DATA-DRIVEN SAFETY & SECURITY OF CYBERPHYSICAL SYSTEMS

by

Omanshu Thapliyal

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Inseok Hwang, Co-Chair

School of Aeronautics and Astronautics

Dr. Arthur E. Frazho, Co-Chair

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Shaoshuai Mou

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory A. Blaisdell

Dedicated to my parents — Om Prakash and Rajani Thapliyal.

ACKNOWLEDGMENTS

The works in this dissertation, and the accompanying writing would not have been possible without the help, motivation, inputs, and support from many people. I would like to thank my advisor, Professor Inseok Hwang, whose constant guidance throughout my graduate career has resulted in parts of the work discussed in this dissertation. I would also like to thank my co-advisor, Professor Arthur Frazho, who never fails to provide the brightest ideas in the most unorthodox manner, and always finds a way to root practical engineering problems in rigorous frameworks. Thanks are due to my advisors (both, Prof. Hwang, and Prof. Frazho) for being exemplary of Freeman Dyson's 'birds and frogs' in academic research (see [1]). My gratitude is also owed to my committee members, Professor Dengfeng Sun, and Professor Shaoshuai Mou, for their input on the disseration, and on my research, in general.

I would also like to extend my thanks to my collaborators, Dr. Shanelle Clarke and (soon to be Dr.) Sounghwan Hwang for their inputs, honest criticisms, and patient contributions. I would also like to thank my labmates at FD&C/HSL, and my friends, Ankit Deo, Raj Deshmukh, Radhika Ravi, and Jairaj Desai (the former for making life at Purdue better by their company, and the latter for making it worse by graduating sequentially).

It would be amiss to not acknowledge the near unlimited support from my fiancée, Ridhi Saran. Lastly, I would like to thank my parents and my sisters for displaying incredible patience and emotional support, without whom none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

2D	two-dimensional
3D	three-dimensional
ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
CPS	Cyber-Physical System
DES	Discrete Event System
DMD	Dynamic Mode Decomposition
DMDc	Dynamic Mode Decomposition with Control
DOF	Degree-of-Freedom
DoS	Denial of Service
EDMD	Extended Dynamic Mode Decomposition
FDI	False Data Injection
GAN	Generative Adversarial Network
HJ	Hamilton-Jacobi
LP	Linear Program
LTI	Linear Time Invariant
MAS	Multi-Agent Systems
MILP	Mixed-Integer Linear Program
MSE	Mean Squared Error
NCS	Network Control System
NN	Neural Network
PDE	Partial Differential Equation
ReLU	Rectified Linear activation Unit
RL	Reinforcement Learning
SDP	Semi-Definite Program
UAM	Urban Air Mobility
UAV	Unmanned Aerial Vehicle

ABSTRACT

Cyberphysical systems (CPSs) are expected to operate in safety-critical scenarios, and are increasingly getting distributed and physically separated. CPSs are characterized by complex dynamical behavior arising from emergent inter-agent interactions, having discrete logic-based programs, data-driven methods employed in-the-loop, or by simply having highly nonlinear dynamics. Despite this, safety and security properties for CPSs need to be computed, often in real-time over analytically accurate solutions of the associated high dimensional partial differential equations (PDEs). In this dissertation, we investigate numerical approximation schemes to compute safety properties (or reachable sets) for CPSs with differing natures of complexities, without solving the associated PDEs. We solve for reachable sets for unknown dynamical systems with polynomial approximations. Similar approximation schemes can be extended to multi-agent systems and dynamical systems with neural-networks-in-the-loop. Such systems are increasingly applicable in real life instances, such as internet of things, urban air mobility, and data-driven controllers in-the-loop. We utilize the system's trajectory data to compute equivalent system models, and utilize the data-driven models to find approximate reachable sets using polytopic or interval approximations, thereby side stepping PDE solutions. We also investigate cyberphysical vulnerabilities in CPSs from emergent multi-agent behavior, and single agent interacting with multiple controllers via supervisory cyber layers. Each problem is accompanied with associated illustrative examples and numerical simulations. Finally, we present an extensive discussion of possible directions for future work, both, that result directly from the works presented in this dissertation, and those that stem from the assumptions that can be handled immediately.

1. INTRODUCTION

Cyberphysical systems (CPSs) find applications from internet-of-things, to sensor networks, to software systems, to robotic teams, and urban and advanced air mobility scenarios. These CPSs are characterized by possibly geographically distributed hardware (physical layers), working towards some common computations/objectives (function/application layers), while collaborating via relaying useful information (communication layers) via communication channels and protocols (information layer), while being operated by business entities engaging in cooperative/competitive market environments (business layer). An instantiation of the same is illustrated in the form of the power grid CPS in Fig. 1.1.

As a result, CPSs lie at the intersection of *Computation, Communication, and Control* (see Fig. 1.2). CPSs leverage functionalities from control, communication and computation

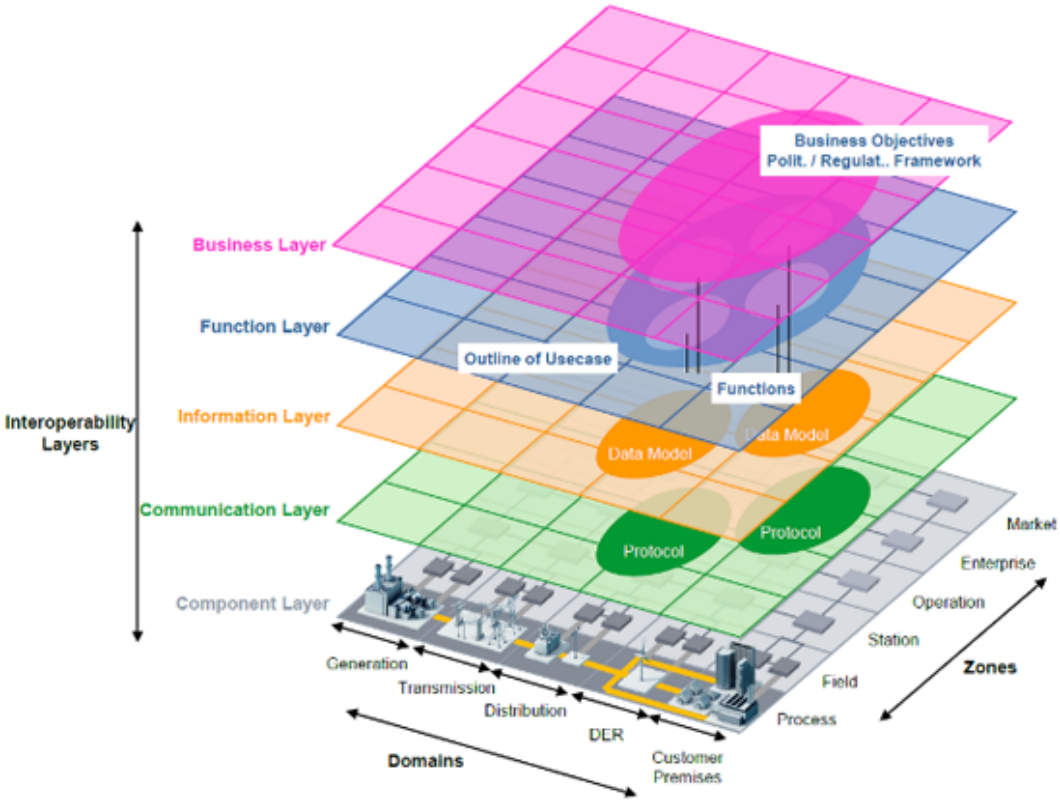

Figure 1.1. The power grid Cyberphysical System

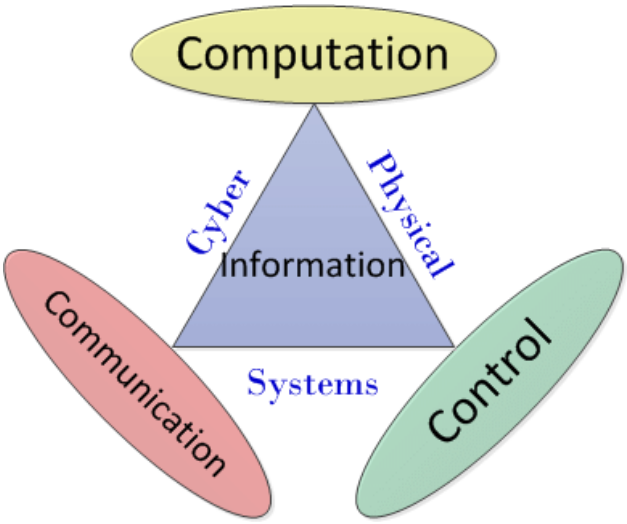

Figure 1.2. Cyberphysical Systems lie at the intersection of control, communication, and computation systems

(C^3) technologies and advances, but also suffer from the pitfalls of C^3 systems and procedures. Additionally, most CPSs are expected to operate in safety critical environments (e.g., power grids, unmanned aerial vehicles (UAVs), urban air mobility (UAM) operational environments, etc.). The focus of this work lies in safety and security properties of CPSs, and how the abundance of available data can be used in a model-driven manner to comment upon these properties, or uncover points of failures.

While most emphasis on safety has been provided from a model-driven perspective, data-driven approaches present a novel semi model-driven method by employing model-discovery. As data becomes more inexpensive, and system capabilities highly nonlinear, and often subject to parameter variations during mission, data-driven approaches are utilized more and more in controls engineering. Further, online and real-time applications increasingly require unknown models and nonlinear interactions to be learned for control synthesis.

To this end, more established fields of system identification are increasingly utilizing data-driven models by using neural networks, autoregressive models, and other machine learning techniques. Due to the expressive nature of such machine learning techniques, they are utilized to learn models of unknown dynamical systems from observed data sequences.

In such approaches, a 'discovered model' is arrived at using data, and employed to perform control design. Even though purely model-driven are slowly getting harder to apply to more recent, realistic, and complex scenarios, they provide a strong basis to study extensions of similar approaches to data-driven techniques for estimation and control. In this research, we attempt to utilize model-driven techniques and extend them to study safety and security properties of cyberphysical systems using data. *The primary aim of this work is to study data-driven approaches to safety, and data-driven vulnerabilities against cyberattacks, of complex control systems.*

1.1 Research Objectives

The research presented in this dissertation concerns itself with the following objectives.

(O1): **Data-Driven System Safety under Parameter Variations:**
Dynamical systems often include data-driven schemes (e.g., Neural Networks (NNs)) in-the-loop to: a) perform system identification from input-output data, and b) devise model-free control law for systems with unknown parameters. *The objective of this research task is to investigate reachability property of dynamical systems with a data-driven system model, while incorporating unstructured changes in system parameters.* For example, consider an unmanned ground robot with a nominal safety computation module. During operation, it loses traction on one of its wheels. Computing reachability property for such scenarios to maintain control authority in real-time, during a mission, is of high importance. This allows the system to adjust to parameter variation during runtime and makes NN based functional approximation methods ubiquitous is control design. However, ensuring safety, viability, or any other set-based properties for systems with parameter variation is worsened due to the presence of NNs in-the-loop. To include parameter variations (e.g., a fault, a failure, or a partial failure), the state space reachability of the system should deviate from the nominal reachability property. If the modified reachable set of the system is considered, a control law to maintain the systems state withing the new set also preserves the systems safety. The

modified reachable sets provide a pruned state space set to search for the new control law for the compromised system.

(O2): **Multi-Agent System Safety under Flexible Mission Requirements:**
Multi-Agent applications are characterized by flexibility in mission requirements. Therefore, associated reachability property computation has to accommodate for the following: a) flexibility in mission profile for different agents, and b) ability to compute reachable sets in a distributed manner, especially when reachable set computation is coupled among agents in a network. *The objective of this research task is to study multi-agent system safety, subject to time-varying network topologies, with distributed computational resources.* For example, the urban air mobility scenario requires unmanned aerial vehicles to coordinate, despite having limited varied objectives and missions. This requires the UAVs to coordinate for shorter duration, followed by pursuing private objectives. Temporary collaboration also implies the reachable sets of the agents are related to each other's state, despite having little information about each other's states.

(O3): **Cybersecurity Vulnerabilities of Complex Cyberphysical Systems against Data-Driven Attacks:**
Complex for the scope of this research stands for multi-agent systems, with parameter uncertainties. Due to their nature, most complex systems employ a complex interaction of physical layer (sensors and actuators), communication layer (communication channels along with their protocols), and cyber layers (computer programs and logic, and supervisory controllers). The additional dependence on cyber layers poses new vulnerabilities that can be exploited by data-driven methods, by a gray box or even white box attackers. *The objective of this research task is to try to uncover newer modes of cybersecurity that can be exploited by an attacker, by virtue of collecting enough data.*

The rest of this dissertation is organized as follows (as shown in Fig. 1.3). Chapters 2, 3, and 4 deal with safety properties in CPSs of varying degrees of complexity. Chapters 5 and 6 address cybersecurity issues present in complex CPSs operating with supervisory layers and across networks, respectively.

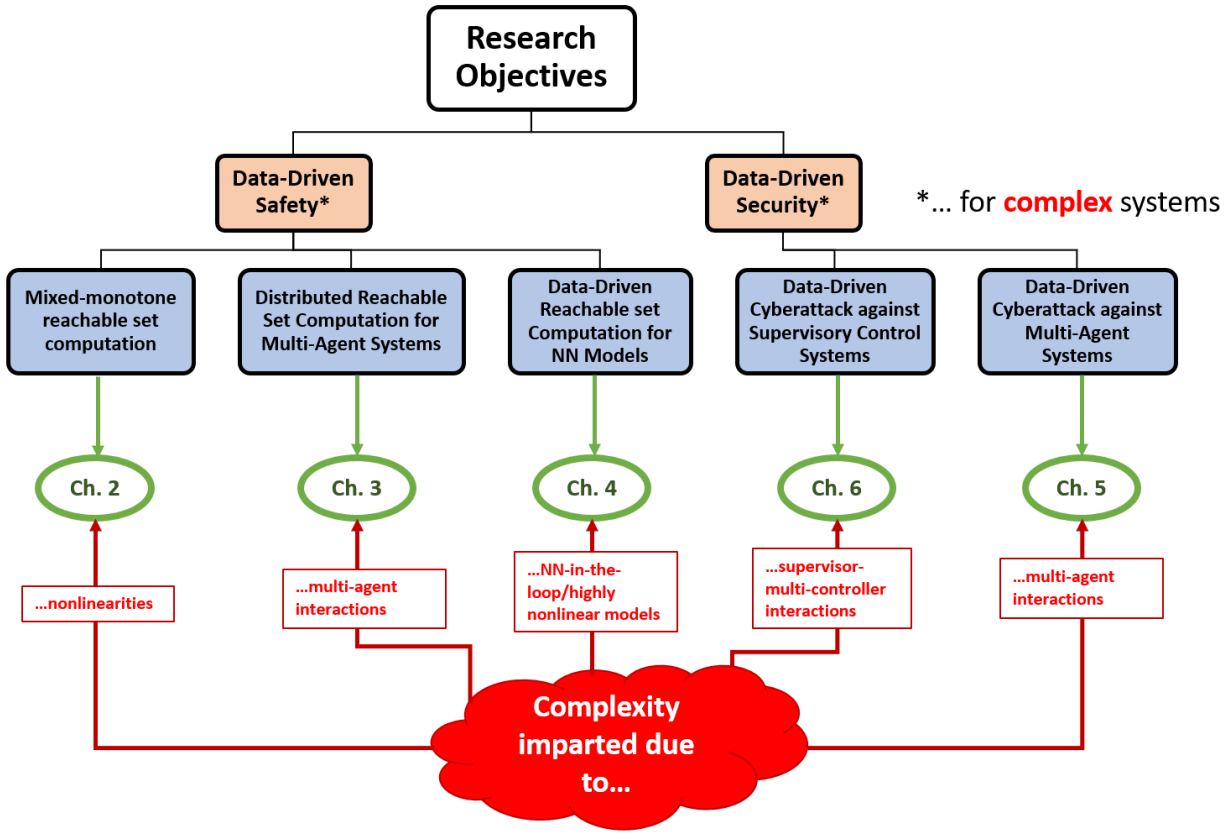

Figure 1.3. Organization of this dissertation

In Chapter 2, we pose the safety property as the problem of computing forward reachable sets for a dynamical system with unknown dynamical models. Under certain conditions, the unknown dynamics can be identified by lifting to observable spaces of higher dimensions, using Koopman Operator theory. The resultant lifted system with polynomial dynamics can be leveraged to obtain rapid over-approximations of reachable sets using mixed monotone decomposition theory. In Chapter 3, we utilize the Koopman Operator theory discussed in the previous Chapter, and apply it to compute real-time approximations of forward reachable sets for neural network-in-the-loop dynamical systems. We develop rapid estimations for reachable sets for NN-in-the-loop control systems, by collecting data from the NN controller by exciting it and noting its output for given trajectory data points. Chapter 4 extends the optimal control based ideas of polytopic reachability to multi-agent systems (MASs). The resulting computations are realized to be distributed equation solving, and are carried out using local information at individual agents in the MAS. Convergence guarantees are provided when the underlying graphs satisfy certain conditions.

Chapter 5 discusses cybersecurity vulnerabilities of MASs against combinations of denial of service (DoS) and false data injection (FDI) attacks of given budgets. It is realized that for MASs engaged in common goals, smarter attackers can choose an agent and cause it to no longer conform to the prescribed missions, agreed upon by the remaining network. In Chapter 6, FDI attack vulnerabilities of supervisory control systems are presented. The problem of performing FDI attacks is posed as a system identification problem, followed by an adversarial machine learning problem to cause unwanted switching in the supervisory logic's outputs. It is demonstrated that similar data-driven and machine learning-based schemes can be employed to solve the dual problem of detecting such injection attacks on the supervisory controller's logic.

Finally, in Chapter 7, we present the concluding remarks, and posit a number of recommendations going forward towards the future works.

2. REACHABLE SETS FOR POLYNOMIAL TYPE DYNAMICS VIA DYNAMIC MODE DECOMPOSITION

Thapliyal, O., & Hwang, I. (2022). Approximate Reachability for Koopman Systems Using Mixed Monotonicity. *IEEE Access*, 10, 84754-84760. [2]
The research presented in this chapter is conducted by **Thapliyal, O.** under the supervision of Hwang, I.

Abstract

In this chapter, a novel approach for computing reachable sets for unknown nonlinear dynamical systems is presented. We propose a method that utilizes Koopman operator to perform system identification, and then utilizes a high dimensional embedding to find approximations for the forward-time reachable sets. The proposed method uses a Koopman operator-based approach and is data-driven, which means it can be applied to systems where mathematical models are unknown or incomplete. Analytical methods to compute reachable sets resort to solving the Hamilton-Jacobi (HJ) partial differential equations (PDEs), and approximation methods find numerical solutions to the PDEs. Either way, both suffer from the 'curse of dimensionality' that characterizes the HJ PDE. To overcome the curse of dimensionality, the proposed method employs mixed-monotone decompositions for a class of Koopman lifted dynamics. This mixed-monotone system is then embedded in a higher-dimensional dynamical model, which is propagated deterministically in time. This enables the computation of over-approximations of forward reachable sets that are accurate while being computationally efficient.

The proposed method is capable of accounting for unknown nonlinear dynamics and providing conservative approximations of reachable sets to a desired degree of accuracy. To demonstrate the effectiveness of the algorithm, an illustrative example with an unknown, nonlinear dynamical model is used. The results are compared to a well-known existing method, highlighting the advantages of the proposed approach.

2.1 Introduction

As dynamical systems become more complex, it becomes increasingly important to guarantee certain properties for controlled systems. While point-wise properties are useful, set-based properties are particularly crucial when dealing with larger-scale systems. Set properties allow us to comment on the properties of bundles, tubes, or collections of trajectories over time. The reachability property of a dynamical system has recently gained significant attention, particularly for the verification and validation of safety-critical systems. The general reachability problem involves finding the set of possible state space solutions to a dynamical model, given a set of admissible initial states and admissible control inputs. This set of solutions forms a "tube" of trajectories starting from the initial conditions on the state space. The ability to calculate reachable sets is crucial in ensuring safety-critical properties of systems, as it allows for analysis of the set of states that a system can reach under different conditions. Such reachability analysis has been used in optimal control [3], safety-critical systems [4], autonomous navigation [5], cybersecurity [6], and differential games [7].

The reachability problem is a critical aspect of the study of complex dynamical systems. Set-based properties of a dynamical system are important in determining the properties of collections of trajectories over time, and the reachability property is especially relevant for verifying and validating safety-critical systems. The general reachability problem involves determining the set of possible state space solutions for a given dynamical model, based on a set of admissible initial states and control inputs. There are three main methods for solving the reachability problem: Hamilton-Jacobi (HJ) reachability, verification methods, and numerical approximation methods. Optimal control or game theoretic formulations of reachability often rely on HJ partial differential equations [3], [7], [8], where reach sets are computed as level sets to obtain games of degree from the initial HJ formulation of games of kind [9], [10]. This simplifies the Boolean problem of determining if a target set is included in the reachable set. However, these formulations are limited by the 'curse of dimensionality', which leads to exponential increases in computational complexity as the system dimension grows. Readers are referred to [9] for a detailed survey of existing HJ reachability methodologies.

Verification methods in the form of formal verification [11] and temporal logic [12] offer a different approach to solving the reachability problem. Instead of using numerical methods, these methods use a formal structure to describe the state transitions, both continuous and discrete, of the dynamical system under consideration. These methods use a finite state machine to represent the system, with each state in the machine corresponding to a region in the state space of the dynamical system. The transitions between states in the machine represent the state transitions of the dynamical system. Verification results are obtained by checking if the system satisfies the desired properties by following runs of the finite state machine transitions. This requires a detailed description of the system properties in the form of logical formalism, such as temporal logic, which allows for the specification of system properties that should be satisfied throughout the evolution of the system. Similar verification techniques have been extended to hybrid system verification as well (see [12], [13], [14]).

The methods discussed previously, such as Hamilton-Jacobi (HJ) reachability and verification methods, are mainly useful for checking the finite state transitions of hybrid systems. However, they may not be as effective for dealing with general nonlinearities in dynamical models. Moreover, the expressibility of system properties and resulting sets obtained from these methods may not always be reliable or accurate. Therefore, it is important to consider alternative methods that can address the limitations of these approaches and provide more robust and accurate results.

In order to solve the reachability problem, numerical methods are used to either over- or under- approximate reachable sets and propagate reachability tubes. This approach is described in various works, such as [15]–[18]. To handle computational and expressibility issues that arise in HJ reachability, these methods employ various approximation and representation techniques. Examples of such techniques include ellipsoidal over-approximations [15], [19], [20], zonotope-based approximations [13], polyhedral approximations [21], and sampling-based methods [22]. The present chapter proposes a solution that addresses the challenges associated with computing conservative reachable sets in the context of unknown dynamical models. To achieve this goal, the chapter introduces an extension to existing numerical formulations that can efficiently incorporate unknown models. The proposed

method does not aim to develop a new system identification algorithm; instead, it leverages a Koopman operator-based approach for system identification. The Koopman operator is a mathematical tool that can be used to transform a nonlinear dynamical system into a linear infinite-dimensional system, enabling the use of linear methods for analysis and control. The proposed method utilizes the Koopman operator to identify the system's underlying dynamics and decompose them into mixed-monotone components. The mixed-monotone components are then embedded into a higher-dimensional dynamical model, allowing for the deterministic propagation of the system in time.

This approach provides over-approximations of the forward reachable sets that are not impacted by the curse of dimensionality. It also enables the computation of conservative approximations of the reachable sets, even in the presence of unknown nonlinear dynamics, in a computationally efficient manner. The proposed method's effectiveness is demonstrated using an illustrative example with an unknown, nonlinear dynamical model, and its performance is compared to that of a well-known existing method.

In this work, we propose a method to handle the issues mentioned earlier by utilizing a Koopman operator based approach to compute conservative reachable sets for systems with unknown nonlinear dynamical models. The Koopman operator is an infinite-dimensional linear operator that can be used to lift the dynamics of a system to a higher dimension, thereby linearizing the dynamics. By formulating the reachability of unknown nonlinear dynamical models in terms of the spectral properties of the Koopman operator, we can find finite-dimensional approximations of the operator using data-driven techniques. The properties of the original nonlinear function can be represented as transition maps that are dependent on the eigenfunctions of the Koopman operator, under a suitable choice of basis functions. Assuming the uniqueness of state transition maps and a connected, compact state space, the existence and uniqueness of a lifting through the Koopman operator is guaranteed [23]. By exploiting the properties of the Koopman operator, it becomes possible to express the reachability of unknown nonlinear dynamical models in terms of the spectral properties of the linear operator. The eigenfunctions can be thought of as generalized Laplacian averages over time [23]. Finite dimensional approximations of the operator can be easily found using data-driven techniques. With an appropriate selection of basis functions for the data-driven

technique, properties of the original nonlinear function can be expressed as transition maps that are dependent on the eigenfunctions.

Utilizing mixed-monotone embeddings for such suitable bases allows us to construct embeddings in higher dimensions with twice as many states, which we use to calculate set-based reachability properties as point-to-point trajectories in a deterministic setting [16]. This approach enables us to express over-approximations of reachable sets in a numerically efficient manner, making it applicable to systems with unknown nonlinear dynamical models. As a result, this chapter's primary contribution is the proposal of *an extension to numerical formulations to incorporate unknown dynamical models to compute conservative reachable sets efficiently using a Koopman operator-based method.*

The structure of this chapter is as follows. First, in Section 2.2, we will define the data-driven reachability problem for a nonlinear system. Then, in Section 2.3, we present the main results for data-driven reachability and introduce a mixed-monotone embedding for a Koopman lifting of the nonlinear system. This method enables us to solve a corresponding reachability problem of the Koopman observables using the selected basis functions. In Section 2.4, we demonstrate the effectiveness of the proposed algorithm through a numerical example. Finally, in Section 2.5, we offer concluding remarks and discuss future research directions.

Notations: Let a_{i} denote the i^{th} component of vector a, and let its transpose be denoted by a^T. The vector $\mathbb{1}$ is a vector of appropriate dimensions consisting of all ones. The inner product of two vectors a and b is denoted by $\langle a, b\rangle$. We also use the notation $a \leq (\text{or } \geq), b$ for two vectors a and b to mean that $a_{\mathrm{i}} \leq (\text{or } \geq), b_{\mathrm{i}}$ for all i. For a vector-valued function $\phi(x, y)$, we denote the partial derivative of ϕ with respect to x as ϕ_x. Similarly, $\phi_{x_{\mathrm{i}}}$ denotes the partial derivative of ϕ with respect to x_{i}. The set $\mathbb{R}_{\geq 0}$ is defined as $x \in \mathbb{R} : x \geq 0$. An extended hyperrectangle $\Gamma = [a, b]$ is defined as the set $\Gamma = x : a \leq x \leq b$ for $a, b \in \mathbb{R}^n$. A hyperrectangle $r := (a, b) \in \mathbb{R}^{2n}$ is defined as a bounded extended hyperrectangle, denoted by $[![r]!] := [a, b]$. The composition of functions g and h is denoted by $g \circ h$, where $g : G \to F$ and $h : H \to G$, implying that $g \circ h : H \to F$. For any vector $x \in \mathbb{R}^{2n}$, $[![x]!]$ denotes a hyperrectangle defined by the first n and last n components of x.

2.2 Problem Formulation

Consider the nonlinear dynamics of state $x \in \mathcal{X}$ under control input u, as follows:

$$\dot{x} = f\left(x, u\right), \text{ and } x_0 \in \mathcal{X}_0, u : [0, T] \to \mathcal{U},\ t \in [0, T] \tag{2.1}$$

The dynamics in (2.1) induces a state-transition map $\Phi^f(t; x, u)$, where $\Phi^f(\tau; x, u) = x(\tau)$ for $t \geq 0$. The reachability problem is to compute the T-time reachable set under the dynamics f, given $\mathcal{X}_0$ and $\mathcal{U}$, defined as:

$$\begin{aligned} \mathcal{R}^f\left(T; x_0\right) &\coloneqq \{x(T) \in \mathcal{X} : \dot{x} = f(x, u), x_0 \in \mathcal{X}_0, u \in \mathcal{U}\} \\ &= \left\{\Phi^f(T; x_0, u)) \in \mathcal{X} : x_0 \in \mathcal{X}_0, u \in \mathcal{U}\right\} \end{aligned} \tag{2.2}$$

We assume that the dynamics f are unknown but can be characterized by state information from the transition map. Specifically, we have access to a 'time-moving snapshot' of the state-transition, which consists of the state transition data $x_{\mathrm{i}}, \cdots, x_{\mathrm{i}+N}$ and $x_{\mathrm{i}+1}, \cdots, x_{\mathrm{i}+N+1}$ for $\mathrm{i} = 0, \Delta t, \cdots, \mathrm{i}\Delta t, \cdots$, with a sampling time of Δt. This snapshot is obtained by sampling the continuous time state evolution, where $x_{\mathrm{i}} = \Phi^f(\mathrm{i}\Delta t; x_0, u)$.

To solve the reachability problem defined in (2.2) using the snapshot data, we make the following assumptions.

A1. The sets $\mathcal{X}_0$ and $\mathcal{U}$ are both finite and bounded, which implies that they are also compact when viewed as subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Specifically, we can bound $\mathcal{X}_0$ and $\mathcal{U}$ using hyperrectangles of the form $[\underline{x}, \overline{x}]$ and $[\underline{u}, \overline{u}]$, respectively.

A2. For all times $\tau \in [0, T]$, there exists a unique state transition map Φ^f which is induced by the dynamics f.

A3. The unknown dynamics can be reconstructed from state-output data using the complete state x and control input u, which are accessible at all times. It is important to note that in practice, state information is often obtained from onboard sensor data and the associated state estimation methods.

2.3 Data-Driven Reachability

In order to solve the problem in (2.2), we need to determine a state-transition map $\Phi^{\tilde{f}}$ that can be used to calculate approximate reachable sets $\mathcal{R}^{\tilde{f}}(T; x_0)$ for a surrogate model $\tilde{f}$. To develop the surrogate model $\tilde{f}$ for the nonlinear system in (2.1), we will use the "snapshot" data. For the nonlinear system in (2.1), the set of all observables is defined as the space $\mathcal{F}$ which includes all functions of the form $g : \mathcal{X} \times \mathcal{U} \to \mathbb{C}$.

For most dynamical systems, the space $\mathcal{F}$ of observables is a Hilbert space of square-integrable functions [23]. In this space, the Koopman operator is defined as an operator $\mathcal{K}_f^t : \mathcal{F} \to \mathcal{F}$ that acts on an observable $g \in \mathcal{F}$ as $\mathcal{K}_f^t g(x, u) = g(\Phi_t^f(x, u))$. The operator $\mathcal{K}_f^t$ maps observables to observables, and it describes the evolution of the system's observables over time. Specifically, it describes how a given observable g evolves over time under the dynamics f. By repeatedly applying the operator $\mathcal{K}_f^t$, one can compute the time evolution of any observable g in the system. Then, the Koopman operator is defined as operator $\mathcal{K}_f^t : \mathcal{F} \to \mathcal{F}$ that acts on the observable to itself as follows:

$$\mathcal{K}_f^t \circ g(x, u) \coloneqq g(f(x, u), u) \tag{2.3}$$

Assumption A2 implies the existence and uniqueness of operator K_f^t, as stated in [23]. Furthermore, as the operator K_f^t defined in (2.3) is linear, its eigendecomposition can be defined. The eigenfunctions of $\mathcal{K}$ are denoted as special observables $\varphi_{\mathrm{j}} \in \mathcal{F}$ defined as:

$$\mathcal{K}\varphi_{\mathrm{j}}(x, u) = \lambda_{\mathrm{j}}\varphi_{\mathrm{j}}(x, u),\ \mathrm{j} \in \mathbb{N} \tag{2.4}$$

where λ_{j}'s are the corresponding eigenvalues.

Given compact sets $\mathcal{X}$ and $\mathcal{U}$ in Euclidean spaces $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, and observables $g \in \mathcal{F}$ that span $\mathcal{F}$, one can find the Koopman eigenfunctions as a linear combination. The Koopman operator $\mathcal{K}_f^t : \mathcal{F} \to \mathcal{F}$ acts on the observable g and is defined as follows: for any $g \in \mathcal{F}$ and $t \in [0, T]$, $\mathcal{K}_f^t g$ is the observable g evaluated at $\Phi^f(t, x_0, u)$, the state of the system at time t starting from the initial condition x_0 and applying the input u. To find the Koopman eigenfunctions, one can perform a spectral analysis of the operator $\mathcal{K}_f^t$ on a

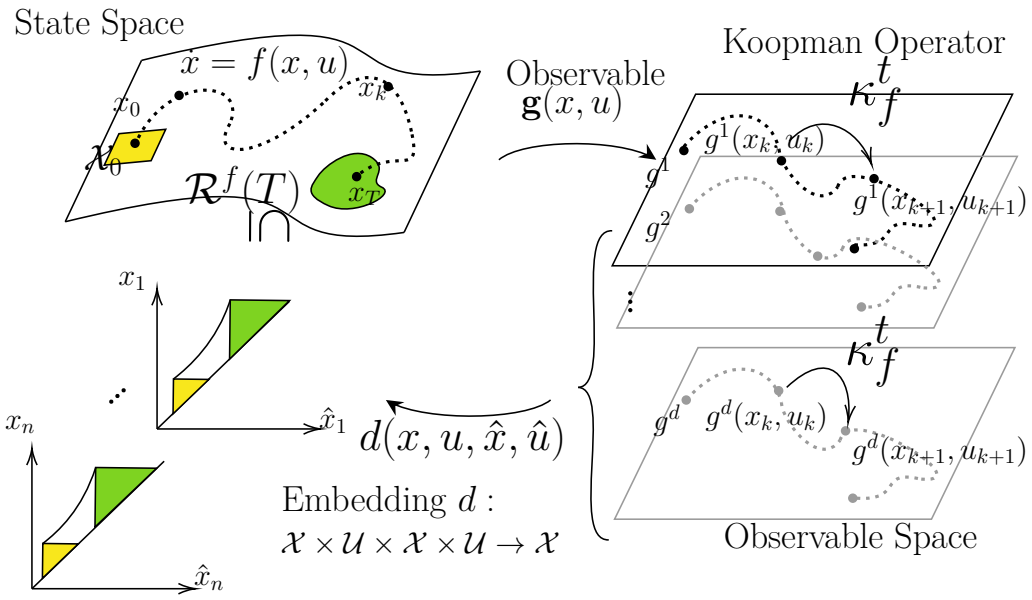

Figure 2.1. Reachability for EDMD system using a mixed-monotone embedding

suitable basis for $\mathcal{F}$. This can be done by finding the eigenfunctions ψ_{i} and eigenvalues λ_{i} that satisfy $\mathcal{K}_f^t \psi_{\mathrm{i}} = \lambda_{\mathrm{i}} \psi_{\mathrm{i}}$, where $\psi_{\mathrm{i}} \in \mathcal{F}$ and $\lambda_{\mathrm{i}} \in \mathbb{C}$.

When $\mathcal{F}$ is a Hilbert space of square-integrable functions, such as for most dynamical systems, one can express any observable $g \in \mathcal{F}$ as a linear combination of the eigenfunctions ψ_{i}, i.e., $g(x,u) = \sum_{\mathrm{i}=1}^{\infty} c_{\mathrm{i}} \psi_{\mathrm{i}}(x,u)$, where $c_{\mathrm{i}} \in \mathbb{C}$ [23]. This allows one to approximate the operator $\mathcal{K}_f^t$ using a finite number of eigenfunctions and eigenvalues. Spectral techniques, such as extended dynamic mode decomposition (EDMD), can be used to compute these eigenfunctions and eigenvalues from the snapshot data.

That is:

$$\mathcal{K}_f^t \circ g(x,u) = \sum_{k=1}^{\infty} \xi_k \varphi_k(x,u) \tag{2.5}$$

The EDMD technique is utilized to approximate the infinite-dimensional Koopman operator $\mathcal{K}_f^t$ using the snapshot data. This allows for the determination of dissipative or conservative dynamical modes based on the eigenvalues [24], [25].

2.3.1 Extended Dynamic Mode Decomposition (EDMD)

By using the state transition data, the infinite-dimensional operator in (2.3) can be utilized to compute the discrete-time dynamics of observable g as $\mathcal{K}_f^t g(x_k, u_k) = g(x_{k+1}, u_k)$. As per (2.5), vector-valued observables can be expressed in terms of a linear combination of Koopman eigenfunctions, given as follows:

$$\boldsymbol{g}(x,u) = \begin{bmatrix} g_1(x,u) \\ g_2(x,u) \\ \vdots \\ g_l(x,u) \end{bmatrix} = \sum_{k=1}^{\infty} \xi_k \varphi_k(x,u) = \langle \boldsymbol{\xi}, \boldsymbol{\varphi}(x,u) \rangle \tag{2.6}$$

The evolution of observables can be written using the definition in (2.3) as $\mathcal{K}_f^t \boldsymbol{g}(x,u)$, which can be rewritten using (2.4) and (2.6) as:

$$\mathcal{K}_f^t \boldsymbol{g}(x,u) = \mathcal{K}_f^t \sum_{k=1}^{\infty} \xi_k \varphi_k(x,u) = \sum_{k=1}^{\infty} \lambda_k \xi_k \varphi_k(x,u) \tag{2.7}$$

Thus, the spectral properties of $\mathcal{K}_f^t$ not only provide a representation of the evolution of $\boldsymbol{g}$, but also allow for lifting measurements of the form $y = \boldsymbol{g}(x,u)$ to obtain a linear mapping from data $\Psi(Y)$ to $\Psi(Y^+)$. In other words, an eigenfunction decomposition of the infinite-dimensional system can be obtained from the finite-dimensional data in the form of 'snapshots' of the unknown nonlinear system. Hence, when a sufficiently large observable set is available, the spectral properties of the Koopman operator are directly tied to EDMD eigenvalues [24]. Without loss of generality, let the snapshot data contain the full state information in the form of $z_k \coloneqq [x_k^T, u_k^T]^T$ as follows:

$$Z = [z_0, \cdots, z_N], \; Z^+ = [z_1, \cdots, z_{N+1}] \tag{2.8}$$

, where x_k and u_k are the state and control inputs at time step k, respectively. Then, the corresponding snapshot data under the observable map is defined as:

$$\begin{aligned} \Psi(Z) &= [\boldsymbol{g}(z_0), \cdots, \boldsymbol{g}(z_N)] \\ \Psi(Z^+) &= [\boldsymbol{g}(z_1), \cdots, \boldsymbol{g}(z_{N+1})] \end{aligned} \tag{2.9}$$

The EDMD approximation to $\mathcal{K}_f^t$ can then be viewed as a solution to the minimization problem:

$$\tilde{K} \coloneqq \underset{K \in \mathbb{R}^{n \times (n+l)}}{\arg\min} \sum_{\mathrm{i}=1}^{N} \|\Psi(Z^+) - K\Psi(Z)\| \tag{2.10}$$

Numerous methods exist for performing the minimization above in numerically efficient ways (see [24], [26]). The matrix $\tilde{K}$ that was introduced in (2.10) serves as a finite dimensional representation of the operator $\mathcal{K}_f^t$ and, in turn, our surrogate model $\tilde{f}$. By computing the eigenvalues of $\tilde{K}$, we obtain the eigenvalues of $\mathcal{K}_f^t$. Additionally, the eigenfunctions of $\mathcal{K}_f^t$ can be obtained by projecting z_k onto the left eigenvectors of $\tilde{K}$. The eigendecomposition $\tilde{K}v_{\mathrm{j}} = \lambda_{\mathrm{j}} v_{\mathrm{j}}$ and $w_{\mathrm{j}}\tilde{K} = \mu_{\mathrm{j}} w_{\mathrm{j}}$ allows rewriting the observable evolutions as:

$$\boldsymbol{g}(z_{k+1}) = \sum_{\mathrm{j}} \langle \boldsymbol{g}(z_k), w_{\mathrm{j}} \rangle v_{\mathrm{j}} \tag{2.11}$$

The idea of using the approximate spectral properties of the Koopman operator, which are obtained through $\tilde{K}$, to propagate observable dynamics is based on a well-known result [24]. The computation of matrix $\tilde{K}$ is typically accomplished through the singular value decomposition (SVD) of data matrices $\Psi(X)$ and $\Psi(X^+)$. While for linear systems, the Koopman operator and matrix $\tilde{K}$ are equivalent representations, this is not the case for non-linear systems due to finite snapshot size and the necessity of finite-dimensional truncation. Therefore, some error is always incurred in the resulting approximation. To address the approximation, we quantify the approximation error incurred in using the EDMD technique for reconstructing the state $\tilde{x}_k$ through (2.11), using r data-points, when the true state is given by $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$. This error bound will be used to improve the reachability analysis discussed subsequently.

Theorem 2.3.1 (EDMD Error Bounds)**.** *Suppose the state is reconstructed from EDMD $\tilde{x}_k$ through (2.11), using r data-points, and the true state is given by $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$. Then there exist $\underline{\varepsilon}_{n,r}$ and $\overline{\varepsilon}_{n,r}$, such that $x_k \in [\tilde{x}_k - \underline{\varepsilon}_{n,r}, \tilde{x}_k + \overline{\varepsilon}_{n,r}]$ (where $a \leq b$ implies $a_{\mathrm{i}} \leq b_{\mathrm{i}}$, element-wise).*

Proof. From (2.5), the EDMD reconstructed observables evolve according to the discrete time recursive relation $\boldsymbol{g}(\tilde{x}_{k+1}, u_{k+1}) = \tilde{K}\boldsymbol{g}(\tilde{x}_k, u_k)$, while the original Koopman system evolves according to $g(x_{k+1}, u_{k+1}) = \mathcal{K}_f^t g(x_k, u_k)$. Let the matrix $\tilde{K}$ be calculated using r data-points, for the state $x \in \mathcal{X} \subseteq \mathbb{R}^n$. From Theorem 8.4 in [25], $\|\mathcal{K}_f^t \boldsymbol{g} - \tilde{K}\boldsymbol{g}\| \leq c\,(\log r/r)^{2n/n+1}$ for $c > 0$. Let $\delta_{n,r} = c\,(\log r/r)^{2n/n+1}$, then $\|\mathcal{K}_f^t \boldsymbol{g}\| \in [\tilde{K}\boldsymbol{g} - \delta_{n,r}, \tilde{K}\boldsymbol{g} + \delta_{n,r}]$. Therefore, $\tilde{K}\boldsymbol{g} - \delta_{n,r}\mathbb{1} \leq \mathcal{K}_f^t \boldsymbol{g} \leq \tilde{K}\boldsymbol{g} + \mathbb{1}\delta_{n,r}$ necessarily holds. Let $\Pi_{\mathcal{X}}$ be the projection on to $\mathcal{X}$, such that the state x is obtained from the observables as $\Pi_{\mathcal{X}}\boldsymbol{g}$. That is, $\Pi_{\mathcal{X}}\left(\tilde{K}\boldsymbol{g} - \delta_{n,r}\mathbb{1}\right) \leq \mathcal{K}_f^t \boldsymbol{g} \leq \Pi_{\mathcal{X}}\left(\tilde{K}\boldsymbol{g} + \delta_{n,r}\mathbb{1}\right)$. Finally, let $\overline{\varepsilon}_{n,r} = \delta_{n,r}\Pi_{\mathcal{X}}\mathbb{1} = \underline{\varepsilon}_{n,r}$ for the proof to hold. □

2.3.2 Mixed-Monotone Embedding for the EDMD System

So far, the choice of observables $\boldsymbol{g}$ has not been restricted, leaving room for problem-specific guidelines. Indeed, the challenges associated with observable choice is very much an open problem. However, for the purpose of solving the reachability problem, we will limit our discussion to polynomial observables of degree at most d. Polynomials are a natural choice for square-integrable observables on compact sets and are commonly used in the EDMD/DMD literature [24]. Henceforth, we denote the set of observables as $\boldsymbol{g}^d$, which consists of *polynomial basis functions with degree at most d*. This restriction on the observables will be essential in utilizing mixed-monotonicity results from [16], [27] for reachability analysis. These results provide a way to analyze the monotonicity of (hyperrectangular) sets under dynamical systems, and have been used in various control and optimization problems. The restriction to polynomial observables of degree at most d allows us to leverage these results towards the reachability problem.

In order to proceed, we will begin by providing a definition for the *mixed-monotonicity* of a function $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. From [16], $F(u, v)$ is *mixed-monotone*, if for some $\underline{J}_u \in \mathbb{R}^{n\times n}_{\leq 0}, \underline{J}_v \in \mathbb{R}^{n\times m}_{\leq 0}, \overline{J}_u \in \mathbb{R}^{n\times n}_{\geq 0}$ and $\overline{J}_v \in \mathbb{R}^{n\times m}_{\geq 0}$, the following holds true:

$$\begin{aligned} &F_u(u, v) \in [\underline{J}_u, \overline{J}_u], F_v(u, v) \in [\underline{J}_v, \overline{J}_v], \ \forall u \in \mathcal{U}, v \in \mathcal{V} \\ &(\underline{J}_u)_{\mathrm{i,j}} > -\infty \text{ or } (\overline{J}_u)_{\mathrm{i,j}} < \infty, \ \forall \mathrm{i} \neq \mathrm{j}, \text{ and} \\ &(\underline{J}_v)_{\mathrm{i,j}} > -\infty \text{ or } (\overline{J}_v)_{\mathrm{i,j}} < \infty, \ \forall \mathrm{i}, \mathrm{j} \end{aligned} \tag{2.12}$$

Subsequently, we can consider an embedding into a higher dimensional space given by the decomposition $d : \mathcal{U} \times \mathcal{V} \times \mathcal{U} \times \mathcal{V}$. The map $F(u, v)$ is considered to be mixed-monotone with respect to d if, for all $u, \hat{u} \in \mathcal{U}$ and $v, \hat{v} \in \mathcal{V}$, the following condition holds true (refer to [16], [28]):

$$\begin{aligned} F(u, v) &= d(u, v, u, v), \\ \Big(d_{u_\mathrm{j}}(u, v, \hat{u}, \hat{v})\Big)_\mathrm{i} \geq 0, \ \Big(d_{\hat{u}_\mathrm{j}}(u, v, \hat{u}, \hat{v})\Big)_\mathrm{i} &\leq 0, \ \forall \mathrm{i}, \mathrm{j} \\ \Big(d_{v_\mathrm{j}}(u, v, \hat{u}, \hat{v})\Big)_\mathrm{i} \geq 0, \ \Big(d_{\hat{v}_\mathrm{j}}(u, v, \hat{u}, \hat{v})\Big)_\mathrm{i} &\leq 0, \ \forall \mathrm{i}, \mathrm{j} \end{aligned} \tag{2.13}$$

The concept of mixed-monotonicity is an important tool for analyzing the behavior of dynamical systems. In particular, mixed-monotonicity with respect to a decomposition d is defined in Equation (2.13) and involves a higher dimensional embedding of the function F. Specifically, a function F is said to be mixed-monotonic with respect to d if it is monotonically increasing with respect to the variables (u, v) and monotonically decreasing with respect to the variables $(\hat{u}, \hat{v})$.

In other words, if we consider the function F as a map from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$, then mixed-monotonicity requires that F is non-increasing along any coordinate direction in one of the two sets of coordinates (u, v) or $(\hat{u}, \hat{v})$, and non-decreasing along any coordinate direction in the other set of coordinates. In particular, mixed-monotonicity with respect to a decomposition d can be used to derive sufficient conditions for reachability that are computationally tractable, making it a powerful tool for analyzing complex dynamical systems.

Lemma 1 (Existence of Decomposition d)**.** *Consider the EDMD dynamics given below:*

$$x_{k+1} = \Pi_{\mathcal{X}} \left(\sum_{\mathrm{j}} \langle \mathbf{g^d}, w_{\mathrm{j}} \rangle v_{\mathrm{j}} \right) + \zeta_k =: F^d(x, \zeta) \tag{2.14}$$

where $\zeta \in [\underline{\varepsilon}_{n,r}, \overline{\varepsilon}_{n,r}]$ is an unknown but bounded vector. A decomposition function $d : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \times \mathcal{U}$ always exists w.r.t. which the dynamics in (2.14) is mixed-monotone.

Proof. The set of all polynomial functions of (x, u), of degree at most d, is denoted by $\mathcal{F}_d$. Since we have chosen a basis, it follows that $\langle \boldsymbol{g^d}, w_{\mathrm{j}} \rangle \in \bigvee \boldsymbol{g^d} = \mathcal{F}_d$. Moreover, since its argument $y \in \mathcal{F}d$, $\Pi\mathcal{X}(y) \in \mathcal{F}d$. Therefore, by construction, the EDMD dynamics in (2.14) are polynomials of degree at most d. Finally, Assumption A1 implies that $\boldsymbol{g^d_x}$ and $\boldsymbol{g}^d_u$ are both bounded above and below since the sets $\mathcal{X}0$ and $\mathcal{U}$ are compact, which in turn yields $\underline{J}(\cdot)$ and $\overline{J}(\cdot)$ as defined in (2.12). Therefore, due to Theorem 1 in [28], the dynamics in (2.14) is mixed-monotone w.r.t. the decomposition d defined as follows:

$$d_{\mathrm{i}}(x, \zeta, \hat{x}, \hat{\zeta}) = \begin{cases} \min\limits_{\substack{v \in [x, \hat{x}] \\ v_{\mathrm{i}} = x_{\mathrm{i}} \\ w \in [\zeta, \hat{\zeta}]}} F^d_{\mathrm{i}}(v, w) & \text{if } x \leq \hat{x} \text{ and } \zeta \leq \hat{\zeta} \\ \max\limits_{\substack{v \in [\hat{x}, x] \\ v_{\mathrm{i}} = x_{\mathrm{i}} \\ w \in [\hat{\zeta}, \zeta]}} F^d_{\mathrm{i}}(v, w) & \text{if } x \geq \hat{x} \text{ and } \zeta \geq \hat{\zeta} \end{cases} \tag{2.15}$$

The decomposition above is tight for $F^d(\cdot, \cdot)$ due to [28]. □

Given Lemma 1, it is always possible to find a decomposition function d such that the dynamics $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$ are mixed-monotone. We can then define the state transition function for the EDMD system $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$ as $\Phi^g(T; x_0, u)$, such that $x(t) = \Phi^g(t; x_0, u)$. Using this, we can define the forward reachable set of the EDMD system $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$, similar to (2.2), as:

$$\mathcal{R}^g(T; x_0) \coloneqq \left\{ x : x_{k+1} = \Pi_{\mathcal{X}} \langle \boldsymbol{g^d}, w_{\mathrm{j}} \rangle v_{\mathrm{j}} : x_0 \in \mathcal{X}_0, u \in \mathcal{U} \right\} \tag{2.16}$$

Theorem 2.3.2 (Mixed Monotone Forward Reachability)**.** *Let the EDMD dynamics be mixed-monotone w.r.t. the decomposition d in (2.15). Further define the following embedding by fixing the unknown vector ζ as:*

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \Theta(x, \hat{x}) \coloneqq \begin{bmatrix} d(x, \underline{\varepsilon}_{n,r}, \hat{x}, \overline{\varepsilon}_{n,r}) \\ d(\hat{x}, \underline{\varepsilon}_{n,r}, x, \overline{\varepsilon}_{n,r}) \end{bmatrix} \tag{2.17}$$

Let the corresponding state transition function for Θ be given by $\Phi^{\Theta}(t; (x, \hat{x}))$, then if $\mathcal{X}_0 = [\underline{x}, \overline{x}]$, the original reachable set for the unknown dynamics in (2.1) satisfies:

$$\mathcal{R}^f(T; x_0) \subseteq [\![\Phi^{\Theta}(t; (\underline{x}, \overline{x}))]\!] \tag{2.18}$$

Proof. Theorem 2.3.2 guarantees that $x_k \in [\tilde{x} - \underline{\varepsilon} n, r, \tilde{x} + \overline{\varepsilon} n, r]$, which implies that $\mathcal{R}^f(T; x_0) \subset \mathcal{R}^g(T; x_0)$, where $\mathcal{R}^f(T; x_0)$ and $\mathcal{R}^g(T; x_0)$ denote the forward reachable sets of systems f and g, respectively. To determine the reachable set of the mixed-monotone system g, we can define the state transition function for the EDMD system $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$ as $\Phi^g(T; x_0, u)$ such that $x(t) = \Phi^g(t; x_0, u)$. Additionally, we can define the forward reachable set of the EDMD system $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$. We can always find a decomposition function d with respect to which the dynamics $\langle \boldsymbol{g^d}(z_k), w_{\mathrm{j}} \rangle$ are mixed-monotone, according to Lemma 1. Moreover, since the mappings $d(x, \cdot, \cdot, \cdot)$ and $d(\cdot, \cdot, \hat{x}, \cdot)$ are polynomials in $\boldsymbol{g^d}$, they can be expressed explicitly from (2.15). This implies that the state transition function $\Phi^{\Theta}(t; (x, \hat{x}))$ can be found in closed form. Based on Theorem 2 from [16], since $\Theta(x, \hat{x})$ is a polynomial embedding, we have $\mathcal{R}^g(T; x_0) \subseteq [\![\Phi^{\Theta}(t; (\underline{x}, \overline{x}))]\!]$, where $[\![\cdot]\!]$ denotes the hyperrectangle bounding the set. Hence, we have $\mathcal{R}^f(T; x_0) \subseteq \mathcal{R}^g(T; x_0) \subseteq [\![\Phi^{\Theta}(t; (\underline{x}, \overline{x}))]\!]$. In other words, the solutions of the transition function Θ in x and $\hat{x}$ coordinates, provide the lower and upper corners, respectively, of the hyperrectangle bounding the reachable set of g. □

The findings from Theorem 2 offer a computationally efficient approach to determine forward reachable sets by utilizing the deterministic embedding Θ stated in (2.17). Since we have opted for polynomial basis functions as observables in our EDMD technique, an embedding Θ can always be found through the d decomposition. The approach presented here

extends the reachability analysis framework proposed in [27], [28] to account for unknown dynamics models. Moreover, [27] provides techniques for computing the decomposition d in a straightforward manner. Notably, computing d involves solving min/max problems of polynomials, which can be tackled over compact sets by leveraging Assumption A1. The assumption that $\mathcal{X}_0 = [\overline{x}, \underline{x}]$ is not restrictive since bounding hyperrectangles can always be obtained for any bounded $\mathcal{X}_0$. The proposed method combines unknown model reachable problem with computationally efficient reachable set computation, albeit at the cost of hyperrectangular approximations to the reachable set. As the method relies on EDMD approximations of f, the accuracy of the approximations improves with the number of data points used to construct $\mathcal{K}_f^t$. Therefore, the proposed method is particularly useful for safety-critical systems that require fast updates to reachable set computations and can tolerate conservative estimates of the sets. If the unsafe region lies outside of the over-approximated reachable sets, the original system's safety is guaranteed.

Therefore, by selecting polynomial observables of degree at most d, we can construct an embedding that is mixed monotone, allowing us to compute over-approximations of reachable sets for dynamical systems using EDMD. This approach offers a complete, end-to-end framework for tackling computationally intensive reachable set computations for systems with unknown models. While the over-approximations may be conservative, they provide a quick update to the reachable set computations, making this method especially suitable for safety-critical systems where swift response times are crucial. Additionally, it is worth noting that finding bounding hyperrectangles for bounded $\mathcal{X}_0$ is always possible, so the assumption of $\mathcal{X}_0 = [\overline{x}, \underline{x}]$ is not restrictive. Finally, since the accuracy of the approximations improves with an increasing number of data-points, the method can be expected to yield increasingly accurate results as the size of the dataset grows. The proposed method is summarized in Algorithm 1.

Remark 1. *The action of the Koopman operator over observables is a lifting to a higher dimensional space, in order to utilize its spectral properties, where the linear operator's spectral properties can be utilized to study the unknown dynamics. Similarly, the decomposition d can*

be viewed as a lifting w.r.t. which the original dynamics are mixed-monotone, as summarized in Fig. 2.1.

Algorithm 1: Mixed Monotone Reachability for EDMD System

Input: $T, Z, Z^+, \mathcal{X}_0 \subseteq [\underline{x}, \overline{x}]$, $\mathcal{U} \subseteq [\underline{u}, \overline{u}]$

Parameters : $\boldsymbol{g}^d, d, r$

while $\tau \leq T$ **do**

obtain $\Psi(Z), \Psi(Z^+)$

solve for $\tilde{K}$ using EDMD (2.10)

calculate $\underline{\varepsilon}, \overline{\varepsilon}$

compute decomposition d using (2.15)

for i $\leftarrow 0$ **to** n **do**

solve for min/max F_{i}^d using `fmincon`

end

compute embedding Θ using (2.17)

propagate deterministic system $\Phi^{\Theta}(\tau; \underline{x}, \overline{x})$

end

return **Output:** $[\![\Phi^{\Theta}(T; (\underline{x}, \overline{x})]\!]$

2.4 Numerical Example

In this section, we will consider the Laub-Loomis model, a high dimensional nonlinear dynamical system introduced in [29]. This model is commonly used to study a class of enzymatic activities. The equation describing the model is as follows:

$$\begin{aligned}
&\dot{x}_1 = 1.4x_3 - 0.9x_1,\ \dot{x}_2 = 2.5x_5 - 1.5x_2,\\
&\dot{x}_3 = 0.6x_7 - 0.8x_2x_3,\ \dot{x}_4 = 2 - 1.3x_3x_4,\\
&\dot{x}_5 = 0.7x_1 - x_4x_5,\ \dot{x}_6 = 0.3x_1 - 3.1x_6,\\
&\dot{x}_7 = 1.8x_6 - 1.5x_2x_7
\end{aligned} \tag{2.19}$$

The Laub-Loomis model is often used to benchmark reachability problems (see [29] for benchmarking results). The initial set $\mathcal{X}_0 \subset \mathbb{R}^7$ for the Laub-Loom model in (2.19) is a hyperrectangle of fixed width $W = 0.15$, with a lower corner:

$$x_1(0) = 1.2, x_2(0) = 1.05, x_3(0) = 1.5, x_4(0) = 2.4,$$
$$x_5(0) = 1, x_6(0) = 0.1, x_7(0) = 0.45$$

The forward reachable sets are computed for $t \in [0, 10]$ seconds.

To construct EDMD approximations of the Laub-Loomis model given in (2.19), we use full-state information sampled via discretization at $\Delta T = 0.002$s. In this analysis, we restrict the choice of observables to $d = 2$. Notably, we observe that $\mathcal{F}_d$ includes the unknown dynamics of the system, meaning that $\bigvee \boldsymbol{g}^d$ contains the nonlinear dynamical map. As a result, the EDMD reconstruction of the system dynamics in (2.19) is exact. This is also seen in the trajectory reconstructed using EDMD in Fig. 2.2.

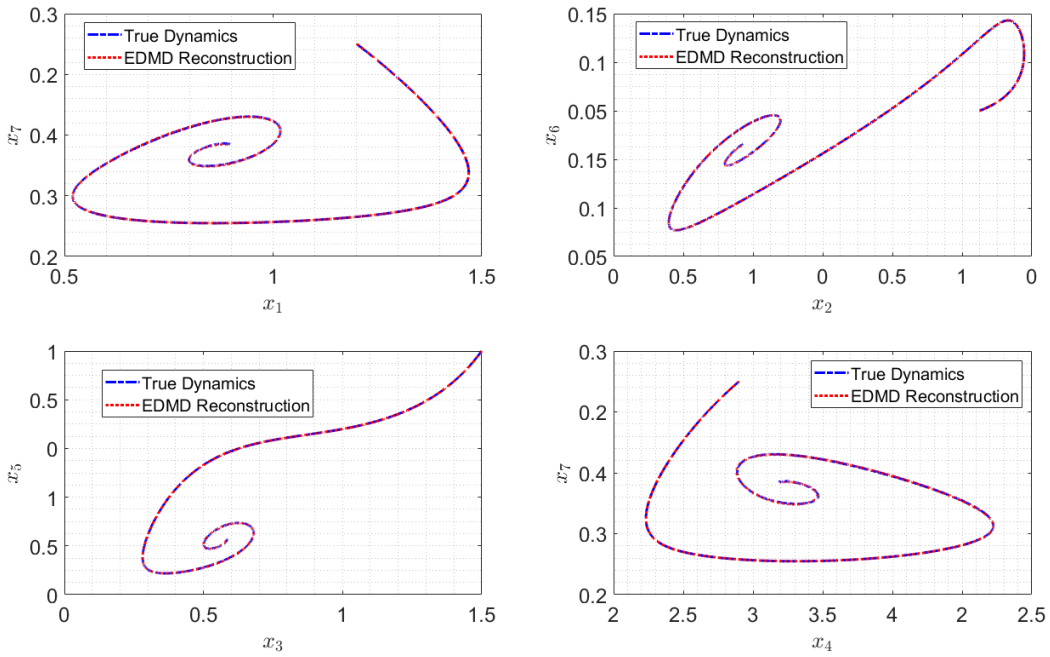

Figure 2.2. The Laub Loomis Model's true dynamics are plotted in blue and compared with the EDMD reconstruction, which is represented in red

Using the EDMD model reconstruction via observables of the form $\boldsymbol{g}^d$, from Theorem 1, we establish $\underline{\varepsilon}$ and $\overline{\varepsilon}$ values. To compute the forward reachable sets $\mathcal{R}^g(\tau; x_0)$, where $\tau \leq T$, we first construct d using (2.15). Next, we use the embedding Θ to propagate the deterministic system Φ^Θ forwards in time. Finally, we project the resulting propagations back onto $\mathcal{X}$ to obtain the reachable set for the original nonlinear system, i.e., $\mathcal{R}^f(\tau; x_0)$.

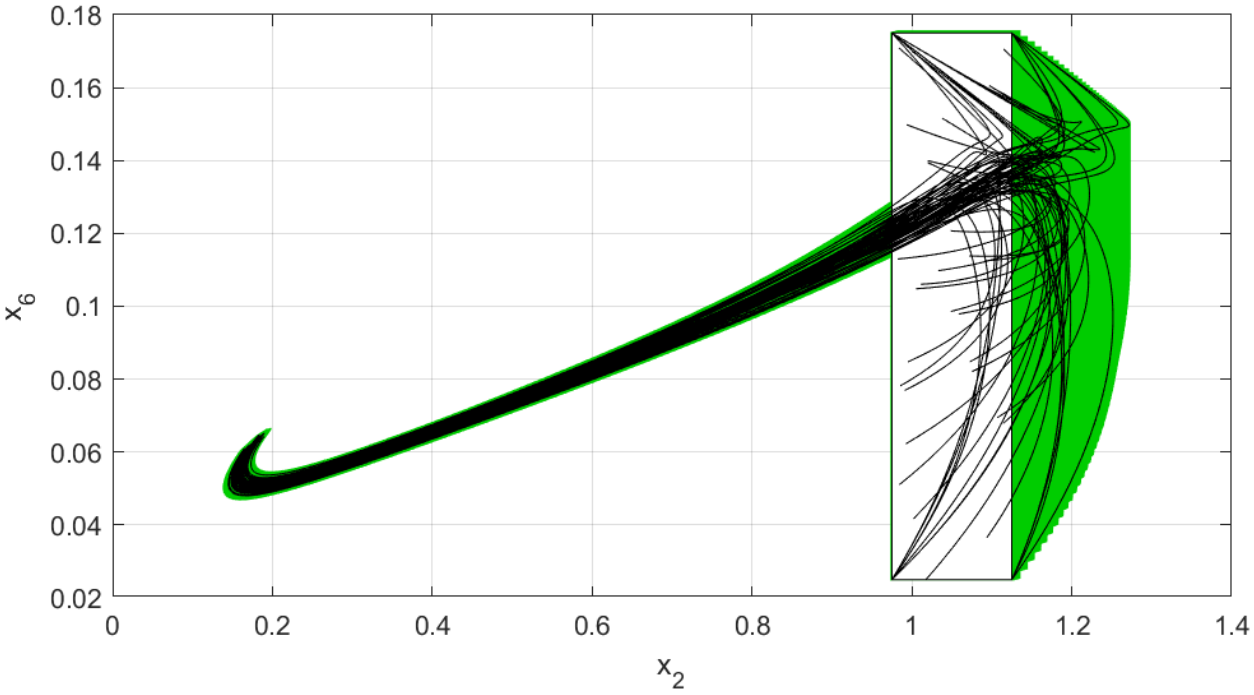

Figure 2.3. The CORA tool was used to compute the reachable set $\mathcal{R}^f(10; x_0)$ for the original Laub-Loomis system. The initial condition x_0 is denoted by the black circle, and the reachable set is represented by the light blue region.

A projection of the reachable set $\mathcal{R}^f(\tau; x_0)$ is shown in Figs. 2.3 and 2.4 in the $x_2 - x_6$ dimensions. Figure 2.3 displays the reachable sets computed using COntinuous Reachability Analyzer (CORA), a toolbox based on MATLAB (for further information, see [13]). In comparison, Fig. 2.4 shows the reachable sets obtained using the proposed method. Additionally, Figs. 2.5 and 2.6 present the reachable set projections in other dimensions for the Laub-Loomis model given in Equation 2.19.

According to Theorem 2, the reachable sets obtained from a mixed-monotone embedding of the EDMD system must necessarily contain the reachable set for the original system

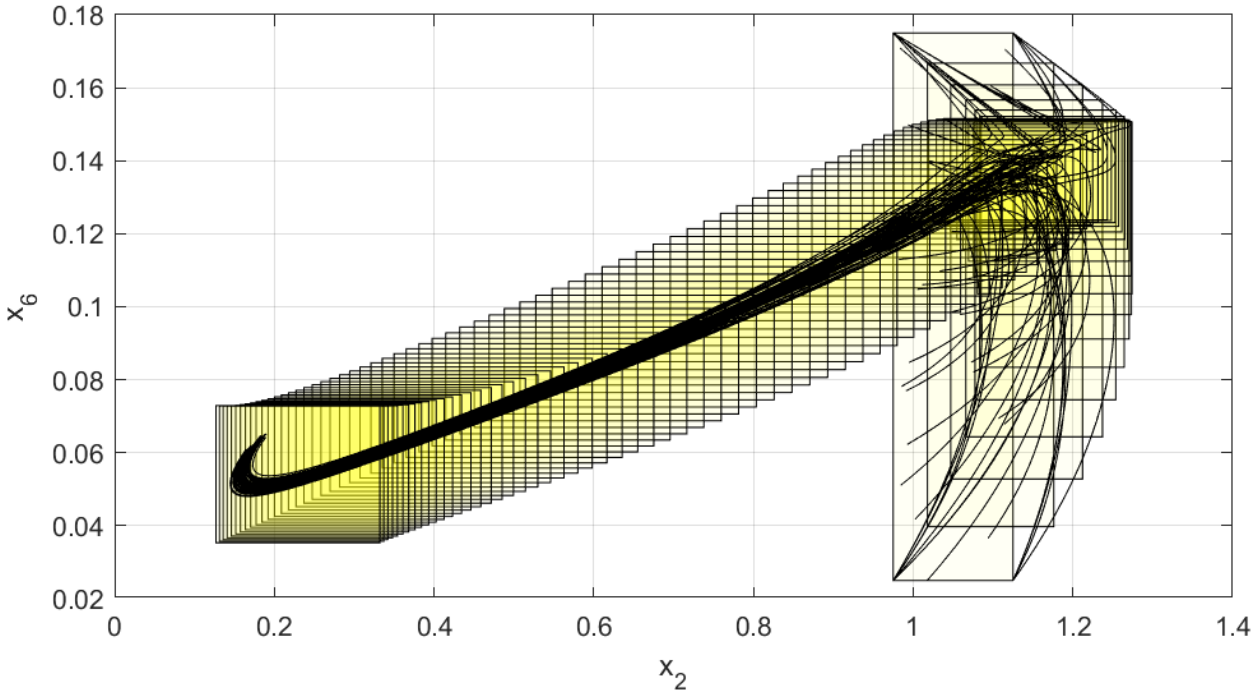

Figure 2.4. Over-approximation of the reachable set $\mathcal{R}^g(10; x_0)$ obtained using the proposed method for the EDMD reconstruction of the Laub-Loomis system

described by (2.19). This is illustrated in the figures, and we further confirm this in Fig. 2.4 by randomly sampling trajectories starting from $\mathcal{X}_0$. The figure provides evidence that the reachable sets computed using CORA lie within the over-approximations obtained using the proposed method's deterministic map Φ^{Θ}.

The proposed approach aims to over-approximate reachable sets of unknown dynamical models with faster computation times. This is evident in Fig. 2.7, where the proposed method consistently outperforms CORA. The experiments were conducted using MATLAB R2020a on a Windows 10 machine with an Intel Core i5-8265U 1.60GHz CPU and 16GB of RAM. It is worth noting that the proposed method performs significantly better than CORA for time horizons greater than 3 seconds. For time horizons shorter than 3 seconds, the computational overhead for computing the decomposition d and its corresponding embedding Θ is substantial. However, in practical applications, large time-horizon reachable sets are

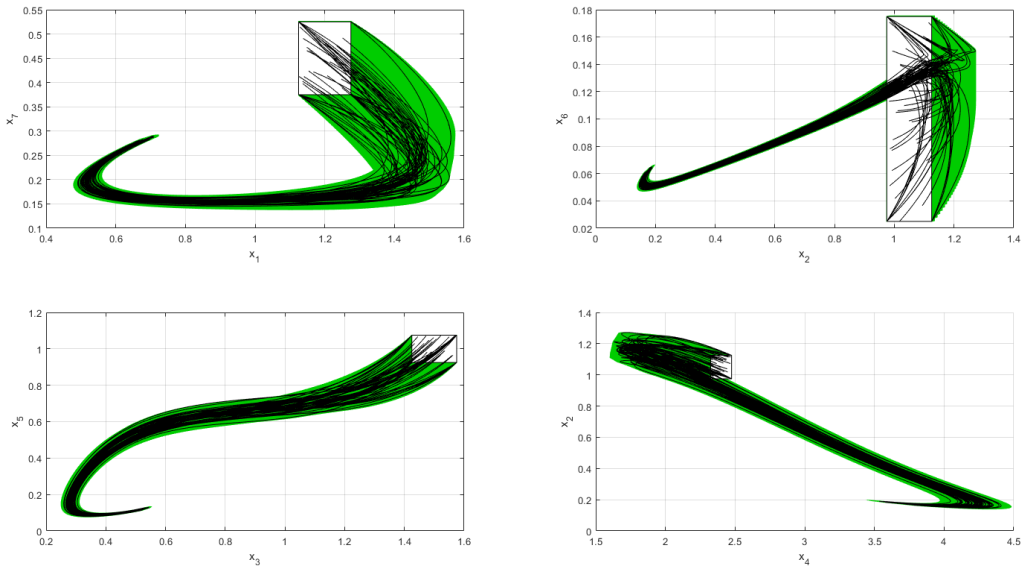

Figure 2.5. The reachable set $\mathcal{R}^f(10; x_0)$ for the Laub-Loomis system using CORA, projected onto the x_1-x_2 plane, x_1-x_3 plane, and x_2-x_3 plane

often used for trajectory planning, control synthesis, and verification and validation of the dynamical system.

Remark 2. *To obtain the embedding* Θ*, we solve the* min / max *problem in (2.15) using* `fmincon`*. However, for dynamical models with a more predictable structure, this computation can be pre-computed in terms of the chosen observables or off-loaded to reduce the computational burden of solving* `fmincon`*. This approach would significantly decrease the overhead cost of the method and make the red line in Fig. 2.7 almost flat.*

Remark 3. *Even though the decomposition* d *is guaranteed to be tight, as per [28], the hyperrectangles may not be tight in relation to the original dynamics. This can be observed in Figs. 2.5 and 2.6, where an additional "looser" rectangle is present, which arises due to modeling the EDMD error as residing in a separate hyperrectangle* $[\![\cdot, \cdot]\!]$.

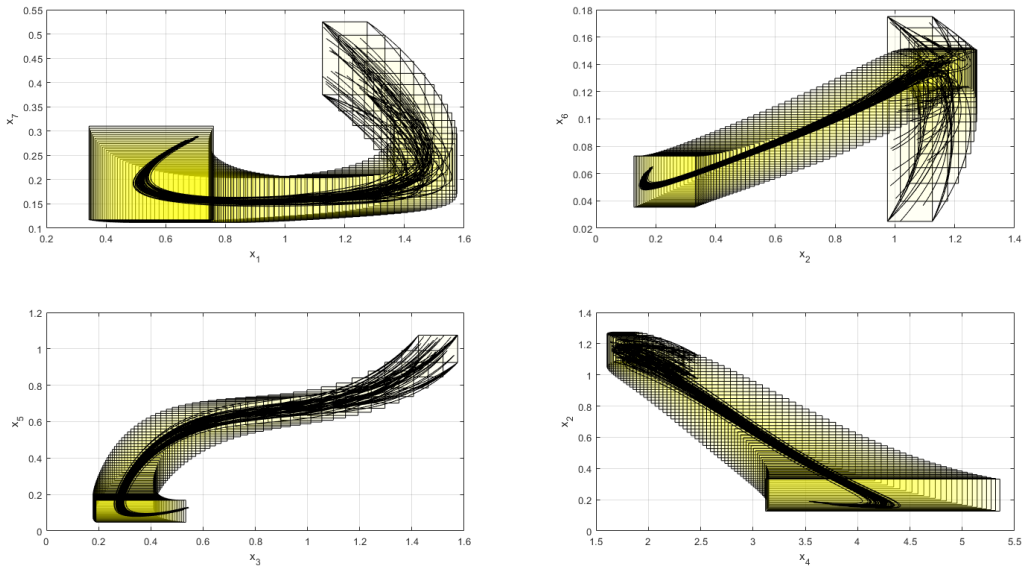

Figure 2.6. Projections of the hyperrectangle bounding the over-approximation of the reachable set $\mathcal{R}^g(10; x_0)$ for the Laub-Loomis system using the proposed method

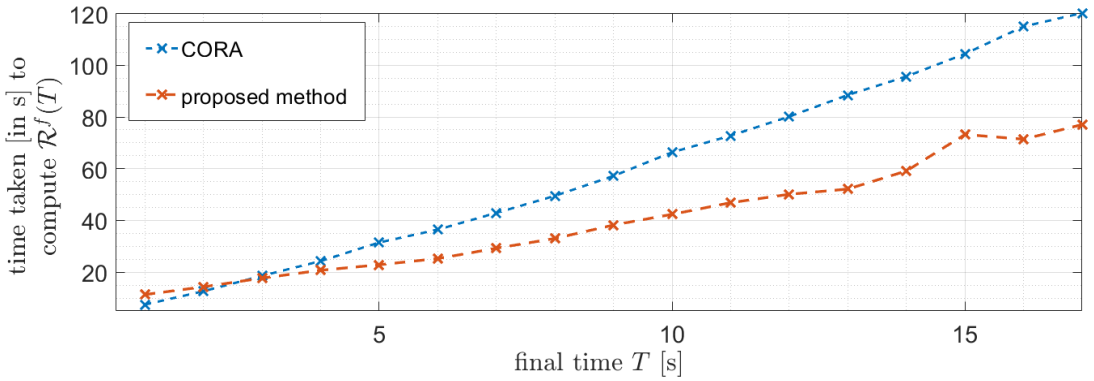

Figure 2.7. The computation time for generating forward reachable sets for the Laub-Loomis system is shown for different values of the final time T.

2.5 Conclusion and Future Work

This chapter introduces a novel data-driven method for approximating reachable sets of unknown nonlinear dynamical systems. Nonlinearities in the dynamics are represented by a linear operator acting on observable functions of the state and control input. The finite-dimensional EDMD method is used to estimate the infinite-dimensional Koopman operator from data snapshots. The resulting EDMD model is then embedded in a higher-dimensional space using mixed-monotone decomposition functions, which can always be found for EDMD models under the proposed form. Over-approximations of reachable sets are obtained by modeling bounded EDMD errors, which contain the original reachable sets. The proposed method is demonstrated on a high-dimensional nonlinear system example and shown to outperform a well-known existing algorithm in terms of computational time.

Our current and future work involves exploring ways to improve the proposed method for computing over-approximations of reachable sets for unknown nonlinear dynamical systems. Specifically, we are investigating tighter representations of embedding functions in higher dimensions, which would enable polytopic geometry to be used instead of interval-based approximations. This would help reduce the inherent conservativeness in the approach and provide more accurate approximations. Additionally, we plan to consider nonlinearities introduced by multi-agent interactions to extend the applicability of the proposed method to

more general nonlinear dynamical systems [30], [31]. Some of these multi-agent interaction scenarios include leaderless consensus and optimal control problems.

3. DMD-BASED REACHABLE SET COMPUTATION FOR NEURAL NETWORK-IN-THE-LOOP MODELS

Thapliyal, O., & Hwang, I. (2023). Approximating Reachable Sets for Neural Network-Based Models in Real Time via Optimal Control. *IEEE Transactions on Control Systems Technology*. [32]

The research presented in this chapter is conducted by **Thapliyal, O.** under the supervision of Hwang, I.

Abstract

In this chapter, a data-driven approach to estimate reachable sets for control systems using neural networks (NNs) is presented. The proposed method utilizes a quadrotor model as a running example, which is learned offline using trajectory data via NNs. During real-time operation, the learned NN can be excited to obtain linear approximations for reachability analysis. Dynamic mode decomposition is employed to obtain linear liftings of the NN model. The linear models obtained can be used with optimal control theory to obtain polytopic approximations of the reachable sets in real-time, with tunable accuracy. The proposed framework is not limited to quadrotor models and can be extended to other nonlinear models that utilize NNs to estimate plant dynamics. The effectiveness of the proposed approach is demonstrated through an illustrative simulation of quadrotor dynamics. In the future, the proposed framework can be further improved by exploring more sophisticated methods to obtain tighter polytopic approximations and by considering other types of nonlinearities introduced by multi-agent interactions.

3.1 Introduction

Control engineers are increasingly turning to machine learning techniques as systems become more complex and data becomes more abundant. The ease of generating simulated data for multiple system trajectories has made neural networks (NNs) a popular choice for modeling various aspects of control systems, including plants, actuators, controllers, and

even human operator behavior. Data-driven approaches have proven particularly useful for discovering underlying physical models of dynamical systems. These approaches involve using data to identify patterns and relationships between system inputs and outputs. Once a model has been identified, it can be used to design and optimize control algorithms. Overall, data-driven techniques hold great promise for advancing the field of control engineering, especially as the complexity of systems continues to increase. By leveraging large amounts of data to discover models that capture the underlying physics of these systems, engineers can design more effective control algorithms that are better suited to real-world scenarios.

Set-based properties like safety, reachability, and controllability offer a strong analytical foundation to quantify the performance of a system, especially under uncertainties. Reachability, in particular, is closely linked to other properties such as viability, controllability, and safety [3]. Estimating reachability is also useful for optimal control synthesis and high-level decision making [33]. While reachable sets can be computed analytically by solving Hamilton-Jacobi partial differential equations (PDEs), this approach is time-consuming and suffers from the "curse of dimensionality", making it challenging for real-time applications. Further, it requires finding suitable value functions or level sets to formulate the system at hand as a suitable Hamiltonian (see [9] for more details). To alleviate this, various numerical approximation techniques have been proposed to compute approximate reachable sets without explicit solutions to the associated HJ PDEs. These techniques employ polytopic approximations [34], Pontryagin's optimal control [35], numerical differential equation solvers [36], and numerical functional differential equations [37], to name a few. Based on these, a number of reachable set computation tools are available that utilize numerical techniques for approximate reachable sets and tubes (such as [29], [38], [39]), both forward and backwards in time.

Designing control signals and analyzing system properties under operational noise, parameter uncertainties, and nonlinearities are crucial aspects of control system design. Machine learning, particularly neural networks (NNs), has found applications in various stages of the control system design process, including system identification, output tracking, control synthesis, state estimation, and supervisory control logic design. Recent research has demonstrated the effectiveness of NNs in learning nonlinear dynamical models of varying

complexities. For example, in [40], iterative learning was used to infer the inverse dynamics of a robotic arm using real data from an iCub robotic arm. In [41], a learning-based approach was proposed to infer supervisory control logic for cybersecurity analysis of supervisory control systems. In [42], an iterative learning-based method was implemented to learn the dynamics of train wheel adhesion. In [43], control signals were synthesized to control a quadrotor by learning its dynamics. The utilization of machine learning techniques for system identification has been noted to be particularly useful in recent system modeling and identification texts such as [44], [45], and [46]. Overall, the ability of NNs to handle uncertainties and nonlinearities in control system design makes them a promising tool for a wide range of applications. Estimating reachable sets in real-time is vital, especially for safety-critical operations. However, despite the widespread usage of neural networks (NNs) in solving dynamics and control problems, their reliability under uncertainties and operating conditions that demand safety guarantees is limited due to the absence of reachable set computation/approximation tools for NN based models. However, since machine learning models use data from an unknown dynamical system, numerical approaches to compute approximate reachable sets can be extended to the learned models themselves. To this end, Koopman Operator theory provides a method to find a computationally scalable, equivalent linear lifted model for NNs [47]. This approach has been used to estimate reachable sets as polytopes by applying linear control methods to the linear lifted models. Additionally, an optimal control problem can be formulated to propagate these polytopes over time [48]. On the other hand, learning-based methods are also used to learn the Koopman operator itself for control synthesis. The accuracy of the proposed method depends on two key factors. Firstly, the ability of the NN to accurately approximate unknown plant models. Secondly, the ability of the Koopman operator to lift the NN model to a higher dimensional manifold and approximate it as a linear system. However, it should be noted that the proposed method can achieve arbitrarily accurate polytopic reachable set approximations [35]. Fortunately, both methods are proven to have an approximation accuracy that is asymptotic in the number of data points. As a result, the proposed methods allows for a real-time reachable set estimation framework for the learned model.

3.1.1 Related Works

The problem of computing output bounds of a neural network (NN) can be related to the problem of computing reachable sets for learned dynamical models. However, the computation of output bounds for NN-based controllers using specific activation functions is not easily amenable to real-time applications due to the complexity of the problem. Some approaches have been proposed in the literature, such as solving mixed-integer linear programs (MILPs) [49] or relaxed linear programs (LPs) [50], but these methods are computationally expensive and not suitable for real-time implementation. Methods that exploit individual perceptrons in a neural network (NN), connected through rectified linear unit (ReLu) activations, have been used to obtain output bounds. For example, in [51], Bernstein polynomials are used to obtain Taylor approximations of NN-based control systems to obtain reachability flow pipes. Exact reachable sets have been computed for a control system employing ReLu activation functions-based NNs with specific switched linear dynamics in [4]. However, these methods rely on explicit system dynamics or specific activation functions to obtain reachable sets for NN models. Most NNs used for learning system dynamics can be arbitrarily nonlinear. On the other hand, reachable set computation/approximation using HJ methods or polytopes is extant in controls literature [3], [34], [35]. Exact model-based methods depend on having complete knowledge of the system's dynamical modes, which is not the case for NN-based modeling. As far as we know, there are no techniques that use optimal control theory to extend local linear system approximations of NNs to obtain approximate reachable sets for these models.

3.1.2 Contributions

The proposed method has several main contributions that enhance the efficiency and effectiveness of NN-based reachability analysis. First, the method employs a dynamic mode decomposition (DMD) based framework to obtain approximate linear models for the given nonlinear dynamics learned by the NN. This allows for the use of optimal control theory to obtain polytopic approximations to the reachable sets for the approximately equivalent

linear system. The use of linear models reduces the complexity of the problem, making it more amenable to real-time applications.

Second, the proposed framework is numerically efficient and can be used in real-time applications. This is achieved by avoiding the need to solve mixed-integer linear programs (MILPs) or relaxed linear programs (LPs) at each time step. Instead, the proposed method uses the DMD-based framework and optimal control theory to obtain polytopic approximations of reachable sets in a computationally efficient manner.

Third, the framework is flexible and allows for the use of various reachability assessment tools for the approximately equivalent linear systems. The introduced polytopic reachable set approximation methods can be replaced by other reachability modules, as shown in Fig.3.1. This allows for the integration of other reachability assessment tools, providing flexibility and facilitating comparisons of different reachability analysis methods.

To demonstrate the effectiveness of the proposed method, the authors provide a detailed example of real-time reachable set approximation for a quadrotor model. The quadrotor model is a widely studied system for identification and control using NNs. The results of the example show the efficiency and effectiveness of the proposed method and its ability to provide accurate approximations of reachable sets in real-time applications.

The structure of this chapter is as follows. In Section 2, we define the reachability problem for a NN-learned model. In Section 3, we introduce our framework for estimating reachable sets for nonlinear models learned by NNs. We formulate the approximate reachable set computation as an optimal control problem to obtain polytopic approximations. To illustrate the effectiveness of our method, we use a quadrotor model learned from trajectory data using a NN as a running example. Section 4 provides a detailed implementation of our reachability estimation framework on the quadrotor example. We first consider a nominal quadrotor reachability case and then investigate a scenario where two of the rotors have failed. Finally, in Section 5, we present our concluding remarks. We emphasize the efficiency and accuracy of our proposed method, and highlight its potential for real-time applications.

Notations: For two vectors u and v, their inner product is denoted by $\langle u, v\rangle$. For a matrix A, we denote its transpose by A^T and its Frobenius norm by $\|A\| F$. For two sets A and B, we denote their Minkowski sum as $A \oplus B \triangleq \{a + b \,|\, a \in A, b \in B\}$. Similarly, we

denote their Minkowski difference by $A \ominus B \triangleq (A^c \oplus (-B))^c$. For a finite set A, if a random variable x is distributed uniformly in the set A, we write $x \sim \mathcal{U}_A$.

3.2 Problem Formulation

Consider the nonlinear dynamical system given as follows:

$$\dot{x}(t) = f\left(x(t), u(t)\right), \text{ and } x(0) \in \mathcal{X}_0, u(t) \in \Omega,\ t \geq 0 \tag{3.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the state, x_0 is the initial state in a known initial set $\mathcal{X}_0$, and the control input $u \in \mathbb{R}^{n_u}$ resides in the set Ω at all times t. The dynamical map f is considered to be unknown. Given some input-state data in the form of $X_k \triangleq \{x_0, \cdots, x_N\}$, $U_k \triangleq \{u_0, \cdots, u_N\}$ over multiple trajectories $k = 0, \cdots, n_T$, the unknown dynamical map f is learned using the trajectory data. The trajectory data is sampled from $\dot{x} = f(x, u)$ in (3.1) at some sampling rate Δt, such that $(x_{\mathrm{i}}, u_{\mathrm{i}}) \triangleq (x(\mathrm{i}\Delta t), u(\mathrm{i}\Delta t))$ for $\mathrm{i} = 1, \cdots, N$.

A data-driven method, such as a neural network (NN), is employed to estimate the unknown dynamics f from the time series data trajectories as:

$$\dot{\tilde{x}}(t) = \tilde{f}_\Theta\left(\tilde{x}(t), u(t)\right) \tag{3.2}$$

The state obtained by the neural network (NN) from the data X_k, U_k is denoted as $\tilde{x}(t)$, which serves as an approximation to the true state $x(t)$, as long as the learned dynamics $\tilde{f}\Theta$ is close to the true dynamics f. The approximate dynamics $\tilde{f}\Theta$ is characterized by the parameters Θ, which include the weights and biases of the NN. It is assumed, without loss of generality, that the initial state $\tilde{x}(0)$ belongs to the set of initial states $\mathcal{X}_0$, and the control input u belongs to the set Ω. The reachability problem for the NN model is then to find:

$$\mathcal{R}^{\tilde{f}}\left(\tau; \mathcal{X}_0\right) \triangleq \left\{\tilde{x}(\tau) \,|\, \dot{\tilde{x}} = \tilde{f}_\Theta\left(\tilde{x}, u\right), \tilde{x}(0) \in \mathcal{X}_0, u \in \Omega; \Theta\right\} \tag{3.3}$$

at some time τ, given initial conditions $\mathcal{X}_0$ and admissible control set Ω. Note that the problem to compute $\mathcal{R}^{\tilde{f}}(\tau; \mathcal{X}_0)$ is nontrivial due to the arbitrary, unstructured nonlinearities present in the NN modeling (parameterized by Θ).

Remark 4 (NNs as Universal Approximator [52])**.** *A causal NN with parameters Θ can be utilized to approximate f given time series trajectory data X_k, U_k. The state $\tilde{x}(t)$ obtained from the NN approximates the true state $x(t)$ as long as the approximation $\tilde{f}_\Theta$ is close to f. The parameters of the learning method, including the NN weights and biases, are included in Θ. Without loss of generality, it is assumed that the initial state $\tilde{x}(0)$ is in $\mathcal{X}_0$ and the control input u is in Ω. As the number of available trajectories n_T approaches infinity, it is expected that $\tilde{f}_\Theta$ approaches f, and therefore, $\tilde{x}(\tau)$ approaches $x(\tau)$ for $0 \leq \tau \leq N\Delta t$.*

The set $\mathcal{R}^{\tilde{f}}(\tau; \mathcal{X}_0)$ provides a good approximation of the reachable set for the nonlinear system in (3.1). Since we are not proposing a new learning method, our focus will be on $\mathcal{R}^{\tilde{f}}(\tau; \mathcal{X}_0)$. Going forward, *we will assume that the system's dynamics are described by $\tilde{f}_\Theta$, as the NN can be trained effectively according to Remark 1.*

3.3 Reachability Framework for Neural Network Models

In this section, we assume that the nonlinear dynamical system has been learned using NN techniques and focus on approximating reachable sets of the NN dynamics $\tilde{f}_\Theta$ using a computationally efficient method. To achieve this, we use a formulation of dynamic mode decomposition (DMD) that accommodates control inputs [53]. By doing so, we can construct a finite-dimensional approximation to the infinite-dimensional Koopman operator and obtain equivalent time-varying linear system models. The Koopman operator is a linear operator that maps a function of the state of a dynamical system to its evolution in time. In the case of a nonlinear dynamical system represented by a NN, the Koopman operator is typically infinite dimensional and difficult to compute. However, the DMD approach provides a way to construct a finite-dimensional approximation of the Koopman operator by leveraging input-output data from the system.

The basic idea of DMD is to perform a spectral decomposition of the data matrix containing the input-output pairs of the system, which leads to a set of modes that can be

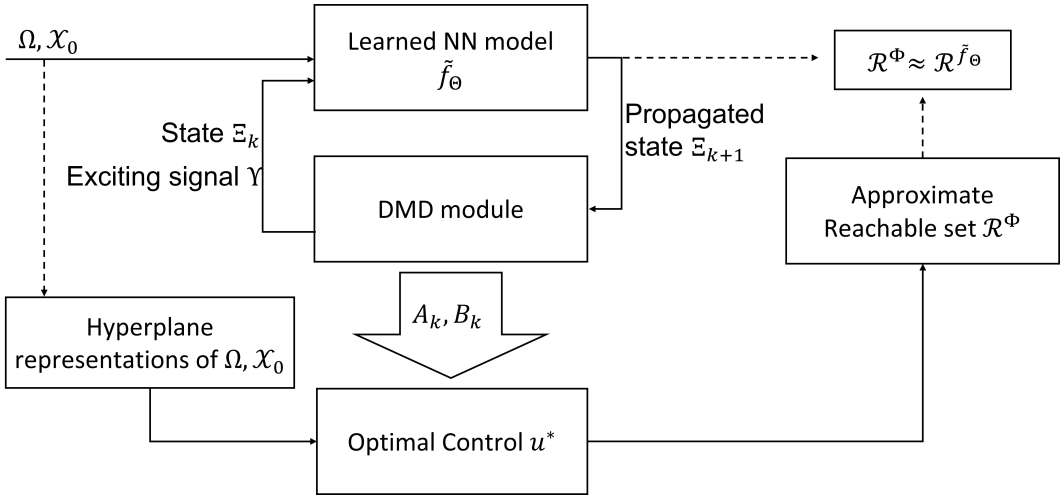

Figure 3.1. A schematic of the proposed data-driven framework for approximate reachable set computation

used to reconstruct the system behavior. By incorporating control inputs into the DMD formulation, we can obtain a more accurate approximation of the Koopman operator, which in turn allows us to obtain an equivalent linear time-varying system model. This linear system model can then be used to obtain polytopic approximations of the reachable sets using optimal control theory, as described in the following section.

3.3.1 Dynamic Mode Decomposition with Control

Let us look at a data-driven method for approximating the Koopman Operator, called dynamic mode decomposition. Nominal forms of DMD involve trajectory data (called 'snapshots') consisting of state evolutions over time x_k. The trajectory data get mapped under a linear operator as $x_{k+1} \approx Ax_k$. DMD with control (DMDc) was proposed to include input-state relations to such trajectory evolutions in [53]. Given an input-state data point x_k, u_k, DMDc attempts to find the pair of operators A, B such that $x_{k+1} \approx Ax_k + Bu_k$ for data

points on state $x_k \in \mathbb{R}^{n_x}$ and input $u_k \in \mathbb{R}^{n_u}$. The data matrices at time step k are temporal snapshots of the trajectory, of width $w \in \mathbb{N}$, given by:

$$\Xi_{k,w} \triangleq \begin{bmatrix} | & & | \\ x_k & \cdots & x_{k+w} \\ | & & | \end{bmatrix}, \Upsilon_{k,w} \triangleq \begin{bmatrix} | & & | \\ u_k & \cdots & u_{k+w} \\ | & & | \end{bmatrix} \tag{3.4}$$

The snapshot with data points propagated one-step in time, can then be represented as:

$$\Xi_{k+1,w} = \Gamma_{k,w} \begin{bmatrix} \Xi_{k,w} \\ \Upsilon_{k,w} \end{bmatrix}, \text{ where } \Gamma_{k,w} \triangleq \begin{bmatrix} A & B \end{bmatrix} \tag{3.5}$$

Note that the mapping Γ varies over time, and is parameterized by the snapshot width w. The DMDc solution to (3.5) can be viewed as a least-square regression problem to find a $\Gamma \in \mathbb{R}^{n_x \times (n_x + n_u)}$ such that:

$$\begin{aligned} \Gamma_{k,w} &= \arg\min \left\| \Xi_{k+1,w} - \Gamma_{k,w} \begin{bmatrix} \Xi_{k,w} \\ \Upsilon_{k,w} \end{bmatrix} \right\| F, \text{ or} \\ \Gamma_{k,w} &= \begin{bmatrix} A & B \end{bmatrix} = \Xi_{k+1,w} \begin{bmatrix} \Xi_{k,w} \\ \Upsilon_{k,w} \end{bmatrix}^{\dagger} \end{aligned} \tag{3.6}$$

where $[\,\cdot\,]^{\dagger}$ denotes the pseudo-inverse. Extracting columns from the least-square solution in (3.6), we get a linear time-varying system such that $x_{k+1} \approx A_k x_k + B_k u_k$.

One way to efficiently compute the least-squares solution is by utilizing singular-value decompositions of data snapshot matrices [53]. In real-time, matrices A_k and B_k can be estimated over time using a time-moving window of width w. In the proposed method, the moving window obtains snapshot data from the input-state relation learned by the NN. To obtain the state snapshot data, the learned model $\tilde{f}\Theta$ is excited by sampling an arbitrary control input from Ω that forms $\Gamma k, w$, and the output of the NN is noted into the propagated state data snapshot $\Xi_{k+1,w}$. This allows for an independent framework that uses excitations of the NN model to obtain approximate linear models, as depicted in Fig. 3.1. Using this

approach, the proposed framework can efficiently approximate the reachable sets for nonlinear models learned by a NN, which is much more amenable to real-time applications than solving MILPs or relaxed LPs at each time step.

3.3.2 Approximate Reachable Set Computation using DMDc Model

After obtaining linear approximations of the form $x_{k+1} = A_k x_k + B_k u_k$, the focus shifts to determining the reachable sets for the linear time-varying system $(A(t), B(t))$. The system matrices $A(t)$ and $B(t)$ satisfy the conditions $\exp A(t)\Delta t = A_k$ and $\int_0^{\Delta t} \exp A(s) s B(s) ds = B_k$ for $t \in [k\Delta t, (k+1)\Delta t)$. Thus, A_k and B_k are updated at each time step $k+1$ as new snapshot data is received through equation (3.5), and $A(t)$ and $B(t)$ are updated at time $(k+1)\Delta t$. The state-transition function $\Phi(t, 0)$ associated with the linear system $(A(t), B(t))$ is defined as $\dot{\Phi}(t, 0) = -A(t)\Phi(t)$, with $\Phi(0, 0) = I$. The reachable set of the system $(A(t), B(t))$ at time τ is denoted as $\mathcal{R}^{\Phi}(\tau; \mathcal{X}_0, \Omega)$. If the DMDc approximation is accurate, approximating $\mathcal{R}^{\Phi}(\tau; \mathcal{X}_0, \Omega)$ should suffice.

Without loss of generality, we assume that the admissible control set and the initial state set are polytopes as:

$$\begin{aligned} \mathcal{X}_0 &= \bigcap_{\mathrm{i}=1}^{n_1} \{v \in \mathbb{R}^{n_x} \mid \langle c_{\mathrm{i}}(0), v\rangle \leq \gamma_{\mathrm{i}}(0)\} \\ \Omega &= \bigcap_{\mathrm{i}=1}^{n_2} \{u \in \mathbb{R}^{n_u} \mid \langle d_{\mathrm{i}}, u\rangle \leq \varepsilon_{\mathrm{i}}\} \end{aligned} \tag{3.7}$$

defined for arbitrary vectors v. Additionally, the hyperplanes H_{i} define each face of the polytopes in (3.7) using normal vectors c_{i} and d_{i}. At points $x_{\mathrm{i}}^*(0)$ for $\mathrm{i} = 1, \cdots, n_1$, the hyperplanes H_{i} touch the set $\mathcal{X}_0$ with $\langle c_{\mathrm{i}}(0), x_{\mathrm{i}}^*(0)\rangle = \gamma_{\mathrm{i}}(0)$. Although this assumption is based on polytopes, it is not a limitation, as it is possible to bound arbitrarily convex, compact sets $\mathcal{X}_0$ and Ω with tight polytopes using (3.7). Clearly, the following hold true:

$$\begin{aligned} x_{\mathrm{i}}^*(0) &= \arg\max_{v \in \mathcal{X}_0} \{\langle c_{\mathrm{i}}(0), v\rangle\}, \text{ and} \\ \gamma_{\mathrm{i}}(0) &= \max_{v \in \mathcal{X}_0} \{\langle c_{\mathrm{i}}(0), v\rangle\} \end{aligned} \tag{3.8}$$

A polytopic approximation of the reachable set considers only the points of contact $x_{\mathrm{i}}^*(0)$ of reachable sets of the linear system, as described in [34], [35]. Likewise, at time τ, let $x_{\mathrm{i}}^*(\tau)$ denote the point of contact of $\mathcal{R}^{\Phi}(\tau; \mathcal{X}_0, \Omega)$ with the hyperplane $H_{\mathrm{i}}(\tau)$, which is defined as:

$$H_{\mathrm{i}}^*(\tau) = \{x \,|\, \langle c_{\mathrm{i}}(\tau), x_{\mathrm{i}}^*(\tau)\rangle = \gamma_{\mathrm{i}}(\tau)\} \tag{3.9}$$

for $\mathrm{i} = 1, \cdots, n_1$, at time $\tau > 0$. The observation above applies to compact and convex sets $\mathcal{X}_0$ and Ω. According to [35], these conditions ensure that the reachable set $\mathcal{R}^{\Phi}$ remains both compact and convex for all time τ. This, in turn, allows us to find hyperplanes $H_{\mathrm{i}}^*(\tau)$ to support the reachable set. Following an approach similar to that of (3.8), we can define the point of contact between $H_{\mathrm{i}}^*(\tau)$ and $\mathcal{R}^{\Phi}$ at any time τ as:

$$\begin{aligned} x_{\mathrm{i}}^*(0) &= \underset{v \in \mathcal{R}^{\Phi}(\tau; \mathcal{X}_0, \Omega)}{\arg\max} \langle c_{\mathrm{i}}(\tau), v\rangle \\ &= \arg\max \left\{ \langle c_{\mathrm{i}}(\tau), v\rangle \ \text{ s.t. } \ v(t) = \Phi(t)x(0) + \int_0^t \Phi(t,s)B(s)u(s)ds, u(t) \in \Omega, t \leq \tau \right\} \end{aligned} \tag{3.10}$$

Also, from (3.8), the distance between $H_{\mathrm{i}}^*(\tau)$ and $\mathcal{R}^{\Phi}$ can be expressed as:

$$\begin{aligned} \gamma_{\mathrm{i}}(\tau) &= \max_{v \in \mathcal{R}^{\Phi}(\tau; \mathcal{X}_0, \Omega)} \langle c_{\mathrm{i}}(\tau), v\rangle \\ &= \max \left\{ \langle c_{\mathrm{i}}(\tau), v\rangle \ \text{ s.t. } \ v(t) = \Phi(t)x(0) + \int_0^t \Phi(t,s)B(s)u(s)ds, u(t) \in \Omega, t \leq \tau \right\} \end{aligned} \tag{3.11}$$

For a given set of initial points of contact $x_{\mathrm{i}}^*(0)$, equations (3.10) and (3.11) depend only on the choice of $u \in \Omega$, thereby forming an optimal control problem.

Theorem 3.3.1. *Let the optimal control $u_{\mathrm{i}}^*(\tau)$ be the solution to $\arg\max_{u(\tau)} \langle c_{\mathrm{i}}(\tau), B(\tau)u(\tau)\rangle$. Then, for $\dot{c}_{\mathrm{i}} = -A(\tau)^T c_{\mathrm{i}}(\tau)$ with the initial condition $\gamma_{\mathrm{i}}^*(0)$, the reachable set $\mathcal{R}^{\Phi}$ is supported by the hyperplane $H_{\mathrm{i}}^*(\tau) \equiv \langle c_{\mathrm{i}}(\tau), x^*(\tau)\rangle$.*

Proof. The proof that follows is based on Pontryagin's maximum principle [34]. When we have the contact point $x_{\mathrm{i}}^*(\tau)$, it evolves as $\dot{x_{\mathrm{i}}^*} = A(t)x(\tau) + B(t)u_{\mathrm{i}}^*(\tau)$ for the given optimal control. For the linear system with matrices $A(t), B(t)$, the costate $\lambda(t)$ evolves as

$\dot{\lambda}(t) = A(t)^T\lambda(t)$. We can choose $c_\mathrm{i}(\tau)$ to be equal to the i-th component of the costate vector $\lambda(\tau)$, i.e., $c_\mathrm{i}(\tau) = \lambda_\mathrm{i}(\tau)$. The costate equations can be combined with the initial condition $\lambda_\mathrm{i}(0) = \gamma_\mathrm{i}^*(0)$ and we can suppress the time indices for brevity. This choice of $c_\mathrm{i}(\tau)$ ensures that $\langle c_\mathrm{i}(\tau), x(\tau)\rangle$ is a decreasing function of time. Therefore, the hyperplane $H_\mathrm{i}(\tau)$ defined by $\langle c_\mathrm{i}(\tau), x(\tau)\rangle = \gamma_\mathrm{i}(\tau)$ will touch the reachable set $\mathcal{R}^\Phi(\tau; \mathcal{X}_0, \Omega)$ at the point $x_\mathrm{i}^*(\tau)$ for which $\langle c_\mathrm{i}(\tau), x_\mathrm{i}^*(\tau)\rangle = \gamma_\mathrm{i}(\tau)$. As a result, the time derivative of $\langle \lambda_\mathrm{i}, x\rangle$ equals:

$$\begin{aligned}
&\frac{d}{d\tau}\langle \lambda_\mathrm{i}, x\rangle = \langle \dot{\lambda}_\mathrm{i}, x\rangle + \langle \lambda_\mathrm{i}, \dot{x}\rangle \\
&= \langle -A^T\lambda_\mathrm{i}, x\rangle + \langle \lambda_\mathrm{i}, Ax + Bu\rangle = \langle \lambda_\mathrm{i}, u\rangle \leq \langle \lambda_\mathrm{i}, u_\mathrm{i}^*\rangle \\
\Rightarrow &\frac{d}{d\tau}\langle \lambda_\mathrm{i}, x_\mathrm{i}^*\rangle = \frac{d}{d\tau}\gamma^*
\end{aligned}$$

Combined with the initial conditions on the points of contact, i.e., $\langle \lambda_\mathrm{i}(0), x_\mathrm{i}^*(0)\rangle = \gamma_\mathrm{i}^*(0)$, one gets $\langle \lambda_\mathrm{i}(\tau), x_\mathrm{i}(\tau)\rangle \leq \langle \lambda_\mathrm{i}(\tau), x_\mathrm{i}^*(\tau)\rangle = \gamma_\mathrm{i}^*(\tau)$. Hence, the hyperplane defined by c_i^* and x_i^* touches the reachable set. □

Remark 5. *From [35], the polytopic approximation can be made arbitrarily accurate. In fact, at time τ:*

$$\textit{convex hull}\{x_1^*, \cdots, x_{n_1}^*\} \subset \mathcal{R}^\Phi(\tau) \subset \cap_1^{n_1}\{\lambda_\mathrm{i}, x\} \leq \gamma_\mathrm{i}^*$$

that is, the convex hull of the supporting points provide an under-approximation of the reachable set. At the same time, the hyperplanes provide the over-approximation of the same.

The polytopic reachable set approximation is a widely used method to efficiently compute reachable sets and is utilized by many reachability computation tools. The τ-time reachable 'tube' resulting from this method can be defined as the Minkowski sum of reachable sets $\oplus_{s=0}^{\tau}\mathcal{R}^\Phi(s; \mathcal{X}_0, \Omega)$. Combining this with the DMDc-based method to obtain linear approximations of the NN model provides a scalable way to estimate reachable sets for NN models. It is important to note that the proposed numerical method relies on the universal approximation capabilities of NNs, as described in [52]. Moreover, DMDc (and DMD more generally) converges in operator norm to the Koopman operator with an increasing number

of data points [54], ensuring that the linear approximation of the NN model is more accurate as more data points are used. The proposed framework has the potential to achieve arbitrarily accurate reachable set approximations, given a sufficient number of data points and snapshot widths. This is a common feature of most data-driven approaches. However, in practice, the proposed framework offers a computationally efficient method to compute approximate reachable sets for learned models. The additional computational steps required only involve matrix inversions in DMDc and matrix exponentiation in propagating λ's, making the method relatively inexpensive compared to other numerical methods for reachable set approximation.

The process of mapping the data snapshots and discovering the mappings by the NN model and the DMDc method is depicted in Fig. 3.2. The learned model takes temporal trajectory data and attempts to map it to a higher-dimensional manifold known as the "feature space," which is parameterized by the NN parameters Θ. On the other hand, the DMDc method tries to approximate the infinite dimensional Koopman operator by finding finite-dimensional truncations of it. This is accomplished by considering the temporal trajectories in some “observable space" where the trajectory evolution can be approximated by the action of a linear operator [23], [53]. This approach makes the proposed framework suitable for real-time implementation, and in the subsequent section, we will present it using a quadrotor as an example. It is worth noting that this framework can achieve arbitrarily accurate reachable set approximations if there are enough data points and snapshot widths. However, the proposed method is computationally efficient and only requires additional computation steps involving matrix inversions in DMDc and matrix exponentiation in propagating λ's.

Remark 6. *Once the model is learned, computing the reachable set requires matrix exponentiation and matrix-vector multiplications, which have a computational expense of* $\mathcal{O}([\cdot]^{-1})$ *and* $\mathcal{O}(\exp[\,\cdot\,])$*, respectively. These operations can be efficiently performed using existing linear algebra libraries, such as PyLops [55] and Armadillo [56], making the overall computational expense relatively inexpensive.*

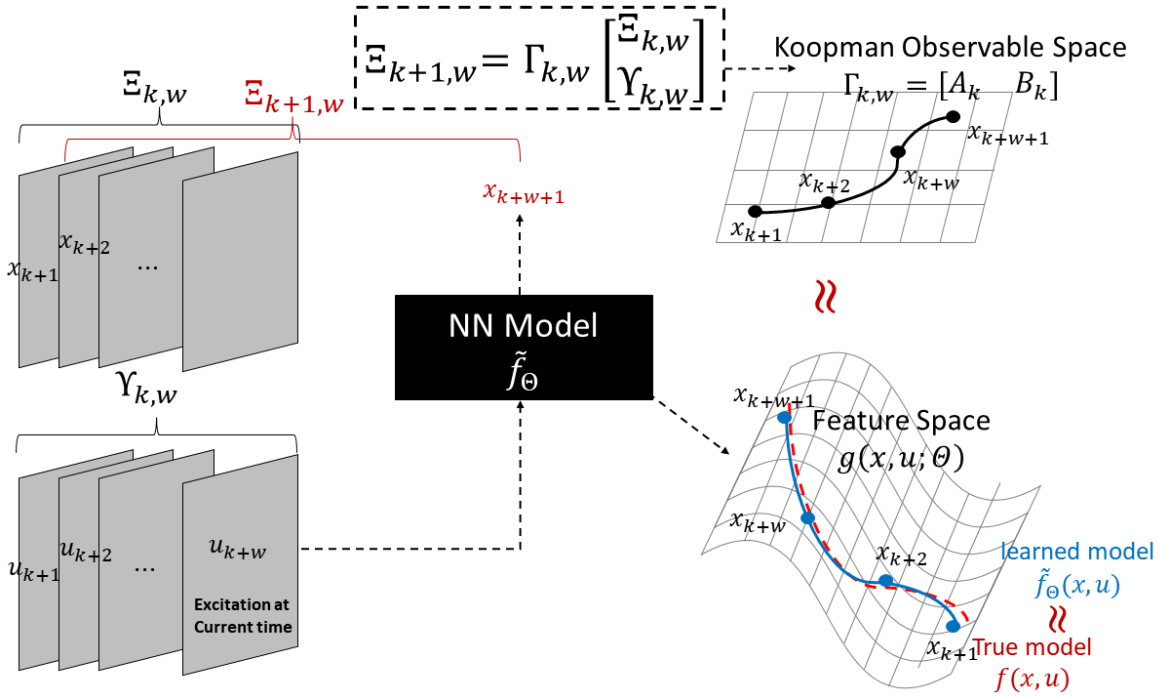

Figure 3.2. Schematic comparison of the temporal data snapshots, as learned by the NN vs. the DMDc approximation

3.4 Reachable Sets for a Quadrotor

To capture the wide range of dynamics of a quadrotor, a fully nonlinear model is required, although locally linear models are often used for control synthesis [57]. Therefore, in this study, we utilize a fully nonlinear 12 degree-of-freedom (DOF) quadrotor model based on [58] to demonstrate the proposed framework. Specifically, in this section, we focus on the computation of reachable sets over time for the 12DOF nonlinear dynamics, given a set of initial states $\mathcal{X}_0$ and control inputs Ω. This problem is of utmost importance in safety-critical applications, where we need to ensure that the quadrotor remains within a specified set of safe states during its operation.

The state vector ξ can be represented as $\xi = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \psi, \theta, \dot{\phi}, \dot{\psi}, \dot{\theta}]^T$, where the first three components denote the 3D position and the next three components denote the corresponding velocities. The remaining components represent the 3D angular attitude and

the respective angular velocities. Thus, the state vector ξ is a 12-dimensional vector, i.e., $\xi \in \mathbb{R}^{12}$. The 12-DOF state ξ evolves as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{u_1}{m}\left(\sin\phi\cos\psi + \cos\phi\cos\psi\sin\theta\right) \\ -\frac{u_1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\psi\right) \\ g - \frac{u_1}{m}\left(\cos\phi\cos\theta\right) \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} I_{xx}\ddot{\phi} \\ I_{yy}\ddot{\theta} \\ I_{zz}\ddot{\psi} \end{bmatrix} = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} - \begin{bmatrix} (I_{zz} - I_{yy})\dot{\theta}\dot{\psi} \\ (I_{xx} - I_{zz})\dot{\phi}\dot{\psi} \\ (I_{yy} - I_{xx})\dot{\theta}\dot{\phi} \end{bmatrix} \tag{3.13}$$

where I_{xx}, I_{yy}, I_{zz} are the moments of inertia along the 3 axes, g is the acceleration due to gravity, and m is the quadrotor's mass. Variables $u_1, \cdots, u_4$ relate to the actual angular velocity command at the four rotors $\omega_1, \ldots, \omega_4$ as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ -lk_f & lk_f & lk_f & -lk_f \\ lk_f & lk_f & -lk_f & -lk_f \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.14}$$

via the aerodynamic force and moment constants k_f and k_m, respectively, and the distance of rotors from the center l (see [58] for the derivation details). Experimental system identification is required to determine the force and moment constants k_f and k_m, making the task of learning the nonlinear dynamics in (3.12) through (3.14) challenging. Due to the complexity of the system, several works have explored learning the nonlinear dynamics for a quadrotor to aid in control synthesis.

We assume that we have access to n_T input-state trajectories X_k, U_k, where each trajectory starts with a randomly initialized state ξ_0 drawn from a uniform distribution $\mathcal{U}\mathcal{X}_0$. In order to learn the approximate dynamics $\tilde{f}\Theta$ from the time series input-state trajectories, we employ a causal multistep neural network (NN) based on [59]. The multistep NN is capable of recovering the nonlinear dynamics by using time series trajectories over a certain number of steps, say $\xi_k, \xi_{k-1}, \ldots, \xi_{k-m}$ and $\omega_{\mathrm{i},k}, \cdots, \omega_{\mathrm{i},k-m}$ for $\mathrm{i} = 1, \cdots, 4$, and finding appropriate

weights for a nonlinear function that maps the m-step trajectory to ξ_{k+1} (see [59] for more details). The weights Θ of the multistep NN are determined by minimizing the mean-squared error over each m-step slice of the trajectory data, for a fixed m. It is important to note that the actual functional approximation is performed by the multistep NN, and not by the DMDc method, which is used to obtain linear approximations of the learned model.

We utilized a multistep neural network, which is a type of multi-layer perceptron NN designed for system identification of differential equation based dynamical systems, as described in [59]. The goal of this NN is to approximate the system dynamics $\dot{x} = f(x)$ using trajectory data $x_k k = 0^N$ by unrolling a trajectory of length N and estimating the dynamical map $\tilde{f}\Theta : x_k \to x_{k+1}$ for $0 \leq k \leq N-1$. Our multistep NN consisted of 3 layers, including one hidden layer, with 12, 256, and 12 neurons, respectively. We trained the multistep NN using the Adams-Moulton scheme, which uses the trapezoidal rule to extend the function between k and $k+1$. The training was performed over 100 trajectories with a discretization time of $\Delta T = 0.1$s, a hyperbolic tangent activation function ($\tanh(\cdot)$), and mean square error (MSE) loss function with adaptive moment estimation (ADAM) as the optimizer. The training process took approximately 80 seconds using Python on an Intel Xeon CPU running at 2.20 GHz with a 13 GB memory and a 56 MB cache size. The multistep NN was trained over 2,000 epochs and converged to an MSE loss of 1.26×10^{-3}.

3.4.1 Reachable Set Computation

Fig. 3.3 illustrates an example of the reachable set computation problem for the quadrotor model described by equations (3.12)-(3.14). The computation starts with a randomly selected initial state within the given initial set (shown in green), and the set of all possible evolutions of the quadrotor's state after a specified time τ is represented by the set $\mathcal{R}^{\tilde{f}}(\tau; \mathcal{X}_0, \Omega, \Theta)$ (shown in yellow). It should be noted that the command input to each rotor is subject to additive noise, which is modeled as follows:

$$\omega_{\mathrm{i}}(t) = v_{\mathrm{i}}(t) + w_{\mathrm{i}}, w_{\mathrm{i}}(t) \sim \mathcal{U}_{[-0.25,0.25]} \text{ for } \mathrm{i} = 1, \cdots, 4 \tag{3.15}$$

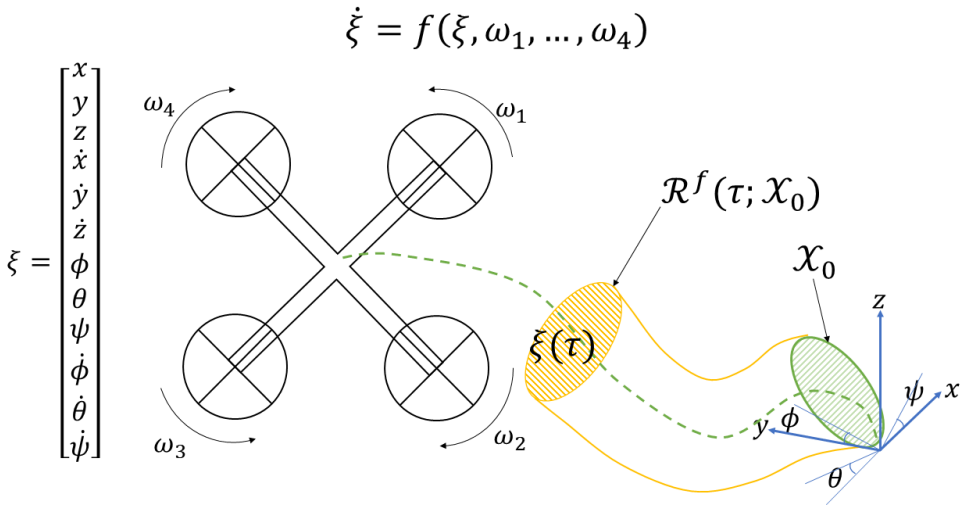

Figure 3.3. Computing the reachable set for a quadrotor involves determining the possible states ξ at a given time τ, along a trajectory originating from a given initial set $\mathcal{X}_0$.

and the admissible control set is defined as:

$$\Omega \triangleq \{\omega_1, \cdots, \omega_4 \,|\, \omega_{\mathrm{i}}(t) = v_{\mathrm{i}}(t) + w_{\mathrm{i}}, \forall \mathrm{i}\} \tag{3.16}$$

where the additive noise w_{i} is assumed to be independent, identically distributed at all times t. The reachable set approximation problem can be modeled by incorporating actuator noise using equations (3.15) and (3.16). Specifically, given a set of initial states $\mathcal{X}_0$ and a set of admissible control inputs Ω, the set of all possible states that can be reached in time τ under the noisy rotor command input set Ω is contained within $\mathcal{R}^{\tilde{f}}(\tau; \mathcal{X}_0, \Omega, \Theta)$, where $\tilde{f}$ is the learned approximate dynamics function.

We started by training the nonlinear model $\tilde{f}\Theta$ on $n_T = 100$ trajectories, each starting from a random position with x, y, z coordinates sampled from $\mathcal{U}[-0.5, 0.5]$ (in meters), and a random pose with ϕ, ψ, θ angles sampled from $\mathcal{U}_{[-0.1,0.1]}$ (in radians). This random initialization defines the initial set $\mathcal{X}_0$ and also provides explicit forms of hyperplanes H_{i} at time $\tau = 0$. To generate the training and testing datasets, we randomly generated trajectories

starting from $\mathcal{X}_0$ and applied control sequences sampled from Ω to each of the four rotors, as shown in Figure 3.5.

We utilized the DMDc-based framework described in Section 3 to develop approximate linear time-varying models (A_k, B_k) for $k = \mathrm{i}\Delta t$, where $\Delta t = 0.1$s and k takes values from 0 to 50. Based on a dataset with only 100 trajectories, the multistep NN was able to accurately reconstruct the true trajectories, as shown in Fig.3.4. Furthermore, despite the nonlinearities in the learned model $\tilde{f}_\Theta$, the DMD reconstructions were also accurate with a relatively small width of $n_w = 8$. Fig.**??** provides a comparison of the pose state reconstruction for the quadrotor. The initial set $\mathcal{X}_0$ was randomly generated using a uniform distribution, with the x, y, z coordinates within $[-0.5, 0.5]$ m and ϕ, ψ, θ within $[-0.1, 0.1]$ radians. To generate the training and testing datasets, trajectories were randomly sampled from $\mathcal{X}_0$ and control sequences were randomly sampled from Ω, as depicted in Fig. 3.5.

Next, we represent the initial set $\mathcal{X}_0$ using $n_1 = 24$ hyperplanes and the admissible control set Ω using $n_2 = 8$ hyperplanes. Each hyperplane supports $\mathcal{X}_0$ at a contact point ξ_{i}^*, and the τ-time reachability problem becomes the τ-time optimal control problem using the lifted system A_k, B_k. That is, at a time $t = k\Delta t$, the lifted model A_k, B_k is used to solve the optimal control problem in (3.10) and (3.11), propagating $\xi_{\mathrm{i}}^*(k\Delta t)$ under the optimal control input $\omega_{\mathrm{i}}^*(t) = \omega_{\mathrm{i}}^*$ for $k\Delta t \leq t < k\Delta t + \tau$. By propagating each contact point from time t to $t + \tau$, we obtain the supporting structure for the reachable set over time τ. As the amount of data increases, the multistep NN model approaches the true unknown dynamical map, and the DMD lifted system approaches the learned model. This accuracy in approximation is validated in Figs. 3.5 and 3.6.

The green envelope shown in Figs. 3.5 and 3.6 represents the admissible control set Ω, which consists of sinusoidal angular velocity inputs $v_{\mathrm{i}}(t)$ applied to the four rotors, with added noise. The optimal control input, represented by the solid green plots to the right in the figures, is used to propagate an arbitrary hyperplane's contact point ξ_{i}^* over time by applying rotor control $\omega_1^*, \cdots, \omega^*4$. The red points depict the contact points of the hyperplanes at each time step $0, \Delta t, 2\Delta t, \cdots$ for a simulation duration of 5 seconds. The reachable tubes' inner approximations are simply Minkowski sums of the convex hulls in red.

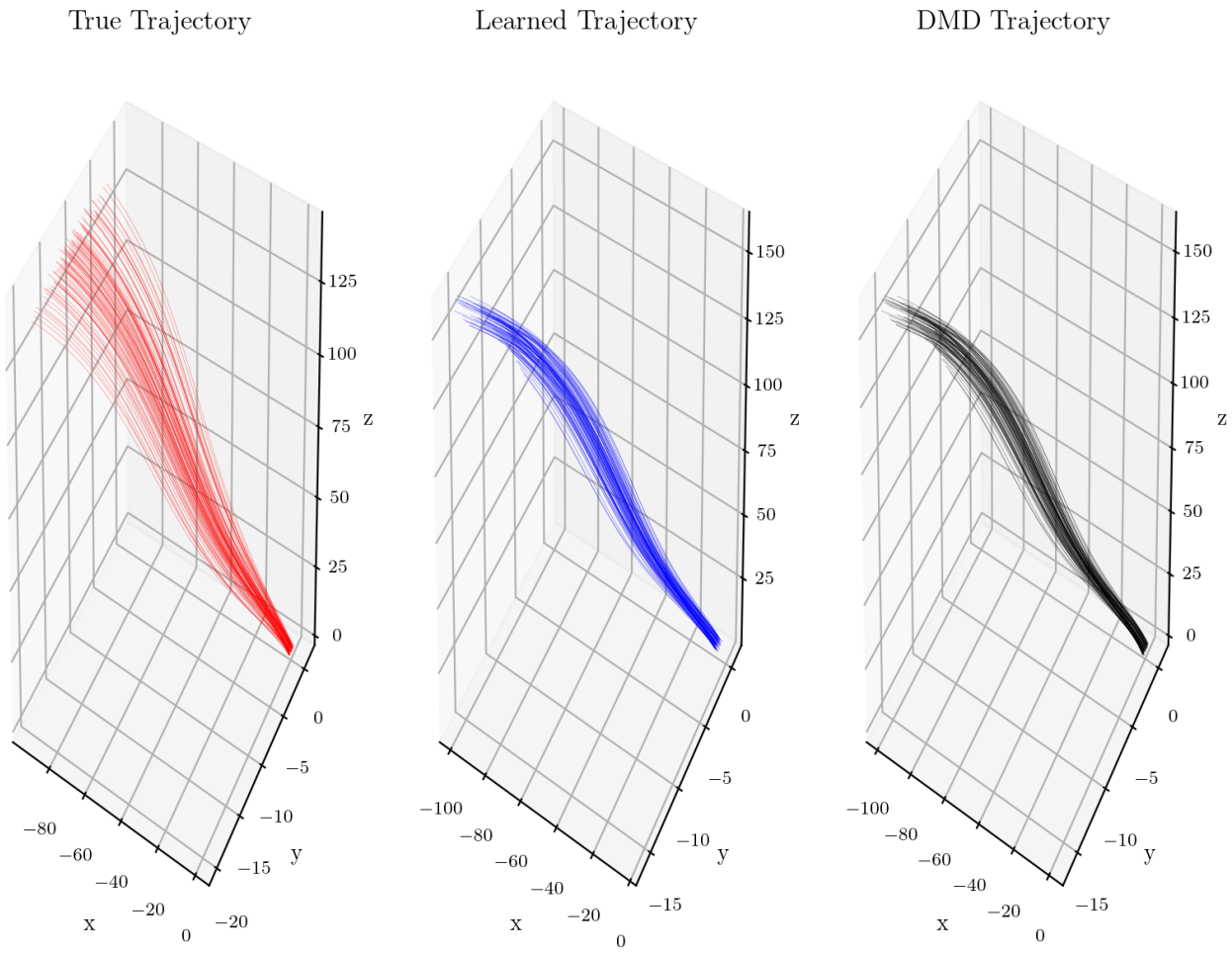

Figure 3.4. The true 3D state trajectories are depicted in red, while the trajectories reconstructed by the multistep NN and DMDc are shown in blue and black, respectively.

Using the multistep NN, an arbitrary trajectory starting from an $\xi(0)$ sampled from $\mathcal{U}\mathcal{X}_0$ is reconstructed and shown as a solid black line.

Despite the reasonable accuracy of the trained multistep NN shown in Fig. 3.4, the DMDc reconstruction technique is able to find accurate linear system models, which allows for real-time estimation of approximate reachable sets. As mentioned in Section 3.1.1, there are currently no existing methods that extend optimal control theory to obtain approximate reachable sets for NN models. The closest methods require exact, detailed knowledge of the NN architecture and utilize LPs [50] and MILPs [49] to obtain reachable sets or out-

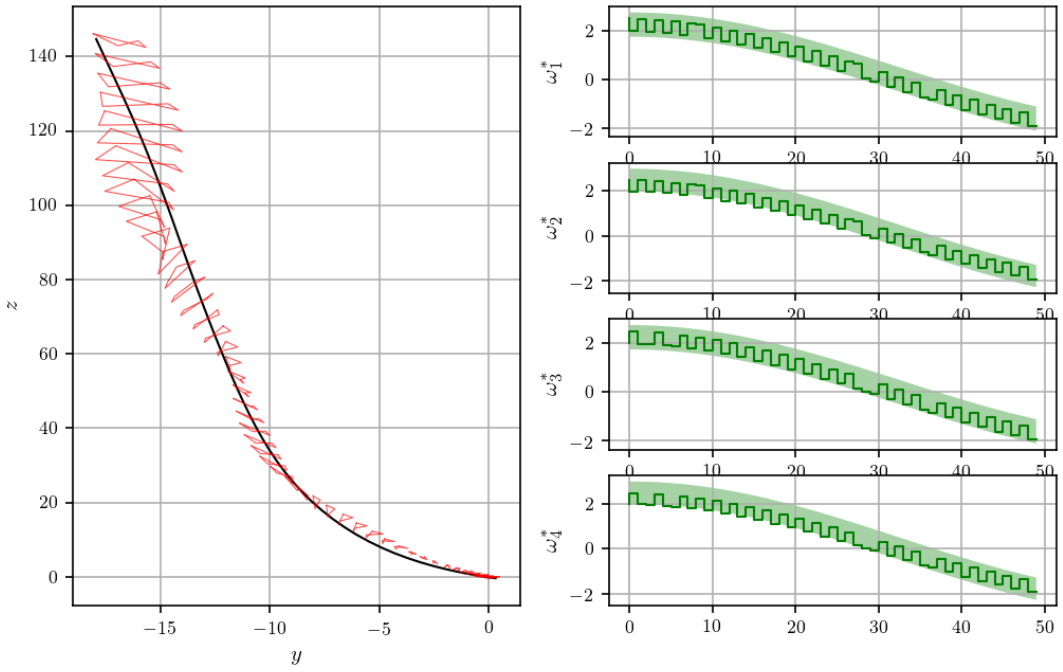

Figure 3.5. The inner approximations of reachable sets in the $y - z$ plane obtained by computing the convex hulls of the points ξ_i^*

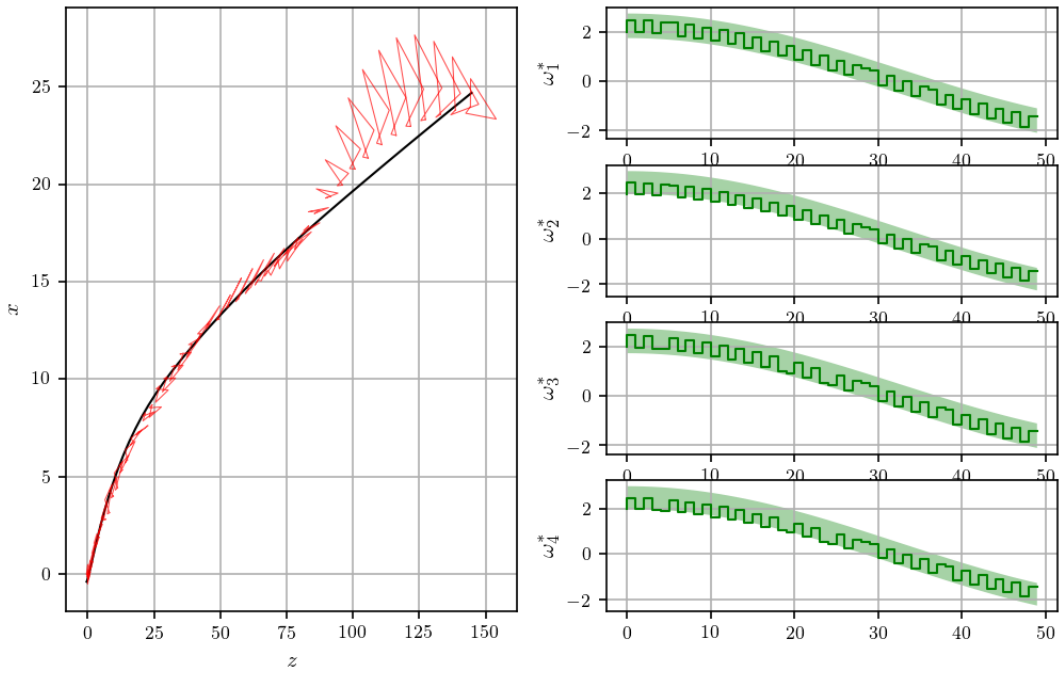

Figure 3.6. The inner approximations of reachable sets in the $z - x$ plane obtained by computing the convex hulls of the points ξ_i^*

put bounds. Since the proposed method treats the NN as a black box and does not require knowledge of its internal architecture, a direct comparison with these methods is not straightforward. However, one can compare the computational costs of the different methods, as shown in the table below.

Table 3.1. Comparing LP, MILP, and the proposed methods

Method	NN model	most expensive operation	Computational Cost w.r.t.	
			#layers	#variables
LP method [50]	required	solve LP	$\mathcal{O}(L)$	$\mathcal{O}(n^{\alpha})$
MILP method [49]	required	solve MILP	$\mathcal{O}(L)$	exponential
Proposed method	X	$\exp[\cdot]$	$\mathcal{O}(1)$	$\mathcal{O}(n^3)$

It is important to note that each layer L in the neural network (NN) architecture introduces approximately $\mathcal{O}(Ln)$ variables for the LP and MILP-based methods. While the MILP problems are known to be NP-hard, the method proposed in [49] has a worst-case computational complexity similar to a brute force search, which is exponential. On the other hand, the LP-based method has a computational complexity of approximately $\mathcal{O}[L(4nL)^{2.5}]$, which corresponds to the cost of solving one LP for each layer. In contrast, the computational costs associated with the proposed method are solely based on the expense of the matrix exponential operation, which is of the order $\mathcal{O}[(n+n_h)^3]$ for n_h hyperplanes. While the LP-based method is computationally cheaper, it only provides hyperrectangular approximations to the reachable sets, which are loose overapproximations. As such, it is more similar to interval reachable set methods such as [2], [60]. Additionally, introducing MILP and LP encodings incurs additional computational overhead. Finally, because the propagation of contact points for each hyperplane can be performed independently, the proposed method is better suited for parallelization than the MILP and LP formulations, which have inter-layer variable dependencies.

3.4.2 Reachable set Computation under Rotor Failure

In order to evaluate the capabilities of the system under compromised conditions, we estimate the reachable sets for a set of trajectories subject to rotor failures. This is an

important step in assessing the performance of a system when faced with actuator failures. In this particular scenario, we simulate total failures of rotors 2 and 3, which results in them appearing only as noise in the input channel. As a consequence, this significantly modifies the admissible control set Ω that can be applied to the system. By computing the reachable sets under such conditions, we can evaluate the system's ability to operate effectively and safely in the presence of such failures. As a result of the rotor failures, the quadrotor's

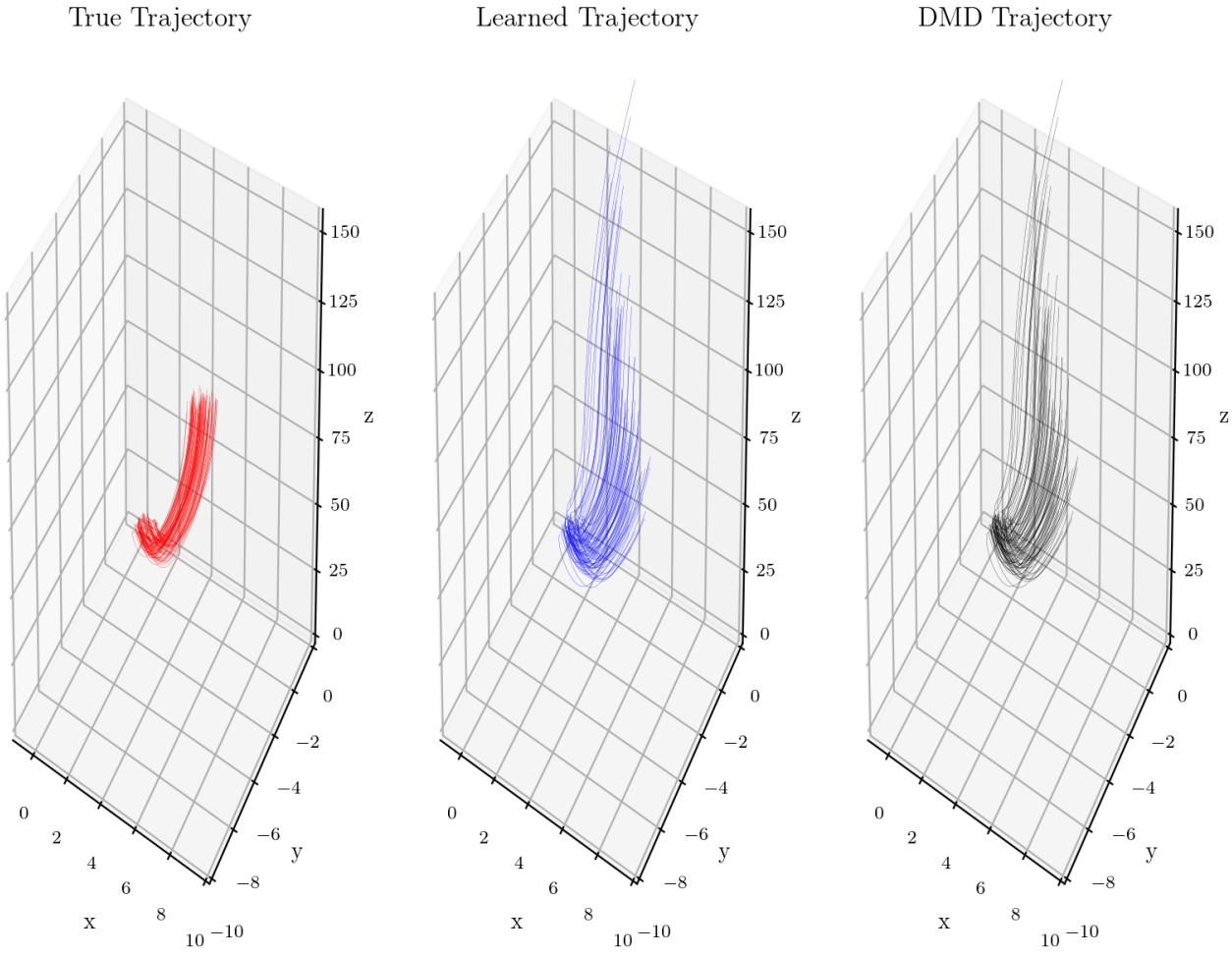

Figure 3.7. The 3D state trajectories under rotor failure at rotors 2 and 3 are represented by the true trajectories in red, while the trajectories reconstructed by the multistep NN and DMDc are shown in blue and black, respectively.

behavior deviates significantly from the expected trajectory, leading to discrepancies between the true trajectory and the NN model's predictions, as depicted in Fig. 3.7. The plot also illustrates that the DMDc model's performance is comparable to that of the NN model,

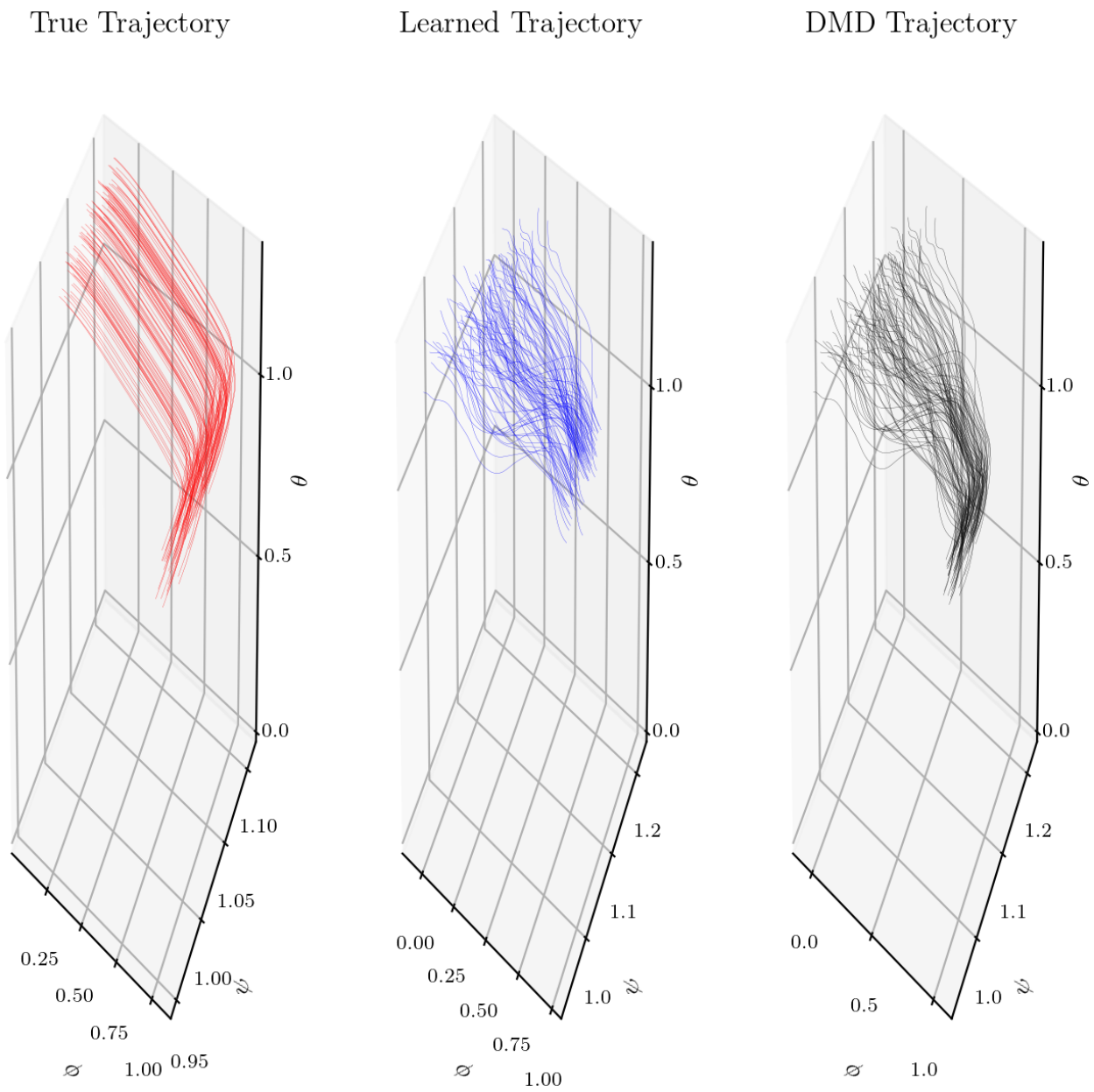

Figure 3.8. The 3D attitude trajectories under rotor failure at rotors 2 and 3 are depicted by the true trajectories in red. The trajectories reconstructed by the multistep NN and DMDc are represented in blue and black, respectively.

despite the disturbances caused by the actuator failures. This is further emphasized in Fig. 3.8, which shows the ϕ, ψ, θ trajectories. It should be noted that while DMDc is capable of reconstructing trajectories by exciting the learned model, both the NN model and the DMDc model's performance is poor compared to the true angular pose trajectories. However, it is evident that the DMDc model is still performing relatively well compared to the learned model.

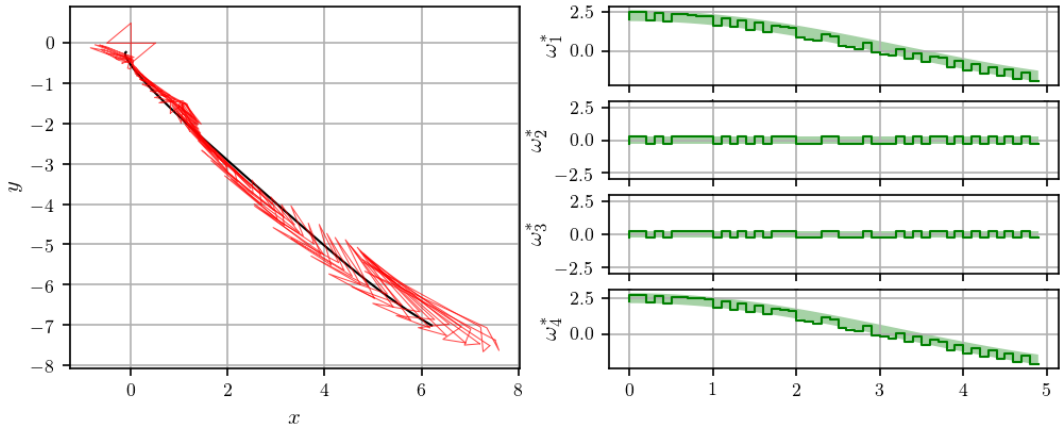

Figure 3.9. When there is rotor failure at rotors 2 and 3, the inner approximations of reachable sets in the $x - y$ plane are obtained by computing the convex hulls of the points ξ_i^*

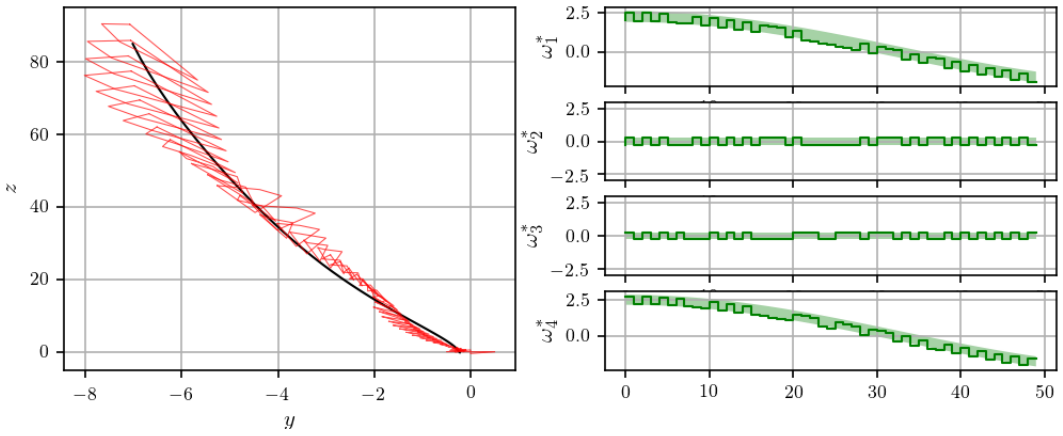

Figure 3.10. When there is rotor failure at rotors 2 and 3, the inner approximations of reachable sets in the $y - z$ plane are obtained by computing the convex hulls of the points ξ_i^*

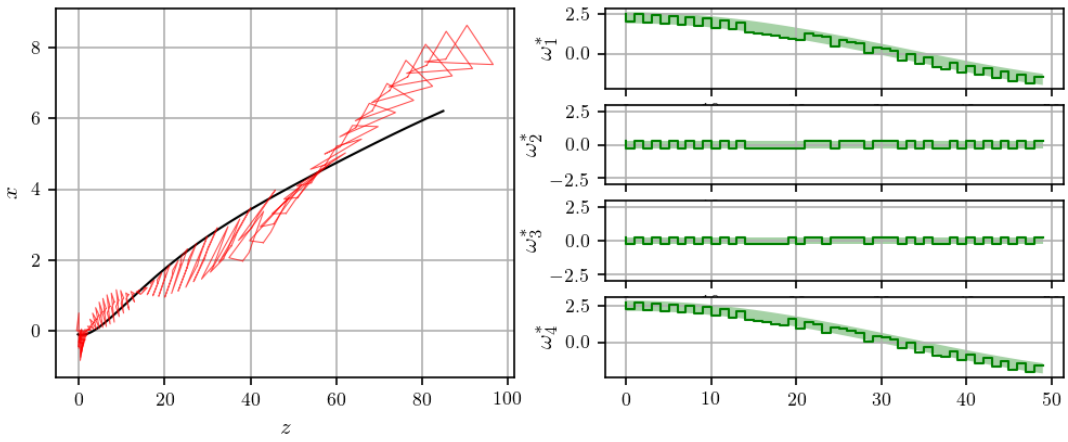

Figure 3.11. When there is rotor failure at rotors 2 and 3, the inner approximations of reachable sets in the $z - x$ plane are obtained by computing the convex hulls of the points ξ_i^*

The failure of rotors 2 and 3 has a significant impact on the quadrotor's reachable sets, resulting in completely different admissible control sets Ω. The admissible control sets are visualized as green envelopes in Figs. 3.9, 3.10, and 3.11. As shown in the figures, the admissible control sets are greatly constrained due to ω_2 and ω_3 being bounded noise, while the remaining rotors provide the nominal angular velocity command. This constraint is also reflected in the optimal control for each point of contact on an arbitrary hyperplane, which is shown as a solid green line in the figures. The convex hulls of the points of contact represent inner approximations of the reachable sets. The impact of the rotor failure is reflected in the significant change in the reachable sets and the constrained admissible control sets.

The proposed method shows promising results even in scenarios with highly nonlinear models and rotor failures, using relatively small data sets (n_T and n_w). This allows for accurate computation of reachable sets with a computational increment of less than 0.5 seconds per step. As a result, the proposed method is suitable for real-time approximations of reachable sets. These findings demonstrate the effectiveness and efficiency of the proposed method, which can be used for assessing the system's capabilities under various failure scenarios and inform control decisions. Therefore, the proposed method provides a practical and computationally efficient approach for computing reachable sets in real-time.

3.5 Conclusion

In this chapter, a new approach for computing approximate reachable sets for nonlinear models learned using neural networks was presented. The proposed method is data-driven and computationally efficient. It uses a lifting-based technique to find linear approximations of the learned model by exciting the neural network at each time step. The framework was shown to be suitable for use in conjunction with other reachability tool sets and can be made more accurate by incorporating additional data. The real-time application of the proposed framework was demonstrated through a quadrotor example. The framework was used to compute approximate reachable sets for a causal neural network that learned the quadrotor's dynamics. In addition, the framework was used to compute modified reachable sets for the quadrotor in a scenario that modeled rotor failures. The results showed that the proposed method is particularly useful in safety-critical scenarios where real-time approximation and update of reachable sets are necessary. The proposed data-driven framework offers a computationally cheap and accurate method for computing approximate reachable sets for nonlinear models learned using neural networks. It has the potential to be widely used in various fields, including robotics, control systems, and safety-critical applications.

Our next step is to explore the space complexity of our framework and analyze the amount of data required to achieve a certain level of accuracy in computing reachable sets. Furthermore, we intend to investigate the space complexity necessary to ensure a desired level of robustness against parameter variation. Additionally, we aim to develop efficient codes to create a wrapper that allows easy integration with existing reachability toolboxes.

4. REACHABLE SETS FOR MULTI-AGENT SYSTEMS

The research presented in this chapter is conducted by **Thapliyal, O.** under the supervision of Hwang, I.

Abstract

In this paper, we consider the problem of distributed reachable set computation for multi-agent systems (MASs) interacting over an undirected, stationary graph. A full state-feedback control input for such MASs depends no only on the current agent's state, but also of its neighbors. However, in most MAS applications, the dynamics are obscured by individual agents. This makes reachable set computation, in a fully distributed manner, a challenging problem. We utilize the ideas of polytopic reachable set approximation and generalize it to a MAS setup. Our idea of distributed polytopic reachable set has three main sub-problems: (a) solving for optimal control, (b) co-state propagation for the optimal control problem, and (c) state propagation. We formulate the resulting sub-problems in a fully distributed manner and provide convergence guarantees for the associated computations. Finally, we demonstrate the efficacy of our method in a multi-agent formation flight setting, and compare our proposed method to a fully centralized method.

4.1 Introduction

Multi-agent systems (MASs) appear in widely different fields, ranging from social networks, biological systems, computer networks, smart grid, to multi-robot systems [61]. Such systems often consist of cheaper or smaller components with limited capabilities, to perform complex tasks that an individual agent could not perform otherwise – such as formation control, target tracking and capture, distributed computing, and distributed learning tasks [62]. Multi-robot systems are employed to enhance capabilities of individual units by virtue of the additional redundancy, resilience, robustness against agent failure, and modular task assignment. As a result, the emergent behavior of multi-agent systems can result in behaviors far more complex than the comprising individual agents. Due to the above benefits of

MASs, such systems find ubiquitous applications, often in safety-critical environments [63]. Therefore, the ability to compute safety properties on-line for MASs is highly desirable.

Of the various set-properties that characterize safety, the ability to compute reachability properties for dynamical systems is an important task for control synthesis, validation of control protocols, and online safety checking. The reachability property is desirable to be computed for systems that are required to operate in safety-critical applications. For the more elaborate dynamical systems with dedicated computational resources, a centralized reachability problem is limited only by the accuracy of the dynamical model. Further, multi-agent systems (MAS) are characterized by the following key differences from centralized systems. However, MAS dynamics have to be flexible enough to allow for transient collaboration, followed by periods of non-cooperative, 'selfish' dynamics [64]–[66]. Additionally, MASs are usually cheaper components in a bigger network of agents, internet-of-things network, or a system-of-systems, with limited computational capabilities. The limited computational capability at each agent is the most severe bottleneck in computing properties of an MAS in a distributed manner. Despite the above limitation, a safe operational requirements for individual components of MAS is desirable. This is particularly instantiated in the Urban Air Mobility (UAM) scenarios, where individual components can be operated by different entities [67]. [67]–[70]. Individual components in the UAM corridors (i.e., unmanned aerial vehicles (UAVs)) can enter and leave the corridors flexibly, and cooperate in safety-critical environments while operating within the UAM constraints. Further, the sensed and communicated information from within the neighborhood of any agent can affect its own reachable sets (therefore, its own safety property) in non-trivial ways (see Fig. 4.1). The distributed reachable set computation problem is further exacerbated by the individual vehicular dynamics being obscured across the MAS network of interacting UAVs. The distributed reachable set computation for MASs attempts to serve the above requirements, subject to limited communication and computational resources on each component.

Reachability problems for more elaborate dynamical systems have been dealt with in literature in a centralized manner [2], [6], [9], [34]–[36], [43], [71]–[73]. Computing exact reachable sets often requires solving Hamilton Jacobi (HJ) equations [9], [43], or finding level set functions [34] to represent set boundaries. On the other hand, authors in [2],

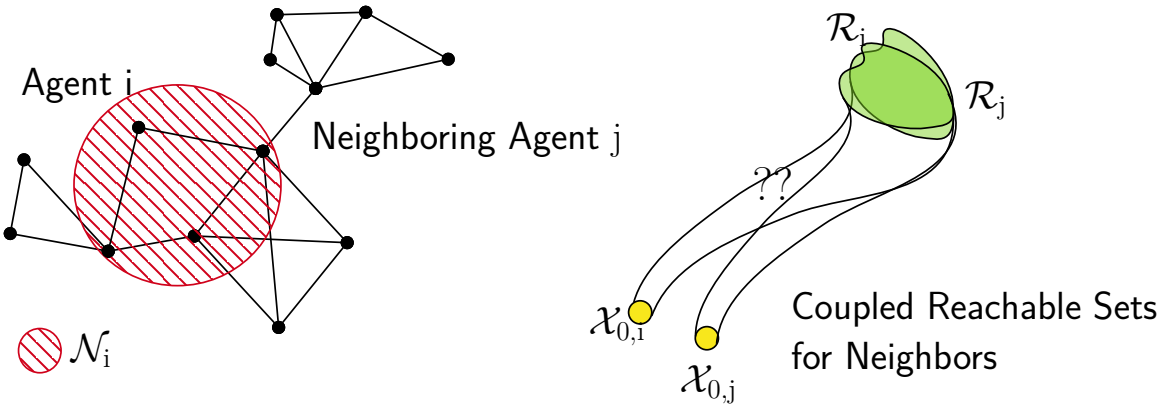

Figure 4.1. Dynamical coupling of reachable sets in an MAS

[6], [35] provide centralized reachable set approximation methods by bounding the exact reachable sets with ellipses, polytopes, and intervals, respectively. However, a centralized reachability framework does not take into account limited computational capabilities at each agent. Neither does a centralized reachability framework address the limited information sharing structure of decentralized MASs. A few works in the literature consider multi-agent applications of reachability [74]–[78]. Authors in [74] utilize centralized definitions of reachability applied to certain cases of multi-agent networks. The actual computation of the reachable sets does not require communication of information, or is dependent on the states of neighboring agents. In [76] a multi-agent reinforcement learning (RL) framework is considered where the overall safety requirements are written in a formal mixed-integer linear programming (MILP) notation. However, a central MILP is solved to compute reachable sets for the multi-agent RL framework. Methods proposed in [75] utilize Hamilton Jacobi reachability, therefore, suffer from the well known 'curse of dimensionality'. Authors in [77], [78] consider similar centralized variations of the reachable set computation problem. However, none of the existing methods address the limited computational abilities, or the couple dynamical models of individual agents. Further, the manner in which agents cooperate is not mission flexible, and little allowance is given to graph connectivity (except [78]).

To this end, we attempt to extend the ideas of approximate reachable set computation, and implement a fully distributed algorithm that incorporates the inter-agent dynamical cou-

pling behavior. The proposed algorithm has convergence guarantees for static inter-agent networks that are strongly connected. Our main contributions are: (i) posing reachable set computation for MASs as a distributed problem, (ii) proposing a fully distributed algorithm to compute tight approximations to the reachable sets, and (iii) providing convergence guarantees for the proposed algorithm. The resulting algorithm is computationally inexpensive, does not require solving MILPs or HJ equations, and the required computations can be easily computed locally by individual agents. The proposed method is also promising for time-varying networks for uniformly-strongly connected inter-agent networks [79].

The remainder of this paper is organized as follows. In Section 4.2 we detail the distributed reachability problem. Section 4.3 contains the proposed methodology to solve the reachability problem for multi-agent systems in a fully distributed manner. In Section **??** we present a practically motivating example of multi-UAV formation flight, and formulate the corresponding distributed reachability problem. We demonstrate the proposed algorithm to compute reachable sets in real-time and compare our method with a centralized method. Finally, in Section 4.4 we present our concluding remarks and future directions.

4.2 Problem Formulation

Consider a multi-agent system (MAS) where individual agent states are coupled by a state feedback control law, with multiplicative gains for each network in the agents as:

$$\dot{x}_{\mathrm{i}}(t) = A_{\mathrm{i}}x_{\mathrm{i}}(t) + B_{\mathrm{i}}u_{\mathrm{i}}(t) + B_{1,\mathrm{i}}w_{\mathrm{i}}(t) \tag{4.1}$$

$$\text{where } u_{\mathrm{i}}(t) = K_{\mathrm{ii}}x_{\mathrm{i}}(t) + \sum_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}} K_{\mathrm{ij}}(x_{\mathrm{i}}(t) - x_{\mathrm{j}}(t)) \tag{4.2}$$

Here the state of agent i is $x_{\mathrm{i}} \in \mathbb{R}^{n_x}$, the control input $u_{\mathrm{i}} \in \mathbb{R}^{n_u}$, and an unknown, exogenous control $w_{\mathrm{i}} \in \mathbb{R}^{n_w}$, with gain matrices of appropriate sizes K_{i}, and system matrices $A_{\mathrm{i}}, B_{\mathrm{i}}$ and $B_{1,\mathrm{i}}$. The agents can communicate over a network denoted by the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of agents, and $\mathcal{E} \subseteq \{(\mathrm{i}, \mathrm{j}) : \mathrm{i}, \mathrm{j} \in \mathcal{V}\}$ is the set of all edges. The problem is characterized by agents having their dynamics coupled directly through their neighboring agents' states $x_{\mathrm{j}} \in \mathcal{N}_{\mathrm{i}}$ feeding into their own control inputs to

modify state x_i from (4.1). Additionally, the dynamics of different agents are unknown to each other, i.e., $(A_\mathrm{i}, B_\mathrm{i})$ is known only to the agent i themselves.

Given the dynamical system in (4.1), we can now define the reachable set computation problem. The reachable set for agent $\mathrm{i} \in \mathcal{V}$ is defined as the set of all possible states for agent i at some time $\tau \geq t_0$ as:

$$\begin{aligned} \mathcal{R}_\mathrm{i}(\tau; \mathcal{X}_{0,\mathrm{i}}, \mathcal{W}_\mathrm{i}) \triangleq \{x(\tau) \in \mathbb{R}^{n_x} : \dot{x} = A_\mathrm{i} x + B_\mathrm{i} u_\mathrm{i} + B_{1,\mathrm{i}} w_\mathrm{i}, \\ w_\mathrm{i}(t) \in \mathcal{W}_\mathrm{i}, x(t_0) \in \mathcal{X}_{0,\mathrm{i}}\} \end{aligned} \tag{4.3}$$

for initial set $\mathcal{X}_{0,\mathrm{i}}$, and admissible exogenous control set $\mathcal{W}_\mathrm{i}$.

Assumption 1. *Let the exogenous input set be bounded as* $\mathcal{W}_\mathrm{i} \triangleq \{w_\mathrm{i}(t) \in \mathbb{R}^{n_w} \mid \|w_\mathrm{i}(t)\| 2 \leq \rho_\mathrm{i}\}$.

In case the assumption above does not hold, the set $\mathcal{W}$ can easily be bounded by a sphere of radius ρ_i for a finite set $\mathcal{W}$. From the 'neighborhood-state feedback' control from (4.2), the reachable set $\mathcal{R}_\mathrm{i}(\tau; \mathcal{X}_{0,\mathrm{i}}, \mathcal{W}_\mathrm{i})$ clearly depends on $x_\mathrm{j} \, \forall \mathrm{j} \in \mathcal{N}_\mathrm{i}$. Simultaneously, x_j also evolves under dynamics $(A_\mathrm{j}, B_\mathrm{j})$. The distributed reachability problem is to compute the reachable sets in (4.3) for each agent, under coupled control input in (4.2), such that the dynamics $(A_\mathrm{i}, B_\mathrm{i})$ are not shared among agents.

4.3 Distributed Reachable Set Computation

To deal with the distributed reachability problem above, let us consider a simpler scenario. Let $\mathcal{L}$ be the Laplacian of the graph corresponding to $\mathcal{G}$, i.e.,

$$\mathcal{L}_\mathcal{G} \triangleq \mathrm{D}\left[\mathcal{G}\right] - \mathrm{Adj}\left[\mathcal{G}\right] \tag{4.4}$$

where $\mathrm{D}\left[\bullet\right]$ and $\mathrm{Adj}\left[\bullet\right]$ are the graph degree and adjacency matrices, respectively. Before we attempt to solve the distributed reachability problem for the MAS, let us consider a

centralized approach. To this end, we define the dynamics of the *stacked system* for the centralized representation of the MAS by redefining the following variables:

$$\begin{aligned} \xi(t) &\triangleq [x_1(t)^T, \cdots, x_N(t)^T]^T, \\ W(t) &\triangleq [w_1(t)^T, \cdots, w_N(t)^T]^T \\ \mathbb{A}(t) &\triangleq \text{diag}\left\{A_{\text{i}} + B_{\text{i}}K_{\text{ii}}\right\}_{\text{i}\in\mathcal{V}} + \mathcal{L}_{\mathcal{G}} \otimes B_{\text{i}}K_{\text{ij}}, \text{ and} \\ \mathbb{B} &\triangleq \text{diag}\left\{B_{1,\text{i}}\right\}_{\text{i}\in\mathcal{V}} \end{aligned} \tag{4.5}$$

This allows us to consider the evolution of the centralized MAS states $\xi(t) \in \mathbb{R}^{n_x \times N}$ as:

$$\dot{\xi}(t) = \mathbb{A}(t)\xi(t) + \mathbb{B}W(t) \tag{4.6}$$

The stacked system in (4.5), (4.6) allows us to consider a simpler, centralized reachability problem.

4.3.1 Centralized Reachable Set Computation

If a central computing entity were to compute the reachable sets $\mathcal{R}_{\text{i}}(\tau; \mathcal{X}_{o,\text{i}}, \mathcal{W}_{\text{i}})$, the equivalent centralized reachability problem can written compactly as:

$$\begin{aligned} \mathcal{R}^{\xi}(\tau; \Xi_0, \mathcal{W}) \triangleq \Big\{ &\xi(\tau) : \dot{\xi}(t) = \mathbb{A}\xi(t) + \mathbb{B}W(t), \tau \geq t_0, \\ &\xi_{\text{i}}(t_0) \in \Xi_0, W(t) \in \mathcal{W} \Big\} \end{aligned} \tag{4.7a}$$

$$\text{where } \Xi_0 \triangleq \prod_{\text{i}=1}^{N} \mathcal{X}_{\text{i},0} = \mathcal{X}_{1,0} \times \cdots \times \mathcal{X}_{N,0}\,, \text{ and}$$

$$\mathcal{W} \triangleq \prod_{\text{i}=1}^{N} \mathcal{W}_{\text{i}} = \mathcal{W}_1 \times \cdots \times \mathcal{W}_N \tag{4.7b}$$

From (4.7), the centralized reachability problem for the MAS is clearly a linear-system reachable set computation problem. Note that if the graph $\mathcal{G}$ is static (e.g., a smart grid, internet network, robot flocking, etc.), centralized system matrix $\mathbb{A}(t) = \mathbb{A}$ is time invariant. For simplicity of analysis, we will initially concern ourselves with the stationary graph such that $\mathcal{G}(t) = \mathcal{G}$. Therefore, we first investigate reachable set computation for the linear

time invariant (LTI) system $(\mathbb{A}, \mathbb{B})$ given characterizations of initial central state set Ξ_0 and admissible exogenous input set $\mathcal{W}$.

Lemma 2. *The admissible exogenous input set $\mathcal{W}$ is convex and bounded.*

Proof. Let $v_1, v_2 \in \mathcal{W}$. Let $v_1 = [a_1, \cdots, a_n]^T$ and $v_2 = [b_1, \cdots, b_n]^T$. From Assumption 1, we have $\|a_{\mathrm{i}}\| 2 \leq \rho_{\mathrm{i}}$ and $\|b_{\mathrm{i}}\| 2 \leq \rho_{\mathrm{i}}$. Consider their convex combination

$$\alpha v_1 + (1-\alpha) v_2 = \begin{bmatrix} \alpha a_1 + (1-\alpha) b_1 \\ \vdots \\ \alpha a_n + (1-\alpha) b_n \end{bmatrix}$$

for some $\alpha \in (0,1)$. Then, the following holds: $\|\alpha a_{\mathrm{j}} + (1-\alpha) b_{\mathrm{j}}\| 2 \leq \alpha \|a_{\mathrm{j}}\| 2 + (1-\alpha) \|b_{\mathrm{j}}\| 2 \leq \alpha \rho_{\mathrm{j}} + (1-\alpha)\rho_{\mathrm{j}} = \rho_{\mathrm{j}}$. So the convex combination of v_1, v_2 also lies in $\mathcal{W}$, also, the set $\mathcal{W}$ is bounded. □

We will concern ourselves with polytopic approximations of Ξ_0 and $\mathcal{W}$ utilizing approximate reachable set computation from [35], [72]. As a result, without a loss of generality, consider the initial state set and the admissible exogenous control set to be polytopes as:

$$\Xi_0 = \bigcap_{\mathrm{j}=1}^{n_1} \{v \in \mathbb{R}^{N n_x} \, : \, \langle c_{\mathrm{j}}(t_0), v \rangle \leq \gamma_{\mathrm{j}}(t_0)\} \tag{4.8}$$

$$\mathcal{W} = \bigcap_{\mathrm{j}=1}^{n_2} \{u \in \mathbb{R}^{N n_u} \, : \, \langle d_{\mathrm{j}}, u \rangle \leq \varepsilon_{\mathrm{j}}\} \tag{4.9}$$

where the polytopic sets Ξ_0 and $\mathcal{W}$ have n_1 and n_2 faces, respectively, and variables $c_{\mathrm{j}}(t_0), \gamma_{\mathrm{j}}(t_0), d_{\mathrm{j}}, \varepsilon_{\mathrm{j}}$ parameterize the hyperplanes defining the faces of the polytopes. Thus, the j^{th} hyperplane supporting the convex set Ξ_0 can be denoted by the tuple $H_{\mathrm{j}} \triangleq (c_{\mathrm{j}}(t_0), \gamma_{\mathrm{j}}(t_0))$, as shown in Fig. 4.2.

Theorem 4.3.1 (Polytopic Reachability [35])**.** *Let hyperplane $H_{\mathrm{j}} = (c_{\mathrm{j}}(t_0), \gamma_{\mathrm{j}}(t_0))$ support the initial set Ξ_0 at time t_0, at a point $\xi^*_{\mathrm{j}}(t_0)$. Then the point of contact $\xi^*_{\mathrm{j}}(t_0)$ evolves as:*

$$\dot{\xi}_{\mathrm{j}}(\tau) = \mathbb{A}\xi_{\mathrm{j}}(\tau) + \mathbb{B}W^*_{\mathrm{j}}(\tau) \tag{4.10}$$

where $W_{\mathrm{j}}^*(\tau)$ *solves the optimal control problem:*

$$W_{\mathrm{j}}^*(\tau) = \arg\max_{W \in \mathcal{W}} \Big\{ \langle \lambda_{\mathrm{j}}^*(\tau), \mathbb{A}\xi_{\mathrm{j}}^*(\tau) + \mathbb{B}W \rangle \Big\} \tag{4.11}$$

$$\dot{\lambda}_{\mathrm{j}}^*(\tau) = -\mathbb{A}^T \lambda_{\mathrm{j}}^*(\tau) \ \ s.t. \ \ , \lambda_{\mathrm{j}}^*(t_0) = c_{\mathrm{j}}(t_0), \ t_0 \leq \tau \leq t \tag{4.12}$$

for the j^{th} *hyperplane, and the co-state variable* $\lambda_{\mathrm{j}}^*(\tau)$*. Additionally, for* $\gamma_{\mathrm{j}}^*(\tau) \triangleq \langle \lambda_{\mathrm{j}}^*(\tau), \xi_{\mathrm{j}}^*(\tau) \rangle$*, the hyperplane* $(\lambda_{\mathrm{j}}^*(\tau), \gamma_{\mathrm{j}}^*(\tau))$ *supports the reachable set* $\mathcal{R}^{\xi}(\tau; \Xi_0, \mathcal{W})$ *at time* τ*, at the point* $\xi_{\mathrm{j}}^*(\tau)$*.*

Proof. Proof follows from Varaiya et al. in [35] and Theorem 1 in [72]. □

A geometric representation of Theorem 1 is depicted in Fig. 4.2, showing the evolution of the point of contact of hyperplane H_{j} over time. This allows us to find the evolution of the supporting points of all n_1 hyperplanes by solving equations (4.10) through (4.12) for each j in $1 \leq \mathrm{j} \leq n_1$. As a direct consequence of Theorem 1, the reachable set is bounded by the polytope:

$$\mathcal{R}^{\xi}(\tau) \subset \bigcap_{\mathrm{j}=1}^{n_1} \Big\{ \xi : \langle \lambda_{\mathrm{j}}^*(\tau), \xi \rangle \leq \gamma_{\mathrm{j}}^*(\tau) \Big\} \tag{4.13}$$

where the hyperplane $(\lambda_{\mathrm{j}}^*(\tau), \gamma_{\mathrm{j}}^*(\tau))$ touches the reachable set at the point $\xi_{\mathrm{j}}^*(\tau)$, as shown in Fig. 4.2.

Suppose $\dot{z} = M(t)z + B(t)u(t)$ is an arbitrary linear dynamical system. We define the state transition matrix the (at some time t) linear system (M, B) as $\Phi_M(t, t_0)$ such that $\Phi_M(t, t_0) = \dot{M}(t)\Phi_M(t, t_0)$, and $\Phi_M(t_1, t_1) = I$. Therefore, due to Theorem 4.3.1, the centralized computation of polytopic approximations of the reachable set can be carried out using the following steps.

S1: Initialize $\lambda_{\mathrm{j}}^*(t_0) = c_{\mathrm{j}}(t_0)$, then propagate the co-state as $\lambda_{\mathrm{j}}^*(\tau) = \Phi_{-\mathbb{A}^T}(\tau, t_0)\lambda_{\mathrm{j}}^*(t_0)$ from (4.12).

S2: Compute $W_{\mathrm{j}}^*(\tau)$ in (4.11) using the co-state from step 1:
$W_{\mathrm{j}}^*(\tau) = \arg\max_{W \in \mathcal{W}} \Big\{ \langle \Phi_{-\mathbb{A}^T}(\tau, t_0)\lambda_{\mathrm{j}}^*(t_0), W \rangle \Big\}$.

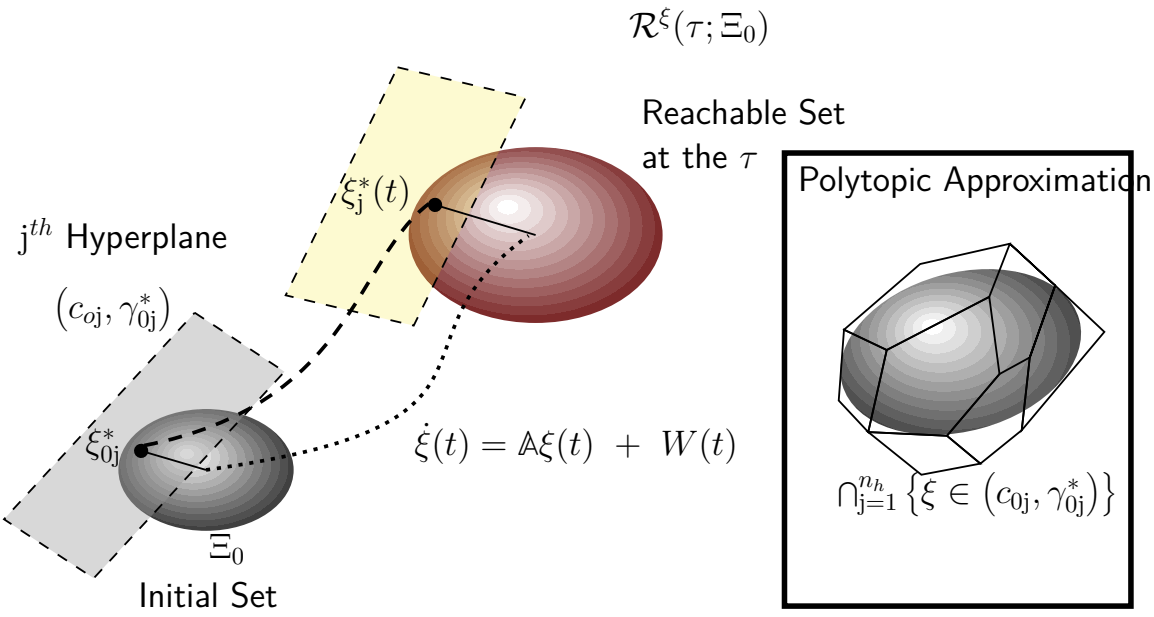

Figure 4.2. Polytopic Approximation of Reachable Set: Evolution of a selected Hyperplane

S3: Finally, the contact point is propagated using (4.10) as:

$$\xi^*_{\mathrm{j}}(\tau) = \Phi_{\mathbb{A}}(\tau, t_0)\xi_{\mathrm{j}}(t_0)^* + \int_{t_0}^{\tau} \Phi_{\mathbb{A}}(\tau, s)\mathbb{B}W^*_{\mathrm{j}}(s)ds \tag{4.14}$$

It must be noted from [35], that the polytopic representation in (4.13) simultaneously provides the outer and inner approximations to the reachable set as:

$$\begin{aligned} &\text{convex hull}\{\xi^*_{\mathrm{i}}(\tau), \cdots, \xi^*_{n_1}(\tau)\} \subset \mathcal{R}^{\xi}(\tau), \text{ and} \\ &\mathcal{R}^{\xi}(\tau) \subset \bigcap_{\mathrm{j}=1}^{n_1} \left\{\xi : \langle \lambda^*_{\mathrm{j}}(\tau), \xi\rangle \leq \gamma^*_{\mathrm{j}}(\tau)\right\} \end{aligned} \tag{4.15}$$

Remark 7 (Tightness of Approximation in (4.15))**.** *The approximation in (4.15) is tight in the sense that there does not exist a polytope with n_1 vertices $\underline{P}_{n_1}$, or a polytope $\overline{P}_{n_1}$ with n_1 faces, such that convex hull$\{\xi^*_{\mathrm{i}}(\tau), \cdots, \xi^*_{n_1}(\tau)\} \subset \underline{P}_{n_1} \subset \mathcal{R}^{\xi}(\tau) \subset \overline{P}_{n_1} \subset \bigcap_{\mathrm{j}=1}^{n_1} \left\{\xi : \langle \lambda^*_{\mathrm{j}}(\tau), \xi\rangle \leq \gamma^*_{\mathrm{j}}(\tau)\right\}$ (see [35]).*

4.3.2 Distributed Reachable Set Algorithm

Now that we established the procedure to compute centralized approximations (both inner, and outer) to the reachable set $\mathcal{R}^{\xi}(\tau)$ in steps S1-S3, we investigate the decentralized problem for the MAS in (4.1). Therefore, the information available to agent i can be summarized as:

$$\mathcal{I}_{\mathrm{i}} \triangleq \left\{ A_{\mathrm{i}}, B_{\mathrm{i}}, B_{1,\mathrm{i}}, \mathcal{N}_{\mathrm{i}}, \{K_{\mathrm{ij}}\}_{\mathrm{j}\in\mathcal{N}_{\mathrm{i}}}, \rho_{\mathrm{i}}, \{F_{\mathrm{i}}^{k}\}_{k=1}^{n_1}, \{H_{\mathrm{i}}^{k}\}_{k=1}^{n_2} \right\} \tag{4.16}$$

where $\{F_{\mathrm{i}}^{\mathrm{j}}\}_{\mathrm{j}=1}^{n_1}$ and $\{H_{\mathrm{i}}^{\mathrm{j}}\}_{\mathrm{j}=1}^{n_2}$ are the hyperplanes defining the bounding polytopes for the local sets $\mathcal{X}_{0,\mathrm{i}}$ and $\mathcal{W}_{\mathrm{i}}$, respectively. For simplicity of analysis, we assume the polytopes for each agent have the same number of faces n_1 and n_2.

To observe the distributed nature of the parameters of the polytopes, consider a simple case with $n = 1$ and $N = 3$, i.e., a one-dimensional, 3 agent MAS. The hyperplanes F_{i}^{k} defining initial sets are simply inequalities, as shown in Fig. 4.3 (a). The local hyperplanes are simply $F_{\mathrm{i}}^{1}, F_{\mathrm{i}}^{2}$, denoting the inequalities $x_1 \geq 0$ and $x_{\mathrm{i}} \leq 1$, respectively. As a result, the centralized state ξ lies in the cube of unit size as shown in Fig. 4.3 (b).

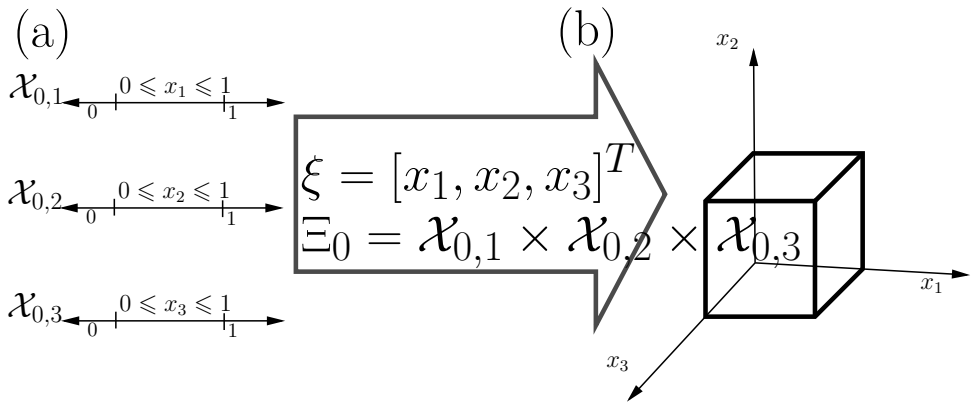

Figure 4.3. (a) Distributed, and (b) centralized structures of the polytope

Observe that due to the structure of $\Xi_0 = \mathcal{X}_{0,1} \times \mathcal{X}_{0,2} \times \mathcal{X}_{0,3}$, each agent is aware of $n_1 = 2$ faces defining the polytope Ξ_0. Each agent introduces n_1 number of planes to define Ξ_0, hence, Ξ_0 has Nn_1 total faces. The same is true for the polytope $\mathcal{W}$. That is, *the faces defining both sets, Ξ_0 and $\mathcal{W}$ are distributed across the graph $\mathcal{G}$* in the distributed reachability

problem. Similarly, the matrices $\mathbb{A}$ and $\mathbb{B}$ are also (row-wise) distributed across the graph $\mathcal{G}$ (see Fig. 4.4 (a)). This distributed information structure affects all steps S1 through S3 previously described. To this end, we need to carry out the steps S1 and S3 in a distributed manner, the 'stacked state' for the j^{th} hyperplane $\xi^*_{\mathrm{j}}(\tau)$ and the corresponding 'stacked co-state' $\lambda^*_{\mathrm{j}}(\tau)$ evolution equations in (4.10), and (4.12), respectively. This can be achieved by converting the corresponding state evolution differential equations to linear equations using their Laplace transforms. Note that for distributed linear dynamics (see Fig. 4.4 (a)), the corresponding Laplace transform of the dynamics is a distributed linear equation (d-LE) as (see Fig. 4.4 (b)) which can be written as:

$$\dot{\lambda}^*_{\mathrm{j}}(\tau) = -\mathbb{A}^T\lambda^*_{\mathrm{j}}(\tau) \xrightarrow{\mathcal{L}(\bullet)} s\Lambda^*_{\mathrm{j}}(s) = -\mathbb{A}^T\Lambda^*_{\mathrm{j}}(s) \tag{4.17}$$

$$\Rightarrow (sI + \mathbb{A}^T)\Lambda^*_{\mathrm{j}}(s) = \lambda^*_{\mathrm{j}}(t_0) = c^*_{\mathrm{j}}(t_0) \xrightarrow{\mathcal{L}^{-1}(\bullet)} \lambda^*_{\mathrm{j}}(\tau) \tag{4.18}$$

where $\Lambda^*_{\mathrm{j}}(s) \triangleq \mathcal{L}(\lambda^*_{\mathrm{j}}(t))$ is the Laplace transform of the co-state variable. Therefore, the co-state equation is a d-LE in the Laplace domain, where each agent i has information if its corresponding columns of the matrix based on local information set $\mathcal{I}_{\mathrm{i}}$, as shown in Fig. 4.4 (b). Similarly, for the Laplace transform of the state variable, $\mathcal{L}(\xi^*_{\mathrm{j}}(t)) \triangleq X^*_{\mathrm{j}}(s), \mathcal{L}(W^*_{\mathrm{j}}(t)) \triangleq W^*_{\mathrm{j}}(s)$, the stacked state evolves as:

$$(4.10) \xrightarrow{\mathcal{L}(\bullet)} (sI - \mathbb{A})X^*_{\mathrm{j}}(s) = \mathbb{B}W^*_{\mathrm{j}}(s) + \xi^*_{\mathrm{j}}(t_0) \tag{4.19}$$

where $W^*_{\mathrm{j}}(t)$ is given by the stacked optimal control law in (4.11). Since each agent i knows their own parameterizations for the local polytopes for $\mathcal{X}_{\mathrm{i},0}, \mathcal{W}_{\mathrm{i}}$, and their optimal control, $w^*\mathrm{j}(\tau)$, the state and co-state propagation equations in (4.18), and (4.18), both admit a d-LE form.

Therefore, we first outline a distributed linear equation solving method adapted from [79], and next carry out the distribute optimal control step S2. Consider a general d-LE as follows. Solve $Ax = b$, where agent i has access to certain rows of the problem, given by the

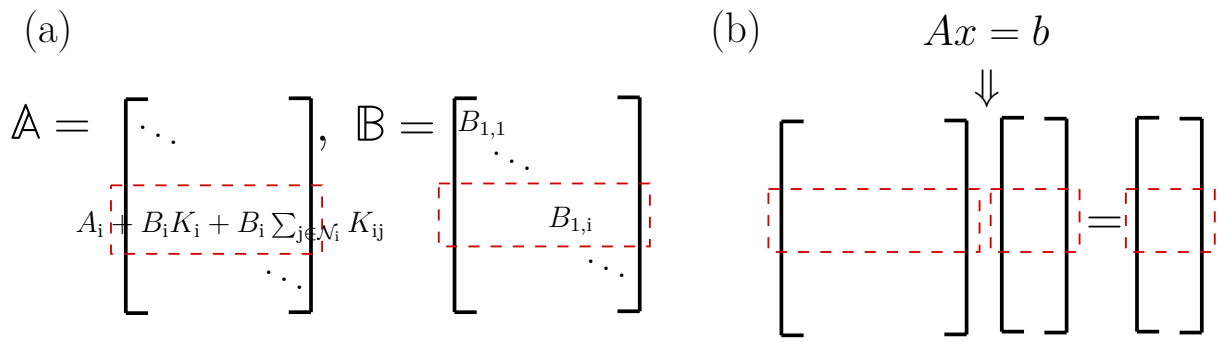

Figure 4.4. Distribution of system matrices information across the agents

tuple $(A_{[i,:]}, b_i)$. Additionally, the agents are connected according to the graph $\mathcal{G}$. Then, the d-LE can be solved by each agent as:

$$\hat{x}_i^{k+1} = \hat{x}_i^k - \text{Proj}_{\text{Ker}(A_{[i,:]})} \left[\hat{x}_i^k - \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \hat{x}_j^k \right] \tag{4.20}$$

where $\text{Proj}_{\mathbb{S}}[\boldsymbol{s}]$ is the orthogonal projection of vector $\boldsymbol{s}$ on to the set $\mathbb{S}$, and $\text{Ker}(\bullet)$ is the Kernel of the matrix $\bullet$ [79]. For the d-LE solution iterations in (4.20), the following Lemma holds.

Lemma 3. *If the graph $\mathcal{G}(t)$ is (repeatedly jointly) strongly connected, the iterations in (4.20) converges to the solution to the equation $Ax = b$ exponentially fast.*

Proof. See Theorem 1 from [79]. □

Next, we carry out the step S2 to compute the optimal control in (4.11) across the graph $\mathcal{G}(t)$. To compute $W_j^*(\tau)$, we can easily observe that we have polytopic bounds on the set $\mathcal{W}$ distributed among agents in the graph $\mathcal{G}$. Note that the set $\mathcal{W}$ is a polytope, whose vertices are spread across agents in the MAS but can be computed offline (similar to Fig. 4.4 (b)). Let the set $V \triangleq \{v_1, \cdots, v_K\}$ be the vertex set defining $\mathcal{W}$, and each agent be aware of a

subset V_i of the vertices (i.e., the vertex sets V_i's partition V). The problem of finding the optimal control $W_\mathrm{j}^*(t)$ in (4.11) can be rewritten as:

$$W_\mathrm{j}^*(t) = \underset{W_\mathrm{j}(t)\in\text{convex hull}\{V\}}{\arg\max} \langle \Phi_{-\mathbb{A}^T}(\tau, t_0)\lambda_\mathrm{j}^*(t_0), W_\mathrm{j}\rangle \tag{4.21}$$

However, the computation above is drastically simplified since Pontryagin's maximum principle guarantees that the maximum occurs on one of the vertices in V [35]. Therefore, each agent i can simply exchange the maximum across their neighborhoods as:

$$\begin{aligned} \hat{W}_\mathrm{i}^* &\leftarrow \underset{v\in V_\mathrm{i}}{\arg\max}\{\langle \Phi_{-\mathbb{A}^T}(\tau, t_0)\lambda_\mathrm{j}^*(t_0), v\rangle\} \\ \hat{W}_\mathrm{i}^* &\leftarrow \max_{\mathrm{j}\in\mathcal{N}_\mathrm{i}}\{W_\mathrm{i}^*, W_\mathrm{j}^*\} \end{aligned} \tag{4.22}$$

From (4.22), agents in the MAS simply keep a track of the vertex in the set V_i that maximizes the optimal control cost, and keep a local copy of the maximum of such vertices among its own neighborhood $\mathcal{N}_\mathrm{i}$. This simple procedure computes the maximum in at most T steps, where T is the graph diameter $\mathrm{diam}\,(\mathcal{G})$. Clearly, this requires the graph to be strongly connected, such that the local maximum is communicated through each neighborhood, and eventually the maximum is computed (in the worst case) along the longest path with the path length equaling the graph diameter.

This completes the fully distributed reachable set computation method, where we first proposed the centralized polytopic reachable set computation scheme and modified it to accommodate fully distributed MASs applications, *for a time-invariant, undirected, strongly connected graph*. For simplicity, we had assumed that the graph $\mathcal{G}(t)$ is strongly connected, and static at all times, i.e., $\mathcal{G}(t) = \mathcal{G}$. We now present our main results for time-varying MAS graphs.

4.3.3 Convergence of Algorithm 2 for Time-Varying Graphs

We will consider the convergence of Algorithm 2 for a specific class of time-varying graphs. To aid the convergence proof, we utilize the following properties from graph theory for sequences of graphs.

Definition 1. *Consider two graphs $\mathcal{G}_1, \mathcal{G}_2$ with the corresponding adjacency matrices* $\mathrm{Adj}\,[\mathcal{G}_1]$ *and* $\mathrm{Adj}\,[\mathcal{G}_2]$*, respectively. Then their* graph composition *is defined as the graph corresponding to the product of the two adjacency matrices as* $\mathcal{G}_1 \circ \mathcal{G}_2 \triangleq \mathcal{G}(\mathrm{Adj}\,[\mathcal{G}_1]\,\mathrm{Adj}\,[\mathcal{G}_2])$.

Definition 2. *Consider a (possibly infinite) time-varying graph sequence* $\mathbb{G} \triangleq \{\mathcal{G}(t_1), \mathcal{G}(t_2), \cdots\}$. *Then the sequence* $\mathbb{G}$ *(see [79]) is* repeatedly jointly strongly connected*, if a over a finite interval* $[\underline{t}, \overline{t}]$ *the graph composition* $\mathcal{G}_{\circ[\underline{t},\overline{t}]}$ *is strongly connected.*

The concept of strong connectivity of a graph $\mathcal{G}$ is based on the existence of a path between any two arbitrary agents in the graph at a given time t. However, when dealing with time-varying graphs, it is often necessary to consider the existence of a path between any two agents over a time interval $[\underline{t}, \overline{t}]$, rather than just at a single time t. This is where the concept of a repeatedly-jointly connected graph sequence $\mathbb{G}$ comes in. A repeatedly-jointly connected graph sequence $\mathbb{G}$ requires that there exists a path between any two arbitrary agents over the time interval $[\underline{t}, \overline{t}]$, for all $t \in [\underline{t}, \overline{t}]$. This is a generalization of the notion of strong connectivity, allowing for the graph to vary with time. Based on these definitions, the convergence of the proposed method can be considered for time-varying, undirected, repeatedly jointly strongly connected graphs. This means that the communication graph must be connected for all time intervals $[\underline{t}, \overline{t}]$, and there must exist a path between any two agents over each of these time intervals.

Theorem 4.3.2. *Let the MAS under consideration evolve according to (4.1) where the information* $\mathcal{I}_\mathrm{i}$ *is distributed across the graph* $\mathcal{G}(t)$*. If the time-varying graph sequence* $\mathbb{G}$ *is repeatedly jointly strongly connected, then the reachable sets* $\mathcal{R}_\mathrm{i}(\tau; \mathcal{X}_{0,\mathrm{i}}, \mathcal{W}_\mathrm{i})$ *can be computed efficiently. Specifically, the intermediate distributed computations can be performed exponentially fast (for steps S1 and S3) and the optimal control in S2 can be computed in finite time.*

Proof. Similar to $\mathrm{diam}\,(\mathcal{G})$ in (4.22), the optimal control step for time-varying graphs now depends on the maximum diameter of the graph composition over the interval $[\underline{t}, \overline{t}]$. Hence, step S2 converges in at most $\sup_{t\in[\underline{t},\overline{t}]} \mathrm{diam}\left(\mathcal{G}_{\circ[\underline{t},\overline{t}]}\right)$ steps. However, note that the slowest steps are still S1 and S3, both of which now require solving the distributed linear equations over

the graph sequence $\mathbb{G}$. From Theorem 1 in [79], the update rule for d-LE in (4.20) converges exponentially fast to the solution of $Ax = b$ over repeatedly jointly strongly connected graph sequence $\mathbb{G}$. Therefore, distributed solution to S2 converges to optimal control solution in finite time, while distributed solutions to S1 and S3 converge exponentially. □

4.4 Conclusions

In this research, we addressed the challenge of computing reachable sets for a multi-agent system (MAS) in a distributed manner. Specifically, we focused on approximating polytopic reachable sets for a given time horizon τ. We proposed a fully distributed method and a corresponding computation scheme that leverages techniques from distributed linear equation solving. Our proposed method was designed to work for communication graphs of the MAS that are repeatedly jointly strongly connected. We proved that the reachable sets obtained using our approach converge exponentially with respect to the number of iterations. The significance of our work lies in its potential to enable efficient and scalable computation of reachable sets for large-scale multi-agent systems, without relying on a centralized controller. This is particularly relevant in domains such as robotics, where distributed systems are commonly employed.

In our future work, we plan to investigate the use of directed graphs for inter-agent communication in the MAS to further extend the reachability analysis capabilities. This will enable us to model and analyze more complex scenarios with varying communication patterns. Additionally, we plan to explore numerically efficient methods for computing reachable sets in a fully distributed manner for time-varying MASs. One such approach we plan to investigate is the use of ellipsoidal reachability, which has been shown to be an effective approximation method for reachable sets in nonlinear systems [71]. By developing such methods, we aim to improve the efficiency and scalability of our distributed reachable set computation framework for MASs, making it applicable to larger and more complex systems.

Algorithm 2: Distributed Reachable Set Computation

Input: t_0, τ, T, local copies of $\mathcal{I}_\mathrm{i}$ (for each agent $\mathrm{i} \in \mathcal{V}$) from (4.16)

Parameters : initial polytopes: $\{\lambda_\mathrm{j}^*(t_0), c_\mathrm{j}^*(t_0)\}_{\mathrm{j}=1}^{n_1}, \{W_\mathrm{j}^*\}_{\mathrm{j}=1}^{n_2}$

compute initial contact points $\{\xi_\mathrm{j}^*(t_0)\}_{\mathrm{j}=1}^{n_1}$

while $\tau \leq T$ **do**

  for $\mathrm{j} \leftarrow 0$ **to** n_1 **do**

    $\lambda_\mathrm{j}^*(\tau) \leftarrow$ co-state update using `dLP_solver(•)` on (4.18)

  end

  for $\mathrm{j} \in \mathcal{V}$ **do**

    solve for optimal control $\hat{W}_\mathrm{i}^*$ using (4.22)

  end

  for $\mathrm{j} \leftarrow 0$ **to** n_1 **do**

    update $\xi_\mathrm{j}^*(\tau)$ for j^{th} hyperplane using $\lambda_\mathrm{j}^*(\tau), W_\mathrm{j}^*(\tau)$

    $\xi_\mathrm{j}^*(\tau) \leftarrow$ `dLP_solver(•)` on (4.19)

  end

  $\overline{\mathcal{R}_\mathrm{i}}(\tau) \leftarrow \{\lambda_\mathrm{i}^*(\tau), \xi_\mathrm{i}^*(\tau)\}$; `// outer approx.`

  $\underline{\mathcal{R}_\mathrm{i}}(\tau) \leftarrow$ convex hull$\{\xi_\mathrm{i}^*(\tau)\}$; `// inner approx.`

end

return **Output:** $\{\overline{\mathcal{R}_\mathrm{i}}(\tau; \mathcal{X}_{0,\mathrm{i}}, \mathcal{W}_\mathrm{i}), \underline{\mathcal{R}_\mathrm{i}}(\tau; \mathcal{X}_{0,\mathrm{i}}, \mathcal{W}_\mathrm{i})\}_{\mathrm{i}\in\mathcal{V}}$

Function `dLP_solver(`$(A_{[\mathrm{i},:]}, b_\mathrm{i}; \mathrm{i} \in \mathcal{V})$, $\mathcal{G}(t)$, n_ε`):`

  $\hat{x}_\mathrm{i}^0 \leftarrow \mathbf{0}$, $\forall \mathrm{i} \in \mathcal{V}$; `// arbitrary initialization for dLP solver`

  for $k \leftarrow 0$ **to** n_ε **do**

    Update $\hat{x}_\mathrm{i}^{k+1}$ using (4.20)

  end

  return x_i^*, $\forall \mathrm{i} \in \mathcal{V}$

5. DATA-DRIVEN CYBERATTACKS ON NETWORK CONTROL SYSTEMS

Thapliyal, O., & Hwang, I. (2022). Data-driven Cyberattack Synthesis against Network Control Systems. *arXiv preprint arXiv:2211.05203; Submitted to the 22nd World Congress of the International Federation of Automatic Control*. [80]

The research presented in this chapter is conducted by **Thapliyal, O.** under the supervision of Hwang, I.

Abstract

Network Control Systems (NCSs) are systems that rely heavily on communication channels, making them vulnerable to cyberattacks. Cyberattacks on NCSs can lead to eavesdropping, false data injection (FDI), and denial of service (DoS), resulting in degraded system performance. To launch an attack on an NCS, attackers can combine multiple techniques to cause a more significant impact. In this study, we propose a white-box cyberattack synthesis technique that begins with eavesdropping to gather system data and construct an equivalent system model. This equivalent model is then used to synthesize hybrid cyberattacks, combining FDI and DoS attacks on the NCS. The attack synthesis problem can be formulated as an optimal control problem. To solve the attack synthesis problem, we employ a data-driven reachable set computation, which provides real-time guidance on selecting NCS agents to be attacked while adhering to a specified attacker budget. The reachable sets are computed for the equivalent NCS model, providing quick and accurate decision-making capabilities for the attacker.

The proposed method provides a more realistic approach to cyberattack synthesis against NCSs with unknown parameters, making it useful for real-world scenarios. To demonstrate the effectiveness of the proposed method, we applied it to a multi-aerial vehicle formation control scenario. The results showed that the proposed method could efficiently synthesize hybrid cyberattacks that significantly degrade the NCS's performance, highlighting the importance of securing NCSs against cyberattacks.

5.1 Introduction

Recent trends in combining networks with control systems allow control engineers to solve complex tasks, design sophisticated control schemes, and model cooperation of spatially separate entities, via data sharing across communication networks. As a result, network control systems (NCSs) find varied applications in wireless sensor networks, industrial automation, distributed control systems, power grids, multi-robot cooperation, etc. [81]–[83]. However, due to increased reliance on communication, often over insecure channels or in the presence of malicious entities, NCSs face increased cybersecurity threats compared to their centralized counterparts. [81] have noted numerous occurrences of such cyberattacks on NCSs: from the famous StuxNet, and the capturing of RQ-170 reconnaissance aircraft, to cybersecurity threats against autonomous driving systems. Therefore, control theoretic methods can be utilized to inform NCS cybersecurity issues – both, from the control designer and attacker perspectives.

Related Works: In a previous study [84], cyberattacks were classified into two categories: black-box and white-box attacks, depending on the attacker's access to NCS information. Black-box attacks assume that the attacker has no knowledge of the inner workings of the NCS, while white-box attacks assume complete knowledge. These terms are borrowed from the adversarial machine learning community [85]–[87]. White-box attacks are more suited for identifying vulnerabilities in the control algorithms used in NCSs, while black-box attacks provide a more realistic approach to attack synthesis problems. The distinction between these types of attacks is crucial for designing effective countermeasures. For instance, a black-box attacker may exploit vulnerabilities in the communication channels of the NCS, while a white-box attacker may focus on manipulating the control algorithms to cause system instability.

The literature on control theory has thoroughly covered white-box cyberattacks against NCS. For instance, studies such as [88], [89], and [90] discuss this issue from the control designer's perspective. In [88], a "leader-follower" game is proposed as a model for the interaction between the attacker and defender. The Nash equilibrium solution is then compared to the optimal responses from both players. On the other hand, [89] quantifies NCS security

against stealthy injection attacks utilizing a reachability-based metric, assuming the attacker has complete knowledge of the system. A Kalman filter augmented with a neural network is employed to detect false data injection (FDI) attacks an NCS by [81], while guaranteeing controller stability. On the other hand, more realistic and sophisticated cyberattacks are often carried out as a combination of different attacks. Such white-box attacks often utilize statistical methods to develop system models [86], [91], by exploiting well known communication vulnerabilities of NCSs (see [81], [92] for more detailed survey on NCS vulnerabilities, and FDI attack procedures, respectively). As noted by [84], attackers can eavesdrop on black-box system data to develop auxiliary system models and perform injection attacks based on the model thus developed. [92] consider a different white-box FDI attack against a smart power grid (and the corresponding defense mechanism). They assume the attacker is able to tap into communication channels to eavesdrop for power grid NCS measurements, and then carries out FDI attacks that are stealthy to chi-squared detection tests on the measurement residue.

However, most black-box attack methods require offline training [91], computationally expensive machine learning [84], strong reachability assumptions [92], or complete knowledge of detection schemes [89]. To this end, we would like to investigate the synthesis of white-box FDI attacks against NCSs, while including realistic attacker capabilities. We assume the attacker is capable of observing the dynamical behavior of the NCS (in practice implemented by eavesdropping) to develop equivalent dynamical models. We propose a dynamic mode decomposition (DMD) approach to obtain rapid real-time auxiliary approximations to the NCS dynamics. The attack capabilities can be modeled as reachable sets of the auxiliary system, obtained in real-time through polytopic approximations [34], [35]. Additionally, the impact of the attack is demonstrated to be enhanced by isolating more vulnerable agents in the NCS.

Contributions: In our work, we employ DMD combined with polytopic reachable set computation, to model the unknown NCS and attacker capabilities, respectively. Therefore, the cyberattacks can be carried out with limited system data, and in real-time. The agent isolation is posed as a repeated semi-definite program (SDP), thus allowing the attacker to carry out more directed attacks against the NCS, while only relying on the system data. As

a result, the proposed method provides a realistic attack synthesis scenario under practical attacker capabilities.

The remainder of this paper is organized as follows. We present the problem formulation of cyberattack synthesis against the partially known network control system in Section 5.2. We formulate a DMD-based scheme combined with reachable set estimation to devise the cyberattack synthesis methodology in Section 5.3. In Section 5.4, we demonstrate the proposed method using a motivating scenario of injection attacks in actuator channels of a network of unmanned aerial vehicles (UAVs) engaging in formation flight and trajectory tracking. Finally, we present our concluding remarks in Section **??**.

5.2 Problem Formulation

Consider a network control system (NCS) where the individual agents have a linear time-invariant (LTI) dynamics, under a coupled state feedback control law, as follows:

$$\begin{aligned} \dot{x}_{\mathrm{i}}(t) &= A_{\mathrm{i}} x_{\mathrm{i}}(t) + B_{\mathrm{i}} u_{\mathrm{i}}(t), \\ u_{\mathrm{i}}(t) &= K_{\mathrm{ii}} x_{\mathrm{i}}(t) + \sum_{\mathrm{j} \in \mathcal{N}_{\mathrm{i}}(t)} K_{\mathrm{ij}} \left(x_{\mathrm{i}}(t) - x_{\mathrm{j}}(t) \right) \end{aligned} \tag{5.1}$$

Here, the state of the i^{th} agent is given by $x_{\mathrm{i}} \in \mathbb{R}^n$, the coupled control input $u_{\mathrm{i}} \in \mathbb{R}^p$, the LTI system matrices $(A_{\mathrm{i}}, B_{\mathrm{i}})$, and the state feedback gains $\{K_{\mathrm{ij}}\}_{\mathrm{j}=1}^{N}$, for a total of N agents. The agents in the NCS are connected according to an underlying graph $\mathcal{G}(t) = (\mathcal{E}(t), \mathcal{V})$, a tuple of the node set $\mathcal{V}$ and the edge set $\mathcal{E}(t) \subseteq \mathcal{V} \times \mathcal{V}$. Also, $\mathcal{N}_{\mathrm{i}}(t)$ is the neighborhood of the i^{th} agent at time t, defined as the set $\{\mathrm{j} : (\mathrm{i}, \mathrm{j}) \in \mathcal{E}(t), \mathrm{i} \neq \mathrm{j}\}$. Let $\mathrm{Adj}\left[\mathcal{G}(t)\right]$ denote the graph adjacency matrix, and $\mathcal{L}_{\mathcal{G}(t)}$ the graph Laplacian, defined as:

$$\begin{aligned} \mathrm{Adj}\left[\mathcal{G}(t)\right] \triangleq [a_{\mathrm{ij}}] = 1 \text{ if } (\mathrm{i}, \mathrm{j}) \in \mathcal{E}(t), 0 \text{ otherwise} \\ \mathcal{L}_{\mathcal{G}} \triangleq \mathrm{D}\left[\mathcal{G}(t)\right] - \mathrm{Adj}\left[\mathcal{G}(t)\right] \end{aligned} \tag{5.2}$$

where $\mathrm{D}\left[\mathcal{G}(t)\right]$ is the degree matrix with diagonal entries that denote the incoming degree (or $|\mathcal{N}_{\mathrm{i}}(t)|$) of the i^{th} agent.

From the attacker's perspective, the NCS dynamics can be rewritten as:

$$\dot{\boldsymbol{x}}(t) \triangleq \mathbb{A}_{\mathcal{G}(t)}\boldsymbol{x}(t) \tag{5.3}$$

using the 'stacked vector' definitions as:

$$\begin{aligned} \boldsymbol{x}(t) &\triangleq [x_1(t)^T, x_2(t)^T, \cdots, x_N(t)^T]^T \\ \mathbb{A}_{\mathcal{G}(t)} &\triangleq \text{diag}\left\{A_{\text{i}} + B_{\text{i}}K_{\text{ii}}\right\}_{\text{i}\in\mathcal{V}} + B_{\text{i}}K_{\text{ij}} \otimes \mathcal{L}_{\mathcal{G}(t)} \end{aligned} \tag{5.4}$$

The attacker intends to perform injection attacks to the system (5.3) with the following aims: (a) cause safety violations/collisions between any two agents in the NCS, (b) while remaining undetected by the NCS monitoring protocols. That is, false data injection (FDI) attacks to the system (5.3) can be written as $\dot{\boldsymbol{x}}(t) = \mathbb{A}_{\mathcal{G}(t)}\boldsymbol{x}(t) + \mathbb{B}^a\boldsymbol{u}^a(t)$, for the injection attack vector $\boldsymbol{u}^a(t)$, and a 'vehicle selection matrix' $\mathbb{B}^a$. The attack synthesis problem can then be written as:

$$\begin{aligned} &\text{find } \mathbb{B}^a, \boldsymbol{u}^a(t) \\ &\text{such that } \dot{\boldsymbol{x}}(t) = \mathbb{A}_{\mathcal{G}(t)}\boldsymbol{x}(t) + \mathbb{B}^a\boldsymbol{u}^a(t), \\ &\qquad \exists\, t^* \geq t_1 \text{ where } \left\|x_{\text{i}}(t^*) - x_{\text{j}}(t^*)\right\|_2 \geq d^*, \\ &\qquad \left\|\boldsymbol{u}^a(t)\right\|_2 \leq \rho \text{ for all } t \geq t_1 \end{aligned} \tag{5.5}$$

Here, ρ denotes the FDI budget, and d^* is some separation distance that the attacker wants to induce. Note that in case of a consensus problem (instead of formation flight), the attacker replaces the constraint above as $\leq d^*$ instead. Similarly, d^* denotes the safety violation threshold. We make the following assumption for the FDI problem:

Assumption 2. *The attacker is unaware of the system matrices* $\left\{A_{\text{i}}(t), B_{\text{i}}(t), \{K_{\text{ij}}\}_{\text{j}\in\mathcal{N}_{\text{i}}(t)}\right\}_{\text{i}\in\mathcal{V}}$, *but can collect discrete-time trajectory data* $\{x_1(t), \cdots, x_N(t)\}_{t=t_0}^{t=t_1}$ *over some time interval.*

Note that Assumption 2 characterizes the attack problem that we call the *gray-box approach to cyberattack synthesis*. Compared with the white-box approach (where the attacker has access to the entire NCS data, including system matrices), or the black-box approach (the attacker is allowed no data at all), gray-box approaches acknowledge the fact that more

severe cyberattacks can utilize tapped system data. In practice, the data collection in Assumption 1 is carried out via data association & state estimation, or eavesdropping into communicated data that the agents use to formulate individual control laws. Furthermore, severe cyberattacks are often preceded by simpler attacks to tap/eavesdrop on the system monitoring protocols or communication channels [84], [93]. As shown in Fig. 5.1, the attacker observes the state evolutions of the NCS agents.

5.3 Methodology

To devise the injection attack, we first outline a method to infer the system matrix $\mathbb{A}_{\mathcal{G}(t)}$. We will recap a Koopman operator based method (see works from [23], [94] for more detail) that estimates underlying time-varying dynamical structure.

Note that the NCS in (5.1) exhibits linear dynamics, but the stacked system in (5.3) is not perfectly linear due to the dependence on the time-varying graph $\mathcal{G}(t)$. As a result, we employ a Koopman operator-based approach to identify the linear dynamics. The Koopman operator can be thought of as a linear operator over measurable functional spaces that comes close (in operator norm) to the observed dynamics in the state space. Consider the dynamics of some trajectory data collected from some dynamical map:

$$\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k) \tag{5.6}$$

Even though the underlying dynamics can be nonlinear, the Koopman operator $\mathcal{K}$ acts on the space of all measurable functions $g : \mathcal{X} \to \mathcal{X}$ such that the evolution of these functions is linear. That is, the trajectory of a function g evolves linearly

$$g \circ F(\boldsymbol{x}_k) = \mathcal{K} \circ g(\boldsymbol{x}_k) \Rightarrow g(\boldsymbol{x}_{k+1}) = K \circ g(\boldsymbol{x}_k) \tag{5.7}$$

under the action of $\mathcal{K} : \mathscr{G} \to \mathscr{G}$, where $\mathscr{G}$ is the space of *observable functions*.

If the underlying state space is not finite, the Koopman operator, such that the observable trajectories are invariant in $\mathscr{G}$ and linear in $\mathcal{K}$, is infinite dimensional. Almost all existing literature that relies on Koopman operator theory to perform system identification from

state data, relies on finite dimensional approximations of $\mathcal{K}$ as a matrix K. If we restrict ourselves to observable functions spanned by full-state measurements $\boldsymbol{x}$, we try to find a matrix K that approximates $\mathcal{K}$. In reality, we carry out this approximation K given *data snapshots* as follows:

$$\mathrm{X} \triangleq \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_w \\ | & | & & | \end{bmatrix}, \mathrm{X}^+ \triangleq \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_{w+1} \\ | & | & & | \end{bmatrix} \tag{5.8}$$

$$K = \arg\min_K \left\| \mathrm{X}^+ - K\mathrm{X} \right\|_F = \mathrm{X}^+\mathrm{X}^\dagger \tag{5.9}$$

where $[\,\bullet\,]^\dagger$ denotes pseudoinverse and $\|\bullet\|_F$ is the Frobenius norm. Approximation in (5.9) is referred to as *dynamic mode decomposition* (DMD) [23], and can be thought of as linear regression given data snapshots. The resulting approximation K is a rank-r (usually, $r < w$ for *snapshot width* w) approximation of $\mathcal{K}$ confined to observables $g(\boldsymbol{x_k}) = \boldsymbol{x}_k$. Henceforth, we assume that we can obtain data snapshots of system (5.1) by virtue of Assumption 1.

Remark 8. *More complex observable families can be employed (e.g., Gaussian basis, polynomial basis, and neural networks to learn basis functions) depending on the degree of nonlinearity exhibited by the dynamical system, to result in extended dynamic mode decomposition methods. However, for our linear time-varying system identification, simple DMD suffices.*

Having solved for K, we have essentially performed regression for $\mathrm{X}^+ \approx K\mathrm{X}$. As a result, we can solve a modified version of the optimal control problem in (5.5) as:

$$\begin{aligned} &\text{find } \mathbb{B}^a, \boldsymbol{u}_k^a \\ &\text{such that } \dot{\boldsymbol{x}} = K_k\boldsymbol{x} + \mathbb{B}^a\boldsymbol{u}^a, \\ &\qquad \exists\, t^* \text{ where } \left\|x_{\mathrm{i}}(t^*) - x_{\mathrm{j}}(t^*)\right\|_2 \le d^*, \\ &\qquad \left\|\boldsymbol{u}^a\right\|_2 \le \rho \text{ for all } k \end{aligned} \tag{5.10}$$

To this end, we first define the set of all possible states that the agents in the NCS can take, when subjected to the norm-bounded injection attacks, as the *reachable set* for agent i as:

$$\mathcal{R}_{\mathrm{i}}(\tau;\rho) \triangleq \{x_{\mathrm{i},k} : k \leq \tau, \|u^a\|_2 \leq \rho, \text{ and } \dot{\boldsymbol{x}} = K_k \boldsymbol{x}\} \tag{5.11}$$

Note that for our linearized system, reachable sets at some time τ are lent useful properties of convexity under system linearity. As a result, reachable sets of linear systems can be represented by propagating their boundaries or characterizations of their boundaries, e.g., interval representations in our previous work [2], polytopic approximations by [32], [34], or ellipsoidal representations proposed by [71].

Next, we summarize the polytopic reachable set computation method for linear systems based on our work [32], applied to the DMD approximation $\dot{\boldsymbol{x}} = K_k \boldsymbol{x} + \mathbb{B}\boldsymbol{u}^a$, where $\boldsymbol{u}^a(t)$ lies in a bounded set $\Omega \triangleq \{u \in \mathbb{R}^{pN} : \|u\|_2 \leq \rho\}$. We first bound Ω with an s-faced polytope, where the i$^{\text{th}}$ face is given by:

$$\Omega \subseteq \bigcap_{\mathrm{i}=1}^{s} \{u \in \mathbb{R}^{pN} \mid \langle \nu_{\mathrm{i}}, u \rangle \leq d_{\mathrm{i}}\} \tag{5.12}$$

where ν_{i} is the normal vector, and d_{i} the distance from the origin, for the i$^{\text{th}}$ face. Let $H(a,b) \triangleq \{x \in \mathbb{R}^n : \langle a, x \rangle = b\}$ be the hyperplane with some normal vector a, at a distance b. The evolution of the reachable set can then be expressed using the polytopic characterization of Ω from (5.12) as follows.

Lemma 4. *Let hyperplanes $H_{\mathrm{i}}(\lambda_{\mathrm{i},0}, \gamma_{\mathrm{i},0})$ support the initial reachable set at time $t = 0$ at points $x^*_{\mathrm{i},0}$ for each hyperplane* i. *Then the reachable set at some time τ, under a bounded input, u^a, can be represented as:*

$$\mathcal{R}(\tau;\rho) \subseteq \bigcap_{\mathrm{i}=1}^{s} H_{\mathrm{i}}(\tau) \text{ where } H_{\mathrm{i}}(\tau) \equiv (\lambda_{\mathrm{i}}(\tau), \gamma_{\mathrm{i}}(\tau)) \tag{5.13}$$

where the time-varying support points evolve according to $\dot{x}^*_{\mathrm{i}} = K_k x^*_{\mathrm{i}} + \mathbb{B}\boldsymbol{u}^*$, and $\boldsymbol{u}^*$ solves the optimal control problem $\arg\max \langle \lambda_{\mathrm{i}}, K_k x^*_{\mathrm{i}} + \mathbb{B}\boldsymbol{u} \rangle$, *and the costate evolution is given by*

$\dot{\lambda}_{\mathrm{i}} = -K_k^T \lambda_{\mathrm{i}}$. *The variable* $\gamma_{\mathrm{i}}(\tau)$ *is the distance of the hyperplane* H_{i}, *i.e.,* $\langle \lambda_{\mathrm{i}}(\tau), x_{\mathrm{i}}^*(\tau) \rangle = \gamma_{\mathrm{i}}(\tau)$

Proof. The proof follows from [35] and Theorem 1 in [32]. □

Once the reachable set for the stacked system is obtained, $\mathcal{R}_{\mathrm{i}}$ is obtained by projecting $\mathcal{R}$ on to the i^{th} state space coordinate. Note that polytopic reachable sets are efficient to compute as the characterizations of reachable sets are a predefined number of hyperplanes. Furthermore, hyperplanes are characterized by points of contact x_{i}^* and normal vectors λ_{i}, which evolve under the linear dynamics. This evolution takes place under the optimal control law $\boldsymbol{u}_{\mathrm{i}}^*$, which is also easy to compute for the polytopic set Ω. This is because the maximum is guaranteed to occur on one of the vertices of the polytope in (5.12) that bounds Ω [35].

Once $\mathcal{R}_{\mathrm{i}}(\tau; \rho)$ are computed for some time τ, (5.10) is solved as follows. The attacker chooses $\mathbb{B}$ to indicate the agents being attacked (e.g., if the i^{th} agent is being attacked, $\mathbb{B}$ has an identity in the i^{th} place, and zero matrices for the remaining agents). The FDI attack vector $\boldsymbol{u}^*$ that solves $d(\mathcal{R}_{\mathrm{i}}(\tau + \Delta t), \mathcal{R}_{\mathrm{j}}(\tau + \Delta t)) \geq d^*$ for some Δt, also solves (5.10) for $t^* \in [\tau, \tau + \Delta t]$. This is because the FDI attack of $\boldsymbol{u}^*$ pushes the reachable sets for agents i and j at least d^* units apart in some time Δt, which necessarily means that the constraints in (5.5) are automatically satisfied. As a result, the reachable sets thus found encode the attacker's aim in the original problem. A detailed application of the proposed method is presented in the following section, and a summary of the method is summarized in Fig. 5.1.

5.4 False Data Injection Attack against Formation Control of UAVs

In this section, we consider a network of 5-UAVs performing formation control while trying to follow a desired trajectory. Such problems often arise in the NCS literature, NCS cybersecurity, distributed control & estimation [95], imperfections in state measurements for state estimation [96], [97], and UAV path planning problems [98].

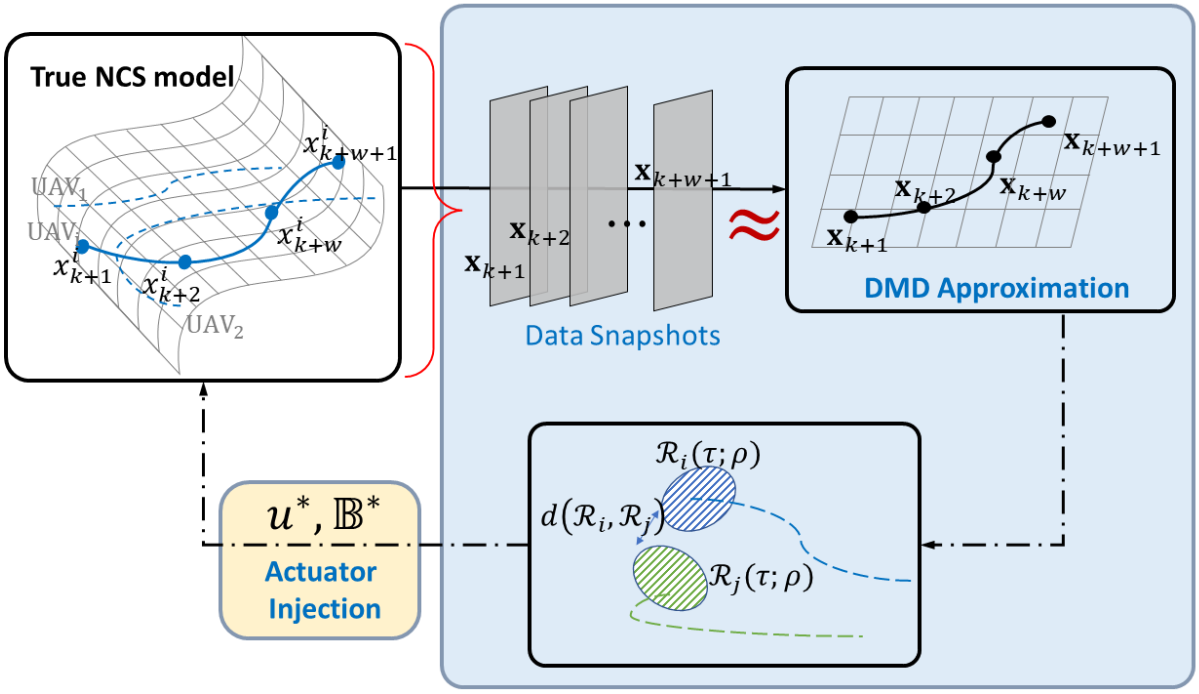

Figure 5.1. Schematic of the proposed cyberattack synthesis method

5.4.1 NCS Model

Although the proposed method is developed for continuous time NCSs, the numerical implementation has to be carried out in discrete time. The i$^{\text{th}}$ UAV in the NCS has its dynamics governed by the discrete-time version of (5.1) as $x_{\text{i},k+1} = A_{\text{i}} x_{\text{i},k} + B_{\text{i}} u_{\text{i},k}$, with the system matrices:

$$A_{\text{i}} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_{\text{i}} = \begin{bmatrix} \Delta t^2/2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix}$$

The dynamics above are the general double integrator in the 2D plane, and the i$^{\text{th}}$ UAV's state is defined as $x_{\text{i}} = [x, \dot{x}, y, \dot{y}]^T$. The formation control problem for the UAVs is solved by the following distributed control input:

$$u_{1,k} = K_1(x_{1,k} - x_k^\star) + \sum_{\text{j}\in\mathcal{N}_1} K_{1\text{j}}(x_{1,k} - x_{\text{j},k} - x_{1\text{j}}^\star) \tag{5.14}$$

$$u_{\text{i},k} = \sum_{\text{j}\in\mathcal{N}_\text{i}} K_{\text{ij}}(x_{\text{i},k} - x_{\text{j},k} - x_{\text{ij}}^\star),\ \text{i} = \{2\cdots,5\} \tag{5.15}$$

Equation (5.14) denotes the control input of the leader UAV, according to the reference trajectory $x_k^\star$ which the entire formation must follow. Equation (5.15) denotes the control sequence for all of the remaining follower UAVs who try to conform to a prescribed desired formation $x_{\text{ij}}^\star$. Note that for the UAV NCS described by (5.14) and (5.15), the NCS error dynamics follows (5.4), as noted by [98]. The desired formation trajectory is given by $x_k^\star = [-k\sin(3k/100), 1, -k\cos(3k/100), 1]^T$, for a simulation time of 100s, a sampling time of $\Delta t = 0.2$s, and a desired formation as shown in Fig. 5.2. The stabilizing gain to guarantee formation control was found by [98] to be:

$$K_{\text{ij}} = \begin{bmatrix} -0.2263 & -0.4712 & 0 & 0 \\ 0 & 0 & -0.2263 & -0.4712 \end{bmatrix}$$

The resulting trajectory of the 5-UAV NCS is shown in Fig. 5.3 (left). The initial positions of the UAVs, denoted by circles, were chosen randomly. The final positions denoted by crosses confirm a successful formation and trajectory tracking by the leader UAV.

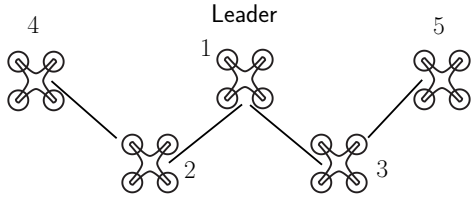

Figure 5.2. Desired UAV NCS formation

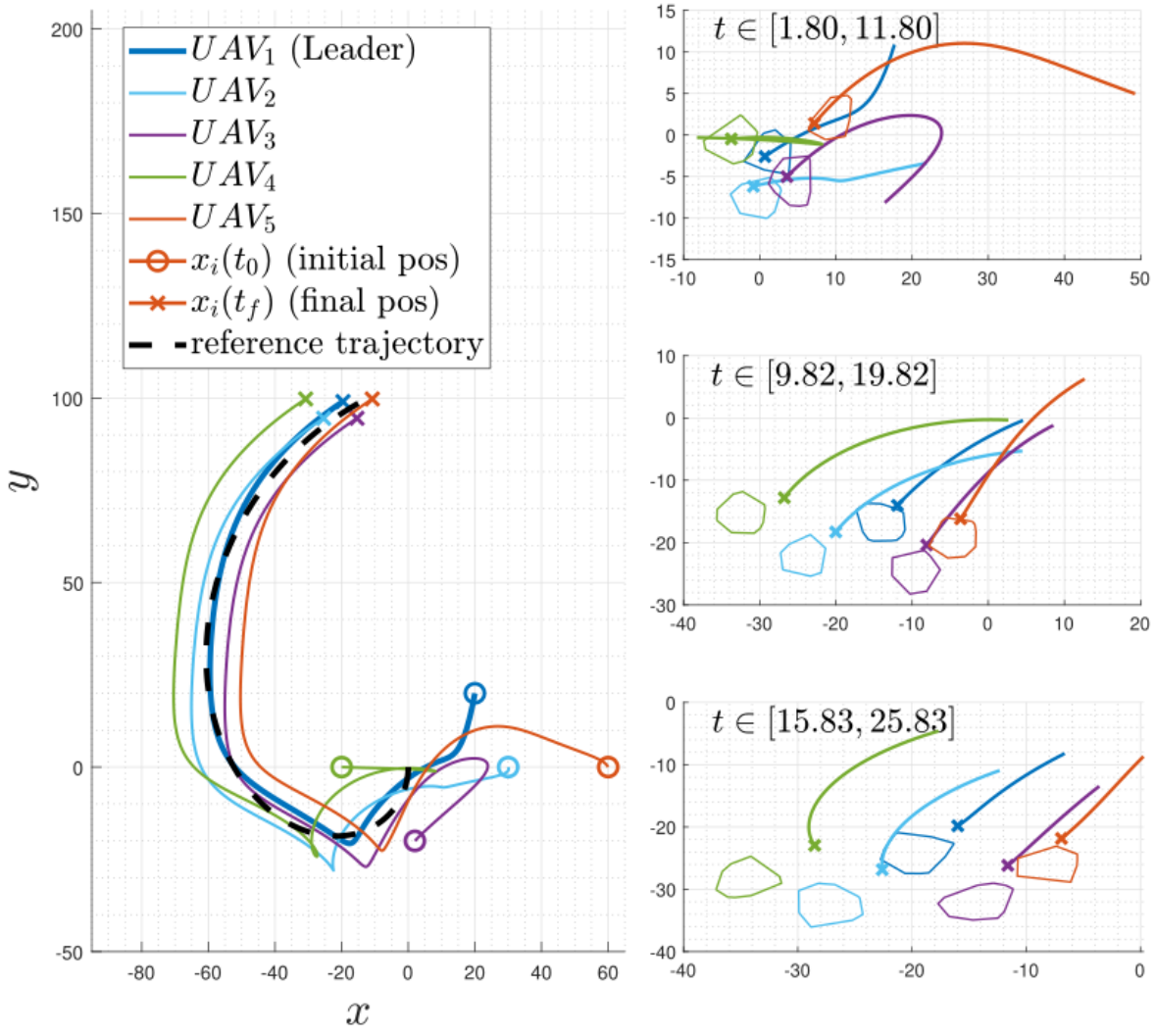

Figure 5.3. Formation control for 5-UAV network under gains K_{ij} *(left)*; Approximate reachable sets for DMD-based auxiliary model *(right)*

5.4.2 Attack Synthesis

In order to find the auxiliary model $\boldsymbol{x}_{k+1} \approx K\boldsymbol{x}_k$ using DMD, it is assumed that the attacker performs state estimation to obtain the data matrices X and X^+ in (5.9), with a snapshot width $w = 50$, at a sampling rate of $\Delta t = 0.2$s. The FDI attack budget is given

by $\rho = 0.05$. We approximate this admissible attack set as an 8-sided polygon with vertices chosen randomly such that the $\|\boldsymbol{u^a}\| \leq \rho$ is circumscribed by the polygon. The attacker then computes the reachable sets according to Lemma 1, and propagates them over time to synthesize FDI attacks. The propagated reachable sets at selected time instances are shown in Fig. 5.3 (right). The tails in the comet plots denote the previous trajectory of the given UAV over the intervals shown.

Since UAVs are trying to conform to a specified formation, attacked agents are selected based on the reachable sets at a given time τ as:

$$(\mathrm{i}^*, \mathrm{j}^*) = \arg\max_{\mathrm{i,j}} d\left(\mathcal{R}_\mathrm{i}(\tau;\rho), \mathcal{R}_\mathrm{j}(\tau;\rho)\right) \tag{5.16}$$

$$\boldsymbol{u}^a = \arg\max_{\|u\|\leq 0.05} d\left(\mathcal{R}_\mathrm{i}(\tau+\Delta t;\rho), \mathcal{R}_\mathrm{j}(\tau+\Delta t;\rho)\right) \tag{5.17}$$

where the UAVs to be attacked $(\mathrm{i}^*, \mathrm{j}^*)$ are chosen whose reachable sets are farthest apart (5.16), and the FDI attack is chosen to drive them further apart at the next time step, while being subject to the attack budget. These distances can be computed efficiently as the sets are polytopic.

The targeted agents change over time, depending on the relative locations of their reachable sets, as in Fig. 5.1. Finally, $\mathbb{B}^a$ is the corresponding matrix with a zero matrix at all positions, and an identity matrix at positions $(\mathrm{i}^*, \mathrm{j}^*)$ to denote FDI attacks taking place at the selected agents. The effect of such an attack over time can be observed in the trajectories shown in Fig. 5.4 (left). Note that the stacked system evolves under $\mathbb{A}_\mathcal{G}$, and the formation error is guaranteed to go to zero under the proposed gain, as the dynamics of the stacked error is stable [98]. As a result, the FDI attack can be seen to cause a disruption in only the trajectory tracking error, and the desired formation is still attained (see Fig. 5.4 (left)).

To cause a disruption in the formation, the stability of the stacked system can be compromised if the underlying connected graph can be disconnected. This is carried out by preceding the FDI attack with a sustained denial of service (DoS) attack on a 'vulnerable agent', whose communication link when disrupted causes the underlying graph to be disconnected. However, the auxiliary DMD system evolves under a time-varying matrix K, which

does not necessarily preserve the underlying graph structure in (5.4). This is because the matrix K was obtained by minimizing the trajectory error in (5.9), and does not necessarily find the matrix $\mathcal{L}_{\mathcal{G}}$ (as DMD can be though of as a special case of L_2 regression [23]). Therefore, to find an estimate of the underlying graph Laplacian, the attacker solves the following equation:

$$\hat{\mathcal{L}} = \arg\min_{L} \|K - (S + T \otimes L)\|_F \tag{5.18}$$

for some matrices S and T, such that L is a candidate Laplacian, i.e., L satisfies $L \succeq 0$ and $L\mathbb{1} = 0$. The minimization of the Kronecker product above can be rewritten as a minimization over γ for some bound $(K - (S + T \otimes L))^T (K - (S + T \otimes L)) \preceq \gamma^2 I$. This can be rewritten as a 'quasi' semi-definite program (SDP) using Schur complement as:

$$\begin{aligned} &\min \gamma \\ &\text{subject to } \begin{bmatrix} \gamma I & K - (S + T \otimes L) \\ [K - (S + T \otimes L)]^T & \gamma I \end{bmatrix} \succ 0 \\ &L \succeq 0, L\mathbb{1} = 0 \end{aligned} \tag{5.19}$$

Note that the equation above is a proper SDP if the matrix T is fixed. Similar SDP formulations to minimize matrix norms over Kronecker product are proposed by [99]. We can solve (5.19) as an alternating SDP as follows. Starting with an arbitrary but fixed value of S and T, we solve the SDP in (5.19) to find L. Next, fix T and the value L thus obtained to solve for S, followed by solving for T after fixing L and S, respectively. The alternating SDP procedure is carried out until γ stops improving above a prefixed threshold, resulting in $\hat{\mathcal{L}}$. This alternating SDP procedure is derived by [99] in more detail. Using this alternating SDP method, the attacker was able to recover the graph Laplacian matrix $\hat{\mathcal{L}}$ in 16 iterations of the alternating SDP where γ stopped improving beyond 10^{-6}. Next, solving for the second largest eigenvalue of the Laplacian $\lambda_2(\hat{\mathcal{L}})$ immediately provides the vulnerable nodes in the underlying strongly connected graph (i.e., there exist connecting paths between any two arbitrary nodes of the graph). This is because the second largest eigenvalue of

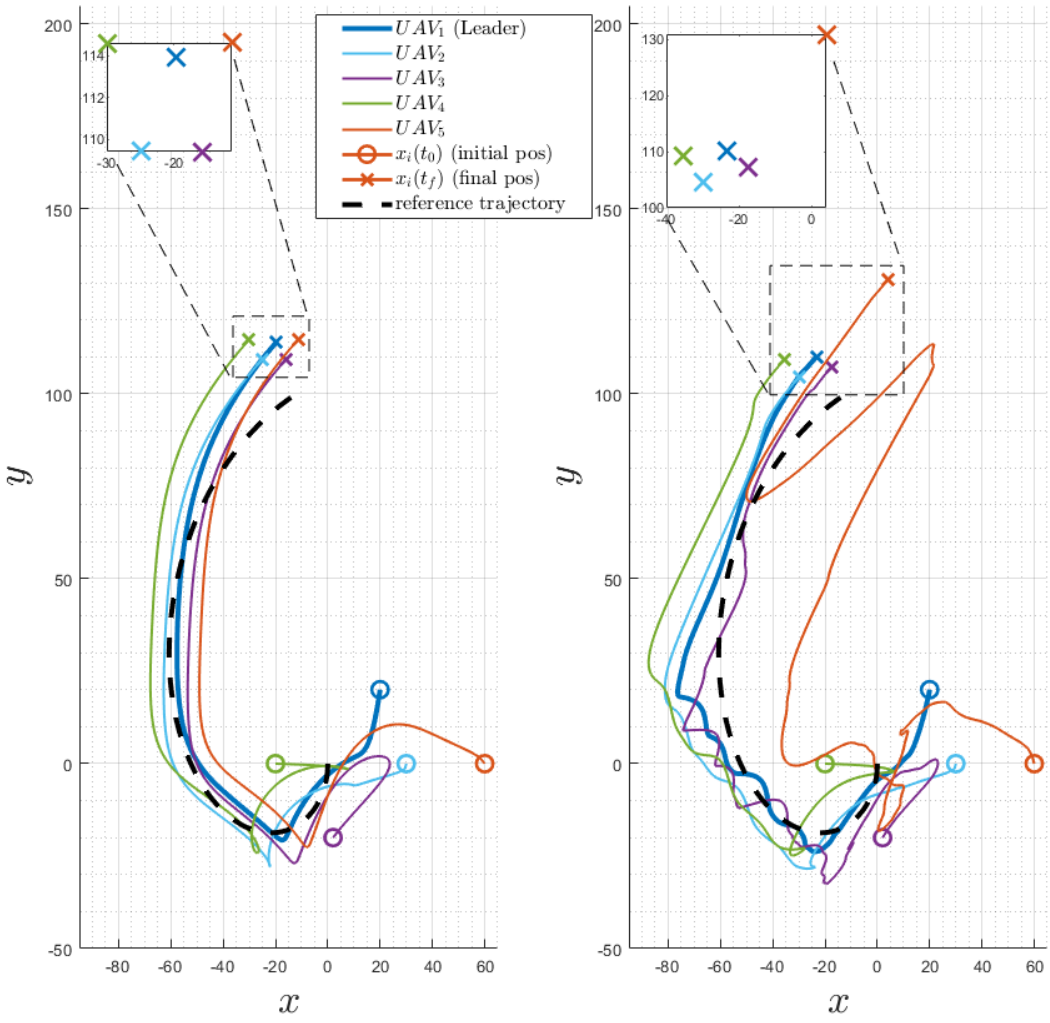

Figure 5.4. Impact of false data injection attacks on UAV trajectories: FDI attacks *(left)*; FDI combined with DoS on agent 5, link 5–3 *(right)*

the graph Laplacian is an immediate metric of graph connectivity [100], and all distributed control of the form (5.1) relies on the underlying graph being strongly connected [97]. The eigenvector corresponding to $\lambda_2(\mathcal{L}_\mathcal{G})$, called Fiedler eigenvector, provides immediate relative importance of each node of the graph $\mathcal{G}$ towards graph connectivity [100]. As a result, the attacker chooses to DoS the i$^{\text{th}}$ UAV (link 5–3), thereby causing the underlying graph to be

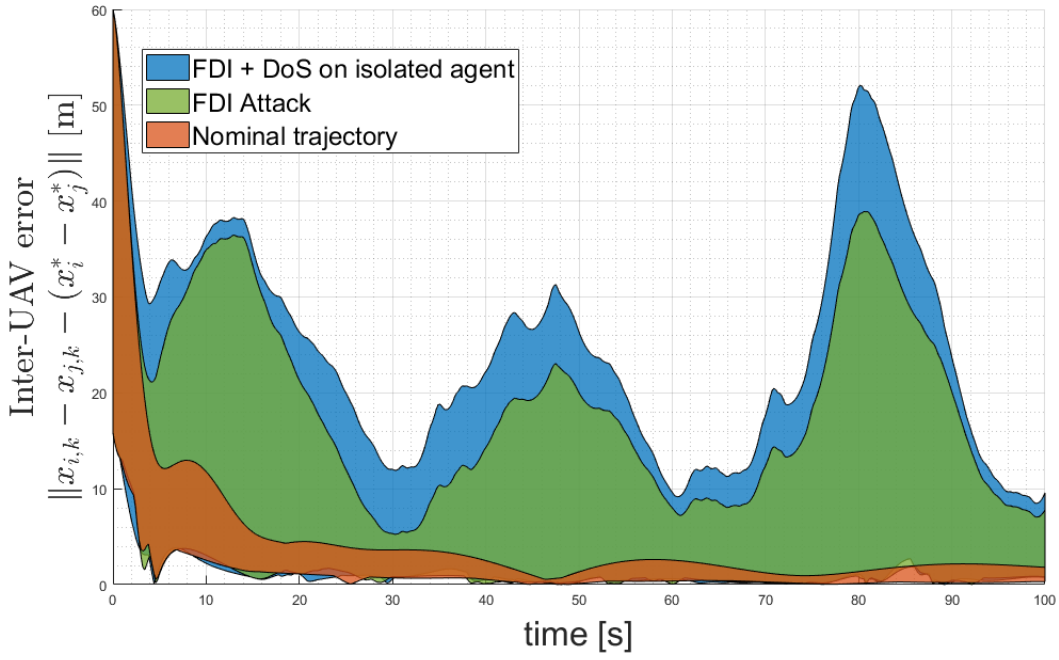

Figure 5.5. Inter-UAV errors plotted against time

disconnected, and the stacked system is no longer stable. The FDI attacks are carried out as outlined earlier.

The resulting trajectory can be seen in Fig. 5.4 (right). Compared to the reachable set-based FDI attacks, the attack mechanism of DoS attack on UAV 5 followed by FDI attacks causes disruption in formation as well as a failure in trajectory tracking. The same is observed in the plot of inter-UAV position errors, $\|x_{\mathrm{i},k} - x_{\mathrm{j},k} - (x_{\mathrm{i}}^* - x_{\mathrm{j}}^*)\|$, shown for the nominal NCS, FDI attacks on the NCS, and the final FDI combined with alternating SDP-based DoS attacks to isolate UAV 5 (see Fig. 5.5). As a result, the attacker can cause the formation errors to increase and accumulate over time. Therefore, by utilizing the reachable sets, the attacker can cause disruption in formation control and trajectory tracking in the UAV NCS discussed.

5.5 Conclusion

In this paper, we developed a novel cyberattack synthesis mechanism targeting network control systems (NCSs) with unknown dynamics. We proposed a dynamic mode decomposition-based method to first estimate reachable sets of the NCS agents, followed by false data injection (FDI) attacks by driving the reachable sets as far apart as possible. We demonstrated the proposed method using an illustrative scenario of unmanned aerial vehicle (UAV) formation flight and trajectory tracking. The proposed method was observed to cause failure in both, formation and trajectory tracking, upon preceding the FDI attacks by denial of service (DoS) attacks to isolate certain agents.

Future work will be to utilize exact reachable sets from the defender's perspective to provide guarantees on the controller against FDI attacks of a fixed budget.

6. DATA-DRIVEN CYBERATTACKS ON SUPERVISORY CONTROL SYSTEMS

Thapliyal, O., & Hwang, I. (2021, June). Learning based Cyberattack Design and Defense for Supervisory Control Systems. *In 2021 European Control Conference (ECC)* (pp. 144-149). IEEE. [84]

The research presented in this chapter is conducted by **Thapliyal, O.** under the supervision of Hwang, I.

Abstract

Cyber-physical systems (CPSs) are complex and often require multiple controllers, each with specific design requirements. Hybrid controllers for CPSs employ a supervisor to switch among a set of controllers to ensure proper operation. The supervisor is typically a computer program, separate from the physical layer of the CPS. While individual controllers' vulnerabilities to cyberattacks and their resilience to such attacks have been studied, smarter cyberattacks could destabilize CPSs by exploiting vulnerabilities in the supervisory logic itself. This could be followed by a targeted attack on an individual controller. However, in realistic scenarios, little to no information is available about the CPS models and dynamics, making machine learning-based approaches more appropriate for realistic cyberattacks.

This chapter focuses on cyberattack and defense design for supervisory controllers in CPSs when there is only partial information available to both the attacker and the defender. The proposed approach is advantageous because it does not rely on detailed knowledge of the CPS's underlying mathematical models, which is often not available in practice. Additionally, the paper proposes a data-driven defense scheme for such attacks, where the attack mechanism is unknown to the designer. Overall, the paper provides a novel and practical approach for addressing cyberattack vulnerabilities in CPSs with multiple controllers.

6.1 Introduction

Supervisory control is a versatile control methodology that has found applications in a variety of systems, including hybrid, nonlinear, and networked control systems. One of the most common applications is in systems utilizing Supervisory Control and Data Acquisition (SCADA), such as chemical plants, industrial control systems, and process control systems. Another important application of supervisory control is in multi-agent systems, where agents are under supervision. Supervisory control can model human-machine interactions by allowing the human operator to retain supervisory status over the plant ([101], [102]) or by delegating supervision to an automatic logic or a computer program ([103], [104]). While human-machine interface (HMI) design deals with problems related to the former, automatic control deals with the latter. Cybersecurity is crucial for CPSs due to their connected and hybrid nature. Recent cyberattacks against SCADA have highlighted the importance of developing robust cyberattack defense strategies for such systems. Therefore, mathematical analysis of cyberattack design and defense strategies has become increasingly important for addressing the challenges arising from real-life applications of complicated networked and CPSs.

Although literature has extensively studied cyberattack vulnerabilities of linear systems, for instance in [105] and [6], most methods suffer from two drawbacks. Firstly, for attack design, they assume a *white box* model where the attacker knows the details of the linear system. Secondly, hybrid controllers can bypass the vulnerabilities of linear systems by switching among usable controllers when subjected to cyberattacks. This capability of supervisory controllers to minimize the effects of cyberattacks by switching among a bank of controllers is discussed in [106]. Readers are referred to a more comprehensive survey of existing cyberattack mechanisms, vulnerabilities of CPSs, and attack types in [93], [107].

While white box type attacks are best suited for a vulnerability analysis, since they assume a worst case scenario where the adversary has complete knowledge of the system, these results are obviously not useful to attackers to implement *black box* (no intricacies of the CPS are known by the attacker), or even *gray box* attacks (some intricacies of the CPS are known by the attacker), since mathematical details of the plants, actuators, and sensors

are almost always obscured from attackers. White box attacks assume that the attacker has complete knowledge of the system, including its mathematical details, and therefore are best suited for vulnerability analysis. However, in practice, attackers rarely have access to such information and thus resort to black box or gray box attacks. Black box attacks assume no knowledge of the system, while gray box attacks assume some level of knowledge about the system, such as its structure or behavior.

The results of vulnerability analysis based on white box attacks may not be useful to attackers attempting to perform black box or gray box attacks. Therefore, developing defense mechanisms that can withstand attacks of unknown synthesis mechanisms, as well as designing attack strategies that can be effective in the absence of complete system knowledge, is important. The proposed learning-based methodology for gray box cyberattacks and corresponding defense methods addresses this challenge and offers advantages such as realism and real-time performance. Vulnerabilities of supervisory controllers, modeled as discrete event systems (DESs), subject to cyberattack have been studied in [108], [109]. Although supervisory controllers are widely used in cyber-physical systems (CPSs), the vulnerabilities of such controllers have not been extensively studied in the literature. The interplay between continuous and discrete states of each agent in a CPS creates a fundamentally different problem compared to the DES approach that is commonly discussed in literature. This added complexity is manifested in supervisory controllers as an abstract switching mechanism on top of individual controller units. As a result, attackers can exploit these vulnerabilities by causing unwanted switching behavior, which may activate incorrect controllers. Even if all controllers are designed to reject attacks, an attacker can still cause the so-called “chattering phenomenon" resulting in high-frequency “bang-bang" control cycles that can significantly degrade the system’s performance and reduce the longevity of the actuators.

The proposed method in this chapter aims to exploit the intricate nature of supervisory controllers, allowing attackers to launch injection attacks on the supervisory protocol itself. The actual plant model and actuator details are obscured from the attacker, while it is assumed that the attacker is allowed to insert itself in the communication layer between the controllers and the supervisor stealthily, as shown in Fig. 1. attack synthesis consists of two key steps: the attacker develops an auxiliary statistical model of the supervisor’s transition

logic, and an optimal attack based on the auxiliary model is injected to cause switching in the plant's controllers. From the partial knowledge of the system, the attacker constructs an 'auxiliary model' for the system, which assists in performing the attacks. Modeling supervisory logic poses a significant challenge due to the presence of discontinuities and transition conditions based on mixed continuous and Boolean logic (as discussed in [110]). To address this issue and accommodate unknown plant models, we incorporate artificial neural networks (ANNs). The injection of optimal attacks is then transformed into an adversarial machine learning problem (as detailed in [111]). We devise a defense mechanism that utilizes generative adversarial networks (GANs), which have previously been employed in anomaly detection and reconstruction in both time-series data and images (as reported in [112], [113]). In this chapter, the defense-GAN comprises a detector ANN and a reconstructor ANN, which are trained against each other in a two-person game, following the standard GAN training procedure.

We propose a novel approach for performing gray box cyberattacks on supervisory controllers and developing corresponding defense methods in situations where the attacker has incomplete knowledge about the CPS. This is a significant contribution to the field, as it provides a more realistic approach to attacking and defending CPSs without making any assumptions about the target CPS or attack synthesis model. The proposed method is advantageous in two ways. Firstly, it allows for more realistic attacks and defenses as it does not rely on white box assumptions. This means that the approach is not dependent on specific details of the target CPS or attack synthesis model, which makes it more applicable to real-world scenarios. Secondly, once the statistical models are trained, the attack and defense methods can be performed in real time, which makes it a practical solution for protecting CPSs against cyberattacks.

This chapter is organized as follows. In Section 6.2, we introduce the hybrid system and supervisory controller model that we study. Section 6.3 presents the formulation of the attack and defense design problems as an adversarial machine learning problem. In Section 6.4, we provide a practical example of a multi-robot formation controller to demonstrate the proposed attack method, and we train a defensive GAN to counter these attacks. Lastly, in Section 6.5, we conclude this chpter with our final remarks.

6.2 Problem Formulation

This section presents a mathematical model of the cyber-physical system (CPS) under attack. Consider a discrete-time linear system with its dynamics given as follows:

$$x_{k+1} = Ax_k + Bu_k(\sigma_k) \tag{6.1}$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state, $u_k(\sigma_k) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input as advised by the supervisor based on the discrete state $\sigma_k \in \Sigma$, and A and B are system matrices of appropriate dimensions for time indices $k \in \mathbb{N}$. The discrete event nature of control law $u_k(\sigma_k)$ is introduced by the supervisory logic, while the plant itself evolves based on the difference equation on x_k (see [103]). The supervisor itself is a finite automaton given by the tuple $S = (\Sigma, \mathcal{X}, \mathcal{U}, \delta, \phi)$, with the discrete state $\sigma_k \in \Sigma$, discrete state transition function $\delta : \Sigma \times \mathcal{X} \to \Sigma$, and controller output function $\phi : \Sigma \to \mathcal{U}$. The controller's discrete state σ_k and controller output u_k then evolve as:

$$\sigma_k = \delta(\sigma_{k-1}, x_k), \text{ and } u_k = \phi(\sigma_k) \tag{6.2}$$

A bank of state feedback controllers is a finite set $\Sigma \subset \mathbb{R}$, each matched to a partition of the state space, is given by:

$$\begin{aligned} \sigma_k = \delta(\sigma_{k-1}, x_k) = \mathrm{j} \text{ if } \mathcal{G}_{\sigma_{k-1}\mathrm{j}}(x_k) \geq 0 \\ u_k(\mathrm{j}) = K_\mathrm{j} x_k \ \text{, for gains indexed by } \mathrm{j} \in \Sigma \end{aligned} \tag{6.3}$$

where $\mathcal{X} = \bigcup_{\mathrm{i}\in\Sigma} \mathscr{S}_\mathrm{i}$ and $\mathscr{S}_\mathrm{i} \cap \mathscr{S}_\mathrm{j} = \emptyset$ for $\mathrm{i} \neq \mathrm{j}$, $\mathcal{G}_{\mathrm{ij}}$ is the boundary between $\mathscr{S}_\mathrm{i}, \mathscr{S}_\mathrm{j}$, or the *guard condition* for the $\mathrm{i} - \mathrm{j}$ discrete state transition, and K_j the 'mode-matched controller gain' for the jth discrete state. This ensures that only one controller is active at a time, and there is no ambiguity while switching controllers. Here the supervisor's discrete state σ_k simply

specifies the controller operating at time k. Further, the protocol governing transitions among controllers is usually implemented in the supervisor as a computer program or logic.

$$\sigma_k = \mathrm{j} \text{ if } \mathcal{G}_{\mathrm{ij}}(x_k) \geq 0 \tag{6.4}$$

where the guards $\mathcal{G}_{\mathrm{ij}}(x_k)$ partition the space $\mathcal{X}$ as

$$\begin{gathered} \bigcup_{\mathrm{i} \in \mathscr{N}} \mathscr{S}_{\mathrm{i}} = \mathcal{X}, \quad \mathscr{S}_{\mathrm{i}} \bigcap_{\mathrm{i,j} \in \mathscr{N}, \mathrm{i} \neq \mathrm{j}} \mathscr{S}_{\mathrm{j}} = \emptyset \\ \text{where } \mathscr{S}_{\mathrm{j}} = \{x \in \mathcal{X} | \mathcal{G}_{\mathrm{ij}}(x) \geq 0\} \end{gathered} \tag{6.5}$$

$\mathscr{N} \times \mathscr{N} \times \mathbb{R}^n$, as shown in Fig. 6.2. Equation (6.3) describes the switching control scheme for the CPS under supervisory control. The discrete nature of the transition logic is captured using the guard condition $\mathcal{G}_{\mathrm{ij}}(x_k)$ between controllers i and j. Such a CPS is shown in Fig. 6.1. The plant in (6.1) together with the supervisor S described by (6.2) forms the target supervised control system. The CPS (Cyber-Physical System) under consideration consists of different components that work together to achieve a specific task. The *physical layer* includes the plants, sensors, and actuators that interact with the physical environment. The *communication layer* comprises the wired or wireless channels through which signals are transmitted between the physical layer and the cyber layer. The *cyber layer* consists of the supervisory logic that governs the physical layer through wireless communication channels. The supervisory logic acts as a controller that takes inputs from sensors, generates control signals, and sends them to the physical layer to control the system's behavior. Fig. 6.1 provides a visual representation of the CPS, illustrating the interactions between the physical, communication, and cyber layers.

The inter-layer boundaries are often more vulnerable to cyberattacks. Consider a scenario where an attacker has the capability to infiltrate the communication-cyber layers of a CPS, as described above. The primary goal of this attacker is to cause the supervisory logic to produce incorrect output, whenever feasible, by injecting an additive attack $\|a_k\| \infty \leq \rho$ into the continuous state x_k being transmitted to the supervisor S. It is important to note that the attacker is not authorized to directly alter the actual control inputs u_k or the plant states

x_k. Instead, the attacker can only inject a fictitious signal that is visible to the supervisor, as illustrated in Fig. 6.1. In other words, the attacker intends to introduce an *allowable undesired injection attack* into the supervisor.

The attacker's goal is to trigger undesired transitions in the supervisor's internal discrete states within a limited attack budget ρ. This means that the attacker aims to inject attacks that do not exceed the budget and can remain undetected for a prolonged period, causing practical disruptions in the system. To capture the attacker's knowledge and behavior, we assume that the attacker has an understanding of the system's vulnerabilities and can use a valid guard condition $\mathcal{G}$ to perform allowable attacks. This implies that the attacker can exploit the system's weaknesses and cause undesired state transitions using a set of conditions that trigger the attack. We acknowledge that the attacker's actions may not always have a negative impact on the system, but they can still be practical disruptions over long periods. Therefore, we assume that the attacker can remain stealthy, allowing them to carry out the attacks without being detected. The budget ρ captures the attacker's knowledge of injecting bounded attacks causing undesired supervisor switching, while remaining stealthy. This need not have a negative impact on the system, but the attacker remains stealthy and can cause practical disruptions over long time windows [114]. We make the following assumptions for modeling the attacker's intentions.

Assumption 1: The attacker has a partial *a priori* knowledge of the supervisor S in the form of the languages and symbols $(\Sigma, \mathcal{X}, \mathcal{U})$. That is, the attacker knows what symbols the supervisor finite automaton can input or output.

Assumption 2: The attacker can make a finite number of queries to the supervisor to construct the map δ.

The assumptions above essentially model the capability of the attacker to either make queries to the supervisor or observe its open loop behavior. This is a reasonable assumption, often observed in spoofing and data injection type attacks, and exploits already present inter-layer vulnerabilities [93].

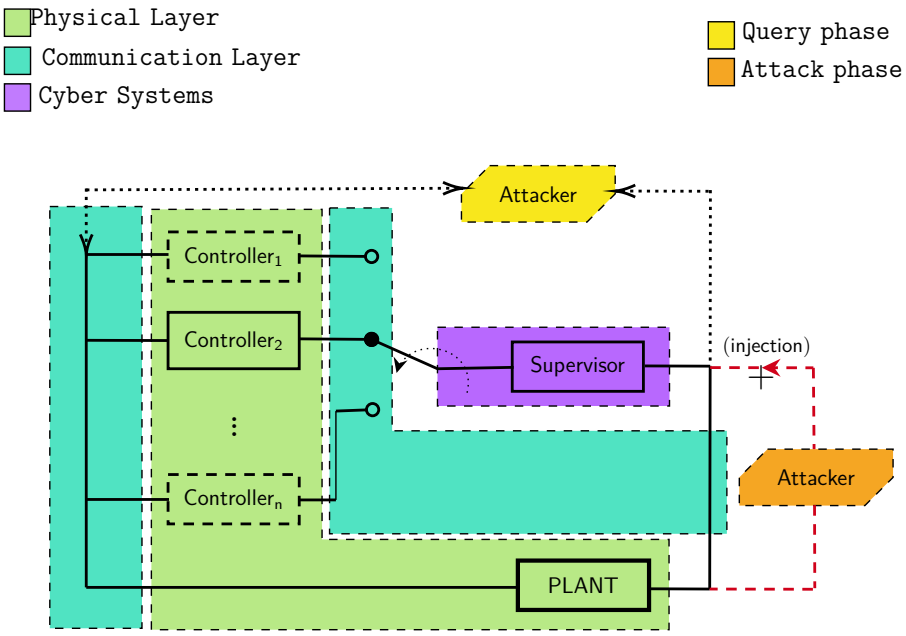

Figure 6.1. Supervisory control system under attack

6.3 Attack & Defense Synthesis

In this section, we will sequentially consider the attack and defense mechanism design with almost no knowledge of the adversary's intent or model. Because of this, we propose a data driven scheme for both cyberattack and defense synthesis.

6.3.1 Cyberattack Synthesis

Assumptions 1 & 2 ensure that the attacker knows no details about the physical layer in the plant model (A, B) itself. These assumptions help us to model the attacker's *a priori* knowledge of the system, and the ability to disrupt the supervisor by data injection attacks. This makes the problem more practical yet challenging, and a data-driven approach more suitable. As seen above, we consider the scenario where the plant and actuator models are

obscured from the attacker. Further, the attacker can only interact with the CPS in a limited manner. The attacker's aim at time k can then be captured as:

$$\begin{aligned} &\text{find } a_k, \|a_k\| \infty \leq \rho \\ &\text{such that } \delta(\sigma_{k-1}, x_k) \neq \delta(\sigma_{k-1}, x_k + a_k) \\ &x_{k+1} = (A + BK_{\sigma_k})x_k \\ &\text{given: } \Sigma, \mathcal{X}, \mathcal{U} \end{aligned} \tag{6.6}$$

for some attack budget $\rho > 0$ for the injected attack a_k. This can be posed an optimization problem, with the following characteristics: (i) due to the realistic case of partial knowledge of supervisor S and no knowledge about the plant, the attacker first needs to estimate the map $\delta(\cdot)$, and (ii) if an unlimited attack budget ρ is available, one can always find an attack that can cause an allowable undesired switching of discrete states in the supervisor. To carry out the norm bounded injection in (6.6), the attacker first infers an approximate model for the supervisor $\hat{S} = (\Sigma, \mathcal{X}, \mathcal{U}, \hat{\delta}, \phi)$. Note that the attacker need not know ϕ, as will be seen later. For this, we propose a data-driven approach to first approximate the supervisor model by estimating the discrete state transition map δ. To this end, the attacker samples $\sigma \in \Sigma, x \in \mathcal{X}$ from the discrete and continuous state spaces, respectively, and performs queries with the open-loop supervisor as $y_{\mathrm{i}} = \delta(\sigma_{\mathrm{i}}, x_{\mathrm{i}})$.

$$y = \delta(\sigma, x) \tag{6.7}$$

The attacker's aim in the query phase is to build the labeled dataset $\mathcal{D} = \{(y_{\mathrm{i}}, x_{\mathrm{i}})\}$, for $\mathrm{i} = 1, \cdots, N$. Following the query phase, the attacker must create an estimated map $\hat{\delta}$ of the unknown discrete state transition map δ based on the collected data $\mathcal{D}$. This results in a supervised $M-$class classification problem on a labeled dataset $\mathcal{D}$, for $|\Sigma| = M$ number of

discrete states. It is known that the cost function for a binary classification problem in the form of the parameterized estimator function $h_\theta(x)$ is given as:

$$\begin{aligned} J(\theta) = -\frac{1}{N}\sum_{\mathrm{i}=1}^{N}\left[y_\mathrm{i}\log\left(h_\theta(x_\mathrm{i})\right) + (1-y_\mathrm{i})\log 1 - (h_\theta(x_\mathrm{i}))\right] \\ + \frac{\lambda}{2N}\left\|\theta\right\|2^2 \end{aligned} \tag{6.8}$$

with a regularization parameter λ. For instance, for a single layered artificial neural network (ANN), the parameterized function could be sigmoidal $h_\theta(x) = \texttt{sigmoid}(\theta^T x)$, linear $\theta^T x$, rectified linear unit, or other activation functions with tensor operations. Due to the universal approximation properties [115], ANNs are good candidate classification methods for our problem, especially because of the inherent discontinuities and Boolean logic in discrete state transition maps [110]. Therefore, for an $M-$class classification over the dataset $\mathcal{D}$, the training problem is to minimize the following cost function:

$$\begin{aligned} J(\boldsymbol{\Theta}) = -\frac{1}{N}\sum_{\mathrm{i}=1}^{N}\sum_{\mathrm{j}=1}^{M}\Big[y_\mathrm{i}^{(\mathrm{j})}\log\left(h_{\boldsymbol{\Theta}}(x_\mathrm{i})\right)^{(\mathrm{j})} \\ +(1-y_\mathrm{i}^{(\mathrm{j})})\log\left(h_{\boldsymbol{\Theta}}(x_\mathrm{i})\right)^{(\mathrm{j})}\Big] + \frac{\lambda}{2N}\sum_{l=1}^{L}\left\|\Theta\right\|2^2 \end{aligned} \tag{6.9}$$

over $x_\mathrm{i}, y_\mathrm{i} \in \mathcal{D}$, for an $L-$layer ANN. The learning problem to minimize (6.9) is solved using stochastic gradient descent. Presenting detailed derivations of ANN based function approximation is not an aim for this work and standard solving procedures can be found in [113]. This yields the estimate $\hat{\delta}$ that minimizes the cost function over seen examples as $\hat{\delta}_{\boldsymbol{\Theta}^*}$ for $\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta})$. Under the proposed scheme, the attacker uses the learned map $\hat{\delta}_{\boldsymbol{\Theta}^*}$ instead of δ in (6.6) as:

$$\begin{aligned} &\text{find } a_k, \left\|a_k\right\|\infty \leq \rho \\ &\text{such that } \hat{\delta}(\sigma_{k-1}, x_k; \boldsymbol{\Theta}^*) \neq \hat{\delta}(\sigma_{k-1}, x_k + a_k; \boldsymbol{\Theta}^*) \\ &\text{given: } \Sigma, \mathcal{X}, \mathcal{U} \end{aligned} \tag{6.10}$$

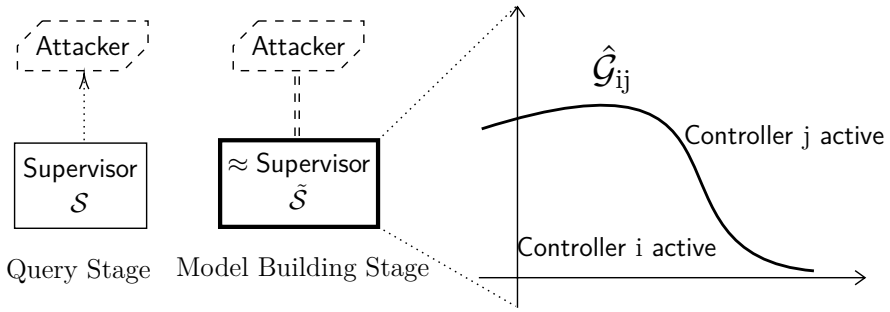

Figure 6.2. Attacker's approximation of the guard conditions

The problem in (6.10) is then the same as misclassifying the output of the trained ANN model $\hat{\delta}$, as shown in Fig. 6.2. Now the attacker has learned a query based approximation $\hat{S}$ of supervisor S. This is often addressed as an *adversarial machine learning* problem, as in [111]. Since $\hat{\delta}$ is the minimizer of the cost function $J(\boldsymbol{\Theta})$, the following holds:

$$\begin{aligned} &\text{if } \hat{\delta}(\sigma_{k-1}, x_k; \boldsymbol{\Theta}^*) \neq \hat{\delta}(\sigma_{k-1}, x_k + a_k; \boldsymbol{\Theta}^*) \\ &\quad \Rightarrow J(x_k; \boldsymbol{\Theta}^*) \leq J(x_k + a_k; \boldsymbol{\Theta}) \end{aligned} \tag{6.11}$$

where $\boldsymbol{\Theta}^*$ is the (possibly non-unique) minimizer of (6.9). Equation (6.11) states that any injection a_k into supervisor input x_k will worsen the optimum cost function. Therefore, once $\hat{\delta}$ is learned, the misclassification problem can be posed as a maximization of this optimum cost function as follows:

$$\begin{aligned} &\underset{a_k}{\text{maximize}}\ J(x_k + a_k; \boldsymbol{\Theta}^*) \\ &\text{subject to } \|a_k\| \infty \leq \rho \end{aligned} \tag{6.12}$$

Note that based on the choice of activation functions and tensor operations to describe $h_{\boldsymbol{\Theta}}(x)$, the gradients $\nabla_x J(\boldsymbol{\Theta}^*)$ are known. Further, calculation of this gradient is performed in the so-called backpropagation step of training the ANN, making the gradients readily available. Using the first order approximation for J, the optimal attack on $\hat{\delta}$ is found to be:

$$a_k^* = \rho \cdot \text{sign}\{\nabla_{x_k} J(\boldsymbol{\Theta}^*)\} \tag{6.13}$$

which is a known result from the fast gradient sign method (FGSM) from [111]. This is gradient descent in the steepest direction along which the cost increases the most, moving towards the decision boundary. This gives the *maximum allowable attack* on the supervisor $\hat{S}$. It follows from (6.2) that the maximum allowable attack is also an undesired attack on $\hat{S}$ if it results in the discrete state transition $\hat{\delta}(\sigma_k, x_k; \boldsymbol{\Theta}^*) \neq \hat{\delta}(\sigma_k, x_k + a_k^*; \boldsymbol{\Theta}^*)$. While a solution to (6.6) may not exist, the maximization problem in (6.12) can always be solved, which comes at the cost that the resulting solution is always a maximum allowable attack, but not necessarily undesired. Further remarks on the space complexity of similar ANN based methods are in Appendix 6.5.1.

6.3.2 Defense Synthesis

A realistic defense mechanism is unaware of attack synthesis strategies in (6.6), or the attack injection mechanisms. As a result, we are required to design a defense mechanism where neither the attack mechanism, nor the upper bounds ρ on injection attacks in (6.10) are known. A successful defense mechanism must have two functions: a) a detection mechanism to predict whether the supervisory control system has been compromised, and b) reconstructing the compromised signal to the supervisor. In the absence of attack synthesis details, we follow a data driven defense scheme to address the required functionality as a generative adversarial network (GAN), shown in Fig. 6.3. Such a GAN based defense method is inspired from similar methods well known in adversarial machine learning [112].

Let $\tilde{x}_k$ denote the perturbed signal to the supervisor under some unknown attack a_k, and $\tilde{\sigma}_k$ the resulting supervisor output. The *detector* $d : \mathcal{X} \to [0, 1]$ is trained to detect the probability of an undesired switching having occurred due to the attack as

$$d(z) = P(\sigma_k \neq \tilde{\sigma}_k; z | \tilde{x}_k) \tag{6.14}$$

for a small number ε. The *reconstructor* $g : \mathcal{X} \times \Sigma \to \mathcal{X}$ is trained to be capable of generating realistic supervisor input by sampling from the latent space $\mathcal{X} \times \Sigma$ (usually sampled as multivariate Gaussian [112]). In implementation, the detector and reconstructor are both neural networks, trained against each other from sampled trajectories of the model

in (6.1) as a GAN. The detector d is then trained as a binary classifier in (6.8) against samples generated by reconstructor g by drawing random samples from the latent space. The standard GAN training is done as a two-person game such that the detector learns to correctly classify generated supervisor input as an attacked input or genuine supervisor input. Simultaneously, the reconstructor attempts to deceive the detector by generating sample supervisor inputs [113]. That is, for a similar cost function as (6.9), for the detector's cost $J(\boldsymbol{\Theta}_d, \boldsymbol{\Theta}_g)$, the reconstructor's payoff is given by $-J(\boldsymbol{\Theta}_d, \boldsymbol{\Theta}_g)$.

An ideally trained detector-reconstructor pair should function in a way that the reconstructor is able to generate perfect samples, and the detector can identify all compromised signals. The goal of training a detector-reconstructor pair in a GAN is to achieve perfect performance where the reconstructor can generate flawless samples and the detector can identify all compromised signals. In a GAN, the trained detector serves as a binary classifier, outputting probabilities from a logistic model (referred to as a logit output). When the detector produces a probability of $P(\sigma_k \neq \tilde{\sigma}_k) \geq 1 - \epsilon$, using $d(\cdot)$ for a small value of ϵ, an attack is detected, and the compromised supervisory input is corrected by the reconstructor, as illustrated in Fig. 6.3. However, if the supervisor input is not detected as being compromised, then it is used as is. This defense mechanism can be trained offline, does not depend on attacker dynamics, and can be adjusted to a specific level of accuracy. It should also be noted that the defense-GAN is integrated upstream of the supervisor in the cyber system layer.

It is important to achieve a balance between the accuracy of the detector and the reconstructor. If the detector is too sensitive, it may produce a high number of false positives and over-correct legitimate signals. Conversely, if the detector is not sensitive enough, it may miss actual attacks, leading to system failure. In the proposed defense mechanism, the reconstructor learns to generate samples that closely resemble the expected signals, which ensures that the correction process is as accurate as possible.

The defense-GAN provides an effective approach to protecting cyber-physical systems from cyberattacks. Its ability to detect attacks without relying on attacker dynamics, its offline training capabilities, and its adjustable accuracy make it a useful tool for safeguarding a wide range of systems.

$$\boldsymbol{\Theta}_d^*, \boldsymbol{\Theta}_g^* = \arg\min_g \max_d J(\boldsymbol{\Theta}_d, \boldsymbol{\Theta}_g) \tag{6.15}$$

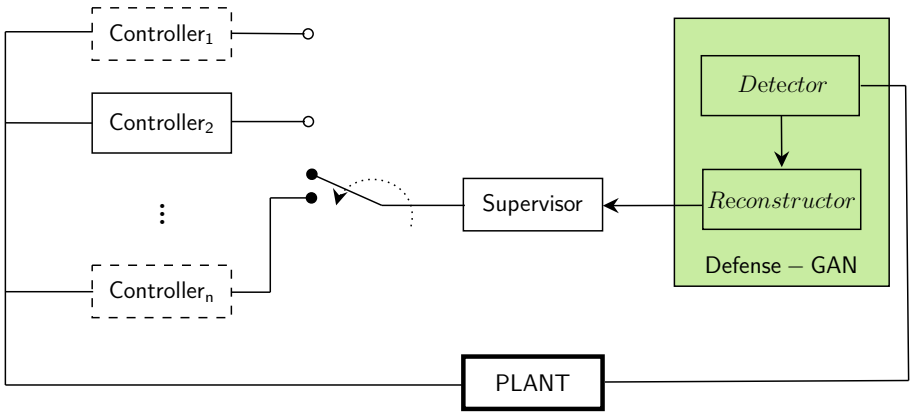

Figure 6.3. A GAN to function as a detector-reconstructor against unknown injection attacks

6.4 Case Study: Formation Control of a network of robots

In this section, we consider a formation control example for a group of nonholonomic robots, adapted from [116]. A network of $N-$nonholonomic robots are required to follow a fixed trajectory, while maintaining a desired formation. The robots have their dynamics in continuous time governed by Dubin's equations given below:

$$\dot{x}_{\mathrm{i}} = v_{\mathrm{i}} \cos\theta_{\mathrm{i}}, \quad \dot{y}_{\mathrm{i}} = v_{\mathrm{i}} \sin\theta_{\mathrm{i}}, \quad \dot{\theta}_{\mathrm{i}} = \omega_{\mathrm{i}} \tag{6.16}$$

Here $(x_{\mathrm{i}}, y_{\mathrm{i}}, \theta_{\mathrm{i}})$ denotes the i^{th} robot's coordinates and $(v_{\mathrm{i}}, \omega_{\mathrm{i}})$ denotes the fixed linear and rotational velocities, respectively, as shown in Fig. 6.4 . Linearization of (6.16) results in the matrices A_{i} in (6.1), for each robot i. Of these, one robot is assigned as the 'leader' with a predefined trajectory and the remaining are 'follower' robots. Based on sensor ranges of each robot, [116] and [117] proposed a bank of state feedback controllers with gains $\left\{K^{\mathrm{i}}_{(1)}, K^{\mathrm{i,j}}_{(2)}, K^{\mathrm{i}}_{(3)}, K^{\mathrm{i,j}}_{(4)}\right\}^{N}_{\mathrm{i,j}=1}$ where subscripts represent the discrete states, and superscripts the active robots. The supervisor's discrete states are $\sigma_k = 1 \equiv$ Separation Bearing Control

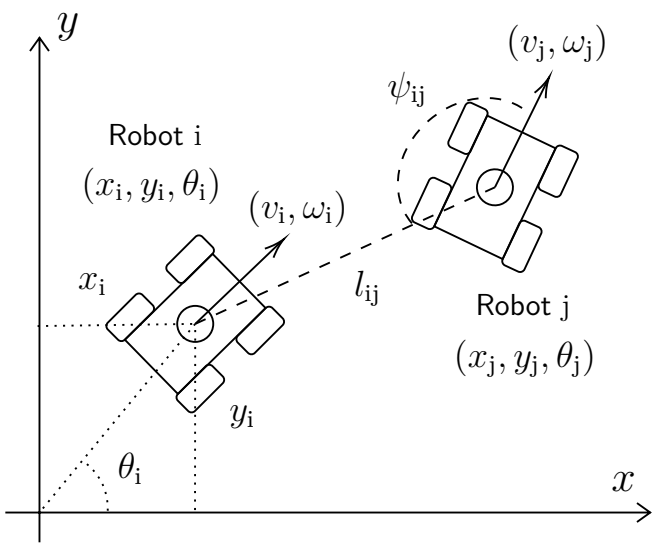

Figure 6.4. Coordinate scheme for formation control

($SB_{ij}C$, robot i follows j), $\sigma_k = 2 \equiv$ Separation Separation Control ($S_{ij}S_{jk}C$, robot k follows i, j), $\sigma_k = 3 \equiv$ Separation Distance-to-Obstacle Control (SD_oC), and $\sigma_k = 4 \equiv$ Autonomous navigation ($u \equiv 0$) as per (6.16). The formation control problem is then to find a switching strategy among candidate controllers such that starting from an initial configuration, an arbitrary desired configuration can be reached. The exponentially stable controllers proposed by [116] are detailed in (6.20). For a 3−robot case where robot 3 follows robots 1 & 2, the guaranteed exponentially stable control switching scheme for arbitrary configurations is given as:

$$
\begin{aligned}
&\text{If } (l_{13} < l_{23}) \wedge (l_{23} > r_1) \wedge (l_{13} < r_2) \text{ then } SB_{13}C, \\
&\quad \sigma_k = 1 \text{ for robot 3} \\
&\text{If } (l_{13} > l_{23}) \wedge (l_{23} > r_1) \wedge (l_{23} < r_2) \text{ then } SB_{23}C, \\
&\quad \sigma_k = 2 \text{ for robot 3} \\
&\text{If } (l_{13} < r_1) \wedge (l_{23} < r_1) \text{ then } S_{13}S_{13}C, \\
&\quad \sigma_k = 3 \text{ for robot 3} \\
&\text{If } (l_{13} > r_2) \wedge (l_{23} > r_2) \text{ then Autonomous,} \\
&\quad \sigma_k = 4 \text{ for robot 3}
\end{aligned}
\tag{6.17}
$$

Here l_{ij} denotes the relative distance between robots i and j, and R_{i} the sensor range for robot i. This gives an analytical form of discrete state transition map δ from (6.2) with implicit partitions of the state space. The differential equations are discretized to get system matrices conforming with the discrete time linear plant in (6.1), and the parameters used in simulation are given in Appendix 6.5.2.

6.4.1 ANN Design

Each robot determines its active controller by sending the relative distance l_{ij} from every other robot. The supervisor then switches to the appropriate controller based on the received information, using (6.17). However, in this demonstration, an attacker aims to inject signals a_k into the state of robot 3. To achieve this goal, the attacker needs to infer the switching logic by first obtaining query data from the system and then solving the optimization problem described in (6.9). To obtain the necessary query data, the attacker creates a sample of l_{ij}'s and communicates with the supervisory logic to record the corresponding values of σ_k. An artificial neural network (ANN) is then employed, where the cost function in (6.12) is the least squared error between the assigned and estimated controller states. It should be noted that in reality, the architecture complexity n_p and the number of data points N available to the attacker are interdependent, as shown in (6.19) in Appendix 6.5.1. Therefore, the attacker must carefully consider the architecture of the ANN and the amount of data required to successfully infer the switching logic of the system.

Overall, the presented approach highlights the potential vulnerability of cyber-physical systems to attacks and emphasizes the importance of developing robust defense mechanisms to protect against such threats. The findings of this study can be valuable in various domains, including robotics, manufacturing, and transportation systems, where the security and reliability of cyber-physical systems are essential.

After collecting 500 data points, the attacker deduces a discrete state switching map $\hat{\delta}$ for the switching regulated by (6.17). The details of the ANN's parameters are provided in Appendix 6.5.2. Subsequently, the attacker gains access to the communication channel of robot 3's queries to the supervisor and injects a_k^*, which is calculated using (6.13). Robot 3

was arbitrarily selected because of the symmetry of controllers in (6.17) around the leader robot. The backpropagation step during the ANN training provides the gradients of the mean squared error cost function.

Figure 6.6 displays the trajectories of the robots, which were initially and finally arranged according to the formation specified in [116]. To cause the formation to deteriorate, the attacker uses adversarial switching tactics based on the guard conditions outlined in (6.17) with a value of $\rho = 0.375$. The decision boundaries of the artificial neural network used in the attack are determined through interactions with the supervisor. The decision boundaries and the guard conditions are illustrated side by side in Fig. 6.5. The cyber attacker has the

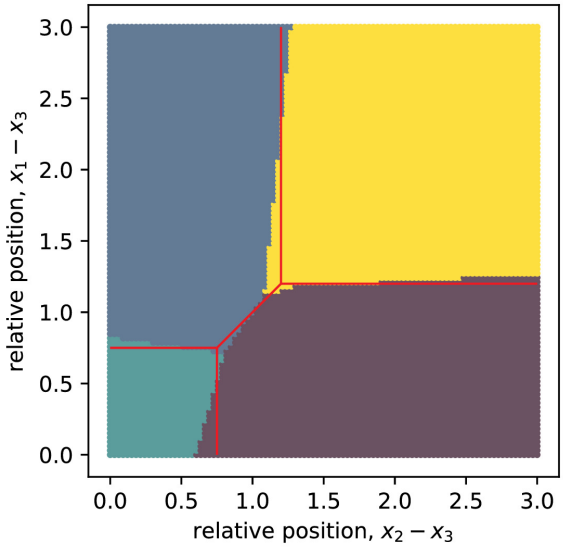

Figure 6.5. Analytical guard conditions (in red) compared with guard conditions estimated by the ANN (in color)

capability to cause undesired behavior not only in the evolution of σ_k over time, but also in the trajectory of the targeted robot, leading to a deviation from the nominal trajectory as depicted in Fig. 6.6. Robot 3, highlighted in red, is particularly affected by the attack and deviates from the desired formation, failing to recover its trajectory during the attack period. Meanwhile, robots 1 & 2 continue to follow the nominal trajectories.

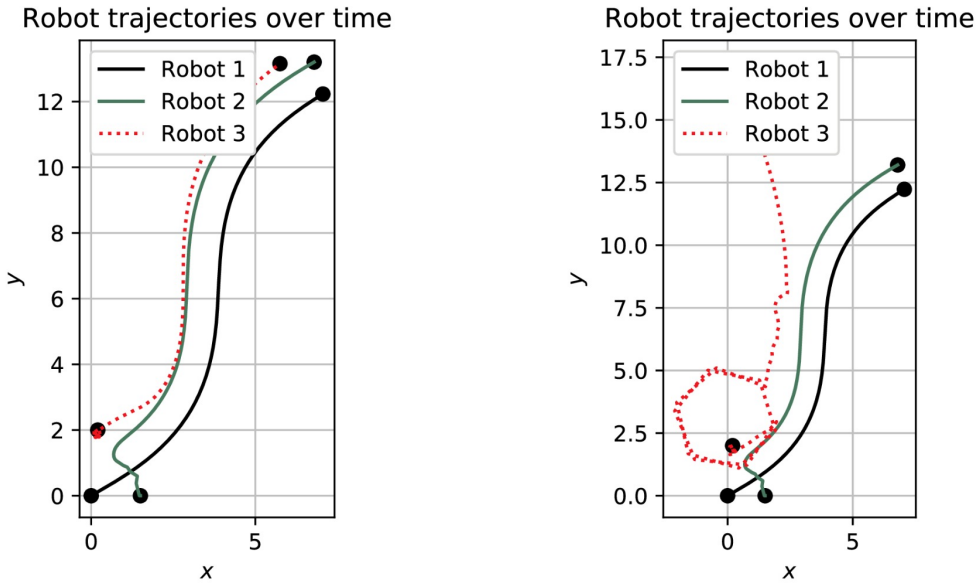

Figure 6.6. *Left:* Trajectories of the formation when using defense-GAN; *Right:* Trajectories when robot 3 is subject to attacks, with no defense mechanism

6.4.2 Defense-GAN Design & Results

The article presents a new technique for enhancing the security of CPSs using a defense-generative adversarial network (defense-GAN). The defense-GAN comprises two ANNs – a detector and a reconstructor, that work together to detect and mitigate cyberattacks on the system. The detector ANN has a single output unit and takes input that is compatible with the output of the reconstructor. The reconstructor ANN generates a candidate trajectory, and the detector ANN classifies the given trajectory points as anomalous or not. Both ANNs are designed with 4 hidden layers and use 'leaky rectified linear units' as activation functions at each layer. The defense-GAN is trained over 10,000 epochs, during which the reconstructor ANN generates trajectories that the detector ANN classifies as anomalous or not. Through the training process, the reconstructor learns to generate tighter clusters of data in feature space, which correspond to discrete state-specific clusters in the latent space.

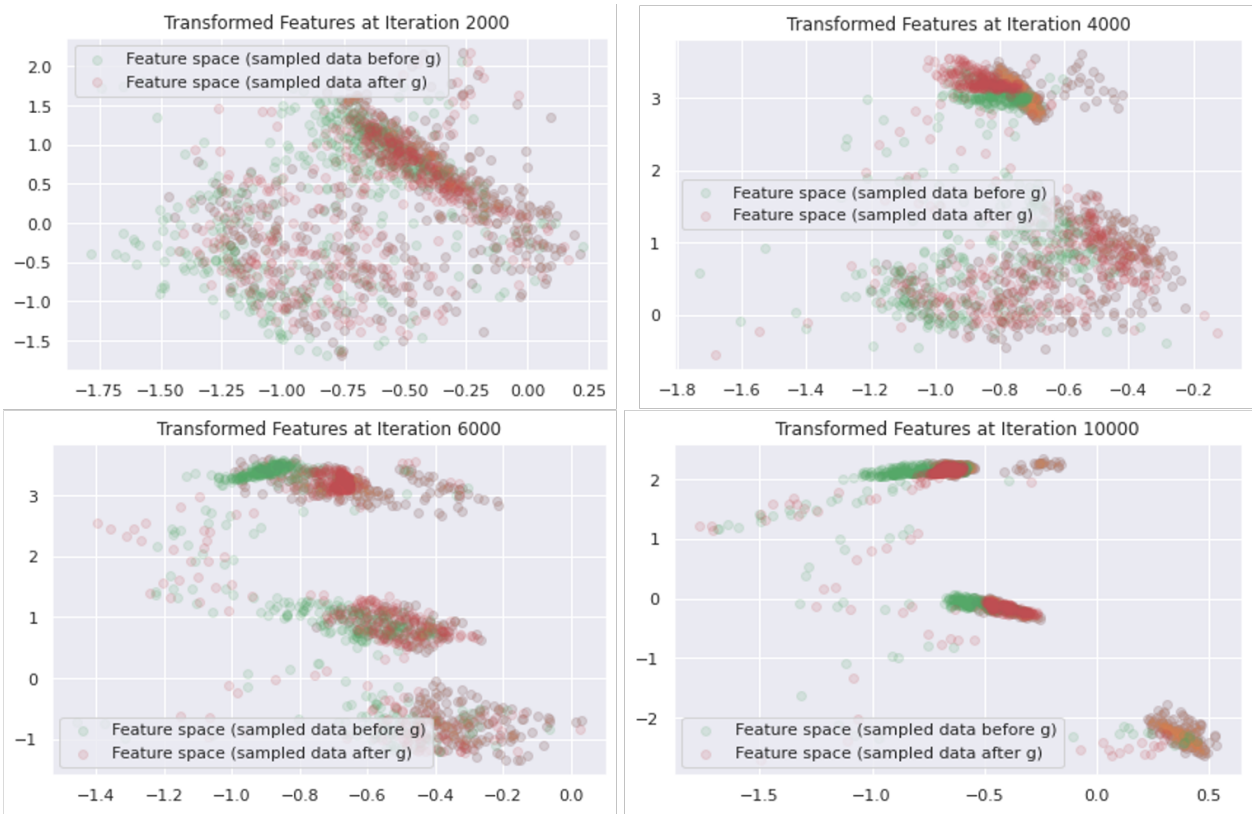

Figure 6.7. Training procedure of the reconstructor visualized (in an intermediate feature space)

The purpose of this training is to ensure that the defense-GAN can identify and mitigate cyberattacks on the CPS.

Figure 6.7 illustrates the performance of the GAN during the training process. The results show that the GAN performance increases slowly over time, as evidenced by the emergence of cohesive clusters of generated data in the figure. This indicates that the defense-GAN is able to generate trajectories that closely resemble the true trajectories and can accurately detect anomalous data points.

Overall, the proposed design and training of the defense-GAN demonstrates the potential of using machine learning techniques to improve the security of cyber-physical systems against cyberattacks. The paper's findings can be valuable in various domains, including industrial automation, transportation systems, and healthcare, where the security and resilience of cyber-physical systems are critical.

The attacks based on gradient result in σ_k from (6.6), which lead to undesired switching. As shown in Fig. 6.8, the nominal trajectory of robot 3 during this simulation spends most of the time dwelling in $\sigma_k \equiv SB_{23}C$. Nonetheless, the gradient based attack can inject $\|a_k\| \infty \leq 0.375$, which causes the trajectory points lying close to the guard condition between the blue and yellow regions in Fig. 6.5 to move from one region to the other. As a result, the supervisor's discrete state fluctuates between $\sigma_k = 2$ and $\sigma_k = 4$, as seen in Fig. 6.8. However, the defense-GAN effectively mitigates these attacks, leaving the discrete states almost unaltered. The proposed method shows that despite each individual controller mode

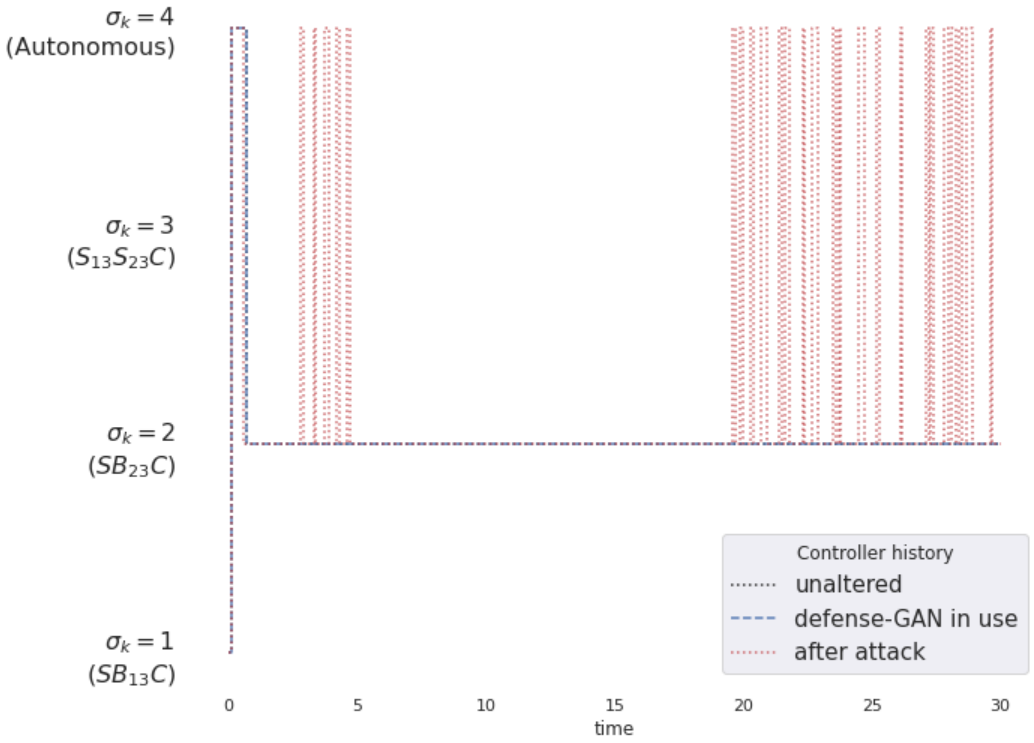

Figure 6.8. Controller switching induced by the attacker

σ_k being stable and the supervisory controller being designed to be exponentially stable, an attacker can still destabilize the formation within the given attack budget. The attacker can achieve this by inducing allowable undesired discrete state transitions. These state switchings can be detected along the shown trajectory when using the defense-GAN. The method further emphasizes the importance of assessing the performance of defense mechanisms by using appropriate metrics. In this regard, the paper proposes a more reliable metric for evaluating

the performance of defense-GAN, which is its cyberattack detection accuracy for a large number of trajectories. In particular, the paper evaluates the performance of the defense-GAN by running 10,000 trajectories. The results show that the defense-GAN was able to detect cyberattacks with an accuracy of 97.6%, while the remaining 2.4% accounts for both Type-I and Type-II errors.

It's important to note that in the absence of true discrete state, the occurrences of Type-I and II errors are reported combined. This highlights the need for using appropriate metrics that take into account the limitations and uncertainties in the system, as well as the potential trade-offs between accuracy and false positives or false negatives. By using appropriate metrics, researchers can better evaluate the effectiveness of defense mechanisms and identify areas for further improvement.

6.5 Conclusion

This chapter introduces a novel approach to designing cyberattacks and defense mechanisms for supervisory controllers in cyber-physical systems (CPSs) using machine learning techniques. The proposed method leverages adversarial machine learning methodologies and the numerical properties of neural networks to enhance the system's security against cyber threats. To illustrate the effectiveness of the proposed approach, the paper presents a practical example of a multi-robot system's formation control, in which an attacker induces allowable undesired discrete state transitions in an exponentially stable supervisory controller. The proposed data-driven method is utilized to carry out this attack, and a defensive generative adversarial network is designed to identify and mitigate these attacks. The results demonstrate the effectiveness of the proposed method in enhancing the system's resilience against cyberattacks. The method improves the system's security by utilizing machine learning techniques to identify and prevent cyberattacks. By leveraging adversarial machine learning and neural network properties, the proposed method can enhance the system's resilience to cyber threats in CPSs. The practical example provides evidence of the method's effectiveness and demonstrates its potential to be applied to other cyber-physical systems.

As a future research direction, the authors plan to implement the proposed attack-defense scenario to an unmanned robotic system. This will help to evaluate the real-world applicability and effectiveness of the proposed method in a practical setting. The proposed method has the potential to provide a significant contribution to the development of secure and resilient cyber-physical systems, which are increasingly important in various applications, such as industrial automation, transportation systems, and healthcare.

6.5.1 Space complexity of ANNs

The space complexity of an approximation scheme is the number of data points required to achieve a probabilistic estimation error of $1-\alpha$ (where α is the misclassification probability). This is studied as *Probably asymptotically correct*-ness (PAC). based on which, the following remarks can be made about the space complexity of the ANN based method.

Remark 9. *For an ANN with n_p units, d data points, and k input dimension, the order of data points for a specified PAC accuracy (due to [118]) is defined as follows. The estimation error with respect to the Bayes regressor is given by the following due to [118]:*

$$P\left(\left\|\hat{\delta}_{Bayes}-\hat{\delta}\right\| 2^2 \leq 1-\alpha\right) \leq \mathcal{O}\left(\frac{1}{n_p}\right)+ \mathcal{O}\left(\sqrt{\frac{n_p k \log n_p d-\log \alpha}{d}}\right) \tag{6.18}$$

$$n_p(N) \sim \mathcal{O}\left(\left[\frac{4N}{k \log(n_p^*(N)\cdot N)}\right]^{1/3}\right) \tag{6.19}$$

This provides guidelines for the order of number of data points for a specified PAC accuracy.

6.5.2 Simulation Parameters

The individual controllers for different modes $\sigma_k \in \{1, \cdots, 4\}$ for robot 3, as devised by [116], are given as:

$$\begin{aligned}
&\text{If } \sigma_k = \mathrm{i} \in \{1, 2\}, \\
&\dot{l}_{\mathrm{i}3} = k_1(l^d_{\mathrm{i}3} - l_{\mathrm{i}3}), \dot{\psi}_{\mathrm{i}3} = k_2(\psi^d_{\mathrm{i}3} - \psi_{\mathrm{i}3}), \dot{\theta}_3 = \omega_3 \\
&\text{If } \sigma_k = 3, \\
&\dot{l}_{13} = k_1(l^d_{13} - l_{13}), \dot{l}_{23} = k_1(l^d_{23} - l_{23}), \dot{\theta}_3 = \omega_3 \\
&\text{If } \sigma_k = 4, \text{ Eq. (6.16)}
\end{aligned} \tag{6.20}$$

The simulation parameters are given in the table below.

Table 6.1. Multi-robot parameters for simulation on supervisory controllers subject to cyberattacks

Separation gain	k_1	1.5
Separation angle gain	k_2	1.5
Sensor ranges	(R_1, R_2, R_3)	$(0.75, 1.2, 2)$
Initial positions	robot 1 (leader)	$(0, 0, 30°)$
	robot 2 (follower)	$(1.5, 0, 0°)$
	robot 3 (follower)	$(0.2, 2, 30°)$
Initial velocities	robot 1 (leader)	$\omega_1 = 0.1 \sin 0.2t$, $v_1 = 0.5$
Desired formation	separating distance	$l^d_{12} = l^d_{13} = l^d_{23} = 1$
	separating angle	$\psi^d_{12} = 90°$

The ANNs designed in Section 6.4 have the following parameters.

Table 6.2. NN parameters for simulation on supervisory controllers subject to cyberattacks

Number of epochs, batch size	100, 5 (attacker)
Optimizer	Adam
Loss function	mean squared error
Number of hidden layers	1
Hidden layer activation	softmax
Dropout rate	0.10
Trainable parameters	66
Query distribution (to collect $\mathcal{D}$)	$\mathcal{U}_{[0,3]\times[0,3]}$
No. of epochs, batch size	10000, 512 (for defense GAN)
No. of hidden layers (both g & d)	4
Hidden layer activation	leaky ReLU

7. CONCLUSIONS AND RECOMMENDATIONS

This dissertation investigated data-driven safety and security methods for complex cyber-physical systems (CPSs), where complexity is imparted to the systems via unknown non-linearities (Chapters 2 & 4), usage of data-driven methods-in-the-loop (Chapter 4), complex multi-agent interactions over time-varying networks (Chapters 3 & 5), and supervisory/switched control systems with distinct cyber and physical layers (Chapter 6).

7.1 Research Contributions

As per the multi-fold objectives outlined in Chapter 1, the research contributions follow appropriately. Set properties are generally more stronger properties than point-wise, or even trajectory-wise properties for control systems. Of these, safety and security are of particular importance, especially since more and more control systems employ physically separated entities interacting over insecure communication networks. This is worsened by inter-agent interactions of complex systems resulting in emergent points of failures in safety, or points of cyberphysical vulnerabilities. This dissertation aims to perform a systematic research into uncovering challenges relating to these set properties, and points of failures of related operations of complex systems resulting from such an emergence.

To this end, the first point of contribution is to assist in rapid computation of reachable sets, essential for determining safety properties. While numerical methods exist that compute such set properties, the method proposed in Chapter 2 accomplishes the same with an advantage of real-time applicability, while handling totally unknown system dynamics, at the expense of the disadvantage of loose over-approximations. Such a method would be more applicable when rapid reachable sets are to be computed and loose over approximations are acceptable (e.g., initial rapid path planning). Chapter 3 considers similar unknown dynamical systems, that have neural networks-in-the-loop. The contributions of this chapter are as follows. First, computing approximate reachable sets for NNs in real-time is very much an open problem, and the results can be far reaching as NNs are used in all parts of control design lately (from estimation, to system identification to control synthesis). This provides us with a systematic way to compute bounds on NN outputs in real-time, while respecting

system's dynamical data. The dynamics uncovered for these systems are used to obtain a hybrid model+data-driven approach compute reachable sets for NN-in-the-loop systems. As hard model-based control gets more and more elusive these days due to the presence of widely unknown models, and agents being expected to collaborate with unknown entities, the methods proposed in Chapter 3 pose wide utilities for practical problems and control prototyping under highly nonlinear and uncertain environments, especially when machine learning models are being employed in conjunction with modern control.

This dissertation also looks at the problem of safety for multi-agent systems. Related reachable set computation can be performed on a central computational node, however, for N number of n-dimensional dynamical systems, the computational cost grows exponentially with $n \times N$ due to the associated curse of dimensionality in the Hamilton-Jacobi equations. To alleviate this, we propose a distributed computational strategy where the reachable sets of the agents are affected by each other in unknown ways, but inter-agent communication can be utilized to compute the reachable sets using distributed optimization. Obviously, the related disadvantages from distributed optimization are inherited in the schemes in Chapter 4. Convergence is slower in sparse networks, and uniform-joint-connectivity is to be maintained at all times for the solutions to be consistent. In reality, these assumptions can get violated, especially if subject to malicious entities.

To this end, we dealt with the dual problem of uncovering vulnerabilities of complex CPSs subject to malicious intent, under realistic scenarios of underlying CPS models being completely obscured from the attacker, except for the ability to interact with the CPS physical layer in an open loop manner. Such assumptions are very realistic as many real life cyberattacks against drones are prepared/trained on relatively inexpensively and readily available off-the drones. We utilized such open loop interactions to determine equivalent machine learning models for supervisory control systems in Chapter 6, and used Koopman theory combined with semi-definite programming to find the equivalent systems in Chapter 5. The equivalent (or learned) models are sued by attackers to perform attacks with limited budget, to remain undetected against naive detection schemes (e.g., threshold based detection), while causing degradation in system performance (in Chapter 6), and violating system safety (in Chapter 5).

7.2 Recommendations for Future Work

Based on the above summarizing discussion, the following points of recommendations and further investigation can follow.

1. Computation of flight envelopes of quadrotors and UAVs in real-time is a challenging problem. The methods discussed in Chapters 2 & 3 can effectively be utilized in combination to result in a “hybrid reachable set computation" scheme. Such a practical scheme should utilize loose approximations (when computation speed is desired), and switch to slower, more accurate method (when operating close to flight envelope, or when safety is desired).

2. For the methods in Chapters 3 and 5, convergence of the methods discussed over stochastic networks is an open problem. Techniques from distributed optimization over random graphs must be utilized to extend the results obtained in the Chapters above to hold over random communication networks.

3. For over/under-approximate reachable set computation methods, developing associated approximation metrics (e.g., volume of difference between approximate set and true reachable set) must be computed, and in a recursive manner. Works from Kurzhansky et.al. in [71] are an excellent starting point for certifying approximation level of such reachable sets. The methods proposed in Chapters 2 & 3 are ripe for augmentation with such approximation certifying schemes, or metrics of approximation.

4. In this work, numerous nonlinearities in dynamics were handled while computing reachable sets with unknown dynamics. However, discrete event systems (DESs) or hybrid dynamical systems are a more general class of nonlinear dynamical systems that are found everywhere from aerospace to electrical and software systems. Extension of rapid, over-approximate reachable set computation methods to such nonlinearities would enhance the applicability of this work.

5. Hybrid cyber attacks were eluded to in Chapters 5 & 6, however, the scope was limited to FDI and DoS attacks only. Smarter attackers can perform stealthy cyber attacks against unknown systems (while being able to interact with their open-loop copies)

6. Computing real-time capabilities of UAVs is very important for UAM operating environments and practical applications of full autonomy in UAS control systems. The proposed reachability methods can be utilized to compute system capabilities and for real-time health monitoring of autonomous UASs. On-board applications should be worth consider and investigation, but was beyond the scope of this work.

7. For the hybrid attack schemes in Chapter 5, the defenders can perform evasive maneuvers by utilizing the distributed reachable set computation schemes in Chapter 3. A full attacker-defender analysis can thus be analyzed from the combination of the strategies above, and be used to determine, a priori, the range of all attack budgets which result in system safety being violated. This is a stronger property, and thus we believe could be of utility even if computed offline.

8. The mixed monotone schemes from Chapter 2 can be extended to hybrid dynamical EDMD systems, where guard conditions themselves are also polynomial manifolds. A two-level system identification can be performed to identify the resulting polynomial hybrid system with polynomial guard conditions.

REFERENCES

[1] F. Dyson, "Birds and frogs," *Notices of the AMS*, vol. 56, no. 2, pp. 212–223, 2009.

[2] O. Thapliyal and I. Hwang, "Approximate reachability for koopman systems using mixed monotonicity," *IEEE Access*, vol. 10, pp. 84 754–84 760, 2022.

[3] J. Lygeros, "On reachability and minimum cost optimal control," *Automatica*, vol. 40, no. 6, pp. 917–927, 2004.

[4] W. Xiang, H.-D. Tran, J. A. Rosenfeld, and T. T. Johnson, "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 1574–1579.

[5] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 1758–1765.

[6] C. Kwon and I. Hwang, "Reachability analysis for safety assurance of cyber-physical systems against cyber attacks," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2272–2279, 2017.

[7] X. Feng, M. E. Villanueva, and B. Houska, "Backward-forward reachable set splitting for state-constrained differential games," *Automatica*, vol. 111, p. 108 602, 2020.

[8] J. Ji, G. K. Yang, S. Fu, and C. Pan, "Application of reachability analysis based on time-dependent hamilton-jacobi-isaacs equations," in *IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012)*, IET, 2012, pp. 1–4.

[9] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2242–2253.

[10] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[11] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 15.

[12] M. Chen, Q. Tam, S. C. Livingston, and M. Pavone, "Signal temporal logic meets hamilton-jacobi reachability: Connections and applications," in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[13] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[14] S. Kong, S. Gao, W. Chen, and E. Clarke, "Dreach: δ-reachability analysis for hybrid systems," in *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, Springer, 2015, pp. 200–205.

[15] S. Lee and I. Hwang, "Reachable set computation for spacecraft relative motion with energy-limited low-thrust," *Aerospace Science and Technology*, vol. 77, pp. 180–188, 2018.

[16] S. Coogan, "Mixed monotonicity for reachability and safety in dynamical systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 5074–5085.

[17] I. M. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by hamilton-jacobi projections," *journal of Scientific Computing*, vol. 19, no. 1, pp. 323–346, 2003.

[18] A. Matviychuk and O. Matviychuk, "A method of approximate reachable set construction on the plane for a bilinear control system with uncertainty," in *AIP Conference Proceedings*, AIP Publishing LLC, vol. 2025, 2018, p. 100 004.

[19] Z. Zuo, D. Ho, and Y. Wang, "[brief paper] reachable set estimation for linear systems in the presence of both discrete and distributed delays," *IET control theory & applications*, vol. 5, no. 15, pp. 1808–1812, 2011.

[20] J. Zhao, "Algebraic criteria for reachable set estimation of delayed memristive neural networks," *IET Control Theory & Applications*, vol. 13, no. 11, pp. 1736–1743, 2019.

[21] R. Hu, L. Shao, and Y. Cong, "Polyhedral approximation method for reachable sets of linear delay systems," *IET Control Theory & Applications*, vol. 14, no. 12, pp. 1548–1556, 2020.

[22] P.-J. Meyer, S. Coogan, and M. Arcak, "Sampled-data reachability analysis using sensitivity and mixed-monotonicity," *IEEE control systems letters*, vol. 2, no. 4, pp. 761–766, 2018.

[23] A. Mauroy, I. Mezi, and Y. Susuki, *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer Nature, 2020, vol. 484.

[24] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[25] A. J. Kurdila and P. Bobade, "Koopman theory and linear approximation spaces," *arXiv preprint arXiv:1811.10809*, 2018.

[26] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[27] M. Abate and S. Coogan, "Computing robustly forward invariant sets for mixed-monotone systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 4553–4559.

[28] M. Abate, M. Dutreix, and S. Coogan, "Tight decomposition functions for continuous-time mixed-monotone systems with disturbances," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 139–144, 2020.

[29] F. Immler, M. Althoff, X. Chen, *et al.*, "Arch-comp18 category report: Continuous and hybrid systems with nonlinear dynamics," in *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018.

[30] H. Ren, Y. Wang, M. Liu, and H. Li, "An optimal estimation framework of multi-agent systems with random transport protocol," *IEEE Transactions on Signal Processing*, 2022.

[31] J. M. Kim, J. B. Park, and Y. H. Choi, "Leaderless and leader-following consensus for heterogeneous multi-agent systems with random link failures," *IET Control Theory & Applications*, vol. 8, no. 1, pp. 51–60, 2014.

[32] O. Thapliyal and I. Hwang, "Approximating reachable sets for neural network based models in real-time via optimal control," *arXiv preprint arXiv:2211.03732*, 2022. arXiv: 2211.03732 [eess.SY].

[33] H. Ahn, K. Berntorp, P. Inani, A. J. Ram, and S. Di Cairano, "Reachability-based decision-making for autonomous driving: Theory and experiments," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 1907–1921, 2020.

[34] I. Hwang, D. M. Stipanovi, and C. J. Tomlin, "Polytopic approximations of reachable sets applied to linear dynamic games and a class of nonlinear systems," in *Advances in control, communication networks, and transportation systems*, Springer, 2005, pp. 3–19.

[35] P. Varaiya, "Reach set computation using optimal control," in *Verification of Digital and Hybrid Systems*, Springer, 2000, pp. 323–331.

[36] J. A. dit Sandretto and J. Wan, "Reachability analysis of nonlinear odes using polytopic based validated runge-kutta," in *International Conference on Reachability Problems*, Springer, 2018, pp. 1–14.

[37] J. Gayek, "A survey of techniques for approximating reachable and controllable sets," in *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, IEEE, 1991, pp. 1724–1729.

[38] F. Torrisi and A. Bemporad, "Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 235–249, 2004. DOI: 10.1109/TCST.2004.824309.

[39] M. Althoff, D. Grebenyuk, and N. Kochdumper, "Implementation of taylor models in cora 2018," in *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018.

[40] D. Romeres, M. Zorzi, R. Camoriano, S. Traversaro, and A. Chiuso, "Derivative-free online learning of inverse dynamics models," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 816–830, 2019.

[41] O. Thapliyal and I. Hwang, "Learning based cyberattack design and defense for supervisory control systems," in *2021 European Control Conference (ECC)*, 2021.

[42] D. Huang, W. Yang, T. Huang, N. Qin, Y. Chen, and Y. Tan, "Iterative learning operation control of high-speed trains with adhesion dynamics," *IEEE Transactions on Control Systems Technology*, 2021.

[43] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 4653–4660.

[44] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 281–304, 2019.

[45] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[46] A. Garg, K. Tai, and B. Panda, "System identification: Survey on modeling methods and models," in *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, Springer, 2017, pp. 607–615.

[47] C. Folkestad and J. W. Burdick, "Koopman nmpc: Koopman-based learning and nonlinear model predictive control of control-affine systems," *arXiv preprint arXiv:2105.08036*, 2021.

[48] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 1890–1895.

[49] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*, Springer, 2018, pp. 121–138.

[50] W. Xiang and T. T. Johnson, "Reachability analysis and safety verification for neural network control systems," *arXiv preprint arXiv:1805.09944*, 2018.

[51] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "Reachnn: Reachability analysis of neural-network controlled systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–22, 2019.

[52] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, 2017.

[53] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[54] I. Mezic, "On numerical approximations of the koopman operator," *arXiv preprint arXiv:2009.05883*, 2020.

[55] M. Ravasi and I. Vasconcelos, "Pylopsa linear-operator python library for scalable algebra and optimization," *SoftwareX*, vol. 11, p. 100 361, 2020.

[56] C. Sanderson *et al.*, "Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments," 2010.

[57] M. Bergamasco and M. Lovera, "Identification of linear models for the dynamics of a hovering quadrotor," *IEEE transactions on control systems technology*, vol. 22, no. 5, pp. 1696–1707, 2014.

[58] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, IEEE, 2016, pp. 498–503.

[59] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," *arXiv preprint arXiv:1801.01236*, 2018.

[60] M. Abate and S. Coogan, "Computing robustly forward invariant sets for mixed-monotone systems," *IEEE Transactions on Automatic Control*, 2022.

[61] F. Chen, W. Ren, *et al.*, "On the control of multi-agent systems: A survey," *Foundations and Trendső in Systems and Control*, vol. 6, no. 4, pp. 339–499, 2019.

[62] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[63] L. Iocchi, D. Nardi, and M. Salerno, "Reactivity and deliberation: A survey on multi-robot systems," in *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, Springer, 2000, pp. 9–32.

[64] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[65] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 1859–1864.

[66] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *The Knowledge engineering review*, vol. 19, no. 4, pp. 281–316, 2004.

[67] K. H. Goodrich and C. R. Theodore, "Description of the nasa urban air mobility maturity level (uml) scale," in *AIAA Scitech 2021 Forum*, 2021, p. 1627.

[68] O. Cokorilo, "Urban air mobility: Safety challenges," *Transportation research procedia*, vol. 45, pp. 21–29, 2020.

[69] S. Bharadwaj, S. Carr, N. Neogi, H. Poonawala, A. B. Chueca, and U. Topcu, "Traffic management for urban air mobility," in *NASA Formal Methods Symposium*, Springer, 2019, pp. 71–87.

[70] R. Rothfeld, A. Straubinger, M. Fu, C. Al Haddad, and C. Antoniou, "Urban air mobility," in *Demand for Emerging Transportation Systems*, Elsevier, 2020, pp. 267–284.

[71] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *International workshop on hybrid systems: Computation and control*, Springer, 2000, pp. 202–214.

[72] O. Thapliyal and I. Hwang, "Approximating reachable sets for neural network based models in real-time via optimal control," *IEEE transactions on control systems technology*, forthcoming.

[73] M. Fauré, J. Cieslak, D. Henry, A. Verhaegen, and F. Ankersen, "A survey on reachable set techniques for fault recoverability assessment," *IFAC-PapersOnLine*, vol. 55, no. 6, pp. 272–277, 2022.

[74] N. Kariotoglou, D. M. Raimondo, S. J. Summers, and J. Lygeros, "Multi-agent autonomous surveillance: A framework based on stochastic reachability and hierarchical task allocation," *Journal of dynamic systems, measurement, and control*, vol. 137, no. 3, p. 031 008, 2015.

[75] X. Wang, K. Leung, and M. Pavone, "Infusing reachability-based safety into planning and control for multi-agent interactions," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 6252–6259.

[76] X. Wang, J. Peng, S. Li, and B. Li, "Formal reachability analysis for multi-agent reinforcement learning systems," *IEEE Access*, vol. 9, pp. 45 812–45 821, 2021.

[77] M. M. Share Pasand and M. Montazeri, "L-step reachability and observability of networked control systems with bandwidth limitations: Feasible lower bounds on communication periods," *Asian Journal of Control*, vol. 19, no. 4, pp. 1620–1629, 2017.

[78] B. Tian, J. Wang, W. Huang, and Y. Huang, "Reachable set estimation of multi-agent systems under semi-markov switching topology with partially unknown rates," in *2021 40th Chinese Control Conference (CCC)*, IEEE, 2021, pp. 5148–5153.

[79] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[80] O. Thapliyal and I. Hwang, "Data-driven cyberattack synthesis against network control systems," *arXiv preprint arXiv:2211.05203*, 2022.

[81] A. Sargolzaei, A. Abbaspour, M. A. Al Faruque, A. Salah Eddin, and K. Yen, "Security challenges of networked control systems," in *Sustainable interdependent networks*, Springer, 2018, pp. 77–95.

[82] R. A. Gupta and M.-Y. Chow, "Networked control system: Overview and research trends," *IEEE transactions on industrial electronics*, vol. 57, no. 7, pp. 2527–2535, 2009.

[83] T. C. Yang, "Networked control system: A brief survey," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 4, pp. 403–412, 2006.

[84] O. Thapliyal and I. Hwang, "Learning based cyberattack design and defense for supervisory control systems," in *2021 European Control Conference (ECC)*, IEEE, 2021, pp. 144–149.

[85] K. N. Kumar, C. Vishnu, R. Mitra, and C. K. Mohan, "Black-box adversarial attacks in autonomous vehicle technology," in *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, IEEE, 2020, pp. 1–7.

[86] J. Kalin, M. Ciolino, D. Noever, and G. Dozier, "Black box to white box: Discover model characteristics based on strategic probing," in *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, IEEE, 2020, pp. 60–63.

[87] S. Zhang, X. Xie, and Y. Xu, "A brute-force black-box method to attack machine learning-based systems in cybersecurity," *IEEE Access*, vol. 8, pp. 128 250–128 263, 2020.

[88] Y. Li, D. Shi, and T. Chen, "False data injection attacks on networked control systems: A stackelberg game analysis," *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3503–3509, 2018.

[89] T. Yang, C. Murguia, and C. Lv, "Risk assessment for connected vehicles under stealthy attacks on vehicle-to-vehicle networks," *arXiv preprint arXiv:2109.01553*, 2021.

[90] A. Sargolzaei, K. Yazdani, A. Abbaspour, C. D. Crane III, and W. E. Dixon, "Detection and mitigation of false data injection attacks in networked control systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4281–4292, 2019.

[91] Y. Miao, C. Chen, L. Pan, Q.-L. Han, J. Zhang, and Y. Xiang, "Machine learning–based cyber attacks targeting on controlled information: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–36, 2021.

[92] R. Liu, H. M. Mustafa, Z. Nie, and A. K. Srivastava, "Reachability-based false data injection attacks and defence mechanisms for cyberpower system," *Energies*, vol. 15, no. 5, p. 1754, 2022.

[93] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674–1683, 2018.

[94] S. L. Brunton, M. Budii, E. Kaiser, and J. N. Kutz, "Modern koopman theory for dynamical systems," *arXiv preprint arXiv:2102.12086*, 2021.

[95] R. Deshmukh, O. Thapliyal, C. Kwon, and I. Hwang, "Distributed state estimation for a stochastic linear hybrid system over a sensor network," *IET Control Theory & Applications*, vol. 12, no. 10, pp. 1456–1464, 2018.

[96] O. Thapliyal, J. S. Nandiganahalli, and I. Hwang, "Optimal state estimation for lti systems with imperfect observations," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2795–2800.

[97] S. G. Clarke, O. Thapliyal, S. Hwang, and I. Hwang, "Attack-resilient distributed optimization-based control of multi-agent systems with dual interaction networks," in *AIAA SCITECH 2022 Forum*, 2022, p. 2342.

[98] C. Kwon and I. Hwang, "Sensing-based distributed state estimation for cooperative multiagent systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 6, pp. 2368–2382, 2018.

[99] M. Dressler, A. Uschmajew, and V. Chandrasekaran, "Kronecker product approximation of operators in spectral norm via alternating sdp," *arXiv preprint arXiv:2207.03186*, 2022.

[100] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[101] B. G. Coury and R. D. Semmel, "Supervisory control and the design of intelligent user interfaces," in *Automation and human performance*, Routledge, 2018, pp. 221–242.

[102] J. L. Wright, J. Y. Chen, and M. J. Barnes, "Human–automation interaction for multiple robot control: The effect of varying automation assistance and individual differences on operator performance," *Ergonomics*, vol. 61, no. 8, pp. 1033–1045, 2018.

[103] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.

[104] W. M. Wonham, "Supervisory control of discrete-event systems," *Encyclopedia of systems and control*, pp. 1396–1404, 2015.

[105] Z. Guo, D. Shi, K. H. Johansson, and L. Shi, "Optimal linear cyber-attack on remote state estimation," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 4–13, 2016.

[106] C. Kwon and I. Hwang, "Cyberattack-resilient hybrid controller design with application to UAS," in *Safe, Autonomous and Intelligent Vehicles*, Springer, 2019, pp. 33–56.

[107] G. Wu, J. Sun, and J. Chen, "A survey on the security of cyber-physical systems," *Control Theory and Technology*, vol. 14, no. 1, pp. 2–10, 2016.

[108] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *Dynamic Games and Applications*, pp. 1–19, 2017.

[109] Y. Wang and M. Pajic, "Supervisory control of discrete event systems in the presence of sensor and actuator attacks," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 5350–5355.

[110] J. P. Hespanha, "Tutorial on supervisory control," in *Lecture Notes for the workshop Control using Logic and Switching for the 40th Conf. on Decision and Contr., Orlando, Florida*, 2001.

[111] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, ACM, 2017, pp. 506–519.

[112] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*, Springer, 2019, pp. 703–716.

[113] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[114] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[115] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[116] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski, "Hybrid control of formations of robots," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, IEEE, vol. 1, 2001, pp. 157–162.

[117] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, IEEE, vol. 4, 1998, pp. 2864–2869.

[118] P. Niyogi, "The informational complexity of learning from examples," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

VITA

Omanshu Thapliyal received his B.Tech. in Aerospace Engineering from Indian Institute of Technology Kanpur, India, in 2014. He obtained the M.S. degree in Aeronautics & Astronautics Engineering from Purdue University, West Lafayette, in 2017. He finished his M.S. thesis titled "Kalman Filtering for LTI Systems with State Dependent Packet Losses" under the supervision of Professor Inseok Hwang. He worked at the MathWorks Inc., Natick, during 2018, on improving Kalman Filters in MATLAB and Simulink.

He is currently pursuing his PhD. in Aeronautics & Astronautics Engineering at the School of Aeronautics & Astronautics Engineering at Purdue University, West Lafayette. His dissertation is titled "Data-Driven Safety & Security for Cyberphysical Systems". He is currently employed as an Urban Air Mobility Researcher at Hitachi America Ltd., Santa Clara. His research interests include data-driven safety and security of cyber-physical systems, multi-agent state estimation, and optimization and control in networked systems.

PUBLICATIONS

Thapliyal, O., Nandiganahalli, J. S., & Hwang, I. (2017, December). Optimal state estimation for LTI systems with imperfect observations. In 2017 *IEEE 56th Annual Conference on Decision and Control (CDC)* (pp. 2795-2800). IEEE.

Deshmukh, R., **Thapliyal, O.**, Kwon, C., & Hwang, I. (2018). Distributed state estimation for a stochastic linear hybrid system over a sensor network. *IET Control Theory & Applications*, 12(10), 1456-1464.

Thapliyal, O., Nandiganahalli, J. S., & Hwang, I. (2019). Kalman filtering with state-dependent packet losses. *IET Control Theory & Applications*, 13(2), 306-312.

Sivaramakrishnan, V., **Thapliyal, O.**, Vinod, A., Oishi, M., & Hwang, I. (2019, December). Predicting Mode Confusion Through Mixed Integer Linear Programming. In 2019 *IEEE 58th Conference on Decision and Control (CDC)* (pp. 2442-2448). IEEE.

Thapliyal, O., & Hwang, I. (2021, June). Learning based Cyberattack Design and Defense for Supervisory Control Systems. In *2021 European Control Conference (ECC)* (pp. 144-149). IEEE.

Shethia, S., Gupta, A., **Thapliyal, O.**, & Hwang, I. (2021, October). Distributed Fast-Tracking Alternating Direction Method of Multipliers (ADMM) Algorithm with Optimal Convergence Rate. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 976-981). IEEE.

Thapliyal, O., & Hwang, I. (2021, May). Path Planning for a Network of Robots with Distributed Multi-Objective Linear Programming. In *2021 American Control Conference (ACC)* (pp. 4643-4648). IEEE.

Thapliyal, O., & Hwang, I. (2022). Approximate Reachability for Koopman Systems Using Mixed Monotonicity. *IEEE Access*, 10, 84754-84760.

Clarke, S. G., **Thapliyal, O.**, Hwang, S., & Hwang, I. (2022). Attack-Resilient Distributed Optimization-based Control of Multi-Agent Systems with Dual Interaction Networks. In *AIAA SCITECH 2022 Forum* (p. 2342).

Thapliyal, O., & Hwang, I. (2022). Data-driven Cyberattack Synthesis against Network Control Systems. arXiv preprint arXiv:2211.05203.

Clarke, S. G., **Thapliyal, O.**, & Hwang, I. (2022). Provably Stabilizing Model-Free Q-Learning for Unknown Bilinear Systems. arXiv preprint arXiv:2208.13843.

Thapliyal, O., & Hwang, I. (2023). Approximating Reachable Sets for Neural Network-Based Models in Real Time via Optimal Control. *IEEE Transactions on Control Systems Technology*.