TUHH
Hamburg University of Technology

ICS
Institute of Control Systems

Master's thesis

Comparison of data-driven control algorithms using Koopman operator and LPV techniques for evaluating the tracking performance of an Arm-Driven Inverted Pendulum

Author:
Yashasvi Sai Sukhavasi

Supervisor: Patrick Göttsch, M.Sc.

Examiner: 1. Prof. Dr. Herbert Werner
2. Prof. Dr. Tobias Knopp

January 20, 2021

ICS
Institute of Control Systems

Master's thesis **M — 7 — 2020**
for Mr. Yashasvi Sai Sukhavasi Student ID: 21757720 Study Course: IMPMEC

Title:

Comparison of data-driven control algorithms using Koopman operator and LPV techniques for evaluating the tracking performance of an Arm-Driven Inverted Pendulum

Project Description:

The Koopman operator theory [1], is an increasingly popular formalism of dynamical systems theory that enables analysis, prediction and control of nonlinear dynamics from measurement data. Such data-driven techniques allow for devising linear control strategies for highly nonlinear dynamical systems for which first principle models are not available or are impractical to be constructed.

The motivation for this Master's thesis is derived from the desire to generate dynamical models of an ADIP through measured data. In particular it is of interest to synthesize linear model-based controllers using these data-driven models in simulation and experimentally compare the performance of data-driven controllers with LPV contrrollers based on first principles in tracking a reference trajectory.

Tasks:

1. Literature review of state-of-the-art architecture for data-driven control systems
2. Develop a data-driven approach using Koopman operator theory for constructing a linear representation of the ADIP plant for the purpose of swing-up and stabilizing control
3. Synthesize a quasi-LPV controller governing the swing-up dynamics and a stabilizing Linear Quadratic Regulator and a stabilizing quasi-LPV controller for the aforementioned data-driven model
4. Simulate and compare the performance of the derived controllers in tracking a reference trajectory
5. Experimentally validate the simulation results
6. Optional: Extend the above framework to synthesizing a stabilizing data-driven quasi-LPV Predictive Controller using Koopman operator techniques(KqLMPC) and compare it with the existing results on the performance of qLMPC control strategy applied on the model obtained from first principles, [2].

References:

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

[2] P. Cisneros and H. Werner, "Wide range stabilization of a pendubot using quasi-lpv predictive control", *IFAC-PapersOnLine*, vol. 52, pp. 164–169, Jan. 2019.

Supervisor:	Patrick Göttsch, M.Sc.
Examiner:	1. Prof. Dr. Herbert Werner
	2. Prof. Dr. Tobias Knopp

Deutscher Titel: Vergleich von datengetriebenen Regelungsalgorithmen basierend auf dem Koopman Operator und LPV-Techniken zur Evaluation der Tracking-Performance eines armgetriebenen invertierten Pendels

Start Date:	16.06.2020
Due Date:	20.01.2021

26.06.2020, Prof. Dr. H. Werner

Hereby I declare that I produced the present work myself only with the help of the indicated aids and sources.

Hamburg, January 20, 2021 Yashasvi Sai Sukhavasi

Abstract

This work investigates the Koopman operator framework's implementation to approximate linear predictors for the Arm-Driven Inverted Pendulum, a nonlinear controlled dynamical system. The Koopman approach promises to apply the tools developed for linear dynamical systems to the dynamical systems defined by the Koopman operator; thus obtaining a linear approximation of a nonlinear system without directly linearizing around a fixed point. This is done by lifting or embedding the underlying nonlinear dynamics into a higher dimensional space where the Koopman operator's evolution is approximately linear. Although linear, the Koopman operator is infinite-dimensional, which renders it unsuitable for application purposes and therefore, a finite-dimensional approximation is computed by applying the Extended Dynamic Mode Decomposition. Such models' tracking performance is evaluated against a model obtained from first-principles linearized around a fixed point. It is seen that the Koopman model exhibits performance superior to a locally linearized model. Importantly, the approach to compute such a model is completely data-driven and extremely simple. It involves only a nonlinear transformation of the measured data (also called *lifting* the dimension of the data) and a linear least-squares problem in the higher dimensional lifted space that can be readily solved even for large data sets. This approximated linear Koopman model can be readily used to synthesize controllers for the underlying nonlinear dynamical system using linear control design methodologies. This work also compares the performance of non-predictive controllers obtained in this way against that of predictive controllers. Predictive controllers such as the Model Predictive Controller offers numerous advantages over non-predictive controllers such as the Linear Quadratic Regulator, more so in the case of data-driven control.

Contents

Introduction

Dynamical systems provide a mathematical framework to analyze, predict and understand the behaviour of systems that co-evolve in time. This formulation encompasses a range of phenomena, including those observed in classical mechanical systems, climate sciences, finance, epidemiology and all other systems that evolve in time. The formulations for such physical systems were traditionally done by making ideal approximations and then deriving simple differential models via Newton's second law of motion. Furthermore, the derivations could often be simplified by exploiting symmetries and different coordinate systems, as highlighted by the success of Lagrangian and Hamiltonian dynamics. The dynamical models obtained through these approaches are known as *first principles models* or *white-box models*.

First-principles formulation, however, becomes demanding and sometimes untenable with the increasing complexity of the state interactions, which is often the case with real-world systems of interest. Also, the derived analytical models do not capture any external disturbances. As a result, first principle models often have limited use or poor prediction over longer time spans in the context of control. Nevertheless, a representation of dynamical systems is central to most engineering and scientific applications.

An alternative to first-principles modelling is experimental system identification, i. e. obtaining a model from the input and output experimental data of a system. Recently, owing to the increased computational abilities, availability of inexpensive sensors and large storage capabilities, an unprecedented amount of data can be recorded and processed easily. Proven data-driven modelling methods are driving the present-day industry towards data-driven modelling of the complex as well as simple dynamical systems. As Brunton et al. [1] observe:

> Modern dynamical systems is currently undergoing a renaissance, with analytical derivations and first principles models giving way to data-driven approaches. The confluence of big data and machine learning is driving a paradigm shift in the analysis and understanding of dynamical systems in science and engineering.

Motivation to Obtain Linear Representations of Nonlinear Systems: Almost all of the real-world systems of interest exhibit nonlinear behaviour. Obtaining linear representations of such nonlinear systems from data can potentially revolutionize the ability to predict and control these systems as there exist a wealth of techniques for the analysis, prediction, numerical simulation, estimation and control of linear dynamical systems. Thus, whenever possible it is desirable to work with linear dynamics of the form,

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} \, , \tag{0.1}$$

the solution to which is given by

$$\mathbf{x}(t_0 + t) = e^{\mathbf{A}t}\mathbf{x}(t_0) \, . \tag{0.2}$$

Such systems admit closed-form solutions and their dynamics are completely characterized by the spectral decomposition of $\mathbf{A}$, also called the *system* matrix:

$$\mathbf{AT} = \mathbf{T\Lambda} \ , \tag{0.3}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_j and $\mathbf{T}$ is a matrix whose columns are linearly independent eigenvectors ξ_j associated with eigenvalues λ_j. In this case, it is possible to write $\mathbf{A} = \mathbf{T\Lambda T^{-1}}$, and the solution in Eq. 0.2 becomes

$$\mathbf{x}(t_0 + t) = \mathbf{T}e^{\mathbf{\Lambda t}}\mathbf{T^{-1}}\mathbf{x}(t_0) \ . \tag{0.4}$$

The matrix $\mathbf{T^{-1}}$ defines a transformation, $\mathbf{z} = \mathbf{T^{-1}x}$, into intrinsic eigenvector coordinate, $\mathbf{z}$, where the dynamics become decoupled:

$$\frac{d}{dt}\mathbf{z} = \mathbf{\Lambda z} \ , \tag{0.5}$$

i. e. each coordinate, z_j, only depends on itself, with simple dynamics given by

$$\dot{z} = \lambda_j z_j \ . \tag{0.6}$$

Thus, it is possible to transform any linear system into eigenvector coordinates where the dynamics become decoupled. No such closed-form solution or simple linear change of coordinates exist in general for nonlinear systems, motivating the search for a linear operator that can express nonlinear dynamics in a linear framework.

Koopman operator theory provides just the framework to represent a nonlinear dynamical system in terms of a linear operator. The Koopman operator advances measurement functions of the state, also called *observables*, to the next time step, thereby making accurate predictions and enabling the possibility of control of those states [2]. Recently, Koopman theory has been increasingly applied to system identification [3]–[6] estimation [7] and control [8], [9] of nonlinear systems. Although the Koopman operator seems like a promising and a simple way to represent nonlinear systems, it must be noted that space spanned by all the measurement functions of the state is usually not bounded and can be described by infinitely many functions of the state. Therefore, the Koopman operator although provides a linear representation of a nonlinear system is infinite-dimensional.
Obtaining a finite-dimensional approximation of the Koopman operator is still an open challenge. Several approaches have been explored to approximate this Koopman operator. Extended Dynamic Mode Decomposition (EDMD) is one such method of approximation where the input-output data is used to regress a finite-dimensional best-fit linear operator that advances measurements of the state or observables forward in time [8], [10]–[12]. Sparse Identification of Nonlinear Dynamics (SINDy) [1], [13] is another data-driven linear regression approach which leverages the fact that many dynamical systems, described by nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, have dynamics $\mathbf{f}$ with only a few active terms in the space of possible right-hand side functions, and it is possible to solve for only these relevant terms that are active in dynamics using sparse regression. The SINDy algorithm can be easily adapted for classical mechanical systems since there is usually some knowledge of the right-hand side functions of the dynamics. Nevertheless, the success of the above-described

methods lies in the choice of observables. This thesis work explores the possibilities of application of data-driven control techniques to an under-actuated nonlinear system, the arm driven inverted pendulum (ADIP). It is of interest to formulate data-driven models and evaluate the performance of data-driven-model-based controllers against that of a first-principles model-based controller.

This thesis work is organized as follows: Chapter 1 introduces the necessary data regression concepts like Dynamic Mode Decomposition (1.1), Koopman operator framework (1.2), Extended Dynamic Mode Decomposition (1.3), Sparse Identification of Nonlinear Dynamics (1.4) which form the foundation for this work. This chapter also discusses an optimization technique used for tuning the controllers called Genetic Algorithm (1.5) which is based on natural selection, the process that drives biological evolution. Furthermore, since Section 3.2 deals with the derivation of first-principles model for the ADIP, Chapter 1 also covers a brief introduction of the Euler-Lagrange method (1.6) to obtain governing equations of motion. Chapter 2 summarises some of the available literature on the application of Koopman operator theory. Chapter 3 describes the arm-driven inverted pendulum and presents the first-principles model of the same (3.2). This chapter also explores the approaches employed to derive a data-driven model of the ADIP (3.3). Chapter 4 gives a brief introduction of the controllers like the LQR (4.2) and MPC (4.3), used in this thesis work. Chapter 5 discusses the simulation of the derived models and compares the performance of the controllers applied to the first-principles model and the data-driven model. Finally, Chapter 6 summarises the work done in this thesis and discusses the future scope in this area.

1 Preliminaries

This chapter will introduce some concepts related to data-driven science and engineering that will be essential throughout the thesis and go beyond what can be considered as common knowledge. Accordingly, there will be no introduction into the basics of control theory. More suitable literature covering these topics is available in abundance, e.g. [14] or [15].

1.1 Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) is a powerful data-driven modelling technique that is used for the discovery of dynamical systems, usually from high dimensional data. The method first originated in the fluid dynamics community as a method to decompose complex flows into a simple representation based on spatiotemporal coherent structures. At its core, the DMD can be thought of as an ideal combination of spatial dimensionality-reduction techniques, such as the proper orthogonal decomposition (POD), with Fourier transforms in time. This method relies simply on collecting snapshots of data $\mathbf{x}_k \in \mathbb{R}^n$ from a dynamical system at several times t_k, where $k = 1, 2, 3, \dots, M$, and regressing this data onto locally linear dynamics

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k \ , \tag{1.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is chosen to minimize

$$\|\mathbf{x}_{k+1} - \mathbf{A}\mathbf{x}_k\|_2 \tag{1.2}$$

over the $k = 1, 2, 3, \dots, M$ snapshots. The DMD is simple to execute and makes almost no assumptions about the underlying system. The cost of the algorithm is a singular value decomposition (SVD) of the snapshot matrix constructed from the data $\mathbf{x}_k$. The optimality of this approximation though holds only over the sampling window where $\mathbf{A}$ is constructed. The DMD architecture is explained in brief as follows:

- The data $\mathbf{x}_k$ is collected across M snapshots and is arranged into two large data matrices $\mathbf{X}$ and $\mathbf{Y}$, where $\mathbf{Y}$ is simply the time-shifted version of $\mathbf{X}$ i.e., $\mathbf{X}^+$, in the case of discrete-time data.
  It is important to note that the data can (and is in-fact recommended to) be obtained from multiple trajectories, N, to sufficiently capture the dynamics of the system in question. The data is then arranged as:

$$\mathbf{X} = \begin{bmatrix} | & | & & | & | & | & & | \\ \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \dots & \mathbf{x}_{1,M}, & \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \dots & \mathbf{x}_{N,M} \\ | & | & & | & | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times NM} \ , \tag{1.3}$$

$$\mathbf{Y} = \begin{bmatrix} | & | & & | & | & | & & | \\ \mathbf{y}_{1,1} & \mathbf{y}_{1,2} & \dots & \mathbf{y}_{1,M}, & \mathbf{y}_{2,1} & \mathbf{y}_{2,1} & \dots & \mathbf{y}_{N,M} \\ | & | & & | & | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times NM} \ . \tag{1.4}$$

An equivalent representation also exists for continuous-time data where the time-shifted matrix is replaced by a matrix of derivatives of the corresponding state. But for this thesis work, all the data captured is discrete-time data and therefore, the formulations will be in discrete-time unless specified otherwise.
The locally linear approximation in Eq. 1.1 can then be written in terms of these data matrices as

$$\mathbf{Y} \approx \mathbf{A}\mathbf{X}\ . \tag{1.5}$$

The best fit matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is then given by

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^{\dagger} \tag{1.6}$$

where † is the Moore-Penrose pseudoinverse. The psuedoinverse is equivalent to finding the best-fit solution in the least square sense. Thus, this solution minimizes the error

$$\|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F \tag{1.7}$$

where $\|.\|_F$ is the Frobenius norm given by

$$\|\mathbf{X}\|_F = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m} X_{jk}^2}\ . \tag{1.8}$$

Note that the best-fit linear operator $\mathbf{A}$ can be solved in the above shown way for low-dimensional data only.

In case, the snapshots $\mathbf{x}_k$ are high dimensional, i. e. when the dimension of the measurement vector or the dimension of the observables vector is huge, the resulting matrix $\mathbf{A}$ also will be high dimensional which maybe difficult to represent or decompose. Therefore, instead of solving for $\mathbf{A}$ directly, the data is first projected onto a low-rank subspace defined by POD modes and then solved for a low-dimensional evolution of $\tilde{\mathbf{A}}$ that evolves on these POD mode coefficients. The DMD algorithm then uses this low dimensional evolution $\tilde{\mathbf{A}}$ to reconstruct the leading nonzero eigenvalues and eigenvectors of the full-dimensional operator $\mathbf{A}$ without ever explicitly computing $\mathbf{A}$. The eigenvalues and eigenvectors of the linear operator $\mathbf{A}$ are of particular importance, and more will be discussed about their relevance in further sections. The procedure for computing them is discussed in brief as follows:

1. First, compute the SVD of $\mathbf{X}$:

$$\mathbf{X} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*} \tag{1.9}$$

where * denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{n\times r}$, $\mathbf{\Sigma} \in \mathbb{C}^{r\times r}$, and $\mathbf{V} \in \mathbb{C}^{m\times r}$. Here r is the rank of the reduced SVD approximation to $\mathbf{X}$. This is selected on a subjective basis. The left singular vectors $\mathbf{U}$ are POD modes.

2. Compute $\tilde{\mathbf{A}} \in \mathbb{R}^{r\times r}$, the projection of the full matrix $\mathbf{A}$ onto POD modes.

$$\tilde{\mathbf{A}} = \mathbf{U}^{*}\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\ . \tag{1.10}$$

The matrix $\tilde{\mathbf{A}}$ defines a low-dimensional linear model of the dynamical system on POD coordinates:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k \tag{1.11}$$

3. Compute the eigendecomposition of $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda} \tag{1.12}$$

where columns of $\mathbf{W}$ are eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix of corresponding eigenvalues λ_k.

4. Finally, the eigendecomposition of $\mathbf{A}$ is reconstructed from $\mathbf{W}$ and $\boldsymbol{\Lambda}$. The eigenvalues of $\mathbf{A}$ are given by $\boldsymbol{\Lambda}$ and the eigenvectors of $\mathbf{A}$ are given by columns of $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \mathbf{Y}\mathbf{V}\boldsymbol{\Sigma}^{-\mathbf{1}}\mathbf{W} \ . \tag{1.13}$$

The columns of $\boldsymbol{\Phi}$ are also known as the DMD modes. The DMD modes are spatially coherent and oscillate and/or grow or decay at the fixed frequency λ.

With the low-rank approximations of both the eigenvalues and eigenvectors, the projected future solution can be constructed for all time in the future. Two important advantages of the DMD framework is its simple formulation in terms of well-established techniques from linear algebra, and that it is formulated entirely in terms of measurement data. For this reason, it may be applied to a broad range of applications including fluid dynamics, epidemiology, neuroscience, finance etc.
The key takeaways from the DMD approach for the context of this thesis work though, are the estimation of linear matrices $\mathbf{A}$ and $\tilde{\mathbf{A}}$, the computation of DMD modes $\boldsymbol{\Phi}$ and the DMD eigenvalues $\boldsymbol{\Lambda}$. Most importantly, these can be connected to the Koopman operator theory, with the DMD method viewed as computing the eigenvalues and eigenvectors of a finite-dimensional linear operator that approximates the infinite-dimensional Koopman operator, introduced in the following section.

1.2 Koopman Operator Framework

Koopman operator theory [16] proposed by B.O.Koopman in 1931, has recently emerged as a promising framework to embed nonlinear dynamics in a global linear representation. The spectral decomposition of this linear Koopman operator completely characterizes the behaviour of a nonlinear system analogous to Eq. 0.1 [2] thus enabling the use of a wealth of linear control techniques to control a dynamical system.
The Koopman operator is formally defined as follows:

Definition 1.1 *Consider a continuous-time dynamical system*

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) \ , \tag{1.14}$$

where $\mathbf{x} \in \mathcal{M}$ *is a state on a smooth n-dimensional manifold* $\mathcal{M}$ *and* $\mathcal{M} \subseteq \mathbb{R}^n$*. The Koopman operator* $\mathcal{K}$ *is an infinite dimensional linear operator that acts on functions of state space,* $g \in \mathcal{F}$*, with* $g : \mathcal{M} \to \mathbb{C}$ *so that*

$$\mathcal{K}g(\mathbf{x}) = g(\mathbf{f}(\mathbf{x})) \ . \tag{1.15}$$

The definition (1.15) can alternatively be represented by a composition of the observables with the nonlinear evolution: $\mathcal{K}g = g \circ \mathbf{f}$.
It is assumed that, $\mathcal{F} = L^2(\mathcal{M}, \rho)$, where ρ is a positive, single valued analytic function defined on $\mathcal{M}$, but not necessarily an invariant measure of the underlying dynamical system. This assumption made before in the literature by Koopman [16] and Budisic et al. [17], is required so that the inner products can be taken in $\mathcal{F}$. The Koopman operator is also defined for discrete-time dynamical systems. The dynamical system (1.14) will induce a discrete-time dynamical system $(\mathcal{M}, k, \mathbf{F})$ where $k \in \mathbb{Z}$ is an integer index and the flow map $\mathbf{F} : \mathcal{M} \to \mathcal{M}$, where $\mathbf{F}$ is the evolution operator that maps the state $\mathbf{x}(t_0)$ to a future time state $\mathbf{x}(t_0 + t)$:

$$\mathbf{F}(\mathbf{x}(t_0)) = \mathbf{x}(t_0 + t) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau))\, d\tau \ . \tag{1.16}$$

This induces the discrete-time dynamical system

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k) \ , \tag{1.17}$$

where $\mathbf{x}_k = \mathbf{x}(kt)$. The analogous discrete-time Koopman operator is then given by $\mathcal{K}$ such that

$$g(\mathbf{x}_{k+1}) = g(\mathbf{F}(\mathbf{x}_k)) = \mathcal{K}g(\mathbf{x}_k) \ . \tag{1.18}$$

As can be seen from the above equations, the Koopman operator maps functions of state space to functions of state space and not states to states. In other words, the Koopman operator defines a new dynamical system, $(\mathcal{F}, k, \mathcal{K})$, that evolves the observables $g \in \mathcal{F}$ in discrete-time.

The idea of the Koopman operator theory is to analyse the flow dynamics governed by (1.17) or its continuous-time equivalent (1.14), only from available data - collected either numerically or experimentally using the eigenfunctions and eigenvalies of $\mathcal{K}$. To this end, let $\varphi_j : \mathcal{M} \to \mathbb{R}$ denote eigenfunctions and $\lambda_j \in \mathbb{C}$ denote eigenvalues of the Koopman operator,

$$\mathcal{K}\varphi_j = \lambda_j \varphi_j \ , \qquad j = 1, 2, \ldots, \infty \tag{1.19}$$

The dynamical system defined by $\mathbf{F}$ in (1.17), and the one defined by $\mathcal{K}$ in (1.18), represent a system with the same fundamental behavior. The link between the two representations is the *full state observable*, $\boldsymbol{g}(\mathbf{x}) = \mathbf{x}$ and the *Koopman mode decomposition* [18], $\{(\lambda_j, \varphi_j, \boldsymbol{v}_j)\}_{j=1}^{K}$, the set of K tuples of Koopman eigenvalues, eigenfunctions and modes required to reconstruct the full state. K could (and often will) be infinite. Although $\boldsymbol{g}$ is a vector valued observable, each component of it is a scalar valued observable, i.e., $g_i \in \mathcal{F}$ where g_i is the i-th component of $\boldsymbol{g}$. Assuming g is in the span of the set of K eigenfunctions, it can be expanded [2] as

$$g_i(\mathbf{x}) = \sum_{j=1}^{K} v_{ij}\varphi_j(\mathbf{x}), \qquad v_{jk} \in \mathbb{C}.$$

Then $\boldsymbol{g}$ can be obtained by "stacking" these weights into vectors i.e., $\boldsymbol{v}_j = [v_{1j}, v_{2j}, \ldots, v_{nj}]^\top$. As a result, the full-state observable $\boldsymbol{g}(\mathbf{x}) = \mathbf{x}$ can be written as

$$\mathbf{x} = \boldsymbol{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{bmatrix} = \sum_{j=1}^{K} \boldsymbol{v}_j \varphi_j(\mathbf{x}) \ , \tag{1.20}$$

where $\boldsymbol{v_j}$ is the jth Koopman mode associated with the jth Koopman eigenfunction φ_j. In summary, Koopman modes are the weights needed to expres the full state in the Koopman eigenfunction basis.
Given the decomposition in Eq. 1.20, the system state at future times can be obtained either by directly evolving $\mathbf{x}$ as in (1.18) or by evolving the full state observable through Koopman:

$$\mathbf{F}(\mathbf{x}) = \mathcal{K}g(\mathbf{x}) = \mathcal{K}\sum_{j=1}^{K} \varphi_j(\mathbf{x})\boldsymbol{v}_j = \sum_{j=1}^{K} \mathcal{K}\varphi_j(\mathbf{x})\boldsymbol{v}_j = \sum_{j=1}^{K} \lambda_j\varphi_j(\mathbf{x})\boldsymbol{v}_j \tag{1.21}$$

This representation of $\mathbf{F}(\mathbf{x})$ is advantageous because its corresponding eigenvalue determines the dynamics associated with each eigenfunction.
Furthermore, the Koopman mode decomposition can be connected to the dynamic mode decomposition algorithm introduced in the previous chapter. It is then possible to determine finite dimensional approximation of Koopman eigenvalues and modes directly from data under suitable conditions through the DMD, as shown by Rowley et al. [18]. Importantly, the choice of *observables* or the measurement functions play a crucial role in the success of the Koopman method. The DMD algorithm, as presented in the previous section, will now run on an augmented set of data matrices, which are the data matrices of the observables. This is further explored in the next sections.

Another important concept to take note of is the Koopman invariant subspace defined in [12].

Definition 1.2 *A Koopman invariant subspace is given by span*$\{g_1, g_2, \dots, g_p\}$ *if all functions* g *in this subspace,*

$$g = \alpha_1 g_1 + \alpha_2 g_2 + \cdots + \alpha_p g_p \ , \tag{1.22}$$

remain in the subspace after being acted on by the Koopman operator $\mathcal{K}$*:*

$$\mathcal{K} g = \beta_1 g_1 + \beta_2 g_2 + \cdots + \beta_p g_p \ . \tag{1.23}$$

Instead of capturing the evolution of all measurement functions in a Hilbert space, applied Koopman analysis approximates the evolution on a subspace spanned by a finite set of measurement functions, which is the Koopman invariant subspace. For functions in these invariant subspaces, it is possible to restrict the Koopman operator to this subspace, yielding a finite-dimensional linear operator $\mathbf{K}$. $\mathbf{K}$ acts on a vector space $\mathbb{R}^p$, with the coordinates given by the values $g_k(\mathbf{x})$. Any finite set of eigenfunctions of the Koopman operator will span an invariant subspace, and importantly, these eigenfunctions provide intrinsic coordinates along which the dynamics behave linearly.

The key takeaway from Eq. 1.21, and this section in general, is that the Koopman operator captures everything about the nonlinear dynamical system (1.14), and its eigenfunctions define a nonlinear change of coordinates in which the system becomes linear. Of particular interest here is the "slow" subspace of the Koopman operator, which is the span of the eigenfunctions associated with the eigenvalues near the unit circle in discrete time or near the imaginary axis in the continuous-time. These eigenvalues and eigenfunctions capture the long term dynamics of observables that appear after the fast transients have subsided and could serve as a low dimensional approximation of the otherwise infinite-dimensional operator when a spectral gap, which clearly delineates the "fast" and "slow" temporal dynamics, is present [6]. In principle, this framework is quite broadly applicable, and useful even for problems with multiple attractors that cannot be accurately approximated using models based on local linearization.

Thus, discovering Koopman eigenfunctions enables globally linear representations of nonlinear systems, but discovering these eigenfunctions is challenging and still an open problem. Several new machine learning techniques seek to identify relevant terms in dynamics from data. Two of the most popular techniques, the Extended Dynamic Mode Decomposition and the Sparse Identification of Nonlinear Dynamics [13] are discussed in the following sections.

1.3 Extended Dynamic Mode Decomposition

The extended dynamic mode decomposition(EDMD) first introduced by Williams et al. [6] aims to approximate the Koopman operator and therefore the Koopman eigenvalue, eigenfunction and modes solely from data and can be considered as an extension to the DMD which approximates only the Koopman eigenvalue and modes. The EDMD also allows for nonlinear measurements of state to be included in the vector of observables to better approximate the dynamical interactions. Thus, the system is first *lifted* into a higher dimensional space that is defined by a set of nonlinear functions. The original state space can also be included in this higher dimensional space, thereby making the best of both worlds i. e. obtaining a nearly accurate linear representation of the nonlinear dynamics and the availability of the original state space for control, as will be illustrated in the next chapters. An important extension of the EDMD is the EDMD with control (EDMDc) which enables incorporation of input measurements into the vector of observables to disambiguate the effect of internal dynamics from actuation or external inputs [19]. This method is beneficial in identifying actuated dynamical systems and enable closed-loop feedback control, for example, it has been used by Korda et al. for feedback control of a bilinear motor through Koopman driven model predictive control [8].

The EDMD approach requires only

1. a data set of snapshot pairs of the system state, $\{\mathbf{x}_k, \mathbf{x}_{k+1}\}_{k=1}^{M}$ arranged as in Eqs. 1.3 and 1.4, and
2. a *dictionary* of observables, $\mathcal{D} = \{\psi_1, \quad \psi_2, \ \ldots, \ \psi_K\}$ where $\psi_i \in \mathcal{F}$, whose span is denoted as $\mathcal{F}_{\mathcal{D}} \subset \mathcal{F}$. $\mathcal{D}$ must be "rich enough" to accurately approximate a few of the leading Koopman eigenfunctions.

Although the selection of observables which most closely approximate the dynamics of a system is still an open question, there are several approaches to choose the best possible observables. For example,

- they can be selected based on physical insights of the system,
- partial or complete knowledge of the right hand side functions of the dynamical system,
- as Fourier basis functions, Radial basis functions, Hermite polynomials etc.

The list is not exhaustive. However, it is always advised to run a quick check to validate the behaviour of the system for the choice of observables. It is possible that the evolution of some observables is not bounded, resulting in an infinite-dimensional linear operator.

Approximating the Koopman operator through the EDMD approach [6]:

- The state measurements are collected as detailed in Eqs. 1.3 and 1.4.
- A vector of observables defined by functions of state measurements is constructed to represent the dynamics of the system in terms of a *lifted* state, $\mathbf{\Psi}(\mathbf{x}) : \mathcal{M} \to \mathbb{C}^{1\times K}$,

$$\mathbf{\Psi}(\mathbf{x}) = [\psi_1(\mathbf{x}) \quad \psi_2(\mathbf{x}) \ \ldots \ \psi_K(\mathbf{x})] \ . \tag{1.24}$$

  The data set of snapshots is typically constructed from multiple short bursts of simulation or experimental data. $\mathbf{\Psi}(\mathbf{x})$ may contain the original state $\mathbf{x}$ as well as nonlinear measurements, so often $K \gg n$. For this thesis work, the scalar measurements of the state form the first n entries of $\mathbf{\Psi}(\mathbf{x})$ and the remaining $(K-n)$ observables are nonlinear functions of the state measurements.
- The EDMD then seeks to approximate a linear operator $\mathcal{A} \in \mathbb{R}^{K\times K}$ that approximately relates the lifted measurements of state, at least for short periods of time:

$$\mathbf{\Psi}(\mathbf{x}_{k+1}) \approx \mathcal{A}\mathbf{\Psi}(\mathbf{x}) \ . \tag{1.25}$$

  By definition, a function $g \in \mathcal{F}_\mathcal{D}$ can be written as

$$g = \sum_{j=1}^{K} a_j \psi_j = \mathbf{\Psi}\mathbf{a} \ , \tag{1.26}$$

  the linear representation of the K elements in the dictionary with the weights $\mathbf{a}$. Because $\mathcal{F}_\mathcal{D}$ is typically not an invariant subspace of $\mathcal{K}$,

$$\mathcal{K}g = (\mathbf{\Psi} \circ \mathbf{F})\mathbf{a} = \mathbf{\Psi}(\mathcal{A}\mathbf{a}) + r \tag{1.27}$$

  which includes the residual term $r \in \mathcal{F}$. $\mathcal{A}$ is determined by minimizing,

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{M} |r(\mathbf{x}_k)|^2 \ , \\ &= \frac{1}{2}\sum_{k=1}^{M} |((\mathbf{\Psi} \circ \mathbf{F})(\mathbf{x}_k) - \mathbf{\Psi}(\mathbf{x}_k)\mathcal{A})\mathbf{a}|^2 \\ &= \frac{1}{2}\sum_{k=1}^{M} |(\mathbf{\Psi}(\mathbf{x}_{k+1}) - \mathbf{\Psi}(\mathbf{x}_k)\mathcal{A})\mathbf{a}|^2 \ . \end{aligned} \tag{1.28}$$

  Eq. 1.28 is a least squares problem, and therefore cannot have multiple isolated local minima; it must either have a unique global minimizer or a continuous family of minimizers. As a result, regularization may be required to ensure the solution is unique and the $\mathcal{A}$ that minimizes (1.28) is:

$$\mathcal{A} = \mathbf{G}_1^{\dagger}\mathbf{G}_2 \ , \tag{1.29}$$

where

$$\mathbf{G}_1 = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\Psi}(\mathbf{x}_k)^*\boldsymbol{\Psi}(\mathbf{x}_k)\ , \tag{1.30}$$

$$\mathbf{G}_2 = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\Psi}(\mathbf{x}_k)^*\boldsymbol{\Psi}(\mathbf{x}_{k+1})\ . \tag{1.31}$$

with $\mathcal{A}, \mathbf{G}_1, \mathbf{G}_2 \in \mathbb{C}^{K\times K}$. Thus, $\mathcal{A}$ is a finite dimensional linear operator that maps $g \in \mathcal{F}_\mathcal{D}$ to some other $\hat{g} \in \mathcal{F}_\mathcal{D}$ by minimizing the residuals at the data points. It is interesting to note that $\mathcal{A}$ is the transpose of the finite dimensional approximation of the Koopman operator and is exactly equal to the finite dimensional Koopman operator approximated through DMD on special set of measurements: the measurements of full-state. As a consequence, if $\boldsymbol{\xi}_j$ is the j-th right eigenvector of $\mathcal{A}$ with the eigenvalue λ_j, then the EDMD approximation of an eigenfunction of $\mathcal{K}$ is

$$\varphi_j = \boldsymbol{\Psi}\boldsymbol{\xi}_j\ . \tag{1.32}$$

And consequently a vector valued function $\boldsymbol{\Phi} : \mathcal{M} \to \mathbb{C}^K$ where,

$$\boldsymbol{\Phi}(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_K(\mathbf{x})] \tag{1.33}$$

can be written as:

$$\boldsymbol{\Phi}(\mathbf{x}) = \boldsymbol{\Psi}(\mathbf{x})\Xi, \qquad \Xi = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_K]\ . \tag{1.34}$$

The computational burden of this approach grows as the dimension of $\boldsymbol{\Psi}(\mathbf{x})$ increases. However, also, this approach generally yields a better approximation as the dimension of $\boldsymbol{\Psi}(\mathbf{x})$ increases. Therefore, the choice of observables is crucial. Furthermore, the number of data points and their distribution across the state-space will have a large effect on the computed $\mathcal{A}$ matrix, since the optimality of the approximation holds only in the region of the obtained data.

- EDMD also computes the approximation of Koopman modes for the full state observable. Here, only the final result is presented. An in detail derivation can be found in [6]. The full state observable can be expressed as,

$$\boldsymbol{g}(\mathbf{x}) = \boldsymbol{V}\boldsymbol{\Psi}(\mathbf{x})^\top = \sum_{j=1}^{K} \boldsymbol{v}_j\varphi_j(\mathbf{x}), \qquad \boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K] = \mathbf{W}^*\mathbf{B}^\top \tag{1.35}$$

where $\boldsymbol{v}_i = (\mathbf{w}_i^*\mathbf{B})^\top$ is the i-th Koopman mode and $\mathbf{w}_i$ is the i-th left eigenvector of $\mathcal{A}$ associated with λ_i, and $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n], \mathbf{B} \in \mathbb{C}^{K\times n}$ is some appropriate vector of weights:

$$g_i(\mathbf{x}) = \sum_{j=1}^{K} \psi_j(x)b_{j,i} = \boldsymbol{\Psi}(\mathbf{x})\mathbf{b}_i.$$

In summary, EDMD computes a finite dimensional approximation of the Koopman operator from snapshot pairs of a data set, a dictionary of observables and the assumption that the leading Koopman eigenfunctions are (nearly) contained within $\mathcal{F}_{\mathcal{D}}$. Furthermore, the eigenvalues of $\mathcal{A}$ are the EDMD approximations of the Koopman eigenvalues, the right eigenvectors of $\mathcal{A}$ generate the approximations of the eigenfunctions and the left eigenvectors of $\mathcal{A}$ generate the approximations of the Koopman modes.

As previously discussed, the EDMD approach can readily be extended to actuated systems of the form

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u} \ , \tag{1.36}$$

to estimate the approximate $\mathbf{A}$, and $\mathbf{B}$ matrices. The approach employed is largely similar to the one described above. The state measurements are first lifted to a higher dimensional $\mathbf{\Psi}(\mathbf{x})$ and additionally, the corresponding input measurements are recorded as

$$\mathbf{u} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2, & \dots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} \ . \tag{1.37}$$

$\mathbf{\Gamma}(\mathbf{u}_k)$ can simply be scalar measurements of input. For the purpose of this thesis work, $\mathbf{\Gamma} \in \mathbb{R}^m$, is a vector of scalar measurements of input. The state observables vector, (1.24), and the input vector, (1.37), are then combined to form an augmented observables vector, $\mathbf{\Omega} \in \mathbb{C}^{(K+1)\times m}$

$$\mathbf{\Omega}(\mathbf{x}, \mathbf{u}) = [\mathbf{\Psi} \quad \mathbf{\Gamma}]^\top \ . \tag{1.38}$$

The EDMDc operator then seeks to approximate the system in Eq. 1.36:

$$\mathbf{\Omega}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) = [\hat{\mathbf{A}} \ \ \hat{\mathbf{B}}] \begin{bmatrix} \mathbf{\Psi}_k \\ \mathbf{\Gamma}_k \end{bmatrix} \ , \tag{1.39}$$

where the approximate system matrix $\hat{\mathbf{A}} \in \mathbb{R}^{K\times K}$ and the approximate input matrix $\hat{\mathbf{B}} \in \mathbb{R}^m$ can be easily computed by decomposition of the Koopman operator in (1.29) obtained by replacing $\mathbf{\Psi}(\mathbf{x_k})$ with the corresponding augmented observables vector $\mathbf{\Omega}(\mathbf{x_k}, \mathbf{u_k})$ in Eqs. [1.29 - 1.31].

In summary, the enriched dictionary of observables $\mathcal{D}$ provides a larger basis in which the Koopman operator can be approximated. It has been shown that in the limit of infinite snapshots, the EDMD operator converges to the Koopman operator projected onto the subspace spanned by $\mathcal{D}$ [20]. However, it must be noted that if $\mathcal{D}$ does not span a Koopman invariant subspace, then the projected operator may not have any resemblance to the original Koopman operator, as all of the eigenvalues and eigenvectors may be different. In fact, it was shown that EDMD operator would have spurious eigenvalues and eigenvectors unless it is represented in terms of a Koopman invariant subspace [12]. Therefore, it is essential to use validation and cross-validation techniques to ensure that the EDMD models are not overfitting. This is possible by leveraging the SINDy regression [13] to identify Koopman eigenfunctions corresponding to a particular eigenvalue λ, selecting only a few active terms in the dictionary $\mathcal{D}$ to avoid overfitting.

1.4 Sparse Identification of Nonlinear Dynamics

Sparse identification of Nonlinear dynamics (SINDy) technique is another equation-free data-driven method to identify a dynamical system. SINDy re-envisions the discovery of dynamical system explored previously through EDMD from the perspective of sparse regression. Specifically, Brunton et al. [13] take advantage of the fact that most physical systems have only a few relevant terms that define the dynamics, therefore, making the governing equations sparse in a high-dimensional nonlinear function space. The SINDy algorithm applied to discrete-time data [12], is briefly described as follows:

- Collect measurements of state as in Eqs. 1.3 and 1.4 and augment them into a larger vector $\mathbf{\Theta}(\mathbf{x})$, where $\mathbf{\Theta}(\mathbf{x})$ is a dictionary of all possible candidate terms in the right hand side dynamics $\mathbf{F}_t$ in Eq. 1.17.

$$\mathbf{\Theta}(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) \\ \theta_2(\mathbf{x}) \\ \vdots \\ \theta_L(\mathbf{x}) \end{bmatrix} \tag{1.40}$$

  $\mathbf{\Theta}(\mathbf{x})$ here is similar to $\mathbf{\Psi}(\mathbf{x})$ introduced in the previous section, except that $\mathbf{\Theta}(\mathbf{x})$ almost always contains a lot more candidate terms than in $\mathbf{\Psi}(\mathbf{x})$, $L \gg K$.
  Note that representation of $\mathbf{\Theta}(\mathbf{x})$ and the following equations in this section are taken from [12]. These are equivalent to the equations in the original SINDy algorithm in [13]. The equations in [12] are merely the transposed version of the original.

- The following matrix system of equations is then set up:

$$\begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3, & \dots & \mathbf{x}_m \\ | & | & & | \end{bmatrix} = \begin{bmatrix} — \ \xi_1^\top \ — \\ — \ \xi_2^\top \ — \\ \vdots \\ — \ \xi_n^\top \ — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{\Theta}(\mathbf{x}_1) & \mathbf{\Theta}(\mathbf{x}_2), & \dots & \mathbf{\Theta}(\mathbf{x}_{m-1}) \\ | & | & & | \end{bmatrix} , \tag{1.41}$$

  Eq. 1.41 can be written in matrix short-hand as:

$$\mathbf{X}^+ = \mathbf{\Xi}^\top \mathbf{\Theta}(\mathbf{X}) \ , \tag{1.42}$$

  Note that the above equation is considered only for one trajectory ($N = 1$) for convenience of representation. It is however recommended that the data matrices are constructed from multiple trajectories spanning the desired state space. The data from multiple trajectories is arranged as previously shown in Eqs. 1.3 and 1.4. The row vectors $\xi_\mathbf{k}^\top$ ($k = 1, 2, \dots, n$) determine which nonlinear terms in $\mathbf{\Theta}(\mathbf{x})$ are active in the k-th row of $\mathbf{F}_t$. Typically, $\xi_\mathbf{k}$ will be a sparse vector, since only a few terms are active in the right hand side of many dynamical systems of interest. In this case, sparse regression is used to solve for each sparse row $\xi_\mathbf{k}^\top$. The sparse matrix $\mathbf{\Xi}^\top$ then yields a nonlinear discrete-time model for Eq. 1.3, obtained purely from data:

$$\mathbf{x}_{k+1} = \mathbf{\Xi}^\top \mathbf{\Theta}(\mathbf{x}_k) \ . \tag{1.43}$$

Note that $\boldsymbol{\Theta}(\mathbf{x})$ is a vector of symbolic functions of elements of $\mathbf{x}$ (Eq. 1.40), as opposed to $\boldsymbol{\Theta}(\mathbf{X})$, which is a data matrix.
With the active terms in the nonlinear dynamics identified as the nonzero entries in the rows of $\Xi^\top$, it is possible to include these funcitons in the Koopman subspace. $\Xi^\top$ is only a mapping from the observable state-space to the original state space and is not the Koopman operator. However, if $\boldsymbol{\Theta}(\mathbf{x}) = \mathbf{x}$, the problem in Eq. 1.42 reduces to the standard DMD problem:

$$\mathbf{X}^{+} = \Xi\mathbf{X} \,, \tag{1.44}$$

and $\Xi \in \mathbb{R}^{n\times n}$ becomes the linear operator that advances the state of the system.

Similar to EDMDc, the SINDy method can also be generalized to include inputs and control [21]. This only requires building a larger library $\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})$ of candidate functions that include $\mathbf{u}$ and consequently measurements of state $\mathbf{x}$ and input $\mathbf{u}$. The candidate functions can also include nonlinear cross terms in $\mathbf{x}$ and $\mathbf{u}$.
If the input $\mathbf{u}$ corresponds to external forcing, the Eq. 1.42 can be solved for sparse coefficients Ξ:

$$\mathbf{X}^{+} = \Xi^\top\boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\Gamma}) \,. \tag{1.45}$$

However, if the input $\mathbf{u}$ corresponds to a feedback control signal, where $\mathbf{u} = \mathbf{k}(\mathbf{x})$, then the SINDy regression becomes ill-conditioned and it will be impossible to disambiguate the effect of feedback control $\mathbf{u}$ with internal feedback terms $\mathbf{k}(\mathbf{x})$ within the dynamical system. In this case, the feedback input $\mathbf{u}$ is identified as a function of the state:

$$\boldsymbol{\Gamma} = \Xi_u^\top\boldsymbol{\Theta}(\mathbf{X}) \,. \tag{1.46}$$

And the coefficients of the states are identified by perturbing the input u with a sufficiently large white noise or occasionally kicking the system with a large impulse or step in $\mathbf{u}$ to distinguish the input $\mathbf{u}$ from $\mathbf{k}(\mathbf{x})$ terms.

Brunton et al. [21] demonstrate the above detailed SINDy with control algorithm on the Lorenz equations with external forcing and feedback control.

1.5 Genetic Algorithm

Genetic algorithm (GA) is a search heuristic to find globally optimizing solutions for both constrained and unconstrained optimization problems inspired by Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are the most probable to be selected to reproduce offspring of the next generation. Thus, the population evolves toward an optimum solution over successive generations. In this thesis work, this notion of natural selection is applied to find optimal tuning parameters for controllers. This is particularly advantageous for tuning controllers applied to data-driven models as it enables quick and reliable solutions to the optimizing problems for controlling data-driven models. The concept and the application of GA are vast, and this section provides only the basic knowledge required to execute a GA in Matlab. Importantly, there are five phases considered in a genetic algorithm.

1. **Initial Population:** The process of natural selection starts with a set of *individuals* called *population*, where each individual is a solution to the problem one wants to solve. An individual is characterized by a set of variables known as *genes*. To understand this better, consider for example, the problem of finding optimal tuning parameters $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ for a linear quadratic regulator problem:

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,n} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,n} \\ \vdots & \vdots & & \vdots \\ q_{n,1} & q_{n,2} & \cdots & q_{n,n} \end{bmatrix} \qquad \text{and} \qquad \mathbf{R} = diag(r_1, r_2, \ldots, r_m) \ .$$

   Here, the entries of the matrix **Q** and **R** are the variables (the genes) and the set of variables $[q_{1,1}, \ldots, q_{n,n}, r_1, ...r_m]$ is a solution (an individual), and a collection of such sets of individuals forms a population. Care must be taken in choosing the initial population as the 'evolution' naturally depends on it.

2. **Fitness Function:** The fitness function or simply, the cost function determines the ability of an individual to compete with other individuals, i.e. it decides how to fit an individual through a *fitness score*. The individuals with the highest fitness scores have a high probability to be selected for reproduction. The fitness function can be anything subject to the requirement. For example, it can evaluate performance measures such as rise time, settling time, overshoot, tracking error, etc.

3. **Selection:** The idea of selection phase is to select the fittest individuals and pass their genes to the next generation.

4. **Crossover:** Crossover is the most significant phase in a GA. In this phase, a randomly chosen number of genes specified by the crossover point is exchanged between the parents (fittest individuals) to create offspring. The new offspring are then added to the population.

5. **Mutation:** In certain new offspring, some of the genes can be subjected to mutation with a low random probability. Mutation ensures diversity within the population and prevents premature convergence.

6. **Termination:** Finally, the algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation) or if the number of specified generations is reached. The GA provides a set of solutions to the problem when this happens.

The crossover and mutation options used in this thesis, *crossoverscattered* and *mutationgaussian*, respectively, were the GA's default options toolbox. Likewise, the crossover fraction is also unchanged from the default value of 0.8. More information on the options can be found on the Matlab help page for the toolbox. Furthermore, a fitness limit of 0.02 was specified after some initial trials and observations. This is subjective to the fitness function and objective at hand. A few general guidelines when creating a genetic algorithm can help in the successful implementation of the same. They are enlisted as follows:

- Population size: Generally, the population size depends on the number of genes. Longer genes would mean a larger population size as the solution space has a higher dimension.
- Number of generations: The maximum number of generations should be chosen such that the algorithm ideally terminates when the specified fitness value is reached.

However, the above-presented guidelines are not hard and fast rules; they only serve as a guide to initiate a GA. Consequently, in this thesis work, six genes are defined for LQi tuning. Five represent the states (original state of the system + integrated error of arm angle, ϕ_1) and the remaining gene represents the penalty on input. Similarly, the MPC problem has five genes. As an initial guess, population size and a generation size of 10 each produced reasonably good results. The computation, however, stopped as the number of generations were reached and not because the fitness value was achieved.

The next important task is formulating the fitness function. Again, this is subjective to the requirements of the designer. The fitness function returns a scalar fitness value used to select the 'elite' genes. One can specify various performance objectives to calculate this cost, for example, rise time, settling time, peak overshoot etc. . In this thesis work, however, the requirement was to track a reference trajectory as closely as possible and therefore, the fitness function consisted of a simple quadratic error of the states and the reference,

$$J = \sum_k (\mathbf{x}_k - \mathbf{r}_k)^2 \ , \tag{1.47}$$

where k is the discrete-time index, and r_k is the desired reference trajectory. Since the tracking of arm angle ϕ_1 while maintaining the pendulum in the upright position is the goal of this thesis, one can further simplify the computation of the above-specified cost function by just considering the quadratic error of the states (ϕ_1 and $\dot{\phi}_1$). Furthermore, in this thesis, the quadratic errors of the two states were weighted as follows:

$$J = \sum_k (\mathbf{x}_k - \mathbf{r}_{x_k})^2 + (\dot{\mathbf{x}}_k - \beta \mathbf{r}_{\dot{x}_k})^2 \ , \qquad \beta = 0.1 \ . \tag{1.48}$$

The value of β can be varied as per the requirement of the user. In this thesis, it was required to track ϕ_1 with reasonably good rise and settling times, and therefore it was intuitively chosen as $\beta = 0.1$.

1.6 Euler-Lagrange Formulation

The *equations of motion* describe how a rigid body or a system moves under the influence of a force as a function of time. There are several ways to derive the equations of motion, the most common being applying Newton's second law of motion. While this method's application is simple and straight-forward for systems with Cartesian coordinate systems and/or low dimensional systems, the complexity of the computations increases with the dimension of the system or if the coordinate system is not perpendicular. The go-to solution for formulating the equations of motion in such cases is the *Euler-Lagrange* formulation based on energy method. Unlike the Newtonian approach, which involves vector quantities, the Lagrangian approach involves scalar quantities such as the kinetic energy and the potential energy and proves to be easier than the Newtonian approach for most systems.

Lagrangian formulation involves describing a rigid body system in terms of *generalized coordinates.* These generalized coordinates can be chosen freely as per the requirement. They can be position coordinates, angles, momentum of particles, or charges. There can even be additional dependent generalized coordinates to help in formulating effective control laws.

A brief procedure of deriving the governing equation of motion through Lagrangian formulation based on the energy method can be given as follows:

1. Choose a set of generalized coordinates $\mathbf{q} \in \mathbb{R}^n$ that completely describes the system configuration.
2. Define a set of generalized forces $\mathbf{u} \in \mathbb{R}^m$.
3. Compute the kinetic energy $T(q, \dot{q})$ and the potential energy, $V(q)$ of the system.
4. Compute the Lagrangian function $L(q, \dot{q})$ defined as the difference between the overall kinetic energy of the system and the potential energy of the system.
$$L = T - V \ , \tag{1.49}$$
5. The equations of motion can now be expressed in terms of the Lagrangian as follows:
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = u \ . \tag{1.50}$$

The dynamical model formulated using the Lagrange formulation method can then be written in the following form:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{u} \ , \tag{1.51}$$

where $\ddot{\mathbf{q}} = [\ddot{q}_1, \ddot{q}_2, \ldots, \ddot{q}_n]^\top$ is the vector of accelerations, $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]^\top$ is the vector of velocities, and $\mathbf{q} = [q_1, q_2, \ldots, q_n]^\top$ is the vector of generalized coordinates. $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis and centripetal force matrix, and $\mathbf{G}(\mathbf{q})$ is the gravitational force matrix. $\mathbf{u} = [u_1, u_2, \ldots, u_m]^\top$ is the vector of generalized forces.

2 Literature Review

In this chapter, a few of the seminal works in the area of application of Koopman operator theory is explored. The following works explore various approaches to obtain the Koopman operator and/or control nonlinear systems through linear Koopman embeddings. Each of these works also demonstrates the application of the derived Koopman operator on various physical systems.

Abraham et al. [9] explore the application of Koopman operator theory to model and control various robotic systems. In particular, the authors explore the utility of the Koopman operator in deriving a dynamical model of the Sphero SPRK solely from data and evaluate open-loop and closed-loop controller performance of the robot on various terrains. The Sphero SPRK is a differential drive mobile robot enclosed in a spherical case. The underlying mechanism and consequently, the controllers are proprietary and therefore is a good example of applying the data-driven modelling and control strategies. The authors present a generalized way of approximating the dynamical system through linearization of approximated Koopman operator. The linearized Koopman operator is then used for model-based control synthesis. Importantly, this linearized Koopman operator propagates forward only the quantities of interest, for example, the states of the system, thereby limiting the order of the approximated dynamical system while preserving its nonlinear interactions. Furthermore, the authors also explore the possibility of augmenting the Koopman operator to model the interactions of a Vertical Take-Off and Landing (VTOL) drone with an unactuated pendulum attached to it. Only the dynamics of the VTOL are known in this case, and the pendulum is swung up and stabilized based on a good choice of observables and the corresponding data collected. The authors evaluate and compare the overall performance of the system for various choices of observables like polynomial basis and Fourier basis functions. Throughout the work, the authors emphasize the effect of the choice of observables, the number of data points collected and their distribution across the state-space on the fidelity of the generated Koopman Operator.

Cisneros et al. [22] explore Koopman operator-based identification techniques to obtain a velocity based quasi-linear parametric varying (qLPV) model of a control moment gyroscope through the application of EDMD algorithm as illustrated in ([9] and [6]). The qLPV model of the CMG is recovered both in the state-space form (where the knowledge of states is assumed to be available) as well as input/output form (where no knowledge of states is assumed). A model predictive controller is then synthesized for these models. Importantly, the authors exploit a priori knowledge about the plant to construct the vector of observables. The authors also note that even in the absence of a priori knowledge of the system dynamics, when it is possible, observables derived from the intuitive knowledge of the system and physical insight perform much better than generalized basis functions like radial basis functions (RBFs) used by Korda et al. Importantly, the approximate Koopman operator here was computed online. This was done by running a short open-loop experiment using chirp test signals to boot-strap the Koopman algorithm and give a meaningful initial model to the MPC controller. The Koopman operator was updated online recursively subject to certain threshold constraints where the Koopman operator was updated only when novel dynamics recognized by significant prediction errors were encountered. Furthermore, the authors compared the closed-loop performances of the model recovered through velocity based qLPV and the one recovered through direct Koopman approach by Korda et al. [8], on the same tracking scenario. The authors noted that the former model which has fewer states (and therefore easier to tune) consequently performed much better in tracking a trajectory than the latter which had many more lifted states and also was highly sensitive to any perturbations in the tuning weights. The higher number of lifted states also meant that the computational complexity of the model recovered through direct Koopman approach was significantly higher than the other. The authors thus conclude that converting high dimensional Koopman model via velocity linearization into a much lower dimensional qLPV system and then using the qLMPC approach proposed by Cisneros et al. in [23], led to superior control performance and numerical efficiency over the methods used by Korda et al. The above claims were corroborated through experimentation on a control moment gyroscope.

Korda et al. [8] extend the definition of Koopman operator to controlled dynamic systems by viewing the controlled dynamical system as an uncontrolled one evolving on an extended state-space given by the product of the original state-space and the space of all control sequences. A modified version of the extended dynamic mode decomposition is then used to compute a finite-dimensional approximation of the Koopman operator with control. The authors test the linear predictors (linear A, B and C matrices) obtained through this approach on a Van der Pol oscillator to predict the evolution of the system dynamics and conclude that these linear predictors exhibited a superior predictive performance as compared to the one obtained by Carleman linearization and local linearization methods. Additionally, the authors also extend the application of these linear predictors to synthesize a model predictive feedback controller. Importantly, the authors show that the computational complexity of the model predictive controllers synthesized based on these linear predictors was comparable to that of linear dynamical systems with the same number of control input and states. This was achieved by using *dense-form* of an MPC problem whose computational complexity is independent of the number of states (or the dimension of the lifted system) and only dependent on the number of control inputs. This was experimentally demonstrated by comparing the performance of feedback control of a bilinear motor in tracking a reference trajectory. Moreover, the comparison is made between an MPC synthesized for a model derived solely from data with no knowledge of the states and an MPC synthesized for a model where the exact dynamics were known and linearized locally.

3 Plant

3.1 Plant Description

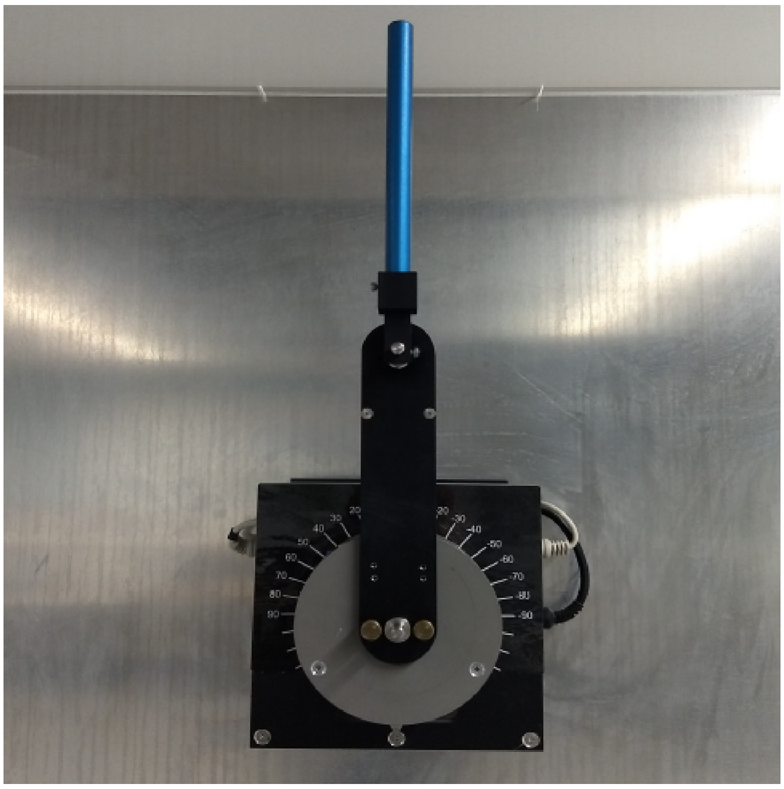


Figure 3.1: ADIP in Up-Up configuration

Dr Y. Nishi originally designed the ADIP for training purposes at Kawasaki Heavy Industry in Japan. Here, the pendulum is the top link and is driven by the rotating arm, which is the bottom link. A DC motor actuates the arm. It is, therefore, a single input multi output system where the angles of the arm, ϕ_1, and the pendulum, ϕ_2, are the outputs measured by incremental rotary encoders. It is worth noting that the dynamics of the ADIP are highly non-linear and chaotic. Additionally, the ADIP is an under-actuated system. This makes it a challenging problem for control. The ADIP has infinitely many fixed points, and of those, an important fixed point is the stable equilibrium of the system described as the 'down-down' configuration, where both the arm and pendulum are in the downward position i. e. $\phi_1 = \phi_2 = \pi$ radians. This system is characterized by only one stable equilibrium point, and the rest of the fixed points constitute unstable equilibria. For the scope of this work, a set of unstable equilibria is described as the 'up-up' configuration where both the arm and pendulum are positioned as $-\pi/2 < \phi_i < \pi/2, i = 1, 2$. The case of an up-up configuration when $\phi_1 = \phi_2 = 0$ is shown in Fig. 3.1.

3.2 First-Principles Model

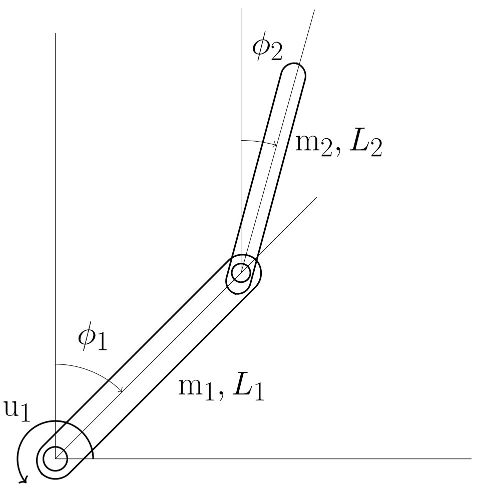


Figure 3.2: A schematic of the ADIP

Consider a schematic ADIP shown in Fig. 3.2 . The arm is of length L_1 and has a mass m_1 which is assumed to be concentrated at a distance l_1, the approximate centre of mass of the arm. Similarly, the pendulum is of length L_2 and has a mass m_2 assumed to be concentrated l_2, the approximate centre of mass of the pendulum. An encoder is placed at the link between the arm and the pendulum to measure the angle made by the pendulum with the vertical. The encoder, along with its mounting on the arm, has a mass m_3 located at L_1.

The states and inputs are defined in a right hand Cartesian coordinate system. Since the ADIP is a planar setup, only the x and y axes are considered. The angular rotation of the arm ϕ_1, and of the pendulum ϕ_2, are measured to the y axis where a clockwise direction is positive. $\phi_1 = \phi_2 = \pi$ when the ADIP is in the stable equilibrium position.

The DC motor applies a torque u_1 to the arm, and the link between the arm and the pendulum is not actuated but free to rotate. As a consequence of the torque in the arm, a disturbance torque u_2 is experienced by the pendulum. A few assumptions are made before deriving the dynamics of the system. These assumptions are referenced from those made for a similar system, the Furuta pendulum, as presented by Cazzolato et al., [24]. These are:

1. The motor shaft and the arm are assumed to be rigidly coupled and infinitely stiff,
2. The pendulum is assumed to be infinitely stiff,
3. Only viscous friction at the joints with damping coefficients C_{arm} and C_{pend} for arm and pendulum respectively are considered. All other forms of damping, if necessary, can be added to the final governing equations of motion appropriately.

The equations of motion for the ADIP are formulated through the Euler-Lagrangian energy-based method. This formulation needs only the change in the kinetic and potential energies of the system.

Since the arm and the pendulum are rigid bodies and have uniformly distributed mass along their length, it is convenient to calculate the total change in kinetic energy, T, as the sum of kinetic energies of the masses m_1, m_2 and m_3.

$$T = \frac{1}{2}J_{arm}\dot{\phi_1}^2 + \frac{1}{2}J_{pend}\dot{\phi_2}^2 + \frac{1}{2}m_2{v_c}^2 + \frac{1}{2}J_{enc}\dot{\phi_1}^2 + \frac{1}{2}J_{motor}\dot{\phi_1}^2 ,$$

where J_{arm} is the moment of inertia of the arm at the point of rotation of the arm, J_{pend} is the moment of inertia of the pendulum about its centre of mass, v_c is the velocity of the centre of mass of the pendulum, J_{enc} is the moment of inertia of mass m_3 at the point of rotation of the arm, and J_{motor} is the rotational inertia of the DC motor. The rotational kinetic energy caused due to the rotational inertia of the DC motor is also added to the total kinetic energy since this value is not negligible and has a considerable effect on the dynamics of the system.

The change in potential energy of the system is similarly calculated as:

$$V = V_1 + V_2 + V_3 ,$$
$$V = m_1 g l_1(1-\cos\phi_1) + m_2 g(L_1(1-\cos\phi_1) + l_2(1-\cos\phi_2)) + m_3 g L_1(1-\cos\phi_1) .$$

The equations of motion for the ADIP can then be computed from Eqs. 1.50 and 1.51 as

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{u} , \tag{3.1}$$

where

$$\mathbf{M} = \begin{bmatrix} \theta_1 & \theta_3 c_{12} \\ \theta_3 c_{12} & \theta_2 \end{bmatrix} , \quad \mathbf{C} = \begin{bmatrix} C_{arm} & \theta_3\dot{\phi}_2 s_{12} \\ -\theta_3\dot{\phi}_1 s_{12} & C_{pend} \end{bmatrix} ,$$

$$\mathbf{G} = \begin{bmatrix} -\theta_4 g\sin(\phi_1) \\ -\theta_5 g\sin(\phi_2) \end{bmatrix} ,$$

with

$$\theta_1 = J_{arm} + J_{motor} + J_{enc} + m_1 l_1^2 + m_2 L_1^2 ,$$
$$\theta_2 = m_2 l_2^2 + J_{pend} ,$$
$$\theta_3 = m_2 L_1 l_2 ,$$
$$\theta_4 = m_1 l_1 + m_2 L_1 + m_3 L_1 ,$$
$$\theta_5 = m_2 l_2 ,$$
$$s_{12} = \sin(\phi_1 - \phi_2) \quad \text{and} \quad c_{12} = \cos(\phi_1 - \phi_2).$$

For the ADIP, $\mathbf{q} = [\phi_1 \ \ \phi_2]^\top$, $\mathbf{u} = [u_1 \ \ 0]^\top$. The physical parameters of the ADIP and their values are listed in Table 1. The masses and the geometrical entities were measured physically when the plant was disassembled. The motor parameters were taken from the motor manual, which can also be found in the appendix of this report. The friction is considered to be viscous and is estimated through the system identification tool-box in Matlab by comparing the simulated and experimental data.

Parameter	Description	Value
L_1	Length of arm	$0.154\ m$
L_2	Length of pendulum	$0.186\ m$
l_1	Distance to C.G of the arm	$0.077\ m$
l_2	Distance to C.G of the pendulum	$0.093\ m$
m_1	Mass of arm	$0.13\ kg$
m_2	Mass of pendulum	$0.07\ kg$
m_3	Mass of encoder assembly	$0.04\ kg$
J_{arm}	Moment of Inertia of arm	$3.00 \times 10^{-4}\ kg{\cdot}m^2$
J_{pend}	Moment of Inertia of pendulum	$2.02 \times 10^{-4}\ kg{\cdot}m^2$
J_{motor}	Moment of Inertia of motor	$1.58 \times 10^{-4}\ kg{\cdot}m^2$
J_{enc}	Moment of Inertia of encoder assembly	$7.37 \times 10^{-4}\ kg{\cdot}m^2$
C_{arm}	Viscous damping co-efficient of arm	$2.89 \times 10^{-3}\ N{\cdot}s/m$
C_{pend}	Viscous damping co-efficient of pendulum	$1.41 \times 10^{-5}\ N{\cdot}s/m$
g	Acceleration due to gravity	$9.81\ m/s^2$

Table 1: ADIP parameters

3.3 Data-Driven Model

A data-driven model is derived solely from available measurement data with almost no assumptions made about the plant. However, one can always augment the obtained measurements by *lifting* the measurements through some linear or nonlinear functions of the same. This enables one to predict the behaviour of a model over longer time spans and deduce a linear operator which governs the evolution of these measurements. This linear operator is obtained by applying the previously mentioned methods like DMD, EDMD or SINDy. This section elaborates on the observables that are relevant for this thesis work. As previously discussed in Chapter 1, the choice of observables is crucial to estimate the Koopman operator, but discovering the 'correct' observables is still an ongoing topic of research. Currently, there are several approaches to approximate the observables, for example, exploiting apriori knowledge of system dynamics [12], physical intuition of the system dynamics [22], applying popular function approximators like radial basis functions [20] or polynomial basis and Fourier basis functions [9], estimating the dominant candidates in a dynamic equation through sparse regression as done in SINDy [13] etc.

During the course of this thesis work, all of the above-stated options were explored. While some worked well for prediction, the same could not be applied for control purposes. The reason why this could not be done is left to further research. However, it was observed that SINDy worked well in predicting the dynamics of the system, and this could be used as a first step in identifying reliable observables, at least for prediction, if not control. It was also observed that the measured state of the system predicted the dynamics of the system very well in the vicinity of the equilibrium and as the system moved farther away, the addition of some nonlinear states helped in better prediction of the system states.

This is better explained in the Chapter 5 where the effect of the choice of observables is evaluated in open-loop and closed-loop system identification.

In this thesis, a combination of states and nonlinear functions form the observables vector. Again, the choice of observables vary with the purpose. Many different combinations of observables were tried and for the purpose of the Chapter 5, four different vectors of observables are defined as follows

$$\boldsymbol{\Psi}_1 = [\phi_1 \quad \phi_2 \quad \dot{\phi}_1 \quad \dot{\phi}_2 \quad u]^\top \in \mathbb{R}^5 \ , \tag{3.2}$$

Thus, $\boldsymbol{\Psi}_1$ is just the state variables $\boldsymbol{\phi}^\top$ augmented with the input variable.

The state variables is augmented by a vector of polynomials up to second degree of the states which form the second observable vector,

$$\boldsymbol{\Psi}_2 = [\boldsymbol{\phi}^\top \quad \psi_1 \quad \psi_2 \quad \dots \quad \psi_{10} \quad u] \in \mathbb{R}^{15} \ , \tag{3.3}$$

$$\psi_i = \phi_1^{\alpha_i} \phi_2^{\beta_i} \dot{\phi}_1^{\gamma_i} \dot{\phi}_2^{\delta_i} \ , \tag{3.4}$$

where $\alpha_i, \beta_i, \gamma_i, \delta_i$ are non-negative numbers and index i tabulates all the combinations such that $\alpha_i + \beta_i + \gamma_i + \delta_i \leq Q$ and $Q > 1$ defines the largest allowed polynomial degree. In the above case, $Q = 2$.

The third observables vector extends $\boldsymbol{\Psi}_2$ by including a combination of polynomial and sine functions of the state variables,

$$\boldsymbol{\Psi}_3 = [\boldsymbol{\phi}^\top \quad \boldsymbol{\psi}^\top \quad \phi_1 s_1 \quad \phi_1 s_2 \quad \phi_1 c_1 \quad \phi_1 c_2 \quad \phi_2 s_1 \quad \dots \quad \dot{\phi}_2 c_2 \quad u] \in \mathbb{R}^{31} \ , \tag{3.5}$$

where $s_1 = \sin \phi_1$, $c_1 = \cos \phi_1$ and similarly for others. Note that only the first two states were considered for the sine and cosine functions. Including the derivatives did not improve the prediction significantly and hence were discarded.

Finally, a combination of state variables and radial basis functions was also used. A *radial function* is a real-valued function κ, that is radially symmetric around some point called the function's centre. This can be either the origin so that, $\kappa(\mathbf{x}) = \kappa(\|\mathbf{x}\|)$, or in general, some other point $\mathbf{c}$ so that $\kappa(\mathbf{x}, \mathbf{c}) = \kappa(r)$, where $r = \|\mathbf{x} - \mathbf{c}\|_2$ is the Euclidean distance between the points $\mathbf{x}$ and $\mathbf{c}$. Radial functions are typically used as a basis (and hence the name) to construct an approximation $\tilde{f}$, of a function of the form $y_i = f(\mathbf{x}_i)$ such that,

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{w}_i \kappa(\|\mathbf{x} - \mathbf{c}_i\|) \ , \tag{3.6}$$

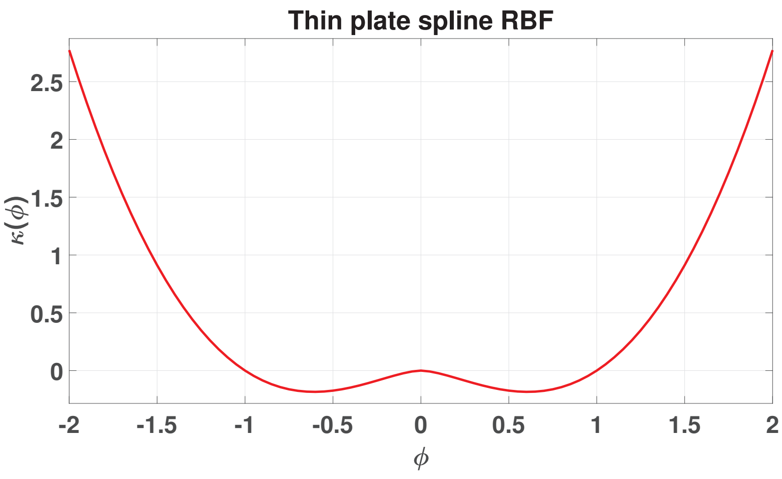
where the approximating function $\tilde{f}$ is represented as a sum of N radial basis functions (RBFs), each associated with a different centre $\mathbf{c}_i$, and weighted by an approximate coefficient $\boldsymbol{w}_i$. The weights $\boldsymbol{w}_i$ are determined by ensuring that the approximation will exactly match the given data at data points i.e., $\tilde{f}(\mathbf{x}_i) = y_i$. This is accomplished by using the matrix methods of linear least squares, as the approximating function is linear in the weights $\boldsymbol{w}_i$. A detailed treatment of RBFs can be found in [25]. There are several different types of radial basis functions, for example, gaussian RBFs, multi quadratic RBFs, polyharmonic RBFs, thin-plate splines etc. Thin plate spline RBFs usually work very well for function approximations [20] and are independent of any manual tuning parameters except for the choice of function centres. In this thesis work, therefore, 50 thin-plate spline RBFs are used along with the state vector to form the fourth observable vector,

$$\mathbf{\Psi}_4 = [\boldsymbol{\phi}^\top \quad \kappa_1 \quad \kappa_2 \quad \dots \quad \kappa_{50} \quad u] \in \mathbb{R}^{55} \,, \tag{3.7}$$

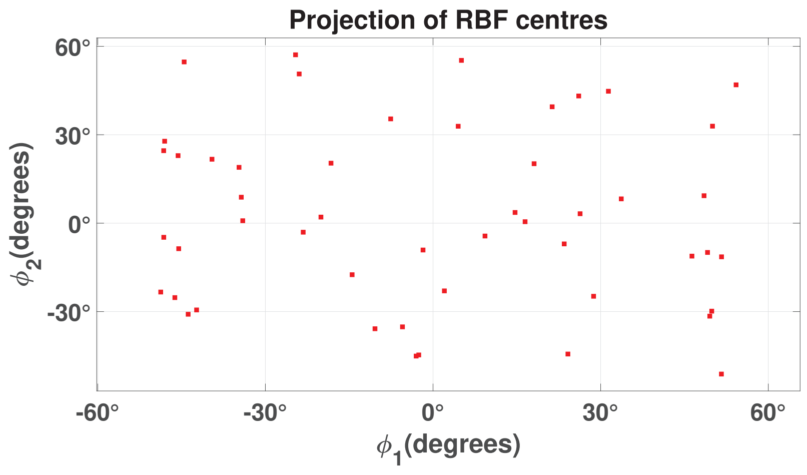
where

$$\kappa(\boldsymbol{\phi}) = \|\boldsymbol{\phi} - c\|_2^2 \log\left(\|\boldsymbol{\phi} - c\|_2\right) \,. \tag{3.8}$$

In this thesis, the centres are chosen randomly within a set defined by the operating range of the system. Figure 3.3a shows a selection of such centres and Figure 3.3b shows a graph of the thin plate spline RBF centred at the origin ($c = 0$). It is clearly seen that the value of κ increases with the distance of the function from the centre. Therefore, it can be interpreted as if the RBF will be inactive when the centres are chosen near or at the measured data points, which is what is required since the observables vector also contains the measured state as observables.



(a) Thin plate spline RBF



(b) RBF centres projected onto a plane for $\mathbf{\Psi_4}$

Figure 3.3: Thin plate splines RBF and the centres

4 Controller Design

This section discusses the theoretical background of controllers relevant for this thesis work in brief. An overview of related topics such as energy-based swing-up control, the linear quadratic regulator (LQR) and model predictive control (MPC) is presented.

4.1 Swing-up Control

The swing-up control problem for the ADIP is to swing the system (the arm and the pendulum) up from its stable equilibrium at $\phi_1 = \phi_2 = \pi$ rad., and balance it about its unstable equilibrium at $\phi_1 = \phi_2 = 0$. This is achieved by swinging up the system from its stable equilibrium position through a 'swing-up' controller and bringing it into a *basin of attraction* in the up-up configuration where the swing-up controller is switched to a locally convergent stabilizing controller, usually an LQR, that will balance the system about its up-up position. Swing-up control for inverted pendulum systems, in general, has been traditionally achieved through feedback linearization techniques [26], or energy-based control [24]. Though the feedback linearization technique has almost always worked perfectly well in solving the swing-up control problem, it does not come with any stability analysis or performance guarantees.

Fantoni et al. [27] solve this issue by presenting a control strategy based on energy approach and passivity properties of the pendubot system (which is the same as ADIP). This control strategy ensures that the state is brought either arbitrarily close to the up-up position or to a homoclinic orbit that will eventually enter the basin of attraction of any locally convergent controller. Stability analysis for the same is presented based on LaSalle's theorem. The main result of the work by the authors is presented as a theorem in [27]. The swing-up control law is given as:

$$u = \frac{-k_D F - t_M(\dot{\phi}_1 + k_P \phi_1)}{t_M k_E \tilde{E} + k_D \theta_2} \,, \tag{4.1}$$

with

$$\begin{aligned} F &= \theta_2\theta_3 \sin(\phi_1 - \phi_2)\dot{\phi}_2^2 + \theta_3^2 \cos(\phi_1 - \phi_2)\sin(\phi_1 - \phi_2)\dot{\phi_1}^2 \\ &\quad + \theta_2\theta_4 g \sin\phi_1 + \theta_3\theta_5 g \sin(\phi_1 - 2\phi_2) + C_{arm}\theta_2\dot{\phi}_1 + C_{pend}\theta_3 \cos(\phi_1 - \phi_2)\dot{\phi}_2 \,, \\ t_M &= (\theta_1\theta_2 - \theta_3^2 \cos^2(\phi_1 - \phi_2)) \,, \\ \tilde{E} &= E - E_{top} \end{aligned}$$

where $k_E, k_D,$ and k_P are strictly positive constants and $\tilde{E}$ is the difference in the actual total energy of the system and total energy at the up-up position. The $\theta_i,\ i = \{1, 2, \ldots, 5\}$ are given in Chapter 3.
Furthermore, Xin et al. [28] propose a sufficient condition for the parameters: $k_E, k_D,$ and k_P in the control law such that the total energy of the Pendubot will converge to the potential energy of its top upright position. The initial parameters were obtained through the method proposed by the authors and thereafter, a GA was used to fine tune the parameters.

4.2 Linear Quadratic Regulator

The overarching goal of this thesis is to reformulate strongly nonlinear dynamics in a linear framework to enable the use of powerful optimal and robust control techniques available for linear systems. Consider the nonlinear system affected by an external input:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}) \ , \quad \mathbf{x}(0) = \mathbf{x}_0 \ , \tag{4.2}$$

with multi-channel control input $\mathbf{u} \in \mathbb{R}^q$ and continuously differentiable dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Without loss of generality, the origin is an equillibrium: $\mathbf{f}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$. Inifinte-horizon optimal control then minimizes the following quadratic cost functional

$$J(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\int_0^\infty \mathbf{x}^\top(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\top\mathbf{R}\mathbf{u}(t) \ dt \tag{4.3}$$

with state and input weight matrices $\mathbf{Q} \in \mathbf{R}^{n \times n}$ and $\mathbb{R}^{m \times m}$. Both matrices are symmetric and fulfill $\mathbf{Q} > 0$ and $\mathbf{R} \geq 0$. A full-state feedback control law

$$\mathbf{u}(\mathbf{x}) = -\mathbf{F}(\mathbf{x})\mathbf{x} \tag{4.4}$$

with gain $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^{m \times n}$ is sought to minimize the cost function in Eq. 4.3 subject to the state dynamics in Eq. 4.2 to drive the system to the origin, i.e.

$$\lim_{x \to \infty} \mathbf{x}(t) = \mathbf{0} \quad \forall \ \mathbf{x}.$$

The above control problem simplifies considerably for linear systems of the form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{4.5}$$

Using $\mathbf{z} = \mathbf{x}^\top\mathbf{P}\mathbf{x}$ for the co-state, the optimal control is given by

$$\mathbf{u} = -\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{z} = -\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P}\mathbf{x} \ , \tag{4.6}$$

with constant gain $\mathbf{F} = -\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P}$ and where the semi-definte matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ is the solution to the algebraic Riccati equation (ARE):

$$\mathbf{Q} + \mathbf{P}\mathbf{A} + \mathbf{A}^\top\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P} = \mathbf{0} \ . \tag{4.7}$$

The above formulation is referred to as the linear quadratic regulator for which the solution of ARE yields a global state-feedback control law. This simplicity motivates efforts to find linear representations for nonlinear systems and explains why nonlinear embeddings via Koopman operator theory are so appealing [29].

4.3 Model Predictive Control

Model predictive control (MPC) is an advanced method of feedback control that is used to control a system while satisfying a set of constraints on the input and/or the state. It offers several advantages over the infinite horizon LQR; to list a few, the input is computed over a receding finite time horizon, i.e., at each time step, an optimal control input vector is computed by minimizing a user-specified cost function (e.g., energy or tracking error) over a predefined time horizon N_p, and only the first value of this input vector is applied to progress the system to the next time step where the entire process is repeated, thereby enabling optimal tracking of the reference trajectory. Although this appears as a computationally expensive process, which it is when compared to the LQR, it performs a whole lot better in the event of any external disturbances or sudden changes in reference trajectory. This is especially advantageous in tracking problems where the controller is able to 'predict' any setpoint changes and react accordingly. Consequently, this also means that the controller can predict and provide early warnings of any potential problems. Also, any constraints on input and output can be imposed in a simple way in this technique.

For this thesis, the state-space form of MPC is considered. There is abundant literature available describing the construction and implementation of the MPC; the reader is encouraged to refer [30] for a detailed explanation of the MPC. This section will therefore only briefly explain the cost function used in this thesis and an approach explored in Korda et al. [8] to eliminate the dependence of the cost function on the dimension of state of the system which will eventually prove useful for formulating the MPC for high-dimensional systems.

The following formulations are an adaptation of the formulations presented in [8].
Assume a nonlinear system of the form $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ can be approximated in terms of a linear model of the form,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \ , \tag{4.8}$$

The model predictive controller solves at each time step k of the closed-loop iteration the optimization problem,

$$\begin{array}{lll} \min\limits_{\mathbf{u}_i,\mathbf{x}_i} & J\left((\mathbf{u}_i)_{i=0}^{N_p-1}, (\mathbf{x}_i)_{i=0}^{N_p-1}\right) & \\ \text{subject to} & \mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i \ , & i = 0, 1, \ldots, N_p - 1 \\ & \mathbf{E}_i\mathbf{x}_i + \mathbf{F}_i\mathbf{u}_i \leq \mathbf{b}_i \ , & i = 0, 1, \ldots, N_p - 1 \\ & \mathbf{E}_{N_p}\mathbf{x}_{N_p} \leq \mathbf{b}_{N_p} \ , & \\ \text{parameter} & \mathbf{x}_0 = \mathbf{x}_k \ , & \end{array} \tag{4.9}$$

where the matrices $\mathbf{E}_i \in \mathbb{R}^{n_c \times n}$, $\mathbf{F}_i \in \mathbb{R}^{n_c \times m}$ define the state and input polyhedral constraints, respectively and $\mathbf{b}_i \in \mathbb{R}^{n_c}$ is the vector of constraints, and the convex quadratic cost function J is given by,

$$J = \mathbf{x}_{N_p}^\top \mathbf{Q}_{N_p} \mathbf{x}_{N_p} + \sum_{i=0}^{N_p-1} \mathbf{x}_i^\top \mathbf{Q} \mathbf{x}_i + \mathbf{u}_i^\top \mathbf{R} \mathbf{u}_i \ , \tag{4.10}$$

where $\mathbf{x}_{N_p}$ is the final state at time instance N_p, $\mathbf{Q}_{N_p} \in \mathbb{R}^{n\times n}$ is the corresponding penalty on the final state.

The optimization problem (4.9) parametrized by the current state of the nonlinear dynamical system $\mathbf{x}_k$ then defines a feedback controller $F(\mathbf{x}) = \mathbf{u_0}^*(\mathbf{x}_k)$, where $\mathbf{u_0}^*(\mathbf{x}_k)$ is the optimal solution to the optimization problem.

It is always beneficial to eliminate the dependence of the optimization problem on the dimension of the state n. Consequently, the computational complexity of solving the optimization problem will be rendered independent of the dimension n. This is especially useful when dealing with the approximation of nonlinear systems through the Koopman operator theory, where the state might be *lifted* to a higher dimension which results in a linear dynamical system that approximates the nonlinear dynamical system. This elimination can be achieved by formulating the dynamics in the following way: consider the discrete-time or the *one step-ahead predictor* state-space model

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \qquad \mathbf{x}(0) = \mathbf{x}_0 \ , \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k \ , \end{aligned} \tag{4.11}$$

where $\mathbf{x}_{k+1} \in \mathbb{R}^n$ is the 'future' value of the current state $\mathbf{x}_k$ at a time instance $k = [0, 1, 2, \ldots, N_p - 1]$, $\mathbf{u}_k \in \mathbb{R}^m$ is the input at time instance k and $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{n\times m}$ and $\mathbf{C} \in \mathbb{R}^{l\times n}$ are the system, input and output matrices, respectively. Naturally, the state at time instance $k+2$ can then be predicted from Eq. 4.11 as

$$\mathbf{x}_{k+2} = \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{u}_{k+1} \ , \tag{4.12}$$

Subsituting Eq. 4.11 in Eq. 4.12, the future state at the second time instance can be obtained in the terms of the initial state $\mathbf{x}_k$ as

$$\mathbf{x}_{k+2} = \mathbf{A}^2\mathbf{x}_k + \mathbf{A}\mathbf{B}\mathbf{u}_k + \mathbf{B}\mathbf{u}_{k+1} \ . \tag{4.13}$$

The one-step ahead predictor model therefore, can be used recursively to predict the evolution of state for the next N_p steps of interest where N_p is the time interval over which a cost function is evaluated to compute the optimal input corresponding to the desired state evolution in that time interval.

$$\underbrace{\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{k+2} \\ \vdots \\ \mathbf{x}_{k+N_p} \end{bmatrix}}_{\mathbf{X}_{k+1}} = \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N_p} \end{bmatrix}}_{\bar{\mathbf{A}}} \mathbf{x}_k + \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{A}\mathbf{B} & \mathbf{B} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ldots & \vdots \\ \mathbf{A}^{N_p-1}\mathbf{B} & \mathbf{A}^{N_p-2}\mathbf{B} & \ldots & \mathbf{B} \end{bmatrix}}_{\bar{\mathbf{B}}} \underbrace{\begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_{k+1} \\ \vdots \\ \mathbf{u}_{k+N_p-1} \end{bmatrix}}_{\mathbf{U}_k} . \tag{4.14}$$

Eq. 4.14 can be compactly written as,

$$\mathbf{X}_{k+1} = \bar{\mathbf{A}}\mathbf{x}_k + \bar{\mathbf{B}}\mathbf{U}_k \ , \tag{4.15}$$

where $\mathbf{X}_{k+1} \in \mathbb{R}^{N_p n}$ is the vector of state predictions, $\bar{\mathbf{A}} \in \mathbb{R}^{N_p n\times n}$, $\bar{\mathbf{B}} \in \mathbb{R}^{N_p n\times N_p n}$ is a Toeplitz matrix (or a diagonal-constant matrix) and $\mathbf{U}_k \in \mathbb{R}^{N_p m}$ is the vector of predicted sequence of control inputs. Note that Eq. 4.15 depends only on the initial state $\mathbf{x}_0$ and the sequence of control inputs $\mathbf{U}_k$.

Consequently, the output equation is

$$\mathbf{Y}_{k+1} = \bar{\mathbf{C}}\bar{\mathbf{A}}\mathbf{x}_k + \bar{\mathbf{C}}\bar{\mathbf{B}}\mathbf{U}_k \ , \tag{4.16}$$

where $\mathbf{Y}_{k+1} \in \mathbb{R}^{N_p l}$ is the vector of predicted future outputs and $\bar{\mathbf{C}} \in \mathbb{R}^{N_p l \times N_p n}$ is the computed as the Kronecker product $\mathbf{I}_{N_p} \otimes \mathbf{C}$. Note that the notations used in this section, $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}$ correspond to the notations $\mathbf{A_b}, \mathbf{B_b}, \mathbf{C_b}$ used in the Matlab script.

It is then possible to transform the optimization problem (4.9) to the so-called *dense form*

$$\begin{aligned} \min_{\mathbf{U}\in\mathbb{R}^{mN_p}} \quad & \mathbf{U}^\top\mathbf{H}\mathbf{U} + \mathbf{x}_0^\top\mathbf{G}\mathbf{U} \\ \text{subject to} \quad & \mathbf{L}\mathbf{U} + \mathbf{M}\mathbf{x}_0 \le \mathbf{c} \\ \text{parameter} \quad & \mathbf{x}_0 = \mathbf{x}_k \end{aligned} \tag{4.17}$$

for some positive-definite matrix, the *Hessian* $\mathbf{H} \in \mathbb{R}^{mN_p \times mN_p}$ and some matrices, $\mathbf{G} \in \mathbb{R}^{n \times mN_p}$, $\mathbf{L} \in \mathbb{R}^{n_c N_p \times mN_p}$, $\mathbf{M} \in \mathbb{R}^{n_c N_p \times n}$ and vector of constraints $\mathbf{c} \in \mathbb{R}^{n_c N_p}$. The matrices are defined as

$$\begin{aligned} \mathbf{H} &= \bar{\mathbf{R}} + \bar{\mathbf{B}}^\top\bar{\mathbf{Q}}\bar{\mathbf{B}} \ , \quad & \mathbf{G} &= 2\bar{\mathbf{A}}^\top\bar{\mathbf{Q}}\bar{\mathbf{B}} \ , \\ \mathbf{L} &= \bar{\mathbf{F}} + \bar{\mathbf{E}}\bar{\mathbf{B}} \ , & \mathbf{M} &= \bar{\mathbf{E}}\bar{\mathbf{A}} \ , \qquad \mathbf{c} = [\mathbf{b}_0^\top, \ldots, \mathbf{b}_{N_p}^\top]^\top \end{aligned} \tag{4.18}$$

where $\bar{\mathbf{Q}} = \mathbf{I}_{N_p} \otimes \mathbf{Q} \ \in \mathbb{R}^{nN_p \times nN_p}$, $\bar{\mathbf{R}} = \mathbf{I}_{N_p} \otimes \mathbf{R} \ \in \mathbb{R}^{N_p m \times N_p m}$, $\mathbf{E} = \mathbf{I}_{N_p} \otimes \mathbf{E} \ \in \mathbb{R}^{n_c N_p \times n_c N_p}$, and

$$\bar{\mathbf{F}} = \begin{bmatrix} F_0 & 0 & \ldots & 0 \\ 0 & F_1 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & F_{N_p-1} \\ 0 & 0 & \ldots & 0 \end{bmatrix}$$

Note that (4.17) is formulated in a purely quadratic form and one can use any of the available QP solvers available. The optimization is done over the predicted control inputs $\mathbf{U} = [\mathbf{u}_0^\top, \mathbf{u}_1^\top, \ldots, \mathbf{u}_{N_p-1}^\top]^\top$. Importantly, note that the size of the $\mathbf{H}$ and that of the vector of constraints $\mathbf{c}$ is independent of the dimension of the state. This formulation, when extended to lifted systems with a lifted dimension K of the state, proves very useful as the above said matrices are independent of K and the matrices can, in fact, be precomputed offline before deploying the controller. One needs only to evaluate the nonlinear mapping or the lifting of state at each time step, thereby making the cost of computation inexpensive. In summary, once the data matrices in (4.17) are formed, the cost of solving the optimization problem is independent of the dimension of the state space and therefore, a nonlinear MPC problem can be solved as a linear MPC problem provided linear predictors of the form (4.8) can be estimated using the Koopman operator theory. Section 3.3 deals with such *lifting* variables that can approximate nonlinear systems.

5 Simulation and Results

This chapter summarises the observations, implementations, and consequent results of Koopman operator theory's application on the Arm-Driven Inverted Pendulum. Since it is infeasible to work with infinite-dimensional models, even though they are linear, the infinite-dimensional Koopman operator is projected onto a finite-dimensional space where the finite-dimensional (approximate) Koopman operator is a linear operator approximated by data-driven regression methods such as Dynamic Mode Decomposition. The various data-driven regression methods and their respective applications have been discussed in the previous sections.

The essence of data-driven regression is naturally the measured data, and care must be taken on the choices made to record this data; for example, whether the data must be recorded with or without the influence of a forcing function, whether the identification should be made in open-loop or closed-loop, whether the data should be captured from a single trajectory or multiple trajectories etc. This chapter aims to answer such questions by investigating the results of various choices.

The choice of system identification (open-loop or closed-loop) to collect the necessary measurement data is first investigated in Section 5.1. Since the final goal is tracking a reference trajectory, it is important to identify the system dynamics in the up-up configuration. However, this poses a serious challenge because firstly, the equilibrium of the up-up configuration is unstable and to add to that, the dynamics of the ADIP are chaotic, i.e., the system can evolve along drastically different trajectories for a small variation in the initial conditions, and it is very challenging (and maybe not even possible) to estimate the dynamics of ADIP in open-loop about the up-up configuration. Therefore, it is necessary to identify the system in closed-loop. This too proves problematic in the context of implementation as one needs to have a stabilizing controller already. Without knowledge of dynamics, it is challenging to synthesize such a controller. Nevertheless, assuming that a non-model based controller can be synthesized for an unstable but controllable nonlinear dynamical system, it is possible to estimate the system's closed-loop dynamics.

In this thesis work, however, since the ADIP is a relatively simple system when compared to systems with complex dynamical interactions of the state (which can also be high-dimensional to add to the complexity), a mathematical model was derived as presented in Chapter 3. A model-based feedback controller, the LQR was subsequently designed for the ADIP, and closed-loop reference tracking was executed to collect the required measurement data. An important observation is in order concerning finding a stabilizing controller in the absence of a mathematical model, as discussed previously. The data-driven controller's ultimate goal is to track a reference trajectory nearly as good as the model-based controller and if possible, extend the range of tracking. However, a data-driven model is only as good as the measured data, i.e. it is most probable to work in the measured data range. To this end, it was of interest to analyze an iterative approach to derive the best possible data-driven controller with data captured from a limited range of trajectory. It was observed that data recorded for a small range of motion was enough to capture the system's dominant closed-loop dynamics. Therefore, the model derived from such data can be iterated to obtain a reasonably close approximation of the model derived from data recorded over a larger range. This particular method of deriving a data-driven approximation could be particularly useful in designing data-driven controllers for highly unstable systems for which no prior knowledge of dynamics exist and can only be controlled in a small region around the unstable equilibrium. This is further explained at the end of this chapter.

It is important to note that although the region of interest for this thesis is the up-up configuration for which it is necessary to perform closed-loop system identification, open-loop identification for the ADIP around the stable equilibrium point provides many useful and important insights into the methods of data collection and the choice of observables for prediction and control. The knowledge derived from open-loop system identification is applied to measurement data and choice of observables for the closed-loop system. This has been elaborately discussed in Section 5.1. Furthermore, the choice of identification signal also varies from open-loop identification to closed-loop identification. In open-loop identification, a chirp signal was used to excite the plant around the stable equilibrium. In closed-loop identification, the input signal was simply the control input generated by a suitable controller corrupted by some white noise. White noise is added to the control input to differentiate the plant dynamics from input dynamics. One can also kick the closed-loop system occasionally with an impulse input for this purpose.

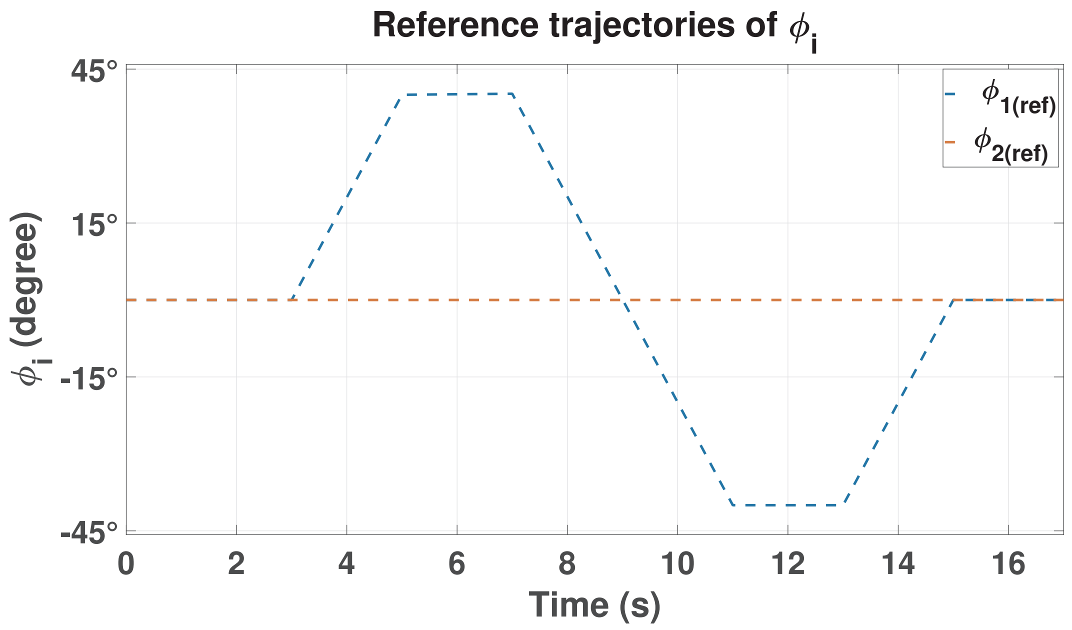
Section 5.2 presents some comparisons of controllers such as the LQR with integral action (LQi) and a model predictive controller (MPC) applied to a *local* model versus a data-driven model. The *local* model is simply a linear approximation of the nonlinear dynamics of the ADIP locally linearized at the Up-Up position ($\phi_1 = \phi_2 = 0°$). The effect of the choice of observables is investigated in this section. Furthermore, the data-driven model's performance is compared with a quasi-linear parameter varying model of the ADIP.

Finally, Section 5.3 compares an energy-based swing up approach as presented in [27] with a linear model predictive control approach where the nonlinear ADIP model is approximated by a quasi-linear parameter varying model as presented in [31]. Although the authors here use the idea to evaluate a reference tracking objective with a qLPV, it was of interest to investigate if this approach could also be applied for linear model based swing-up of the inverted pendulum.

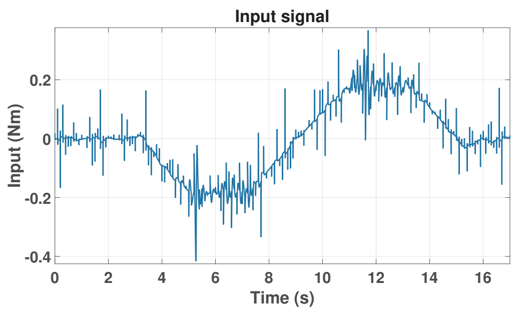
5.1 System identification and identification signal

As discussed, the desired area of interest for this thesis is the up-up configuration of the ADIP. To this end, a closed-loop feedback controller is designed for the local model, which tracks a reference trajectory. The tracking data, which comprises state and input measurements, is then used as the raw data to approximate a data-driven controller. Additionally, Brunton et al. [21], and Kaiser et al. [29] observe that during closed-loop identification since the feedback is a function of the state, there is a need to add sufficiently large white noise or kick the system occasionally with large impulse or step inputs to distinguish the feedback dynamics of the input from the state dynamics. Note that this step is not necessary for open-loop system identification, which will be discussed later.

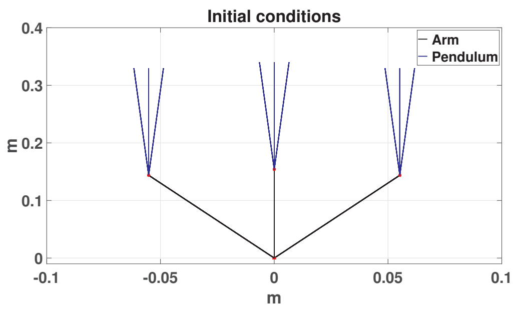
In this thesis work, a reference trajectory, as shown in Figure 5.1a was chosen for which an LQ regulator was designed. The resulting input signal is shown in Figure 5.1b. This input signal is the feedback signal perturbed by a white noise of noise power 0.05.



(a) Reference trajectory of the states (ϕ_1 and ϕ_2).



(b) Input signal



(c) Visualization of the initial conditions

Figure 5.1: Reference trajectory, Input signal and Initial conditions

Data was collected by simulating the system multiple times in closed-loop with random initial conditions given by

$$\mathbf{x}_0 = [\mathcal{U}(-1 \quad 1) l_{\phi_1},\ \mathcal{U}(-1 \quad 1) l_{\phi_2},\ \mathcal{U}(-1 \quad 1) l_{\dot{\phi}_1},\ \mathcal{U}(-1 \quad 1) l_{\dot{\phi}_2}] \ , \tag{5.1}$$

where $\mathcal{U}(-1,\ 1)$ is a uniformly distributed random variable with range -1 to 1, and $l_{\phi_1} = 40°$, $l_{\phi_2} = 2°$, $l_{\dot{\phi}_1} = l_{\dot{\phi}_2} = 0.25\text{rad/s}$. Therefore, the initial condition is uniformly distributed around the unstable equilibrium ($\phi_1 = \phi_2 = \dot{\phi}_1 = \dot{\phi}_2 = 0°$) and its range is defined by $L = [l_{\phi_1}, l_{\phi_2}, l_{\dot{\phi}_1}, l_{\dot{\phi}_2}]$. Figure 5.1c visualizes the distribution of initial conditions with respect to the states ϕ_1 and ϕ_2. L was chosen by trial-and-error and it represents the range limit in which the controller is able to track the subsequent reference trajectory successfully. One might also choose L by evaluating a regulation task instead of tracking. The data was generated by simulations of 17s each on a nonlinear model of the ADIP (refer Chapter 3) where the controller tracked the reference trajectory (Fig. 5.1a). The data was sampled every 5ms. The subsequent approximation of the linear matrices follows the procedure detailed in Section 1.3.

As stated previously, although the region of interest for this thesis work is the up-up configuration for which only closed-loop identification is possible, open-loop identification around the stable equilibrium nevertheless presents numerous important insights into the generation of data and choice of observables, which was then applied to generate data and/or influenced the choice of observables in closed-loop identification. During the course of this thesis, numerous combinations of observables and data were tried and analyzed, and this could not have been easily done in the up-up configuration as it demands a corresponding controller to stabilize the system.

In this thesis, SINDy with control (SINDYc) algorithm (refer Section 1.4) is used to approximate the open-loop dynamics to demonstrate the working of SINDy algorithm. EDMDc can also very well be used for the same. However, SINDYc approximates the states' time evolution directly from data and a set of candidate functions without having to go through the hassle of choosing the 'correct' observables and computing the corresponding Koopman operator. It is important to note that SINDYc has also been used for feedback control of nonlinear systems [21]; however, this has not been implemented in this work.

The forcing function used in open-loop identification is a chirp signal with the frequency defined in the range $[f_{min} \ f_{max}]$ and with amplitude A. The frequencies and amplitudes were chosen such that the system does not translate to the 'chaotic zone' but is also sufficiently perturbed away from its stable equilibrium. The chirp signal can be generated using the *chirp* command in Matlab. In summary, the forcing functions for training and validation purposes are given as,

$$\begin{aligned} \text{Training} &\rightarrow A * chirp(t, f_{min}, t_s, f_{max}, [\ ], -90), \quad [f_{min} \ f_{max}] = [0.3 \ 0.7]\text{Hz}, A = 0.1 \ , \\ \text{Validation} &\rightarrow A * chirp(t, f_{min}, t_s, f_{max}, [\ ], -90), \quad [f_{min} \ f_{max}] = [0.7 \ 0.9]\text{Hz}, A = 0.065 \ . \end{aligned} \tag{5.2}$$

First, the case for data obtained from multiple trajectories is made. As Abraham et al. [9] observed, the number of data points and their distribution across the state space will have a large effect on the computed Koopman operator of the underlying nonlinear system. It is always beneficial to record trajectories from multiple initial conditions as shown in (5.1).

Figures 5.2 and 5.3 compare the identification of the system through SINDYc from training data generated by a single initial condition and training data generated from multiple initial conditions respectively. The training data generated from multiple initial conditions is visualized in Figure 5.4. In this case, the training data is generated from multiple initial conditions (5.1) with $l_{\phi_1} = 210°$, $l_{\phi_2} = 150°$, $l_{\dot{\phi}_1} = l_{\dot{\phi}_2} = 1\text{rad/s}$ and the forcing function (5.2) with $f_{min} = f_{max} = 0.5\text{Hz}$ and $A = 0.1$, respectively. Again, the choice of these values are influenced by the requirement for trajectories to lie in the non-chaotic zone.

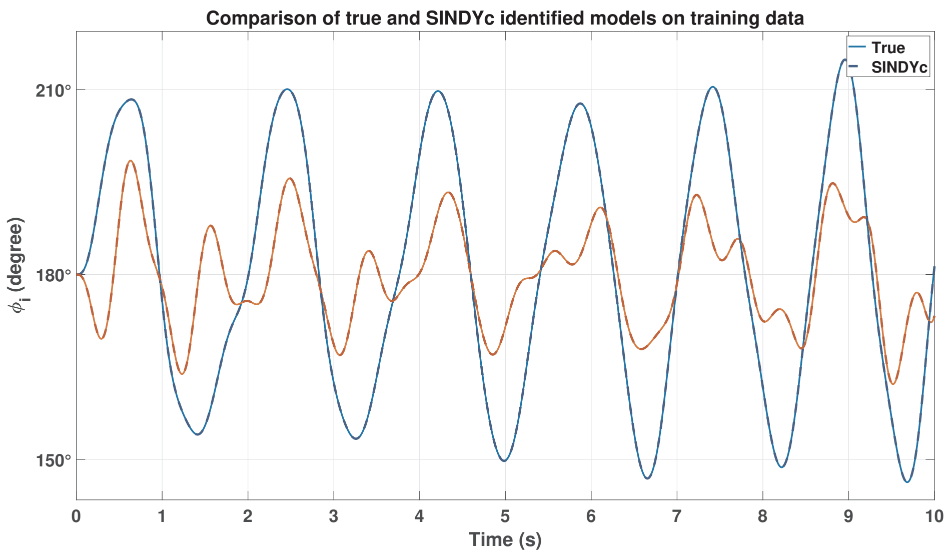


Figure 5.2: Identification with single trajectory and $\mathbf{\Psi_3}$ on training data. (—) and (—) represent the trajectories of ϕ_1 and ϕ_2, respectively

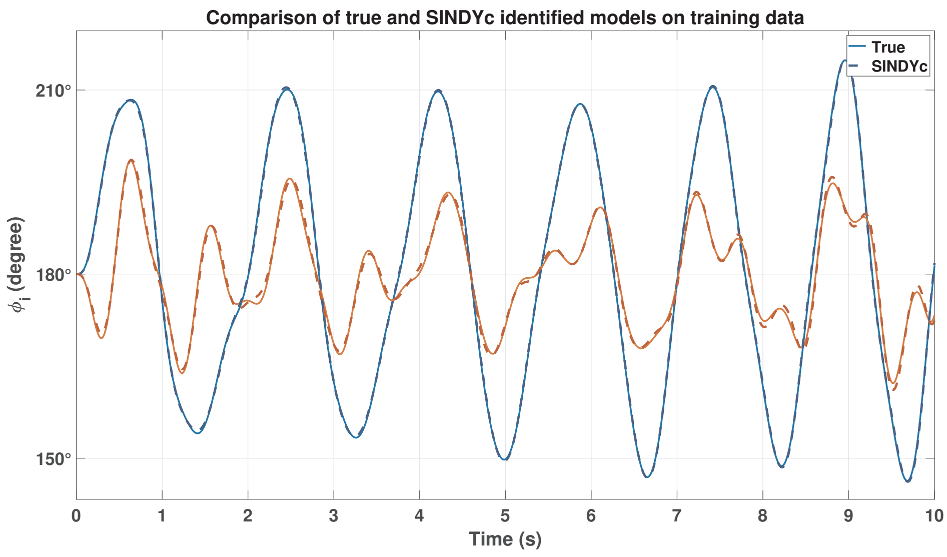


Figure 5.3: Identification with multiple trajectories and $\mathbf{\Psi_3}$ on training data.

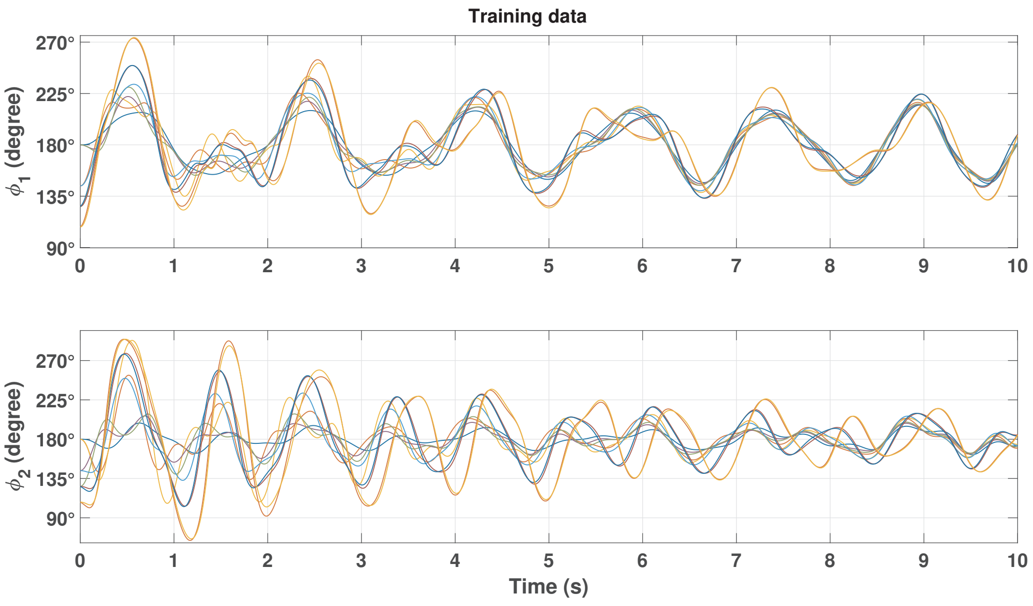


Figure 5.4: Training data from multiple trajectories

In the above figures, $\mathbf{\Psi_3}$ is a vector of observables that is already defined in Section 3.3. From the above figures, one can see that the SINDYc algorithm almost perfectly identifies the time evolution of the state trajectories both in the case of training data obtained from a single initial condition and multiple initial conditions for a specific choice of candidate functions.

Even though it might seem that data from a single trajectory is enough to approximate the dynamics, this will almost always fail when the system is forced with a forcing function that is different from the forcing function used to generate the training data. Revisiting Section 1.4, SINDY-with-control (SINDYc) computes a (tall) sparse matrix of coefficients of the candidate functions $\mathbf{\Xi}$ (which also contains the input function), and this matrix of coefficients is then used to simulate the dynamics for a new forcing function. Figures 5.5 and 5.6 show the prediction of dynamics evaluated on the matrix of coefficients generated from training data for a new 'validation' forcing function in (5.2). It is interesting to note that while the $\mathbf{\Xi}$ generated from a single trajectory worked perfectly well for data that lies within the range of training data, it fails to correlate the dynamics over a longer time span when the forcing function drives the trajectory away from the training trajectory, as observed in Fig 5.5. The predicted trajectory by SINDYc fails to follow the reference trajectory after approximately 1.5 seconds, where both the amplitude and the frequency of the reference trajectory differ from the trajectory used for training. Whereas, in Fig. 5.6 it is seen that the dynamics are approximately predicted for $\mathbf{\Xi}$ generated from multiple trajectories for a 'good' choice of observables. Therefore, it is vital to generate data from multiple trajectories and train the regressors, be it SINDYc or EDMDc, for a close approximation of the dynamics.

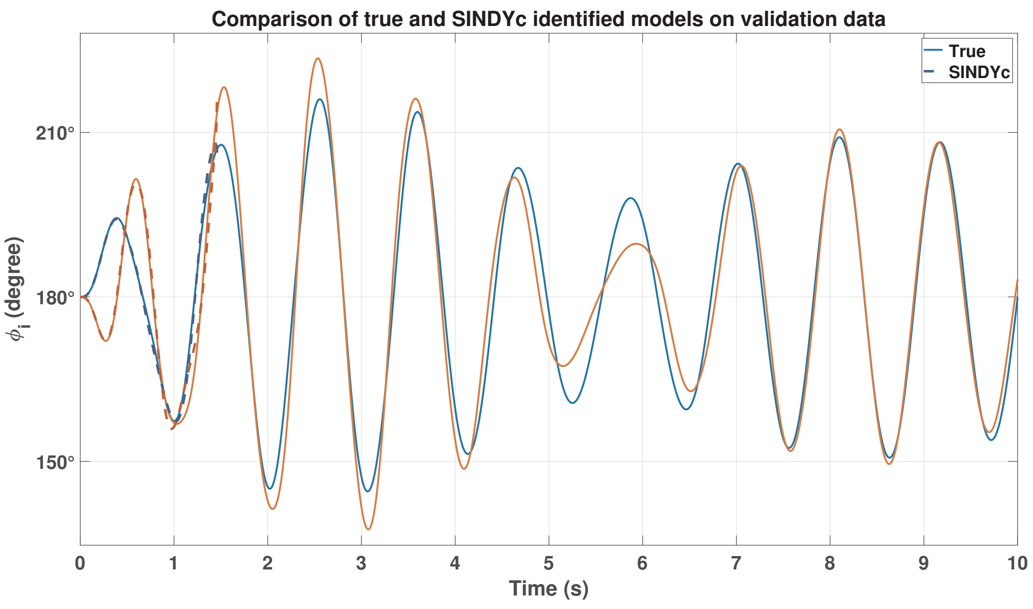


Figure 5.5: Identification with single trajectory and $\mathbf{\Psi_3}$ on validation data

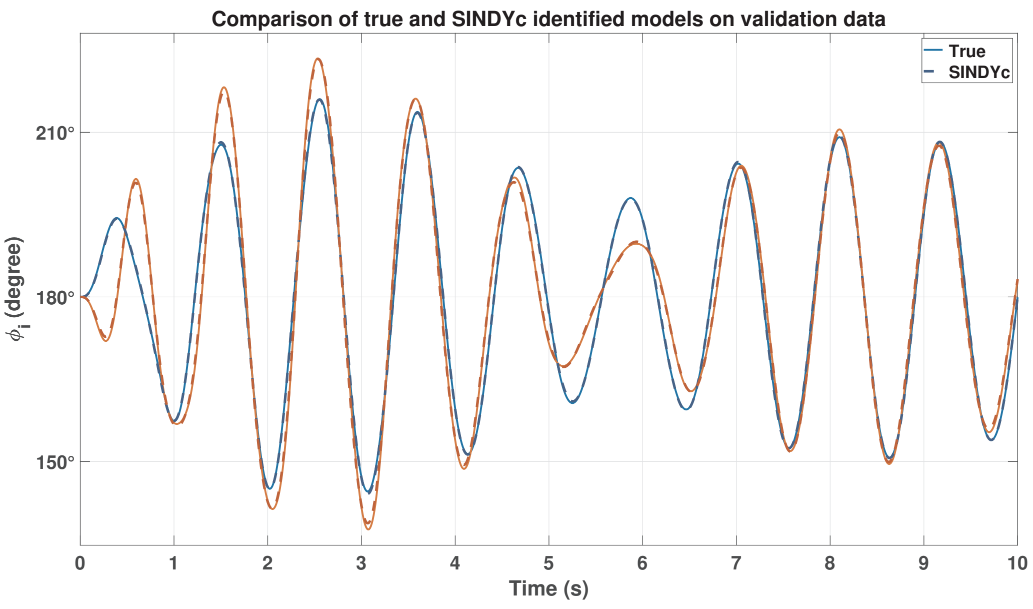


Figure 5.6: Identification with multiple trajectories and $\mathbf{\Psi_3}$ on validation data

Figures 5.7, 5.8 and 5.9 visualize forcing functions and the consequent predictions with a slightly different forcing function from earlier but still within the range defined by (5.2). The forcing function used here for training is a chirp signal with a constant frequency. Again, it is seen that model approximated from data generated by multiple trajectories performs better.

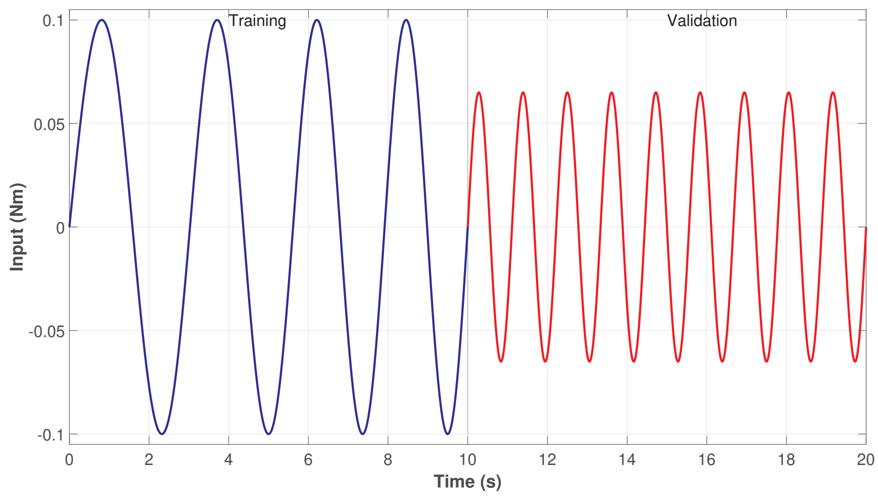


Figure 5.7: The training and validation inputs

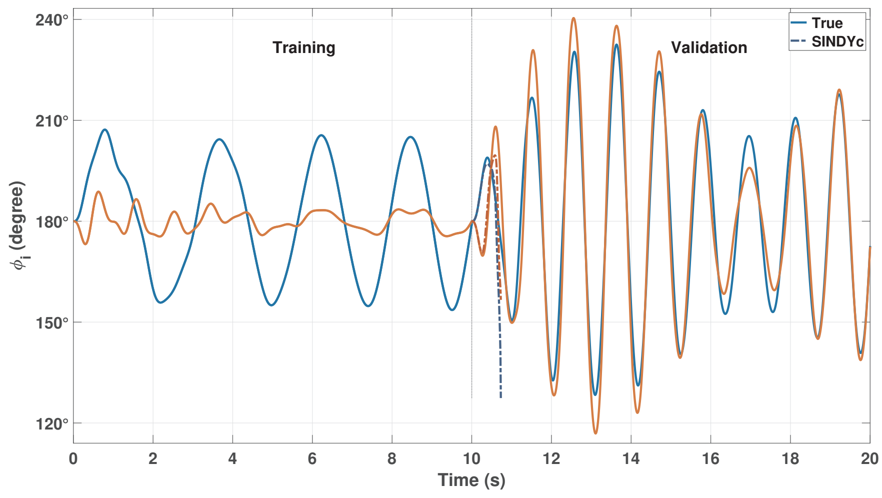


Figure 5.8: Visualization of the validation with single trajectory and $\mathbf{\Psi_3}$

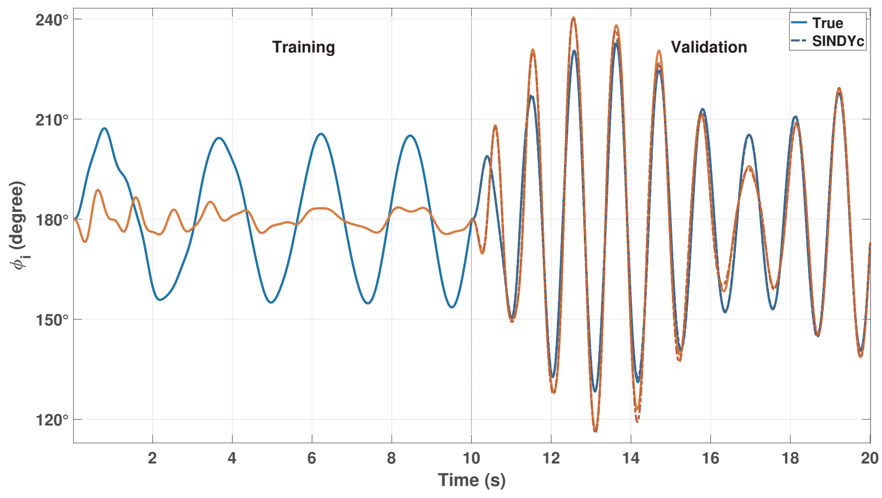


Figure 5.9: Visualization of the validation with multiple trajectories and $\mathbf{\Psi_3}$

The predictions fit the true trajectories for a specific choice of candidate functions, $\boldsymbol{\Psi}_3$, in the above figures. Some or all of the linear or nonlinear functions in this set of candidate functions may form the set of observables required to compute the Koopman operator, generating a linear approximation of the underlying nonlinear system. The sparse coefficients matrix $\boldsymbol{\Xi}$ is a good indicator of such dominant linear or nonlinear functions which can then be considered as observables. It is important to note that the choice of candidate functions, and consequently, the choice of observables is highly subjective, and there are no fixed rules/methods to choose them. Some of the ways have been discussed in Chapter . Figure 5.10 illustrates the effect of choice of candidate functions. The set of candidate functions used in this case to generate the $\boldsymbol{\Xi}$ is $\boldsymbol{\Psi}_2$ which is a set of polynomials up to a second degree. It is seen that $\boldsymbol{\Psi_2}$ can only approximate the dynamics in a small region around the stable equilibrium after which the predicted trajectories do not behave like the reference trajectory. On the other hand, in Figure 5.6, the predicted trajectories closely match the true trajectory. This shows that as one moves away from the desired equilibrium point, the initial state vector needs to be augmented with other nonlinear functions to enable predictions over longer time spans and/or over a larger range.

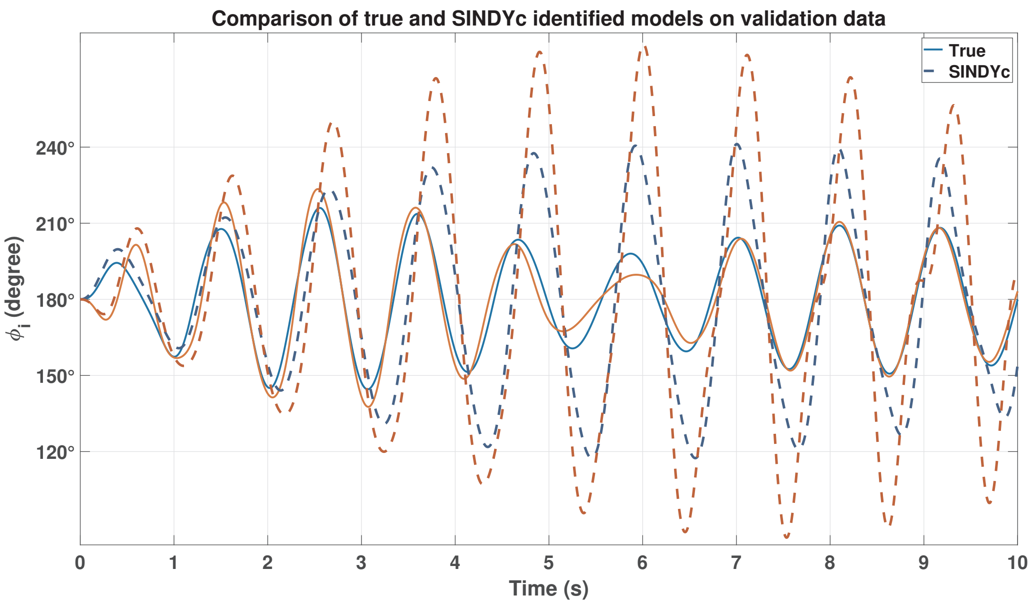


Figure 5.10: Identification with multiple trajectories and $\boldsymbol{\Psi_2}$ on validation data

The following section applies the knowledge obtained from this section to efficiently synthesize a data-driven controller for the ADIP to track a closed-loop reference trajectory.

5.2 Tracking

One can safely say that data-driven control is majorly dependent on the setting up of data, computation of data-driven linear matrices (the finite approximation of the Koopman operator) and stands mostly solved once the parameters of data generation are decided upon. Once the linear matrices are generated, synthesizing a controller follows the same procedure as any linear model. In this thesis, all the controllers used were tuned using a genetic algorithm that simplifies finding the right combination of tuning parameters.

In the following results, for the sake of comparison, the tuning matrices used for all the presented controllers with the exception of the qLMPC are the same with $\mathbf{Q} = \mathrm{diag}(85.38,\ 15.32,\ 0,\ 0.44)$ and $\mathrm{R} = 1.4$. Additionally, the penalty on the integrated error state is $\mathrm{Q}_i = 300$ and the penalty on the final state for MPC is, $\mathrm{Q}_N = Q$.

This section applies the insights and observations from the previous section in tracking a reference trajectory in the up-up configuration of ADIP. The full-state measurements are collected as discussed previously in Section 5.1 (see Figures 5.1a, 5.1b) assuming that a closed-loop controller is available to stabilize the system. Also, the input torque is bounded by the maximum allowable range $[-2.5\ \ 2.5]$Nm.

Figure 5.11 compares the tracking performance of a linear quadratic regulator with integral action on the *local* model generated by Jacobian linearization of the nonlinear dynamics at the point of unstable equilibrium ($\phi_1 = \phi_2 = 0°$) against the Koopman model generated solely from data with the observables vector $\mathbf{\Psi}_1 = [\phi_1\ \phi_2\ \dot{\phi}_1\ \dot{\phi}_2]^\top$, i.e, only the linear measurements of state. It was observed that the linear measurements of the state were enough to track a reference up to 40° with an LQi controller, which also agrees with the LQi applied to the local model. Furthermore, Fig. 5.12 visualizes the quadratic cost J in Eq. 4.3. It can be seen that costs for the local model and the data-driven model are comparable and in fact, the former's cost was a little bit higher than of the latter. Importantly, it must be noted that the Koopman model was generated solely from data with almost no assumptions made about the system.

In the figures 5.11, 5.13, (—) and (—) represent the trajectories of ϕ_1 and ϕ_2 generated by the local model, respectively. Consequently, (- -) and (- -) represent the trajectories of ϕ_1 and ϕ_2 generated by the Koopman model, respectively.

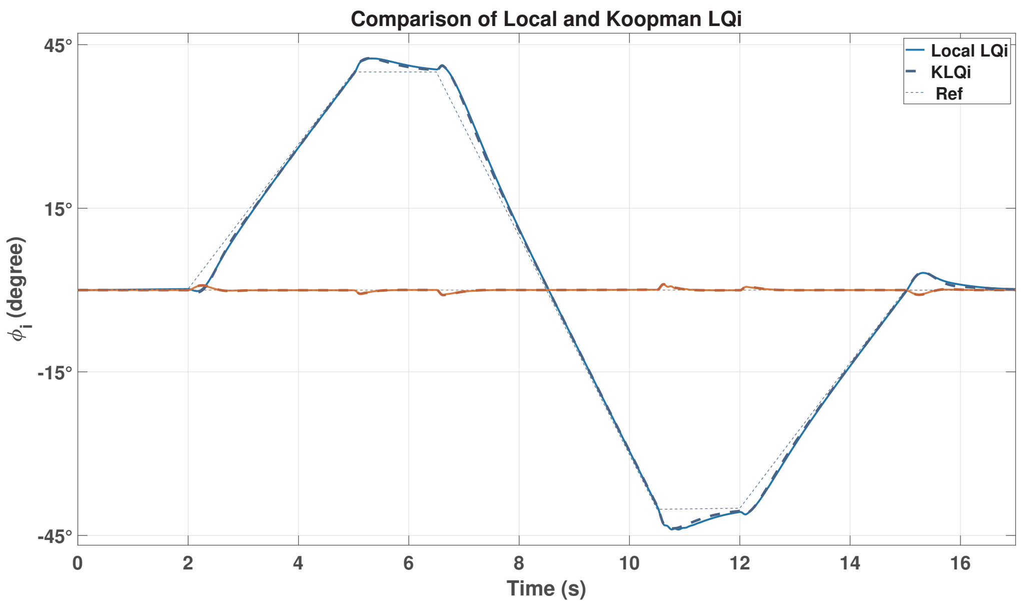


Figure 5.11: Comparison of LQi applied to locally linearized model versus Koopman model with $\mathbf{\Psi_1}$

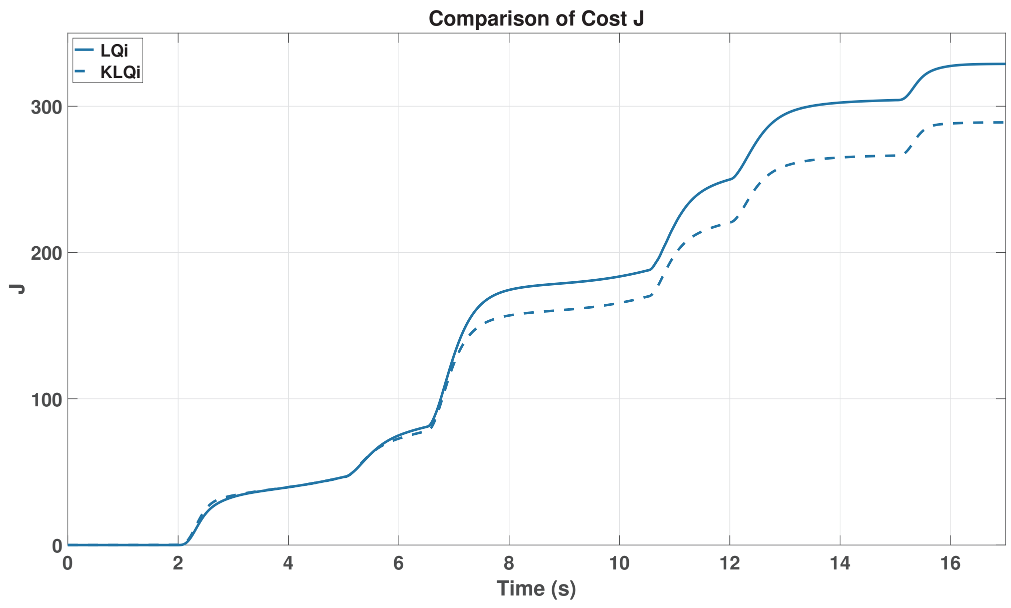


Figure 5.12: Comparison of quadratic cost function J for closed-loop tracking with local and Koopman operator approximated model

Next, a model predictive controller with a prediction horizon of $N_p = 200$, i.e. a prediction of 1 second into the future, was designed for the above-generated models. It was observed that the range of tracking could be increased from 40° to 45°. This was expected as MPC almost always performs better than the linear quadratic regulator, especially more so in tracking problems. This is because, as opposed to the optimization of the quadratic cost function over an infinite time horizon as is the case with LQR (4.3), the MPC optimizes the cost function over a finite time horizon (4.10) and repeatedly computes the optimal input. As a result, the controller can generate optimal control inputs valid for only a small fixed horizon instead of optimal control inputs that are valid over a very long period of time as in the case of a linear quadratic regulator. However, this comes at a cost. The optimal LQR input is computed offline and does not change during the process. In contrast, in MPC, the optimal input is computed online and, therefore, depends on the computer's computational capabilities. This cost can, however, be greatly reduced by formulating the MPC problem in a dense form (Eqs. 4.15 and 4.17) as proposed by Korda et al. [8]. The matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ can be pre-computed offline, and one only needs to update the current state vector $\mathbf{x}_k$ online. This greatly reduces the computational effort and renders the MPC as feasible as the LQi in terms of implementation. Moreover, this dense formulation (Eq. 4.17) is independent of the state's dimension and therefore, augmenting a state with nonlinear observables does not significantly affect the computational time of the MPC. Figure 5.13 compares the local model's tracking performance against the Koopman model. Again, the overall quadratic cost functions are similar for local and Koopman models.

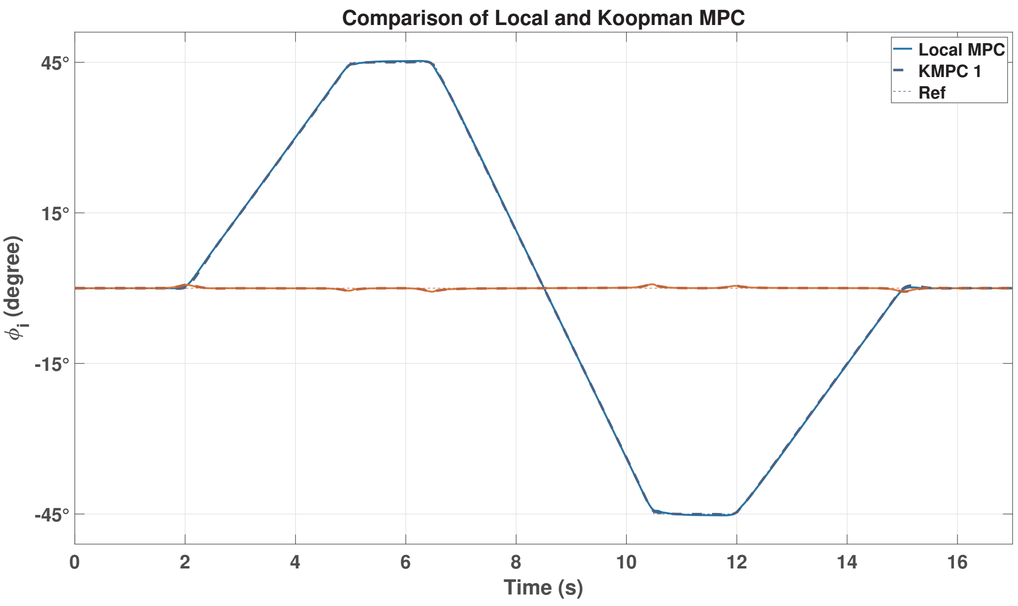


Figure 5.13: Comparison of MPC applied to local model versus Koopman model with $\mathbf{\Psi_1}$

Table 2 presents the corresponding computation times for the local and Koopman models executed with a sampling time of 5ms and 10ms. It should be noted that the controllers for the models generated from data sampled every 10ms are different from the controllers tuned for models generated from data sampled every 5ms. Both the controllers were GA-tuned to de-emphasize the effect of human bias. The comparison between the sampling times is made to highlight the trade-off between computation time and the model-controller combinations' performance. Table 3 presents the relative root-mean-square-error (in %

error w.r.t the reference trajectory of ϕ_1) for the model-controller combinations explored until now in this section.

	Average execution time (ms)	
Model - Controller	$T_s = 5\text{ms}$	$T_s = 10\text{ms}$
Local - MPC	0.56	0.17
KO_1 - MPC	0.59	0.18

Table 2: Average execution time (ms) of the MPC algorithms for different sampling times

	RMSE(%)	
Model - Controller	$T_s = 5\text{ms}$	$T_s = 10\text{ms}$
Local - LQi	6.60	7.13
KO_1 - LQi	6.32	6.59
Local - MPC	0.69	2.84
KO_1 - MPC	0.53	3.53

Table 3: RMSE(%) of tracking errors for different sampling times

As observed in Section 5.1, the choice of observables also plays an important role in approximating the system's dynamics farther away from the point of linearization. While there is no rule book which exactly governs the selection of such observables, it is often possible to work with proven basis functions used to approximate nonlinear dynamics in general. The only issue is that this space of observables spans infinite dimensions and therefore, one may or may not find the 'correct' observables, which will serve the purpose. It is also important to note that not all nonlinear functions of the state approximate the Koopman operator. Some 'observables' may even corrupt the system and/or drive the system to instability. Examples of such approximations are given in Kaiser et al. [29]. The choice of observables, therefore, is particular to the system of interest. In this thesis, the state was augmented with radial basis functions with randomly chosen centres as previously explained in Chapter 3.3. Figure 5.14 compares the performance of an MPC controller acting on the local model, a data-driven model approximated with 4 observables ($\mathbf{\Psi}_1$) that are simply the measurements of the state and another data-driven model with the state vector augmented by 50 radial basis functions ($\mathbf{\Psi}_4$, refer Sec. 3.3), respectively to track a reference trajectory of up to 45°.

Furthermore, it was observed that the range of tracking with MPC applied to the data-driven model approximated with an augmented vector of RBFs could be extended to almost 60°. Figure 5.15 compares the MPC controller designed for a Koopman model approximated with observables vector $\mathbf{\Psi}_1$, which could only track a reference trajectory as good as the local model (w.r.t the range), against a Koopman model approximated with the observables vector $\mathbf{\Psi}_4$ which could extend the range of tracking by almost 50% for a wise choice of observables. As a result, one may hypothesize that there may exist a set of observables that could further extend the tracking range. An efficient way of arriving at such observables, however, is left to further research.

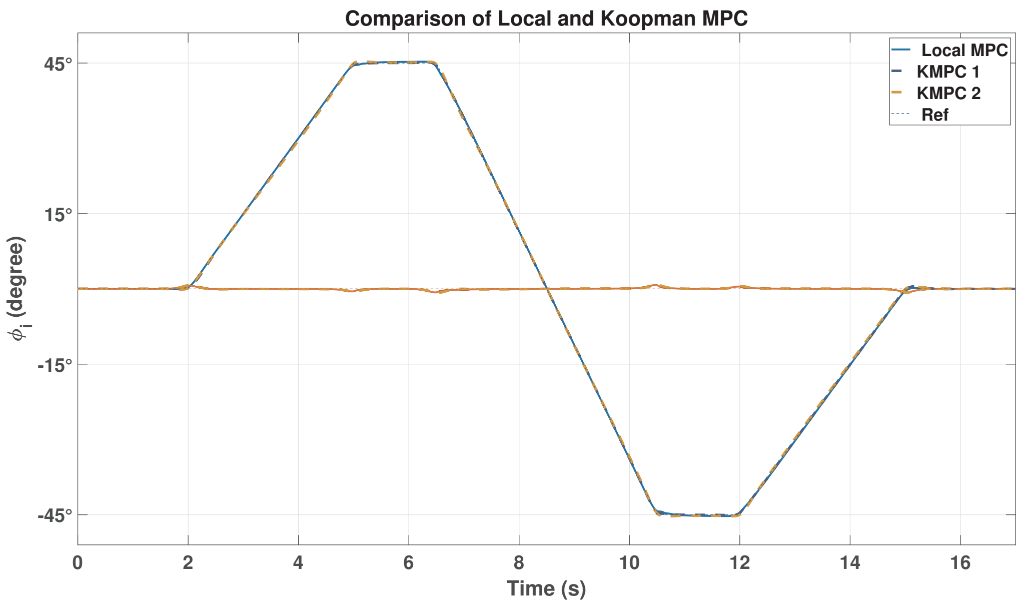


Figure 5.14: Comparison of Local MPC, Koopman MPC for $\mathbf{\Psi_1}$ (KMPC1) and $\mathbf{\Psi}_4$ (KMPC2)

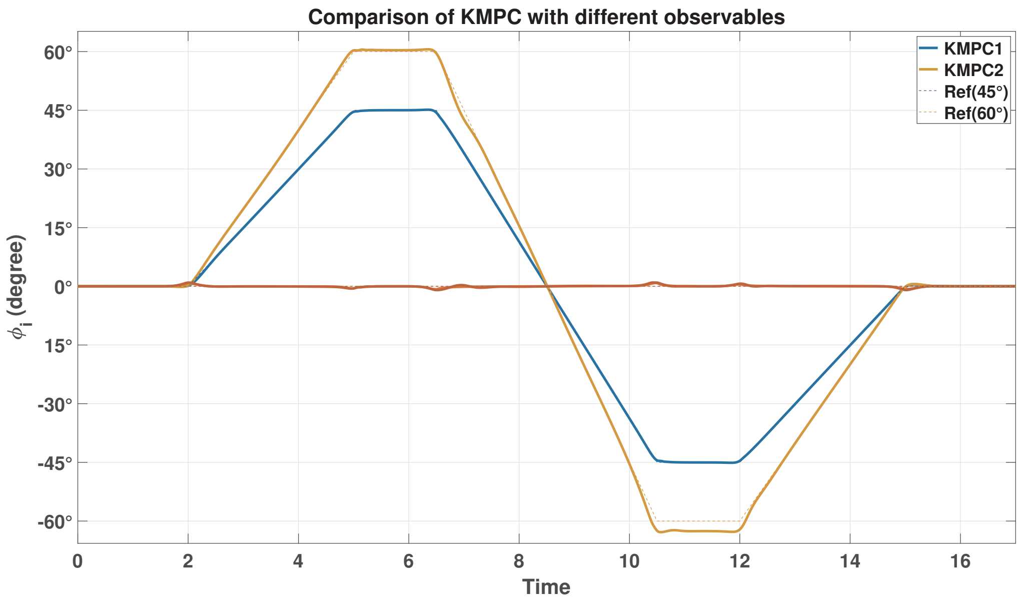


Figure 5.15: Comparison of Koopman MPC for $\mathbf{\Psi_1}$ (KMPC 1) and $\mathbf{\Psi}_4$ (KMPC 2)

Table 4 and Table 5 compare the average execution times and relative RMSE for the above presented model-controller combinations, respectively. It can be seen that while the range of tracking could successfully be increased from 45° to 60° by augmenting the state vector with RBFs, there is no significant increase in the computation time or the relative RMSE (for 45° reference trajectory). This is due to the dense formulation of the MPC, as discussed previously. The slight increase in computation time is because one needs to lift the system's state at every computation step. However, the augmented state would have a negligible effect on the computation time of the optimal control input as the matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are pre-computed offline.

In the following tables in this section, KO_1 and KO_2 are the data-driven models generated by the observables $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_4$, respectively.

	Average execution time (ms)	
Model - Controller	$T_s = 5$ms	$T_s = 10$ms
Local - MPC	0.56	0.17
KO_1 - MPC	0.59	0.18
KO_2 - MPC	0.69	0.28

Table 4: Average execution time (ms) of the MPC algorithms for different sampling times

	RMSE(%)	
Model - Controller	$T_s = 5$ms	$T_s = 10$ms
Local - LQi	6.60	7.13
KO_1 - LQi	6.32	6.59
Local - MPC	0.69	2.84
KO_1 - MPC	0.53	3.53
KO_2 - MPC - 45°	0.72	3.49
KO_2 - MPC - 60°	2.67	2.45

Table 5: RMSE(%) of tracking errors for different sampling times

Next, the robustness of tracking is tested for the synthesized controller. Originally, the controller was synthesized (GA-tuned) for the trajectory shown in Fig. 5.1a where the time to reach the extreme points was $T = 3s$. It is interesting to observe this controller's behaviour on trajectories with steeper slopes, for example, with $T = 1.5s$ and $T = 1s$. Table 6 presents the results of such a comparison. The values shown in red correspond to the percentage increase in the relative RMSE to the relative RMSE of the model-controller combinations for the original reference trajectory, presented in the first column. It must be noted that from here on, the comparisons are made for data-driven models approximated with data sampled at 5ms. The reason 5ms was selected is that the computation times (Table 4) are well within the hardware capabilities of the computer and the relative RMSE (Table 5) for $T_s = 5$ms is much lesser in all the cases when compared to $T_s = 10$ms, making the sampling time of 5ms an obvious choice.

	RMSE(%)		
Model - Controller	$T = 3$s	$T = 1.5$s	$T = 1$s
Local - LQi	7.13	10.03 (+40.7%)	13.35 (+87.2%)
KO_1 - LQi	6.59	9.26 (+40.5%)	12.55 (+90.4%)
Local - MPC	0.69	0.94 (+36.2%)	1.26 (+82.6%)
KO_1 - MPC	0.53	0.85 (+60.3%)	1.29 (+143.4%)
KO_2 - MPC - 45°	0.72	0.86 (+19.4%)	1.58 (+119.4%)
KO_2 - MPC	2.67	2.87 (-)	2.67 (-)

Table 6: RMSE(%) of tracking errors for different target times T

A few observations are in order.

- Firstly, consider the first two rows; it can be seen that the percentage increase in the relative RMSE for the local model and Koopman model KO_1 is comparable, which means that the recorded data captured most of the dynamics of the underlying nonlinear dynamical system. This consequently validates the approach employed in Section 5.1 to record the data and highlights the effectiveness of the data-driven regression methods described in Chapter 1.
- Next, comparing rows 1 and 3 it can be seen that the MPC controller performs slightly better compared to the LQi, which is of course expected. Also, note that the relative RMSE was significantly lower for the MPC controller to start with.
- The comparison of rows 4 and 5 shows that the model with state vector augmented with RBFs seems to capture relatively more relevant dynamics than just the state vector as observables, and consequently, the relative increase in the RMSE for varying slopes is much lesser, i.e., the model-controller combination, $(KO_2 - MPC)$, seems to track the various trajectories much closer than the $KO_1 - MPC$ combination.
- It is seen that the result of comparison between row 2 and row 4, LQi and MPC controller applied to the KO_1 model, respectively, is not as expected. The MPC controller performs relatively worse than an LQi controller. A possible explanation for this could not be ascertained. It could only be guessed that the $KO_1 - MPC$ combination has the lowest relative RMSE of all the combinations for $T = 3s$, and a small but reasonable increase in the RMSE contributes much more than compared to the same in the case of KLQi.
- The last row presents some important insight into the choice of observables. Figure 5.16 visualizes the results of the last row. It can be seen that as the slope becomes steeper, the tracking range of the model-controller combination decreases. A potential explanation to this could be that the choice of observables, while good, may not be the best, and there may exist some other combination of RBFs which perform better. This is left to further research. It can also be construed as a possible limitation of data-driven models, and that these models may not perform as well as a model derived from first principles. This can be seen in the following results.

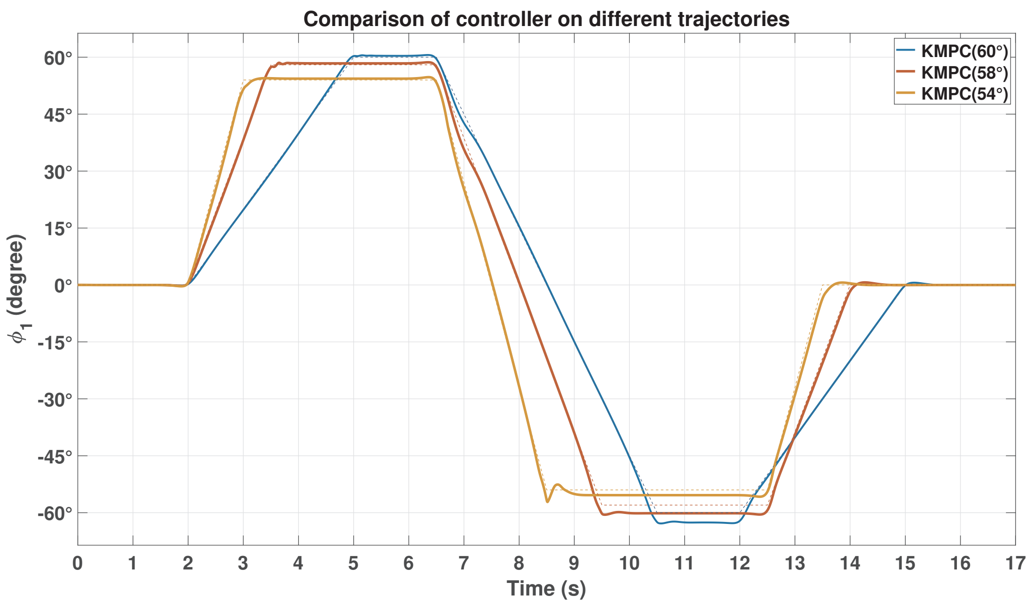


Figure 5.16: Comparison of the synthesized controller on reference trajectories with different degrees of slopes for Koopman model approximated using $\mathbf{\Psi}_4$. (—) – $T = 3s$,(—) – $T = 1.5s$, and (—) – $T = 1s$

Consider a quasi-linear parameter varying (qLPV) model of the ADIP parameterized with scheduling parameters $\rho = [\phi_1 \quad \dot{\phi}_1]$ as described by Cisneros et al. [31]. A corresponding MPC controller is synthesized for this model, tuned by a GA. The tuning parameters are $\mathbf{Q} = \text{diag}(1884.43,\ 1777.43,\ 80.16,\ 0.76)$ and $\text{R} = 26.13$ and $\mathbf{Q}_N = 2219.42\mathbf{Q}$.

It is important to note that the prediction horizon for the qLPV-MPC (here-on, qLMPC) is taken as $N_p = 40$, as against $N_p = 200$, which was the prediction horizon considered previously. The reason to choose a lower prediction horizon in the case of qLMPC is that, at every time step, a linear model is computed for which a new MPC is synthesized which then returns the optimal control input for the next time step and this process is repeated. This is a computationally costly process, and the computation cost further increases exponentially with the increase in the prediction horizon. Table 7 illustrates this.

	Average execution time (ms) for $T_s = 5$ms	
Model - Controller	$N_p = 200$	$N_p = 40$
Local - MPC	0.56	0.05
KO$_1$ - MPC	0.59	0.06
KO$_2$ - MPC	0.69	0.16
qLPMC	10.2	0.40

Table 7: Average execution time (ms) of the MPC algorithms for different prediction horizons

It can be seen that for a prediction horizon of $N_p = 200$, the computation time is more than 10ms for the qLMPC, whereas it is 0.40ms for a prediction horizon of $N_p = 40$. Moreover, the results obtained with $N_p = 40$ can be considered good enough, and there is actually no need to increase the prediction horizon further. In previously considered models, however, the possibility of a longer prediction horizon enables better results. One can use the prediction horizon of $N_p = 200$, simply because the computational cost is well within the hardware requirements. The corresponding results are quite good. The average execution time of the MPC for the local model and data-driven model is significantly low even with a prediction horizon of $N_p = 200$ because the MPC here is based on a linear model whose matrices are pre-computed offline as against online computation of the linear matrices in the case of qLMPC.

Figure 5.17 presents a qLMPC version of the comparison made previously in Fig. 5.16.

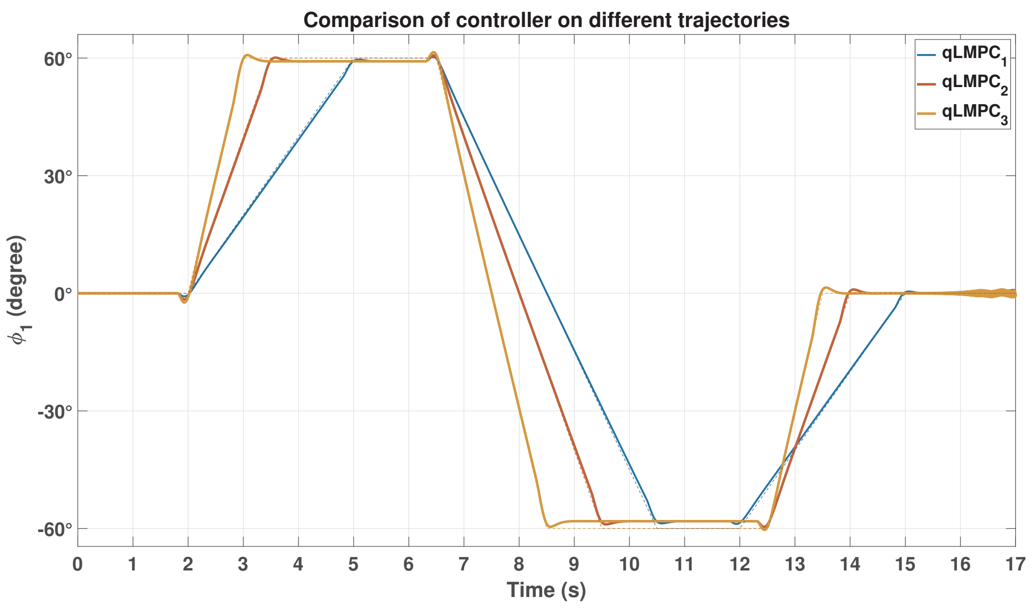


Figure 5.17: Comparison of the synthesized qLMPC controller on reference trajectories with different degrees of slopes. (—) – $T = 3s$,(—) – $T = 1.5s$, and (—) – $T = 1s$

The corresponding relative RMSE values are presented in Table 8.

	RMSE(%)		
Model - Controller	$T = 3$s	$T = 1.5$s	$T = 1$s
qLPV - MPC	2.15	2.24	2.43

Table 8: RMSE(%) of tracking errors for different target times

An important observation can be made from Fig. 5.17 and Table 8; the relative RMSE does not increase significantly with change in the slope of the reference trajectory, and also, the

RMSE for qLMPC is lesser than that of a data-driven model with MPC approximated with $\mathbf{\Psi}_4$. Furthermore, the qLMPC can track up to 60° for all the degrees of slope considered whereas, the $\text{KO}_2 - \text{MPC}$ failed to track the desired target for steeper slopes of the reference trajectory. This points out a key limitation of data-driven models. A data-driven model might work better than a locally linearized first-principles model in the region where training data is captured, but it is possible that the data models may not perform on par with a quasi-linear parameter varying model. With the increase in computational capabilities, the qLPV approach seems much more promising. Nevertheless, one must note that an important assumption made here is the availability of a mathematical model of the system. Without this, the qLPV approach is not possible and the data-driven model stands as the obvious choice.

During the course of implementation, two additional important observations were made. First, although the Koopman approximated system is a linear representation of the underlying nonlinear dynamics, and in theory one can use any of the model-based linear control techniques to control the system, it is highly recommended to use a predictive controller like the MPC rather than a non-predictive controller like LQi for tracking tasks. This is because the extended state of observables more often than not contains nonlinear functions of the state. The LQi which computes the cost over the infinite horizon fails to solve the optimal control problem. It requires careful handpicking of observables to overcome this. Whereas, since the MPC predicts only over a reasonably small time horizon, it is often easier to manage any irregularities caused by nonlinear observables' evolution. Therefore, when working with data-driven models, MPC is a better choice than non-predictive controllers.

Second, as an additional exercise, the possibility of identifying and controlling a highly unstable system like the ADIP itself in the absence of an optimal controller was explored. At the beginning of this chapter, it was assumed that a plant model was already available. Therefore, an optimal controller, such as the linear quadratic regulator could be synthesized. A Koopman approximated linear system could then be modelled and controlled from just the measured closed-loop data. In the absence of such a controller, one may synthesize a non-model based PID controller that may not have the optimal tuning parameters for it to cover the necessary range of the desired trajectory. It was observed that it is possible to extend this range by recursively applying the EDMD approximation on the obtained data to generate a new linear model at every step for which a powerful model-based controller could be designed and consequently extend the range.

As an example, assume that the measurement data could be recorded for only 5° from the up-up configuration after which the controller fails to track, as opposed to data recorded for a range of 40° with an LQi controller(as shown in Fig 5.1a). DMD could then be applied to such data to generate a linear model. With a linear model's availability, one could synthesize a model-based linear controller like the LQi or MPC. It is also possible to generate GA optimized tuning parameters for maximum reach. In this particular case, the range of tracking could be extended by an additional 10° with an MPC's help and with the original measured state as the observable vector. Data was then recorded from this new system with a range of 15°, and the same process was applied to extend the range at each step. Finally, it was observed that the closed-loop tracking performance of the model-controller(KO$_3$-MPC) combination so obtained was comparable to that obtained from original data with the observable vector $\mathbf{\Psi}_1$. Figure 5.18 and Table 9 illustrate the tracking performance of such a model with the same controller as used previously with other models. The results can be significantly improved with a controller specifically tuned for this model. However, it failed to perform as well with the $\mathbf{\Psi}_4$, i.e., with the observable vector containing the state and previously defined thin-plate spline RBFs. A potential reason for this might be the RBFs' choice, and there could be some combination of RBFs that may result in better performance. This is left to future research.

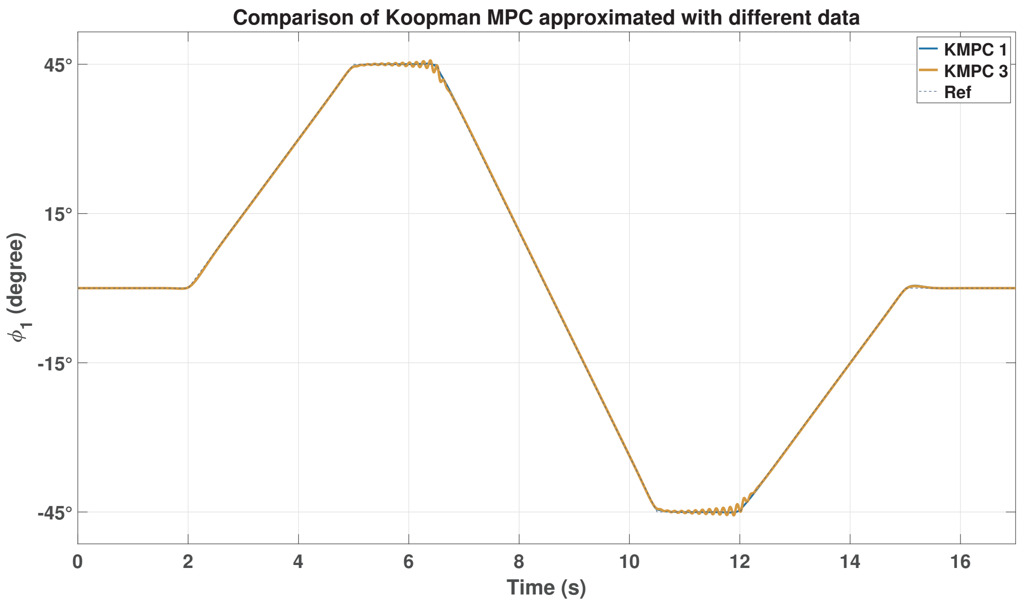


Figure 5.18: Comparison of Koopman MPC on data-driven model generated with data over a range of upto 40° (KO$_1$) versus data-driven model (KO$_3$) constructed recursively with data generated over a range of upto 5°

Model - Controller	$T_s = 5$ms
KO$_1$ - MPC	0.53
KO$_3$ - MPC	0.95

Table 9: RMSE(%) of tracking errors for (KO$_1$) and (KO$_3$)

It is important to reiterate two important assumptions made during the course of this implementation. First, the reference trajectories are fixed and are assumed to be known *a priori*. This is important because the MPC controllers synthesized during the implementation are supplied with future values of the reference to enable the controller to 'see' into the future and anticipate any changes in the trajectory. The controller then acts accordingly. This is different from the normally implemented MPC controllers, where only a current value of the reference is supplied. Figure 5.19 visualises this difference. When the controller is supplied with a reference signal of the next N_p steps, it can anticipate the change in trajectory and respond accordingly.

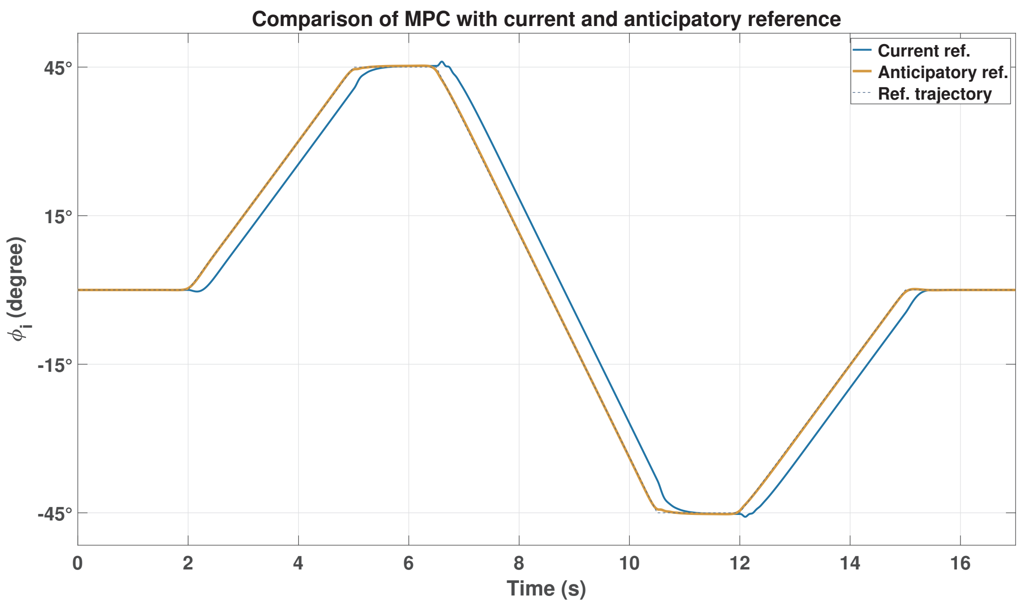


Figure 5.19: Comparison of controller acting on current reference versus anticipatory reference

Second, a major assumption made during the genetic algorithm (GA) powered tuning of parameters in this thesis was that it was assumed a nonlinear simulation model of the plant (ADIP) is available to run the GA at all times. This may not always be the case, especially when dealing with systems where the dynamics are unknown. In such a case, one has to either resort to intuition and knowledge of tuning parameters or estimate a nonlinear black-box model of the unknown plant.

5.3 Swing-up

Finally, a comparison of two different strategies for swing-up is presented in this section. Both the swing-up strategies are model-based. The first strategy is an energy-based swing-up law adopted from Fantoni et al. [27]. Figure 5.20 shows the evolution of the arm and pendulum angles from down-down configuration at $\phi_1 = \phi_2 = 180°$ to up-up configuration with $\phi_1 = \phi_2 = 0°$. The approach presented in [27] ensures that the arm swings up and brings the pendulum into a homoclinic orbit where a stabilizing controller takes over from the energy-based controller and thereafter performs a regulation task to the up-up configuration. The parameters were tuned by a GA which resulted in $k_E = 4.55$, $k_P = 0.4$, and $k_D = 0.8$. Similarly, Figure 5.21 shows the evolution of arm and pendulum angles for a swing-up approach based on the previously introduced qLPV-MPC approach. Here, only one controller is sufficient for both swinging up the pendulum and stabilizing it about the up-up configuration. The swing-up parameters were GA tuned; the parameters are $\mathbf{Q} = \text{diag}(2623,\ 746.36,\ 46.65,\ 0.51)$ and $\text{R} = 61.4$ and $\mathbf{Q}_N = 427.25\mathbf{Q}$. Furthermore, Figure 5.22 shows the change in the system's total energy. The energy-based swing-up law performs significantly better than the qLMPC swing-up law. However, the latter has the advantage that it is just a single controller, unlike the former, which requires an additional stabilizing controller.

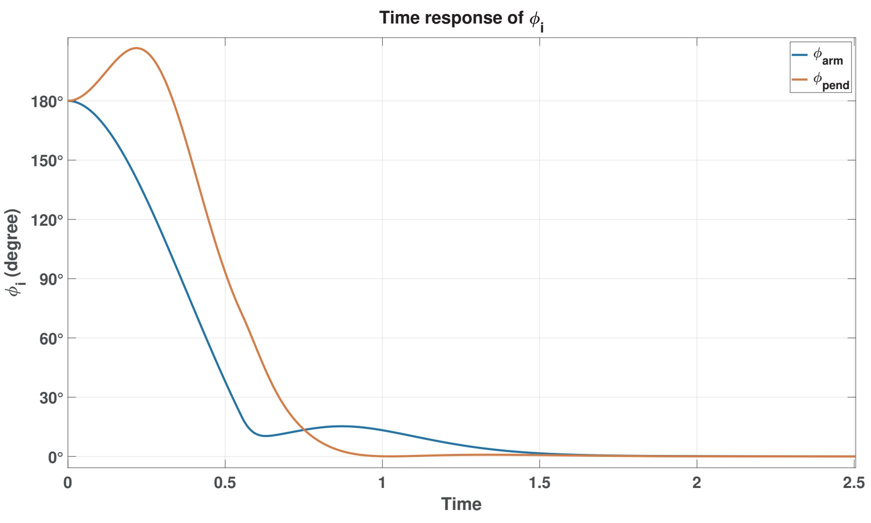


Figure 5.20: Swing-up performed by energy-based controller and a stabilizing LQR.

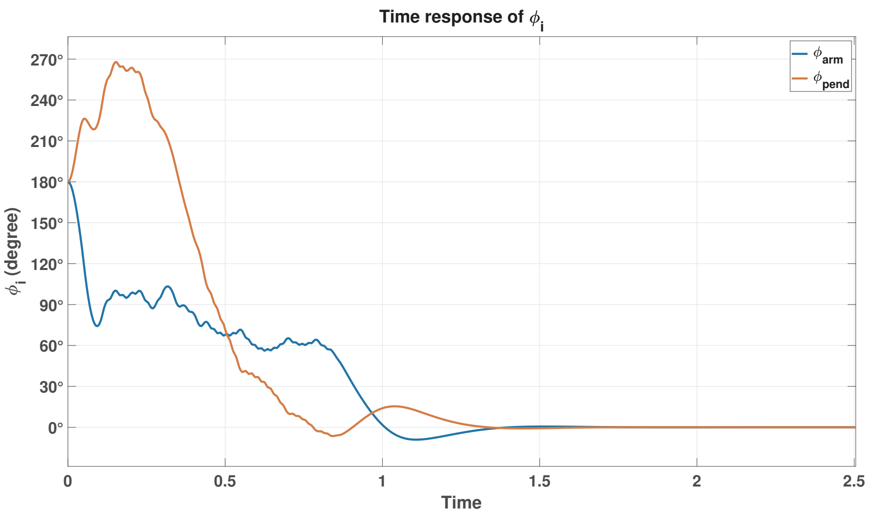


Figure 5.21: Swing-up performed by qLPV-MPC controller

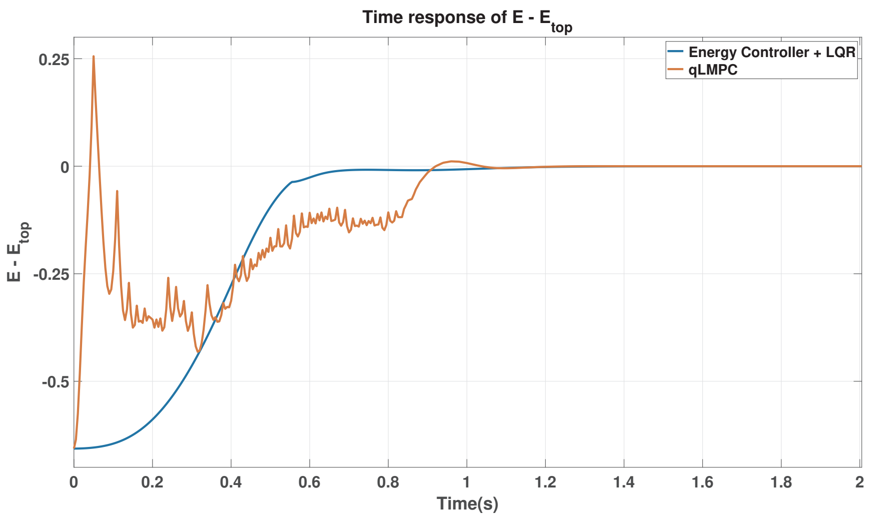


Figure 5.22: Time response of $\tilde{E}$.

6 Conclusion and Future Scope

This thesis work focuses on the merits and demerits of implementing the Koopman operator theory for model-based control. The central idea of the Koopman operator framework is to lift the nonlinear dynamics of a system to a higher dimensional space where the evolution of dynamics becomes approximately linear. Such linear data-driven models were derived using known regression techniques like DMD (Sec. 1.1) and EDMD (Sec. 1.3). Model-based controllers were then synthesized for such data-driven models, and the performance of the controllers was evaluated on a trajectory tracking problem. The data-driven models on a whole exhibit superior performance as compared to locally linearized models and moreover, linear control design methods could be readily applied to these models. It was observed that increasing the 'complexity' of candidate functions, or in simple words, augmenting the state vector with nonlinear functions of the state further improved the performance of data-driven approximation models. For example, in the open-loop identification (Sec 5.1), with the increasing complexity of the candidate functions, the SINDy regression algorithm could predict the state's evolution over a longer time period and for different operating conditions. It was also found that the prediction could be further improved when the data are sampled from various trajectories. The flexibility of data approximation methods allows for data from multiple trajectories and missing data points to be used. Furthermore, it was observed that the volume of data and its distribution across the state space has a significant effect on the computation of the Koopman operator.

These vital observations obtained from open-loop identification were applied in recording the data for closed-loop identification since the Up-Up configuration is where the focus of this thesis work lies. The closed-loop performance of the data-driven controllers was compared to that of the locally linearized controllers. To this end, a first-principles model was derived (Sec. 3.2) and locally linearized by Jacobian linearization at the Up-Up position. Both a non-predictive controller such as the linear quadratic regulator and a predictive controller such as the model predictive controller were synthesized to compare the tracking performance in combination with the data-driven model. It was observed that non-predictive controllers do not perform as well as predictive controllers when the state is augmented with certain nonlinear functions of the state. It is hard to predict which functions might fail non-predictive controllers, and therefore, when working with data-driven models, it is recommended to work with predictive controllers. Moreover, the tracking performance of the predictive controller was far greater than the non-predictive controller (Tab. 5). And, with the recent advancement in computational capabilities, predictive control is much less cost-intensive than previously. Also, it must be noted that the computational complexity of a high-dimensional lifted system is rendered virtually independent of the dimension of the state by formulating the MPC problem using the so-called dense-form [8]. Additionally, with a judicious choice of observables, it was observed that the data-driven models could capture nonlinearities that are otherwise not captured by locally linearized models. As a result, the tracking range of data-driven model-controllers could be significantly improved over the locally linearized model-controllers or even other data-driven models that are approximated with only measurements of state (Fig. 5.15).

The robustness of tracking performance was tested for the data-driven model approximated with nonlinear functions of the state (Fig. 5.16). It was observed that as the slope of trajectory became steeper, the performance of the data-driven model controller decreased. There could be two potential reasons for this. First, the set of observables or the nonlinear functions of the state may not be optimal in the sense of capturing the necessary nonlinearities, and there may exist some other combination of observables that perform better. The possibility of finding these 'best' observables is left to the scope of future research. Second, the data-driven model's performance degradation may also be construed as a possible limitation of data-driven models. It must be noted that 'extended' tracking range here means that the controller can track a trajectory beyond the range of trajectories used for training the data-driven model. Although the controller could successfully track an extended trajectory in a fairly good manner, as the slope of the trajectory increased, the controller could not perform as efficiently as desired. This could mean that the state-space in which the data-driven controllers can perform equally well or even better than the locally linearized first-principles model-controllers is limited to the range of training data. Nevertheless, the possibility of extending the range of a controller by simply augmenting the state with a wise choice of observables seems to be a promising area of research.

Moreover, the data-driven model was compared with a quasi-linear parameter varying model (Fig. 5.17). Some interesting insights could be derived from this comparison. Firstly, although the qLMPC is a computationally expensive process compared with Koopman MPC, the computation time was well within the acceptable hardware limit. Second, the relative RMSE for a qLMPC was significantly lesser than that of the Koopman MPC. Moreover, Cisneros et al. [31] were able to achieve up to $80°$ as the maximum range of tracking. Therefore, it can be concluded that the first-principles model-based qLMPC performs much better than the Koopman MPC, which heavily depends on the choice of observables. However, one major assumption that is made here is the availability of the first-principles model. In the absence of a first-principles model, the Koopman model is a powerful and useful approximation of the underlying nonlinear dynamics. Thus, it can be said that the choice of model depends on the particular use case.

Finally, two different strategies to swing-up the pendulum from its stable equilibrium and balance it about its unstable equilibrium were evaluated. This comparison was made to investigate the possibility of swing-up using MPC controller and evaluate it against a proven swing-up law. The energy-based swing-up law proposed by Fantoni et al. [27] could achieve successful swing-up in just one swing as the swing-up law ensures the convergence of the pendulum trajectory to a homoclinic orbit around the unstable equilibrium. When the pendulum enters this homoclinic orbit, the controller is switched from an energy-based controller to a stabilizing controller such as the LQR to stabilize the pendulum about its unstable equilibrium. This is unlike the second strategy which employs the qLPV approach to linearize the plant at every time step and synthesize a corresponding model predictive controller. Although both the controllers achieve swing-up and stabilization in approximately the same time (Sec. 4.1), it must be noted that the former approach employed two controllers, one for swing-up and one for stabilizing the pendulum, whereas the latter employs only a single controller to achieve both the swing-up and stabilization tasks.

Future Scope

In this thesis work, the observables, especially the RBFs, were chosen at random from a defined set. Although it became evident that a fortuitous set of observables significantly improved the performance of a tracking controller, the case for data-driven controllers can be strengthened if one could find a mathematical approach to exactly identify such observables. The SINDy algorithm is a good start in this research direction. One could also utilise any prior information of the particular dynamical system to discover such observables. However, one must think of a way to validate any such potential observables obtained based on physical insight or intuitive knowledge of the dynamics. Future work should also focus on proving or imposing closed-loop stability guarantees. An interesting direction for future work of this particular thesis work can be in the area of online-learning or active-learning, wherein a new Koopman model is learned online whenever novel dynamics are encountered. This is similar to a linear parameter varying model based on first-principles and has been explored for a Control Moment Gyroscope by Cisneros et al. [22].

References

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

[2] I. Mezić, “Spectral properties of dynamical systems, model reduction and decompositions”, *Nonlinear Dynamics*, vol. 41, pp. 309–325, Aug. 2005.

[3] I. Mezić, “On applications of the spectral theory of the koopman operator in dynamical systems and control theory”, in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 7034–7041.

[4] A. Mauroy and J. Goncalves, *Linear identification of nonlinear systems: A lifting technique based on the koopman operator*, 2016.

[5] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Generalizing koopman theory to allow for inputs and control*, 2016.

[6] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, “Extending data-driven koopman analysis to actuated systems”, *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 704–709, 2016.

[7] A. Surana, “Koopman operator based observer synthesis for control-affine nonlinear systems”, Dec. 2016, pp. 6492–6499.

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

[9] I. Abraham, G. Torre, and T. Murphey, “Model-based control using koopman operators”, Sep. 2017.

[10] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition.* Society for Industrial and Applied Mathematics, 2016.

[11] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, “A data–driven approximation of the koopman operator: Extending dynamic mode decomposition”, *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, Jun. 2015.

[12] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, “Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control”, *PLOS ONE*, vol. 11, no. 2, 2016.

[13] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”, *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[14] K. Ogata, *Modern Control Engineering.* USA: Prentice Hall PTR, 2001, isbn: 0130609072.

[15] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design.* Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005.

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

[17] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, p. 047 510, 2012.

[18] C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows", *Journal of Fluid Mechanics*, vol. 641, pp. 115–127, 2009.

[19] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, 2014.

[20] M. Korda and I. Mezić, "On convergence of extended dynamic mode decomposition to the koopman operator", *Journal of Nonlinear Science*, vol. 28, no. 2, pp. 687–710, 2017.

[21] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control", *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016, 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[22] P. S. G. Cisneros, A. Datar, P. Göttsch, and H. Werner, "Data-driven quasi-lpv model predictive control using koopman operator techniques", in *21st IFAC World Congress*, 2020.

[23] P. S. G. Cisneros, S. Voss, and H. Werner, "Efficient nonlinear model predictive control via quasi-lpv representation", in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3216–3221.

[24] B. Cazzolato and Z. Prime, "On the dynamics of the furuta pendulum", *Hindawi Publishing Corporation Journal of Control Science and Engineering*, vol. 8, Mar. 2011.

[25] M. Mongillo, "Choosing basis functions and shape parameters for radial basis function methods", *SIAM Undergraduate Research Online*, vol. 4, Jan. 2011.

[26] M. W. Spong and D. J. Block, "The pendubot: A mechatronic system for control research and education", in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1, 1995, 555–556 vol.1.

[27] I. Fantoni, R. Lozano, and M. W. Spong, "Energy based control of the pendubot", *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 725–729, 2000.

[28] X. Xin, M. Kaneda, and T. Oki, "The swing up control for the pendubot based on energy control approach", *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 461–466, 2002, 15th IFAC World Congress.

[29] E. Kaiser, J. N. Kutz, and S. L. Brunton, *Data-driven discovery of koopman eigenfunctions for control*, 2020.

[30] J. Rossiter, *Model-Based Predictive Control: A Practical Approach.* Jan. 2003.

[31] P. Cisneros and H. Werner, "Wide range stabilization of a pendubot using quasi-lpv predictive control", *IFAC-PapersOnLine*, vol. 52, pp. 164–169, Jan. 2019.