Department of Informatics

Technical University of Munich

Master's Thesis

Landing a Spaceship with Koopman Operator Theory and Model Predictive Control

Sabin Bhandari

Department of Informatics

Technical University of Munich

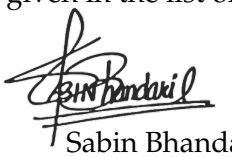
Master's Thesis

Landing a Spaceship with Koopman Operator Theory and Model Predictive Control

Landung eines Raumschiffs mit Koopman-Operator-Theorie und modellbasierter, prädiktiver Regelung

Author:	Sabin Bhandari
Thesis Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	Dr. Felix Dietrich
Submission Date:	November 15th, 2021

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.



November 15th, 2021 Sabin Bhandari

Acknowledgments

I would express my deepest gratitude and appreciation to my committee chair, my advisor Professor Felix Dietrich for his consistent support and guidance during this thesis. I am indebted for his unwavering support through scheduled meetings, prompt response and continuous encouragement. This accomplishment would not have been possible without him.
Thank you.

Abstract

Landing a spacecraft is a challenging task as there are different external (environment) and internal (measurements) factors that play an essential role in precisely landing the ship in the target with small error margins. Even though the task is difficult, the success of such projects can lead to large cost reductions through energy and resource efficiency. Model Predictive Control can help to achieve the task but requires reasonably accurate models of the real spacecraft. The task is non-trivial as the dynamics of the spacecraft are nonlinear and uncertain, which leads to difficulty in constructing models. Different Machine Learning techniques could be used to build such models, however, this results in a nonlinear complex representation that is difficult to understand. The processes and outcomes of such models are difficult to interpret, and they usually function as a "black box" mapping of input and output. Thus, the main idea of this thesis is to use the Koopman operator to model the nonlinear dynamics of the spacecraft into a higher dimensional lifted feature space where its evolution is linearly represented. The linear operators built on the Koopman operator framework are simple, data-driven, and used to design controllers, in our case linear Model Predictive Control without any nonlinear optimization schemes. Furthermore, we explore how the predictors obtained through this method are comparatively superior in terms of performance to the existing linear predictors. When used in a Model Predictive Control scheme, the model is efficient, allowing for quick control loop optimization. As a result of the effective linear model controlling techniques, it is suitable for real-time systems. Landing a spaceship has many requirements that are similar to the stabilization of an inverted pendulum: the spaceship is controlled from below through the thrusters, which act as the cart in the pendulum. The process of building predictors and applying Model Predictive Control is similar. That is why a pendulum system is used as an illustrative example in this study.

Contents

1 Introduction

From the advent of civilization, humans have looked up to the sky and dreamed about reaching the moon. In the last century, thanks to rapid technological advancement, the study of the Earth and its surrounding atmosphere have considerably progressed. The long-range ballistic missiles initially used by Germany during WWII paved the way for launch vehicles. An interest in such missiles later sparked a space race between the Soviet Union and the United States. The first successful feat of space travel was on October 4, 1957, when the Soviets launched their first artificial satellite, Sputnik 1, into space. On April 12, 1961, Russian Lt. Yuri Gagarin became the first human to orbit Earth. Neil Armstrong, the first man to walk on the moon, made his "one great leap for mankind" on July 20, 1969. Between 1969 and 1972, six Apollo missions were launched to investigate the moon [2]. Over the last 50 years, autonomous spacecraft have been used to land multiple rovers on Mars [18], probes in Titan [36], asteroids [8], and return missions for humans in space [10].

Landing a spaceship is a challenging task. Precision landing of spaceships helps research and exploration, and makes spaceship resources reusable. To land a spaceship, several factors must be considered. Spaceships that enter or leave the atmosphere experience extreme conditions. Heating, atmospheric friction, drag forces, and radiation should be accounted for. As the slightest error can have catastrophic effects, landing spaceships should have a small margin for error. The vertical position and velocity should be precisely controlled. Any sudden disturbances such as the wind should not affect the ability to hit the target with precision. The field of vertical landing is still in its early stages of research and resources for spacecraft control are limited, so landing a spaceship vertically is an arduous undertaking. Commercial space stations like Axiom Space and the Bigelow Commercial Space Station, as well as private firms like SpaceX and Blue Origin, have and will continue to revolutionize space travel. Blue Origin's 'New Shepard' rocket successfully landed on its West Texas test site several times.SpaceX managed to land its Falcon 9 rocket on a floating landing platform called the Autonomous Spaceport Droneshop (ASDS) [10].

We use Koopman-based Model Predictive Control MPC in this study to achieve optimal control of an example system (inverted pendulum on cart), which can then be applied to a spaceship landing. Koopman operator theory has been used for control of dynamical systems. A major advantage of the Koopman operator is that it makes use of linear predictors. Linear Predictors are a type of artificial dynamical system that can forecast the future state (or output) of a given nonlinear dynamical system based on measurements of

the present state (or output) as well as the system's current and future inputs. The process of constructing linear predictors is data-driven, i.e. no prior knowledge of the dynamics of the system is required. To compute a finite-dimensional approximation of a controlled Koopman operator, the paper [23] utilized a modified version of the Extended Dynamic Mode Decomposition (EDMD). The collection of observables appearing in the EDMD is given a specific structure, and the resulting approximation of the operator takes form of a linear controlled dynamical system. It is shown that linear predictors generated in this manner outperform Carleman linearization and local linearization approaches in terms of predictive ability.

Nonlinear Model Predictive Control is a powerful approach has been used to control various aerodynamical systems. It is applied for path planning and optimal control problems for multiple mobile robots in a complex three-dimensional environment [34]. Nonlinear MPC is used for control and guidance for a propeller-tilting hybrid unmanned air vehicle [4]. All of these methods have the difficulty of tackling constrained optimization problems in real time. This is why the Koopman-based MPC controller is used in this work.

Koopman MPC has been successfully used in a variety of fields. The paper [1] explores the utility of the Koopman operator theory to control the robotic system. The Sphero SPRK robot is used to investigate the use of the Koopman operator in a reduced state representation context, where increased complexity in the basis function improves open- and closed-loop controller performance in a variety of terrains, including sand. The paper [12] presents the result of the utilization of the Koopman operator as a linear predictor to approximate a nonlinear vehicle model to a higher dimensional space where its system dynamics are linear and used to perform a linear Model Predictive Control design. The nonlinearities modeled in this case are rigid-body dynamics, coordinate system transformations, and the tire. The results from the experiment show that unlike the MPC based on local linearization, the Koopman-based controller is capable of recovering from a situation where the vehicle slides sideways in one continuous motion.

In another work, [6] proposes a methodology for closed-loop feedback control of nonlinear flows that is fully data-driven and model-free, based on the recent development of the Koopman Model Predictive Control framework. The combination of Koopman with MPC outperformed the feedback techniques based on local linearization and with sub-millisecond computation time. Finally, [25] demonstrates the first application of the Koopman MPC for the control of power grid dynamics and transient stabilization. The data-driven control framework allows for the use of efficient linear MPC computational tools to control this highly nonlinear dynamical system. The results look promising with successful cascaded grid stabilization without model knowledge, a distributed control structure (one controller per grid), and quick computation time.

The main objective of this thesis is to implement dense form Koopman MPC for the

control of a dynamical system and compare it with the traditional nonlinear MPC. The Koopman-based linear predictors are generated from the nonlinear dynamical system, which is then used by dense form Koopman MPC to control the position of the cart and the angle of the pendulum. Then, a spaceship model is presented with all of the necessary parameters as well as an optimization function, and the control will be implemented in future work.

The thesis is divided into three main sections. The first section describes the theoretical background of topics related to this thesis such as Koopman theory, linear predictors, lifting functions, Model Predictive Control, and system dynamics. In the second section, the main implementation of the control strategy of Koopman MPC is described. This involves data preparation, model setup, experimental setup and results. Finally, the last section summarizes the work, concludes the thesis report, and discusses future directions for the research.

2 Theoretical Background

2.1 Classical theory of dynamical systems

In the abstract sense, a dynamical system is made up of two parts: a collection of states that is used to describe the evolution of a system, and a rule that governs the evolution. There are many instances of dynamical systems in physics where the evolution of the physical variables in time is described by differential equations. The typical dynamical system is expressed as

$$\dot{x}(t) = F(x(t)) \tag{2.1}$$

where $x(t)$ is the state, an element of the state space $X \subset \mathbb{R}^n$ at $t \geq 0$, X is a compact state and $F : X \to \mathbb{R}^n$ is a continuously differentiable vector field on that state space [5].

The state space is the set of all coordinates that represents the system in a complete sense. The evolution rule predicts the next state or states based on the current state of the system. There are other variables or parameters which are fixed or unknown function of time that may depend upon the model. The dynamical systems can also be represented by the discrete time map as

$$\dot{x}(t+1) = f(x(t)),\ t \in \mathbb{Z} \tag{2.2}$$

where x belongs to the state space $X \subset \mathbb{R}^n$, t is the discrete time index and $f : X \to X$ is the dynamic map. We may need to make some additional assumptions on f, similar to the continuous-time system in (2.1). As the data acquired from dynamical systems are usually in discrete-time samples, this form is also more practical [5] [28] .

2.2 Koopman Operator Theory

Bernard Koopman's early work in 1931 [21] gave birth to the Koopman operator formalism. He developed the linear transformation currently known as the Koopman operator and discovered that it is unitary for Hamiltonian dynamical systems. The Koopman operator was mainly restricted to the study of measure-preserving systems for several decades following Koopman and Von Neumann's work. In 2009, Koopman modes were first used to a complicated fluid flow, particularly a jet in a cross flow; where the complex nonlinear flow was decomposed into Koopman modes, determined from spectral analysis of the Koopman operator [33]. This research demonstrated the potential of Koopman Decomposition in capturing dynamically important flow structures and their related time scales.

After this, Koopman decomposition and Dynamic mode decomposition (DMD) methods have not only been used in nonlinear flows, but also in different fields of study. For example, use of compressed DMD for background modeling [14], image analysis with the DMD in Convergent Path [35], reconstruction and classification of dynamics via Koopman features [40], analysis of traffic data [7], and fault detection of whole-buildings [17].

We interpret data in the context of dynamical systems as knowledge of some variable or variables that are relevant to the state of the system. The logical way to realize this into mathematical form is to assume that data is an evaluation of state functions. These functions are referred to as system observables. We consider an *uncontrolled* discrete-time dynamical system similar to (2.2) (where for simplicity we absorb the term t into the equation) as,

$$x^+ = f(x) \tag{2.3}$$

where $x \in \mathbb{R}^n$ is the state of the system, x^+ is the successor state and f is the transition mapping. Let $\mathcal{F}$ be an infinite dimensional function space invariant under the action of Koopman operator made up of all square-integrable real-valued functions with compact domain $X \subset \mathbb{R}^n$. The elements of this space $\mathcal{F}$ are called observables, and can be lifted into this function space $\mathcal{F}$. Let us consider a real valued observable of the space $\mathcal{F}$, $\psi : X \to \mathbb{R}$. Then, the Koopman operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ is defined by

$$(\mathcal{K}\psi)(x) = \psi(f(x)). \tag{2.4}$$

It describes the evolution of the observables along the trajectories of the system. Importantly, the Koopman operator $\mathcal{K}$ is linear even if the system (2.3) is nonlinear. Thus, the main benefit of lifting the dynamics using Koopman operator is that it offers a linear rule of evolution, however, the space of the observables is infinite dimensional. The linearity of the Koopman operator follows from the linearity of the composition operation, i.e.,

$$\mathcal{K}(\psi_1 + \psi_2)(x) = (\psi_1 + \psi_2)(f(x)) = \psi_1(f(x)) + \psi_2(f(x)) = \mathcal{K}\psi_1(x) + \mathcal{K}\psi_2(x) \tag{2.5}$$

where ψ_1 and ψ_2 are any two observables. If the set of observables $\mathcal{F}$ contains the components of the state x_i, i.e. $x \mapsto x_i$ where $i \in \{1, ..., n\}$, this operator completely captures the features of the underlying dynamical system [23]. Figure (2.1) shows a schematic depiction of the Koopman operator [5].

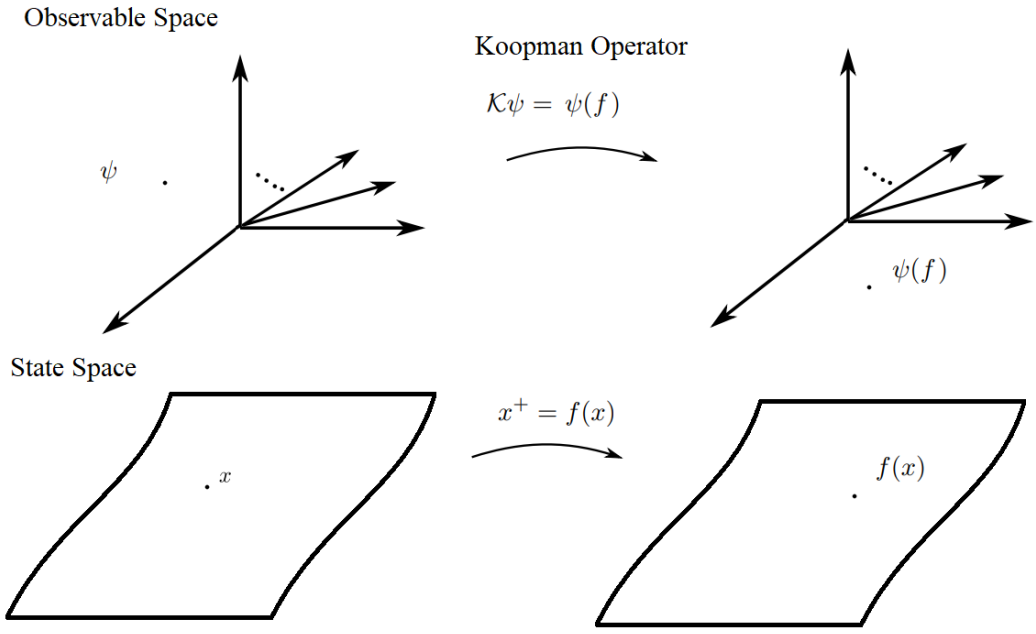


Figure 2.1: The Koopman operator lifts the dynamics from state space to the observable space, where the dynamics is linear but infinite-dimensional. Figure from author, adapted from [23].

2.3 Linear predictors for nonlinear dynamical systems

Let us consider a variation of the system (2.3) which a discrete-time nonlinear controlled dynamical system

$$x^+ = f(x, u), \tag{2.6}$$

where $x \in \mathbb{R}^n$ is the state of the system, x^+ is the successor state and f is the transition mapping and $u \in \mathcal{U} \subset \mathbb{R}^m$ control input. There are n states and m control inputs [23].

This study focuses on predicting the trajectory of (2.6) given an initial state x_0 and control inputs u_0, u_1, and so on. We are specifically seeking for simple predictors with a linear structure that may be used in linear control design techniques (like MPC). These linear predictors are used in nonlinear feedback control and estimation. The use of linear predictor is mature and well understood. Unlike nonlinear methods, the main advantage of this method is that there is fast computation i.e. linear algebra or convex optimization methods, because of which it is easy to rapidly deploy in applications [23].

Let us consider the predictors, which are in the form of controlled linear dynamical

systems, as,

$$\begin{aligned} z^+ &= Az + Bu, \\ \hat{x} &= Cz \end{aligned} \tag{2.7}$$

where $z \in \mathbb{R}^N$, N is the number of lifting functions with (typically) $N \gg n$, $\hat{x}$ is the prediction of x, $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times m}$ and $C \in \mathbb{R}^{n\times N}$. Let us consider an initial condition of the above predictor (2.7) as

$$z_0 = \boldsymbol{\psi}(x_0) := \begin{bmatrix} \psi_1(x_0) \\ \vdots \\ \psi_N(x_0) \end{bmatrix}, \tag{2.8}$$

where x_0 is the initial condition on (2.6), and $\psi_i : \mathbb{R}^N \to \mathbb{R}$ with $i = \{1, ..., N\}$ are the lifting functions which are usually nonlinear and defined by the user. Lifting means raising of the dynamical system's output, rather than the state itself. After applying the lifting function on the states, we get a lifted state z which is N dimensional. The most important thing to notice is that the control inputs u are not lifted at all, and thus linear constraints can be applied to them linearly. Also, because the projected state $\hat{x}$ is a linear function of the lifted state $\hat{z}$ in (2.8), linear restrictions may be easily put on the state [23].

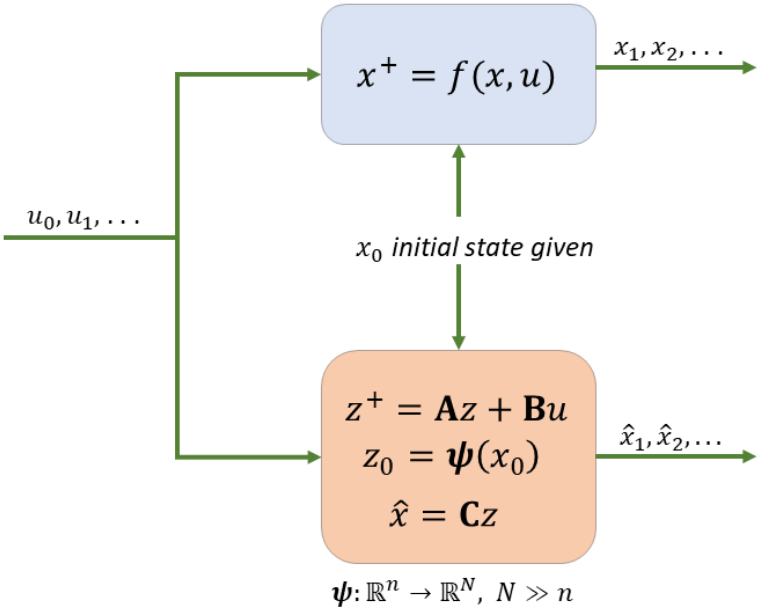


Figure 2.2: Linear predictor, lifting and output. Figure from author, adapted from [22].

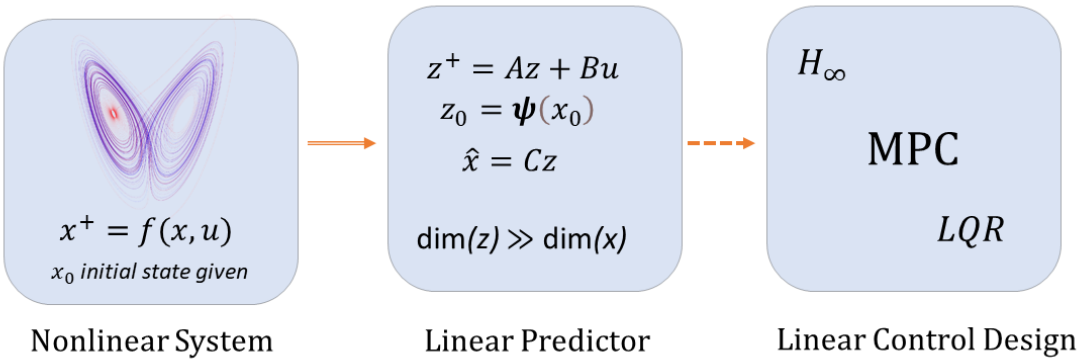


Figure 2.3: Linear predictor for a nonlinear controlled dynamical system used for Linear Control Design, in our case MPC. Figure from author, adapted from [23].

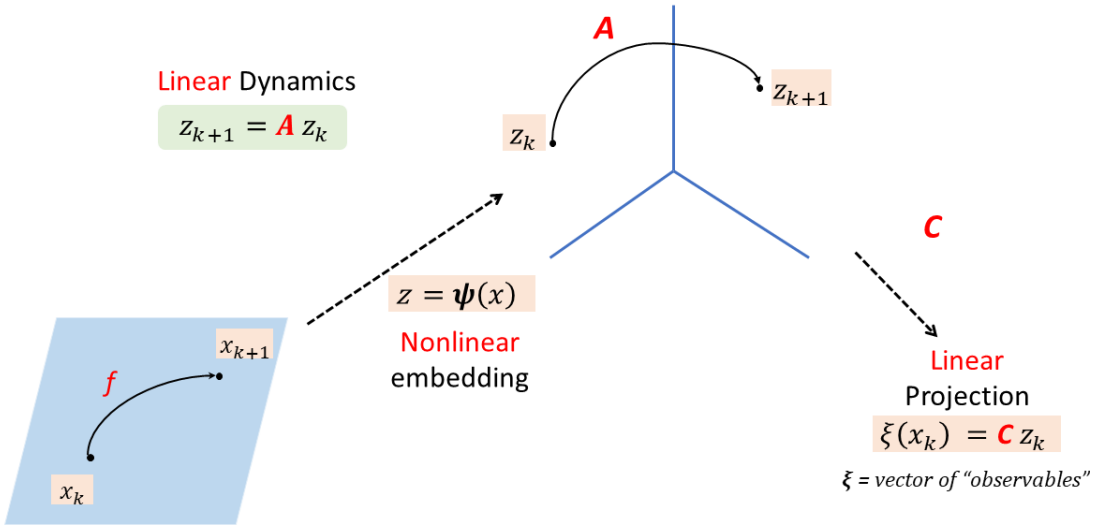


Figure 2.4: Nonlinear Embeddings. Figure from author, adapted from [22].

This type of predictor lends itself well to linear feedback control design approaches. Importantly, the resultant feedback controller will be nonlinear in the original state x, even though it may be linear in the lifted state z (though it is not needed). The feedback controller for (2.6) is given by $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, while the feedback controller for (2.7) is given by $\kappa_{lift} : \mathbb{R}^N \to \mathbb{R}^m$. These two controller are related to each other by

$$\kappa(x) := \kappa_{lift}(\psi(x)). \tag{2.9}$$

The notion is that if for each acceptable input sequence, the real trajectory of x created by

(2.6) and the anticipated trajectory of $\hat{x}$ generated by (2.7) are close, then the best controller for (2.7) should be close to the optimal controller for (2.6). For the prediction $\hat{x}$ embeddings should be ψ [23].

2.4 State-Space Representation of linear time-invariant systems

The state of a dynamical system refers to a minimum set of variables x_i known as state variables, where $i = \{1 \dots n\}$ with n being the order of the system, that completely define the system and its reaction to any given set of inputs. The dynamical behaviour of any state dependant system is governed by these n dimensional states [32].

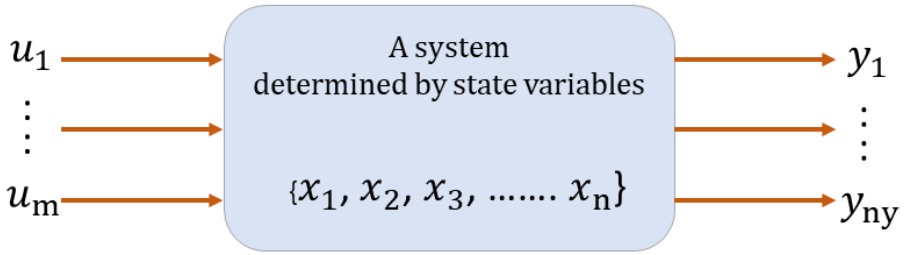


Figure 2.5: System with states, inputs and outputs. Figure from author, adapted from [32].

The standard mathematical expression is given by a set of n first-order ordinary differential state equations, where the time derivatives of the state variables are dependent on their n state variables $\{x_1,\ x_2,\ \dots\ x_n\}$ and m control inputs $\{u_1,\ u_2,\ \dots\ u_m\}$.

$$\begin{aligned} x_1^+ &= f_1(\boldsymbol{x}, \boldsymbol{u}) \\ x_2^+ &= f_2(\boldsymbol{x}, \boldsymbol{u}) \\ &\vdots \qquad\quad \vdots \\ x_n^+ &= f_n(\boldsymbol{x}, \boldsymbol{u}). \end{aligned} \tag{2.10}$$

In the above general form equations, $\boldsymbol{x}$ is a collection of n state equations called state vector, and $\boldsymbol{u}$ is a collection of m control inputs called control vector. The function $f_i(\boldsymbol{x}, \boldsymbol{u})$ where $i = \{1, 2, \dots n\}$ is a general nonlinear, time varying function of the state vectors, the system input vectors, and time. For simplicity purposes, we have absorbed time in the equation itself. In vector notation, the (2.10) can be written as:

$$\boldsymbol{x}^+ = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \tag{2.11}$$

For a linear time-invariant system of dimension n and m inputs, (2.10) is transformed into

a set of n coupled first-order linear differential equations with constant coefficients as

$$\begin{aligned} x_1^+ &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_{11}u_1 + \ldots + b_{1m}u_m, \\ x_2^+ &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_{21}u_1 + \ldots + b_{2m}u_m, \\ &\vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\ x_n^+ &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_{n1}u_1 + \ldots + b_{nm}u_m, \end{aligned} \tag{2.12}$$

where each derivative of the state is the linear combination of the state variables and control inputs weighted with constants a_{ij} and b_{ij}. These constants describe the system. The (2.12) can be expressed in matrix form as

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & \ldots & b_{1m} \\ b_{21} & \ldots & b_{2m} \\ \vdots & & \vdots \\ b_{n1} & \ldots & b_{nm} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}. \tag{2.13}$$

Finally, above equation (2.13) can be written as a matrix-vector form as,

$$\boldsymbol{x}^+ = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.14}$$

where $\boldsymbol{A}$ is a $n \times n$ matrix of constants a_{ij} that weighs the states, $\boldsymbol{x}$ is a n column vector of state variables, $\boldsymbol{B}$ is a $n \times m$ matrix of constants b_{ij} that weights the inputs, and $\boldsymbol{u}$ is a m column vector of control input [32].

The output of the system is defined as all variables that are of relevance or interest. These system outputs can also be written as a linear combination of the state variables and control units, along with some constants c_i and d_i as,

$$y = c_1x_1 + c_2x_2 + \ldots + c_nx_n + d_1u_1 + \ldots + d_mu_m. \tag{2.15}$$

If n_y dimension of outputs are chosen, then there will be n_y such equations which can be

shown as,

$$\begin{aligned} y_1 &= c_{11}x_1 + c_{12}x_2 + \ldots + c_{1n}x_n + d_{11}u_1 + \ldots + d_{1m}u_m, \\ y_2 &= c_{21}x_1 + c_{22}x_2 + \ldots + c_{2n}x_n + d_{21}u_1 + \ldots + d_{2m}u_m, \\ &\vdots \qquad\quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\quad \vdots \\ y_{n_y} &= c_{n_y1}x_1 + c_{n_y2}x_2 + \ldots + c_{n_yn}x_n + d_{n_y1}u_1 + \ldots + d_{n_ym}u_m, \end{aligned} \tag{2.16}$$

and the compact matrix form of the equations can be written as,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_y} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \vdots & & & \vdots \\ c_{n_y1} & c_{n_y2} & \ldots & c_{n_yn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} d_{11} & \ldots & d_{1m} \\ d_{21} & \ldots & d_{2m} \\ \vdots & & \vdots \\ d_{n_y1} & \ldots & d_{n_ym} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}. \tag{2.17}$$

The matrix form in the (2.17) can be written in compact vector form,

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{2.18}$$

where $\boldsymbol{C}$ is a $n_y \times n$ matrix of constants c_{ij} that weighs the states, $\boldsymbol{x}$ is a n column vector of state variables, $\boldsymbol{D}$ is a $n_y \times m$ matrix of constants d_{ij} that weighs the inputs, and $\boldsymbol{u}$ is a m column vector of control input. Generally $\boldsymbol{D}$ is taken as a null matrix, and hence the (2.18) can be written as

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}\,. \tag{2.19}$$

For a linear time-invariant system, a complete system model consists of n state equations and m control outputs defined in terms of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The n_y outputs of the system is expressed in terms of $\boldsymbol{C}$ and $\boldsymbol{D}$. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are system attributes that are dictated by the structure and constituents of the system. The output equation matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ are defined by the specific selection of output variables. In summary, for the state equation based modelling process, the main goal is to determine the constituents of the matrices and express the system model in the form as presented below [32].

$$\begin{aligned} \boldsymbol{x}^+ &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{aligned} \tag{2.20}$$

2.5 Koopman Operator for controlled system

For the controlled dynamical system defined in (2.6), the Koopman operator associated with it is defined as the Koopman operator associated with the uncontrolled dynamical

system developing on the extended state-space. This extended state-space is the product of the original state-space and the space of all control sequences. $\ell(\mathcal{U})$ is the space of all control sequences with elements being $\boldsymbol{u} := (u)_{i=0}^{\infty}$ and $\boldsymbol{u} := u_i \in \mathcal{U}$. As explained in [23] extended (infinite-dimensional) state space which is in $\mathbb{R}^n \times \ell(\mathcal{U})$ is given by

$$\mathcal{X} := \begin{bmatrix} x \\ \boldsymbol{u} \end{bmatrix}. \tag{2.21}$$

This dynamics of the extended state (2.21) is described by

$$\mathcal{X}^+ = \mathcal{F}(\mathcal{X}) := \begin{bmatrix} f(x, \boldsymbol{u}(0)) \\ \boldsymbol{\mathcal{S}u} \end{bmatrix}. \tag{2.22}$$

where $\boldsymbol{\mathcal{S}}$ is a shift operator given by $\boldsymbol{\mathcal{S}u}(i) = \boldsymbol{u}(i+1)$ and $\boldsymbol{u}(i)$ is the i^{th} element of the control sequence $\boldsymbol{u}$.

The Koopman operator $\mathcal{K} : \mathcal{H} \to \mathcal{H}$ associated with (2.22) is defined as

$$(\mathcal{K}\phi)\mathcal{X} = \phi(F(\mathcal{X})) \tag{2.23}$$

where $\phi : \mathbb{R}^n \times \ell(\mathcal{U}) \to \mathbb{R}$ belongs to some space of observables $\mathcal{H}$. The Koopman operator's formulation implicitly requires that $\mathcal{H}$ is invariant under the action of $\mathcal{K}$, and hence $\mathcal{H}$ must contain functions that are dependent on $\boldsymbol{u}$ in the controlled environment [23].

2.6 EDMD for controlled system

Extended Dynamic Mode Decomposition (EDMD) is a data driven method that approximates the Koopman operator and hence the Koopman eigenvalue, eigenfunction, and mode tuples. Because EDMD is a data-driven process, it may be used to analyze data from stochastic systems without requiring any algorithmic modifications. In our case, for the time domain prediction of trajectories produced by (2.6), We build a finite-dimensional approximation of the operator $\mathcal{K}$ that produces a predictor of the form (2.7). EDMD requires a) vector of lifting functions $\boldsymbol{\phi}(\chi)$, as well as (b) a data set of snapshot pairs, $(\chi_j, \chi_j^+)_{j=1}^K$ which we express as two data sets, χ_j and χ_j^+ where $\chi_j^+ = F(\chi_j)$ [38]. K is the number of trajectories. The goal of this approach is to find a matrix $\mathcal{A}$ that is the transpose of the finite-dimensional approximation of $\mathcal{K}$ that minimizes equation

$$\sum_{j=1}^{K} \|\boldsymbol{\phi}(\chi_j^+)) - \mathcal{A}\boldsymbol{\phi}(\chi_j))\|_2^2 \tag{2.24}$$

where

$$\boldsymbol{\phi}(\chi) = [\phi_1(\chi), \ldots, \phi_{N_\phi}(\chi)]^T \tag{2.25}$$

is a dictionary of observables or vectors of lifting functions with $\phi_i : \mathbb{R}^n \times \ell(\mathcal{U}) \to \mathbb{R}, i \in \{1, \ldots, N_\phi\}$. Since the extended state $\chi = (x, \boldsymbol{u})$ is an infinite-dimensional object in general, the goal (2.25) cannot be assessed in a limited amount of time unless the ϕ_i's are chosen in a unique fashion [23].

2.6.1 Linear predictors

We require that the functions ϕ_i be of the form

$$\phi_i(x, \boldsymbol{u}) = \psi_i(x) + \mathcal{L}_i(\boldsymbol{u}) \tag{2.26}$$

in order to construct a linear predictor (2.7) and a computable objective function in (2.26). In this case, $\psi_i : \mathbb{R}^n \to \mathbb{R}$ is nonlinear in general while $\mathcal{L}_i : \ell(\mathcal{U}) \to \mathbb{R}$ is linear. The vector of lifting functions $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_{N_\phi}]^T$ (where $N_\phi = N + m$ without loss of generality) is in the form

$$\boldsymbol{\phi}(x, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{\psi}(x) \\ \boldsymbol{u}(0) \end{bmatrix}, \tag{2.27}$$

where $\boldsymbol{\psi} = [\psi, \ldots, \psi_N]^T$ and $\boldsymbol{u}(0) \in \mathbb{R}^m$ is the initial component of the control sequence $\boldsymbol{u}$. We may ignore the final m components of each term $\boldsymbol{\phi}(\chi_j^+)) - \mathcal{A}\boldsymbol{\phi}(\chi_j)$ in (2.24) since we are not interested in forecasting future values of control sequences. N rows of $\mathcal{A}$ in (2.24) should taken into a new matrix $\hat{\mathcal{A}}$, and it should be decomposed as $\hat{\mathcal{A}} = [A, B]$ where $A \in \mathbb{R}^{N \times N}$ and $B \in \mathbb{R}^{N \times m}$. This leads to a minimization problem

$$\min_{A,B} \sum_{j=1}^{K} \|\boldsymbol{\psi}(x_j^+)) - A\boldsymbol{\psi}(x_j) - B\boldsymbol{u}_j(0))\|_2^2. \tag{2.28}$$

After minimizing (2.28) over A and B, we get the predictor of the form (2.8). The matrix C is obtained by

$$\min_{C} \sum_{j=1}^{K} \|x_j - C\boldsymbol{\psi}(x_j)\|_2^2. \tag{2.29}$$

The equations (2.28) and (2.29) are linear least squares problem and can be solved using linear algebra techniques. The main reason to use linear predictor is that their concepts are mature and well understood. It is fast to compute such predictors and can be rapidly deployed in applications [23].

Remark 2.1. *The solution to* (2.29) *is easy if the set of lifting functions $\psi_1, \ldots, \psi_N$ comprises the state observable, i.e., $\psi_i(x) = x_i$ for all $i \in \{1, \ldots, n\}$ after every conceivable reordering. In this situation, the answer to* (2.29) *is $C = [I, 0]$, where I is the n-dimensional identity matrix [23].*

2.6.2 Predictor Construction

The predictor of type (2.8) can be constructed by using the measured data. Let us assume a set of data of the form

$$\begin{aligned} \mathbf{X} &= [x_1, \ldots, x_K], \\ \mathbf{Y} &= [x_1^+, \ldots, x_K^+], \\ \mathbf{U} &= [u_1, \ldots, u_K] \end{aligned} \tag{2.30}$$

where K is the number of trajectory, each x_i is the collection of states until simulation length N_{sims}, f being the dynamics discretized with sampling period T_s statisfying $x_i^+ = f(x_i, u_i)$. (x_i, x_i^+) is a pair of consecutive state measurements generated by continuous time dynamics with the control input u_i maintained constant throughout the sampling period T_s. The data can be from the real system or artificially collected from a model [23].

The matrices $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{N\times m}$ of the predictor (2.8) are obtained from the solution to the minimization problem

$$\min_{A,B} \|\mathbf{Y}_{\text{lift}} - A\mathbf{X}_{\text{lift}} - B\mathbf{U})\|_F. \tag{2.31}$$

where $\|.\|_F$ is the Frobenius norm of a matrix and

$$\begin{aligned} \mathbf{X}_{\text{lift}} &= [\boldsymbol{\psi}(x_1), \ldots, \boldsymbol{\psi}(x_K)], \\ \mathbf{Y}_{\text{lift}} &= [\boldsymbol{\psi}(x_1^+), \ldots, \boldsymbol{\psi}(x_K^+)] \end{aligned} \tag{2.32}$$

with

$$\boldsymbol{\psi}(x) := \begin{bmatrix} \psi_1(x) \\ \vdots \\ \psi_N(x) \end{bmatrix}, \tag{2.33}$$

is the vector of lifting functions. The matrix $C \in \mathbb{R}^{n\times N}$ is obtained as

$$\min_{C} \|\mathbf{X} - C\mathbf{X}_{\text{lift}}\|_F. \tag{2.34}$$

The analytical solution to (2.31) is given by

$$[A, B] = \mathbf{Y}_{\text{lift}}[\mathbf{X}_{\text{lift}}, \mathbf{U}]^\dagger \tag{2.35}$$

where † denotes the Moore-Penrose pseudoinverse of a matrix. The analytical solution to (2.34) is given by

$$C = \mathbf{X}\mathbf{X}_{\text{lift}}^\dagger \tag{2.36}$$

For the practical considerations, the analytical solutions (2.35) and (2.36) are not the preferred way to generate A, B and C matrices. As explained in [23], for larger datasets with $K \gg N$, it is better to solve normal equation. The normal equations is given by

$$\mathbf{V} = \mathcal{M}\mathbf{G}. \tag{2.37}$$

If we compare (2.35) to the normal form in (2.37), we get normal equations associated to it. In (2.37) variable is $\mathcal{M} = [A, B]$ and data are

$$\begin{aligned} \mathbf{G} &= \begin{bmatrix} \mathbf{X}_{\text{lift}} \\ \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\text{lift}} \\ \mathbf{U} \end{bmatrix}^T, \\ \mathbf{V} &= \mathbf{Y}_{\text{lift}} \begin{bmatrix} \mathbf{X}_{\text{lift}} \\ \mathbf{U} \end{bmatrix}^T. \end{aligned} \tag{2.38}$$

Hence the solution to (2.37) is the solution to (2.35). The size of the matrix $\mathbf{G}$ is $(N + m) \times (N+m)$ and $\mathbf{V}$ is $N \times (N+m)$. This shows that the sizes of matrix in normal equation is independent of the number of samples K of the data set [23] [25].

2.6.3 The Choice of the Lifting functions

The accuracy and rate of convergence of EDMD, like all spectral techniques, is determined by the choice of lifting functions which cover the subspace of observables. Some common choice for the lifting functions are polynomials, fourier modes or radial basis functions (RBF), but the best basis function relies on both the underlying dynamical system and the sampling approach used to gather the data. In principle, any of these sets may be a good fit as a lifting function, albeit infinite domains require some caution to guarantee that any required inner products converge [38].

Due to the non-linear nature of the data, many different RBFs could be used. Among different RBFs, thin plate splines are a particularly useful since they do not require the scaling parameter like other RBFs such as Gaussians. The centers should also be determined around which the RBFs are established. The common way of selecting those centers is to use k-means clustering on the combined data set with a pre-specified value of k [38].

We know that $\mathbf{X}$ is matrix which a collection of trajectory of the state. Let C be RBFs centers which are either selected from the data itself or by any clustering methods. Then the sum of squared difference between the data and centers is defined as r_s i.e. $r_s = sum((X - C)^2)$. Let ε be the kernel width for Gaussian type RBFs and p_k be polyharmonic coefficient for polyharmonic RBFs. Then the different RBFs that can be used are given in the table as

Table 2.1: Different choices of RBFs.

Thin plate	$r_s \cdot log(\sqrt{r_s})$
Gauss	$e^{-\varepsilon^2 \cdot r_s}$
Inverse quadratic	$\frac{1}{1+e^2 \cdot r_s}$
Inverse multi quadratic	$\frac{1}{\sqrt{1+e^2 \cdot r_s}}$
Polyharmonic	$r_s^{p_k/2} \cdot log(\sqrt{r_s})$

2.7 Model Predictive Control

Model Predictive Control (MPC), also called receding horizon control, is a one of the effective advanced control approach that is used to solve complex multivariable constrained control issues. By minimizing an objective function, MPC uses the process model to derive the control signal and generates the control sequence that minimizes the objective function using a model to forecast the process output at future time instants (horizons) [11]. The main idea behind MPC is to forecast the future behavior of the controlled system over a specified time horizon and calculate an optimal control input that minimizes an a priori established cost functional while assuring fulfilment of given system constraints. MPC has been employed in a wide range of applications, such as control of planetary rovers [9], Autonomous Ground Vehicles [29], smart agriculture [13], optimal discontinuous drug delivery [20], robot manipulator [30], and Mobile Medical Robot [19]. Generally, out of two types of MPC, linear and nonlinear, linear MPC solves a convex quadratic problem, allowing for exceptionally quick control input assessment. However, nonlinear MPC addresses a challenging non-convex optimization problem, needing significantly more processing resources and/or depending solely on local solutions.

The process model's output is anticipated in MPC at time instant t in the future. The future control signals $u(t+k|t)$ are used to forecast the output values $y(t+k|t)$ based on the previous input and output values t and the future control signals $u(t+k|t)$. These future control signals are calculated by maximizing particular criteria in relation to the $w(t+k)$ reference trajectory. The error computation between the projected output signal and the predicted reference trajectory might be this criterion. The control signal $u(t)$ is given to process after the error computation and optimization, whereas the subsequent control signals are discarded since the output value $y(t+1)$ is already known at the next sampling moment. With the new value, the process continues, and all sequences are up to date. To put it in another way, the model uses historical inputs and outputs, as well as future inputs, to anticipate outputs. The future errors are calculated by comparing the expected outputs to the reference trajectory. Future errors, as well as the cost function and constraints, are taken as to an optimizer, which optimizes the control signals before returning them to the model. With the new computed values, the cycle continues, and

all sequences are changed correspondingly. Figure (2.6) depicts the MPC's fundamental structure. The prediction model, objective function, and finding the control law are the three key aspects of the MPC method, which may be chosen individually depending on the target issue to be addressed [11] [3].

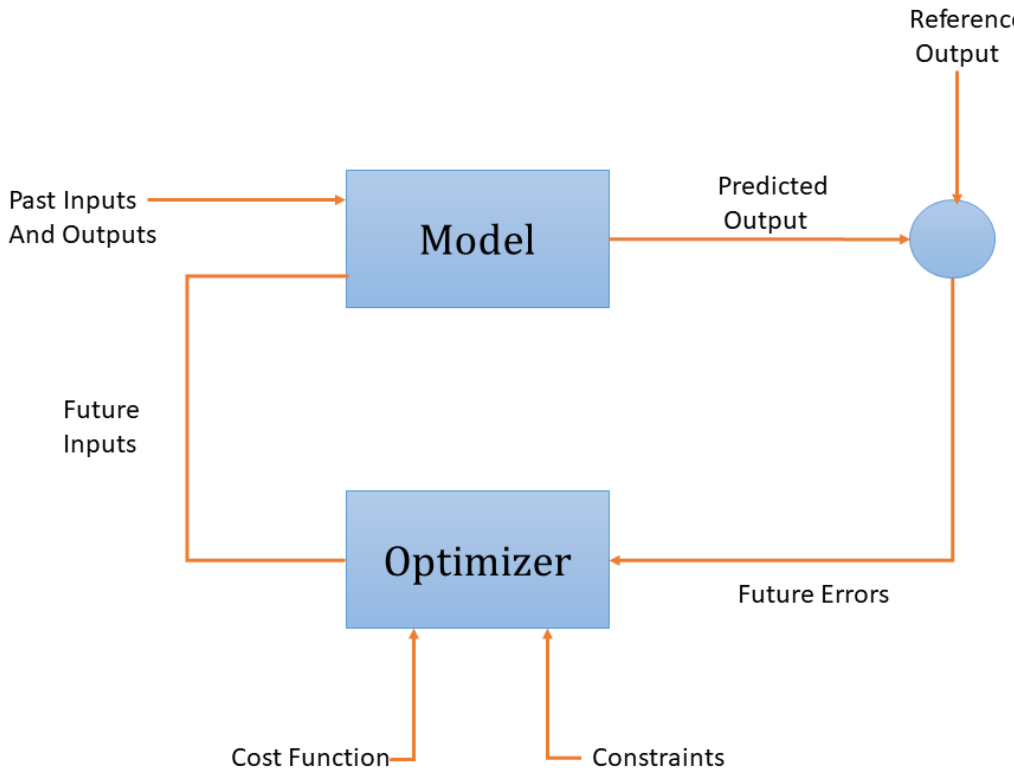


Figure 2.6: Structure of MPC. Figure from author, adapted from [3].

Consider a system at time k which has a reference trajectory that must be followed for a specified horizon p to see how MPC works. MPC uses the system's current states as input and generates simulation of various control inputs from time k to $k + p$. MPC chooses the optimum set of inputs from a variety of options to minimize the cost function. MPC then implements just one of these projected control inputs and restarts the cycle at time $k + 1$. MPC is also known as receding horizon control because iterative cycles across the horizon take one step at a time. The receding control for provided simulation is illustrated in figure (2.7) [3].

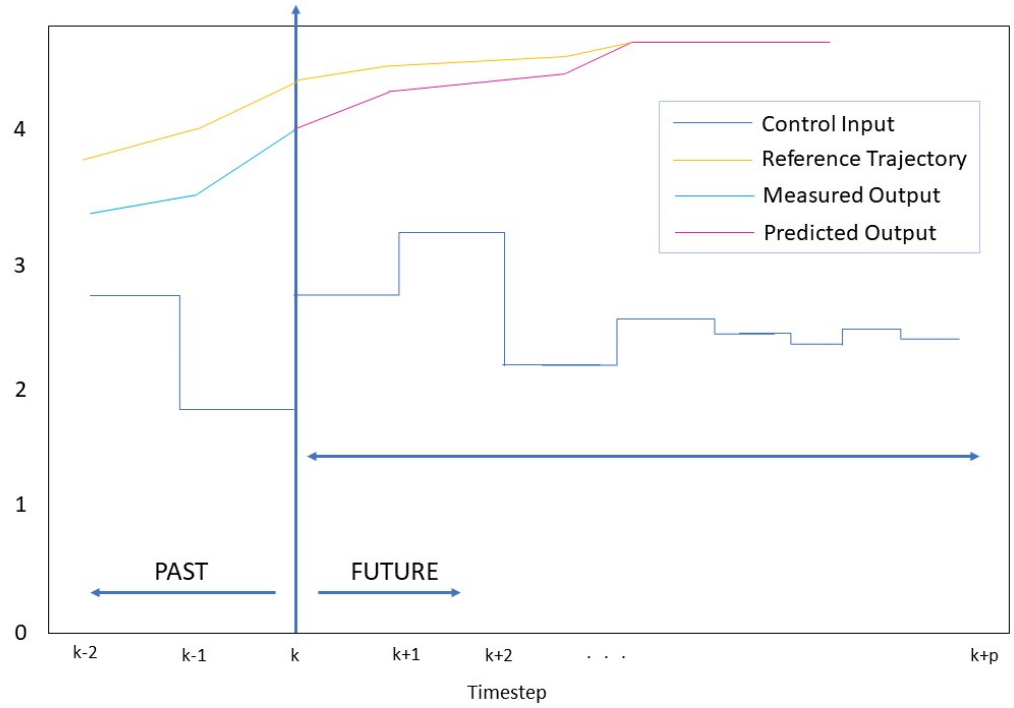


Figure 2.7: Receding horizon of MPC. Figure from author, adapted from [3].

MPC is a simple and straightforward formulation that is based on well-known ideas. It handles constraints directly and makes explicit use of a model. It has well-understood tuning parameters like prediction horizon and optimization problem setup. The development time is significantly reduced compared to competing sophisticated control technologies. It is easier to maintain, in the sense that altering the model or specifications does not necessitate a total redesign, and may occasionally be done on the fly. In conclusion, MPC is a multivariable control technique that incorporates the following elements [11]:

- a dynamic internal representation of the process
- a cost function J over the receding horizon
- Using the control input u, an optimization procedure is used to minimize the cost function J.

2.7.1 Koopman MPC

The Koopman MPC is the lifting-based MPC for the nonlinear systems. Just like the linear MPC, it is based on convex quadratic programming, which allows for extraordinarily quick evaluation of control inputs. The Koopman MPC controller is distinguished by the predictor, which takes the form of a linear dynamical system evolving on an embedded

(or lifted) state space with a larger dimension than the original state space. This predictor is valid worldwide (or in a significant subset of the state space) and fixed once and for all, unlike traditional local linearization strategies [25].

The general form of the optimization problem of Koopman MPC, where the controller solves at each time instance k of the closed-loop operation, is given by,

$$\begin{aligned} \underset{u_i, z_i}{\text{minimize}} \quad & J((u_i)_{i=0}^{N_p-1}, (z_i)_{i=0}^{N_p}) \\ \text{subject to} \quad & z_{i+1} = Az_i + Bu_i, \ i = 0, \ldots, N_p - 1 \\ & E_i z_i + F_i u_i \leq b_i, \ i = 0, \ldots, N_p - 1 \\ & E_{N_p} z_{N_p} \leq b_{N_p}, \\ \text{parameter} \quad & z_0 = \boldsymbol{\psi}(x_k), \end{aligned} \tag{2.39}$$

where N_p is the prediction horizon and J is the convex quadratic cost function given by $J((u_i)_{i=0}^{N_p-1}, (z_i)_{i=0}^{N_p}) = z_{N_p}^T Q_{N_p} z_{N_p} + q_{N_p}^T z_{N_p} + \sum_{i=0}^{N_p-1} z_i^T Q_i z_i + u_i^T R_i u_i + q_i^T z_i + r_i^T u_i$ where $Q_i \in \mathbb{R}^{N \times N}$ and $R_i \in \mathbb{R}^{m \times m}$ are positive semidefinite. State and input polyhedral constraints are defined by the matrices $F_i \in \mathbb{R}^{n_c \times m}$ and $E_i \in \mathbb{R}^{n_c \times N}$, and the vector $b_i \in \mathbb{R}^{n_c}$. The current state of the nonlinear dynamical system x_k is used to parametrize the optimization problem (2.39) [23].

A feedback controller

$$\kappa(x_k) = u_0^*(x_k) \tag{2.40}$$

is defined by this optimization problem. The optimal solution to problem (2.39) parametrized by the present state x_k is denoted by $u_0^*(x_k)$. The predictions are initialized from the lifted state $\boldsymbol{\psi}(x_k)$ at each time step k. By adding these nonlinear functions among the lifting functions ϕ_i, nonlinear functions of the initial state x can be penalized in the cost function and included among the constraints [23].

The figure 2.8 depicts the overall design of Koopman MPC. The design consists of two main parts: the MPC controller and the system. The system provides the state to the controller, while the controller after minimizing the cost returns the optimal control to the system. The MPC controller consists of the optimizer and the predictor. The optimizer handles the minimization of cost function whereas the predictor generates the lifted state.

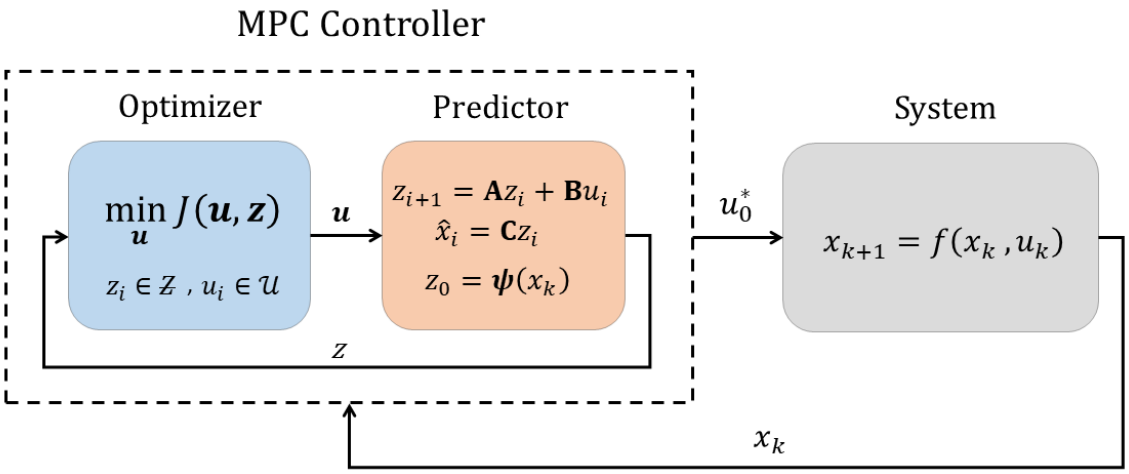


Figure 2.8: Conceptual depiction of the Koopman MPC. Figure from author, adapted from [25].

2.7.2 Nonlinear MPC to Koopman MPC

Let us assume a nonlinear MPC problem that solves the optimization problem at each time step k of the closed-loop procedure is given by

$$\begin{aligned} \underset{u_i, \bar{x}_i}{\text{minimize}} \quad & l_{N_p}(x_{N_p}) + \sum_{i=0}^{N_p-1} l_i(\bar{x}_i) + u_i^T \bar{R}_i u_i + \bar{r}_i^T u_i \\ \text{subject to} \quad & \bar{x}_{i+1} = f(\bar{x}_i, u_i), \ i = 0, \ldots, N_p - 1 \\ & c_{x_i}(\bar{x}_i) + c_{u_i}^T u_i \leq 0, \ i = 0, \ldots, N_p - 1 \\ & c_{xN_p}(\bar{x}_{N_p}) \leq 0, \\ \text{parameter} \quad & \bar{x}_0 = x_k, \end{aligned} \tag{2.41}$$

where x is the true measured state, $\bar{x}$ is the predicted state, the true nonlinear dynamics $x^+ = f(x, u)$ is one of the constraints of (2.41) along with functions l_i and c_i which can be nonlinear too. The optimization problem (2.41) is normally a non-convex optimization problem which is complex and extremely hard to solve [23].

Translating or transforming (2.39) into (2.41) means to find an approximation of the one function to the other. The representation of a dynamical system into equivalent form of the function as the lifted linear predictor is not the same as having a linear system (unless the dynamical system is linear). The first constraint of (2.41) is a dynamical system that is translated by assuming the predictor of the form (2.7) with matrices A and B. These data matrices A and B can be generated by following the steps outlined in subsection (2.6.2). The initialization $z_0 = \psi(x_k)$ can be done using the lifting mapping

$\boldsymbol{\psi} = [\psi_1, \ldots, \psi_N]^T$. Without loss of generality, we can assume that $\psi_i(x) = l_i(x), i = \{0, \ldots N_p\}$ and $\psi_{N_p+i}(x) = c_{x_i}(x), i = \{0, \ldots, N_p\}$. Thus the remaining data is given by $Q_i = 0, R_i = \bar{R}_i, r = \bar{r}_i, q_i = [0_{1\times i}, 0_{1\times N-1-i}], E_i = [0_{1\times N_p+i}, 0_{1\times N-N_p-1-i}] F_i = c_{u_i}^T, b_i = 0$ where $0_{i\times j}$ is the matrix of zeros of size $i \times j$. The constraint functions c_{x_i} and c_{u_i} are scalar-valued in this derivation; for vector-valued constraint functions, the technique is analogous, with the lifting functions ψ_i equal to the individual components of the constraint functions c_{x_i}. Thus this canonical method always results in a linear cost function (2.39) [23].

2.7.3 Dense form MPC

The unique structure of MPC can be used by tailored algorithms to render computation cost independent of the size of the embedding. The minimization in (2.39) should apply to both u_i's and z_i's. However, because z_0 and u_0, …, u_{N-1} determine z_0, …, z_N via $z_{i+1} = Az_i + Bu_i$, the z_i's can be removed, resulting in the so-called dense version of MPC. This removes the dependency on z_i's, and therefore on the dimension of the embedded state z. The dense form MPC is given by

$$\begin{aligned} \underset{U\in\mathbb{R}^{mN_p}}{\text{minimize}} \quad & U^T H U^T + h^T U + z_0^T G H \\ \text{subject to} \quad & LU + Mz_0 \leq c \\ \text{parameter} \quad & z_0 = \boldsymbol{\psi}(x_k). \end{aligned} \tag{2.42}$$

The the data matrices of the problem (2.42) are,

$$\begin{aligned} H = \mathbf{R} + \mathbf{B}^T\mathbf{Q}\mathbf{B}, \quad h = \mathbf{B}^T\mathbf{q} + \mathbf{r}, \quad G = 2\mathbf{A}^T\mathbf{Q}\mathbf{B} \\ L = \mathbf{F} + \mathbf{E}\mathbf{B}, \qquad M = \mathbf{E}\mathbf{A}, \qquad c^T = [b^T, b^T, \ldots, b^T] \end{aligned} \tag{2.43}$$

where

$$\begin{aligned} \mathbf{A} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \ddots & \ddots & \\ A^{N_p-1}B & \ldots & AB & B \end{bmatrix}, \\ \mathbf{Q} = I_{N_p+1} \otimes Q, \qquad \mathbf{R} = I_{N_p} \otimes R, \\ \mathbf{E} = I_{N_p+1} \otimes E, \qquad \mathbf{F} = \begin{bmatrix} I_{N_p} \otimes F \\ 0_{2(N+m)\times mN_p} \end{bmatrix}, \\ F = \begin{bmatrix} 0_{2N\times m} \\ I_m \\ -I_m \end{bmatrix}, \qquad E = \begin{bmatrix} I_N \\ -I_N \\ 0_{2m\times N} \end{bmatrix}, \qquad b = \begin{bmatrix} z_{max} \\ -z_{min} \\ u_{max} \\ -u_{min} \end{bmatrix}. \end{aligned} \tag{2.44}$$

for some positive-semidefinite matrix $H \in \mathbb{R}^{mN_p \times mN_p}$, and for some matrices and vectors, $h \in \mathbb{R}^{mN_p}$, $G \in \mathbb{R}^{N_p \times mN_p}$, $L \in \mathbb{R}^{n_c N_p \times mN_p}$, and $M \in \mathbb{R}^{n_c N_p \times N}$. Before deploying the controller, these matrices are fixed and precomputed offline. The optimization is performed on the predicted control inputs vector $U = [u_0^T, u_1^T, \ldots, u_{N_p-1}^T]$. I_N is the N-fold block-diagonalization operator, with $\otimes$ representing the Kronecker product. This form is especially well-suited for the active set solver qpOASES [15], as it allows for quick warm-up. Since, Hessian H is independent of the size of the lift N, once the nonlinear mapping $z_0 = \psi(x_k)$ is generated, the cost of solving (2.42) is linear [23] [25].

The algorithmic summary of the lifting based MPC is given below.

Algorithm 1 Koopman MPC - closed-loop operation [23]

Require: Predictor (A, B), Cost matrices (Q, R), bounds $(u_{max}, u_{min}, z_{max}, z_{min})$.

1: **for** $k = 0, 1 \ldots$ **do**
2: Measure x_k on the real system
3: Set $z_0 := \psi(x_k)$
4: Solve to get an optimal solution $U^* = (u_0^*, \ldots, u_{N_p-1}^*)$
5: Set $u_k = U_{1:m}^*$
6: Apply $u_k := u_0^*$ to real system
7: **end for**

2.8 Pendulum dynamics

Consider the experimental setup of the inverted pendulum on a cart as shown in figure (2.9). This setup will be utilized on this study for Koopman MPC. A pendulum and a moving cart are connected by a swivel, which allows the pendulum to freely rotate. The cart wheels spin on a rail, and the entire system is powered by a DC motor. The displacement of the cart and the angular rotation of the pendulum are available data from two encoders. From the experimental set-up shown in figure (2.9), x represents the horizontal displacement of the cart and θ represents the angle of the pendulum with respect to the vertical axis. The dynamics of the system are described by a series of ordinary differential equations derived from mass and energy balances. The cart mass and pendulum mass is denoted as M and m respectively. The pendulum rod is considered to be massless, with the entire pendulum mass concentrated at the center of the pendulum bob's center of gravity. The coefficient of friction on the wheels of the cart is denoted by μ [16].

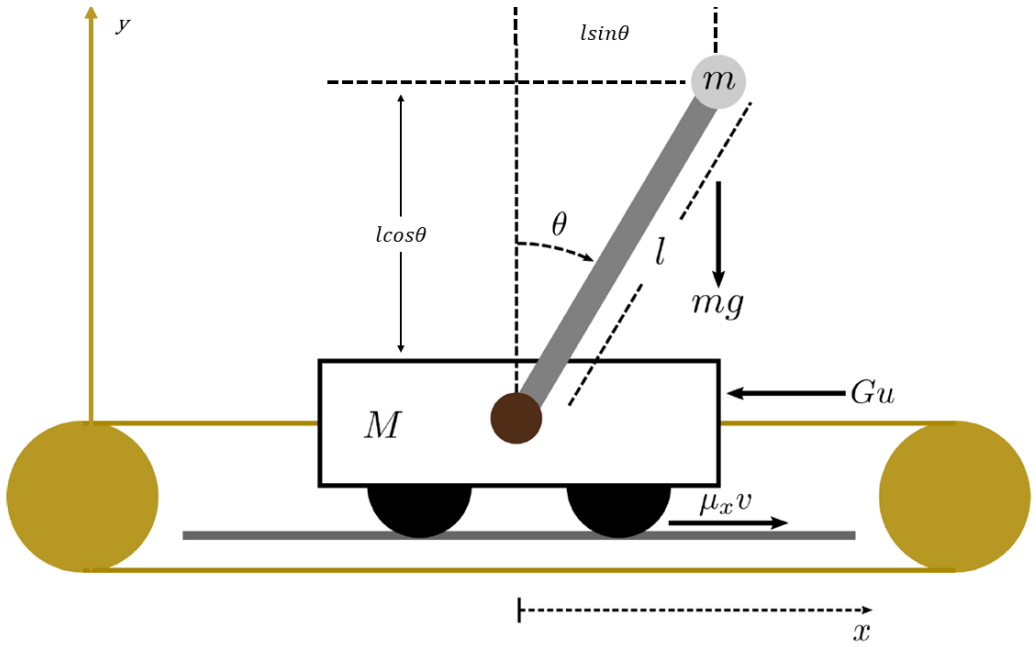


Figure 2.9: Inverted pendulum on a cart. Figure adapted from [16].

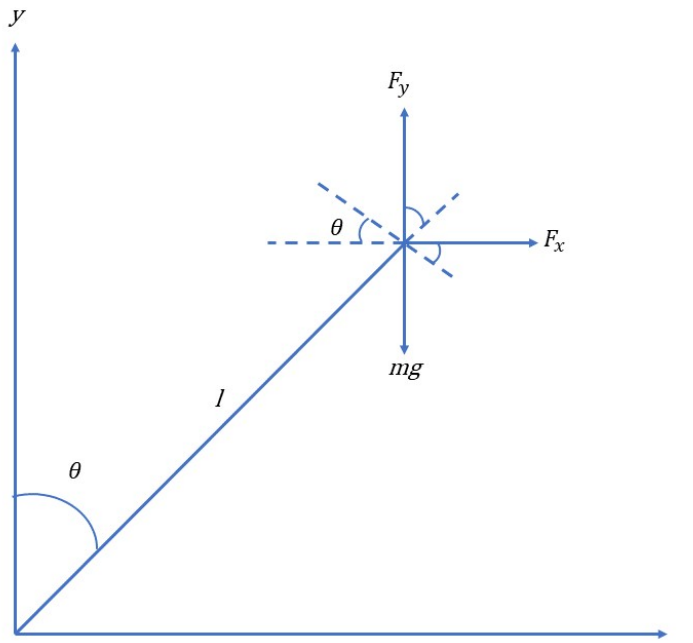


Figure 2.10: Force components of torque balance. Figure from author, adapted from [3].

As explained in [31], in the x-direction, a force balance on the system can be defined as,

$$M\frac{d^2x}{dt^2} + m\frac{d^2x_G}{dt^2} = Gu \tag{2.45}$$

where coordinates (x_G, y_G) are the time dependent center of gravity of the point mass pendulum and G is Tension-Force Gain. The location of the pendulum mass's center of gravity is

$$x_G = x + lsin(\theta), \quad y_G = lcos(\theta) \tag{2.46}$$

where l is the length of the pendulum. Substituting (2.46) in (2.45), we get,

$$(M+m)\ddot{x} + \mu\dot{x} - ml\dot{\theta}^2 sin(\theta) + ml\ddot{\theta}cos(\theta) = Gu \tag{2.47}$$

From the free body diagram in figure (2.10), the force components of the torque balance can be written as

$$(F_x cos(\theta))l - (F_y sin(\theta))l = (mgsin(\theta))l \tag{2.48}$$

where $F_x = m\frac{d^2}{dt^2}x_G$ and $F_y = m\frac{d^2}{dt^2}y_G$ are the force components in x and y directions respectively. Substituting these values in (2.48), we get,

$$m\ddot{x}cos(\theta) + ml\ddot{\theta} = mgsin(\theta)) \tag{2.49}$$

The inverted pendulum considered in this report can be described by two non-linear differential equations (2.47) and (2.49). Using these equations, we can derive the system equations for the dynamics of cart position and pendulum angle as

$$\ddot{x} = \frac{Gu - \mu\dot{x} - ml\ \ddot{\theta}^2\ sin(\theta) - mg\ cos(\theta)sin(\theta)}{M + msin^2(\theta)} \tag{2.50}$$

$$\ddot{\theta} = \frac{(M+m)\ g\ sin(\theta) + \mu\ cos(\theta)\ \dot{x} - ml\ \dot{\theta}^2 sin(\theta)\ cos(\theta)}{l(M + m\ sin^2(\theta))} \tag{2.51}$$

The nonlinear equations (2.50) and (2.51) must be represented in the standard state space form for numerical simulation of the nonlinear model for the inverted pendulum-cart dynamic system:

$$\frac{dx}{dt} = f(x, u, t). \tag{2.52}$$

The state variables are given by

$$x_1 = x, \quad x_2 = \dot{x} = \dot{x}_3, \quad x_3 = \theta, \quad x_4 = \dot{\theta}. \tag{2.53}$$

The final state equation for the nonlinear inverted pendulum system is given by

$$\frac{dx}{dt} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2.54}$$

where

$$f_1 = x_2 \tag{2.55}$$

$$f_2 = \frac{Gu - \mu\dot{x} - ml\,\ddot{\theta}^2\,sin(\theta) - mg\,cos(\theta)sin(\theta)}{M + msin^2(\theta)} \tag{2.56}$$

$$f_3 = x_4 \tag{2.57}$$

$$f_4 = \frac{(M+m)\,g\,sin(\theta) + \mu\,cos(\theta)\,\dot{x} - ml\,\dot{\theta}^2 sin(\theta)\,cos(\theta)}{l(M + m\,sin^2(\theta))} \tag{2.58}$$

Finally, if the cart position x and the pendulum angle θ are variables of interest, then the output equation can be written as

$$\boldsymbol{y} = C\boldsymbol{x}\ or\ \boldsymbol{y} = \begin{bmatrix} x \\ \theta \end{bmatrix} = C\boldsymbol{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{2.59}$$

The nonlinear inverted pendulum-cart dynamic system is completely represented in state space by (2.54) and (2.59) [31].

3 Landing a Spaceship with Koopman Operator Theory and Model Predictive Control

This section will mainly focus on the implementation of Koopman MPC in pendulum system. For the spaceship dynamics, only the specifics of how this implementation can be performed will be discussed.

3.1 Evaluation metric

The evaluation metric considered for the model evaluation are Root Mean Square Error (RMSE) and Relative Root Mean Square Error (rRMSE) are given by

$$\text{RMSE} = \sqrt{\frac{\sum_k ||x_{pred}(kT_s) - x_{true}(kT_s)||^2}{N}} \tag{3.1}$$

$$\text{rRMSE} = 100 \cdot \frac{\sqrt{\sum_k ||x_{pred}(kT_s) - x_{true}(kT_s)||_2^2}}{\sqrt{\sum_k ||x_{true}(kT_s)||_2^2}}. \tag{3.2}$$

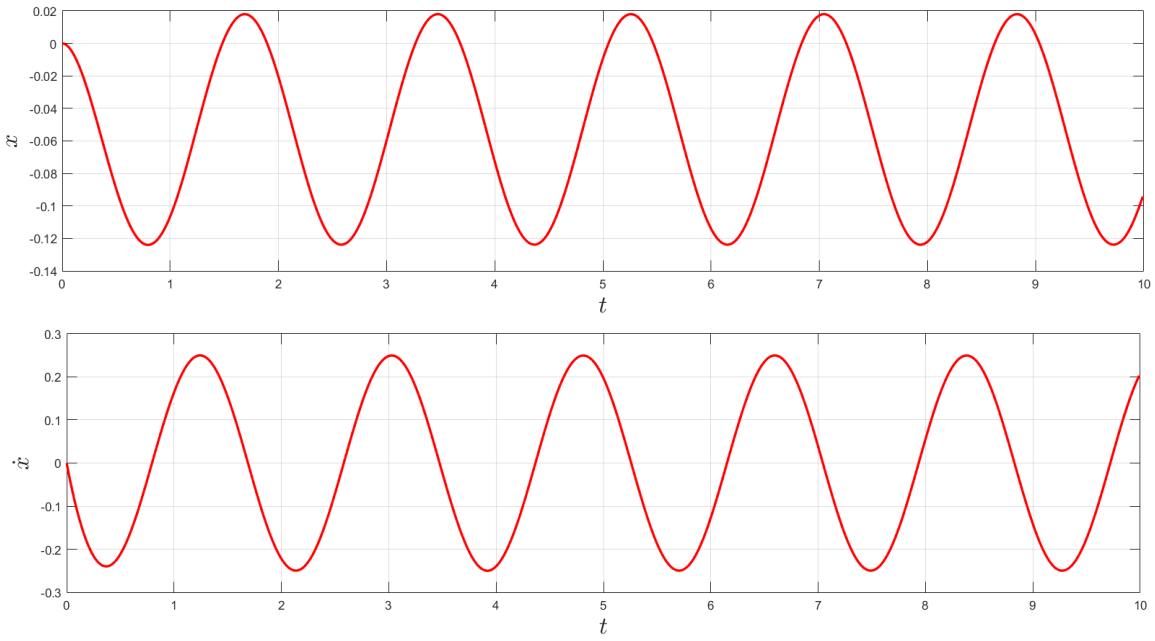
3.2 Data Preparation

In the experimental-setup, an inverted pendulum on a cart was considered as explained in section (2.8). For the task of controlling inverted pendulum on a cart, various parameters related to the pendulum are adjusted. Since the pendulum is a point mass, the mass of the bar was neglected as the center of gravity of the pendulum is concentrated on the bob. Table (3.1) lists the values of the parameters (selection of parameters adapted from [16]).

Table 3.1: Parameters of the inverted pendulum on a cart.

	Parameters	Value	Units
m	Pendulum Mass	1.12	[kg]
M	Cart Mass	0.0905	[kg]
g	Acceleration due to gravity	9.81	[ms $\cdot$ s^{-2}]
G	Tension-Force Gain	0.365	[$\cdot$]
l	Pendulum length	6.65	[m]
μ	Cart Friction	7.5	[$\cdot$]

The numerical integration of 20 trajectories were used in this experiment, with the assumption that all state variables associated with the system were available. The pendulum dynamics with 20 random initial points was supplied to the `ode45()`, a MATLAB builtin function, to handle the numerical integration of the state vector. This MATLAB function uses a 4th order Runge-Kutta to integrate the differential equations specified. For the simulation and experiment, each trajectory was sampled at a constant Δt which was chosen as 0.01 and 1000 simulations for each trajectory were generated. For control units, a sine wave in the range [-1,1] was chosen in this study. Figure (3.2) shows discrete time evolution of some trajectories with their respective control inputs at figure (3.3).



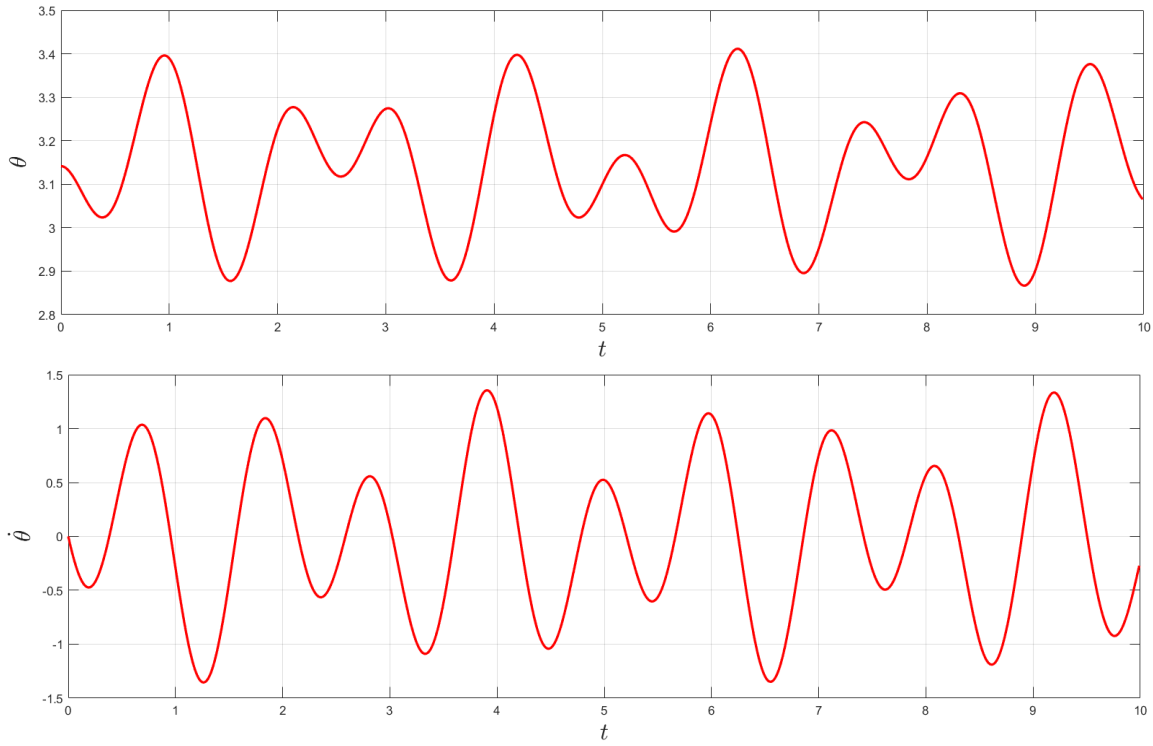


Figure 3.2: A trajectory of the states $[x,\dot{x}, \theta, \dot{\theta}]^T$

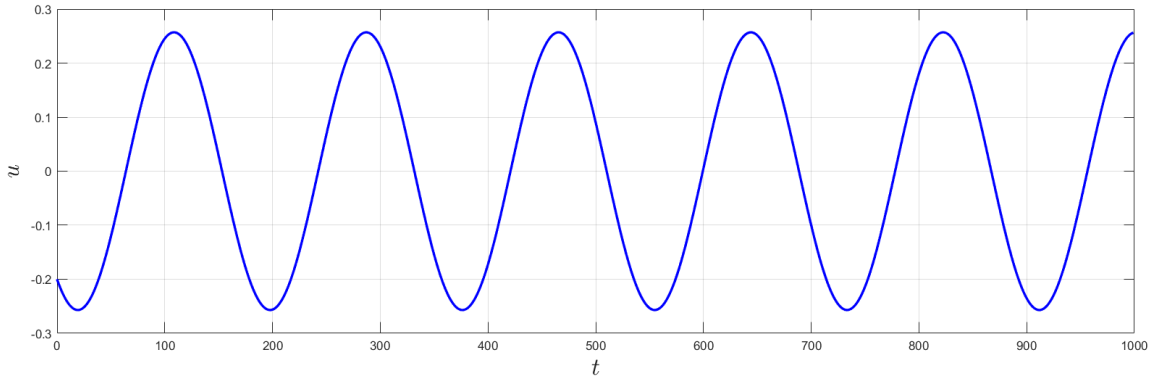


Figure 3.3: A trajectory of sinusoidal control input u.

3.3 Model Setup

To set up the model, data collection was performed. The pendulum dynamics was discretized and 20 trajectories were simulated over 1000 sampling each collected at equal

interval of 0.01 seconds. The trajectories were started from stable initial condition [0, 0, ϕ, 0] and were excited by using sinusoidal signal at various frequencies and amplitudes, i.e.,

$$\begin{bmatrix} x_0 \\ \dot{x}_0 \\ \theta_0 \\ \dot{\theta}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \pi \\ 0 \end{bmatrix}, \qquad u = A_u \cdot sin(\omega t + \phi) \tag{3.3}$$

where the parameters of the control signals are given by $A_u \in (0.1, 1)$, $\omega \in (\pi, 3\pi)$ and $\phi \in (0, 2\pi)$. These parameters were chosen to ensure that the cart movement does not exceed the track restrictions (selection of initial values adapted from [16]). This led to snapshot of data arranged in snapshot matrices as presented in (2.30) as

$$\mathbf{X} = [x_1, \ldots, x_{20}], \qquad \mathbf{Y} = [x_1^+, \ldots, x_{20}^+], \qquad \mathbf{U} = [u_1, \ldots, u_{20}] \tag{3.4}$$

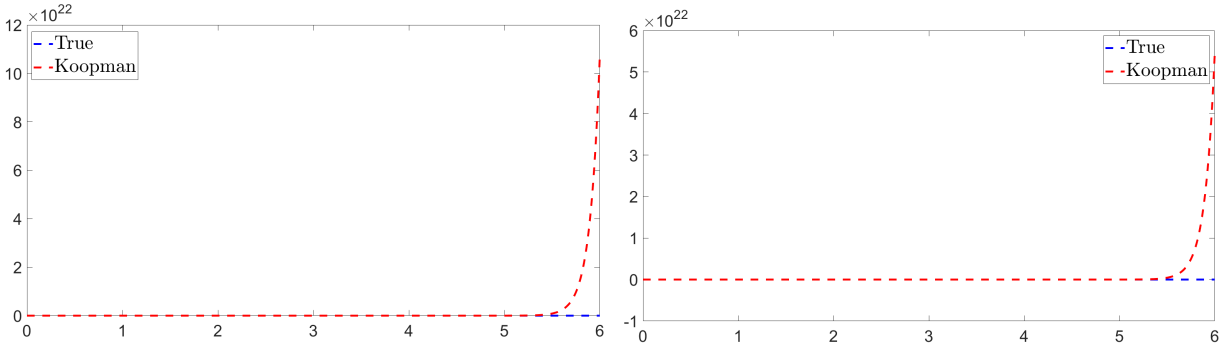
where the next state from the current state was evolved according to the pendulum dynamics. The data collection of pendulum on cart resulted in dimension of $\mathbf{X}$ and $\mathbf{Y}$ as $4 \times 4 \cdot 10^5$ where n = 4 is the number of states and m = 1 is the number of inputs. Similarly, the dimension of $\mathbf{U}$ is $1 \times 4 \cdot 10^5$.

3.3.1 Lifting of snapshot matrices

After the data collection was completed, the snapshot matrices should be lifted to generate the Koopman predictors. The choice of embedding or the lifting function was discussed in section (3.3.2). 100 RBFs centers were randomly chosen from the snapshot matrices. When the lifting function was applied, the original state was prepended to the result of lifting to preserve the information about the original state. Thus, after lifting, the dimension of $\mathbf{X}_{\text{lift}}$ and $\mathbf{Y}_{\text{lift}}$ obtained were $104 \times 4 \cdot 10^5$, where N = 104 is the dimension of lifted space. As outlined in subsection 2.6.2, the regression was performed to obtain the matrices A,B and C by solving the normal equation (2.37). The resultant matrices after regression A, B and C have dimension 104×104, 104×1 and 4×104 respectively.

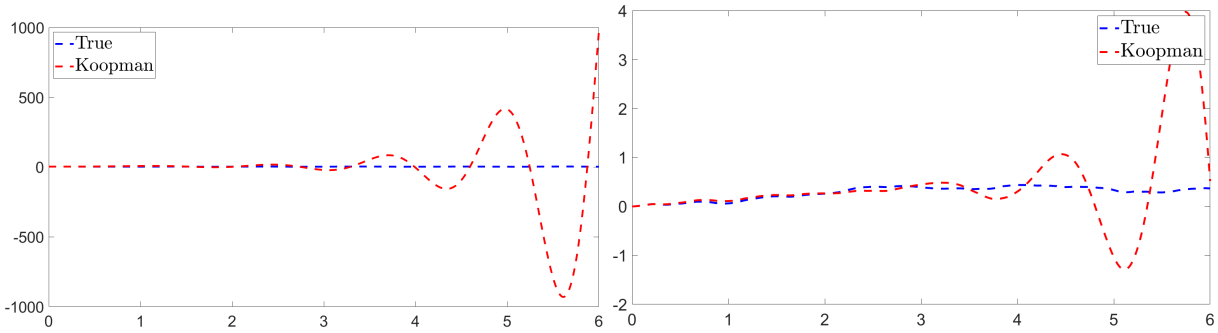
3.3.2 Choosing lifting function

This step involves selecting lifting functions ϕ. RBFs were chosen as the lifting function as it works well with the nonlinear data. There are several choices of RBFs as shown in table (2.1). All the different types of lifting functions were applied to the (3.4). A free run of trajectories from a random initial condition with the identical square wave forcing was performed for 6 seconds to compare true dynamics with Koopman predictions. The plots obtained is shown below.



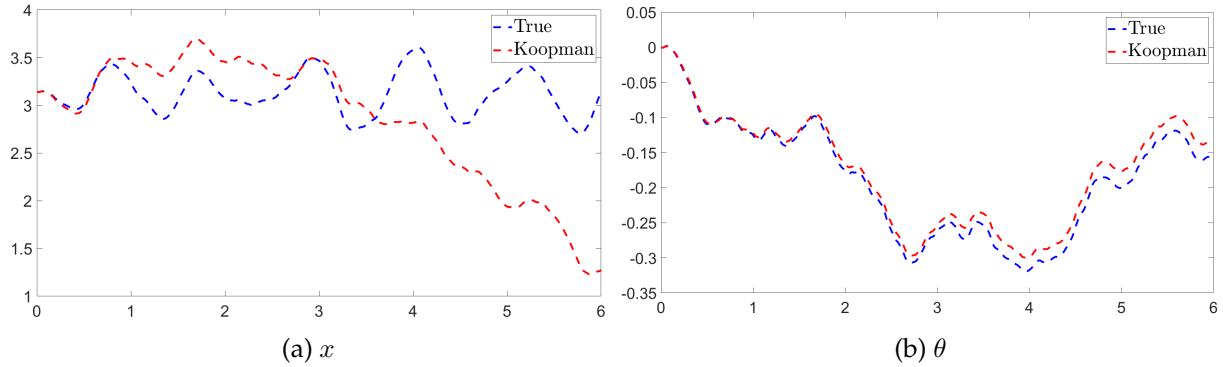
(a) x (b) θ

Figure 3.4: Thinplate RBFs lifting on pendulum position and angle.



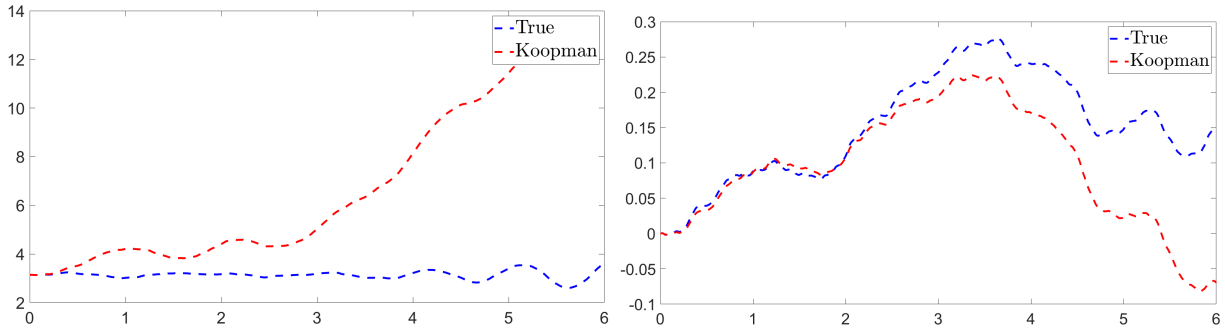
(a) x (b) θ

Figure 3.5: Gaussian RBFs lifting on pendulum position and angle.



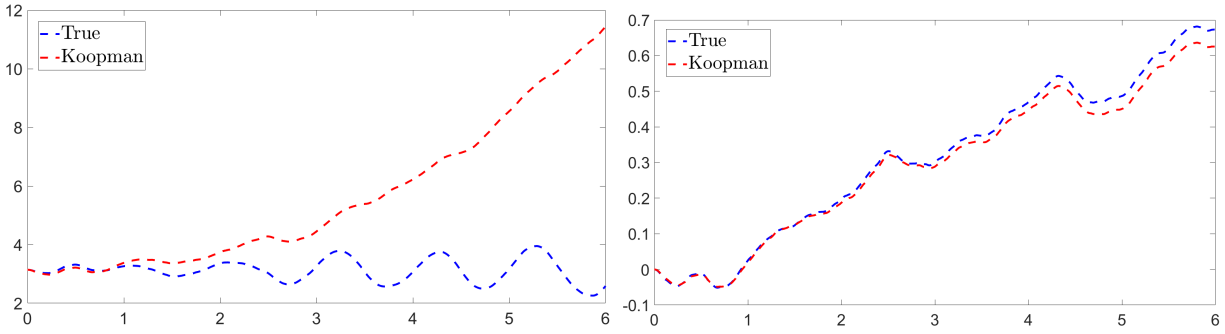
(a) x (b) θ

Figure 3.6: Inverse quadratic RBFs lifting on pendulum position and angle.



(a) x (b) θ

Figure 3.7: Inverse multiquadratic RBFs lifting on pendulum position and angle.



(a) x (b) θ

Figure 3.8: Polyharmonic RBFs lifting on pendulum position and angle.

The comparison between the different choices of RBFs for 100 randomly sampled initial conditions with the same square wave forcing as input was performed. In order to compare the true and predicted values, RMSE was calculated for each of the different lifting functions.

Table 3.2: Average RMSE of position and angle for different embeddings.

RMSE		
RBFs	position (x)	angle(θ)
Thinplate	1.13×10^3	161.25
Gaussian	7.65×10^3	395.6
Inverse quadratic	0.79	0.023
Inverse multiquadratic	2.46	0.04
Polyharmonic	4.7	0.03

From the table (3.2), it can be inferred that Inverse quadratic RBFs have less RMSE for both position and angle compared to other RBFs. Thinplate RBFs and Gaussian are not well suited for embedding the states.

Table (3.3) shows the lifting predictor's prediction accuracy in terms of average rRMSE error as a function of lift dimension N; we see that, as expected, the prediction error decreases with increasing N, though not in a linear fashion. The rRMSE was taken for the free run of the true and predicted state for 3 seconds of simulation.

Table 3.3: Average rRMSE of Koopman predictor and N, for 100 random initial points.

N	5	10	25	50	75	100
Average rRMSE	64.3	61.4	61.6	60.8	60.6	57.3

3.4 Experimental Setup

The Dense form Koopman MPC as explained in (2.42) was implemented with the model parameters summarized in Table (3.4). The data matrices in (2.42) were built from the values A, B, C obtained from lifting and applying regression on the lifted snapshot matrices. `getMPCPend()` function was called with the parameters A, B, C, Q, R, N_p, u_{min}, u_{max}, z_{min} and z_{max}. The prediction horizon was set to one second, which results in N_p = 100. The control input was constrained to [u_{min}, u_{max}] = [-1,1] while the lifted state was constrained to [z_{min}, z_{max}] = [-0.8,0.8]. The constraints for the z was obtained from trail and error, since the constraint for x is in lifted space. The function returned a controller which was utilized later in the simulation loop. A tuple (x_{curr}, y_r) was passed to the controller, where x_{curr} is the current state and y_r is the reference to be tracked. The control objective is to track the cart position and pendulum angle. The value Q penalizes the state while the value R penalizes the control input. R was chosen to be 0.01 so that we can actuate it aggressively. The cost function to minimize is given by

$$J = (y_{N_p} - yr_{N_p})^T Q_{N_p} (y_N - yr_N) + \sum_{i=0}^{N-1} [(y_i - yr_i)^T \, Q \, (y_i - yr_i) + u_i^T \, R \, u_i] \tag{3.5}$$

where after minimization, we got the control law u_{koop} that we applied to the system to get the next state. For the Dense form Koopman MPC, the data matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{E}$ and $\mathbf{F}$ should be computed as shown in (2.44). Following the computation of the controller, the simulation was performed with the current state being lifted and passed along with the reference to the controller at each time step. At every loop of the simulation, the Koopman control was obtained u_{koop}, which was then applied to the system.

Table 3.4: Different parameters for performing Koopman MPC.

n	4
m	1
N	100
N_{sims}	1000
T_{max}	10
Q	120
R	0.01
N_p	100

As we know, MPC is a technique in which a predictive system model is used to evaluate a sequence of future control inputs; an optimization algorithm selects the best of all such control input sequences. The first input of the sequence is usually implemented. The process is repeated after a specific period of time to find a fresh control input. This is simple when the predictive model is deterministic. When the predictive model is uncertain (e.g., stochastic or adversarial), open-loop MPC and closed-loop MPC are frequently used. Similarly, in our case the optimal control, i.e. the first u value from the list of controls was selected.

3.5 Open-loop control

It is important to see how far the Koopman system can predict if no information other than the initial state is given. For open-loop Koopman MPC simulation, the system starts from an initial point, but it is not re-initialized, i.e. no feedback of the true system goes back to the surrogate Koopman model. The results and evaluation of the open-loop control are presented in this section.

3.5.1 Experimental Results

The open-loop control experiment was set up and the results from the experiments are presented below. Figures (3.9) and (3.10) shows the resultant trajectories for position (x) and pendulum angle (θ) where figure (3.11) is the control law obtained from the Koopman controller. Only a single, fixed sequence of future control inputs were considered in open-loop MPC; this sequence must produce good performance under all plausible realizations of the uncertainty.

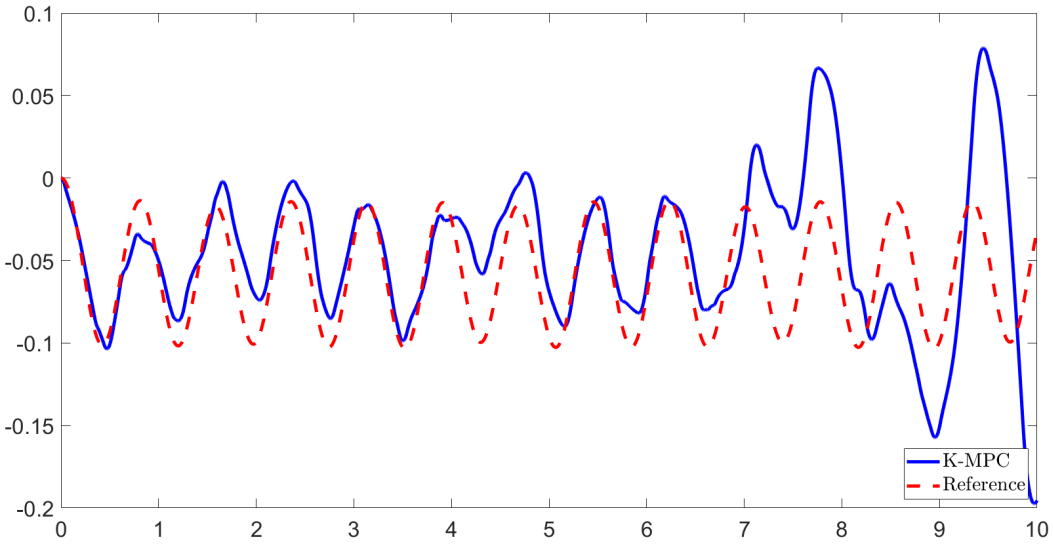


Figure 3.9: Comparing trajectory of cart positions obtained from Koopman MPC and arbitrary reference trajectory in open-loop, with prediction horizon of 1 second, N_p = 100..

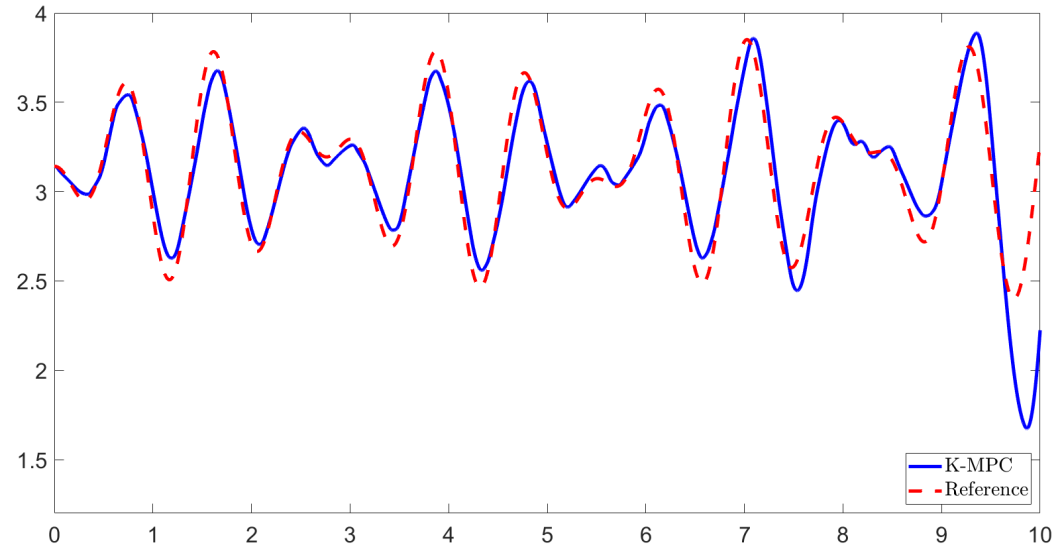


Figure 3.10: Comparing trajectory of pendulum angle obtained from Koopman MPC and arbitrary reference trajectory in open-loop, with prediction horizon of 1 second, N_p = 100..

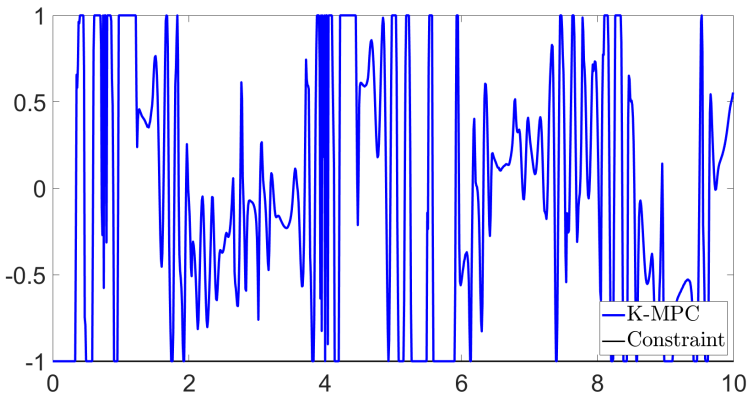


Figure 3.11: Control obtained from Koopman MPC for arbitrary reference in open-loop.

From the figures (3.9) and (3.10), we can observe that the open-loop prediction for both position and angle of pendulum is not accurate after certain time steps. This is evident clearly on figure (3.9), where after 6.5 seconds the trajectory is just an oscillation whose amplitude is higher than that of the reference trajectory. It is because the model does not have enough information of the true state other than the initial points.

3.5.2 Model Evaluation

In this subsection, we compare the Koopman MPC with Original MPC for step and sinusoidal reference in open-loop control. The term "Original system MPC" refers to nonlinear MPC in this context.

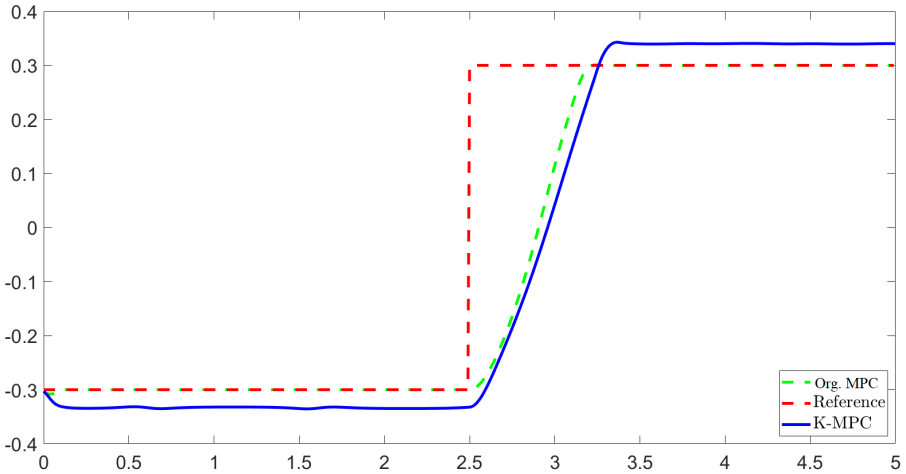
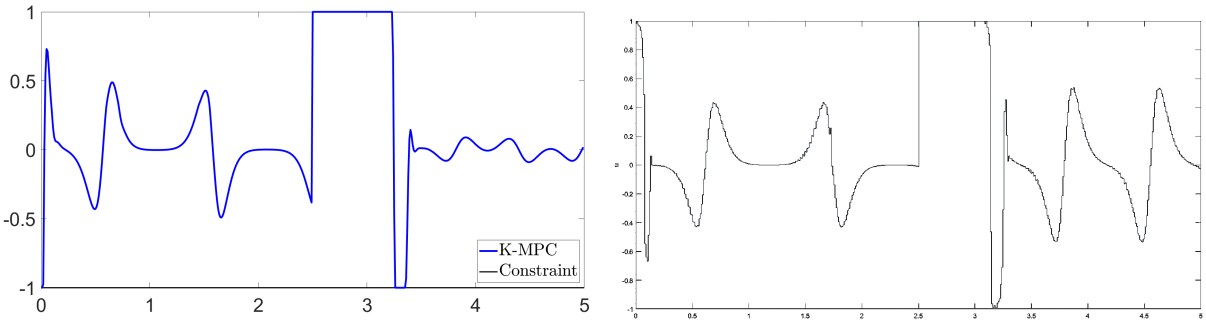


Figure 3.12: Trajectories for step reference (red), Koopman MPC (blue) and Original system MPC (green) for cart position, with prediction Horizon of 1 second, $N_p = 100$.



(a) u for Koopman MPC (b) u for Original system MPC

Figure 3.13: Control obtained from Original system MPC and Koopman MPC for step reference.

From the figures (3.13 (a)) and (3.13 (b)), we can observe that the control law in the Original system MPC in (3.13 (b)) is similar to (3.13 (a)). The control rule is more similar in the early time step, up to 3 seconds, than in the latter time step. We can observe that the original control is more detailed than the Koopman MPC control in the subsequent time step, starting at 3.5 seconds. This is reflected in the figure (3.12), where the Original system MPC better approximates in the later time step. When the sudden change happens in amplitude of reference at 2.5 seconds from -0.3 to 0.3, clearly both methods try to slowly approximate the reference. The original MPC reaches the reference at approximately 2.75 seconds, whereas the Koopman MPC overshoots and parallels the reference.

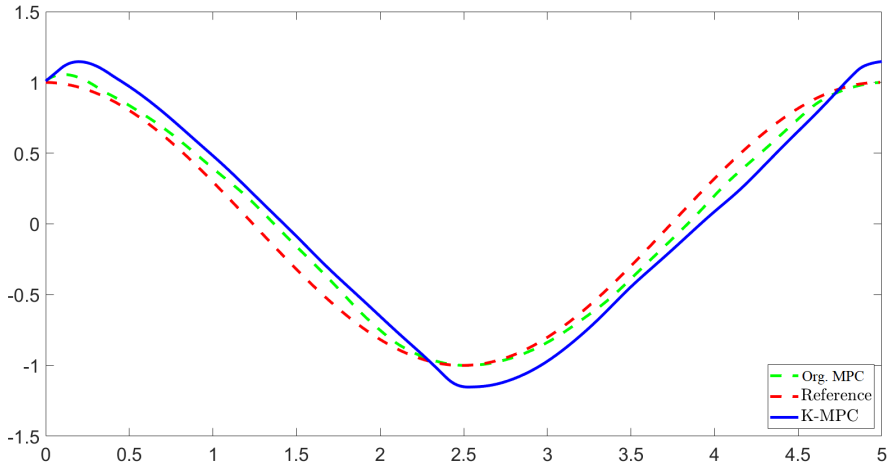
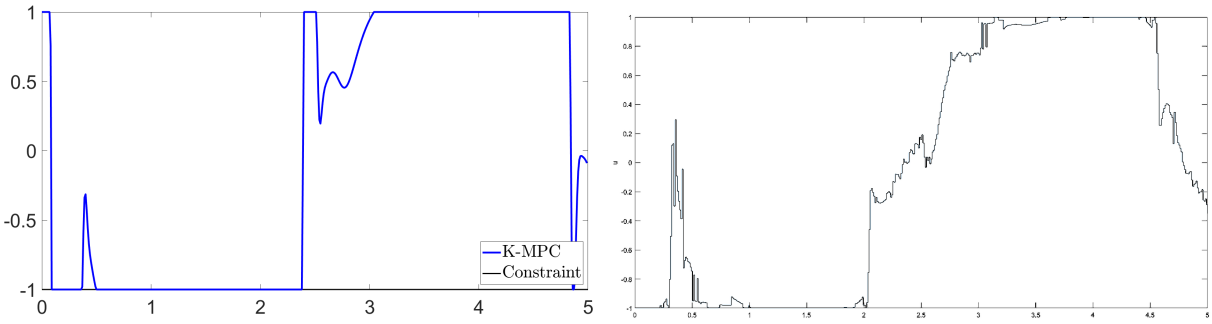


Figure 3.14: Trajectories for sinusoidal reference (red), Koopman MPC (blue) and Original system MPC (green) for cart position, with prediction horizon of 1 second, N_p = 100.



(a) u for Koopman MPC (b) u for Original system MPC

Figure 3.15: Control obtained from Original system MPC and Koopman MPC for sinusoidal reference.

If we compare the trajectories of both MPCs in figure (3.14) with their corresponding control reflected in figure (3.15), we can observe that from time step 2 seconds to 3 seconds, Koopman MPC is less accurate than Original MPC. The control strategy for Original MPC is more detailed than that of Koopman MPC in that time frame. On later time frame after 4.5 seconds, Koopman MPC is superficial as it aggressively applies the control while Original is detailed and precise. This control is reflected in the resultant trajectory, which is more accurate for Original MPC than Koopman MPC.

Table 3.5: Simulation time comparison Koopman MPC and Original MPC.

Time elapsed (s)		
Reference	KMPC	Original MPC
step	0.69	560.231
sinusoidal	0.54	516.136

From table (3.5), it is clear that the Original MPC takes more time to compute than that of KMPC. This result is obvious as Original MPC uses nonlinear programming solver (`fmincon`) to get the control value u, while in case of KPMC, it is data driven and the data matrices are precomputed and used only when the MPC simulation runs. We can observe from figure (3.13) and (3.15) that the control strategy applied for the nonlinear and Koopman MPC follows the similar trajectory.

Table 3.6: RMSE and RRMSE errors for Koopman MPC and Original MPC.

Step Reference		
	RMSE	RRMSE
KMPC	0.166	55.46%
Original MPC	0.147	49.19%

Table 3.7: RMSE and RRMSE errors for Koopman MPC and Original MPC.

Sinusoidal Reference		
	RMSE	RRMSE
KMPC	0.171	24.22%
Original MPC	0.088	12.48%

However, from tables (3.6) and (3.7), it is clear that Original MPC performs better than Koopman MPC. It is evident from the plots shown in figure (3.14) and (3.12) that Original MPC better approximates the reference trajectory. But for the real time systems, Koopman MPC would be applicable as it gives similar trajectory as Original MPC with lower computational complexity.

3.6 Closed-loop control

Closed-loop MPC takes input into account for its recourse. This relates to the idea that the controller will have more information available to it before making future decisions than it does now. As a result, the controller must optimize control policies rather than just inputs. In our simulation, after every second the state is re-initialized to the trajectory points. The results and evaluation of the closed-loop control are presented in this section.

3.6.1 Experimental Results

The closed-loop control experiment was set up in the same way as the open-loop control experiment. Figures (3.16) and (3.17) shows the resultant trajectories for position (x) and pendulum angle (θ) where figure (3.18) is the control law obtained from the Koopman controller.

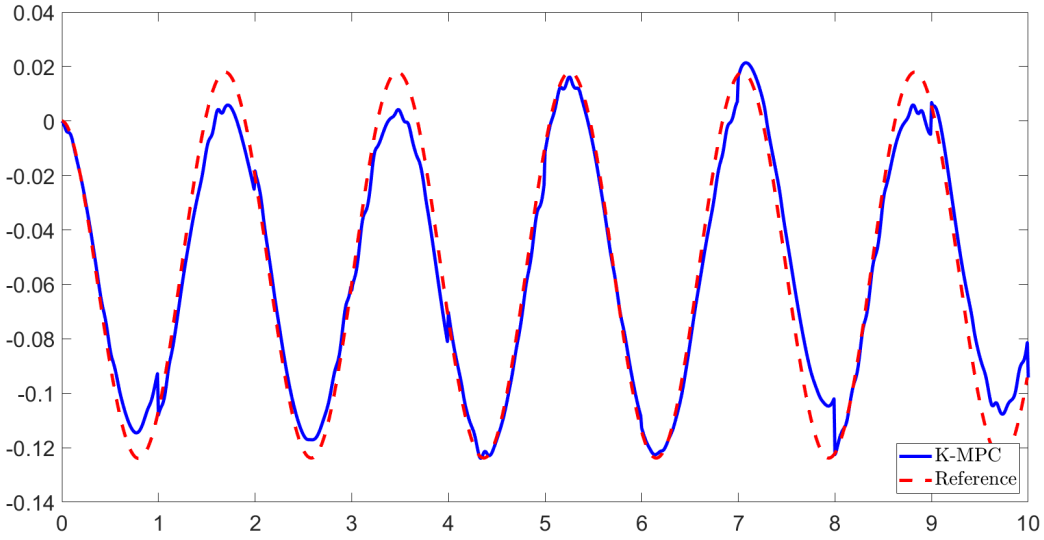


Figure 3.16: Comparing cart positions obtained from Koopman MPC and arbitrary reference trajectory in closed-loop with prediction horizon of 1 second, $N_p = 100$.

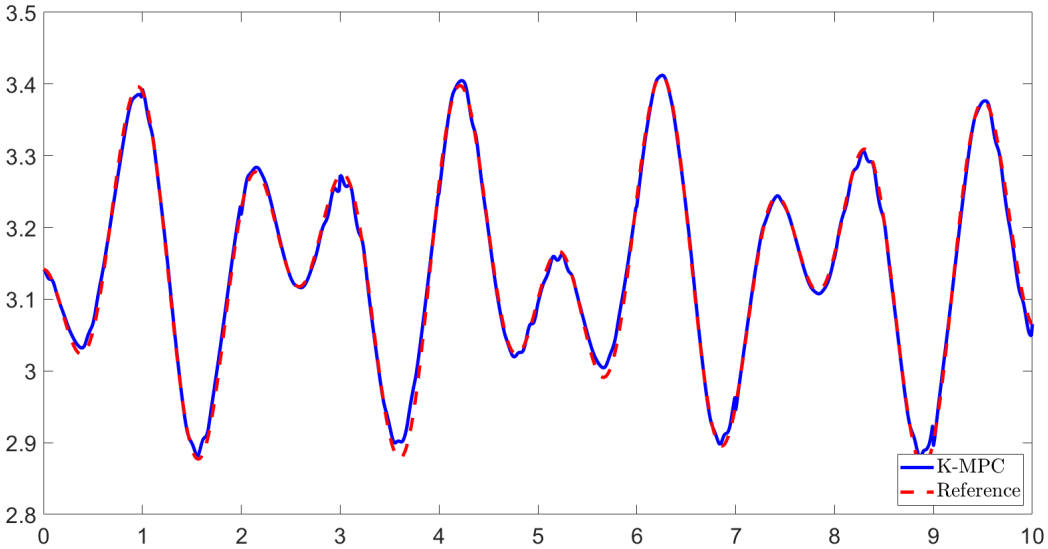


Figure 3.17: Comparing pendulum angle obtained from Koopman MPC and arbitrary reference trajectory in closed-loop with prediction horizon of 1 second, $N_p = 100$.

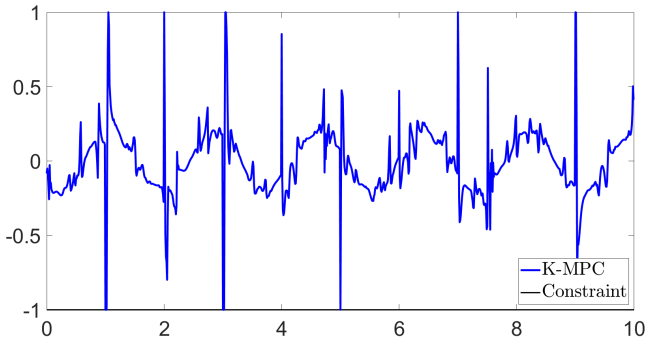


Figure 3.18: Control obtained from Koopman MPC for arbitrary reference in closed-loop.

From the figure (3.16) and (3.17), we can observe that the close-loop prediction for both position and angle of pendulum better approximates the reference trajectory. The approximation is accurate even for the later time steps. It is because the model now has the information of the true state at every 1 second. Due to this re-initialization, the predicted trajectory follows the true one closely and accurately.

3.6.2 Model Evaluation

In this subsection, we compare the Koopman MPC with Original MPC for step and sinusoidal reference in closed-loop control.

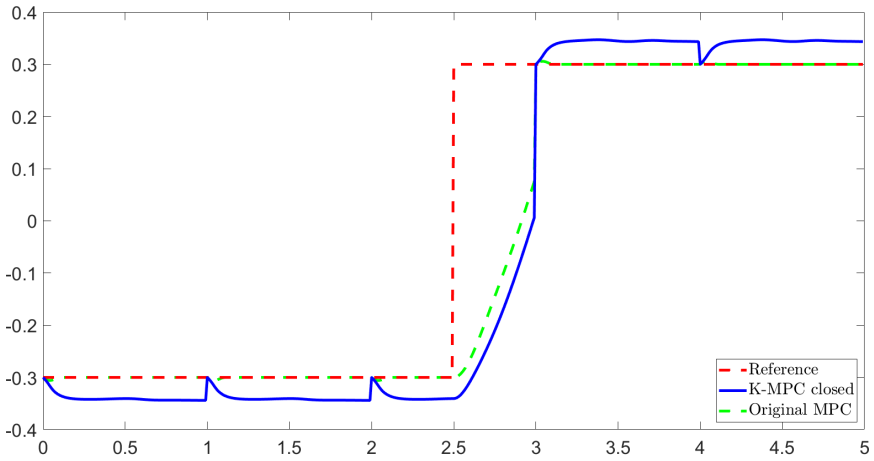
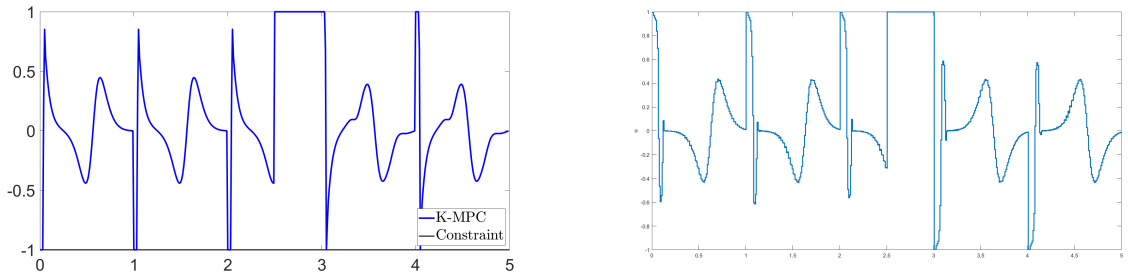


Figure 3.19: Trajectories for step reference (red), Koopman MPC (blue) and Original system MPC (green) for cart position in closed-loop with prediction horizon of 0.1 second, $N_p = 10$.



(a) u for Koopman MPC (b) u for Original system MPC

Figure 3.20: Control obtained from Original system MPC and Koopman MPC for step reference.

For this evaluation, we considered closed-loop scenario where the both Original as well as Koopman MPC were initialized to original trajectory state at every 1 second. For step reference, we can observe that the control is similar for both. However, for Koopman MPC, the control is more aggressive than Original system control. In addition, the trajectory for Koopman MPC deviates from the reference even if it is initialized to the reference state.

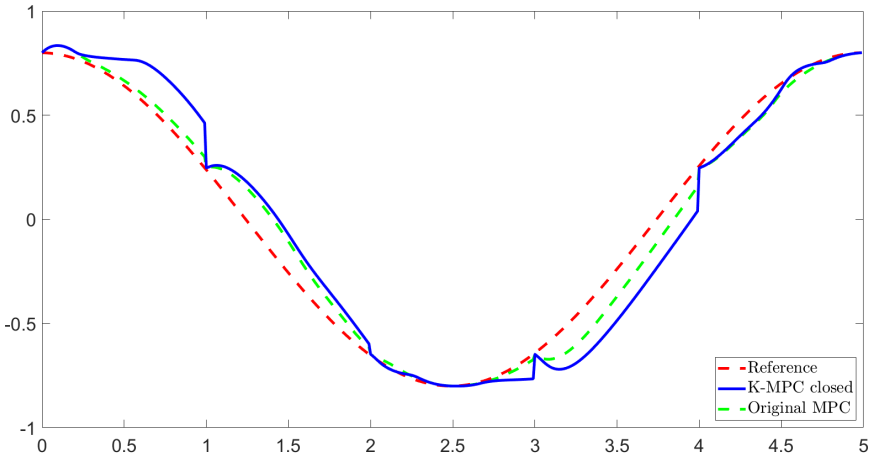
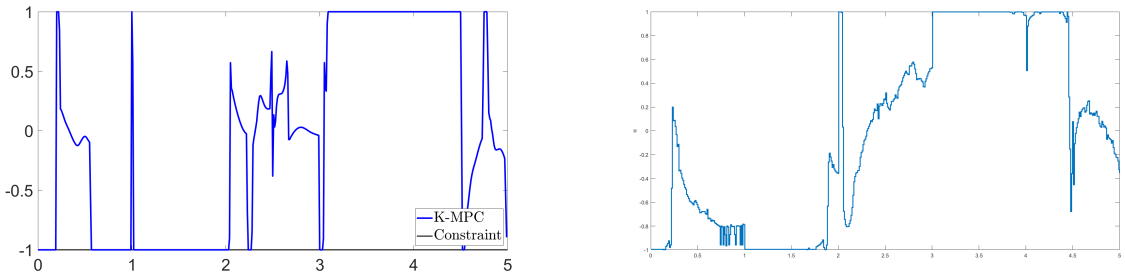


Figure 3.21: Trajectories for sinusoidal reference (red), Koopman MPC (blue) and Original system MPC (green) for position in closed-loop with prediction horizon of 0.1 second, N_p = 10.

For the sinusoidal reference, the re-initialization does not improve the result significantly. This behavior could be due to a shorter prediction horizon for the Koopman MPC.



(a) u for Koopman MPC

(b) u for Original system MPC

Figure 3.22: Control obtained from Original system MPC and Koopman MPC for sinusoidal reference.

Similar to open-loop control, the Original MPC takes more time to compute than that of KMPC. We can observe from figure (3.20) and (3.22) that the control strategy applied for the nonlinear and Koopman MPC follows the similar pattern. The control for the Original system MPC is more detailed than the Koopman MPC, and hence it approximates the reference better than KMPC.

Table 3.8: RMSE and RRMSE errors for Koopman MPC and Original MPC.

Step Reference		
	RMSE	RRMSE
KMPC	0.165	55.3%
Original MPC	0.144	48.2%

Table 3.9: RMSE and RRMSE errors for Koopman MPC and Original MPC.

Sinusoidal Reference		
	RMSE	RRMSE
KMPC	0.13	24.5%
Original MPC	0.08	14.17%

The tables (3.8) and (3.9) shows the error comparison between Original MPC and KMPC. In this case as well, Original MPC performs better than that of KMPC. It is evident from the plots shown in figure (3.21) and (3.19) that Original MPC better approximates the reference trajectory. The error obtained is similar to that of the open-loop control.

The Koopman MPC cannot precisely approach the reference trajectory, as evidenced by both open- and closed-loop control. The weighing variables between the cost of the control and the cost of how quickly the Koopman trajectory should reach the reference could be

one of the reasons for the discrepancy (i.e. norm(u) vs norm(KMPC trajectory - reference trajectory).

3.7 Koopman MPC for Spaceship dynamics

A flying spaceship is subjected to four forces: gravity, drag, lift, and thrust. Figure (3.23) depicts the force direction for a rocket moving uphill.

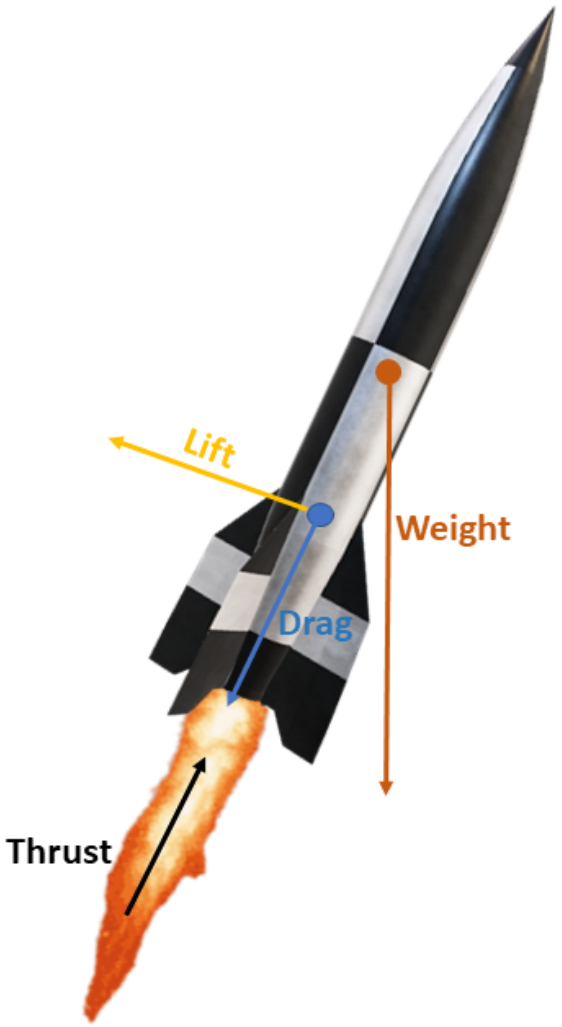


Figure 3.23: The direction of forces acting on a flying spaceship.

The basic mechanism of spaceship flight is based on high-speed gas ejection from the combustion chamber. Irrespective to the size or shape of the spaceship, the general dy-

namics of the spaceship follows Newtonian force. As a result of Newton's third law, this process generates force that lifts the rocket.

Let us define F_T to be a force generated as a result of the thrust which is equal to the product of rate of fuel consumption and its exhaust velocity. The thrust is directed along the rocket's longitudinal axis, through its center of gravity. The gravitational force of the Earth is constantly acting on the rocket, directing it to the planet's center. This force, also known as the object's weight, is proportional to the object's mass and quadratic inverse proportional to the distance from the planet's center. The gravitation force acting on the spaceship is given by

$$F_G = G\frac{m\ M}{(R_e + h)^2} \tag{3.6}$$

where G is the universal gravitational constant, m is the mass of the object on which the force is acted upon, M is the mass of the Earth, R_e is the radius of the Earth and h is the height of the object from the Earth, also known as altitude.

Lift and drag forces are aerodynamic forces that are affected by the rocket's size, shape, and velocity. They are also influenced by the characteristics of the air in which the rocket is traveling. The lift force is determined by the difference in air pressure and the angle of flight. The drag force is perpendicular to the motion and is proportional to the contacting surface area and velocity. The lift force is given by

$$F_L = \frac{1}{2}\rho v^2 S_r e f C_L, \quad \rho = \rho_0 e^{-\beta h}, \quad h = r - R_e \tag{3.7}$$

and the drag force is given as

$$F_D = \frac{1}{2}\rho v^2 S_r e f C_D, \quad \rho = \rho_0 e^{-\beta h}, \quad h = r - R_e \tag{3.8}$$

where ρ is the air density, S_ref is the reference area, C_D is the drag coefficient, C_L is the lift coefficient, ρ_0 is the sea level density, β is the density scale, and h is the flight altitude.

The net force acting on the spaceship can be given by

$$F_{net} = F_T + F_D - F_G. \tag{3.9}$$

For vertical motion, the sign of the drag force FD is determined by the rocket's motion and is equal to the opposite of the vertical velocity direction. Newton's second law can be used to calculate the acceleration of a rocket using the net force acting on it and is given by

$$F = ma \tag{3.10}$$

Let $y = (r_d, s)^T$ denote the position vector of the spaceship and v be its velocity, then from acceleration, we can calculate the vertical velocity and position as

$$v_t = v_0 + a(y_0, v_0)\Delta t \tag{3.11}$$

$$y_t = y_0 + v_t \Delta t \tag{3.12}$$

where y_0 and v_0 are initial position and velocity.

For landing of the spaceship, the acceleration acting on the spaceship can be calculated as

$$a = \frac{F_T - F_D - F_G}{m}. \tag{3.13}$$

The velocity and position calculation for the spaceship can be performed similar to (3.11) and (3.12) as

$$v_{t+1} = v_t + a_t \Delta t \tag{3.14}$$

$$y_{t+1} = y_t + v_{t+1} \Delta t. \tag{3.15}$$

Thus the objective function to minimize is given by

$$J = \sum_{i=1}^{N_p} x^T W x + \lambda u^2 \tag{3.16}$$

where the values of x and W are given as

$$x = \begin{bmatrix} y_r - y \\ v_r - v \end{bmatrix} \tag{3.17}$$

$$W = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.18}$$

In the above equation, x represents the difference between the reference and current states, y_r represents the reference altitude, v_r represents the reference velocity, W represents the weight matrix, u represents the input, and λ represents the input coefficient [3].

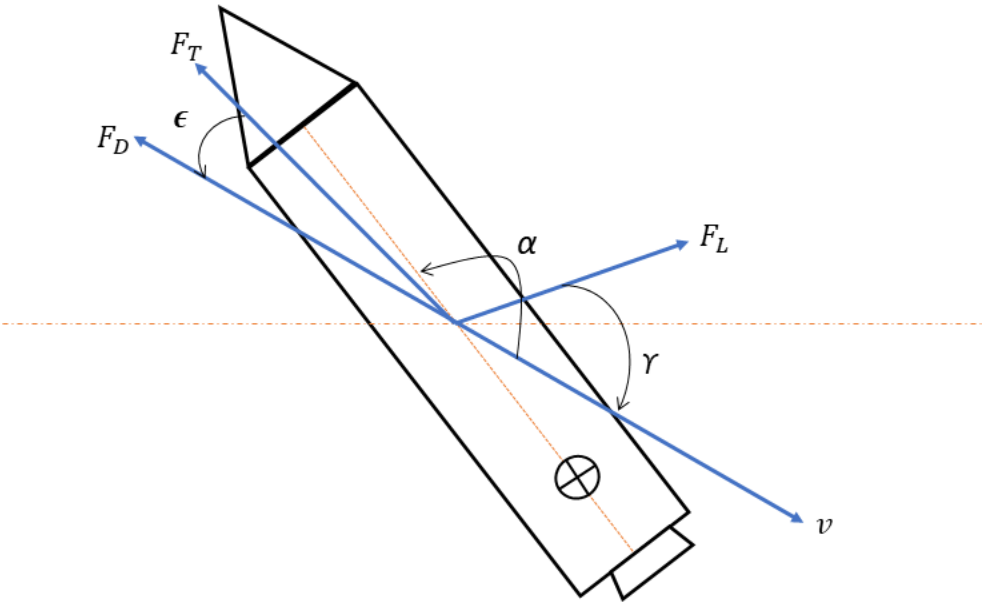


Figure 3.24: Geometry of the spaceship. Figure from author, adapted from [39].

The fuel-optimal spaceship landing problem is addressed in the study [26] using both aerodynamic forces and propulsion as control inputs in the system dynamics. This model is more sophisticated than the model presented above. The aerodynamic lift and drag forces (equivalently, the angle of attack) are considered control inputs in addition to the thrust. The two-dimensional flight of a spaceship over a flat Earth is the subject of the paper. As observed from the geometry (3.24), spaceship's equations of motion can be written as

$$\begin{aligned} \dot{r_d} &= v sin\gamma \\ \dot{s} &= v cos\gamma \\ \dot{v} &= \frac{-F_T cos\,\epsilon - F_D}{m} - \frac{sin\,\gamma}{r^2} \\ \dot{\gamma} &= \frac{-F_T sin\,\epsilon + F_L}{mv} - \frac{cos\,\gamma}{r^2 v} \\ \dot{m} &= \frac{-F_T}{I_{sp}} \end{aligned} \tag{3.19}$$

where r_d is defined as the radial distance from the center of the Earth to the spaceship, s is the downrange, which together forms the position vector of the rocket $(r_d, s)^T$, v is the velocity of the spaceship, γ is the flight path angle, m is the mass of the spaceship, ϵ is the thrust direction which is defined as the angle between the thrust vector and the negative velocity vector, and I_{sp} is the specific impulse of the spaceship engine.

The variable of controls, the angle of attack α, thrust F_T, and thrust direction ϵ in spaceship vertical landing may be constrained by their lower and higher bounds, i.e.,

$$\begin{aligned} \alpha_{min} \leq \alpha \leq \alpha_{max} \\ F_{Tmin} \leq F_T \leq F_{Tmax} \\ \epsilon_{min} \leq \epsilon \leq \epsilon_{max}. \end{aligned} \tag{3.20}$$

As shown in (3.20), the initial and terminal conditions are contained in boundary constraints. Terminal constraints should address landing velocity, attitude, and fuel consumption in order to assure the spaceship's safe landing [37].

$$\begin{aligned} \boldsymbol{x_0} &= [r_d(t_0), s(t_0), v(t_0), \gamma(t_0), m(t_0)]^T \\ \boldsymbol{x_f} &= [r_d(t_f), s(t_f), v(t_f), \gamma(t_f), m(t_f)]^T \end{aligned} \tag{3.21}$$

$$v_f \leq v_{safe}, \quad \gamma_f = 90^0, \quad \|\alpha_f\| \leq \alpha_s afe, \quad m_f \leq m_d ry \tag{3.22}$$

where v_{safe} is the maximum landing speed, α_{safe} is the maximum landing angle of attack and m_{dry} is the structural mass of the spaceship.

Finally, the optimization goal is to reduce the cost of fuel or, in other words, to increase the final mass. As a result, we have the following objective function for a minimization issue [26].

$$J = -m(t_f) \tag{3.23}$$

The final optimization problem is given as

$$
\begin{aligned}
min \quad & -m(t_f) \\
s.t \quad & \dot{r_d} = vsin\gamma \\
& \dot{s} = vcos\gamma \\
& \dot{v} = \frac{-F_T cos\,\epsilon - F_D}{m} - \frac{sin\,\gamma}{r^2} \\
& \dot{\gamma} = \frac{-F_T sin\,\epsilon + F_L}{mv} - \frac{cos\,\gamma}{r^2 v} \\
& \dot{m} = \frac{-F_T}{I_{sp}} \\
& \alpha_{min} \leq \alpha \leq \alpha_{max} \\
& F_{Tmin} \leq F_T \leq F_{Tmax} \\
& \epsilon_{min} \leq \epsilon \leq \epsilon_{max} \\
& \boldsymbol{x_0} = [r_d(t_0), s(t_0), v(t_0), \gamma(t_0), m(t_0)]^T \\
& \boldsymbol{x_f} = [r_d(t_f), s(t_f), v(t_f), \gamma(t_f), m(t_f)]^T \\
& v_f \leq v_{safe} \\
& \gamma_f = 90^0 \\
& \|\alpha_f\| \leq \alpha_s afe \\
& m_f \leq m_d ry
\end{aligned}
\tag{3.24}
$$

The main ideas for applying the MPC in this case can be summarized below [11] [39]:

- **Space Dynamics Model**: To create control signals, the controller uses a simplified model of the system to regulate a space launcher. The model should be defined as per task on hand. It can also be a simple model with no aerodynamic force involved. This model is used in the data collection phase.
- **Controller Design**: The controller must be created as per the methods outlined in subsections 2.6.2 and 2.7.3. The lifting of the data collected from the model should be performed as per subsection 2.6.2, while the dense Koopman MPC should be created as per subsection 2.7.3. The final controller function should take in the reference and lifted state as input, and produce control signals.
- **Cost function minimization**: The controller must minimize the result of a predetermined cost function that determines how effectively it operates. The cost function will be connected to the use of rocket fuel.
- **Adhere to a set of constraints**: For the control of the system, it is important to note that it should not leave a viable operating zone. For example, the spaceship cannot land on a negative altitude, and the thrust provided must be within the capabilities

of the vehicle's engine. The limitations might additionally include the launcher's beginning and intended ending positions and speeds, which must fall within the practical range.

- **Receding horizon strategy**: One of the variables to set before running the MPC is the horizon time. It specifies the starting point for calculating the control action. It is termed a receding horizon because it never reaches the present time, but is always in the future. This can cause some issues while attempting to land a launcher.

After applying the implementation of the code for the pendulum dynamics to the data of spaceship dynamics, first of all, the N dimensional lifted states would be generated from the data trajectory. Appropriate lifting functions would be applied to the system. These lifted states would then be utilized to generate data matrices A, B and C. These data matrices, as well as the hyper-parameters Q and R , would be used to develop an MPC-based controller. For the initialization of controller, the constraints relating to spaceship dynamics outlined in (3.20) (3.21) (3.22) would be applied to the system. The landing reference trajectory should be built using the following information: the final state (0,0) position, 0 velocities, and an orientation of $\pi/2$. With this, we can obtain u_{koop}, which determines the spaceship's control strategy after applying MPC. The reference trajectory would then be compared to the Koopman MPC trajectory to assess how well the Koopman MPC performed.

4 Conclusions

4.1 Summary

The Koopman operator was used to design a Model Predictive Control paradigm for nonlinear systems in this study. The underlying concept is to incorporate nonlinear dynamics in a higher-dimensional environment where their evolution is roughly linearly predictable. The predictors developed in this manner outperform conventional linear predictors (e.g., local linearization [23]) and is easily used for feedback control with MPC in a purely convex manner. In this study, the Koopman-based predictors were generated for controlling a pendulum on a cart.

While the Koopman operator framework is gaining popularity in the control world, there are still areas of study and issues that have yet to be addressed. As mentioned in [27], there are a few outstanding questions in the context of the Koopman operator paradigm applied to control theory. One of the main challenges is dealing with the infinite-dimensional nature of the operator and the inherent approximations of numerical techniques, which is a cost of developing linear methods for nonlinear systems.

4.2 Discussion

The experimental results show that the results from the dense Koompan MPC are similar to the nonlinear MPC, and in terms of time complexity, Koopman MPC is superior to the nonlinear MPC. The computational complexity of dense form MPC is independent of the number of states and only depends on the number of control inputs. Both open- and closed-loop control show that the Koopman MPC cannot exactly reach the reference trajectory. One of the explanations for the mismatch could be the weighing variables between the cost of the control and the cost of how quickly the Koopman trajectory should approach the reference. To achieve better outcomes, the hyperparameters should also be tuned appropriately.

Importantly, the entire control design approach is based on data, with only input–output measurements being required. The choice of the lifting functions and number of lifting parameters played a vital role in the control of the pendulum cart system. From the experimental results, inverse quadratic-based lifting functions were more suited for the task.

4.3 Outlook

The MPC based on Koopman's theory was thoroughly investigated and applied to a nonlinear dynamical system, and the future directions to take for improvement of the study can be outlined below.

- **Implementation of control of spaceship dynamics:** Koopman operator theory and Koopman-based MPC can both be used to control a spaceship using similar principles. This study has only outlined the problem statement and cost function related to space dynamics control; therefore, implementation of this technique should be done as part of future research.
- **Investigation and use of different constraints:** Several relevant constraints can be investigated and implemented in the Koopman MPC. In our implementation, we used constraints on input and lifted state. However, several other constraints should be investigated and applied to the system.
- **Investigate Lifting functions:** The use of lifting functions is one of the most important aspects of Koopman MPC. The accuracy of the algorithm is strongly reliant on the availability of a collection of basis functions that can reflect the underlying system's nonlinear dynamics. RBFs were used in our research, but different embedding functions should be investigated and applied to the system. In the domain of control of the inverted pendulum on cart, trigonometric embedding can be explored [16].
- **Use of Koopman eigenfunctions:** One of the shortcomings of the DMD-type methods is that it is based on linear measurements and does not span a Koopman invariant subspace for many nonlinear systems. Koopman eigenfunctions provide a systematic linear embedding of nonlinear dynamics that results in an intrinsic coordinate system that can be closed using the Koopman operator. The eigenfunctions of the Koopman operators can also be generated for prediction and control as discussed in [24]. These eigenfunctions are optimization-based and no dictionary selection is required for them. This approach exploits the richness of the Koopman operator's spectrum to construct a collection of eigenfunctions such that the state is within the span of these characteristic functions, so is linearly predictable. A new algorithm is proposed to construct these Koopman eigenfunctions from the data. Then, these predictors are readily applied to an MPC framework for control. As part of future work, such an approach should be investigated and applied to implementation.

List of Figures

List of Tables

Glossary

ASDS Autonomous Spaceport Droneshop. 1

EDMD Extended Dynamic Mode Decomposition. 2, 12, 15

KMPC Koopman Model Predictive Control. 37, 42

MPC Model Predictive Control. 6, 16, 18, 26, 32, 33, 35, 38, 40, 48, 49

RBF Radial Basis Function. 15

RMSE Root Mean Square Error. 26, 31, 32

rRMSE relative Root Mean Square Error. 26, 32

SPRK Students, Parents, Robots and Kids. 2

Bibliography

[1] Ian Abraham, Gerardo de la Torre, and Todd Murphey. Model-based control using koopman operators. *Robotics: Science and Systems XIII*, Jul 2017.

[2] aerospace. A brief history of space exploration: The aerospace corporation. https://aerospace.org/article/brief-history-space-exploration, Jun 2018. Accessed: 2021-09-30.

[3] Ganbarov Ali. Autonomous spaceship navigation and landing using model predictive control. Master's thesis, Technical University of Munich, Department of Computer Science, Scientific Computing, 2020.

[4] Mike Allenspach and Guillaume Jacques Joseph Ducard. Nonlinear model predictive control and guidance for a propeller-tilting hybrid unmanned air vehicle. *Automatica*, 132:109790, 2021.

[5] Hassan Arbabi. Introduction to koopman operator theory of dynamical systems. In *Introduction to Koopman operator theory of dynamical systems*, 2018.

[6] Hassan Arbabi, Milan Korda, and Igor Mezic. A data-driven koopman model predictive control framework for nonlinear flows. https://www.mit.edu/~arbabi/research/KoopmanIntro.pdf, 2018. Accessed: 2021-09-30.

[7] Allan M. Avila and Igor Mezić. Data-driven analysis and forecasting of highway traffic dynamics. *Nature Communications*, 11, 2020.

[8] J.-P Bibring, H. Rosenbauer, H. Boehnhardt, S. Ulamec, Jens Biele, S. Espinasse, B. Feuerbacher, Philippe Gaudon, P. Hemmerich, P. Kletzkine, Dioc Moura, R. Mugnuolo, G. Nietner, Brigitte Pätz, R. Roll, H. Scheuerle, Karoly Szego, and K. Wittmann. The rosetta lander ("philae") investigations. *Space Science Reviews*, 128, 02 2007.

[9] Giovanni Binet, R. Krenn, and Alberto Bemporad. Model predictive control applications for planetary rovers. In *Model Predictive Control Applications for Planetary Rovers*, 01 2012.

[10] Lars Blackmore. Autonomous precision landing of space rockets. *The Bridge on Frontiers of Engineering*, 46:15–20, 01 2016.

[11] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer, 2007.

[12] Vít Cibulka, Milan Korda, Tomáš Haniš, and Martin Hromčík. Model predictive control of a vehicle using koopman operator, 03 2021.

[13] Ying Ding, Liang Wang, Yongwei Li, and Daoliang Li. Model predictive control and its application in agriculture: A review. *Computers and Electronics in Agriculture*, 151:104–117, 2018.

[14] N. Benjamin Erichson, Steven L. Brunton, and J. Nathan Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16(5):1479–1492, Nov 2016.

[15] Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6, 12 2014.

[16] Camilo García Tenorio, Gilles Delansnay, Eduardo Mojica-Nava, and Alain Vande-Wouwer. Trigonometric embeddings in polynomial extended mode decomposition—experimental application to an inverted pendulum. *Mathematics*, 9:1119, 05 2021.

[17] Michael Georgescu, Sophie Loire, Don Kasper, and Igor Mezic. Whole-building fault detection: A scalable approach using spectral methods, 2017.

[18] M Golombek, R. Cook, Thanasis Economou, W. Folkner, A. Haldemann, P Kallemeyn, J Knudsen, R Manning, H Moore, Tim Parker, R Rieder, J. Schofield, Peter Smith, and R Vaughan. Overview of the mars pathfinder mission and assessment of landing site predictions. *Science (New York, N.Y.)*, 278:1743–8, 01 1998.

[19] Yingbai Hu, Hang Su, Junling Fu, Hamid Reza Karimi, Giancarlo Ferrigno, Elena De Momi, and Alois Knoll. Nonlinear model predictive control for mobile medical robot using neural optimization. *IEEE Transactions on Industrial Electronics*, 68(12):12636–12645, 2021.

[20] Nicolas Hudon, Martin Guay, Michel Perrier, and Denis Dochain. Nonlinear model predictive control for optimal discontinuous drug delivery. *IFAC Proceedings Volumes*, 39(2):527–532, 2006. 6th IFAC Symposium on Advanced Control of Chemical Processes.

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

[22] Milan Korda. Koopman model predictive control of nonlinear dynamical systems. University Lecture, https://homepages.laas.fr/mkorda/slides/Caltech_2018.pdf, 2018. Accessed: 2021-09-30.

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

[24] Milan Korda and Igor Mezić. Optimal construction of koopman eigenfunctions for prediction and control, 2020.

[25] Milan Korda, Yoshihiko Susuki, and Igor Mezić. Power grid transient stabilization using koopman model predictive control, 2018.

[26] Xinfu Liu. Fuel-optimal rocket landing with aerodynamic controls. *Journal of Guidance Control and Dynamics*, 42, 09 2018.

[27] Alexandre Mauroy, Igor Mezic, and Yoshihiko Susuki. *The Koopman Operator in Systems and Control Concepts, Methodologies, and Applications: Concepts, Methodologies, and Applications*. Springer, 01 2020.

[28] J. Meiss. Dynamical systems. *Scholarpedia*, 2(2):1629, 2007. revision #137210.

[29] Solomon Oyelere. The application of model predictive control (mpc) to fast systems such as autonomous ground vehicles (agv). *IOSR Journal of Computer Engineering 2278-0661*, 3:27–37, 06 2014.

[30] Ph. Poignet and M. Gautier. Nonlinear model predictive control of a robot manipulator. In *6th International Workshop on Advanced Motion Control. Proceedings (Cat. No.00TH8494)*, pages 401–406, 2000.

[31] Lal Prasad, Barjeev Tyagi, and Hari Gupta. Optimal control of nonlinear inverted pendulum system using pid controller and lqr: Performance analysis without and with disturbance input. *International Journal of Automation and Computing*, 11:661–670, 12 2014.

[32] Derek Rowell. State-space representation of lti systems. In *State-Space Representation of LTI Systems*, 2002. Accessed: 2021-09-30.

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

[34] David Shim, Hong Joo Kim, and Shankar Sastry. Decentralized nonlinear model predictive control of multiple flying robots. In *Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots*, volume 4, pages 3621 – 3626 vol.4, 01 2004.

[35] Yongqiang Sun, Yuan Hu, Yuegang Fu, Yueqi Wang, Qi Wang, and Yu Zhao. Image analysis with the dmd in convergent path. In *2018 IEEE International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3M-NANO)*, pages 386–389, 2018.

[36] M. Tomasko, D. Buchhauser, M. Bushroe, L. Dafoe, L. Doose, Andrew Eibl, Chuck Fellows, E. Farlane, G. Prout, M. Pringle, B. Rizk, Chirus See, Peter Smith, and K. Tsetsenekos. The descent imager/spectral radiometer (disr) experiment on the huygens entry probe of titan. *Space Science Reviews*, 104:469–551, 07 2002.

[37] Cong Wang and Zhengyu Song. Convex model predictive control for rocket vertical landing. In *2018 37th Chinese Control Conference (CCC)*, pages 9837–9842, 2018.

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

[39] Guillermo Zaragoza Prous. Guidance and control for launch and vertical descend of reusable launchers using model predictive control and convex optimisation. Master's thesis, Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, 2020.

[40] Wei Zhang, Yao-Chi Yu, and Jr-Shin Li. Dynamics reconstruction and classification via koopman features. *Data Mining and Knowledge Discovery*, 33:1710 – 1735, 2019.