THESIS

MODELING, SIMULATION, AND CONTROL OF SOFT ROBOTS USING KOOPMAN OPERATOR THEORY

Submitted by

Ajai Singh

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2023

Master's Committee:

Advisor: Edwin K.P. Chong

Jianguo Zhao
Sudeep Pasricha

ABSTRACT

MODELING, SIMULATION, AND CONTROL OF SOFT ROBOTS USING KOOPMAN OPERATOR THEORY

In nature, animals with soft body parts can control their parts to different shapes, e.g., an elephant trunk can wrap on a tree branch to pick it up. But most research on manipulators only focuses on how to control the end effector, partly because the arm of the manipulator is rigidly articulated. With recent advances in soft robotics research, controlling a soft manipulator into many different shapes will significantly improve the robot's functionality, such as medical robots morphing their shape to navigate the digestive system and then delivering drugs to the required location. However, controlling the shape of soft robots is challenging since the dynamics of soft robots are highly nonlinear and computationally intensive.

In this research, we leverage a data-driven method using the Koopman operator to realize the shape control of soft robots. The dynamics of a soft manipulator are simulated using a physics-based simulator (PyElastica) to generate the input-output data. The data is used to identify an approximated linear model based on the Koopman operator. We then formulate the shape-control problem as a convex optimization problem that is computationally efficient. We demonstrated the linear model is over 12 times faster than the physics-based model in simulating the manipulator's motion. Further, we can control a soft manipulator into different shapes using model predictive control (MPC), and then in the subsequent chapters, we build a soft grid consisting of 40 such soft manipulators. We then address the issues related to the Extended Dynamic Mode Decomposition (EDMD) algorithm used for approximating the Koopman operator by developing a deep learning-based framework to learn the Koopman embeddings. On comparing the EDMD and deep learning framework it was found that the deep learning framework was far more accurate than the EDMD framework We then show that the proposed methods can be effectively used to control the shapes

of soft robots by having the single soft manipulator morph into "C", "S", and "U" shapes and then extend the shape control method to the soft grid by morphing it into 3 different shapes. We envision that shape control will allow the soft robots to interact with uncertain environments or the shapes of shape-morphing robots to fulfill different tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Soft robots, also known as compliant or flexible robots, are a rapidly growing field in robotics that aims to create robots with bodies that can adapt to their environment. Unlike traditional robots, which are typically made of rigid materials such as metal and plastic, soft robots are typically made from materials such as silicone, rubber, or other flexible polymers. These materials allow the robot to change shape and move in ways that are not possible with traditional robots. Traditional robots are made of rigid materials as a consequence of which they have finite degrees of freedom and are very precise and accurate. However, these rigid-body robots still lack the ability to adapt and have safe interactions with humans. Soft robots have the inherent properties to bridge this gap.

The materials of soft robots are very soft and compliant and somewhat analogous to the ones found in living creatures [1]. The field of soft robotics generally draws its inspiration from the biological beings in nature and the soft robots are designed to accomplish either locomotion or adapt to a new environment or both. For example, an octopus that does not have a skeletal system can easily deform its body structure and squeeze itself through openings that are much smaller when compared to its body size. Moth larvae change their body shape into a circular ring to roll away from predators [2]. Moreover, soft robots absorb more energy during a collision due to their soft composition making them inherently safe when operating in close proximity or alongside humans.

Another advantage of soft robots is their ability to sense and respond to the external environment. Many soft robots including electronic skins [3] and soft robotic fingers [4] have embedded sensors that allow them to detect changes in their external environment and adapt accordingly. This makes soft robots the best choice when it comes to the application where there is a need for exploration and requires the robot to autonomously navigate an unknown terrain.

One might think if soft robots have so many advantages over rigid-bodied robots then why don't we see more soft robots in real life? The answer might lie in methods used in the modeling and control of soft robots. Unlike rigid-bodied robots which have finite degrees of freedom and rigid structure, soft robots have infinite degrees of freedom and deformable structure. Now to be able to exploit soft robots of their inherent properties, one has to take into account the properties of materials, and infinite degrees of freedom. These are some of the many factors that pose difficulty when it comes to analytically modeling a soft robot without having to make simplifying assumptions.

1.2 Modeling of soft robots

Modeling soft robots is a very crucial step in the design of a soft robot. Modeling these machines involves simulating the behavior of these robots in different operating conditions and scenarios. Finite Element Analysis (FEA) [5] is a numerical method used for modeling soft robots. The FEA method is used to numerically solve partial differential equations that describe the geometry, deformation, and material properties of a soft robot. The primary drawback of FEA is the cost of computation, various methods have been developed to model and simulate soft objects in real time using GPU, and Model Order Reduction has also shown promising results when it comes to speeding up the simulation. Multibody dynamics (MBD) [6] is another method that can be used to simulate the behavior of a soft robot, it is used to simulate the interactions between different parts of a soft robot's body such as joints and actuators. However, these methods are computationally expensive.

To increase the computational efficiency reduced order models such as pseudo-rigid mechanics and simplified geometry have been tested on real soft robots, but the problem with these methods is that they hold good over a range in which these simplifying assumptions hold good and when operating outside the subset in which these assumptions were made, they fail to accurately describe the behavior of the robotic system. Once a soft robot model is developed, it can be used to optimize the robot's design. This can be done by changing the robot's parameters, such as its shape or

material properties, and then simulating its behavior to see how the changes affect its performance. This process can be repeated until the optimal design is found.

Data-driven methods for modeling of soft robots are becoming increasingly popular in the field of robotics. These methods use data collected from experiments or simulations to create models of soft robots that can predict their behavior under different conditions.

One of the main advantages of data-driven methods is that they can be used to model soft robots that are too complex or difficult to model using traditional methods. For example, data-driven methods can be used to model the behavior of soft robots made of complex materials, such as hydrogels or elastomers, which can be difficult to simulate using traditional methods. Another advantage of data-driven methods is that they can be used to model the behavior of soft robots in real-world environments. This is important because soft robots are designed to adapt to their environment, and their behavior can be affected by factors such as temperature, humidity, and other environmental conditions.

Machine learning techniques, such as artificial neural networks, are used to model soft robots by training the network on data collected from experiments or simulations. The network can then be used to predict the behavior of the soft robot under different conditions. For example, a neural network can be trained on data collected from experiments where a soft robot is subjected to different loads and then used to predict how the robot will behave under different loads. For example, [7] used machine learning to control the position and orientation of the tip of a soft robot and also demonstrated the power of machine learning techniques by tracking a randomly moving object with the tip of a soft robot in different conditions which included the introduction of obstacles in the simulation environment.

Although data-driven methods have several advantages, they pose many challenges as well. One of the main challenges is the huge volumes of data required to train the model. This can be difficult to obtain, especially when it comes to soft robots, as they can be difficult to test and measure. Another challenge is that data-driven methods can be computationally intensive, which

can make them difficult to use in real-time applications. Additionally, the quality of the data, and the way it is collected, can affect the accuracy of the model.

One of the methods used to overcome this problem is the use of the Koopman operator [8] for the system identification of nonlinear systems. The Koopman operator is a mathematical tool that can be used to perform system identification on nonlinear systems, such as soft robots. The Koopman operator is an infinite-dimensional linear operator that describes the evolution of a system over time. It can be used to represent the dynamics of a system in a high-dimensional space, which allows for the analysis of nonlinear systems. The Koopman operator can be used to estimate the parameters of a system by analyzing the data collected from experiments or simulations.

One of the main advantages of using the Koopman operator for system identification is that it can be used to model nonlinear systems, such as soft robots. Traditional system identification methods, such as linear regression, are limited to linear systems, and cannot be used to model nonlinear systems. The Koopman operator, on the other hand, can be used to model nonlinear systems, which makes it well-suited for modeling soft robots. Another advantage of using the Koopman operator for system identification is that it can be used to estimate the parameters of a system in real-time. This is important for applications where the system needs to adapt to changing conditions, such as in a soft robot that needs to respond to its environment. The Koopman operator can be used to estimate the parameters of a system in real-time, which allows the system to adapt to changing conditions.

There are different methods to compute the Koopman operator from data, such as Dynamic Mode Decomposition (DMD) [9], Extended Dynamic Mode Decomposition (EDMD) [10] and Kernel Dynamic Mode Decomposition (KDMD) [11] among others. These methods use different techniques to estimate the Koopman operator from data, but all of them have the advantage of being able to model nonlinear systems. However, using the Koopman operator for system identification also poses certain challenges. One of the main challenges is that the Koopman operator is an infinite-dimensional operator, which can make it difficult to compute and analyze. Additionally, the methods used to compute the Koopman operator from data are often computationally intensive,

which can make it difficult to use in real-time applications. We will be addressing the Koopman operator in detail in this thesis.

1.3 Shape control of soft robots

One approach to control the shape in soft robots is to use embedded actuators, such as pneumatic or hydraulic systems, that can change the shape of the robot's body [12, 13]. These actuators can be controlled by an onboard computer or another electronic control system, which allows the robot to move and change shape in response to its environment. Another approach is to use materials that are responsive to external stimuli, such as temperature or pH, to change shape [14]. These materials can be incorporated into the robot's body and controlled by an external system, such as a computer or remote control.

One of the greatest advantages of soft robots is their ability to adapt to their environment. For example, they can navigate through tight spaces or navigate around obstacles. They can also be used in applications such as search and rescue, where they can navigate through rubble or other hazardous environments. Their soft body also allows them to interact with humans in a more natural way.

The capacity of soft robots to be integrated with living things is another advantage. They can be used in applications such as medical devices, which can be inserted into the human body and controlled remotely. They can also be used to study the movement and behavior of living organisms in a controlled environment.

In conclusion, a major obstacle to the advancement of soft robots is shape control. However, advances in materials and control systems have made it possible to create soft robots that can adapt to their environment and interact with living organisms. Soft robots have the potential to revolutionize a wide range of industries, from healthcare to search and rescue. As research and development in this field continue, we can expect to see even more advanced and versatile soft robots in the future.

1.4 Outline and Contributions

In this thesis, we design a data-driven approach that can be applied to model a soft robot. To demonstrate the accuracy of the framework we apply it to PyElastica [7], PyElastica is a physics-based simulation engine for modeling and simulation of a soft robot. This simulation engine is based on the Cosserat rod theory and can be used to model a single or a combination of rods to generate more complex robotics systems. Chapter 2 discusses in detail the data-driven modeling technique using the Koopman operator. Using Extended Dynamic Mode Decomposition (EDMD) [10] to approximate the finite-dimensional Koopman operator we have made sure that model approximated is accurate enough to replace the actual physics-based model. To begin with the testing of the framework we test it on a simple Cantilever beam oscillating under gravity, we then use the PyElastica modeling engine to model a single soft manipulator and later build on the concept to extend it to a system of multiple rods connected in a grid fashion giving rise to 4X4 grid consisting of 40 such soft manipulators in chapter 3. We then formulate the shape control problem as an optimal control problem in chapter 4. After the successful modeling of the soft robots and formulation of the shape control problem, we to control the shape of the single soft manipulator and also the 4X4 soft grid formed, and then chapter 5 discusses the current limitations of the methods developed in this dissertation along with future work and scope.

There are two major contributions of this thesis. The first contribution is to develop a deep learning framework that automatically learns the right set of lifting functions for finite-dimensional Koopman operator approximation from data. As a consequence of this, we can easily develop a linear model for a soft robot. The second contribution is to develop closed-loop control using Linear Model Predictive Control theory to control the shape of two different soft robots (i.e., a soft manipulator and a soft grid).

Chapter 2

Data-driven Modeling with Koopman Operator

Despite soft robots being the heart of various industries such as manufacturing, health care, food preparation, etc. soft robots have recently emerged in robotics research. Soft robots unlike their rigid counterparts are made up of soft materials making them lightweight and robust. Soft robots have far more degrees of freedom compared to rigid-bodied robots which opens door to many fields in which soft robots can be of excellent use. For example, soft robots can leverage their inherent mechanical compliance to interact with humans or the external environment.

However, unlike rigid-bodied robots, design and modeling methods for soft robots are lacking. The present modeling techniques cannot adapt well to the heterogeneous design of soft robots due to the difference in the compliance of the underlying materials [15]. Soft robots are infinite-dimensional systems whose behavior with time is described by highly non-linear partial differential equations, which cannot be easily analyzed analytically. Due to this fact, generally simplifying assumptions are made for developing a mathematical model of a soft robot. However because of these simplifying assumptions, one cannot exploit the inherent properties of the soft robot to its full potential.

To address these modeling challenges with soft robots: data-driven modeling techniques have emanated as a very powerful tool. The major advantage of using data-driven techniques for modeling a soft robot is that they do not require any simplifying assumptions such as quasi-static [16] behavior and pseudo-rigid body mechanics [17] and are solely based on input-output data. However, there is also a problem associated with the data-driven modeling technique as it requires data collection from the model across various operating conditions. When compared to rigid-bodied robots, soft robots may be operated under a wide range of operational circumstances without putting them at risk or causing a threat to the environment. Figure 2.1 gives a pictorial representation of physics-based modeling vs data-driven modeling and figure 2.2 provides a block diagram of the steps involved in using data-driven modeling.

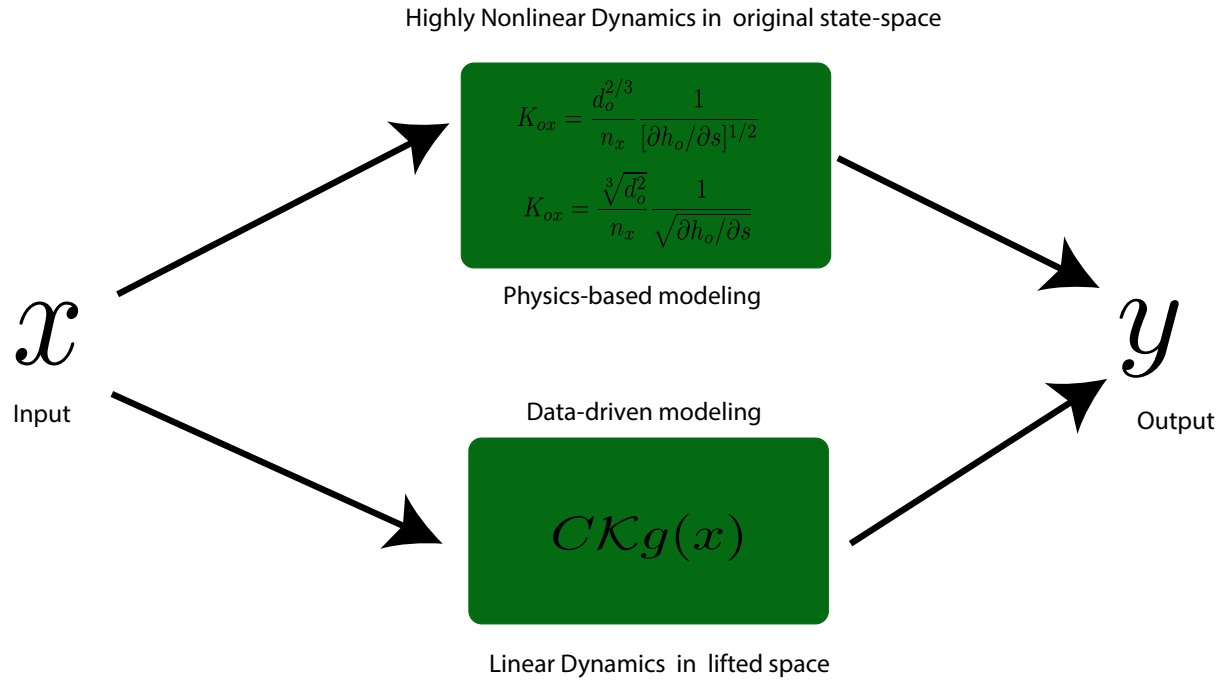


Figure 2.1: This figure gives a pictorial representation of Physics-based modeling and Data-driven modeling. Studying the picture we find that to arrive at the same output y, one has to solve a highly nonlinear dynamical system. This process is not only computationally expensive but there are chances that an analytical solution might not even exist whereas, in the case of data-driven modeling, we can arrive at the same output y in a computationally efficient manner as the dynamics, in this case, are linear in the lifted space and the physics of the system is not required.

Figure 2.2: Figure shows the steps involved in data-driven modeling using the Koopman operator approach. In the case of data-driven modeling, data is collected either by running the experiment or by simulation, and a database of input-output data is created. Then the next step of data-driven modeling is using a framework which can be any system identification framework such as a Neural Network or Koopman operator (used in the figure) to approximate a model that maps the input data to output data. In case the system identification framework is based on the Koopman operator then the resulting model is a Linear Model.

In this chapter, we use a data-driven modeling technique to derive a linear model of the PyElastica [18] soft robot using the Koopman operator theory. The Koopman operator-based system identification relies on lifting the state space of the original dynamical system into an infinite-dimensional function space where the time evolution of the original dynamical system can be described in a linear fashion. The operator that linearly describes the original non-linear dynamical system is called the Koopman Operator.

The rest of the chapter is organized as follows: In section 2.1 we formally introduce the Koopman Operator and cover finite dimensional approximation of the Koopman operator using the EDMD algorithm. Then section 2.2 shows the application of the Koopman-based linear system identification to PyElastica [7] soft robot and section 2.3 concludes the chapter.

2.1 Koopman Operator-based system Identification

In this section, we describe a method based on the Koopman operator to construct a space-state model of a non-autonomous non-linear dynamical system from input-output data. The Koopman operator describes the evolution of a dynamical system in a linear fashion globally unlike linearization about a point where the dynamics fail to describe the behavior of the system when operating away from the point of the linearization. Instead of describing the evolution of the states of the dynamical system, the Koopman operator describes the evolution of the scalar-valued functions of the states. The Koopman operator can be approximated using least-square regression on the data set. Once we have a matrix approximation of the Koopman operator we can derive a state space model of the dynamical system from the Koopman matrix. The following subsections will describe these processes in detail.

2.1.1 The Koopman Operator

Koopman operator theory can be used to construct a linear model of a forced nonlinear system in an infinite dimensional Hilbert space from input-output data of the nonlinear system. With the constructed linear model, we can directly use existing linear system control techniques. The

Koopman operator approach is undeniably becoming an increasingly popular data-driven method for the control of nonlinear dynamical systems. The theoretical underpinnings of the Koopman Operator were formulated by Bernard Koopman in [8], and this framework became popular after the development of the Dynamic Mode Decomposition algorithm [19].

Given a nonlinear dynamical system, the Koopman operator first maps the states of the original system using scalar functions (also called observables) of the states into a so-called lifted space with new state variables. The new system in the lifted space with the new state variables is an infinite dimensional linear system. Unlike the linearization about a point that becomes inaccurate when operating away from the linearizing point, the Koopman operator describes the evolution of the scalar observable throughout the state space in a linear fashion. This makes the Koopman operator approach preferable when realizing linear representation of nonlinear systems [20].

Consider a discrete-time autonomous nonlinear dynamical system given by:

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k)) \\ y(t_k) &= g(x(t_k)) \end{aligned} \tag{2.1}$$

where $x(t_k)$, $x(t_{k+1}) \in \mathbb{R}^n$ are the state vector at time step t_k, t_{k+1} respectively, $y(t_k) \in \mathbb{R}^r$ is the output of the system at time instant t_k. To simplify notations, in the following, we use $x(t_k)$ and x_{t_k} interchangeably.

To map the state x to a lifted space, we use a basis or observation function $\phi(x(t_k)) : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{F}$, where $\mathcal{F}$ is the space of all basis functions. The Koopman operator $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$ is defined as:

$$(\mathcal{K}\phi)(x_{t_k}) = \phi(f(x_{t_k})) \tag{2.2}$$

From equation 2.1 we can re-write the above equation as :

$$(\mathcal{K}\phi)(x_{t_k}) = \phi(x_{t_k+1}) \tag{2.3}$$

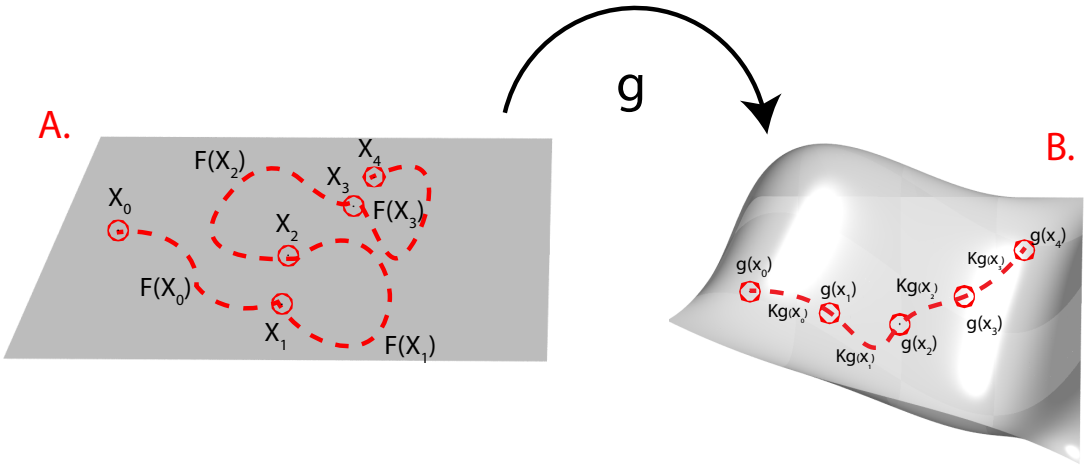


Figure 2.3: This figure is the graphical representation of the Koopman operator. **A.** The evolution of the states of a nonlinear dynamical system is shown in the picture on left. Then the original states are lifted using a set of observables to a different functional space (generally of higher dimension when compared to the original state space.) **B.** The evolution of the observables is described in a linear fashion by a linear operator K and this linear operator is called the Koopman operator.

The above equation states that the Koopman operator updates observation of the state in the lifted space from the current time step to the next step.

$\mathcal{K}$ is an infinite dimensional linear operator [8], but a finite-dimensional approximation of the Koopman operator can be approximated using a finite subspace. The fact that the Koopman operator is a linear operator can be proved in the following manner:

Consider the discrete-time definition given by:

$$x_{n+1} = T(x_n) \tag{2.4}$$

Let the Koopman operator associated with the discrete-time system defined in (2.4) be U. To prove that U is a linear operator, it is sufficient to show that U satisfies:

$$U[c_1 g_1(x) + c_2 g_2(x)] = c_1 U g_1(x) + c_2 U g_2(x) \tag{2.5}$$

where c_1, c_2 are constants and g_1, g_2 are observables.

Proof. Let $h(x) = c_1 g_1(x) + c_2 g_2(x)$,

Then L.H.S. of (2.5) can be written as: $Uh(x)$. It follows from the definition of the Koopman operator (**??**) that for the system defined in (2.4):

$$\begin{aligned} Uh(x) &= h(T(x)) \\ &= c_1 g_1(T(x)) + c_2 g_2(T(x)) \\ &= c_1 U g_1(x) + c_2 U g_2(x) \end{aligned} \tag{2.6}$$

Hence proved that $U[c_1 g_1(x) + c_2 g_2(x)] = c_1 U g_1(x) + c_2 U g_2(x)$ □

Before we see how a finite-dimensional approximation of the Koopman operator can be achieved from data, Let us take a motivational example to observe the potential of using the Koopman operator to describe a non-linear dynamical system. Consider the autonomous dynamical system inspired by [21]:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a x_1 \\ b x_2 + (b - a^2) x_1^2 \end{bmatrix} \tag{2.7}$$

where $a, b \in [0, 1]$.

It is very interesting to see that the same dynamical system as described above can be represented in a linear fashion if one can find the set of "good" observables. In this case, this set of observables is relatively easy to find. If we choose to have the following set of observables given by:

$$\Psi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix} \tag{2.8}$$

Then the dynamical system described in (2.7) can be represented by the following equation:

$$\dot{z} = Az$$

$$y = Cz$$

where: $z = \Psi(x)$ and $C = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$

2.1.2 Linear System Identification based on Koopman Operator

The Koopman operator $\mathcal{K}$ defined in (**??**) acts over a functional space defined by the set of observables, hence it is an infinite dimensional operator. Therefore we have to approximate a finite-dimensional representation of the Koopman operator for practical implementation on a digital computer. Let the finite dimensional approximation of $\mathcal{K}$ be $\bar{\mathcal{K}}$. Now, $\bar{\mathcal{K}}$ operates on $\bar{\mathcal{F}}$, where $\bar{\mathcal{F}} \subset \mathcal{F}$. Here $\mathcal{F}$ is the subspace spanned by the observable functions also known as basis functions.

A finite-dimensional approximation of the Koopman operator can be obtained using the Extended Dynamic Mode Decomposition (EDMD). EDMD makes use of a set of pre-defined lifting functions to lift the original state space to the higher dimensional function space. Then the following optimization problem is solved to extract the Koopman matrix:

$$\underset{A}{\text{minimize}} \quad \sum_{k=0}^{K-1} ||\psi(x_{t_{k+1}}) - A\psi(x_{t_k})||_2^2 \tag{2.9}$$

where:

$\Psi(x_{t_k}) = \begin{bmatrix} \psi_1(x_{t_k}) \\ \psi_2(x_{t_k}) \\ \psi_3(x_{t_k}) \\ \vdots \\ \psi_{N_c}(x_{t_k}) \end{bmatrix}$, $\{\psi_i : \mathbb{R}^n \to \mathbb{R}\}_{i=1}^{N_c}$ Here Ψ_i are the lifting functions

N_c is the total number of lifting functions,

$A \in \mathbb{R}^{N_c \times N_c}$ is the finite-dimensional approximation of the Koopman Operator,

K is the cardinality of the dataset given by $\mathcal{D} = \{x_{t_k}\}_{k=0}^{K}$

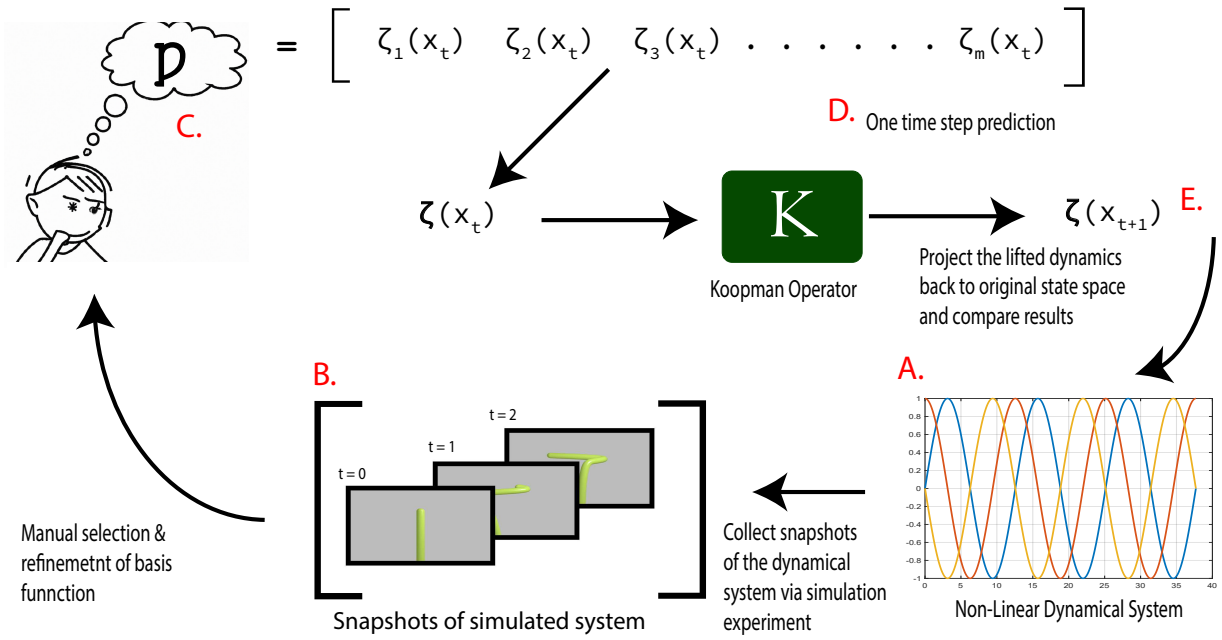


Figure 2.4: This figure demonstrates the Extended Dynamic Model Decomposition Algorithm pictorially. **A.** Sufficient data is collected either by running an actual experiment or by simulating the nonlinear dynamics of the system. **B.** Snapshots of the system measurements are collected and stored. **C.** Then the researcher is required to figure out the right set of basis functions to lift the dynamics. This is an iterative and time-consuming process until one finds the right set of basis functions. **D.** The Koopman operator is approximated using least-square regression. **E.** Once the Koopman operator is approximated then it is used to linearly describe the nonlinear dynamics in the lifted state and at each time step the lifted states are projected back to the original state space using a projection operator.

Solving the problem of (2.9) we can represent the nonlinear dynamical system described in (2.1) as the following discrete-time linear dynamical system:

$$\begin{aligned} z(t_{k+1}) &= Az(t_k) \\ \tilde{y}(t_k) &= Cz(t_k) \end{aligned} \tag{2.10}$$

where $z(t_k) = \psi(x(t_k)) \in \mathbb{R}^{N_c}$ and $\tilde{y}(t_k)$ is the output. The matrix $C \in \mathbb{R}^{r \times N_c}$ is obtained just like A by solving the following minimization problem

$$\underset{C}{\text{minimize}} \quad \sum_{k=1}^{K} ||y(t_k) - C\psi(x(t_k))||_2^2 \tag{2.11}$$

Similarly, for a nonautonomous nonlinear dynamical system with control inputs, the methodology discussed can be used. Consider a discrete nonlinear dynamical system given by

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k), u(t_k)) \\ y(t_k) &= g(x(t_k)) \end{aligned} \tag{2.12}$$

where $u(t_k) \in \mathbb{R}^m$ where m is the dimension of control inputs of the dynamical system. Then in this case to approximate the Koopman Operator, the minimization problem in (2.9) changes to

$$\underset{A,\,B}{\text{minimize}} \quad \sum_{k=0}^{K-1} ||\psi(x(t_{k+1})) - (A\psi(x(t_k)) + Bu(t_k))||_2^2 \tag{2.13}$$

Thus, by solving the minimization problem of (2.13), the finite approximation of the Koopman operator is approximated by *A* and *B*, and it acts as one step predictor of the nonlinear dynamical system described by (2.12). The minimization problem for the output equation i.e. for *C* matrix remains the same as given by (2.11). It is advisable to include velocities in the states when identifying the dynamics of a mechanical system [6]. These can be included by modifying the domain of the basis function such that $\{\psi_i : \mathbb{R}^{n+nd+md} \to \mathbb{R}\}_{i=1}^{N_c}$, where d is the number of delays.

2.1.3 Application of Koopman operator-based model approximation

We assert that the Koopman operator can be used to approximate linear models of a non-linear dynamical system. Rather than applying the Koopman operator-based system identification to a soft robot, we first test the framework on a cantilever beam oscillating in a two-dimensional plane under gravity. The Physical properties of the cantilever beam are listed in the table below:

Table 2.1: Physical Properties of Cantilever Beam

S.No.	Physical Property	Value	Unit
1.	Length of beam	0.4	meter
2.	Modulus of Elasticity	207E9	Pa
3.	Density of beam	8000	kg/m^{-3}
4.	Radius of beam	0.0012	meter

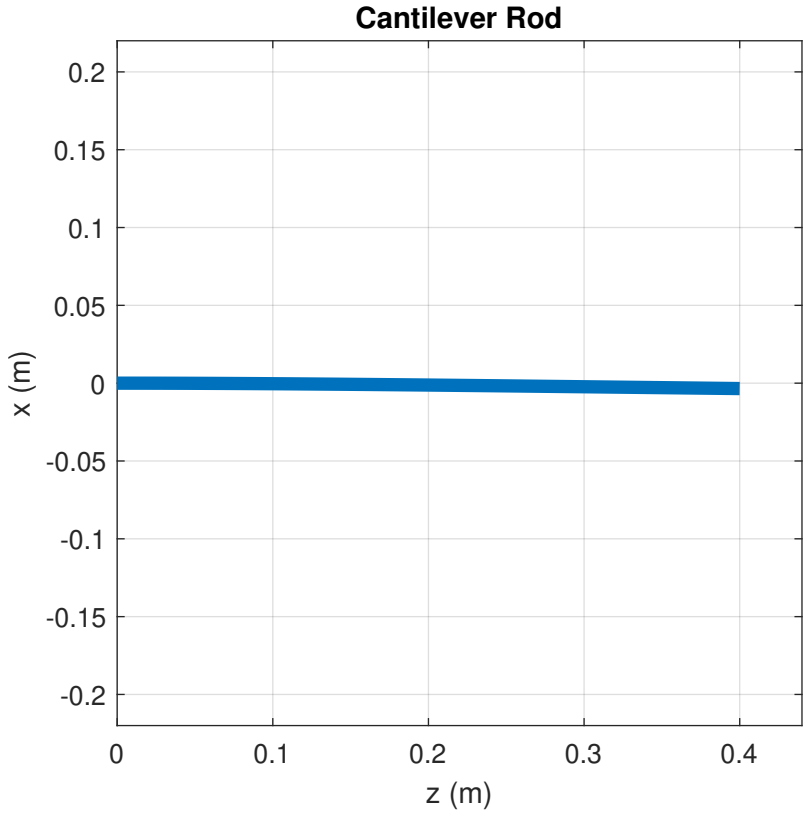


Figure 2.5: Schematic representation of the cantilever rod oscillating in the X-Z plane under gravity. The value of acceleration due to gravity was taken to be 9.81 and is acting in the negative X direction. The snapshot was taken at t = 0.

The data collection for model approximation was carried out by numerically simulating the dynamics of the cantilever beam with physical properties described in Table 2.1 for a duration of 2 seconds with time-step $dt = 0.001s$. The value of acceleration due to gravity used was $g = 9.81m/s^2$.

Given below are the results of the model approximation using the Koopman operator:

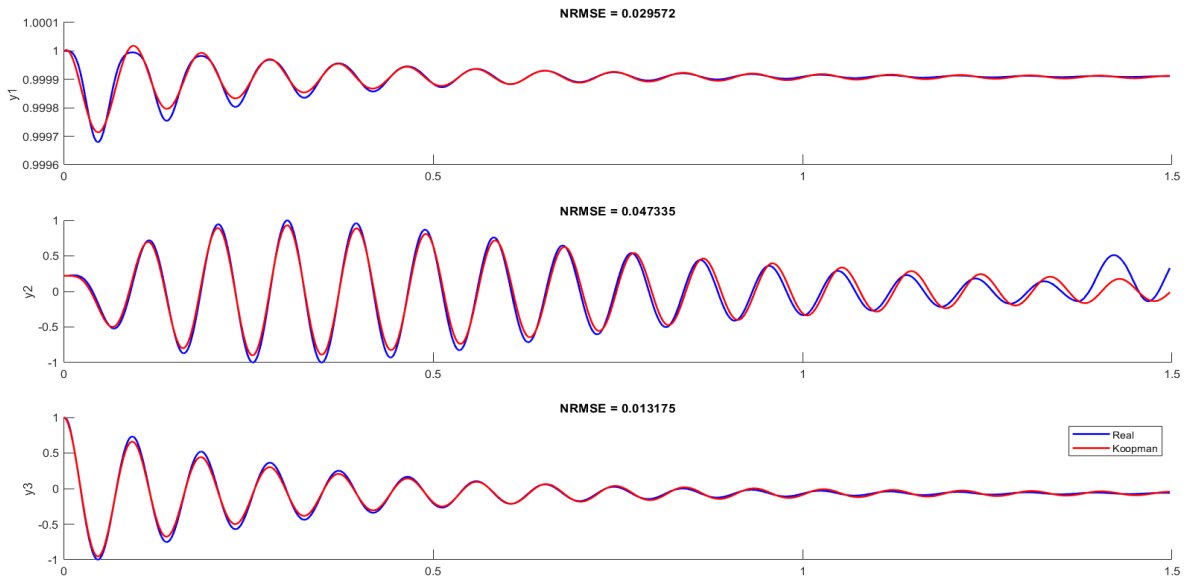


Figure 2.6: Plot showing the prediction of the Koopman-based model (in Red) and actual physics-based model(in Blue) of a simple Cantilever beam oscillating under gravity in the X-Z plane. Looking at the plots we can easily conclude that the Koopman-based approximation of the nonlinear system(Cantilever beam) is very accurate to the actual physics-based model. Please note that the data has been scaled between [-1,1] for convenience.

2.2 Application of Koopman operator-based model approximation to PyElastica Soft robot

2.2.1 Setup for PyElastica

When using PyElastica, we need to set up a simulation where the user is required to define a system of rods, set up initial and boundary conditions on the rod, run the simulation, and collect data for post-processing. The detailed process can be found in [22].

The physical parameters defined for our manipulator are listed in Table 2.2. We fix the base of the robot as the boundary condition. The actuation of the manipulator is achieved by applying torques distributed along the length of the arm. The torques are decomposed into orthogonal torque functions in the local normal and bi-normal directions. The magnitude of the torques in a direction is obtained via continuous splines characterized by N independent control points. In our case, we don't apply the torque in the tangent (axial) direction to twist the robot.

Table 2.2: List of physical parameters for the simulation setup

Physical Parameter	**Value**	**Unit**
Number of tracking points	6	N/A
Starting Position of the rod (vector)	[0.0, 0.0, 0.0]	N/A
Direction in which rod extends(vector)	[0.0, 0.0, 1.0]	N/A
Normal vector of rod	[1.0, 0.0, 0.0]	N/A
Length of rod	1.00	meter (m)
Radius of the tip	0.05	meter (m)
Radius of the base	0.05	meter (m)
Density of the rod	1.0×10^3	kg/m^3
Energy dissipation constant	10.00	N/A
Youngs Modulus	1.00×10^7	Pa
Poisson's Ration	0.50	N/A

*The vectors are defined with respect to the standard right-hand coordinate system.

Note that spline control points for normal and binormal directions are different, and two splines are needed. To generate the two splines for each simulation case, we randomly generate two torques in normal and binormal directions for each control point. In the initial state, the robot is in a straight and upright shape. In the simulation, the two torque splines are kept constant as a step input for the system. We use the position Verlet algorithm as the time stepping algorithm and time step $\Delta t = 0.01$ s to simulate 20 s for each case, which can ensure the soft robot reaches a steady state final shape.

2.2.2 Data collection for system identification

Koopman Operator can be used to construct a linear model of the soft robotic system constructed in section 2.1. To do this, we collect data for system identification by simulating the robot with random input.

We simulate the manipulator for a total of 30 cases, For each simulation case, we collect 2000 snapshots with a sampling time of $T_s = 0.01$ s. These data sets are used for approximating the

Koopman Operator. The system identification was carried out by lifting the collected snapshots using the basis function with delays defined in section III. We used first-order polynomials with delay $d = 1$ as the basis function and then performing least square regression as shown in (10) and (8) to obtain the A, B, and C matrices, respectively [6] with $A \in \mathbb{R}^{47\times47}$, $B \in \mathbb{R}^{47\times10}$, and $C \in \mathbb{R}^{18\times47}$. For the problem, we have chosen a total of 47 basis functions with degree 1. For the 47 functions, we defined the first 18 to be the output of the system, then 28 are the delay coordinates, and a constant 1 has been added so that we do not lift the inputs. One might simply think to add higher degree polynomials or more number of basis functions to minimize the prediction error by the Koopman-based linear model $\dot{\phi}(x) = A\phi(x)$, but with the increase in the order of the polynomial, there are more functions in ϕ, then more derivatives $\dot{\phi}$ have to be expressed by ϕ. As a consequence of increasing the order of the polynomials, the derivatives $\dot{\phi}$ grow in complexity which makes it harder for $\dot{\phi}$ to be expressed by ϕ [23].

2.2.3 Koopman operator based Model approximation results

The accuracy of the Koopman Model is estimated by calculating the error that is defined as the Euclidean distance between the predicted output and the output of the physics-based model. The accuracy of the model approximated by the Koopman operator depends on the number of basis functions and the types of basis functions. We validated the identified linear model against 6 different data sets generated with the same method as mentioned above.

The linear model predicted by the Koopman operator has a maximum mean RMSE of 3.5×10^{-3} meters and a minimum mean RMSE of 6.78×10^{-4} across all states. Fig. 2.7 shows the maximum and minimum mean prediction error between the Koopman-based linear model and Physics based model. The prediction error for other validation cases lies between the maximum mean and minimum mean error mentioned. Here the mean prediction error at time-step t_i is defined as:

$$Error_{mean}(t_i) = \frac{1}{L}\sqrt{\frac{1}{N_x}\sum_{j=1}^{N_x}((x^j(t_i) - x^{j_{ref}}(t_i))^2)} \tag{2.14}$$

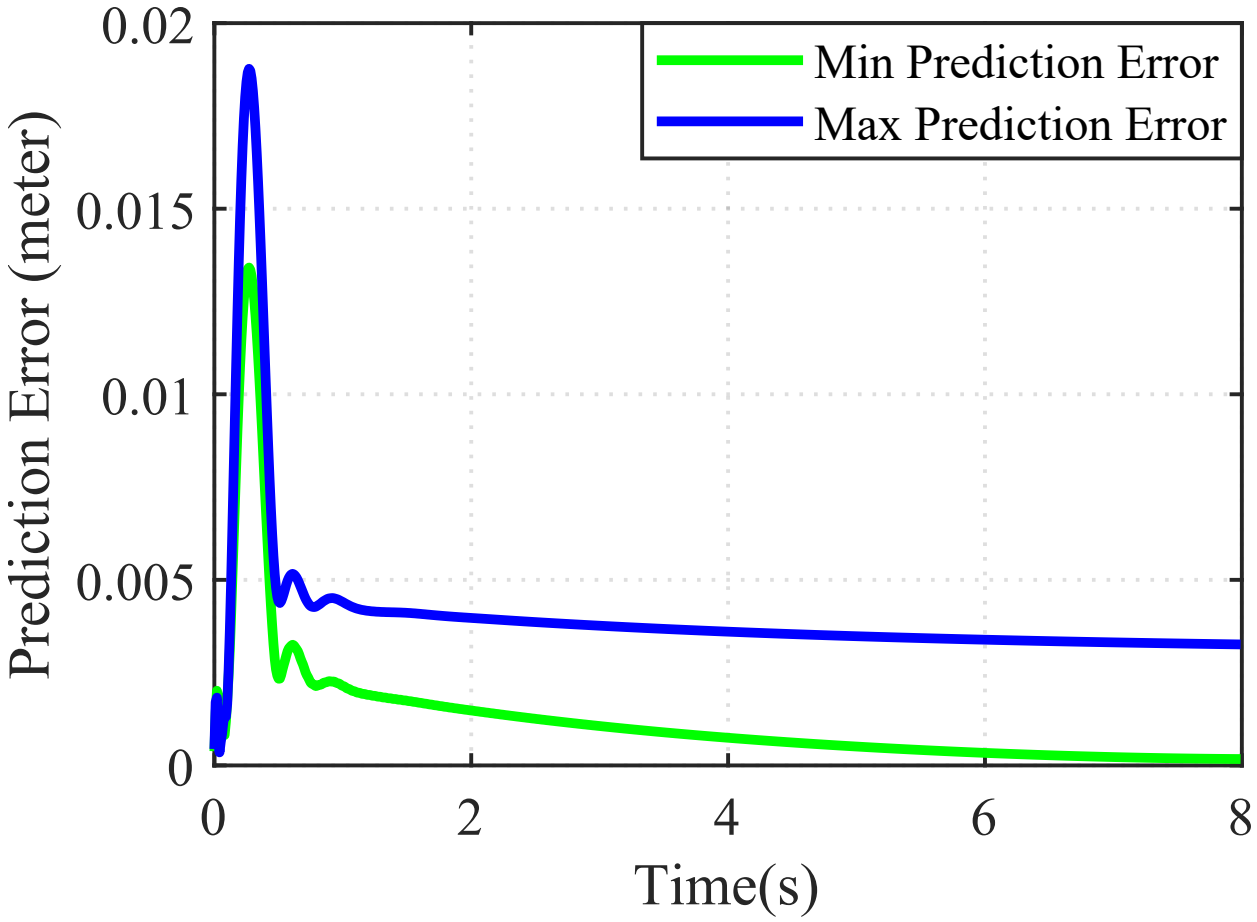


Figure 2.7: The maximum prediction error (in blue) and minimum prediction by the Koopman-based linear model (in green). The error at each time step is called using (12).

where $L = 1$ is the length of the soft manipulator in meter, $N_x = 18$ is the number of states, $x^j(t_i)$, $x^{j_{ref}}(t_i)$ is the j^{th} state vector at time t_i approximated by Koopman based model and actual physics-based model, respectively.

For the computation time, the Koopman-based linear model is much more time efficient compared to its Physics-based counterpart. To ignore the effect of other operating system tasks interfering with simulation we measure the meantime. For the Koopman-based model, the mean time was calculated by running the model 7 times, and each run consisting of 100 loops. Similarly, the physics-based model was 7 times and each run consisted of 10 loops, and $dt = 10^{-4}$ was fixed when running the time performance test. The time comparison tests are run on a computer having 16 GB Random Access Memory (RAM) and a 2.6 GHz CPU. Table 2.3 shows the comparison between the mean time taken by both the models to run for a finite number of time steps. We can see the Koopman-based linear model is much faster compared to the actual physics-based model: over 12 times faster except in the case of the single step.

Table 2.3: Mean time comparison of Physics and Koopman-based model

# Time steps	Physics-based	Koopman-based
1	1.7 ms ± 353 µs	188 µs ± 29.5 µs
10	2.21 ms ± 112 µs	159 µs ± 50.7 µs
100	9.14 ms ± 386 µs	696 µs ± 23.5 µs
1000	60.8 ms ± 1.13 ms	4.75 ms ± 111 µs
10000	568 ms ± 7.2 ms	44.6 ms ± 640 µs
100000	5.6 s ± 49.3 ms	435 ms ± 5.79 ms

* The time shown is the mean time per loop.
* For the Koopman-based model the mean is over 7 runs, 100 loops each
* For the Physics-based model the mean is over 7 runs, 10 loops each

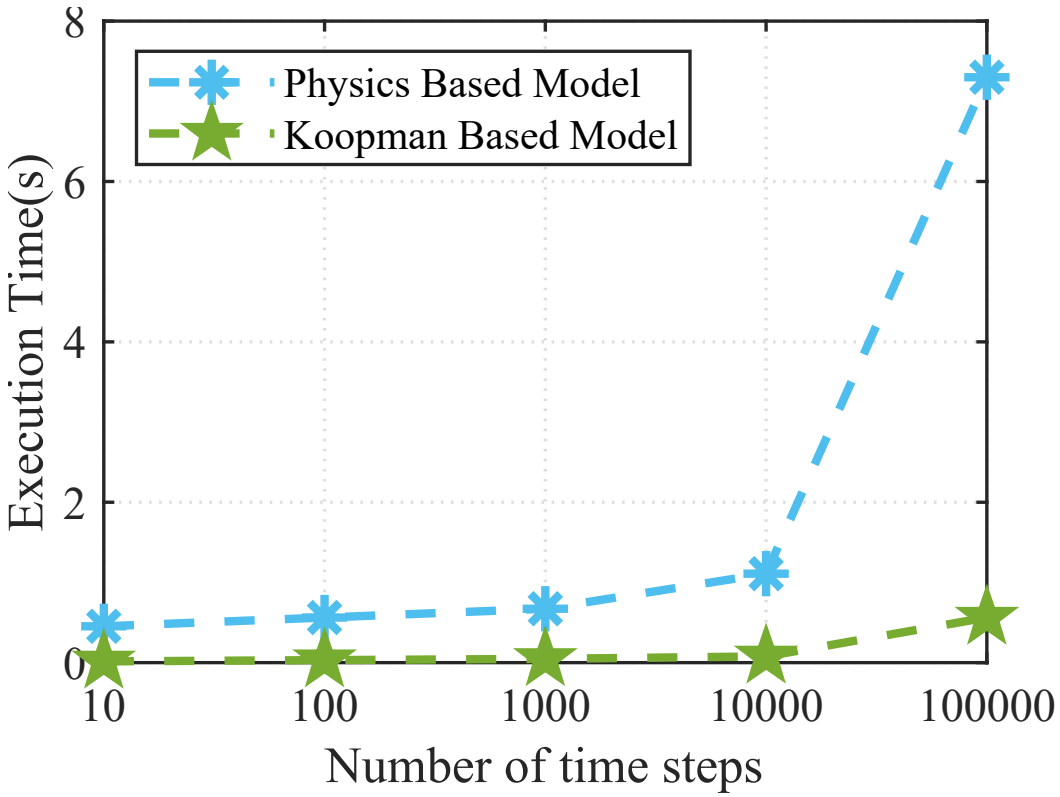


Figure 2.8: Plot showing the time comparison between actual Physics-based model and Koopman-based approximated model

2.3 Conclusion

In this chapter, we have used a data-driven method based on the Koopman operator to establish an approximated linear model of a soft manipulator using the input-output data from the physics-based model that is accurate but not computationally efficient. The linear model is accurate and 12 times faster than the physics-based model.

This approach shows promising results, but significant improvement can be made. For example, a much more accurate model approximation can be obtained if the eigenfunctions of the Koopman Operator can be approximated from data or we can approximate the robot with more tracking points.

Chapter 3

Learning Koopman Eigen function from data

Natural systems and a wide range of disciplines, including science and engineering, have dynamic elements. Most of the dynamical systems found in these fields are highly nonlinear. Nonlinear dynamical systems are of great importance as we find their use in almost every engineering application and other fields including biology, and finance. Despite nonlinear systems having a wide variety of applications, there is no general mathematical framework that allows researchers to analyze and solve the nonlinear system. Linear dynamical systems on the other can be easily analyzed using their spectral properties. For example for a given linear system to analyze the stability of the system, one could easily answer that question by inspecting the eigenvalues of the system.

One could easily analyze nonlinear systems if they could be represented as a linear system. Note that a nonlinear dynamical system can be linearized about their equilibria points. However, we do not follow this approach as the linearization does not hold well enough when operating far away from the point of linearization and thus fails to describe the time evolution of the nonlinear system due to this reason we opt to use the Koopman operator approach.

The use of the Koopman operator to approximate and analyze nonlinear dynamical systems is becoming more and more popular among researchers. This is due to the fact that the Koopman operator-based system identification produces a linear model of the nonlinear dynamical system and this approach is completely data-driven. Fortunately, sufficient data can be obtained either by running simulations or by carrying out actual experiments. Since the Koopman operator-based system identification produces a linear system of the actual nonlinear system therefore the Koopman operator-based framework gives the liberty to researchers to carry out the spectral analysis of the nonlinear system just like one could easily analyze a linear system using the tools from linear systems theory.

The Koopman operator-based system identification came to light to researchers after the development of Dynamic Mode Decomposition also known as the DMD algorithm and become more

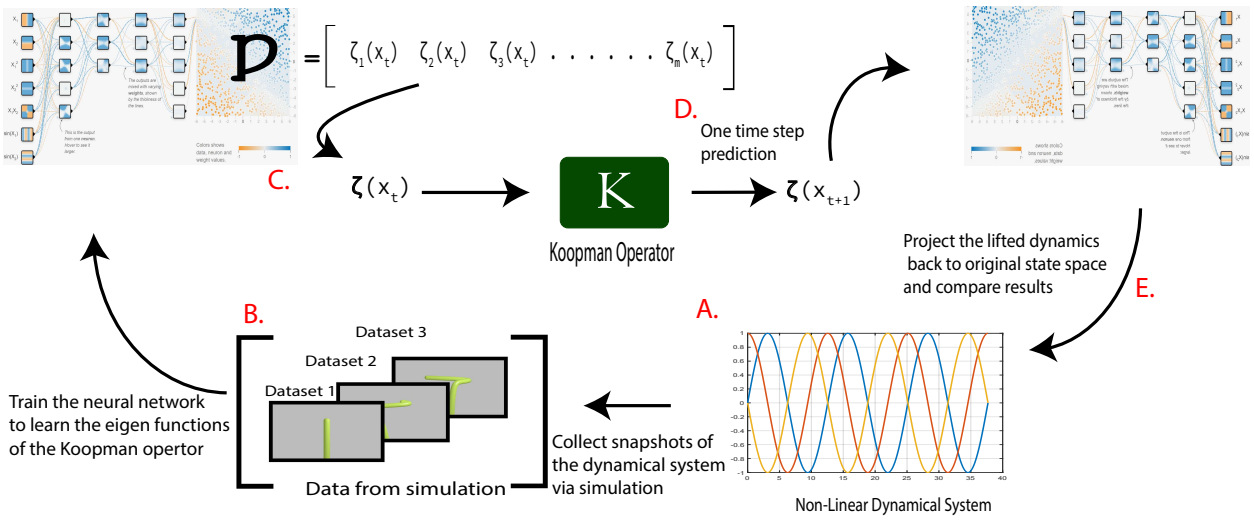


Figure 3.1: The figure shows the step-by-step process of learning the eigenfunctions of the Koopman operator from data. **A.** The nonlinear dynamical system is operated/simulated at various operating conditions. **B.** The data generated from the simulation/experiment is collected and fed to a neural network that learns the eigenfunctions required for lifting the nonlinear dynamics. **C.** These lifting functions are then used to approximate the Koopman operator as it advances the dynamics by on-time step shown in **D.** Then least square regression is used to approximate the Koopman operator. **E.** Then another network is used to project the lifted states back to the original state space. It is to be noted that steps **C**, **D**, and **E** are used together to learn the eigenfunctions of the Koopman operator. On comparing the given figure with figure 2.4 we can clearly notice that the need for manually selecting the basis function has been eliminated with the help of the neural network.

popular after algorithms like Extended Dynamic Mode Decomposition as this algorithm is can be utilized to develop system identification for systems with control inputs. All these algorithms developed so far rely on the method of lifting the original state space to higher dimensional function space where the nonlinear dynamics can be described in a linear fashion without significant loss of accuracy.

Thus, the choice of lifting functions used in Extended Dynamic Mode Decomposition and related algorithms greatly affects the accuracy of the approximated linear model based on the Koopman operator. However, there is no generalized procedure to figure out the type of lifting functions to be used given a particular type of nonlinear dynamical system. This can be seen from the results obtained in chapter 2, where we observe that the basis function used for modeling approximation of the cantilever beam fails to approximate the model of the soft to the required level of accuracy. The choice of these lifting functions greatly depends on the type of dynamical

system. One could also guess the right choice of basis function with experience. Therefore, it is of utmost importance to construct a highly accurate approximation of the Koopman eigenfunctions or discover the right choice of lifting functions.

As we have seen in equation (2.7) in chapter 2 the coordinates yielded by the eigenfunctions of the Koopman operator can describe the given nonlinear dynamical system in a linear fashion globally and the same can be done if we have the right choice of lifting functions which may or may not be the eigenfunctions of the Koopman operator. As a consequence of this there have been many algorithms developed that approximate the Koopman operator for a nonlinear system with control like Dynamic Mode Decomposition with Control [24], Extended dynamic mode Decomposition [10].

In addition to the above algorithms, Deep Learning has been recently used to approximate finite-dimensional Koopman operator like Deep DMD [25]. However, these approaches have only been applied to nonlinear systems without control input i.e. autonomous nonlinear systems.

In this chapter, we leverage the power of Deep Neural Networks to approximate the finite-dimensional Koopman operator for a soft robot which is a highly nonlinear system. The use of this method does not require expertise with the choice of the lifting functions nor requires one to have sufficient experience with it.

The rest of the chapter is organized as : Section 3.1 provides a formal introduction to the eigenfunctions of the Koopman operator and then late covers how eigenfunctions and other observables can be used as coordinates to describe the evolution of the states of a nonlinear dynamical system in a linear fashion. Section 3.2 talks about the use of Deep Learning to learn the eigenfunctions of the Koopman operator and then we build a deep learning framework to learn the eigenfunctions of the Koopman operator from data. Section 3.3 covers the results of the application of the deep learning framework to a single soft manipulator and then to a 4X4 soft grid robot. Section 3.4 concludes the chapter by talking about the possible improvements of the framework developed in this chapter.

3.1 Koopman Eigenfunctions

The linearity property proved in chapter 2 is only an auxiliary tool to replace a nonlinear dynamical system with a linear one. The key is the existence of linearly evolving coordinates also known as the Koopman operator *eigenfunctions* [26]. However, later we will show with the help of an example that one can replace the nonlinear dynamical system with a linear one if one has the right choice of embeddings.

Definition 3.1.0.1. *Any observables ϕ is an eigenfunction of the Koopman operator if it satisfies the following relation:*

$$[\mathcal{K}\phi](x) = \dot{\phi}(x) = exp(\lambda t)\phi(x) \tag{3.1}$$

with associated eigenvalue $\lambda \in \mathbb{C}$.

Let us again consider the motivation example used in chapter 2

which was given by a map $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$:

$$F(x) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} ax_1 \\ bx_2 + (b - a^2)x_1^2 \end{bmatrix} \tag{3.2}$$

where a and $b \in [0, 1]$. The system has a stable equilibrium at the origin and the invariant manifold is given by $x_1 = 0$ and $x_2 = -x_1^2$.

Eigenfunction Coordinates:

The associated principle eigen value and function pair $(\lambda, \phi(x))$ of the system descirbed above are (a, x_1) and $(b, x_1 + x_1^2)$. Using the fact that if (ϕ_1, ϕ_2) are the eigenfunctions of the Koopman operator with associated eigenvalue (λ_1, λ_2) respectively, then $(\phi_1\phi_2)$ will also be an eigenfunction of the Koopman operator with associated eigenvalue $\lambda_1\lambda_2$. We can construct more eigenvalue-eigenfunction pairs as powers of the principle pairs i.e. (a^2, x_1^2). We can use the principle eigenfunction-eigenvalue pair along with the new pair generated to lift the given nonlinear dynamical system into a new linear in time coordinates given by :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ x_2 + x_1^2 \\ x_1^2 \end{bmatrix} \tag{3.3}$$

Now, the dynamical system can be represented as

$$F(x) = V\Lambda\phi(x) \tag{3.4}$$

where: $\Lambda = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & a^2 \end{bmatrix}$, $V = [v_1, v_2, v_3]$ and $v_1 = [1, 0]^\top$, $v_2 = [0, 1]^\top$, $v_3 = [0, -1]^\top$

Obervables as Coordinates

Interestingly the key idea in using the Koopman Operator to describe a nonlinear dynamical system with a linear dynamical system is finding "good" coordinates that necessarily do not have to be the eigenfunction so the Koopman operator but lie in the span of the eigenfunctions [26]. For the system described above if we make the choice of the lifting to be $\boldsymbol{\psi}(\boldsymbol{x})$, where $\boldsymbol{\psi}(\boldsymbol{x}) = [x_1, x_2, x_1^2]^\top$. Then again the same dynamical system can be described with a linear map given by

$$F(x) = CA\psi(x) \tag{3.5}$$

where $C = [I_2, 0]$ and $A = \begin{bmatrix} a & 0 & 0 \\ 0 & b & (b - a^2) \\ 0 & 0 & a^2 \end{bmatrix}$

3.2 Learning Koopman Embeddings

Although Koopman operator theory has recently become a very popular tool in the analysis and systematic representation of nonlinear systems. Given the fact that the Koopman-based representation of the nonlinear systems is linear, there exists no systematic or generalized method for

choosing the "good" set of observables or a method to obtain the eigenfunctions of the Koopman operator for the nonlinear dynamics to evolve linearly in these coordinates.

Many algorithms such as the Dynamic Mode Decomposition popularly known as DMD and the Extended Dynamic Mode Decomposition have been already developed where EDMD makes use of a set of predefined library functions to lift the nonlinear dynamics. However, EDMD does not scale well to high dimensional nonlinear systems as it is highly dependent on the choice of the dictionary functions used. In most cases, the choice of these dictionary functions is dependent on the type of dynamical system and as a consequence of this, representing the nonlinear dynamical system with a linear system becomes more of a manual process.

In this section, we will make use of Deep Neural Networks to learn these lifting functions from data and approximate the best Koopman operator to accurately describe the nonlinear dynamics in a linear fashion. Many researchers [27], [28] including Lush et al [29] have come up with methods to either learn the Koopman eigenfunctions or the "good" lifting functions to describe the nonlinear dynamical system in a linear fashion using a finite-dimensional approximation of the Koopman operator. However, these methods have been only applied to low dimensional and simple nonlinear systems such as the simple pendulum and the *Duffing Oscillator* to name a few.

Since deep learning techniques have improved and can now approximate the finite-dimensional Koopman operator. The use of Deep Neural Networks offers a lucrative data-driven approach to approximate the Koopman operator from data without the need for a set of predefined dictionary of observables. This is due to the fact that a Deep Neural Network can be trained to learn the finite-dimensional representation of the Koopman operator. The reason for opting for the use of Deep Neural Networks is the fact that a Neural Networks is that given Neural Network with sufficient hidden units and a linear output layer is capable of learning any arbitrary function. Since the "good" observables are nothing more than some arbitrary function, we can use neural networks to learn these functions. In accordance with the Universal Approximation Theorem, one can be sure that neural networks can learn [30,31]. Hence, one can guarantee that Deep Learning is capable of fitting any function without having to manually select a basis function or manually design features.

However, Deep Learning has its own set of challenges which we will ignore as they are beyond the scope of this chapter.

The focus of this chapter is on developing a Deep Neural Network model that is capable of learning the "good" observable and thus accurately approximating the finite-dimensional Koopman operator for a given nonlinear dynamical system with control inputs. In our case, we will first try the Deep Learning model on the soft robot used for system identification in chapter 2 and then later use it on a group of 40 such robots combined together to form a grid.

Deep Learning Approach

The architecture of our Deep Learning Model is shown in figure: 3.2. The goal of this network is to learn the key or "good" observable coordinates that are spanned by the set of Koopman eigenfunctions so that the nonlinear dynamics can be described accurately in a linear fashion. The network developed has primarily three requirements which also serve as the three different types of loss functions used in the training of the network:

1. The first requirement of the network is to learn the coordinates along which the nonlinear dynamics evolve linearly along with the inverse so that the original states can be recovered. This is achieved by using an auto-encoder network (see figure 3.2) where N1 is the encoder and N3 is the decoder. The dimension of this network is a hyperparameter usually chosen depending on the type of the system. In practice, the dimension of this network is always greater than the dimension of the original nonlinear dynamical system. The loss function that describes the reconstruction function of the network is given by:

$$L_1 = ||\mathbf{X_k} - \psi^{-1}(\psi(\mathbf{X_k})|| \tag{3.6}$$

where ψ, ψ^{-1} represent the encoder and decoder network respectively and $\mathbf{X_k}$ represents the system state vector in discrete time at time k.

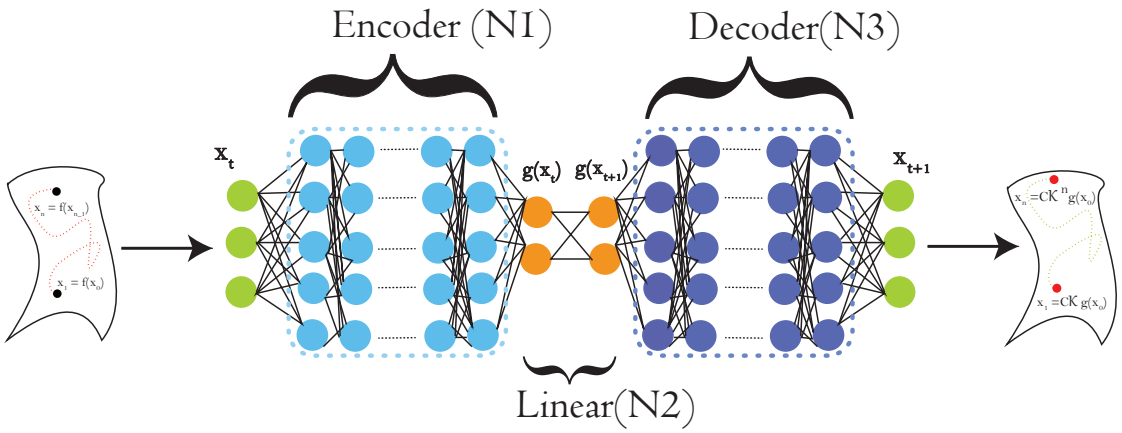


Figure 3.2: A. A schematic of the network used to learn the basis function is shown here. Given on the left is the actual nonlinear dynamics on a smooth manifold M. The system starts from the initial condition x_0 and then evolves with time to reach the state x_n. Note that the map f governing the evolution of the states is nonlinear. B. The current state of the system is fed as input to the network. C. The input is lifted to a higher dimensional space with the help of network 1 (N1) also called the Encoder. D. The dynamics are propagated by a one-time step forward in time with the help of a liner layer represented as Linear(N2). E. Then the original state variables are recovered with the help of the Decoder network represented as Decoder(N3). One can see that the same final state x_n can be reached with the help of the framework developed in a linear fashion. Note that on the state evolution shown on right: C is the projection operator and $\mathcal{K}$ is the Koopman operator.

2. The second requirement of the network is to accurately propagate the nonlinear dynamics in a linear fashion and predict the states at the next time step. To achieve this linear prediction of the nonlinear dynamics, the loss function for training is given by:

$$L_2 = ||X_{k+1} - \psi^{-1}(\mathcal{K}^k \psi(X_k))|| \tag{3.7}$$

where $\mathcal{K}$ is the Koopman Operator.

3. The third requirement of the network is to ensure that the control inputs on the states in the lifted state space have the same effect as on the states in the original state space we use the following loss function for this purpose:

$$L_3 = |\, ||\psi^{-1}(\psi(X_{k+1})) - \psi^{-1}(\psi(X_k))|| - ||X_{k+1} - X_k||\,| \tag{3.8}$$

The norm $||.||$ is the mean-squared error and we also add l_2 regularization to prediction loss and metric loss which is the third loss function. The final training loss function is simply the combination of (3.6),(3.7), and (3.8) and is given by:

$$L = L_1 + \lambda_1 L_2 + \lambda_2 L_3 \tag{3.9}$$

We then minimize the training loss function L in the model developed using the Adam optimizer and the model is ready to be used for system identification after training.

To train the model we generated 1000 trajectories by simulating the actual physics-based model with random inputs for 300 timesteps with dt set to 0.01. The dataset generated was split into training and testing sets. We will present the testing results in the subsequent section.

3.3 Results for PyElastica soft robot

In this section, we present the testing results of the Deep Learning model developed in the previous section. We will also compare the results from chapter 2 of the system identification for the PyElastica soft robot and the system identification results obtained from the Deep Learning framework developed in this chapter.

3.3.1 Demonstration on a single PyElastica soft roobt

To test the deep learning framework we first tested it on the single PyElastica soft robot that we used in chapter 2. For the framework, we generated 1000 data sets with random inputs and used 80% of the data for training the model and 20% of the remaining data for testing the model. Table 3.1 shows the network hyper-parameter for this case of system identification. Looking at figure 3.3 We can easily conclude that the Deep Learning framework developed in this chapter outperforms the EDMD algorithm as the prediction mean squared error when using the EDMD algorithm is 0.05 meters and that of the Deep learning framework is 0.002 meters.

Table 3.1: Network Configuration for Koopman operator approximation for single soft manipulator

S.No.	Network Name	Number of layers	Number of Nodes
1.	Encoder	4	[72,72,72,48]
2.	Linear	1	[48,48]
3.	Decoder	4	[48,72,72,72]
Input = 18,Ouput=18, $\lambda_1 = 3$, $\lambda_2 = 0.4$, Batch-size = 128			

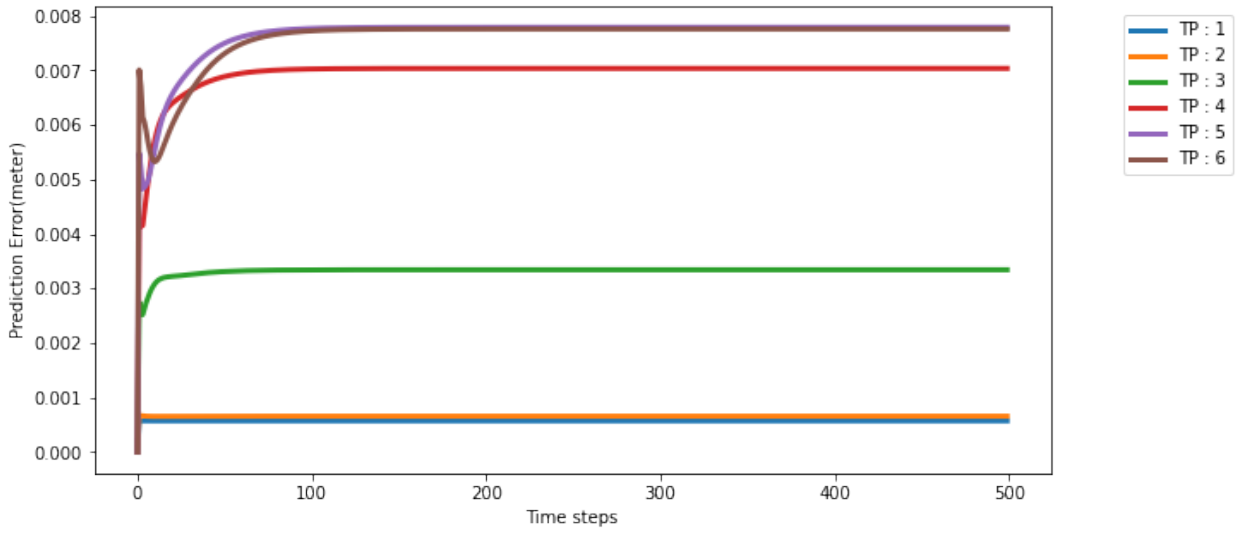


Figure 3.3: Prediction error (absolute error) plot of single PyElastica soft robot as approximated by the deep learning framework developed in this chapter. The linear model approximated using the deep learning framework developed in this chapter is very accurate and has a maximum absolute error of 0.0765 meters. The tracking points along the robot's body have been labeled as TP followed by a number. The first tracking point TP: 1 is located at the base of the soft manipulator and is fixed to the ground and other points are located on the soft manipulator as we move up from the base.

3.3.2 Demonstration on a soft grid

Using EDMD on 2x2 soft grid robot

To test if the EDMD algorithm will produce results as expected in the case of a single robot. We created a simple 2x2 soft grid robot and used the framework developed in chapter 2. The dataset for system identification was created by running the actual physics-based model for 300 time steps with random inputs and consisted of 30 different datasets for Koopman operator approximation and 30 different datasets for validation. The mean RMSE for 30 validation datasets was 1.2648 meters. Figure 3.6 shows the best approximation of the actual physics-based model in a set of

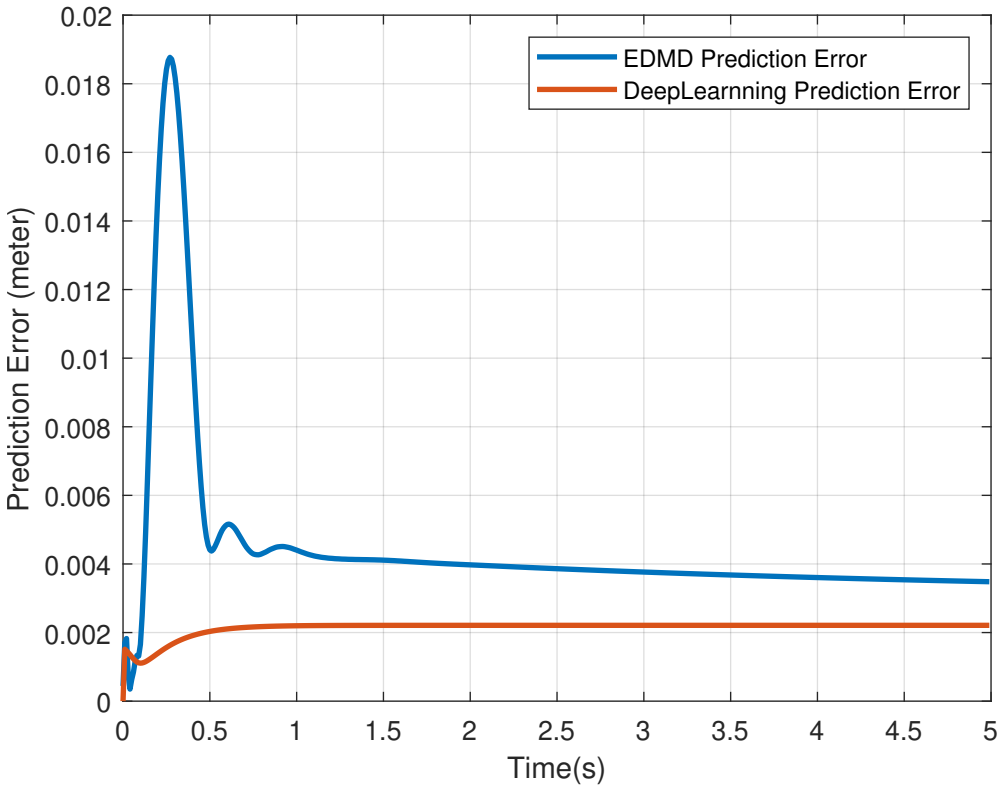


Figure 3.4: Studying the plot we can easily infer that the Koopman approximation based on the deep learning framework developed in this chapter is able to capture the transient behavior with greater accuracy compared to the EDMD-based Koopman approximation. It is to be noted that steady-state error (error after 4 seconds) is almost the same for both frameworks. Also, the prediction error shown in the plot is the highest prediction error for both frameworks when the models were validated against the testing dataset.

30 different datasets. We can easily conclude that the Koopman operator approximation using the EDMD algorithm fails in this case as it has a very large prediction error.

Using Deep learning framework on 4x4 soft grid robot

The soft grid is a soft robot that is formed by combining 40 different single PyElastica soft robots in such a manner that the resulting structure is a 4x4 grid. Figure 3.7 shows the prediction error plot for the Deep Learning framework developed in this chapter and table 3.2 shows the network parameters used for linear system identification of the grid soft robot using the Koopman operator theory. Studying the prediction error plot, we can assert that the linear model approximated is very accurate to that of the actual physics-based model. The prediction error in this case has a mean squared error of 0.0906 meters.

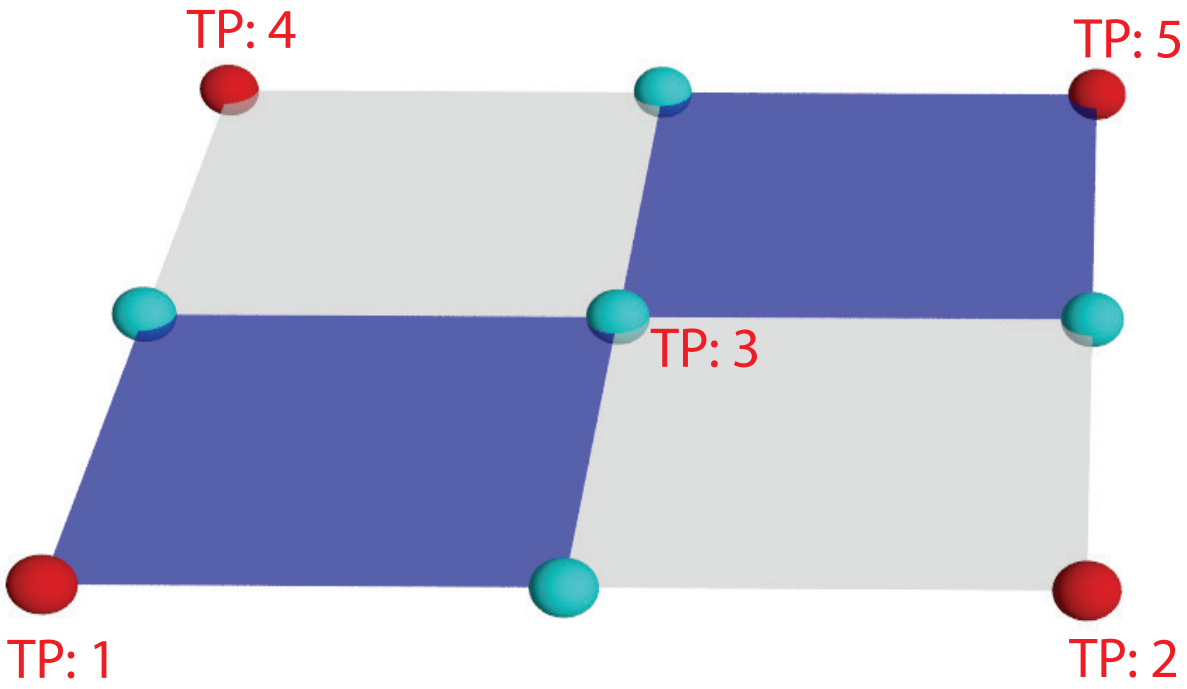


Figure 3.5: Figure showing the 2x2 soft grid robot with tracking points along the edges of each rod. Red spheres denote the end of points of the rods and the cyan sphere denotes the intermediate tracking points. It is to be noted that the total number of tracking points on each edge is 6 and they have been omitted in this figure for illustration purposes. Also, the colored part of the grid has been patched with blue and gray colors for aesthetic reasons. This 2x2 soft grid robot consists of a total of 12 rods connected in a manner such that the resulting structure is a 2x2 grid.

3.4 Conclusion

In this chapter, we leveraged the power of deep neural networks to construct a powerful framework for identifying the best observables that very accurately represent a highly nonlinear dynamical system in a globally linear fashion. In the framework we developed, the primary objective was to automatically learn the best lifting functions which is quite a challenging task and is also the key step behind Koopman operator based system identification.

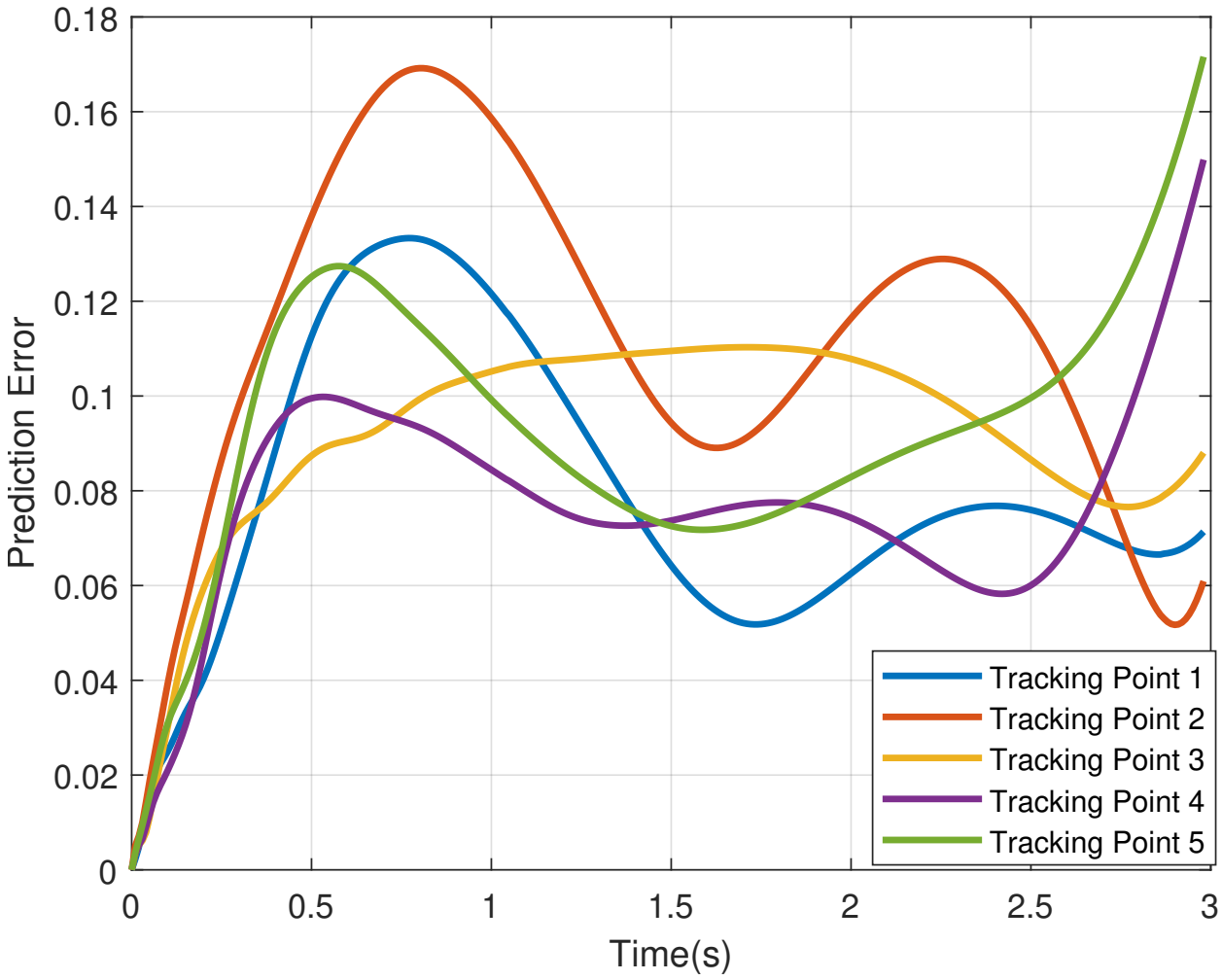


Figure 3.6: Plot shows the best prediction error (root mean square error) using EDMD algorithm for Koopman operator approximation in the case of a 2x2 soft grid robot. Maximum prediction error, in this case, was close to 0.18 meters, and looking at the plot we can assert that the error does not saturate and will continue to grow for the given tracking points. It is to be noted that other tracking points have been omitted for the clarity of the plot.

Table 3.2: Network Configuration for Koopman operator approximation for grid soft robot.

S.No.	Network Name	Number of layers	Number of Nodes
1.	Encoder	4	[1616,1616,1616,816]
2.	Linear	1	[816,816]
3.	Decoder	4	[816,1616,1616,1616]
Input = 720, Output=720, $\lambda_1 = 5$, $\lambda_2 = 0.7$, Batch-size = 128			

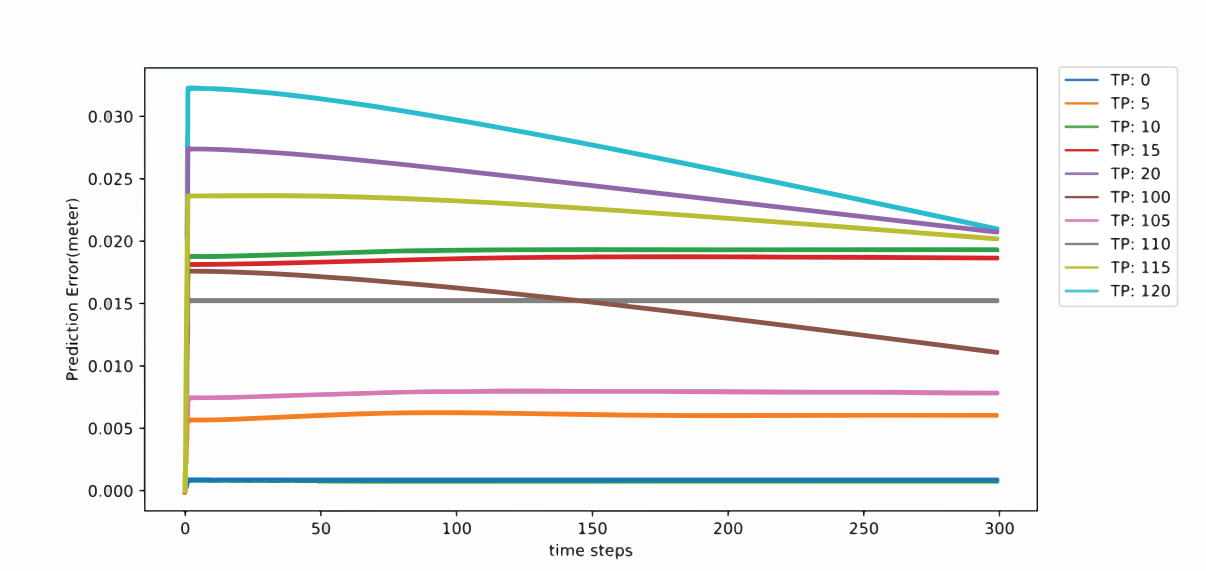


Figure 3.7: Prediction error plot of the linear model approximated using Koopman-based system identification for the soft robot. Different color plots correspond to different tracking points of the grid. The grid soft robot has a total of 240 tracking points, to maintain the clarity of the figure the plot contains a prediction error of only 10 tracking points starting from the least prediction error to the maximum prediction error. The approximated linear model has a maximum root mean square error of 0.034 meters in the case of a 4x4 soft grid robot.

We design a deep auto-encoder network by adding loss functions and constraints to accurately learn the best observable functions that lie in the span of Koopman eigenfunctions and where the nonlinear dynamics evolve linearly. This framework developed accurately replaces the actual high dimensional nonlinear dynamical system with control inputs with a linear system using the Koopman operator theory.

Although the framework developed approximates the nonlinear dynamics very accurately but there are many drawbacks associated with this framework. The very first drawback of this framework is the need for a large and diverse dataset. The framework is also computationally extensive to train. With the use of deep learning involved one might consider that this model should be applicable to systems in general, however, this is not the case. The current framework can definitely deal with any uncertainties introduced in the environment but will perform poorly when the same model is used on any subcomponent of the system or augmented components of the system. For example: Given a trained model for a 2x2 grid soft, it should be able to still approximate the

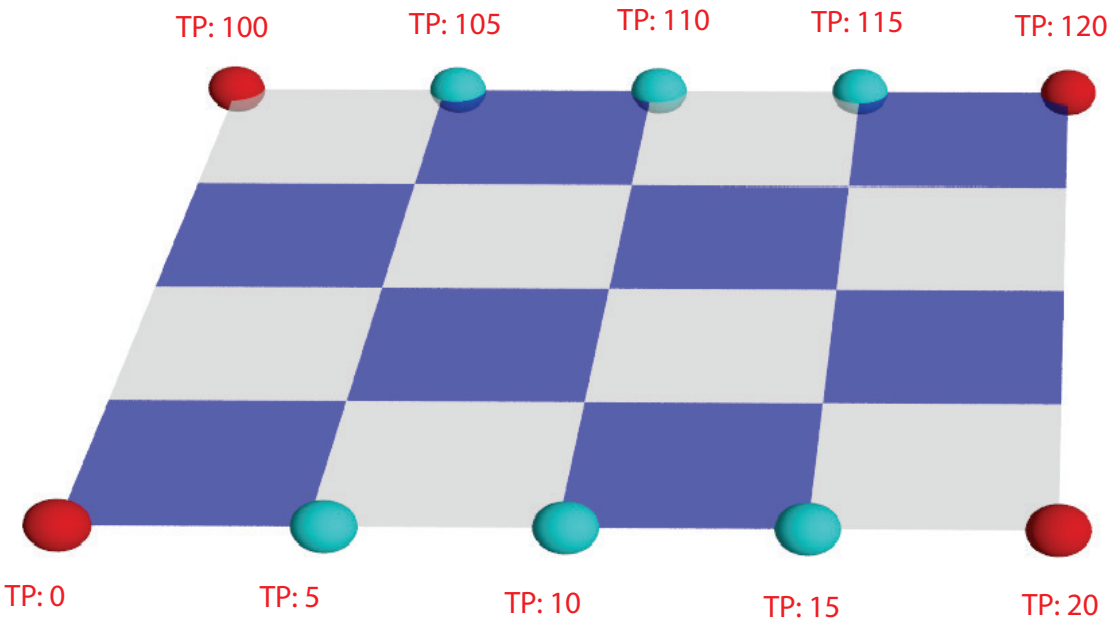


Figure 3.8: 4X4 soft grid with tracking points described in figure 3.7

dynamics if the original system is changed into a 3x3 or a 4x4 grid or even just a square. There is also one very specific limitation of this framework and that is the choice of lifted dimension to accurately describe the nonlinear dynamics in a linear fashion. In this framework, it is a hyperparameter. All these drawbacks hint at future work that will address these issues. The use of deep neural networks in learning the Koopman operator has recently attracted the attention of the robotics community and hopefully, the drawbacks mentioned in this section will be addressed by more powerful models.

Chapter 4

Shape Control

4.1 Introduction

One of the most important reasons for the increased interest in the field of soft robotics is the fact that soft robots are capable of addressing problems that cannot be addressed by rigid-bodied robots. The robotics community is continuously in the search of methods to design robots that are not only human-friendly but also can adapt to their external environment. Although the current robots being used in industries are very fast and incredibly precise for the further development of the manufacturing industry there is a need to deploy these robots that work in close proximity or side by side with humans as this will allow the robotic systems to carry out tasks that require human intervention which is still beyond the abilities of rigid bodied robots. Keeping this in mind there have recent developments of flexible systems in robotics such as Twisted-and-Coiled actuators [32] which have contributed to the development of new mechanisms such as programmable motion [33]. These robots have led to a different category of robots which are referred to today as soft robots.

Despite robots made from rigid bodies being the heart of various industries such as manufacturing, soft robots made from soft materials have recently emerged in robotics research [34, 35]. Unlike rigid ones, soft robots can leverage their inherent mechanical compliance to interact with humans or external environments, leading to many applications such as grasping or manipulation, locomotion, rehabilitation, assistance, medical devices, etc [36, 37].

Recently, the soft robotics community has started to investigate how a robot's shape can enhance the functional capabilities of the robot with inspiration from biological organisms [2]. In fact, various living organisms can change their body shape to cope with different environments and response to external stimuli. For example, an octopus can squeeze its body through gaps that are much smaller than body size [38], and moth larvae can curl up to roll away from predators [39]. Inspired by biological organisms, researchers have developed various robots to leverage different

shapes for different functions. For instance, Shah *et al.* investigated how a soft robot can use different shapes for either crawling or rolling in different environments [40]. Hwang *et al.* leveraged a novel kirigami composite to develop a morphing drone that can autonomously transform from ground to air vehicle [41]. Many other recent research on how shape morphing can enhance a robot's functionality is reported in the review paper [2].

To leverage different shapes to enhance functions, we need to control the shape of a soft robot to the desired ones. But it is challenging to control a soft robot's shape since soft robots exhibit highly nonlinear dynamics. Researchers have developed various physics-based models using different methods such as Cosserat Rod theory [7], and model reduction method [42], among many others [43]. Although such models can achieve high-fidelity simulation of various types of soft robots [44], they generally require a long computational time, making them unsuitable for the shape control of soft robots.

In this chapter, we aim to leverage the existing model for controlling the shape of soft robots. Specifically, we will use the PyElastica [7] simulation software to generate sufficient input-output data for a given soft robot. Using the data, we will then establish a data-driven model based on Koopman operator theory [8] to obtain a finite-dimensional approximation of a soft robot [6]. The Koopman operator can represent a nonlinear dynamical system with a finite-dimensional linear model to approximate the original dynamics of a soft robot. With such a linear model, we can directly use existing control methods such as model predictive control to control a soft robot's shape.

Note that researchers have recently used the Koopman operator theory to control soft robots [6, 45–48]. The work in [6] shows promising results in controlling the tip of the robot to trace a desired trajectory while [45] used the Koopman operator approach in modeling of soft robotic swimmer. But controlling the shape of soft robots is different from existing problems (e.g., tip position control). Therefore, our work adds another application of Koopman operator-based system identification and control to control the shape of soft robots. In other words, we develop a

data-driven method to control a soft robot's shape by leveraging existing physics-based models approximated using Koopman operator theory in chapter 2.

The rest of the paper is organized as follows. In section 4.2, we formulate the shape control problem for a soft robot. In Section 4.3, we talk about the shape control of soft robots using a Model Predictive Controller (MPC). In section 4.4, We describe the results for the tip control of a single soft manipulator, shape control of a single soft manipulator, and shape control of a 4X4 soft grid.

4.2 Shape Control Formulation

In this section, we formulate the shape control problem by using a general soft manipulator. We will show how to use this framework to solve the shape control problem in subsequent sections, but before proceeding to the shape control problem let us briefly review the Koopman-based system identification with control.

Consider a nonautonomous nonlinear dynamical system with control inputs, the methodology discussed can be used. Consider a discrete nonlinear dynamical system given by

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k), u(t_k)) \\ y(t_k) &= g(x(t_k)) \end{aligned} \tag{4.1}$$

where $u(t_k) \in \mathbb{R}^m$ where m is the dimension of control inputs of the dynamical system. Then in this case to approximate the Koopman Operator, the minimization problem in (2.9) changes to

$$\underset{A,\,B}{\text{minimize}} \quad \sum_{k=0}^{K-1} ||\psi(x(t_{k+1})) - (A\psi(x(t_k)) + Bu(t_k))||_2^2 \tag{4.2}$$

Thus, by solving the minimization problem of (2.13), the finite approximation of the Koopman operator is approximated by *A* and *B*, and it acts as one step predictor of the nonlinear dynamical system described by (2.12). The minimization problem for the output equation i.e. for *C* matrix remains the same as given by (2.11).

Now we will proceed with the formulation of the shape control problem.

Given a soft manipulator of length L, we divide it into $N-1$ segments with equal length. The shape for the i-th ($i = 2, \ldots, N$) segment is specified by the section at its top. At a discrete time step t_k, we use $g_i(t_k) \in SE(3)$ to represent the top section's position and orientation in the inertia frame Figure 4.1.

$$g_i(t_k) = \begin{bmatrix} R_i(t_k) & p_i(t_k) \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

where $R_i(t_k) \in SO(3)$ represents the orientation and $p_i(k) \in \mathbb{R}^3$ presents the position for the section's centroid. Denote the i-th segment as L_i. For each segment, we assume we can apply input u in terms of forces/torques at each segment's top. In reality, such forces/torques may be generated by artificial muscles embedded inside soft materials [33]. With such a setup, the shape of the soft manipulator can be approximated by $g_i(t_k) \in SE(3)$ at time step t_k.

Given a desired shape for the soft manipulator represented by $g_{i_{ref}}$ ($i = 1, \ldots, N-1$), the shape control problem can be formulated as finding the control input $u(t_k)$ that minimizes the distance between $g_i(t_k)$ and $g_{i_{ref}}$.

Note that the distance in $SE(3)$ can be defined separately for the position and orientation with the Euclidean distance for position and geodesic distance for orientation. This work will focus on a simplified problem by only considering the position distance. In this case, the problem can be formulated as follows:

$$\underset{u(t_k)}{\text{minimize}} \quad \sum_{i=1}^{N-1} ||p_{ref} - p_i(t_k)||_2^2 \tag{4.4a}$$

$$\text{subject to}$$

$$x(t_{k+1}) = f(x(t_k), u(t_k)), \tag{4.4b}$$

$$h(x(t_k), u(t_k)) \leq 0 \tag{4.4c}$$

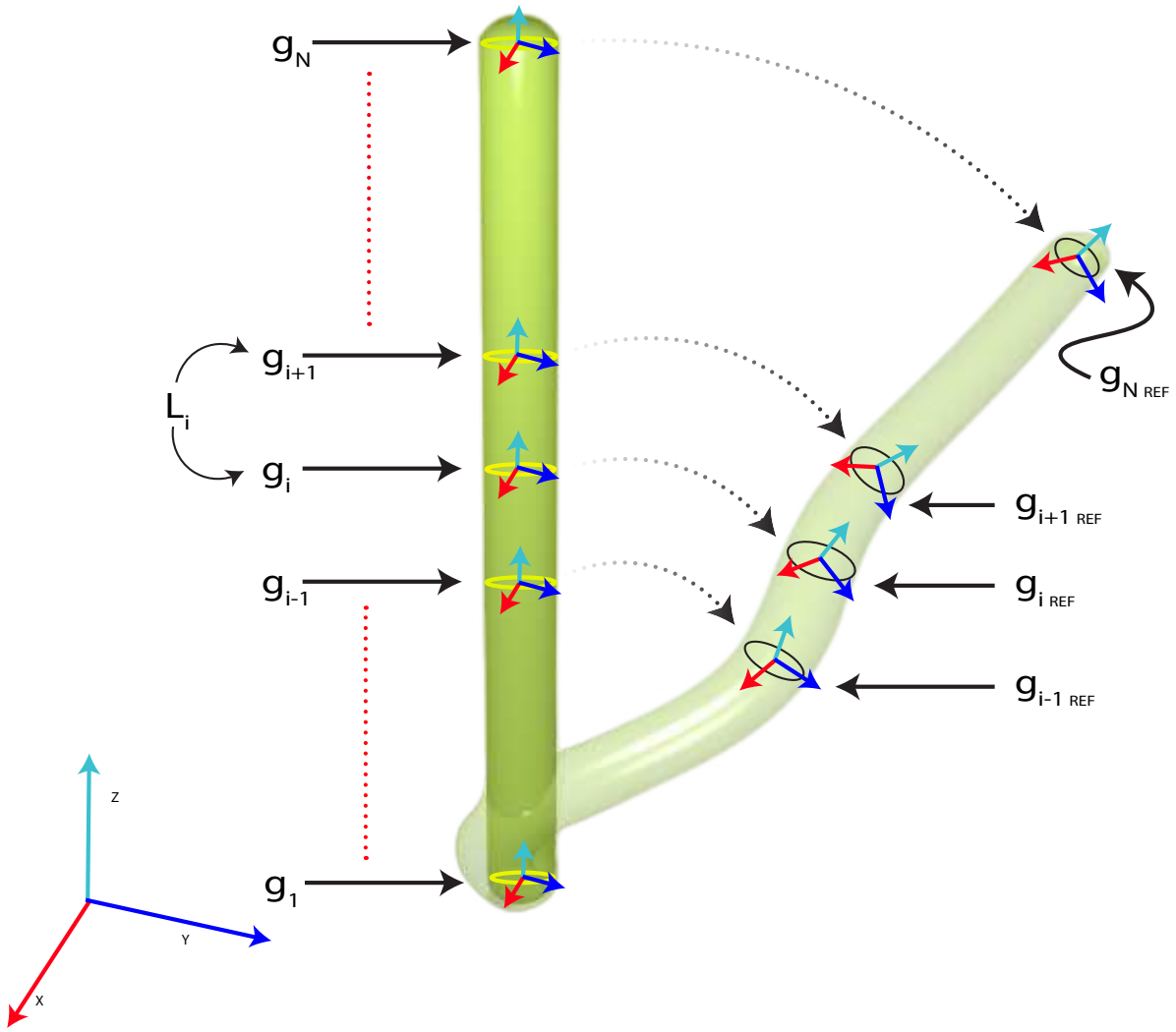


Figure 4.1: Illustration for the shape control problem for a soft manipulator divided into $N-1$ equal length segments with the top of each segment shown as a yellow cross-section. The manipulator shown in solid green color is its initial shape ($t = 0$) and the shape shown in faded green shows the target shape.

where $p_{ref} \in \mathbb{R}^3$ is the desired position, $x(t_k) \in \mathbb{R}^n$, $u(t_k) \in \mathbb{R}^m$ are the states and control input of the system at time instant t_k, respectively. $h(x(t_k), u(t_k)) \leq 0$ are the various constraints applied to the state and control variables which are commonly known as polyhedral constraints.

Figure 4.2 shows a similar concept of shape control represented pictorially in a simplified figure. Generally, the dynamics for a given soft robot (i.e., $x(t_{k+1}) = f(x(t_k), u(t_k))$) is highly nonlinear, involving complicated physics-based models [43]. Such models can only be solved numerically with considerable computation time, preventing them from real-time shape control of

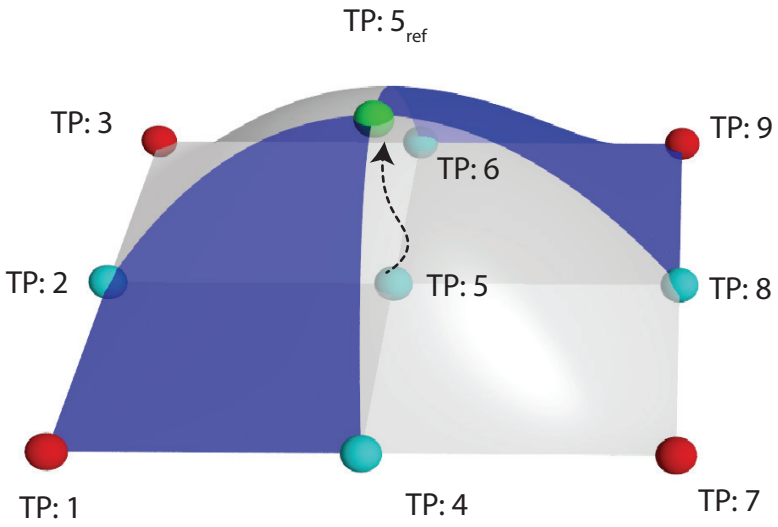


Figure 4.2: Illustration for shape control problem in case of a 2X2 soft grid robot. For simplicity and to maintain clarity, only a few tracking points have been shown. Red spheres denote the endpoint of the edges forming the soft grid and Cyan spheres denote the intermediate tracking points on the edges forming the grid. The gray checker plane is the configuration of the soft robot at $t = 0$ and the Blue-Gray checker shape is the reference shape given to be acquired by the soft grid. The 5^{th} tracking point labeled as $TP : 5$ moves from the center of the grid to $TP : 5_{ref}$ shown by a green sphere. Similarly, all other tracking points follow a desired trajectory to attain the given reference points.

soft robots. Inspired by recent work on using the data-driven method to identify approximated models from either numerical or experimental data for controlling soft robots for manipulation [6, 45–48], we aim to obtain a data-driven model using Koopman operator theory and then use the model to *control the shapes* of soft robots.

4.3 Control of soft robots

Soft robotics presents advantages such as being more flexible and compliant towards the environment (Trivedi et al. (2008)). However, the other major problem soft robots pose apart from modeling is the control aspect. Unlike rigid-body robots where the robotics community has developed advanced control methods for the control and manipulation of these robots, the methods developed for rigid-body robots do not hold well in the case of soft robots as soft robots are made up of complex deformable structures which create a problem in developing the exact mathematical

formulation for the soft robotic model as it requires to take into account the infinite dimensionality of the soft robots state space.

The theory of infinite state space is still confined to relatively simple systems [49]. The use of learning techniques is one possible solution to this problem as one could use the data to develop control policies without having to construct the mathematical model of the system. Such methods have been previously deployed to achieve open-loop and closed-loop control of soft robots. For example: [50] used deep reinforcement learning for the tip control of a soft robotic manipulator, [51] used a combination of recurrent neural networks (RNN) and supervised reinforcement learning to develop a method for closed-loop control of a soft robotic manipulator. The drawback of these control strategies is that the control policies are learned for a specific task and as a consequence of this, they do not hold well in case of arbitrary tasks [6].

It has been observed that model-based control methodologies have a very important role in achieving a higher level of performance in the case of artificial as well as natural systems [49]. Since model-based controllers can predict future occurrences, which leads to the best selection of control inputs and, as a result, improved performance, having an accurate model of the system makes it possible to create controllers that can generate control inputs for any task [6].

Many model-based controllers have been developed for soft robots [52, 53], but most of them rely on simplifying assumptions and mostly are static. Such controllers have been proven to be very efficient for the static control of soft robotic manipulators. The disadvantage of these controllers is that they are only limited to static control and they are not suitable for dynamic control of complex soft robotic systems. Dynamic control of soft robots is achieved by supplementing a piece-wise constant curvature model with data-driven trajectory optimization but again the downside of this approach is that the training is very task-specific [6]. There also have been some more realistic physics-based models, but they are computationally very expensive.

4.3.1 Model Predictive Control

Many model-based controllers have been developed for soft robots [52, 53], but most of them rely on simplifying assumptions and mostly are static. Such controllers have been proven to be very efficient for the static control of soft robotic manipulators. The disadvantage of these controllers is that they are only limited to static control and they are not suitable for dynamic control of complex soft robotic systems. Dynamic control of soft robots is achieved by supplementing a piece-wise constant curvature model with data-driven trajectory optimization but again the downside of this approach is that the training is very task-specific [6]. There also have been some more realistic physics-based models, but they are computationally very expensive.

Model Predictive Control is an advanced control algorithm that makes use of a model to predict the future behavior of the system. Model Predictive Control (MPC) generates the optimal control input by solving an optimization problem which is usually subject to some constraints. The objective function for the minimization problem is also referred to as the cost function and is composed in such a way that the system output tracks a given reference. MPC only applies the first value of the optimal trajectory generated and this step of optimization and prediction is repeated at each time step. The factor that makes MPC distinct from conventional control methods is the combined computation of prediction and optimization at each time step.

From chapter 2, we know that the Koopman operator approximates a linear system of a nonlinear dynamical system from data. In [6], they constructed MPC controller from a linear Koopman representation of a nonlinear dynamical system. We use a similar approach demonstrated in [6] to construct an MPC for shape control of a soft manipulator. Since the identified model is linear, the MPC optimization has computational advantages over nonlinear ones as the MPC optimization problem is convex whereas, for the nonlinear models, it is not. Since the optimization problem is convex, it can be solved very efficiently with any method for convex optimization. For Koopman-based MPC, we first define the objective function as follows:

$$J \;=\; z(t_{N_h})^\top Q(t_{N_h}) z(t_{N_h}) + q(t_{N_h})^\top z(t_{N_h}) \;+\; \sum_{i=0}^{N_h-1} \{z(t_i)^\top Q(t_i) z(t_i) \;+\; u(t_i)^\top R(t_i) u(t_i)\}$$

where $N_h \in \mathbb{N}$ is the prediction horizon, $Q(t_i) \in \mathbb{R}^{N_c \times N_c}$, $R(t_i) \in \mathbb{R}^{m \times m}$ are positive semidefinite matrices. Here Q, R are constant matrices. Then we can iteratively solve a convex quadratic program over a receding horizon as shown below:

$$\underset{u(t_i)}{\text{minimize}} \quad J \tag{4.5a}$$

$$\text{subject to}$$

$$z(t_{i+1}) = Az(t_i) + Bu(t_i), \tag{4.5b}$$

$$E(t_i)z(t_i) + F(t_i)u(t_i) - b(t_i) \leq 0, \tag{4.5c}$$

$$z(0) = \psi(x(t_k)) \tag{4.5d}$$

where $E(t_i) \in \mathbb{R}^{c \times N_c}$ and $F(t_i) \in \mathbb{R}^{c \times m}$ and the vector $b(t_i) \in \mathbb{R}^c$ define state and input polyhedral constraints where c denotes the number of constraints. Every time the optimization routine is called the predictions need to be set to the current lifted state $\psi(x(t_k))$. While the size of the cost and constraint matrices depend on the dimension of the lifted state N_c, [54] shows that these can be rendered independent of N_c by transforming the problem into its so-called dense-form [6]. We use the above framework to solve the shape control problem proposed in this chapter.

4.4 Results

4.4.1 Tip Control with Koopman-based MPC

Controlling the tip of the soft robot is a very special case of the shape control problem that we proposed in section II. Unlike controlling the shape of the soft robot where we are required to control multiple points, we control only one point, i.e., the tip of the manipulator. The problem of controlling the tip of the soft manipulator can also be called as set point tracking in three-dimensional space. We use the linear Model Predictive Controller with a prediction horizon $N_h =$

25 steps. The desired reference set points as [0.1, 0.2, 0.3] shown in section IV. To demonstrate the control of tip, we move the tip of the soft manipulator from the rest position, i.e., from [0, 0, 1] to [0.1, 0.2, 0.3] where the elements of the vector are the x, y, z coordinates (in meter) of the manipulator. The result is shown in Figure 4.3, which shows the Euclidean distance between the tip of the robot and the reference point with time. From the plot is clear that the MPC controller can move the tip to the desired location within 0.5 s.

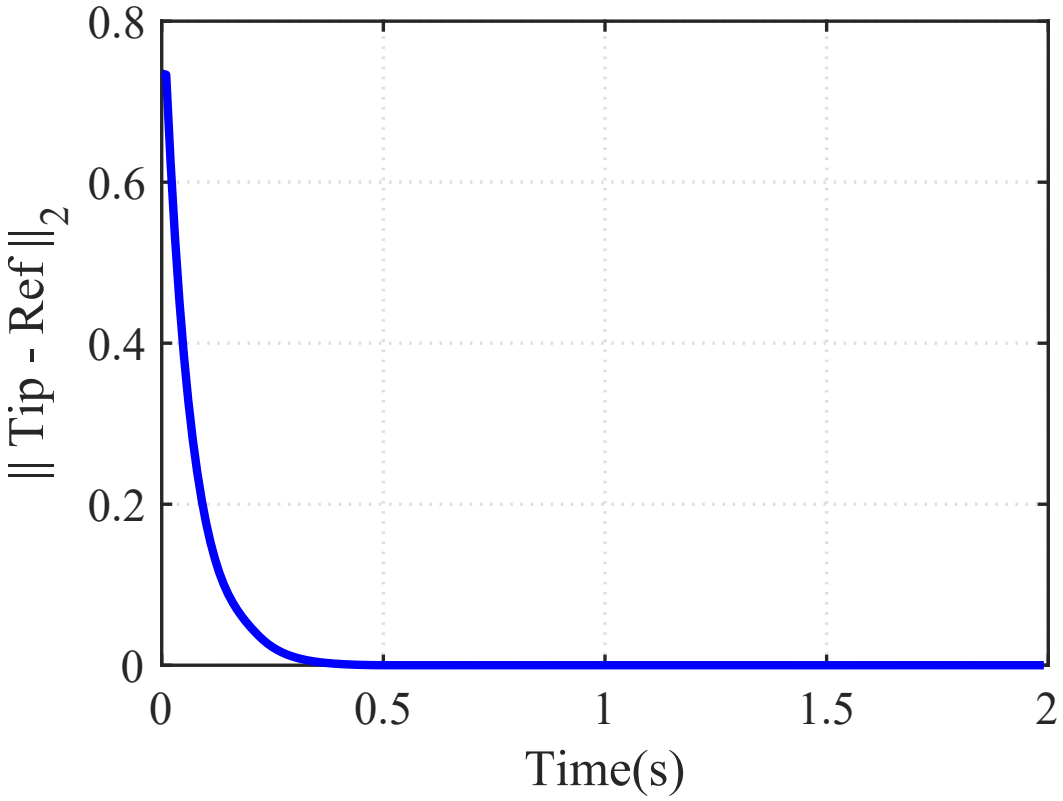


Figure 4.3: Results for the control of the tip of the manipulator. The plot shows the Euclidean distance (in meters) between the tip and the reference set point. Here $||.||_2$ denotes the Euclidean norm.

4.4.2 Application of shape control to single soft robot

We further demonstrate the shape control of a soft manipulator by controlling it to different shapes, specifically three letters: 'C', 'S', and an inverted 'U'. The linear Model Predictive Controller is used to derive the optimal control input over the prediction horizon of $N_h = 25$. For

different shapes, the controller has a similar cost function and no input or state polyhedral constraint. The objective of the controller is to move the points being tracked to the desired reference location in the workspace of the robot. Hence a cost function is chosen in such a way that it penalizes the distance between the reference point and points on the robot's body. To generate the reference or desired shapes, we use the shooting method with the physics-based model. Then these shapes are supplied as the reference shapes to the MPC controller. Note that since the bottom of the manipulator is rigidly fixed to the ground, the segment close to the ground needs to resemble a vertical shape due to the spline used to interpolate the torques in PyElastica. If we do not consider this segment, however, the desired shapes are indeed close to the letters 'C' and 'S'.

The results for the three shapes are illustrated in Fig. 4.4, where we plot the robot's final shape and the desired shape. From the figure, we can see the robot can accomplish the desired shape. To quantify the error between the final and desired shape, we plot the RMSE in meters for different tracking points for the morphed shapes in Figure 4.5. Root Mean Squared Error is calculated as the RMSE between the final position generated by the controller for a tracking point and the corresponding reference point. As we can see from the error in Figure 4.5, the final position error can be quite small ($< 5\%$ with respect to the manipulator's length).

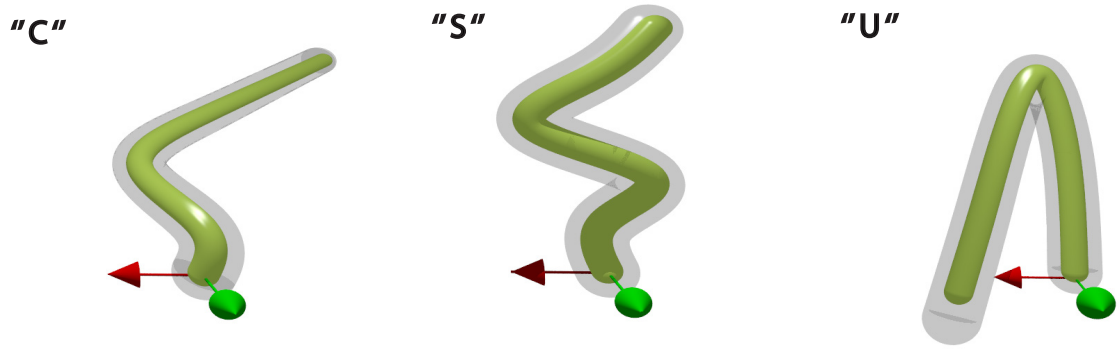


Figure 4.4: Results for controlling the soft robot to three different shapes. The object in solid green is the shape of the actual soft robot and the gray envelope is the reference shape. For reference X axis has been color coded as Green, similarly, the Y axis has been color-coded as Red.

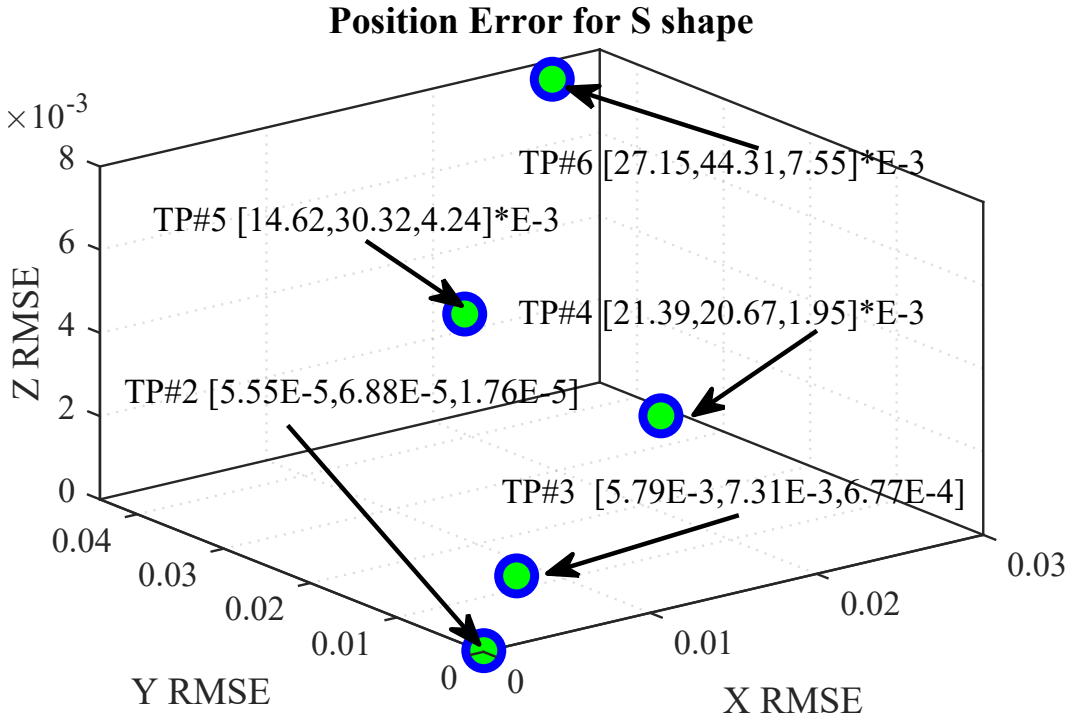


Figure 4.5: Position error of the tracking points corresponding to the S shape acquired by the soft manipulator as shown in Figure 4.4. The term TP in the plot stands for Tracking Point and the vector following TP shows RMSE for X, Y, and Z for a particular tracking point. Here the Root Mean Squared Error (RMSE) is measured in meters. (Note that the error for Tracking point 1 was always 0 as it was static, hence it was ignored in the error plot.)

Table 4.1: RMSE table for "C" Shape

Tracking Point Number	X RMSE	Y RMSE	Z RMSE
2	7.5848×10^{-6}	3.6561×10^{-6}	1.6257×10^{-6}
3	0.0004	0.0011	6.3933×10^{-5}
4	0.0001	0.0030	0.0004
5	0.0021	0.0055	0.0007
6	0.0042	0.0084	0.0012

* Error for tracking point 1 is not included as it was static

4.4.3 Application to 4x4 grid soft robot

We apply the Deep Koopman framework to the 4x4 grid soft robot for system identification. The deep learning framework developed in chapter 3 replaces the nonlinear physics-based

a model with a Koopman-based linear model, once the linear model is obtained, the shape control problem for this 4x4 grid soft robot is similar to the shape control problem proposed in this

Table 4.2: RMSE table for "U" Shape

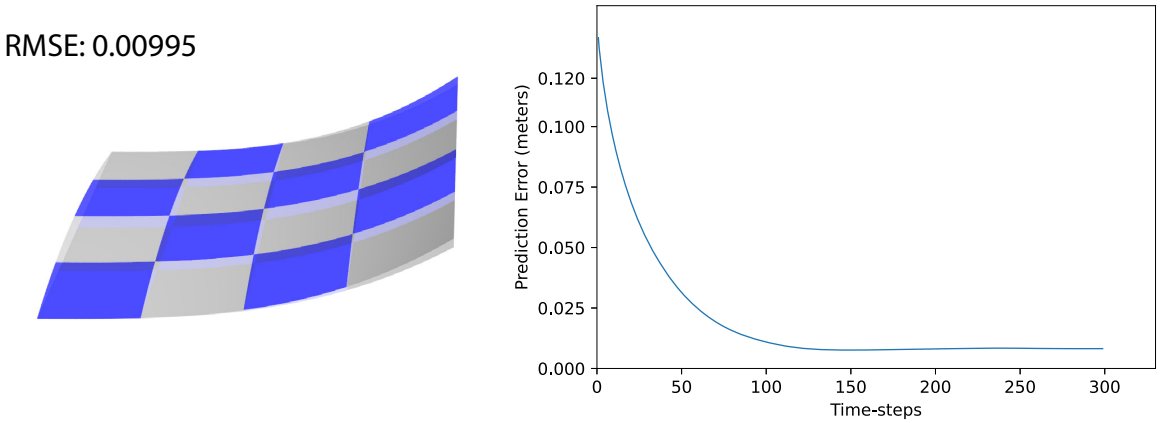
Tracking Point Number	X RMSE	Y RMSE	Z RMSE
2	1.0475×10^{-6}	1.8959×10^{-6}	1.8098×10^{-6}
3	0.00019	0.00010	8.2494×10^{-6}
4	0.00058	0.00021	1.4288×10^{-5}
5	0.00075	0.00023	4.8888×10^{-5}
6	0.00110	0.00043	0.00010

* Error for tracking point 1 is not included as it was static

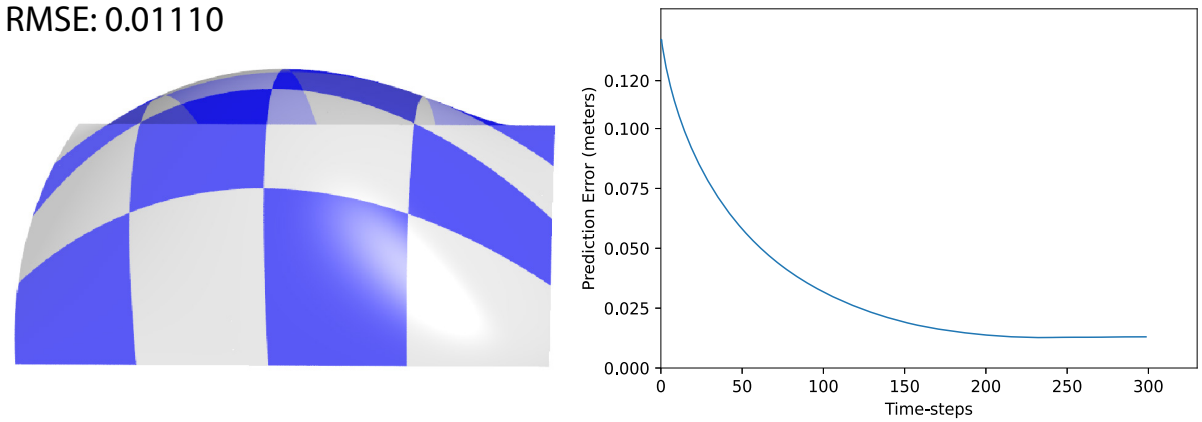
chapter which can be efficiently solved using linear MPC or Linear Quadratic Regulator (LQR) control methods. The shape control problem proposed in this chapter can be solved in a similar way just like we did it for a single soft robot system. In the case of the grid soft robot, we use an LQR controller to accomplish the task of shape control. It is to be noted that the same can be achieved using Linear MPC as the optimization problem are equivalent, we use LQR for ease of use. The cost function is chosen in a way such that it minimizes the euclidean distance between each tracking point on the grid to the reference point. It is similar to what we did in the case of a single soft robot.

To generate the reference shapes we apply uniform forces to individual soft robots forming the grid. for example: To generate a reference shape for a simple case where the grid fold along one of the edges: we apply uniform forces in the positive y direction to individual soft robots forming the edge of the soft grid. We generate complicated shapes such as the one similar to Franke's function using the same procedure discussed for the simple fold case.

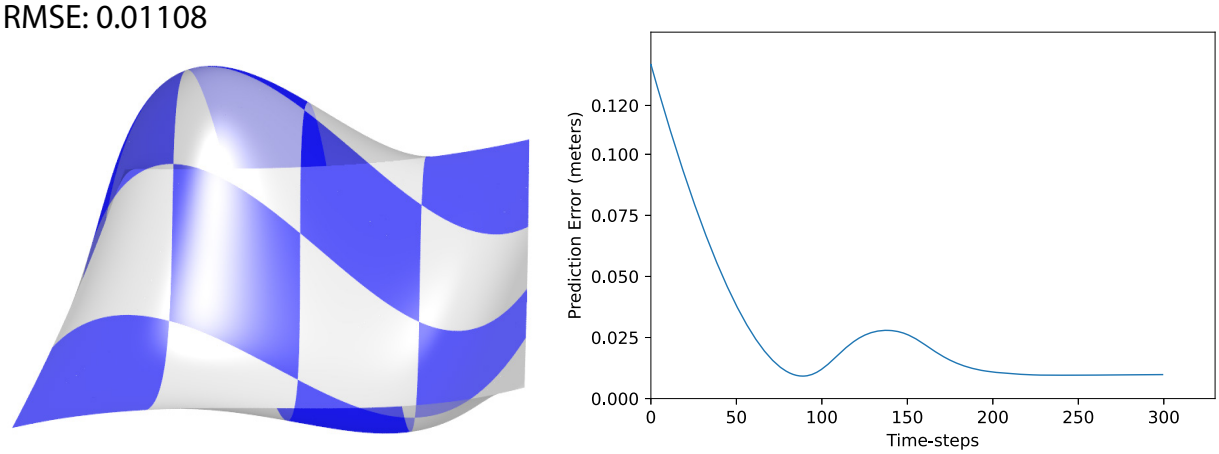
The results for shape control are demonstrated using three reference shapes (simple fold, Up-side down bowl, and Franke's function) are shown in the figure 4.2. The error (in meters) is calculated as the RMSE between the position of the tracking points generated by the controller and the position of the tracking points given as the reference. The final error is quite small (less than 3%)



(a) Shape 1 with bending along one of the edges on left and Error from the reference shape on left



(b) Shape 2 with on left and Error from the reference shape on left



(c) Shape 3 resembling Franke's Function on left and Error from the reference on right

Figure 4.6: Results for shape control in case of a 4X4 soft grid. The plot contains the steady-state Root Mean Square Error printed on the top of the shape acquired and one can see the variation of the shape error as controlled by the controller on the right side of each shape.

Chapter 5

Conclusion and Future Works

The objective of this thesis was to develop an efficient and accurate data-driven driven modeling framework using the Koopman operator theory that could be used to increase the functional capabilities of a soft robot by controlling its shape of the soft robot. To demonstrate the purpose two approaches for the approximation of the Koopman operator were discussed. The first approach used the Extended Dynamic Mode Decomposition (EDMD) [10] where we are required to come up with a dictionary of predefined lifting functions to lift the dynamics from the original state space to the high dimensional functional space. Then the second approach for the finite-dimensional approximation of the Koopman operator relied on the power of neural networks to learn the lifting functions from data to accurately describe the dynamics of the physical system in a linear fashion. These modeling approaches based on Koopman operator theory when combined with optimal control methods can be used to provide closed-loop control for soft robots that can be used to increase the functional capabilities of a soft robot such as controlling the shape of the soft robot

Although, the results provided in this thesis demonstrate the accuracy and efficiency of the framework used there is still more work to be done in this field such that soft robots can match the abilities of biological beings as well as rigid-bodied robots. This chapter is intended to discuss some of the challenges with the framework developed and provide insight into the future work to be done.

5.1 Summary

This section deals with the contribution of this thesis. In chapter 2 we use the Extended Dynamics Mode Decomposition developed by Williams *et. al* [10] to demonstrate the application of the Koopman operator in the system identification of soft robots. We saw that in chapter 2 although the EDMD algorithm was able to successfully able to approximate the Koopman operator that could very accurately describe the dynamics of the soft robot in a linear fashion but failed when the same

framework was used to a more complex system that consisted of 40 of such soft manipulators to form a grid. The primary reason for this is that accuracy of the Koopman operator approximation depends on the type of basis function used and in the case of the EDMD algorithm there is no systematic method leading to the right choice of basis function. When using the EDMD algorithm the researchers solely rely on trial and error for the selection of the right lifting functions which is often time-consuming and does not guarantee that one will find the right set of lifting functions. One might be able to use the knowledge about the dynamics of the system being identified to come up with the right set of lifting functions but again this information about the system is not available all the time.

The problem associated with the right selection of lifting function was solved in chapter 3 where we utilized the power of deep learning to learn the "best" lifting functions for a given dynamical system from input-output data. This method only needs input-output data and a couple of hyper-parameters to approximate the Koopman operator for a given dynamical system giving us the freedom from the choice of lifting function and then in chapter 4, we formulate the problem of shape control into an optimization problem. The shape control problem could be readily solved using any optimal control methods. To demonstrate this we solve the shape control problem for a single soft manipulator using Linear Model Predictive Controller and for the 4x4 soft grid we use Linear Quadratic Regulator to achieve shape control.

5.2 Future Work

Although the results presented for modeling and control of soft robots in this thesis show promising results, especially with the use of Deep Neural Networks to learn the best lifting functions to approximate the Koopman operator the major drawback of this approach is that since the system identification process is an offline process, any changes in the physical system will not be reflected identified model and this will lead to very poor performance and control of a soft robot. Further work in the deep learning framework developed in this chapter might address this problem by either having a very computationally efficient data-driven online system identification or

an offline method that could learn the changes made in the system or environment and update the identified model as well.

Bibliography

[1] Faheem Ahmed, Muhammad Waqas, Bushra Javed, Afaque Manzoor Soomro, Suresh Kumar, Hina Ashraf, Umair Khan, Kyung Hwan Kim, and Kyung Hyun Choi. Decade of bio-inspired soft robots: A review. *Smart Materials and Structures*, 2022.

[2] Dylan Shah, Bilige Yang, Sam Kriegman, Michael Levin, Josh Bongard, and Rebecca Kramer-Bottiglio. Shape changing robots: bioinspiration, simulation, and physical realization. *Advanced Materials*, 33(19):2002882, 2021.

[3] Benjamin Shih, Dylan Shah, Jinxing Li, Thomas G Thuruthel, Yong-Lae Park, Fumiya Iida, Zhenan Bao, Rebecca Kramer-Bottiglio, and Michael T Tolley. Electronic skins and machine learning for intelligent soft robots. *Science Robotics*, 5(41):eaaz9239, 2020.

[4] Ryan L Truby, Robert K Katzschmann, Jennifer A Lewis, and Daniela Rus. Soft robotic fingers with embedded ionogel sensors and discrete actuation modes for somatosensitive manipulation. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 322–329. IEEE, 2019.

[5] David Roylance. Finite element analysis. *Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge*, 2001.

[6] Daniel Bruder, Xun Fu, R Brent Gillespie, C David Remy, and Ram Vasudevan. Data-driven control of soft robots using koopman operator theory. *IEEE Transactions on Robotics*, 37(3):948–961, 2020.

[7] Noel Naughton, Jiarui Sun, Arman Tekinalp, Tejaswin Parthasarathy, Girish Chowdhary, and Mattia Gazzola. Elastica: A compliant mechanics environment for soft robotic control. *IEEE Robotics and Automation Letters*, 6(2):3389–3396, 2021.

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

[9] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[10] 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, 2015.

[11] Yoshinobu Kawahara. Dynamic mode decomposition with reproducing kernels for koopman spectral analysis. *Advances in neural information processing systems*, 29, 2016.

[12] Bobak Mosadegh, Panagiotis Polygerinos, Christoph Keplinger, Sophia Wennstedt, Robert F Shepherd, Unmukt Gupta, Jongmin Shim, Katia Bertoldi, Conor J Walsh, and George M Whitesides. Pneumatic networks for soft robotics that actuate rapidly. *Advanced functional materials*, 24(15):2163–2170, 2014.

[13] Thomas Morzadec, Damien Marcha, and Christian Duriez. Toward shape optimization of soft robots. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 521–526. IEEE, 2019.

[14] Metin Sitti. Miniature soft robots—road to the clinic. *Nature Reviews Materials*, 3(6):74–75, 2018.

[15] Yongchang Zhang, Pengchun Li, Jiale Quan, Longqiu Li, Guangyu Zhang, and Dekai Zhou. Progress, challenges, and prospects of soft robotics for space applications. *Advanced Intelligent Systems*, n/a(n/a):2200071.

[16] Daniel Bruder, Audrey Sedal, Ram Vasudevan, and C. David Remy. Force generation by parallel combinations of fiber-reinforced fluid-driven actuators. *IEEE Robotics and Automation Letters*, 3(4):3999–4006, 2018.

[17] Larry L. Howell, Ashok Midha, and Tony W. Norton. Evaluation of equivalent spring stiffness for use in a pseudo-rigid-body model of large-deflection compliant mechanisms. *Journal of Mechanical Design*, 118:126–131, 1996.

[18] Noel Naughton, Jiarui Sun, Arman Tekinalp, Tejaswin Parthasarathy, Girish Chowdhary, and Mattia Gazzola. Elastica: A compliant mechanics environment for soft robotic control. *IEEE Robotics and Automation Letters*, 6(2):3389–3396, 2021.

[19] PETER J. SCHMID. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

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

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

[22] X Zhang, FK Chan, T Parthasarathy, and M Gazzola. Modeling and simulation of complex dynamic musculoskeletal architectures. *Nature Communications*, 10(1):1–12, 2019.

[23] Vít Cibulka, Tomáš Haniš, and Martin Hromčík. Data-driven identification of vehicle dynamics using koopman operator. In *2019 22nd International Conference on Process Control (PC19)*, pages 167–172. IEEE, 2019.

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

[25] D. J. Alford-Lago, C. W. Curtis, A. T. Ihler, and O. Issan. Deep learning enhanced dynamic mode decomposition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3):033116, 2022.

[26] Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[27] Haojie Shi and Max Q.-H. Meng. Deep koopman operator with control for nonlinear systems. *IEEE Robotics and Automation Letters*, 7(3):7700–7707, 2022.

[28] Mengnan Li and Lijian Jiang. Deep learning nonlinear multiscale dynamic problems using koopman operator. *Journal of Computational Physics*, 446:110660, 2021.

[29] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990.

[31] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[32] Chunbing Wu and Wen Zheng. A modeling of twisted and coiled polymer artificial muscles based on elastic rod theory. *Actuators*, 9(2), 2020.

[33] Jiefeng Sun, Brandon Tighe, Yingxiang Liu, and Jianguo Zhao. Twisted-and-coiled actuators with free strokes enable soft robots with programmable motions. *Soft robotics*, 8(2):213–225, 2021.

[34] George M Whitesides. Soft robotics. *Angewandte Chemie International Edition*, 57(16):4258–4273, 2018.

[35] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, 2015.

[36] Matteo Cianchetti, Cecilia Laschi, Arianna Menciassi, and Paolo Dario. Biomedical applications of soft robotics. *Nature Reviews Materials*, 3(6):143–153, 2018.

[37] Panagiotis Polygerinos, Nikolaus Correll, Stephen A Morin, Bobak Mosadegh, Cagdas D Onal, Kirstin Petersen, Matteo Cianchetti, Michael T Tolley, and Robert F Shepherd. Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing,

control, and applications in human-robot interaction. *Advanced Engineering Materials*, 19(12):1700016, 2017.

[38] Kim J Quillin. Kinematic scaling of locomotion by hydrostatic animals: ontogeny of peristaltic crawling by the earthworm lumbricus terrestris. *Journal of Experimental Biology*, 202(6):661–674, 1999.

[39] RH Armour and JFV Vincent. J bionic eng. 2006, 3, 195-208; c) l. van griethuijsen, b. trimmer. *Biol. Rev*, 89:656–670, 2014.

[40] Dylan S Shah, Joshua P Powers, Liana G Tilton, Sam Kriegman, Josh Bongard, and Rebecca Kramer-Bottiglio. A soft robot that adapts to environments through shape change. *Nature Machine Intelligence*, 3(1):51–59, 2021.

[41] Dohgyu Hwang, Edward J Barron III, ABM Tahidul Haque, and Michael D Bartlett. Shape morphing mechanical metamaterials through reversible plasticity. *Science Robotics*, 7(63):eabg2171, 2022.

[42] Olivier Goury and Christian Duriez. Fast, generic, and reliable control and simulation of soft robots using model order reduction. *IEEE Transactions on Robotics*, 34(6):1565–1576, 2018.

[43] Gianmarco Mengaldo, Federico Renda, Steven L Brunton, Moritz Bächer, Marcello Calisti, Christian Duriez, Gregory S Chirikjian, and Cecilia Laschi. A concise guide to modelling the physics of embodied intelligence in soft robotics. *Nature Reviews Physics*, pages 1–16, 2022.

[44] Jiefeng Sun and Jianguo Zhao. Modeling and simulation of soft robots driven by embedded artificial muscles: an example using twisted-and-coiled actuators. In *2022 American Control Conference (ACC)*, pages 2911–2916. IEEE, 2022.

[45] Maria L Castaño, Andrew Hess, Giorgos Mamakoukas, Tong Gao, Todd Murphey, and Xiaobo Tan. Control-oriented modeling of soft robotic swimmer with koopman operators. In

2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pages 1679–1685. IEEE, 2020.

[46] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[47] David A Haggerty, Michael J Banks, Patrick C Curtis, Igor Mezić, and Elliot W Hawkes. Modeling, reduction, and control of a helically actuated inertial soft robotic arm via the koopman operator. *arXiv preprint arXiv:2011.07939*, 2020.

[48] Ervin Kamenar, N Ćrnjarić-Žic, D Haggerty, Sasa Zelenika, Elliot W Hawkes, and I Mezić. Prediction of the behavior of a pneumatic soft robot based on koopman operator theory. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1169–1173. IEEE, 2020.

[49] Cosimo Della Santina, Robert K Katzschmann, Antonio Biechi, and Daniela Rus. Dynamic control of soft robots interacting with the environment. In *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pages 46–53. IEEE, 2018.

[50] Ben Pawlowski, Charles W Anderson, and Jianguo Zhao. Dynamic control of soft robots using reinforcement learning. In *Dynamic Systems and Control Conference*, volume 59155, page V002T14A006. American Society of Mechanical Engineers, 2019.

[51] Thomas George Thuruthel, Egidio Falotico, Federico Renda, and Cecilia Laschi. Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics*, 35(1):124–134, 2019.

[52] Cosimo Della Santina, Antonio Bicchi, and Daniela Rus. On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control. *IEEE Robotics and Automation Letters*, 5(2):1001–1008, 2020.

[53] Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model based control of soft robots: A survey of the state of the art and open challenges. *arXiv preprint arXiv:2110.01358*, 2021.

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