4.11) as follows:

$$T(x) = \sum_{j=0}^{+\infty} \mathcal{A}^j H(X(x, -j+1)), \tag{4.12}$$

with the definition

$$X(x, j) = \underbrace{f \circ f \circ \cdots \circ f}_{j \text{ times}}(x), \quad X(x, -j) = (f^{-1})^j(x)$$

for $j \in \mathbb{N}_+$.

Although $\phi^0(x) := x + T(x)$ with T defined above satisfies **D1** in the entire set $\mathcal{X}$, the condition **D2** may be not true. Hence, we need to modify the obtained $\phi^0(x)$. By considering the evolution of the trajectories in the x- and $z := \phi(x)$-coordinates respectively, we have

$$z(t_x) = \phi^0(x(t_x)) = \phi^0(X(x, t_x)) = \mathcal{A}^{t_x}\phi^0(x),$$

with $t_x \in \mathbb{N}_+$, thus satisfying $\phi^0(x) = \mathcal{A}^{-t_x}\phi^0(X(x, t_x))$. Then, we modify $\phi^0(x)$ into

$$\phi(x) := \mathcal{A}^{-t_x}[X(x, t_x) + T(X(x, t_x))] \tag{4.13}$$

with a sufficiently large $t_x \in \mathbb{N}_+$.

[2]The second assumption in Brivadis et al. (2019) holds true in any backward invariant compact set. Since contracting systems generally cannot guarantee such invariance, we may modify the dynamics as $x_{t+1} = \bar{f}(x_t)$ with

$$\bar{f}(x) = \begin{cases} f(x), & \text{if } \ x \in \mathtt{cl}(\mathcal{X}) \\ x, & \text{if } \ x \notin \mathcal{X}' \end{cases}$$

with $\mathcal{X} \subset \mathcal{X}'$, and then continue the analysis.

Finally, let us check conditions **D1** and **D2**. For the algebraic condition, we have

$$\begin{aligned}\phi(f(x)) &= \mathcal{A}^{-t_x}\phi^0(X(f(x),t_x)) \\ &= \mathcal{A}^{-t_x}\phi^0(f(X(x,t_x))) \\ &= \mathcal{A}^{-t_x}\cdot\mathcal{A}\phi^0(X(x,t_x)) \\ &= \mathcal{A}\phi(x)\end{aligned}$$

where we have used the fact

$$X(f(x),t_x) = \underbrace{f\circ f\circ\cdots\circ f}_{(j+1)\text{ times}} = f(X(x,t_x))$$

in the second equation. Therefore, $\phi(x)$ defined in (4.13) satisfies the algebraic equation (4.3). Regarding **D2**, let us study the Jacobian of $\phi(x)$ in (4.13), which is given by

$$\frac{\partial\phi}{\partial x}(x) = \mathcal{A}^{-t_x}\left[I + \frac{\partial T}{\partial x}(X(x,t_x))\right]\frac{\partial X}{\partial x}(x).$$

On the other hand, we have that $\nabla_x X$ is full rank and

$$H(x^\star) = 0, \quad \frac{\partial H}{\partial x}(x^\star) = 0,$$

as a result $\nabla T(x^\star) = 0$. If $t_x \in \mathbb{N}_+$ is sufficiently large, the largest singular value of $\nabla T(X(x,t_x))$ would be very small, and then the identity part of $\phi(x)$ will dominate $\nabla\phi(x)$. Hence, $\phi(x)$ is an injection for a large $t_x \in \mathbb{N}_+$.

□

4.3 Model Set

4.3.1 Parametrization of Koopman Matrix

There are many equivalent conditions for enforcing stability of linear systems, including the well-known Lyapunov inequality $P - \mathcal{A}^\top P\mathcal{A} \succ 0$ for some $P \succ 0$, and the

recently proposed parameterization in Gillis et al. (2020), which was used to train stable Koopman operators for fixed observables in Mamakoukas et al. (2020). However, solving optimization problems with these constraints in an efficient manner is non-trivial, especially when jointly searching for the observables.

In the following, an *unconstrained* parameterization of $\mathcal{A}$ is presented, which is a special case of the direct parameterization approach proposed in Revay et al. (2021b).

Proposition 4.1. *Consider the parametric matrix $\mathcal{A}(L, R)$ as*

$$\mathcal{A}(L, R) = 2(M_{11} + M_{22} + R - R^\top)^{-1} M_{21}, \tag{4.14}$$

where

$$M := \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = LL^\top + \epsilon I, \tag{4.15}$$

with ϵ a small positive constant. Then for any real-valued $L \in \mathbb{R}^{2N \times 2N}$ and $R \in \mathbb{R}^{N \times N}$, $\mathcal{A}_0 = \mathcal{A}(L, R)$ is a necessary and sufficient condition for $\mathcal{A}_0$ to be Schur stable.

Proof. (Sufficiency) Let $E = (M_{11} + M_{22} + R - R^\top)/2$, $F = M_{21}$ and $P = M_{22}$. Then we have $\mathcal{A}(L, R) = E^{-1}F$ and

$$M = \begin{bmatrix} E + E^\top - P & F^\top \\ F & P \end{bmatrix}. \tag{4.16}$$

It has been shown that $M \succ \gamma I$ for some $\gamma > 0$ is necessary and sufficient for $E^{-1}F$ to be Schur stable (Tobenkin et al., 2017). Hence if there exists L and R such that (4.14) and (4.15) hold, then by choosing a sufficiently small ϵ, we have $M \succ \gamma I$. Thus $\mathcal{A}(L, R)$ is Schur stable.

(Necessity) To prove necessity, it needs to be shown that there always exists L such that $M = LL^T + \epsilon I$ satisfies $M \succ \gamma I$. By the continuity of eigenvalues of a matrix with respect to its elements (Bhatia, 1996, Chapter 7), one has that $M - \epsilon I$ is

positive definite by choosing a sufficiently small positive $\epsilon \ll \gamma$. Hence the Cholesky factorization guarantees the existence of L such that $M - \epsilon I = LL^T$, as required. $\square$

4.3.2 Parametrization of Koopman Embedding

The observables are proposed to be parameterized as:

$$\phi(x) = Cx + \psi(x, \theta_{NN}), \tag{4.17}$$

where $C = [I_n, 0_{n\times(N-n)}]^\top$. The nonlinear part $\psi(x)$ can be any differentiable function approximator, parameterized by θ_{NN}. For brevity, the dependence on θ_{NN} is dropped in the notation. In this work, $\psi(x)$ is chosen to be a feedforward neural network due to its scalability, but any differentiable function approximator can be used.

The dimensionality of the observables N is a hyperparameter chosen by the user. For $N = n$, the observables will be of the same form as the constructive mapping $\phi^0(x) = x + T(x)$ in Theorem 4.1. To reconstruct the original state x from the observables, a separate function $\phi^L(z)$ has to be trained to compute the left-inverse of $\phi(x)$. Indeed, the left invertibility of ϕ is necessary for condition **D2**. This left inverse function is simply parameterized as another neural network $\phi^L(z, \theta_L)$.

Remark 1. *There are many possible parameterizations of the observables that are compatible with the framework, with Equation* (4.17) *being just the one chosen to mimic the constructive mapping from Theorem 4.1. For some parameterizations, the left inverse may be computed analytically and does not have to be modelled as a separate function. For example, if* $\phi(x) = [x^\top, \psi(x)^\top]^\top$*, then the left inverse is simply* $x = C\phi(x)$*, where* $C = [I, 0]$.

4.3.3 Overall Koopman Model

It may be helpful to think of our model as a linear system with an output $\hat{x}$ that is an estimate of the original state:

$$\begin{aligned} z(t) &= \mathcal{A}z(t-1), \\ \hat{x}(t) &= \phi^L(z(t)), \end{aligned} \tag{4.18}$$

where $z(0) = \phi(x_0)$ and $\phi^L(\cdot)$ is the left-inverse of $\phi(x)$. This system is equivalent to (4.2). In this form, it is clear that as long as $\mathcal{A}$ is stable and ϕ^L is uniformly bounded, then the output $\hat{x}$ will always converge to a single equilibrium.

4.4 Learning Framework

4.4.1 Optimization Formulation

To fit the model parameters $\{\theta_{NN}, \theta_L, L, R\}$ to data, the problem of minimizing the *simulation error* in the embedding space is considered:

$$J_{se} := \frac{1}{T}\sum_{t=0}^{T}\|\tilde{z}_t - z_t\|_2^2, \tag{4.19}$$

where $\tilde{z}_t = \phi(\tilde{x}_t)$, and $z_t = \mathcal{A}^t\phi(\tilde{x}_0)$. While the simulation error in x could be minimized instead, in practice this was found to produce poor results. The simulation error in z can still be large when the simulation error in x is small, hence the Koopman embedding may be fit poorly without including the excess coordinates of z in the minimization.

The complete optimization problem is:

$$\min_{\theta\in\Theta}\frac{1}{T}\sum_{t=0}^{T}\left\|\phi(\tilde{x}_t) - \mathcal{A}(L,R)^t\phi(\tilde{x}_0)\right\|_2^2 + \alpha J_{rec}. \tag{4.20}$$

The reconstruction loss J_{rec} is defined as

$$J_{rec} = \frac{1}{T}\sum_{t=0}^{T}\left\|\tilde{x}_t - \phi^L(\phi(\tilde{x}_t))\right\|_2^2. \tag{4.21}$$

Minimizing J_{rec} gives an approximate left-inverse ϕ_L for the Koopman mapping. The loss J_{rec} can be thought of as a penalty term that relaxes the constraint $x = \phi^L(\phi(x))\ \forall x$, and the constant α is a hyperparameter that determines the weighting of the penalty.

It is worth emphasizing two important properties of Problem (4.20). First, it is an unconstrained optimization problem. The parameter set Θ is the space of real numbers of the appropriate dimensionality. Second, there exists a differentiable mapping from the parameters θ to the objective for any choice of differentiable mapping ϕ_θ, e.g. using the parameterization (4.17) with φ_θ as a neural network.

These two properties enable finding a local optimum to Problem (4.20) using any off-the-shelf first-order optimizer in conjunction with an automatic differentiation (autodiff) toolbox. This significantly simplifies the implementation of the framework. Using an autodiff software package, one only needs to write code that evaluates the objective function at each iteration of the optimization process, and the gradients w.r.t. θ are automatically computed via the chain rule. In contrast, constrained problems such as the one proposed in Mamakoukas et al. (2020) require specialized algorithms to solve. Although the objective (4.20) is nonconvex, deep learning methods have been shown to be effective at finding approximate global minima for such problems; see (Roughgarden, 2020, Chapter 21) for example. It is worth noting that the model class is agnostic to the optimization problem. In fact, the model can be optimized for any differentiable objective function. This is another advantage of an unconstrained parameterization.

4.4.2 Implementation Details

The learning framework was implemented in PyTorch[3] and the Adam optimizer Kingma and Ba (2014) was used to solve Problem (4.20). The neural network parameters θ_{NN} and θ_L are initialized using the default scheme in PyTorch, while L, R, and b are initialized randomly from a uniform distribution.

Fast matrix power computation

As explained in Section 4.4.1, the only code that needs to be implemented for solving Problem (4.20) is the evaluation of the objective function, which is also the main computational bottleneck. In particular, repeatedly computing the matrix power $\mathcal{A}^t$ for the same $\mathcal{A}$ and many t's can be computationally inefficient. Here a simple trick is presented to speed up matrix power computations. Consider the eigendecomposition of $\mathcal{A}$ given by $V\Lambda V^{-1}$, where the columns of V are the eigenvectors and Λ is a diagonal matrix of the eigenvalues. Then it is clear that

$$\mathcal{A}^t = (V\Lambda V^{-1})^t = V\Lambda^t V^{-1} \tag{4.22}$$

for integer t. Notice that Λ^t can be computed element-wisely for each eigenvalue on the diagonal, which offers a significant speed-up over computing a matrix power. This trick assumes $\mathcal{A}$ is diagonalizable, but this can easily be verified in code and, if the condition is not satisfied, the original matrix power computation can be performed instead.

[3]https://github.com/pytorch/pytorch

4.5 Continuous-time Case

In this section, the continuous-time formulation of the learning framework is briefly presented. Recall the definition of the CT Koopman operator in Equation (2.32):

$$\tilde{\mathcal{K}}\phi(x(t)) = \nabla\phi \cdot f(x(t)).$$

4.5.1 Continuous-time Koopman Model

Definition 4.2 (CT Koopman model)**.** *The continuous-time Koopman model is given by:*

$$x(t) = \phi^L(\exp(A_\theta t)\phi(x_0)), \tag{4.23}$$

where the Koopman mapping ϕ is parameterized as in (4.17)*, and the finite-dimensional matrix A_θ is parameterized as:*

$$A_\theta = \frac{1}{2}(NN^\top + \epsilon I)^{-1}(-QQ^\top - \epsilon I + (R - R^\top)), \tag{4.24}$$

with parameters N, Q and R, and a small constant ϵ.

Similar to the discrete-time case, this is an unconstrained parameterization of all CT contracting models (Yi and Manchester, 2022, Theorem 1). Additionally, the A matrix defined by Equation (4.24) parameterizes all Hurwitz (CT stable) matrices, as shown by the following lemma.

Lemma 1. *A matrix A is Hurwitz if and only if there exist $N, Q, R \in \mathbb{R}^{n\times n}$ such that*

$$A = \frac{1}{2}(NN^\top + \epsilon I)^{-1}(-QQ^\top - \epsilon I + (R - R^\top)). \tag{4.25}$$

Proof. (Sufficiency) If there exist $N, Q, R \in \mathbb{R}^{n\times n}$ such that Equation (4.25) holds, then by choosing $P = (NN^\top + \epsilon I)^{-1}$, we have that

$$A^\top P + PA = -(QQ^\top + \epsilon I) \prec 0. \tag{4.26}$$

Hence P is a Lyapunov matrix and $\mathcal{A}$ satisfies the Lyapunov inequality, therefore $\mathcal{A}$ is Hurwitz.

(Necessity) If A is Hurwitz, then there exists $P = P^\top \succ 0$ and $W = W^\top \succ 0$ such that

$$A^\top P + PA + W = 0. \tag{4.27}$$

By the Cholesky decomposition, there always exists triangular $L \in \mathbb{R}^{n\times n}$ with strictly positive diagonal elements such that $P = LL^T$. For such L, there always exists some positive ϵ such that $L = N + \sqrt{\epsilon}I$, i.e.

$$P = NN^\top + \epsilon I. \tag{4.28}$$

The same argument can be made for $W = QQ^\top + \epsilon I$. The Lyapunov equation then becomes:

$$A^\top(NN^\top + \epsilon I) + (NN^\top + \epsilon I)A + QQ^\top + \epsilon I = 0. \tag{4.29}$$

Therefore there exists N, Q, R such that

$$A = \frac{1}{2}(NN^\top + \epsilon I)^{-1}(-QQ^\top - \epsilon I + (R - R^\top)) \tag{4.30}$$

satisfies Equation (4.29) as required. □

4.5.2 Learning Framework

Given full-state trajectory data $\{\tilde{x}_k\}_{k=0}^K$ with corresponding time $\{t_k\}_{k=0}^K$, the simulation error of the Koopman model is to be minimized. The optimization problem is

$$\min_{\theta\in\Theta} \frac{1}{K}\sum_{k=0}^{K} \|\phi_\theta(\tilde{x}_k) - \exp(A_\theta t_k)\phi_\theta(\tilde{x}_0)\|_2^2 + \alpha J_{rec}, \tag{4.31}$$

where J_{rec} is as defined in Equation (4.21). Problem (4.31) is an unconstrained optimization problem just like the discrete-time problem, hence a local minimum can be obtained using a first-order optimizer and an autodiff software package. A

Table 4.1 – Comparison of model sets for the proposed method and prior works.

Method	Learns observables or eigenfunctions	Continuous or discrete time	Stability constraint
Mamakoukas et al. (2020)	Neither	Discrete	✓
Takeishi et al. (2017)	Observables	Discrete	✗
Lusch et al. (2018)	Eigenfunctions	Discrete	✗
Pan and Duraisamy (2020)	Eigenfunctions	Continuous	✓
Our work	Observables	Both	✓

trick similar to that described in Section 4.4.2 can be used to compute the matrix exponential in the objective in (4.31).

Note that in terms of the data required, the only difference between the DT and CT learning frameworks is that the CT case requires the time corresponding to each data point. The CT problem can be useful to consider when the data is sampled at non-uniform time intervals, or when the sampling rate differs between the training and test scenarios.

4.6 Experiments

The proposed framework was validated on the LASA handwriting dataset (Khansari-Zadeh and Billard, 2011), which consists of human-drawn trajectories of various letters and shapes[4]. This dataset has been widely used as a benchmark for learning contracting dynamics in CT (Blocher et al., 2017; Khansari-Zadeh and Billard, 2011; Mohammad Khansari-Zadeh and Billard, 2014; Neumann et al., 2013; Ravichandar et al., 2017). In these results, discrete-time models were trained in order to compare them with existing DT Koopman learning frameworks. Contraction is an important constraint for this data set as unconstrained models can have spurious attractors (Khansari-Zadeh and Billard, 2011), leading to poor generalization to unseen initial conditions.

[4]https://cs.stanford.edu/people/khansari/download.html

For each shape in the dataset, a discrete-time model was trained to regulate to the desired equilibrium point from any initial condition. To prepare the data for learning DT models, splines were fitted to the trajectories and the datapoints were re-sampled at a uniform time interval. The state vector was chosen to be $\tilde{x}_t = [y_t^\top, \dot{y}_t^\top]^\top \in \mathbb{R}^4$, where y_t and $\dot{y}_t$ are the position and velocity vectors at time t. All data was scaled to the range $[-1, 1]$ before training. For each shape in the dataset, leave-one-out cross validation was performed. The trained models were tested by simulating them forward in time from the initial conditions of the trajectories in the test set, and computing the error between the simulated and ground truth trajectories. The ground truth test trajectories are plotted in Figure 4.4 as solid black lines for a subset of the shapes in the dataset.

The metric used to compare different methods was normalized simulation error (NSE), defined as:

$$NSE = \frac{\sum_{t=0}^{T} \|\hat{x}_t - \tilde{x}_t\|_2^2}{\sum_{t=0}^{T} \|\tilde{x}_t\|_2^2}, \tag{4.32}$$

where $\{\hat{x}\}_{t=0}^{T}$ is the simulated trajectory using the learned model, and $\{\tilde{x}\}_{t=0}^{T}$ is the true trajectory. In the following, the proposed framework is referred to as SKEL (Stable Koopman Embedding Learning).

4.6.1 Comparison to Other Koopman Matrix Parameterizations

The proposed unconstrained stable parameterization of the Koopman operator was compared against a constrained stable parameterization (SOC) (Mamakoukas et al., 2020), and a unconstrained parameterization without stability guarantees (LKIS) (Takeishi et al., 2017). The key differences of some recent frameworks are summarized in Table 4.1.

The SOC parameterization is given by $\mathcal{A} = S^{-1}OCS$, where S is invertible, O is orthogonal and C is positive-semidefinite with $\|C\| \leq 1$. A projected gradient descent method was used to solve the optimization problem. The LKIS parameterization is

$\mathcal{A} = Y_1 Y_2^\dagger$, where Y_1 and Y_2 are as defined in Equation (2.37), with parametric $\phi(x)$.

To make it a fair comparison, all other aspects of the optimization problem were kept the same, i.e. using simulation error as the optimization objective and using parametric observables of the form (4.17). The main point of comparisons was the parameterizations of the Koopman operator as the proposed framework is agnostic to choice of objective and observables, and these choices often depend on the particular application.

All instances of $\varphi(x)$ were fully-connected feedforward neural networks with ReLU (rectified linear units) activation functions, 2 hidden layers with 50 nodes each and an output dimensionality of 20. Hyperparameter values were chosen to be $\alpha = 10^3$ and $\epsilon = 10^{-8}$.

A boxplot of the normalized simulation error for the three methods is shown in Figure 4.2. It is clear that SKEL achieves the lowest median NSE on the test set with 95% confidence, while also having few outliers. The number of outliers not shown in the figure with NSE > 1 are: SKEL 1, LKIS 15, and SOC 0. From Figure 4.3, it can be seen that LKIS actually attains the lowest training error, but does not generalize to the test set as well as SKEL. This can be seen as a symptom of overfitting, and shows that the stability guarantees of SKEL have a regularizing effect on the model. With regards to SOC, it was observed that the constrained optimization problem would often converge to poor local minima, which is reflected in the relatively high training and test errors.

4.6.2 Robustness to Perturbations in Initial Conditions

A qualitative evaluation was performed to determine the robustness of the models to small perturbations in the initial condition of the test trajectory. Only SKEL and LKIS were compared as it was clear from Figure 4.2 that SOC underperformed in this setting. The results are plotted in Figure 4.4 and Figure 4.5. It can be seen that the SKEL models produce trajectories that converge to each other due to their contracting property. On the other hand, the LKIS trajectories sometimes diverge

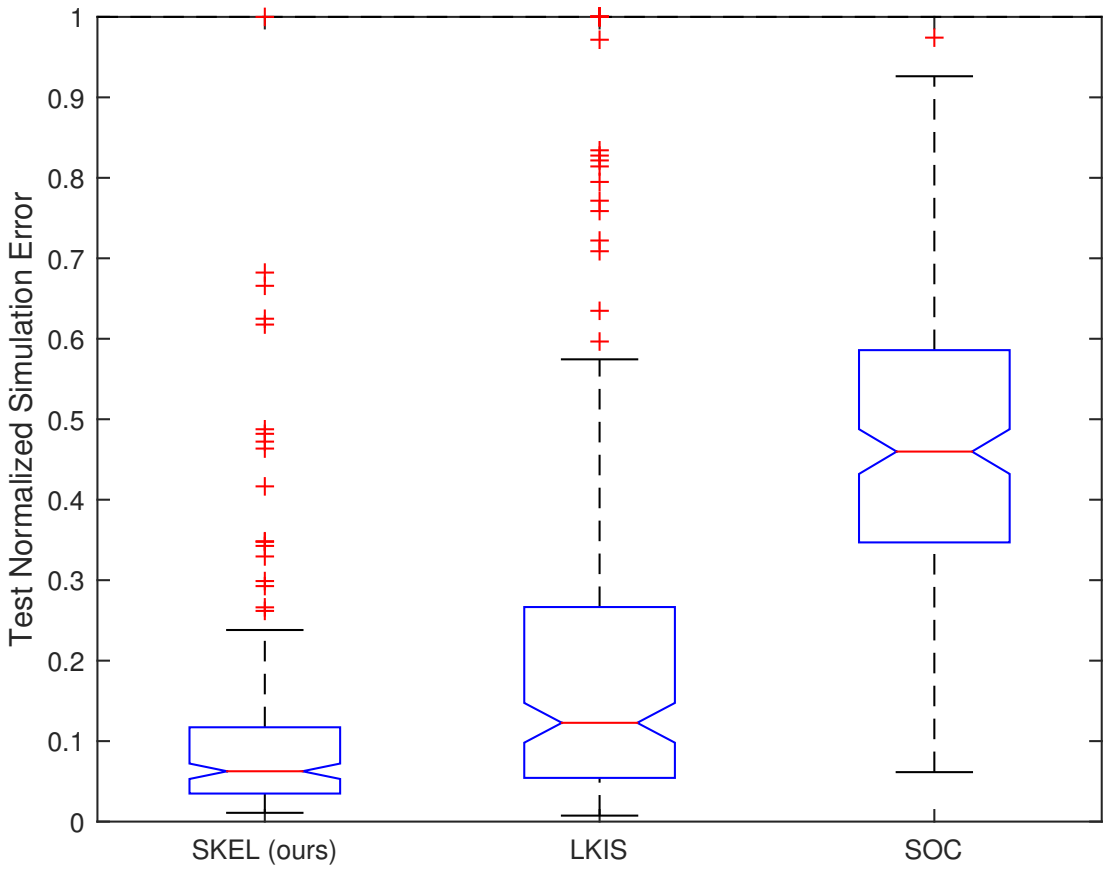


Figure 4.2 – Comparison of SKEL with other Koopman learning methods. Outliers were clipped for better visibility of boxes. Number of outliers with NSE > 1: SKEL 1, LKIS 15, and SOC 0

from each other with only small perturbations to the initial condition, indicating instability of the model.

4.7 Summary

A new class of Koopman models has been proposed, which are guaranteed to be contracting and can be optimized in an unconstrained manner. The regularizing effect of the contraction property of the model was demonstrated in a system identification problem.

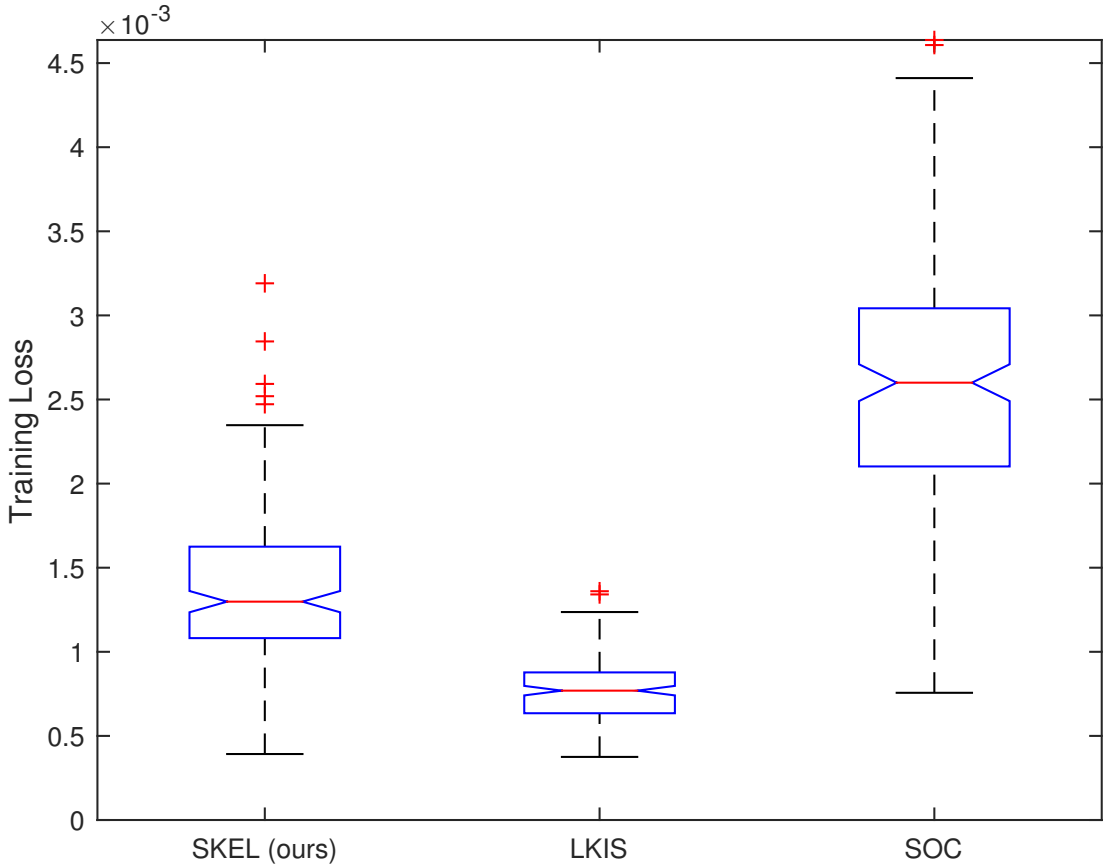


Figure 4.3 – Training loss for each method

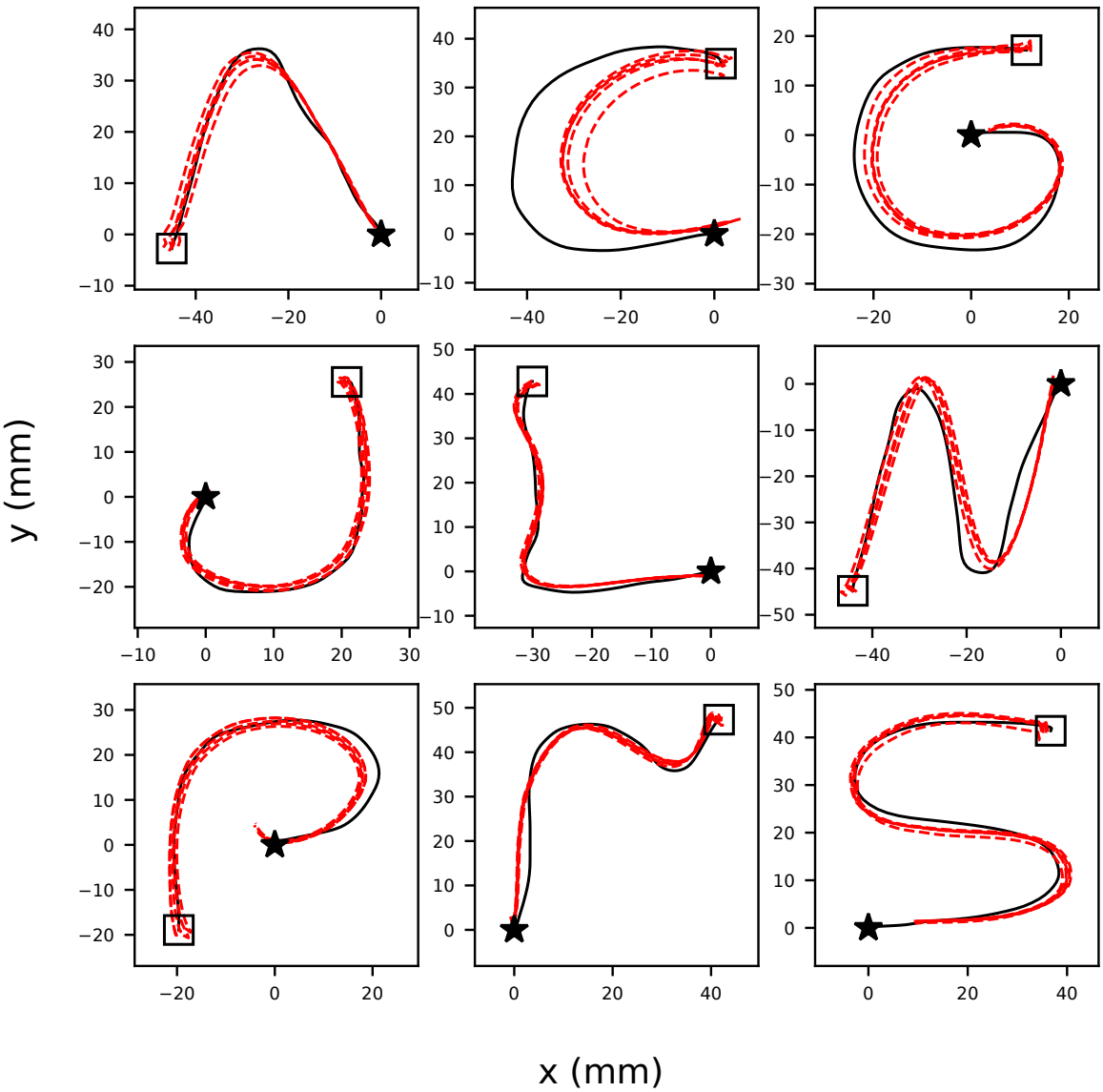


Figure 4.4 – Simulations of SKEL models on a subset of test data. Trajectories from the models are shown as red dotted lines, while the true trajectory is shown as a solid black line. Initial conditions were sampled from a square region of width 2 mm centered at the start point of the true trajectory. The target point is marked by a black star.

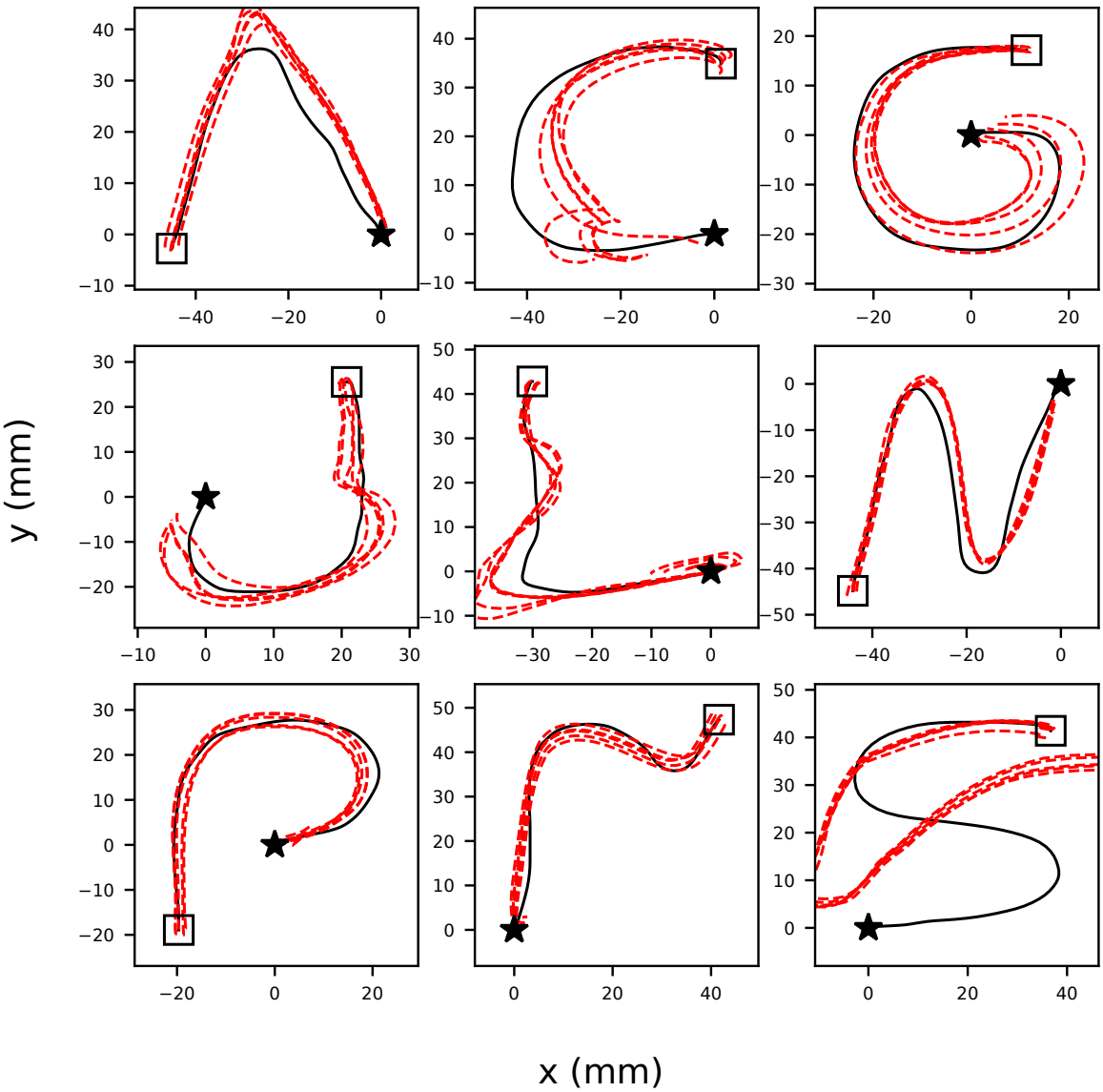


Figure 4.5 – Simulations of LKIS models on a subset of test data. Trajectories from the models are shown as red dotted lines, while the true trajectory is shown as a solid black line. Initial conditions were sampled from a square region of width 2 mm centered at the start point of the true trajectory. The target point is marked by a black star.

Chapter 5

Imitation Learning with Koopman Models

In this chapter, the problem of learning stabilizing controllers using Koopman models is considered. In particular, Koopman models are applied to the imitation learning (IL) problem: the problem of learning a controller from trajectories demonstrated by an expert. One well-studied and widely-used paradigm for IL frames it as a supervised learning problem and directly fits a mapping from state to control input. This is commonly referred to as behavioural cloning (Bain and Sammut, 1995). However, many of these IL methods do not consider the dynamics of the system in their model or optimization problem, which can lead to poor performance at test time (Ross and Bagnell, 2010).

Recently, some works have studied enforcing certain dynamical constraints, such as stability, on the controller during learning, under the assumption of known dynamics (Havens and Hu, 2021; Palan et al., 2020; Tu et al., 2022; Yin et al., 2021). In the present work, stability is used to regularize the imitation learning problem when the dynamics are unknown, to encourage the learned controller to be stabilising. The problem is approached from a system identification perspective, where the controller and dynamics are jointly learnt in order to guarantee stability of the learned model.

5.1 Problem Statement

Let us consider control-affine nonlinear systems of the form:

$$x_{t+1} = f(x_t) + g(x_t)u_t. \tag{5.1}$$

The objective of imitation learning is to learn a control policy that reproduces trajectories of System (5.1) demonstrated by an expert policy $u_t = k^\star(x_t)$. It is assumed that the expert satisfies a notion of optimality, in the sense of a control contraction metric.

Assumption 5.1. *There exists a control contraction metric for System* (5.1)*: that is, there exists a feedback controller $u_t = k(x_t)$ such that the closed loop system*

$$x_{t+1} = f(x_t) + g(x_t)k(x_t) := h(x_t) \tag{5.2}$$

is contracting.

The following quantifiable metric is defined to measure the discrepancy between the demonstrated controller $k^\star$ and the learned controller $\hat{k}$:

Definition 5.1 (Imitation Error)**.**

$$\sum_{t=1}^{T} \left\| \xi_t(k^\star, x_0) - \xi_t(\hat{k}, x_0) \right\|_2^2, \tag{5.3}$$

where $\xi_t(k, x_0)$ is the state of the system at time t from some initial condition x_0, induced by the controller k on the system.

This metric is challenging to optimize directly as an objective function, as $\xi_t(\hat{k}, x_0)$ can only be evaluated by executing the controller on the real system during learning, when the dynamics are unknown. Additionally, if gradient-based methods are used for optimization, the gradient of $\xi_t(\hat{k}, x_0)$ with respect to the parameters of $\hat{k}$ can be challenging to compute.

Behavioural cloning addresses these challenges by solving a surrogate problem instead, which can be stated as follows:

Problem 5.1 (Behavioural Cloning)**.** *Given a dataset of N state-control trajectories $\{\tilde{x}_t^i, \tilde{u}_t^i\}_{t=1:T}^{i=1:N}$, learn a mapping $k : \mathbb{R}^n \to \mathbb{R}^m$ that satisfies:*

$$\min_{\theta \in \Theta} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \left\| \tilde{u}_t^i - k_\theta(\tilde{x}_t^i) \right\|_2^2 + r(\theta), \tag{5.4}$$

where Θ is the parameter set of k and $r(\cdot)$ is a regularization function.

It is clear that if the controller is fit perfectly, i.e. zero error is attained for Equation (5.4), then the imitation metric will also be minimized. However, this is often unachievable in practice. Pfrommer et al. (2022) showed that the imitation error can be upper-bounded by a behavioural cloning loss if the closed-loop system satisfies an input-to-state stability assumption, which is also satisfied by contracting systems. However, in the general case, this upper bound may not hold. In the experiments here, it is shown that achieving a small behavioural cloning loss does not imply the imitation error will also be small, in particular when the learned controller is not stabilizing.

Since stability of the closed-loop system is an important consideration, a stability constraint is proposed to act as a regularizer for the behavioural cloning problem. The stability-regularized imitation learning problem is defined as follows:

Problem 5.2 (Stability-regularized imitation learning)**.** *Given a dataset of N state-control trajectories $\{\tilde{x}_t^i, \tilde{u}_t^i\}_{t=1:T}^{i=1:N}$, learn mappings $k : \mathbb{R}^n \to \mathbb{R}^m$, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^n$ that satisfy:*

$$\begin{aligned} \min_{\theta \in \Theta} \sum_{i=1}^{N} \sum_{t=1}^{T-1} & \left\| \tilde{u}_t^i - k_\theta(\tilde{x}_t^i) \right\|_2^2 + c_1 \left\| \tilde{x}_{t+1}^i - f_\theta(\tilde{x}_t^i) - g_\theta(\tilde{x}_t^i)\tilde{u}_t^i \right\|_2^2 \\ & + c_2 \left\| \tilde{x}_{t+1}^i - f_\theta(\tilde{x}_t^i) - g_\theta(\tilde{x}_t^i) k_\theta(\tilde{x}_t^i) \right\|, \\ & \textit{s.t. the system } x_{t+1} = f(x_t) + g(x_t)k(x_t) \textit{ is contracting.} \end{aligned} \tag{5.5}$$

In Problem (5.2), in addition to fitting the controller k, two other functions are mod-

elled, namely the open-loop dynamics f and input-to-state mapping g. The second term in the loss (5.5) represents the error of the open-loop dynamics model, while the third term represents the closed-loop model error. These two terms can be regarded as the regularization function $r(\theta)$ of behavioural cloning. The hyperparameters c_1 and c_2 determine the relative weightings of each loss term.

By solving Problem (5.2), a controller that is guaranteed to stabilize the *learned* dynamics model is obtained. This in turn means that if the dynamics model is fit perfectly, then the learned controller will stabilize the true system regardless of the error in controller fit. However, even if the learned model does not match the true system, the controller may still be stabilising if it is sufficiently robust to model mismatch. This idea has previously been explored for linear systems and LQR controllers (Dean et al., 2020), where the authors derive bounds on the LQR cost from error bounds on the transition matrices $[A, B]$, for a robust controller obtained using system level synthesis (Wang et al., 2019).

To enforce the contraction condition for the closed loop dynamics, the stable Koopman model proposed in Chapter 4 is built on.

5.2 Stabilizable Koopman Models

In this section, the model set used to represent k, f and g in Problem (5.2) is presented. The objective is to find a model class that implicitly satisfies the contraction constraint in Problem (5.5), so that it can be optimized in an unconstrained manner.

To do so, let us again consider a Koopman embedding that lifts System (5.1) to a linear manifold. It has already been shown in Section 4.2 that for a contracting system $x_{t+1} = h(x_t)$, there exists a mapping $\phi : \mathbb{R}^n \to \mathbb{R}^n$ that satisfies:

$$\phi \circ h(x_t) = A\phi(x_t), \tag{5.6}$$

where $A \in \mathbb{R}^{n \times n}$ is Schur stable.

For a controlled system (5.1), the dynamics of $z := \phi(x_t)$ can be written as (Iacob et al., 2022b):

$$z_{t+1} = Az_t + B(x_t, u_t)u_t, \tag{5.7}$$

where

$$B(x_t, u_t) = \left(\int_0^1 \frac{\partial \phi}{\partial x}(f(x_t) + \lambda g(x_t)u_t)d\lambda \right) g(x_t). \tag{5.8}$$

The fact that the dynamics (5.7) are not affine with respect to u and B is not constant both complicate our analysis. As such, an injective transformation of the control input $u = \alpha(x, v)$ is considered such that the system becomes linear time-invariant:

$$z_{t+1} = Az_t + Bv_t. \tag{5.9}$$

In the following, a discrete-time version of (Yi and Manchester, 2022, Proposition 4) is presented. It is shown that if the Koopman system (5.9) exists and is stabilizable, then the nonlinear system (5.1) admits a control contraction metric (CCM).

5.2.1 Koopman Stabilizability

Theorem 5.1. *For system* (5.1)*, if there exist injective mappings* $\phi(x)$ *and* $u = \alpha(x, v)$ *such that*

$$\phi(f(x_t) + g(x_t)\alpha(x_t, v_t)) = A\phi(x_t) + Bv_t, \tag{5.10}$$

where the dynamics of $z := \phi(x)$ *are*

$$z_{t+1} = Az_t + Bv_t, \tag{5.11}$$

then

1. *any stabilizing controller* $v = -Kz$ *for* (5.11) *renders the closed-loop system*

$$x_{t+1} = f(x_t) + g(x_t)\alpha(x_t, -K\phi(x_t)) \tag{5.12}$$

contracting,

2. *$M(x_t) = \Phi(x_t)^\top P \Phi(x_t)$ is a control contraction metric for system* (5.1).

Proof. First redefine system (5.1) in terms of the new input signal v_t as

$$x_{t+1} = f(x_t) + g(x_t)\alpha(x_t, v_t) := \bar{f}(x_t, v_t). \tag{5.13}$$

Substituting (5.13) and the stabilizing feedback controller $v = -K\phi(x)$ into (5.10):

$$\phi(\bar{f}(x_t, -K\phi(x_t))) = (A - BK)\phi(x_t). \tag{5.14}$$

Taking its partial derivative w.r.t. x,

$$\Phi(x_{t+1})(\bar{F}(x_t) - \bar{G}(x_t)\bar{K}(x_t)) = (A - BK)\Phi(x_t), \tag{5.15}$$

where $\Phi(x) = \frac{\partial \phi}{\partial x}(x)$, $\bar{F}(x) = \frac{\partial \bar{f}}{\partial x}(x)$, $\bar{G}(x) = \frac{\partial \bar{f}}{\partial v}(x)$ and $\bar{K}(x) = K\Phi(x)$.

Now the existence of a stabilizing controller for (5.9) implies the existence of a matrix $P = P^\top \succ 0$ such that

$$P - (A - BK)^\top P(A - BK) \succ Q, \tag{5.16}$$

for some $Q \succ 0$. Since $\phi(x)$ is assumed to be injective, $\Phi(x)$ will have full column rank and hence we can write:

$$\Phi(x_t)^\top (P - (A - BK)^\top P(A - BK))\Phi(x_t) \succ \Phi(x_t)^\top Q \Phi(x_t). \tag{5.17}$$

By choosing the metric:

$$M(x) := \Phi(x)^\top P \Phi(x), \tag{5.18}$$

and substituting (5.15) into (5.17), we obtain

$$\begin{aligned} &M(x_t) - (\bar{F}(x_t) - \bar{G}(x_t)\bar{K}(x_t))^\top M(x_{t+1})(\bar{F}(x_t) - \bar{G}(x_t)\bar{K}(x_t)) \\ \succ &\Phi(x_t)^\top Q \Phi(x_t) \succeq \frac{\lambda_{min}(Q)}{\lambda_{max}(P)} M_t. \end{aligned} \tag{5.19}$$

Without loss of generality, we can choose $Q = \rho P$ with $\rho \in (0, 1)$ such that $0 < \frac{\lambda_{min}(Q)}{\lambda_{max}(P)} < 1$. Then Equation (5.19) corresponds to the discrete CCM condition (2.23), and $v = -Kz$ is a CCM controller for (5.1) as required. □

The Koopman control model defined by Equation (5.10) is closely connected to feedback linearization, a nonlinear control method that finds a state and control input transformation that linearizes the dynamics. Indeed, the existence of a CCM is necessary for feedback linearizability (Manchester and Slotine, 2017). However, unlike in feedback linearization, the state embedding of the Koopman model may be higher-dimensional than the original state.

5.3 Model Set

The parameters of the Koopman control model consist of the Koopman matrices (A, B) and parameters of the mappings $z = \phi(x)$ and $u = \alpha(x, v)$. The statements in Theorem (5.1) suggest several implicit constraints on the parameters, which should be considered when searching over the parameter space. Namely, these constraints are

- (A, B) is stabilizable.
- $\phi(x)$ is injective.
- $\alpha(x, v)$ is bijective with respect to v.
- For any stabilizing controller $v = Kz$, the fixed point of the linear system (5.9) is $(z^\star, v^\star) = (0, 0)$, hence $\phi(x^\star) = 0$ and $u^\star = \alpha(x^\star, 0)$.

In the following, parameterizations are proposed that satisfy these constraints.

5.3.1 Parameterization of Stabilizable Triples

Previous works on stable imitation learning have used linear matrix inequality (LMI) constraints to guarantee stability (Havens and Hu, 2021; Palan et al., 2020; Yin et al., 2021), however such constrained optimization problems become difficult to solve when the dynamics have to be jointly estimated. In the following, an *unconstrained* parameterization of the triple $\{\hat{A}, \hat{B}, \hat{K}\}$ is proposed such that $\hat{A} - \hat{B}\hat{K}$ is guaranteed to be stable.

First note that any open-loop transition matrix A can be re-written in terms of the closed loop response as $A = A_{\text{CL}} + BK$. Thus the stable matrix parameterization proposed in Section 4.3.1 can be used to represent the closed-loop matrix.

The following parameterization for A is proposed.

Proposition 5.1 (Stabilizable Parameterization)**.**

$$A(L, R, B) = A_{CL} + \frac{1}{2} B B^\top M_{22} A_{CL}, \tag{5.20}$$

with

$$A_{CL} = 2(M_{11} + M_{22} + R - R^\top)^{-1} M_{21} \tag{5.21}$$

and

$$M := \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = LL^\top + \epsilon I, \tag{5.22}$$

It can be shown that the parameterization (5.20) is sufficient for guaranteeing that (A, B) is stabilizable.

Lemma 2. *The pair $(\mathcal{A}, \mathcal{B})$ is stabilizable if $\mathcal{A} = A(L, R, B)$ and $\mathcal{B} = B$.*

Proof. Given $\mathcal{A} = A(L, R, B)$ and choosing $K = \frac{1}{2} B^\top M_{22} A_{\text{CL}}$, we have

$$\mathcal{A} - \mathcal{B}K = A_{\text{CL}} = 2(M_{11} + M_{22} + R - R^\top)^{-1} M_{21}, \tag{5.23}$$

which is Schur stable by Proposition 4.1. Hence $(\mathcal{A}, \mathcal{B})$ is stabilizable by definition. □

The sub-block M_{22} is a Lyapunov matrix P for the closed loop system, as it satisfies $M_{22} - (A - BK)^\top M_{22}(A - BK) \succ 0$. The controller gain matrix $K = \frac{1}{2}B^\top P A_{\text{CL}}$ is an LQR controller gain, which can be seen from rearranging the DT infinite-horizon LQR controller:

$$K = (R + B^\top PB)^{-1}B^\top PA, \tag{5.24}$$

using $A_{\text{CL}} = A - BK$ and having the control cost matrix $R = 2I$, where I is the identity matrix. Although limiting the model parameterization to LQR controllers is restrictive, it can be beneficial to learn an LQR controller as they have strong robustness properties. As such, the controller may be robust to model mismatch, which can be modelled as a disturbance signal on the dynamics.

5.3.2 Parameterization of ϕ

The same parameterization of ϕ as in the stable Koopman models is used:

$$\phi(x) = Cx + \psi(x), \tag{5.25}$$

where $C = [I_n, 0_{n\times(N-n)}]^\top$. The nonlinear part $\psi(x)$ is a feedforward neural network, parameterized by θ_{NN}.

To reconstruct the original state x from the observables, a separate function $\phi^L(z)$ is parameterized to compute the left-inverse of $\phi(x)$.

5.3.3 Parameterization of α

There are many possible ways to parameterize the bijective mapping α. In the numerical experiments here, two choices of parameterizations are investigated.

The first parameterization is setting $u = v$, that is, α is fixed and not learned. This means that the control input u will be affine with respect to the lifted Koopman state z, that is,

$$z_{t+1} = Az_t + Bu_t.$$

This form of control-affine Koopman model will only yield an exact representation of the nonlinear system if the matrix B in (5.8) is constant, which is difficult to achieve in general. Nevertheless, this representation has been widely used in learning-based control (Han et al., 2020; Kaiser et al., 2021; Korda and Mezić, 2018), as it readily admits linear control design methods.

The second parameterization is to have $u = \alpha(x, v)$ in the form of an affine coupling layer (Dinh et al., 2017):

$$u = v \odot \exp(s(x)) + t(x), \tag{5.26}$$

where $\odot$ denotes the Hadamard product, and $s(x)$ and $t(x)$ can be arbitrary function approximators. Equation (5.26) has an analytical inverse $v = (u - t(x)) \odot \exp(-s(x))$. To ensure $u^\star = \alpha(x^\star, 0)$, $t(x)$ is represented as $t(x) - t(x^\star) + u^\star$. This parameterization has the same form as the input transformation found in feedback linearization, which is typically written as $u = a(x) + b(x)v$ for some nonsingular function b (Khalil, 2002).

5.3.4 CCM Controller

Putting the components of the model together, one obtains an imitation controller of the form

$$u_t = \alpha(x_t, B^\top P A_{\mathrm{CL}} \phi(x_t)). \tag{5.27}$$

Equation (5.27) is henceforth referred to as the learned CCM controller.

Additionally, an open-loop and closed-loop dynamics model are also learned as byproducts of the stability-regularized imitation learning problem. The OL model is given by:

$$\begin{aligned} z_0 &= \phi(x_0) \\ z_{t+1} &= A z_t + B \alpha^{-1}(x_t, u_t) \\ x_t &= \phi^L(z_t). \end{aligned} \tag{5.28}$$

The CL model is identical to the stable Koopman model (4.18). These dynamics models can be used for future state prediction and trajectory planning, although

they are not utilized in the experiments in this chapter. The OL model can also be used to construct new controllers, e.g. using the SOS procedure in Wei et al. (2021).

5.3.5 Connection to Inverse Optimal Control

Inverse optimal control (IOC) poses the question: given an optimal controller, what cost does it optimize (Kalman, 1964)?

Using the proposed model, it is possible to construct a cost function that is optimized by the learned controller.

Firstly, the learned control contraction metric

$$M(x_t) = \Phi(x_t)^\top P \Phi(x_t) \tag{5.29}$$

is a Lyapunov function since it satisfies the Lyapunov condition (5.19).

Secondly, since the controller (5.27) is an infinite-horizon LQR controller in Koopman coordinates, there exist corresponding cost matrices Q and R such that the controller minimizes

$$\sum_{t=0}^{\infty} c(x_t, u_t) = \sum_{t=0}^{\infty} z_t^\top Q z_t + u_t^\top R u_t, \tag{5.30}$$

where $z_t = \phi(x_t)$. The state cost matrix Q can be computed using the model parameters as:

$$Q = P - \frac{1}{2} A^\top P A_{CL} - \frac{1}{2} A_{CL}^\top P A, \tag{5.31}$$

where A and A_{CL} are as defined in Proposition 5.1, and $P = M_{22}$. It is straightforward to derive Equation (5.31) from the DT algebraic Riccati equation.

For the particular choice of gain matrix $K = \frac{1}{2} B^\top P A_{\mathrm{CL}}$, the control cost matrix R is fixed as $2I$. However, this is generally not restrictive, as any scaling of the control input can be 'absorbed' by α, or the control data can be scaled before learning so that all dimensions are standardized.

Since the cost function $c(x_t, u_t)$ is independent of the dynamics, new controllers can then be optimized for different dynamics models using the learned cost function, as is

commonly done in inverse reinforcement learning (Fu et al., 2017). In a robot learning context, this can be useful when the expert and learner have different embodiments, e.g. when the expert is a human demonstrator.

5.4 Learning Framework

Given the stabilizable model set proposed in Section 5.2, let us now discuss how this model set can be used to solve Problem 5.2. Similar to the autonomous case, the stability constraint can simply be removed in Equation (5.5), and an *unconstrained* optimization problem is obtained:

$$\begin{aligned}\min_{\theta\in\Theta}\sum_{i=1}^{N}\sum_{t=1}^{T-1} c_1\left\|\tilde{z}_{t+1}^i - A\tilde{z}_t^i - B\tilde{v}_t^i\right\|_2^2 + c_2\left\|\tilde{z}_{t+1}^i - A_{\mathrm{CL}}\tilde{z}_t^i\right\|_2^2 \\ + \left\|\tilde{v}_t^i - \frac{1}{2}B^\top P A_{\mathrm{CL}}\tilde{z}_t^i\right\|_2^2 + J_{\mathrm{rec}},\end{aligned} \tag{5.32}$$

where $\tilde{z}_t^i = \phi(\tilde{x}_t^i)$ and $\tilde{v}_t^i = \alpha^{-1}(\tilde{x}_t^i, \tilde{u}_t^i)$. The reconstruction loss J_{rec} is defined as

$$J_{rec} = \frac{1}{T}\sum_{t=0}^{T}\left\|\tilde{x}_t - \phi^L(\phi(\tilde{x}_t))\right\|_2^2. \tag{5.33}$$

The parameter set Θ consists of $\{L, R, B, \theta_{NN}\}$. Just like in the autonomous case, the optimization problem (5.32) can be solved via first-order methods using automatic differentiation software.

Notice that the first term in Eq.(5.32) can be rewritten as:

$$c_1\left\|\tilde{z}_{t+1}^i - A_{\mathrm{CL}}\tilde{z}_t^i - B(\tilde{v}_t^i - \frac{1}{2}B^\top P A_{\mathrm{CL}}\tilde{z}_t^i)\right\|_2^2 \tag{5.34}$$

which is the difference between the closed-loop error and controller error scaled by B. Thus the open-loop error can be seen as a coupling term for the problems of fitting a controller and closed-loop model. Without this term, i.e. if $c_1 = 0$, we are essentially solving two decoupled problems (even though K shares parameters with A_{CL}, B is

a free parameter). This suggests that c_1 should be chosen to be relatively large in order to fit the open-loop dynamics accurately. Otherwise, the open-loop error may appear to be small simply because the closed-loop and controller errors are small.

5.4.1 Linear Case

In the case when the dynamics of the system are linear, the optimization problem is simplified since the mapping ϕ is just the identity map and the control input is not transformed, i.e. $u = v$. The problem then becomes:

$$\min_{A,B,K} \sum_{i=1}^{N} \sum_{t=1}^{T-1} c_1 \left\| \tilde{x}_{t+1}^i - A\tilde{x}_t^i - B\tilde{u}_t^i \right\|^2 + c_2 \left\| \tilde{x}_{t+1}^i - A_{\mathtt{CL}}\tilde{x}_t^i \right\|^2 + \left\| \tilde{u}_t^i - K\tilde{x}_t^i \right\|^2. \tag{5.35}$$

In the linear case, an additional assumption on the rank of the data is required for the learning problem to be well-posed. This can be stated as:

Assumption 5.2 (Persistence of excitation)**.** *Given trajectory data* $\{\tilde{x}_t^i, \tilde{u}_t^i\}_{t=1:T}^{i=1:N}$*, define the data matrices*

$$X_0 = [\tilde{x}_0^1, \ldots, \tilde{x}_0^N],$$

$$U = \left[vec([\tilde{u}_1^1, \ldots, \tilde{u}_T^1]), \ldots, vec([\tilde{u}_1^N, \ldots, \tilde{u}_T^N]) \right].$$

These matrices should satisfy:

$$rank \left(\left[X_0^\top, U^\top \right]^\top \right) = n + mT \tag{5.36}$$

One has to be careful when U is generated by a linear state feedback controller, since U will be linearly dependent on X_0 and the rank condition will not be satisfied. In this case, one can add an excitation signal to the controller to satisfy the rank condition.

5.5 Experiments

Numerical experiments were performed on two simulated systems: an unstable linear system and a nonlinear 2 degrees-of-freedom planar robot arm.

5.5.1 Linear Example

The learning framework was evaluated on the following linear system:

$$A = \begin{bmatrix} 1.01 & 0.01 & & \text{\Large 0} \\ 0.01 & \ddots & \ddots & \\ & \ddots & \ddots & 0.01 \\ \text{\Large 0} & & 0.01 & 1.01 \end{bmatrix}, \; B = I_n. \tag{5.37}$$

This system corresponds to an unstable graph Laplacian system, based on the example used in Dean et al. (2020). The nodes form a weakly connected chain, and each node can be controlled directly. For this example, the system dimensionality was chosen to be $n = 20$.

To generate data for learning, a controller of the form

$$u_t = Kx_t + w_t \tag{5.38}$$

was used, where K is the gain obtained by solving the LQR problem with cost $Q = 10^{-3} I_n,\ R = I_n$. The excitation signal $w_t \sim \mathcal{N}(0, I)$ ensures the system is identifiable, and is unknown to the learning algorithm. Additionally, process noise $p_t \sim \mathcal{N}(0, \sigma_p^2 I)$ ($\sigma_p = 0.1$) was injected into the dynamics. The data is then generated by recursively computing:

$$\begin{aligned} \tilde{x}_1 &\sim \mathcal{U}(-10, 10), \\ \tilde{x}_{t+1} &= A\tilde{x}_t + B\tilde{u}_t + p_t, \\ \tilde{u}_t &= K\tilde{x}_t + w_t, \end{aligned} \tag{5.39}$$

for $t = 1, 2, \ldots, T$, $T = 10$, repeated for $N = 20$ trajectories.

For these experiments, the learning framework was compared against behavioral cloning, which only fits a mapping from the state to control input, without considering the dynamics of the system. In the linear case, this amounts to solving the least squares problem:

$$K_{LS} = \arg\min \sum_{i=1}^{N} \sum_{t=1}^{T-1} \left\| \tilde{u}_t^i - \hat{K} \tilde{x}_t^i \right\|^2, \tag{5.40}$$

which gives

$$K_{LS} = U X^\top (X X^\top)^{-1}, \tag{5.41}$$

where $X = \left[\tilde{x}_1^1, \ldots, \tilde{x}_{T-1}^N\right] \in \mathbb{R}^{n \times N(T-1)}$ and $U = \left[\tilde{u}_1^1, \ldots, \tilde{u}_{T-1}^N\right] \in \mathbb{R}^{m \times N(T-1)}$.

It has been shown that in Celi et al. (2022) that the least-squares solution (5.41) converges to the true K matrix in the limit as $T \to \infty$. However, in the low-data regime, the least-squares estimate can be very noisy and possibly unstable, as will be shown in the experiments.

A comparison was also made against a prior stability-constrained imitation learning method (Havens and Hu, 2021) that assumes exact knowledge of the system matrices A and B. Their method was applied to this problem setting by first estimating $[A, B]$ via least squares as:

$$[A, B] = Y X_U^\top (X_U X_U^\top)^{-1}, \tag{5.42}$$

where $Y = \left[\tilde{x}_2^1, \ldots, \tilde{x}_T^N\right]$ and $X_U = [X^\top, U^\top]^\top$. In the following, this method is denoted as 'AB'. However, it should be noted that this method of estimating the open-loop dynamics will not work in the nonlinear case. As such, the algorithm of Havens and Hu (2021) is not readily extendable to the nonlinear case.

The following metrics were used as a quantitative comparison of the different methods and different weightings of each loss term in the proposed method.

Imitation error (normalized simulation error)

$$\sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{x}_t^i - (A + B\hat{K})^{t-1} \tilde{x}_1^i \right\|^2 \Big/ \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{x}_t^i \right\|^2 . \tag{5.43}$$

Controller error

$$\left\|K - \hat{K}\right\|_{\mathtt{op}}^2 \Big/ \|K\|_{\mathtt{op}}^2 \,. \tag{5.44}$$

Closed-loop stability

$$\rho(A + B\hat{K}). \tag{5.45}$$

The metric (5.43) is the normalized simulation error (NSE) for a set of test trajectories measured against trajectories induced by the learned controller on the true system. It was chosen for $T_{\text{test}} \gg T$, i.e. the time horizon for testing is much longer than the training data. For testing, the noise signals p_t and w_t are set to zero, so that the error will be zero when the controller is fit perfectly. The metric (5.45) is the spectral radius of $A + B\hat{K}$, which is the closed-loop system induced by the learned controller on the real system. The system is stable if $\rho < 1$. The true spectral radius of the system is 0.969.

Effect of c_1

The first aim is to investigate how the weighting on the fit of the open-loop dynamics affects the solution. As mentioned in the problem formulation, fitting the open-loop dynamics is the proposed heuristic for encouraging the learned controller to be stabilizing for the true system. As such, it needs to be verified if increasing c_1 would increase the likelihood of learning a stabilizing controller.

Fifty runs of the experiment were performed using the procedure described before, with the only difference between each trial being the initial conditions and noise signals w_t and p_t.

Figure 5.1 shows histograms of the maximum eigenvalue of the closed-loop matrix $A + B\hat{K}$, i.e. the estimated K applied to the true system. It can be seen that as the weighting c_1 on the open-loop loss increases, the distribution of closed-loop eigenvalues shift towards stability and the true value.

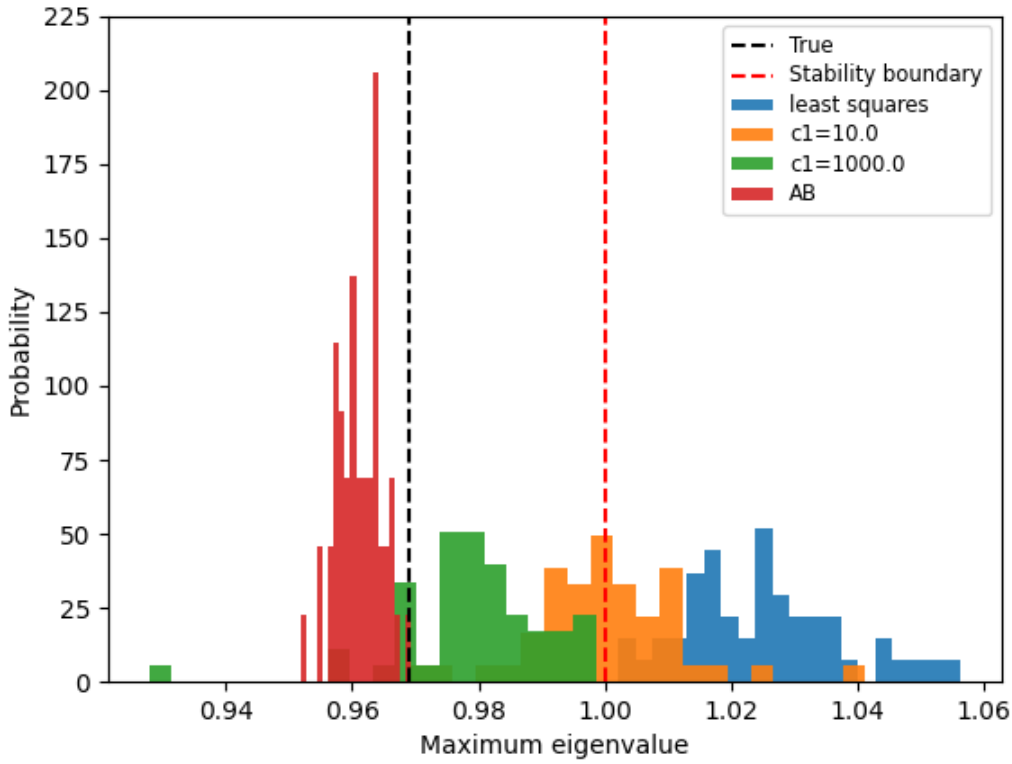


Figure 5.1 – Histogram of the maximum eigenvalue of $A + B\hat{K}$. The histogram shows how the distribution of eigenvalues shifts as c_1 increases. The red vertical line shows the boundary of stability; if the maximum eigenvalue is to the right of the red line, then the system is unstable. The black vertical line shows the true maximum eigenvalue of the closed-loop system. 'AB' shows the distribution for the method of Havens and Hu (2021), while 'least squares' shows the distribution for the baseline method (5.41).

A similar trend can be observed in the imitation error, also known as the normalized simulation error, shown in Figure 5.2. All of the least-squares controllers are unstable, whereas using the proposed parameterization, increasing c_1 produces more stabilizing controllers. The method of Havens and Hu (2021) ('AB') had slightly larger error.

On the other hand, Figure 5.3 shows that least squares attains the lowest error in controller fit, which is directly correlated with the behavioural cloning loss in Problem (5.4), while the other methods have relatively large K error. Despite that, least squares performed the worst by the other metrics. This suggests that minimising

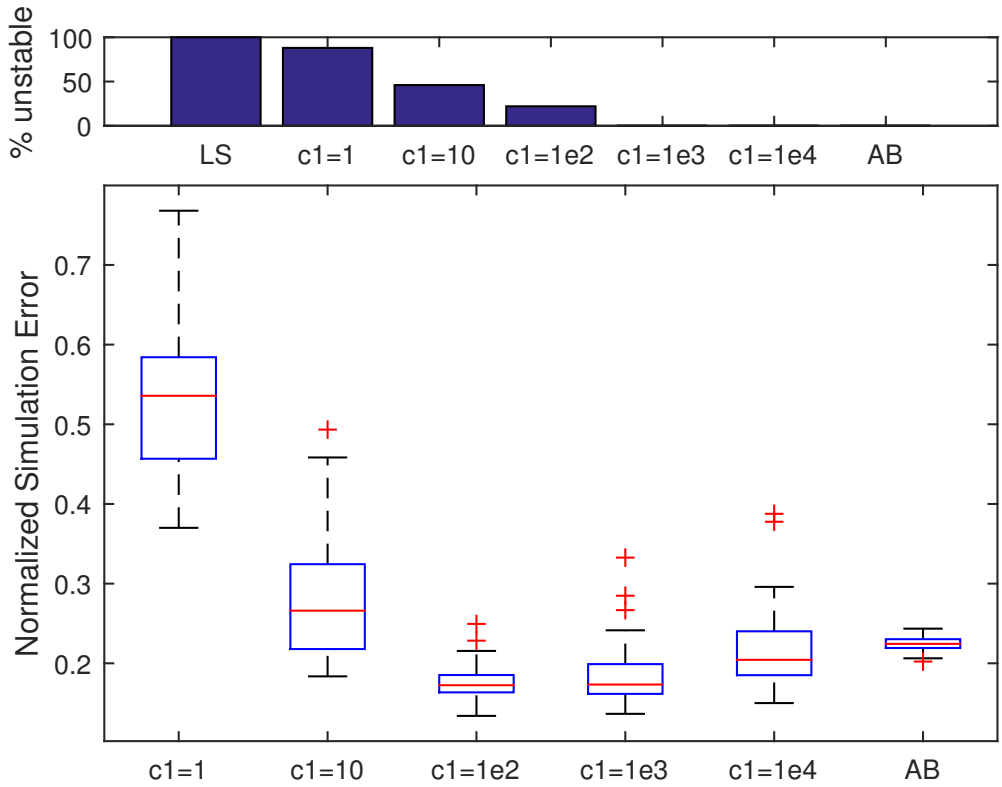


Figure 5.2 – Top: Bar graph showing percentage of unstable controllers. The plot is simply a different way of visualising Figure 5.1. Bottom: Boxplot of normalized simulation error (5.43) for stable controllers. No boxplot for least squares is shown as all LS controllers were unstable.

the K error alone, as in BC, may not result in the best performance in terms of the imitation error.

Scalability

The scalability of the algorithm was evaluated by measuring computation time to convergence of the optimization problem. For the proposed method, this corresponds to the time taken to compute the gradient and update the parameters. In comparison, the projected gradient descent (PGD) algorithm proposed by Havens and Hu (2021) requires solving a semidefinite program at each iteration.

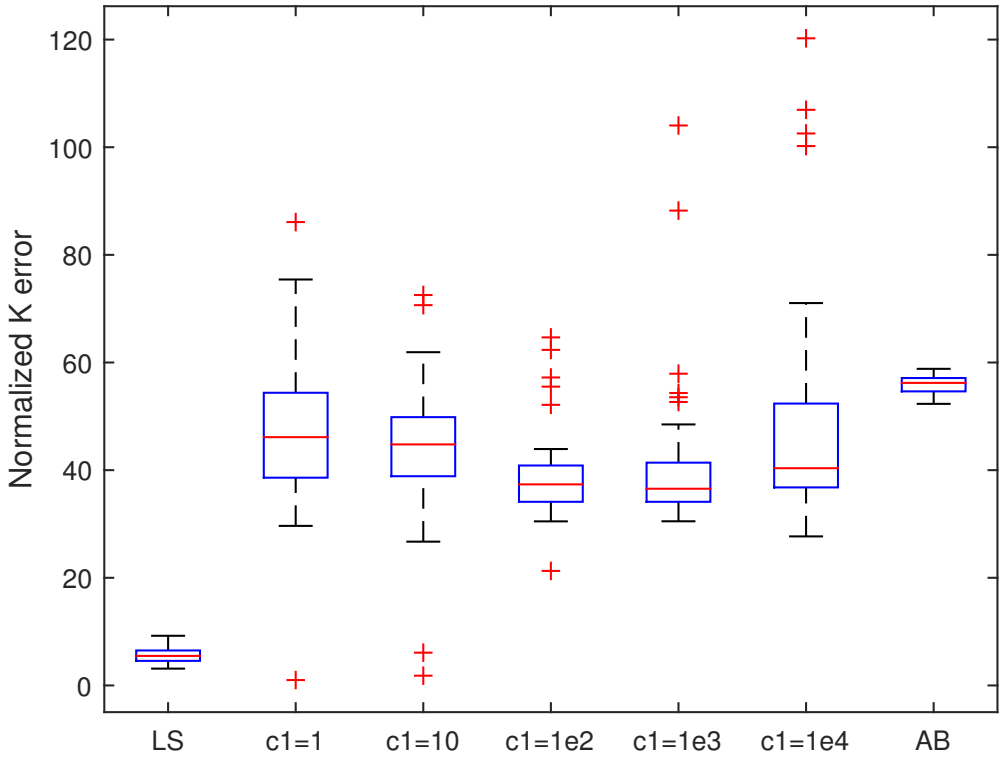


Figure 5.3 – Boxplot of normalized 2-norm error (5.44) of $\hat{K}$ matrix.

Figure 5.4 shows the computation time per iteration of each method and Figure 5.5 shows the total time to convergence. The slopes of the lines of best fit reveal how the computation time scale with the dimensionality of the system. As can be seen from Figure 5.4, the PGD algorithm scales more than cubically in terms of the computation time per iteration, while the proposed method only scales super-linearly. A similar trend is also seen for the total time to convergence. These results demonstrate the advantage in scalability of optimizing an unconstrained model compared to constrained models.

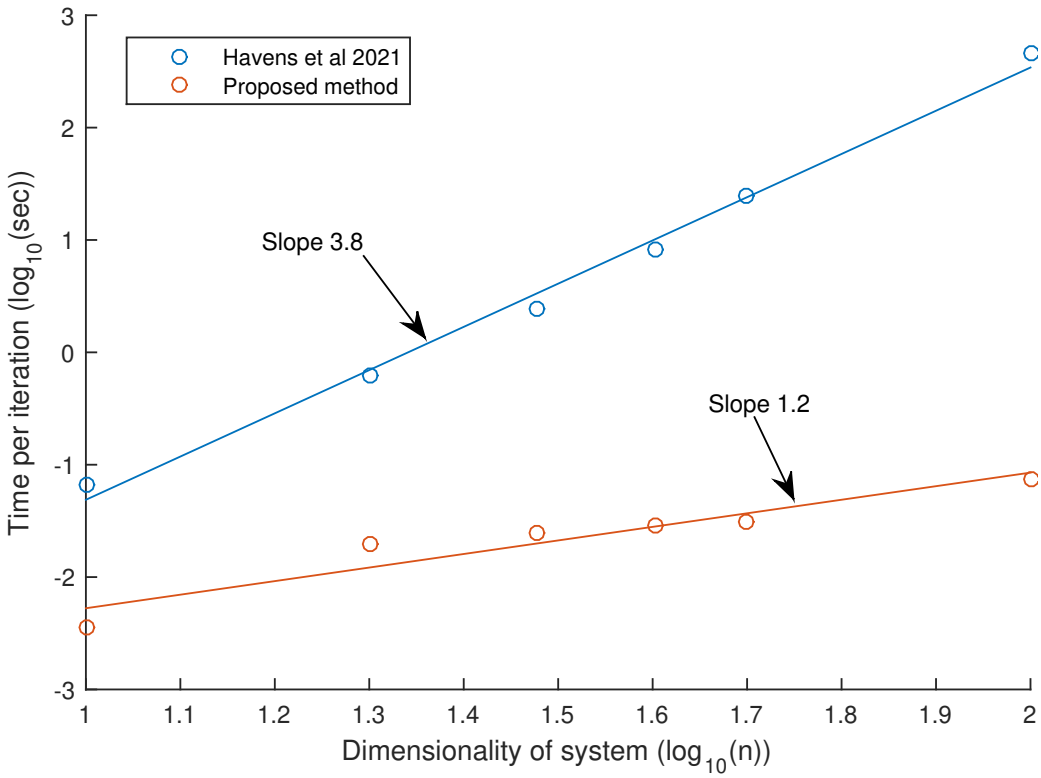


Figure 5.4 – Scatter plot of computation time per iteration of the proposed method vs. the PGD algorithm of Havens and Hu (2021) in log-log scale. Lines of best fit with slope magnitudes are also shown.

5.5.2 Nonlinear Example

Next, the algorithm was validated on a nonlinear problem. Data was obtained from the LASA handwriting dataset (Khansari-Zadeh and Billard, 2011), which consists of trajectories of various hand-drawn letters and symbols. Each datapoint comprises of position, velocity and acceleration in Cartesian coordinates. To incorporate this dataset into a control problem, a 2 DOF robot manipulator is introduced, and has to be controlled such that its end-effector follows the trajectories in the dataset.

The robot dynamics are given by:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = B\tau, \tag{5.46}$$

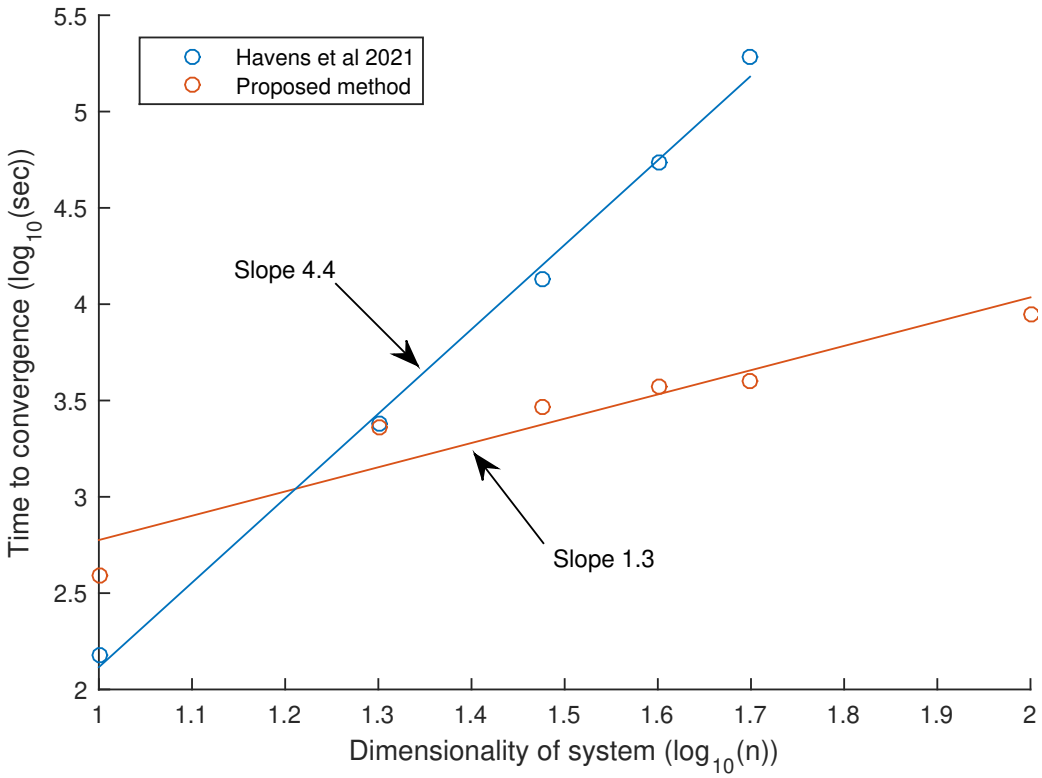


Figure 5.5 – Scatter plot of total time to convergence for the proposed method vs. the PGD algorithm of Havens and Hu (2021) in log-log scale. Lines of best fit with slope magnitudes are also shown. The case of $n = 100$ for the PGD algorithm was not evaluated due to time constraints.

where $q = [q_1, q_2]^\top$ and

$$M = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + m_2 l_1 l_2 \cos(q_2) & I_2 + \frac{1}{2} m_2 l_1 l_2 \cos(q_2) \\ I_2 + \frac{1}{2} m_2 l_1 l_2 \cos(q_2) & I_2 \end{bmatrix}, \tag{5.47}$$

$$C = \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2)\dot{q}_2 & -\frac{1}{2} m_2 l_1 l_2 \sin(q_2)\dot{q}_2 \\ \frac{1}{2} m_2 l_1 l_2 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}, \tag{5.48}$$

$$B = I. \tag{5.49}$$

I_1 and I_2 are the moments of inertia of each link. The nominal parameters of the robot were chosen to be $l_1 = 0.2\ m$, $l_2 = 0.1\ m$, $m_1 = 1\ kg$, $m_2 = 1\ kg$.

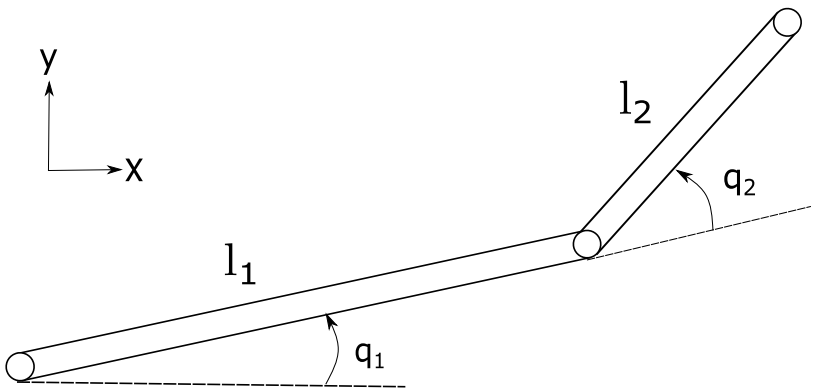


Figure 5.6 – Diagram of robot arm in the horizontal plane. l_1 and l_2 are the lengths of each link.

The Cartesian coordinates of the data can be converted to polar coordinates q with the base joint of the robot as the origin, via the inverse kinematics:

$$\begin{aligned} q_2 &= \cos^{-1}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \\ q_1 &= \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{l_2 \sin(q_2)}{l_1 + l_2 \cos(q_2)}\right) \end{aligned} \tag{5.50}$$

Additionally, the angular velocities $\dot{q}_1$ and $\dot{q}_2$ can be computed from $\dot{x}$ and $\dot{y}$ via the chain rule. The torques associated with each datapoint can then be computed using Equation (5.46).

The sampling rate of the system is assumed to be sufficiently small that zero-order hold is valid, so that a discrete-time model can be fitted and the discrete-time controller can be applied to the robot model (5.46).

Note that the dynamics model was only used to generate the dataset, and is unknown to the learning algorithm. The scenario that this experiment simulates is when demonstrations are provided by a human operator, and the robot states and control torques are recorded.

Comparisons were made of the performance of the learned controller for various values of c_1 and choices of parameterization of α. A comparison was also made against the vanilla behavioural cloning method, which is commonly used as a baseline for evalu-

ating imitation learning algorithms (Fu et al., 2020; Ho and Ermon, 2016; Pfrommer et al., 2022). Behavioural cloning was implemented as fitting a neural network mapping states to control inputs by minimizing a mean-squared error loss on the controller output. The neural networks were chosen to have 2 hidden layers with 20 nodes and tanh activations.

To evaluate the performance of the learned controllers, two quantitative metrics were compared: imitation error and controller fit, which are defined in the following. Unlike the linear case, global closed-loop stability is not trivial to verify.

These two metrics are defined as follows:

Imitation error (normalized simulation error)

$$\sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{x}_t^i - \hat{x}_t^i \right\|^2 \Bigg/ \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{x}_t^i \right\|^2 , \tag{5.51}$$

where $\hat{x}_{t+1} = f(\hat{x}_t) + g(\hat{x}_t)k(\hat{x}_t)$ and $\hat{x}_0 = \tilde{x}_0$.

Controller error

$$\sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{u}_t^i - \hat{u}_t^i \right\|^2 \Bigg/ \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T_{\text{test}}-1} \left\| \tilde{u}_t^i \right\|^2 , \tag{5.52}$$

where $\hat{u}_t = k(\tilde{x}_t)$.

Figure 5.7 shows the imitation error of the learned controllers. Firstly, it can be seen that there is a threshold for c_1 where imitation error is reduced, but performance does not improve if c_1 is further increased. This supports the intuition that the open-loop dynamics error associated with c_1 only acts as a coupling term for the controller and closed-loop dynamics errors. As such, once c_1 is sufficiently large for the corresponding term to dominate the loss, increasing c_1 further does not reduce the imitation error.

It can also be seen from Figure 5.7 that using the nonlinear parameterization for α defined in Equation 5.26 leads to worse performance. This could be due to the linear parameterization being sufficiently expressive to model the observed controller, hence

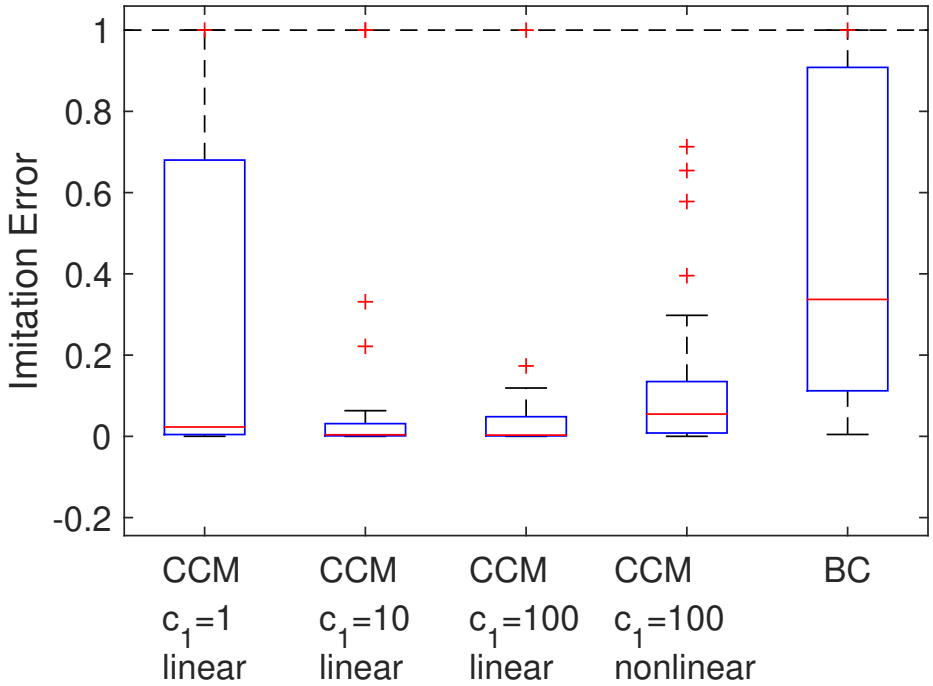


Figure 5.7 – Imitation error (5.51) (aka normalized simulation error) of learned controllers on the test set. From left to right: linear parameterization of α — $c_1 = 0$, $c_1 = 10$ and $c1 = 100$, nonlinear parameterization of α — $c1 = 100$, behavioural cloning (BC). Number of outliers from left to right: 4, 2, 1, 0, 5

adding more parameters through the nonlinear α only increases the variance of the model and leads to overfitting.

Finally, behavioural cloning has the highest imitation error out of all the test cases. This shows that the proposed model does indeed improve the performance of the controller over the baseline.

Figure 5.8 and Figure 5.9 show the training and test error respectively of the controller output. Interestingly, behavioural cloning achieves the lowest controller error on the training data but has the highest error on the test set. This is a symptom of overfitting, and also supports the claim that only minimizing the controller error does not correlate with good imitation performance. This trend is also seen in the nonlinear α case, which achieves slightly lower error than the linear parameterization on the training set but has higher error on the test set.

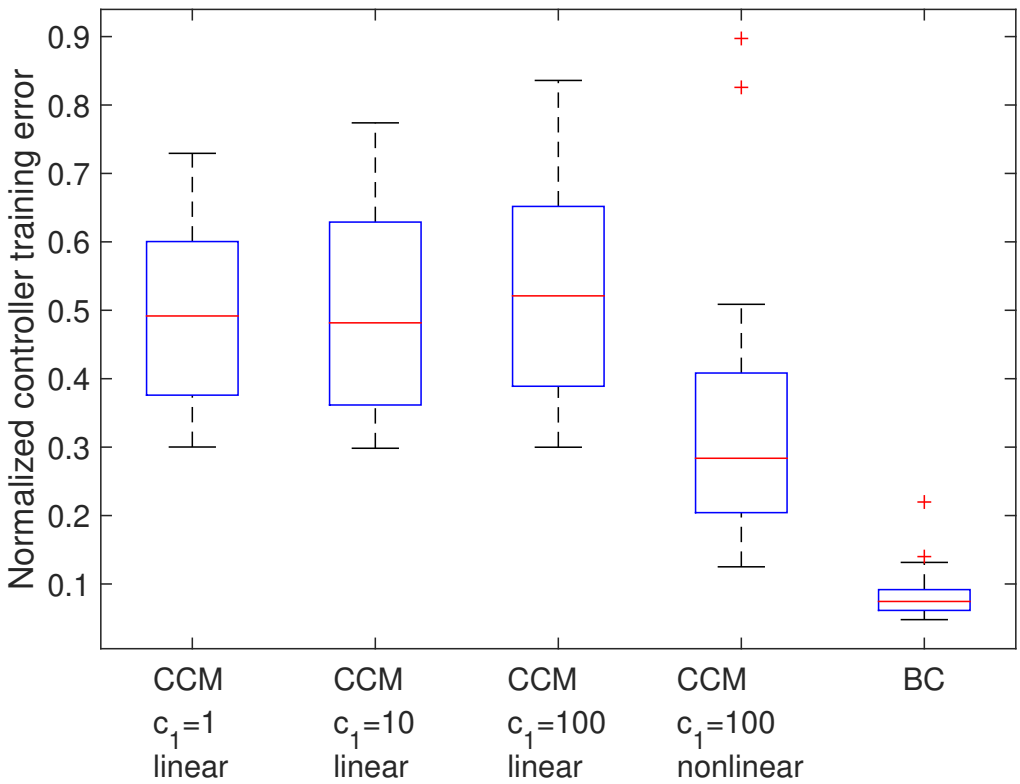


Figure 5.8 – Normalized controller error (5.52) for training data.

Figure 5.10 and Figure 5.11 show a qualitative comparison of the trajectories induced by the CCM controller with linear α and $c_1 = 100$, shown in Figure 5.10, and the behavioural cloning controller, shown in Figure 5.11. It can be seen that the CCM controller induces a (locally) contracting closed-loop system where nearby trajectories converge to a single equilibrium, whereas the trajectories of the BC controller diverge even for small perturbations of the initial condition, which is unacceptable when controlling physical systems. Refer to Appendix A for more plots of the trajectories of other shapes.

Although the learned CCM controller is not guaranteed to be stabilizing, it is clear that it provides a clear improvement over behavioural cloning for the same requirements on the data and without significant increase in computational cost. The results show that the proposed framework does have a regularizing effect on learning stabilizing controllers and outperforms behavioural cloning in terms of the imitation error.

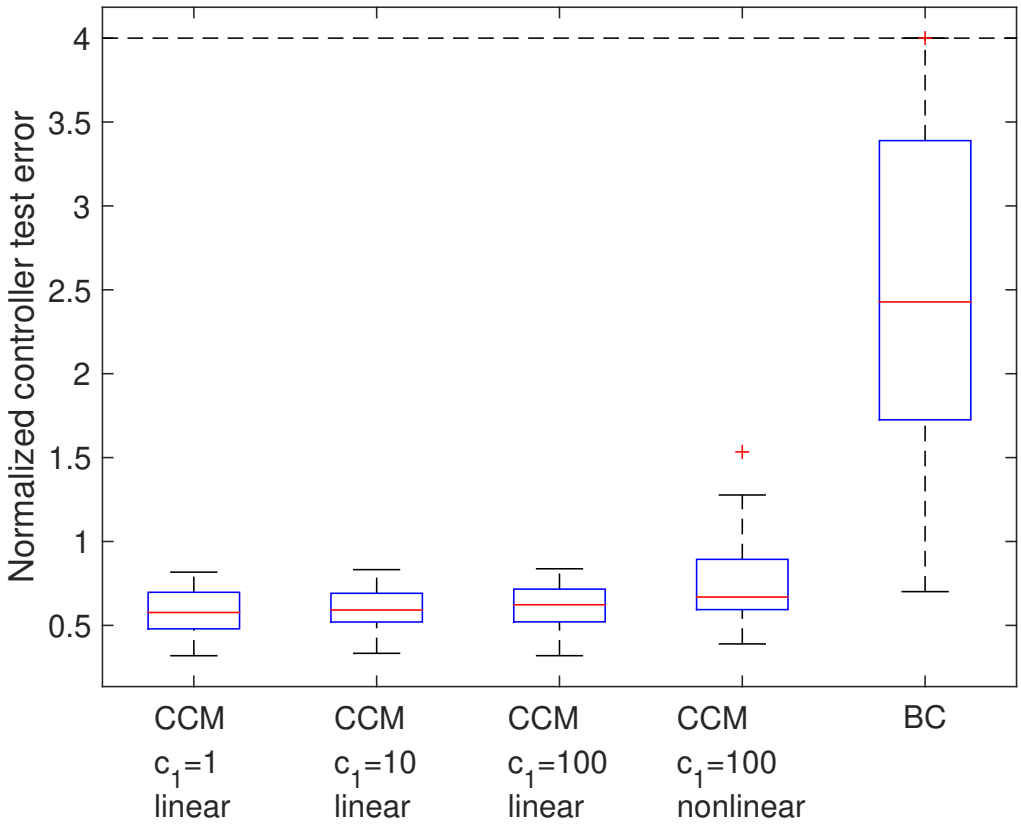


Figure 5.9 – Normalized controller error (5.52) for test data.

5.6 Summary

To summarize this chapter, a class of stabilizable Koopman models has been proposed that is guaranteed to admit a CCM controller for the learned open-loop dynamics. A stability-regularized imitation learning problem was proposed and it was shown how the Koopman model class could be applied to solve this problem. Finally, it was shown that solving this stability-regularized IL problem produces more stable controllers with lower imitation error than unregularized behavioural cloning.

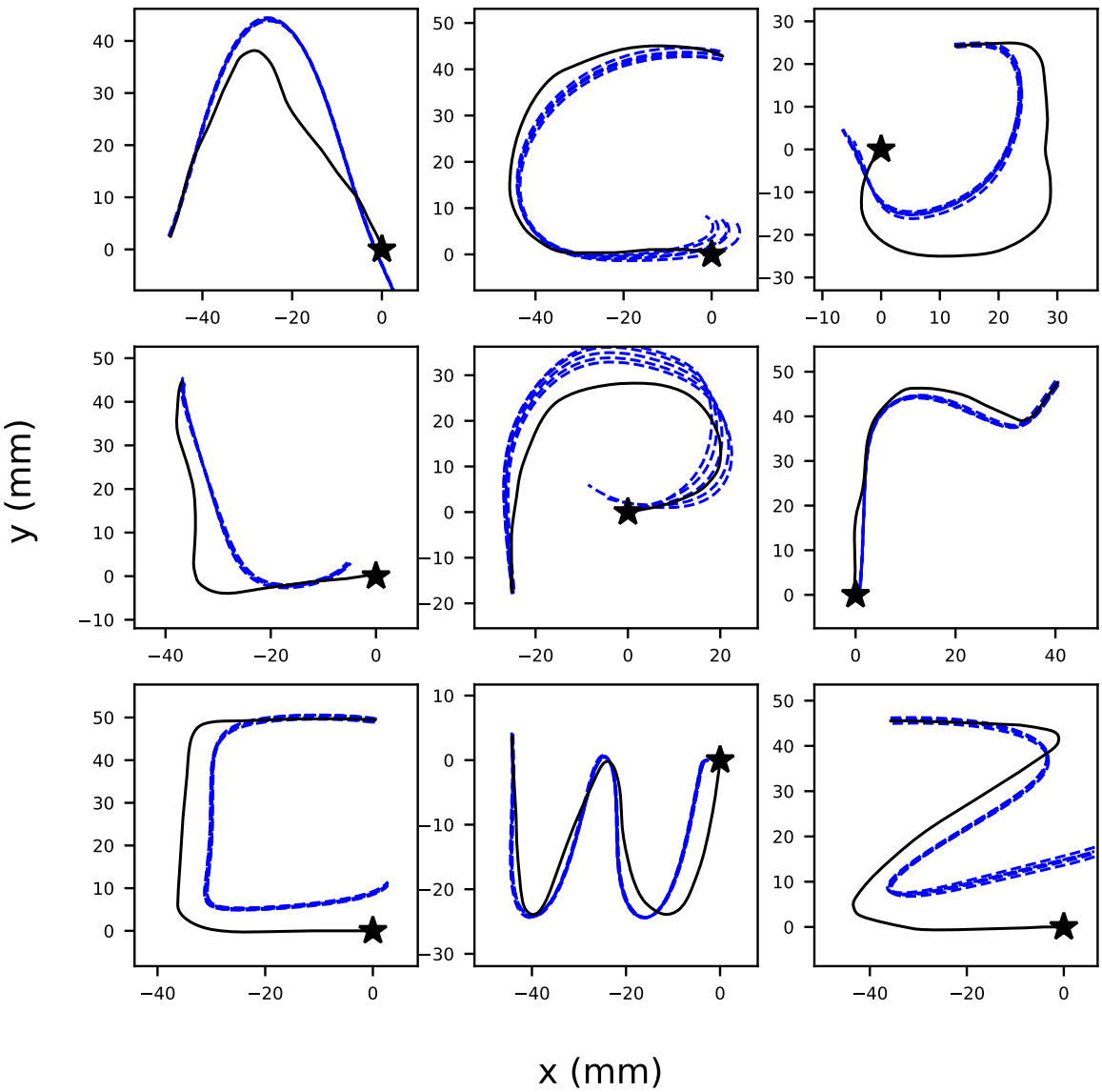


Figure 5.10 – Examples of trajectories induced by learned CCM controller with linear α and $c1 = 100$. The solid black line is the ground truth, while the dotted blue lines are generated by sampling initial conditions in a small box of width 2 mm around the true initial condition.

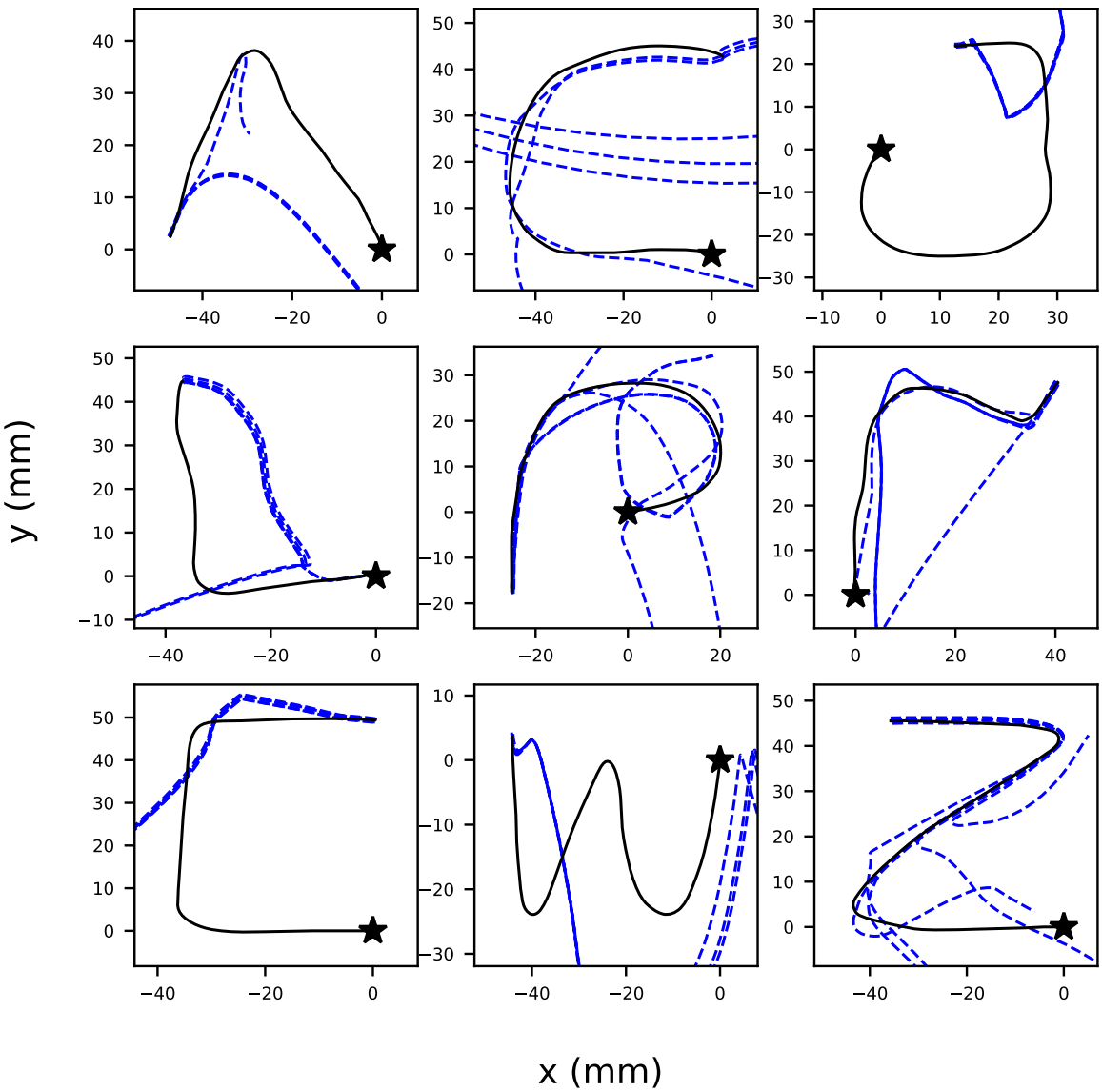


Figure 5.11 – Examples of trajectories induced by behavioural cloning controller. The solid black line is the ground truth, while the dotted blue lines are generated by sampling initial conditions in a small box of width 2 mm around the true initial condition.

Chapter 6

Conclusion

This chapter provides a summary of the contributions of this work as well as some future directions for continuing this line of research.

6.1 Summary

In this thesis, two new classes of Koopman models were proposed and applied to identification and control problems. The first model class is guaranteed to be stable, while the second is guaranteed to be stabilizable with an explicit stabilizing controller that renders the model stable in closed-loop. Furthermore, these models are unconstrained in their parameter sets, hence enabling efficient optimization via gradient-based methods. Theoretical connections between the stability of Koopman models and contraction analysis were established. Experimental results showed empirically that the proposed models achieve better performance over prior methods without stability guarantees.

6.2 Future Work

The field of Koopman-based methods is rich with potential and many exciting directions of research remain to be explored.

Firstly, stable Koopman models can be considered in the context of general stable nonlinear models, and comparisons can be made to prior model parameterizations such as recurrent equilibrium networks (Revay et al., 2021b).

Additionally, the learned Koopman model can be used to analyse the nonlinear system via its spectral decomposition. This can be useful, for example, to obtain a reduced-order model from the dominant eigenfunctions.

The Koopman model can also be extended to represent nonlinear systems with other asymptotic behaviour such as limit cycles, which can be described by contraction in transverse coordinates (Manchester and Slotine, 2014). The connection between transverse contraction and Koopman orbital stability has already been established in Yi and Manchester (2022).

Finally, in the context of imitation learning, the connection of the proposed model to inverse optimal control discussed in Section 5.3.5 can be explored further. The learned cost function can be used to model the intent of the expert, and design new controllers for a different system.

List of References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5.

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

David Angeli. A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410–421, 2002.

Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, number 119 in Proceedings of Machine Learning Research, pages 475–485. PMLR, 2020.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021. ISSN 1367-5788.

Petar Bevanda, Max Beier, Shahab Heshmati-Alamdari, Stefan Sosnowski, and Sandra Hirche. Towards data-driven LQR with Koopmanizing flows. *arXiv preprint arXiv:2201.11640*, 2022a.

Petar Bevanda, Max Beier, Sebastian Kerz, Armin Lederer, Stefan Sosnowski, and Sandra Hirche. Diffeomorphically learning stable Koopman operators. *IEEE Control Systems Letters*, 6:3427–3432, 2022b.

Rajendra Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics, Vol. 169. Springer, 1996.

Caroline Blocher, Matteo Saveriano, and Dongheui Lee. Learning stable dynamical systems using contraction theory. In *14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 124–129, 2017.

Léon Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD. ISBN 978-3-7908-2604-3.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Lucas Brivadis, Vincent Andrieu, and Ulysse Serres. Luenberger observers for discrete-time nonlinear systems. In *58th IEEE Conference on Decision and Control (CDC 2019)*, pages 3435–3440. IEEE, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS One*, 11(2):e0150171, 2016.

Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(1):1–9, 2017.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

Federico Celi, Giacomo Baggio, and Fabio Pasqualetti. Closed-form estimates of the LQR gain from finite data. In *61st IEEE Conference on Decision and Control (CDC 2022)*, 2022.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal Wasserstein imitation learning. In *International Conference on Learning Representations*, 2021.

Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, 20(4):633–679, 2020.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., 2017.

N. Benjamin Erichson, Michael Muehlebach, and Michael W. Mahoney. Physics-informed autoencoders for Lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

Fletcher Fan, Bowen Yi, David Rye, Guodong Shi, and Ian R. Manchester. Learning stable Koopman embeddings. In *2022 American Control Conference (ACC 2022)*, pages 2742–2747, 2022.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning*, number 48 in Proceedings of Machine Learning Research, pages 49–58, New York, New York, USA, 20–22 June 2016. PMLR.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

Carl Folkestad, Skylar X. Wei, and Joel W. Burdick. Koopnet: Joint learning of Koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking. In *Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1344–1350. IEEE, 2022.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the 3rd Conference on Robot Learning (CoRL 2019)*, number 100 in Proceedings of Machine Learning Research, pages 1259–1277. PMLR, 30 October–1 November 2020.

Nicolas Gillis, Michael Karow, and Punit Sharma. A note on approximating the nearest stable discrete-time descriptor systems with fixed rank. *Applied Numerical Mathematics*, 148:131–139, 2020. ISSN 0168-9274.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2672–2680. Curran Associates, Inc., 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

Debdipta Goswami and Derek A. Paley. Global bilinearization and controllability of control-affine nonlinear systems: A Koopman spectral approach. In *56th IEEE Conference on Decision and Control (CDC 2017)*, pages 6107–6112. IEEE, 2017.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Other Titles in Applied Mathematics. SIAM, 2nd edition, 2008. ISBN 978-0-89871-659-7.

Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of Koopman representation for control. In *59th IEEE Conference on Decision and Control (CDC 2020)*, pages 1890–1895. IEEE, 2020.

Masih Haseli and Jorge Cortes. Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric subspace decomposition. *IEEE Transactions on Automatic Control*, 67(7):3442–3457, 2021.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd edition, 2009.

Aaron Havens and Bin Hu. On imitation learning of linear control policies: Enforcing stability and robustness constraints via LMI conditions. In *2021 American Control Conference (ACC 2021)*, pages 882–887. IEEE, 2021.

Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. Curran Associates, Inc., 2016.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Bowen Huang, Xu Ma, and Umesh Vaidya. Feedback stabilization using Koopman operator. In *67th IEEE Conference on Decision and Control (CDC 2018)*, pages 6434–6439. IEEE, 2018.

Lucian Cristian Iacob, Gerben Izaak Beintema, Maarten Schoukens, and Roland Tóth. Deep identification of nonlinear systems in Koopman form. In *60th IEEE Conference on Decision and Control (CDC 2021)*, pages 2288–2293, 2021.

Lucian Cristian Iacob, Roland Tóth, and Maarten Schoukens. Koopman form of nonlinear systems with inputs. *arXiv preprint arXiv:2207.12132*, 2022a.

Lucian Cristian Iacob, Roland Tóth, and Maarten Schoukens. Optimal synthesis of LTI Koopman models for nonlinear systems with inputs. *arXiv preprint arXiv:2206.07534*, 2022b.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, February 2013. ISSN 0899-7667.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106: 620–630, May 1957.

Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2(3):035023, 2021.

R. E. Kalman. When is a linear control system optimal? *Journal of Basic Engineering*, 86(1):51–60, March 1964. ISSN 0021-9223.

Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, May 1931. ISSN 1091-6490.

Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018. ISSN 0005-1098.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pages 1097–1105. Curran Associates, Inc., 2012.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17 (39):1–40, 2016.

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

Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: Interpretable imitation learning from visual demonstrations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., 2017b.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Lennart Ljung. *System Identification*, pages 163–173. Birkhäuser Boston, Boston, MA, 1998. ISBN 978-1-4612-1768-8.

Winfried Lohmiller and Jean-Jacques E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998. ISSN 0005-1098.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9 (1):4950, November 2018. ISSN 2041-1723.

Giorgos Mamakoukas, Orest Xherija, and Todd Murphey. Memory-efficient learning of stable linear dynamical systems for prediction and control. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 13527–13538. Curran Associates, Inc., 2020.

Ian R Manchester and Jean-Jacques E Slotine. Transverse contraction criteria for existence, stability, and robustness of a limit cycle. *Systems & Control Letters*, 63:32–38, 2014.

Ian R. Manchester and Jean-Jacques E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.

Ian R. Manchester, Max Revay, and Ruigang Wang. Contraction-based methods for stable identification and robust machine learning: A tutorial. In *60th IEEE Conference on Decision and Control (CDC 2021)*, pages 2955–2962, 2021.

Gaurav Manek and J. Zico Kolter. Learning stable deep dynamics models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets for deep learning of molecular kinetics. *Nature Communications*, 9(1):5, January 2018. ISSN 2041-1723.

Alexandre Mauroy and Jorge Goncalves. Koopman-based lifting techniques for nonlinear systems identification. *IEEE Transactions on Automatic Control*, 65(6): 2550–2565, 2020.

Alexandre Mauroy and Igor Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, 2016.

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

S. Mohammad Khansari-Zadeh and Aude Billard. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014. ISSN 0921-8890.

Klaus Neumann, Andre Lemme, and Jochen J. Steil. Neural learning of stable dynamical systems based on data-driven Lyapunov candidates. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1216–1222, 2013.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

Takayuki Osa, Naohiko Sugita, and Mamoru Mitsuishi. Online trajectory planning and force control for automation of surgical tasks. *IEEE Transactions on Automation Science and Engineering*, 15(2):675–691, 2017.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1–2):1–179, 2018.

Samuel E. Otto and Clarence W. Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1): 558–593, 2019.

Samuel E. Otto and Clarence W. Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, 2021.

Malayandi Palan, Shane Barratt, Alex McCauley, Dorsa Sadigh, Vikas Sindhwani, and Stephen Boyd. Fitting a linear control policy to demonstrations with a Kalman constraint. In *2nd Annual Conference on Learning for Dynamics and Control*, number 120 in Proceedings of Machine Learning Research, pages 374–383. PMLR, 2020.

Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR, 2013.

Daniel Pfrommer, Thomas T. C. K. Zhang, Stephen Tu, and Nikolai Matni. TaSIL: Taylor series imitation learning. *arXiv preprint arXiv:2205.14812*, 2022.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1): 142–161, 2016.

Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Generalizing Koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

Harish Ravichandar, Iman Salehi, and Ashwin Dani. Learning partially contracting dynamical systems from demonstrations. In *Proceedings of the 1st Conference on Robot Learning (CoRL 2017)*, number 78 in Proceedings of Machine Learning Research, pages 369–378. PMLR, 13–15 November 2017.

Max Revay, Ruigang Wang, and Ian R. Manchester. A convex parameterization of robust recurrent neural networks. *IEEE Control Systems Letters*, 5(4):1363–1368, 2021a.

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Unconstrained learning of stable and robust dynamical models. In *60th IEEE Conference on Decision and Control (CDC 2021)*, pages 2282–2287, 2021b.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, number 9 in Proceedings of Machine Learning Research, pages 661–668. JMLR Workshop and Conference Proceedings, PMLR, 2010.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, number 15 in Proceedings of Machine Learning Research, pages 627–635, Fort Lauderdale, FL, USA, 11–13 April 2011. PMLR.

Tim Roughgarden. *Beyond the Worst-case Analysis of Algorithms*. Cambridge University Press, 2020.

Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 101–103, 1998.

Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In Paolo Dario and Raja Chatila, editors, *Robotics Research: The Eleventh International Symposium*, pages 561–572, Berlin, Heidelberg, 2005. Springer. ISBN 978-3-540-31508-7.

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

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda

Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484, January 2016.

Vikas Sindhwani, Stephen Tu, and Mohi Khansari. Learning contracting vector fields for stable imitation learning. 2018. arXiv preprint 1804.04878v1.

Sumeet Singh, Spencer M Richards, Vikas Sindhwani, Jean-Jacques E Slotine, and Marco Pavone. Learning stabilizable nonlinear dynamics with contraction-based regularization. *The International Journal of Robotics Research*, 40(10-11): 1123–1150, 2021.

Subhrajit Sinha, Sai Pushpak Nandanoori, Jan Drgona, and Draguna Vrabie. Data-driven stabilization of discrete-time control-affine nonlinear systems: A Koopman operator approach. *arXiv preprint arXiv:2203.14114*, 2022.

Eduardo D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989.

Eduardo D. Sontag. Input to state stability: Basic concepts and results. In *Nonlinear and Optimal Control Theory*, pages 163–220. Springer, 2008.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Amit Surana. Koopman operator based observer synthesis for control-affine nonlinear systems. In *55th IEEE Conference on Decision and Control (CDC 2016)*, pages 6492–6499. IEEE, 2016.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 1130–1140. Curran Associates, Inc., 2017.

Mark M. Tobenkin, Ian R. Manchester, and Alexandre Megretski. Convex parameterizations and fidelity bounds for nonlinear identification and reduced-order modelling. *IEEE Transactions on Automatic Control*, 62(7): 3679–3686, 2017.

Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014. ISSN 2158-2491.

Stephen Tu, Alexander Robey, Tingnan Zhang, and Nikolai Matni. On the sample complexity of stability constrained imitation learning. In *Learning for Dynamics and Control Conference*, pages 180–191, 2022.

J. Umenberger and I. R. Manchester. Convex bounds for equation error in stable nonlinear identification. *IEEE Control Systems Letters*, 3(1):73–78, 2019.

V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

Yuh-Shyang Wang, Nikolai Matni, and John C. Doyle. A system-level approach to controller synthesis. *IEEE Transactions on Automatic Control*, 64(10):4079–4093, 2019.

Ziyu Wang, Josh S. Merel, Scott E. Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5324–5333. Curran Associates, Inc., 2017.

Lai Wei, Ryan McCloy, and Jie Bao. Control contraction metric synthesis for discrete-time nonlinear systems. *arXiv preprint arXiv:2104.10352*, 2021.

M. O. Williams, Clarence W. Rowley, and I. G. Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2016.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, December 2015. ISSN 1432-1467.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC 2019)*, pages 4832–4839, 2019.

Bowen Yi and Ian R. Manchester. On the equivalence of contraction and Koopman approaches for nonlinear stability and control. In *61st IEEE Conference on Decision and Control (CDC 2022)*, 2022.

He Yin, Peter Seiler, Ming Jin, and Murat Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 6:409–414, 2021.

Brian D. Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. PhD thesis, Carnegie Mellon University, 2010.

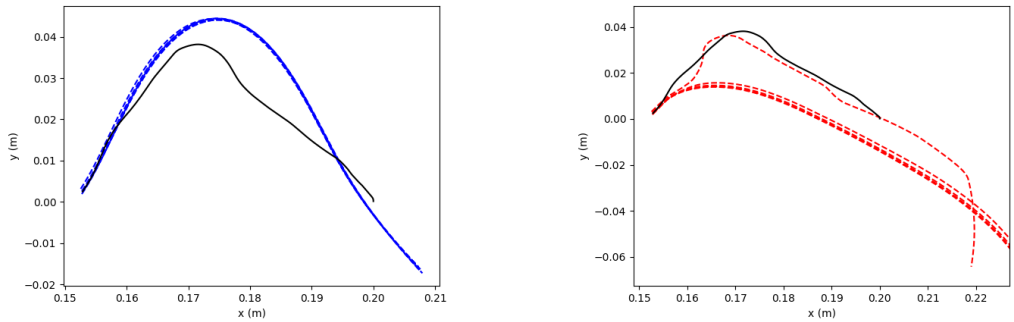
Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 3, pages 1433–1438. AAAI Press, 2008. ISBN 978-1-57735-368-3.

Vrushabh Zinage and Efstathios Bakolas. Neural Koopman Lyapunov control. *arXiv preprint arXiv:2201.05098*, 2022.

Appendix A

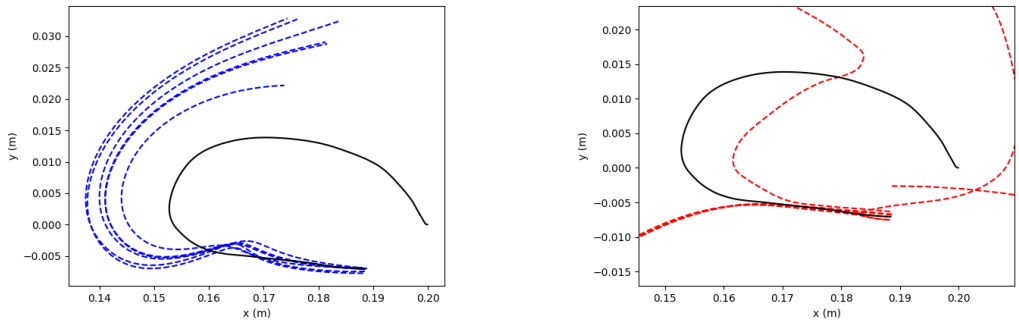
Additional Imitation Learning Trajectories

The following figures show some additional results from the experiments in Section 5.5.2. In each figure, the blue dotted lines are trajectories induced by the control contraction metric (CCM) controller on the true system dynamics, while the red dotted lines are produced by the behavioural cloning (BC) controller. The solid black line is the ground truth, while the dotted lines are generated by sampling initial conditions in a small box of width 2 cm around the true initial condition. The final goal state for all of the shapes is at (0.2, 0.0).



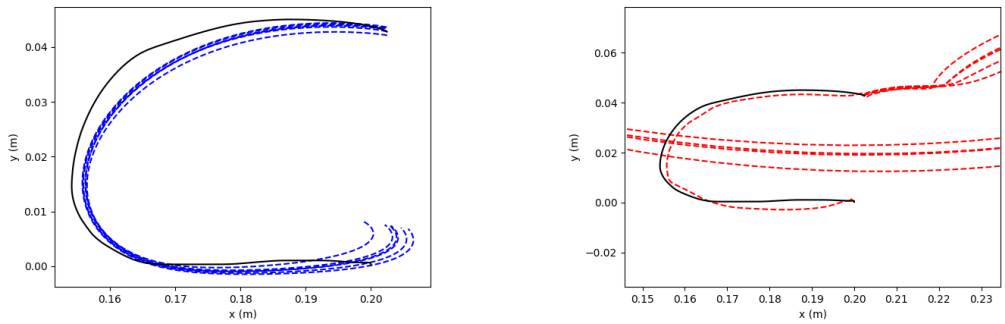
(a) CCM **(b)** BC

Figure A.1



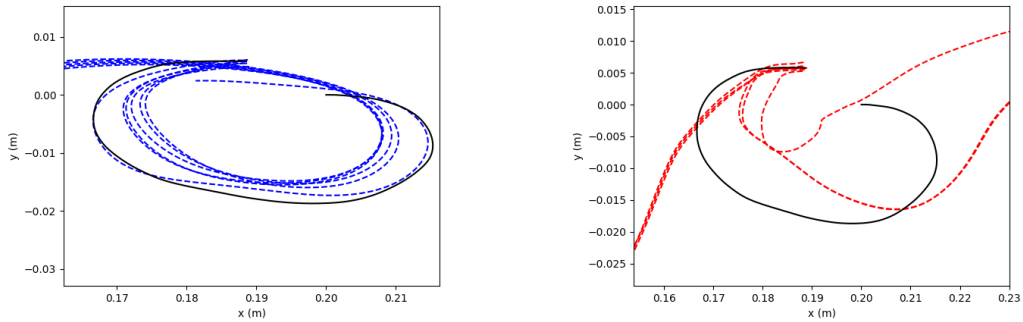
(a) CCM **(b)** BC

Figure A.2



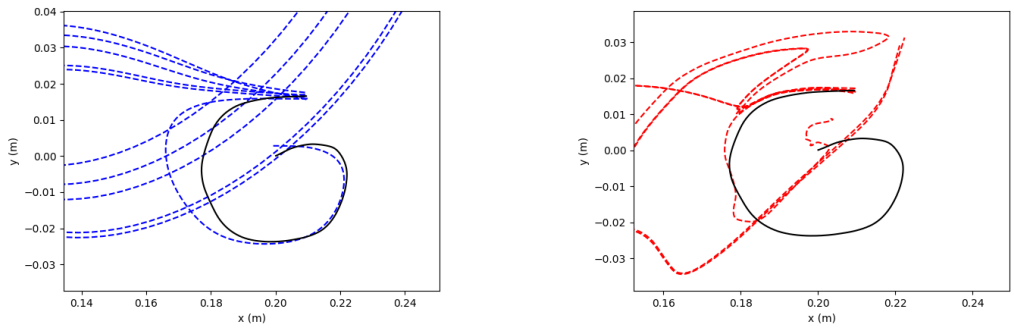
(a) CCM **(b)** BC

Figure A.3



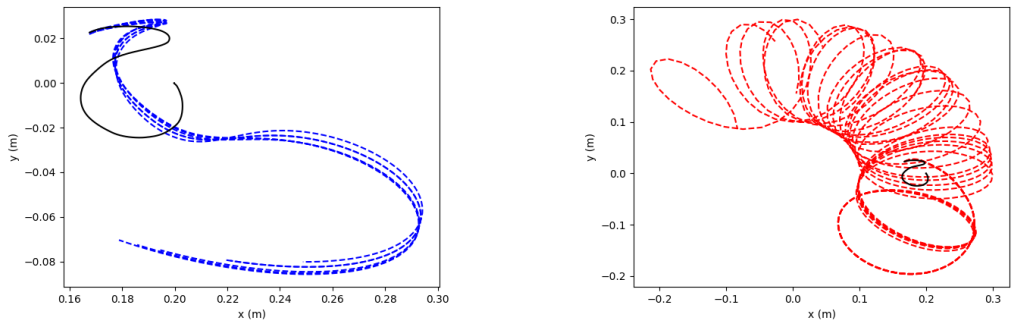
(a) CCM (b) BC

Figure A.4



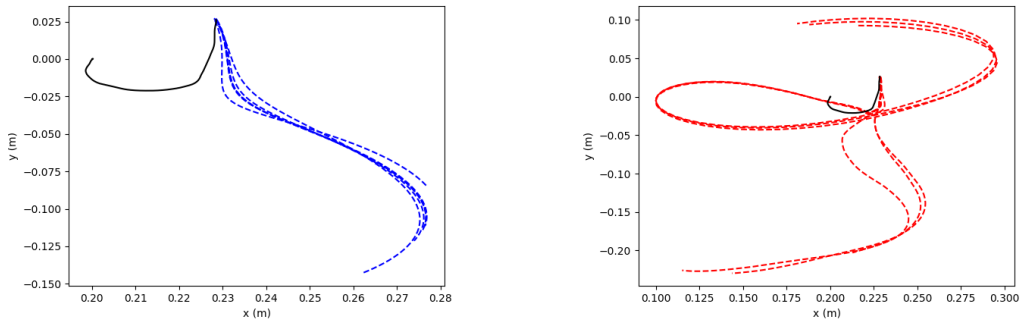
(a) CCM (b) BC

Figure A.5



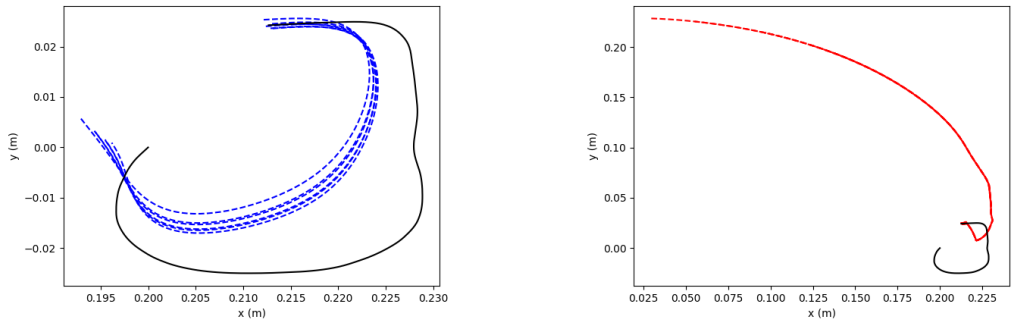
(a) CCM (b) BC

Figure A.6



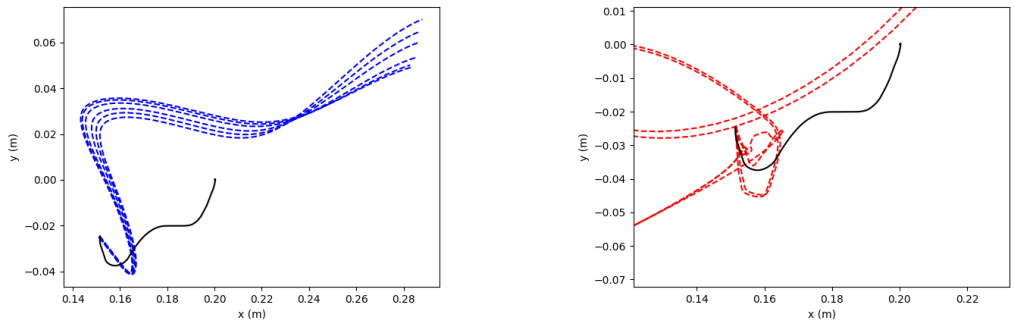
(a) CCM (b) BC

Figure A.7



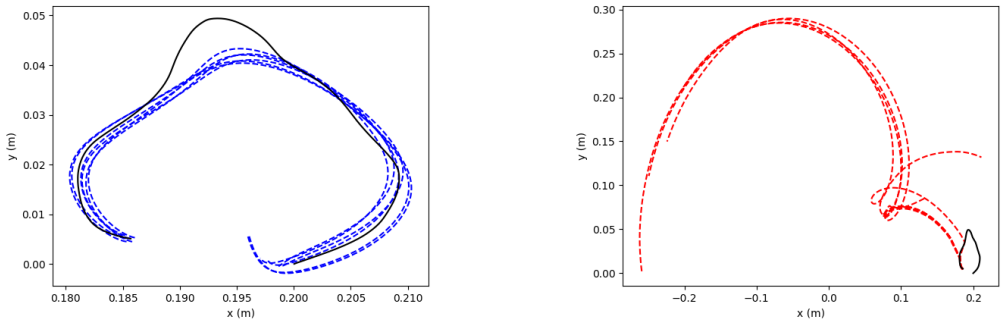
(a) CCM (b) BC

Figure A.8



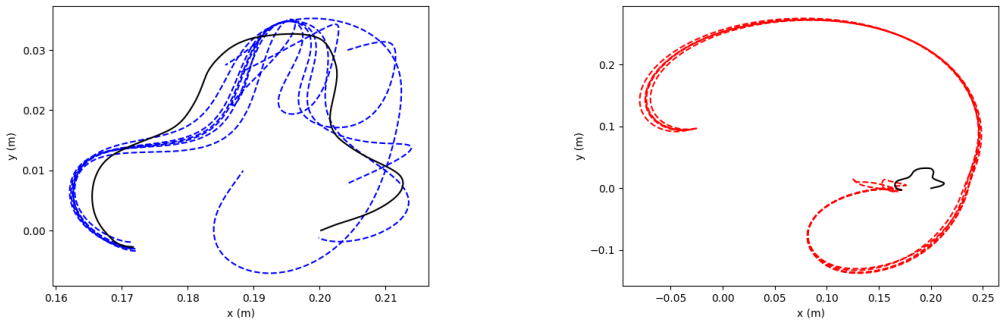
(a) CCM (b) BC

Figure A.9



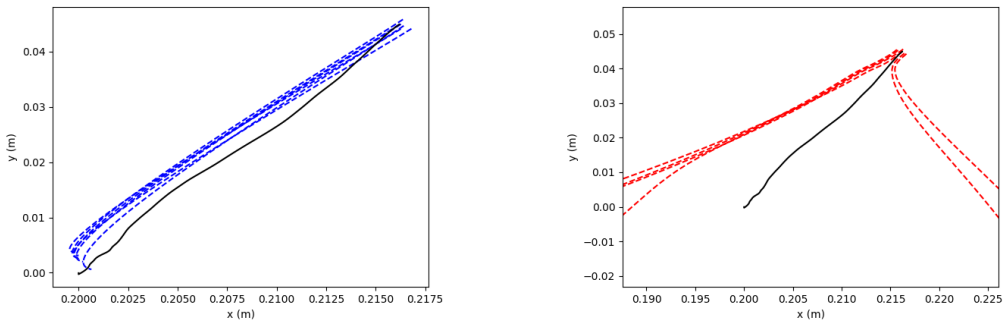
(a) CCM (b) BC

Figure A.10



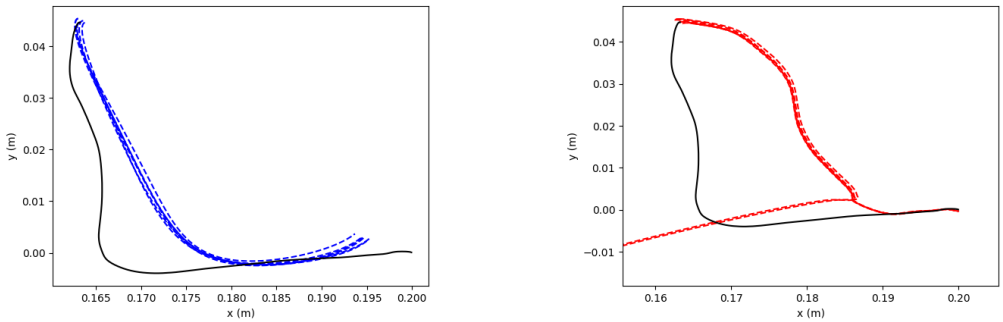
(a) CCM (b) BC

Figure A.11



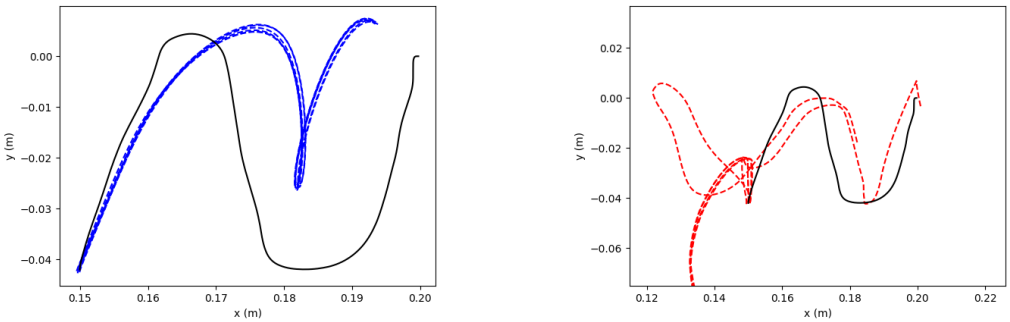
(a) CCM (b) BC

Figure A.12



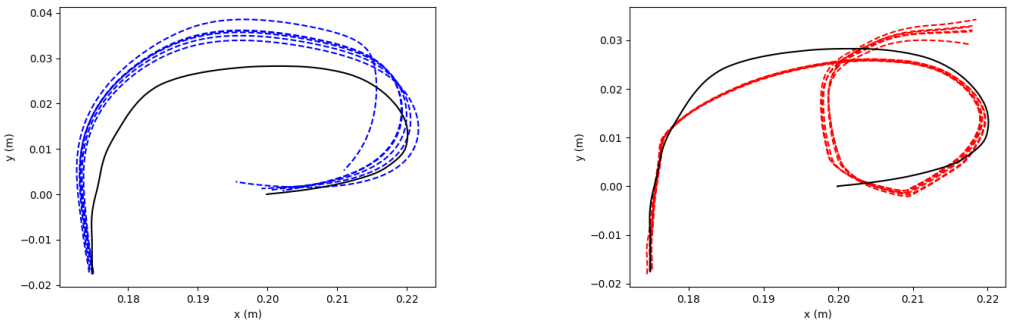
(a) CCM (b) BC

Figure A.13



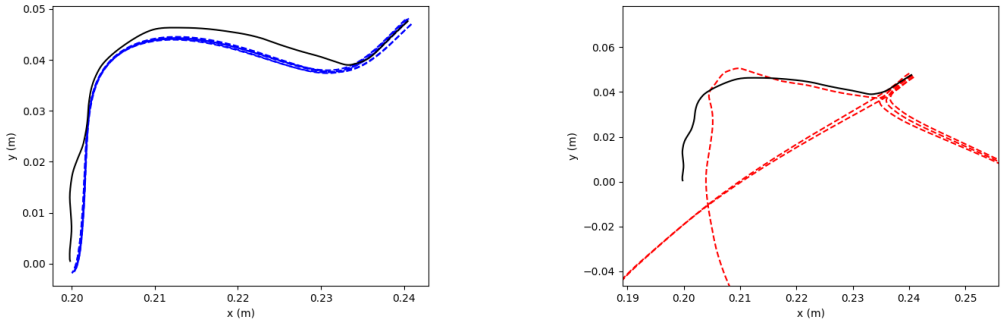
(a) CCM (b) BC

Figure A.14



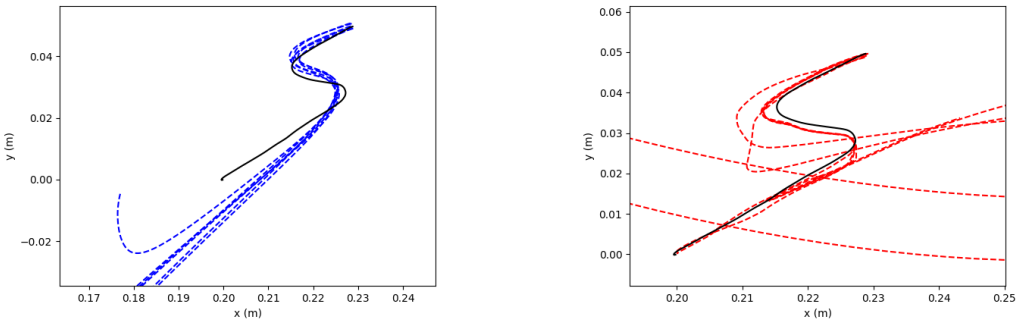
(a) CCM (b) BC

Figure A.15



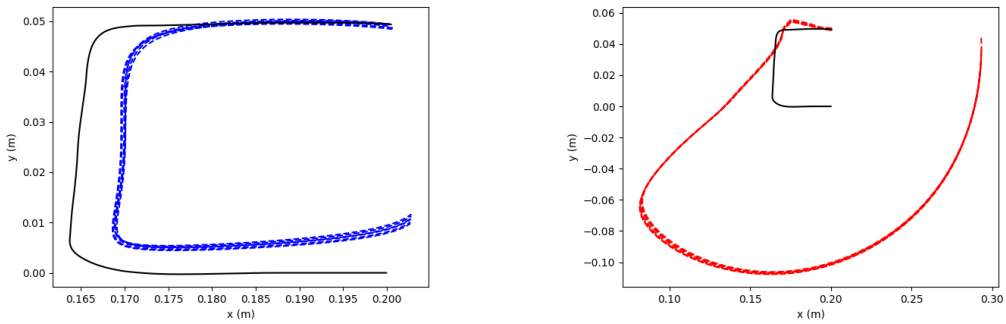
(a) CCM (b) BC

Figure A.16



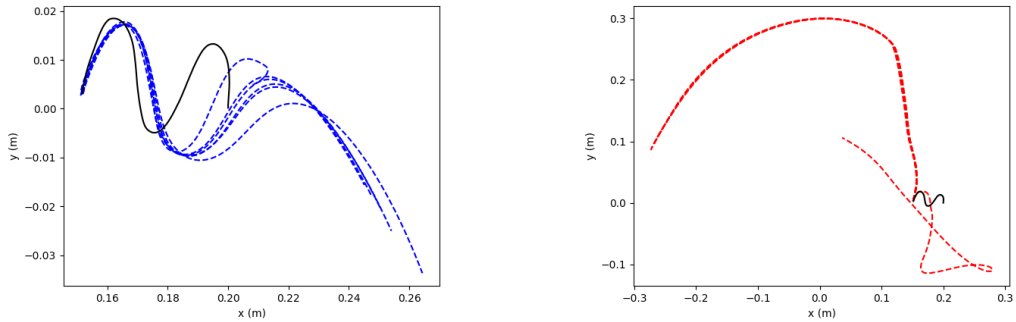
(a) CCM (b) BC

Figure A.17



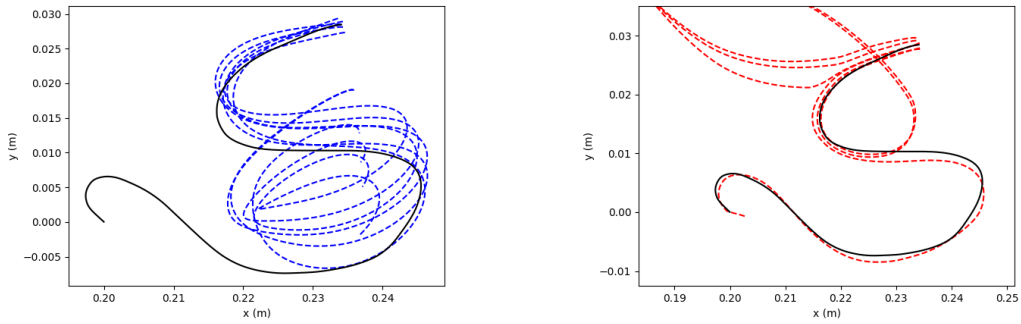
(a) CCM (b) BC

Figure A.18



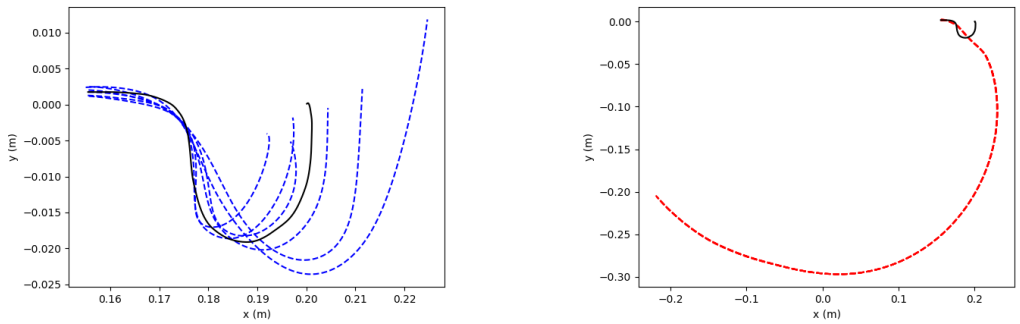
(a) CCM **(b)** BC

Figure A.19



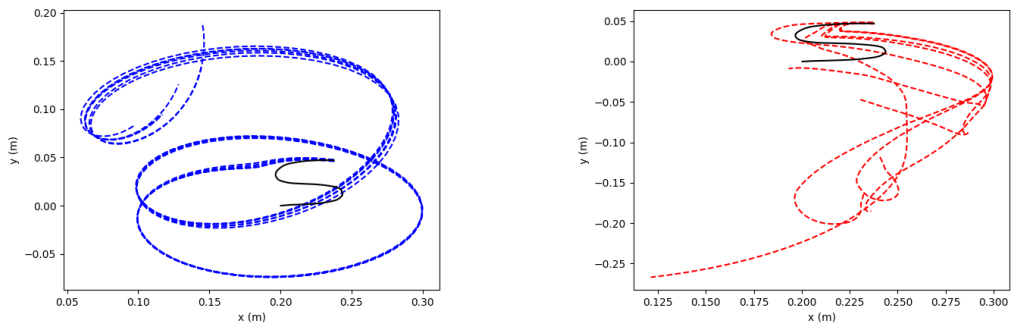
(a) CCM **(b)** BC

Figure A.20



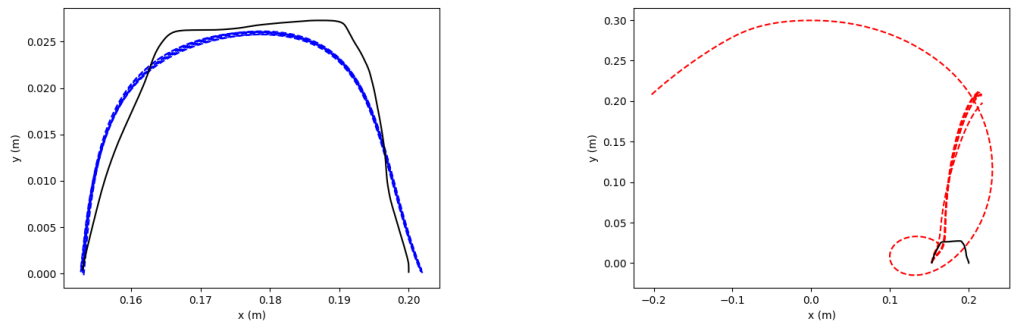
(a) CCM **(b)** BC

Figure A.21



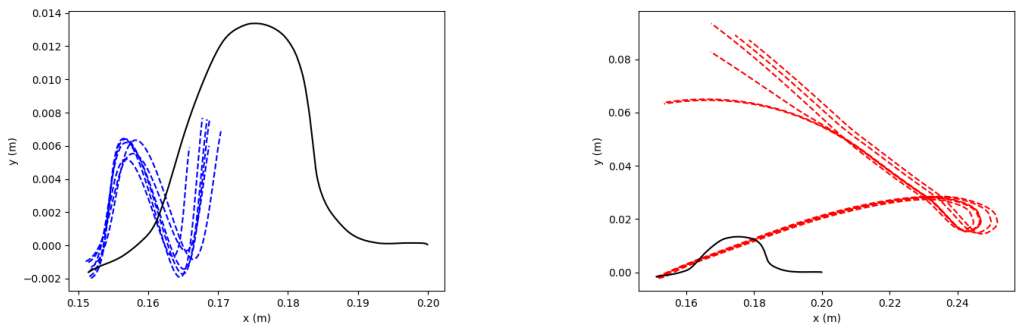
(a) CCM (b) BC

Figure A.22



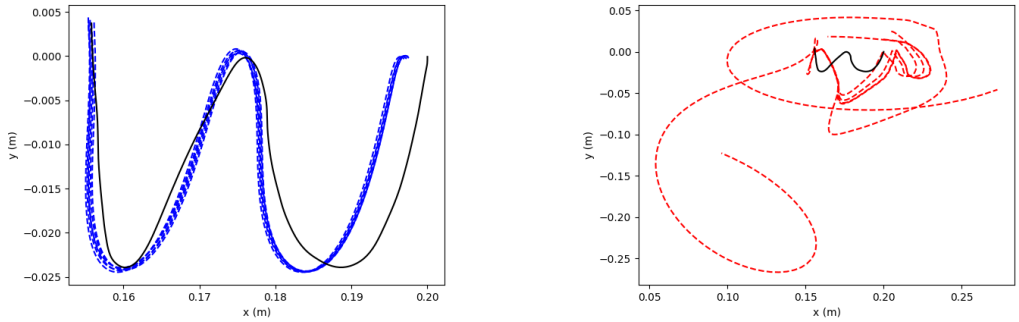
(a) CCM (b) BC

Figure A.23



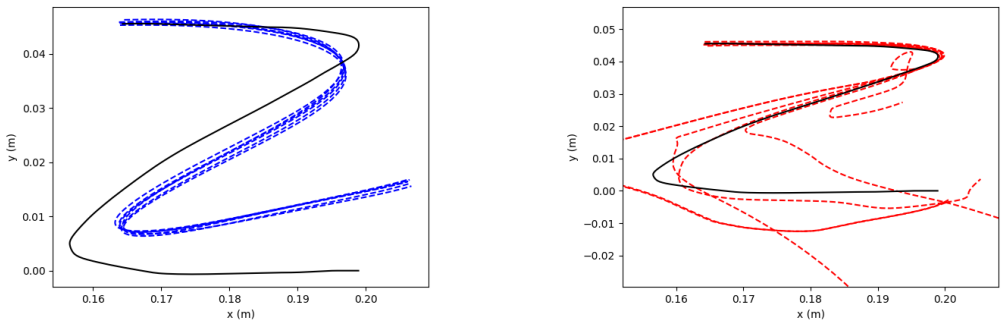
(a) CCM (b) BC

Figure A.24



(a) CCM **(b)** BC

Figure A.25



(a) CCM **(b)** BC

Figure A.26