DATA-DRIVEN DECISION MAKING AND CONTROL OF RATIONAL AGENTS

by

PATRIK KOLARIC, M.Sc.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at Arlington
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

COMMITTEE MEMBERS:
Frank Lewis, Ph.D., Chair
Yan Wan, Ph.D.
Nicholas Gans, Ph.D.
Ramtin Madani, Ph.D.
Jonathan Bredow, Ph.D.

THE UNIVERSITY OF TEXAS AT ARLINGTON
College of Engineering
Department of Electrical Engineering
May 2021

DEDICATION

This work is dedicated to Ivona for her love, support and unrelenting courage.

ACKNOWLEDGEMENTS

Many people had a positive influence on me during my doctoral studies. The knowledge and principles they shared with me are mirrored in this thesis. I will mention some of them here.

My supervisor, Dr. Frank Lewis, the greatest teacher I have ever had. He taught me how to write as a researcher, how to communicate as a researcher and how to think as one. He taught me humility and character. His way of life and principles he follows will remain the strongest source of inspiration and guidance. I have yet to earn these rare gifts.

The professors from my committee, with whom I had insightful conversations during my PhD. I am honored by the attention they gave me and proud for being their student.

Victor, a close collaborator and a mentor. He is an open-minded researcher with an inspiring passion for research and strong discipline. Discussions with Victor, both formal and informal, were very important element of my doctoral studies. He is also a great friend.

Ci, my collaborator and a mentor. First journal paper is very tough and confusing experience for every researcher. Ci helped me to bring class and quality to my first journal paper. In doing so, he set a high standard for my future papers.

I would like to thank my closest collaborators at UTARI: Wenqian, Yusuf, Bosen and Raghu. They are very original thinkers who improved me as a researcher.

I grew up in a small village at the north of Croatia in an open minded environment where curiosity was encouraged, hard work was praised and love was unconditional. I cannot thank enough to my sister Jessy and parents Alenka and Zdravko for fostering these values. They are the ones who planted a seed for this work.

May 2021

DATA-DRIVEN DECISION MAKING AND CONTROL OF RATIONAL AGENTS

Patrik Kolaric, Ph.D.
The University of Texas at Arlington, 2021

Supervising Professors: Frank Lewis, Ph.D. and Co-Supervisor Name, Degree

Abstract This dissertation studies the problem of data-driven optimal decision making. The 4 main contributions of this work are listed here.

First, we develop a model-based and data-driven techniques for learning the cost of an Expert agent. This ties fields of Inverse Optimal Control and Inverse Reinforcement Learning and represents a first data-driven algorithm of this kind in the control community.

Next, we have developed optimally adaptive dynamic control allocation mechanism that optimally re-configures redundant actuators in a model-free fashion, that is, based on collected data. This work pushed the multiple frontiers of control allocation research, since state-of-the-art control allocation was

Next, we have introduced an uncertainty aware trajectory optimization technique that uses the information about the model uncertainty to inform the generation of local feedback policy which makes the open loop solution more reliable and robust.

Finally, a cooperative protocol for distributed formation control has been developed and tested on the real system in the lab. This was among the first real world examples of multi-agent distributed formation control.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1: INTRODUCTION

At the high level, this dissertation studies the problem of optimal decision making. The problem of decision making has been studied in various engineering disciplines such as, control theory, computer science, game theory, machine learning, artificial intelligence, operations research to name a few. The unified solution to this fundamental problem does not exist. Modern technological advancement is critically dependent on our ability to automate decision making process in the machine.

In this dissertation, we focus on decision making of dynamic systems, that is, systems whose state changes over time and depends on the actions taken in the past. Optimal control is one of the best studied frameworks for analyzing dynamic decision making and most of the work in this dissertation relies on it. The theory of optimal control has a long history that can be traced back to the calculus of variations and brachistochrone problem. We omit the overview of this historic development, but interested reader is referred to [97].

Optimal control offers many successful techniques for designing decision making systems. However, it requires human to specify a cost function. In other words, the objectives or motives of a decision making agent have to be understood *a priori*. The designer of optimal controller has to codify these motives and objectives by selecting a proper cost function. This can be a hard problem.

In order to advance decision making theory and automate associated technologies, the research community needs to find a way to mathematically argue about objectives, intentions and motives of agents that perform complex tasks. One of the contributions of this dissertation is the formal introduction of a method that can be used to infer the objectives and motives of a, so called, Expert agent, by determining the cost function that such an agent optimizes.

Another disadvantage of optimal control is that it depends on the model of system dynamics. This means that the efficiency of optimal controller depends on the accuracy of the model. Moreover, optimal controller has to be pre-computed offline.

Both problems mentioned here can be partially addressed using data-driven methods. Specifically, the theory of reinforcement learning has been developed to learn optimal policies through experience. However, the "experience" is again associated with a reward function which is mathematically equivalent to cost function in optimal control. In other words, the objectives, motives and intentions of agent have to be properly represented using the reward function. This brings us to the first contribution:

Contribution 1): *In this work, we develop a model-based and data-driven techniques for learning the cost of an Expert agent. This ties fields of Inverse Optimal Control and Inverse Reinforcement Learning and represents a first data-driven algorithm of this kind in the control community.*

One of the most celebrated data-driven techniques in optimal control is continuous time integral reinforcement learning. Despite the attractive theory and effective algorithms, there have been very few applications reported where this technique is used.

We recognized the problem of control allocation of redundant systems as a very interesting application of integral reinforcement learning. Many robots have a redundancy built into their system. In most cases, the redundancy protects the system in cases of critical failures of some components. In these cases, the system needs to optimally adapt to new circumstance. Moreover, even when the system is functioning healthy, the redundancy present in the system can be utilized by the system. This brings us to next contribution.

Contribution 2): *We have developed optimally adaptive dynamic control allocation mechanism that optimally re-configures redundant actuators in a model-free fashion, that is, based on collected data.*

In our investigation of data-driven techniques in decision making, we have found that data can be very useful for evaluating the level of uncertainty in the models used in optimal control. If the model can inform the algorithm not only about the next state, but also the uncertainty of that prediction, then the information about the uncertainty can be used to make the policy robust.

In this dissertation, we study a case of standard robotic trajectory optimization where the open loop solution is made more robust by adding the feedback term informed by the model uncertainty.

This is our next contribution

Contribution 3): *We have developed an uncertainty aware trajectory optimization technique that uses the information about the model uncertainty to inform the generation of local feedback policy which makes the open loop solution more reliable and robust.*

Finally, optimal control has not yet been well generalized to the problem of multi-agent setting. The problem is that when one deals with multi-agent setting then the global optimality does not follow from the local optimality of each agent. Moreover, the communication graph affects the behaviour of agents. In Chapter 7, we also disclose some preliminary results on applying inverse reinforcement learning to the problem of distributed decision making over a group of networked agents. It turns out that inverse reinforcement learning can be used to encourage global optimality by incentivizing each agent locally.

Contribution 4): *In addition, a cooperative protocol for distributed formation control has been developed and tested on the real system in the lab. This was among the first real world examples of multi-agent distributed formation control.*

Chapter 2: REVIEW OF OPTIMAL DECISION MAKING: OPTIMAL CONTROL AND REINFORCEMENT LEARNING

In this chapter, we will review two well defined frameworks for studying optimal decision making: optimal control and reinforcement learning. Both theories are used throughout the dissertation and therefore deserve to be introduced in this separate chapter.

2.1 Optimal Control Problem

The theory of optimal control has a long history that can be traced back to the calculus of variations and brachistochrone problem. We omit the overview of this historic development, but interested reader is referred to [97].

In optimal control, we are interested in the analysis and design of optimal decisions. The state of the system that we want to control is defined by a real vector x. The real vector u represents the decision trajectory and is often termed as the input vector in the control literature. The evolution of the state $x(t)$ over time, given the initial condition $x(t = 0)$ and decision trajectory $u(t)$ is governed by a non-linear differential equation

$$\begin{aligned} \dot{x} &= F(x, u) \\ F(x, u) &\triangleq f(x) + g(x)u \end{aligned} \tag{2.1}$$

In order to argue about the optimality of decision trajectory $u(t)$, we need a formal ranking criteria that compares the utility of an arbitrary decision trajectory $u_1(t)$ against some other decision trajectory $u_2(t)$. In this scenario, the ranking criteria should claim that $u_1(t)$ is either better, worse or equal compared to $u_2(t)$. This ranking criteria is widely referred to as the cost function or performance index. The definition of cost function is as follows

$$J(x(t), u(t)) \triangleq \int_t^{\infty} L(x(\tau), u(\tau))d\tau \tag{2.2}$$

The cost function $J(x(t), u(t))$ is a performance measure of decision trajectory $u(t)$ applied to the system (2.1).

If the cost function is well defined, then there exists an optimal decision trajectory (not necessarily unique) defined as

$$u^*(t) \triangleq arg\min_{u(t)} J(x(t), u(t)) \tag{2.3}$$

The system of equations (2.1), (2.2) and (2.3) constitutes a well defined framework for analyzing and designing optimal decision making systems.

A powerful specialization of optimal decision making framework based on (2.1), (2.2) and (2.3) is that of state feedback policy design. In this specialization, the decision trajectory is the explicit function of state trajectory as follows

$$u(t) = \pi_w(x(t)) \tag{2.4}$$

where w are the parameters of state feedback policy $\pi_w(x(t))$. The optimal state feedback policy is then defined as

$$\begin{aligned} \pi^*(x(t)) &\triangleq \pi_{w^*}(x(t)) \\ w^*(t) &\triangleq arg\min_{w} J(x(t), \pi_w(x(t))) \end{aligned} \tag{2.5}$$

In order to understand the power of state feedback design, consider the complexity involved with searching for an optimal decision trajectory in (2.3) and compare it with the problem of determining a finite dimensional vector of parameters w in (2.5). Clearly, (2.5) is more compact problem of optimization over the space of parameter w with the solution that generalizes across the state space.

Since the search for the optimal state feedback policy design takes place in the space of parameter w it is crucial to tie that parameter space with the associated cost. In other words, there is a map from parameter space to the cost of associated policy $w \rightarrow J$. This map is widely known as

the value function and is defined as follows

$$J^{\pi_w}(x(t)) \triangleq J(x(t), \pi_w(x(t))) \tag{2.6}$$

Moreover, the optimal value function that measures the performance of the optimal policy $\pi^*(x(t))$ is then given as

$$J^*(x(t)) \triangleq J^{\pi^*}(x(t)) \tag{2.7}$$

where w^* is defined in (2.5).

2.2 Solving Optimal Control Problem

In this subsection, we formalize the procedure for solving optimal control problem. The solution relies on the celebrated Pontryagin's Maximum Principle and related stationarity conditions.

First, define Hamiltonian function as

$$H(x, u, p) \triangleq p^T F(x, u) + L(x, u) \tag{2.8}$$

The state feedback solution to the optimal control problem satisfies certain conditions that are formalized in next theorem (source: [33]).

Theorem 1. *Consider a policy π and its corresponding value function J^π that satisfy set of equalities and inequalities*

$$\begin{aligned} J^\pi(0) &= 0 \\ J^\pi(x) &> 0 \;\; x \in \mathcal{D} \\ \pi(0) &= 0 \end{aligned} \tag{2.9}$$

$$\nabla_x J^\pi(x) F(x, \pi(x)) < 0 \;\; x \in \mathcal{D} \tag{2.10}$$

$$\begin{aligned} H(x, \pi(x), \nabla_x J^\pi(x)) &= 0 \;\; x \in \mathcal{D} \\ H(x, u, \nabla_x J^\pi(x)) &\geq 0 \;\; x \in \mathcal{D} \;\; u \in \mathcal{U} \end{aligned} \tag{2.11}$$

Then, the policy π is stable in the sense of Lyapunov where the value function J^π serves as the Lyapunov function. Moreover, the policy π is the optimal state feedback policy π^* that solves (2.5).

Proof. First write the Bellman equation

$$\begin{aligned} H(x, \pi(x), \nabla_x J^\pi(x)) &= \nabla_x J^\pi(x)^T F(x, \pi(x)) + L(x, \pi(x)) \\ &= \dot{J}^\pi(x) + L(x, \pi(x)) \\ 0 &= \dot{J}^\pi(x) + L(x, \pi(x)) \end{aligned} \tag{2.12}$$

Then, since $L(x, u) > 0 \ \ \forall x, u/\{0\}$ we have

$$\dot{J}^\pi(x) < 0 \tag{2.13}$$

Therefore, using $J^\pi(x) > 0$, we recognize that J^π is Lyapunov function and conclude that policy π is stable in the sense of Lyapunov.

Next, we show the optimality of π. To see that, consider Hamilton inequality

$$\begin{aligned} J(x, u) &= \int_t^\infty L(x(\tau), u(\tau)) d\tau \\ \int_t^\infty (H(x, u, \nabla_x J^\pi(x)) - \nabla_x J^\pi(x))^T F(x, u)) d\tau \\ \int_t^\infty (H(x, u, \nabla_x J^\pi(x)) - \dot{J}^\pi(x)) d\tau \\ J^\pi(x) + \int_t^\infty (H(x, u, \nabla_x J^\pi(x))) d\tau &\geq J^\pi(x) \end{aligned} \tag{2.14}$$

Therefore, $J(x, u) \geq J^\pi(x)$ which implies that $\pi(x)$ is the optimal policy. □

Two important results can be obtained for the case when the input is quadratically weighted in the cost function.

Lemma 1. *Let the stage cost $L(x, u)$ be given as*

$$L(x, u) = Q(x) + u^T R u \tag{2.15}$$

Then the optimal policy is given as

$$\pi^* = -\frac{1}{2}R^{-1}g(x)^T\nabla_x J^{\pi^*}(x) \tag{2.16}$$

Lemma 2. *Let the stage cost $L(x,u)$ be given as*

$$L(x,u) = Q(x) + u^T R u \tag{2.17}$$

Then following holds

$$H(x,u,\nabla_x J^{\pi^*}(x)) = (u-\pi^*)^T R(u-\pi^*) \tag{2.18}$$

2.3 Reinforcement Learning Problem

In this section, we review problem definition of reinforcement learning which can be thought of as a data-driven equivalent of optimal control.

Since the reinforcement learning is a data-driven method, it is typically analyzed using probabilistic approaches and Markov Decision Processes (MDPs). Therefore, we will introduce the problem in probabilistic setting and then switch to deterministic analysis to show the relation between reinforcement learning and optimal control.

Consider the MDP (X, U, P, R) where X is a set of states and U is the set of actions. The set of transition probabilities between states is denoted as P where $P^u_{x,x'}$ indicates the probability of transitioning from state x and to state x' after making action u and the reward function (negative stage cost) is given as R where $R^u_{x,x'}$ is the reward associated with that transition.

2.4 Continuous-Time Reinforcement Learning Algorithms

In this section, we introduce continuous time version of common reinforcement learning algorithms.

2.4.1 Continuous-Time On Policy Reinforcement Learning

We start by looking at equation (2.11) and recognize that the non-linear Lyapunov equation

$$0 = Q(x) + \pi^T(x) R \pi(x) + \nabla_x J^{\pi T}(x) \dot{x} \tag{2.19}$$

is in fact equivalent to the Bellman equation. This becomes more evident if (2.19) is written as

$$0 = Q(x) + \pi^T(x) R \pi(x) + \frac{d}{dt} J^{\pi T}(x) \tag{2.20}$$

and then integrated to get

$$J^{\pi}(x) = \int_t^{\infty} (Q(x) + \pi^T(x) R \pi(x)) d\tau \tag{2.21}$$

extracting the tail gives us

$$J^{\pi}(x) = \int_t^{t+T} (Q(x) + \pi^T(x) R \pi(x)) d\tau + \int_{t+T}^{\infty} (Q(x) + \pi^T(x) R \pi(x)) d\tau \tag{2.22}$$

Finally, the integral form of Bellman equation (2.19) is given as

$$J^{\pi}(x(t)) = \int_t^{t+T} (Q(x) + \pi^T(x) R \pi(x)) d\tau + J^{\pi}(x(t+T)) \tag{2.23}$$

Combining (2.23) with the optimality conditions from (2.11) we can write policy iteration algorithm

Algorithm 2.1 ***Continuous Time - Policy Iteration***

Input: Symmetric cost matrices Q and R
Output: Optimal policy $\pi^*(t) = -R^{-1}g^T(x)\nabla_x J^{\pi^*}$ (see Definition 2).
Initialization: Initialize stabilizing policy π_0.

1: **while** $||\pi_i - \pi_{i-1}|| > \epsilon$ **do**

2: **Policy Evaluation.** Estimate J^{π_i} using

$$J^{\pi_i}(x(t)) - J^{\pi_i}(x(t+T)) = \int_t^{t+T} (Q(x) + \pi_i^T(x)R\pi_i(x))d\tau \tag{2.24}$$

3: **Policy Improvement.** Estimate policy π_{i+1} by using

$$\pi_{i+1}(x) = arg\ \min_z\ \ H(x, z, \nabla_x J^{\pi_i}(x)) \tag{2.25}$$

or explicitly

$$\pi_{i+1}(x) = -\frac{1}{2}R^{-1}g^T(x)\nabla_x J^{\pi_i}(x) \tag{2.26}$$

This result is thoroughly presented in [59].

2.4.2 Continuous-Time Off-Policy Reinforcement Learning

In this section we discuss off-policy version of algorithms from previous subsection. The "off-policy" in reinforcement learning indicates that the policy applied to collect the data for learning is different from the learned policy. The policy which is used for collecting the data is often called *behavior policy* $\pi_b(x)$ and the policy that is actually learned is denoted $\pi_i(x)$ as before.

We will sketch off-policy reinforcement learning derivation. The full derivations and results originate from [48]. Start by writing the system dynamics resulting from behavior policy $\pi_b(x)$ as

$$\dot{x} = f(x) + g(x)\pi_b(x) \tag{2.27}$$

Use the trick to write

$$\dot{x} = f(x) + g(x)\pi_b(x) + g(x)(\pi_i(x) - \pi_i(x)) \tag{2.28}$$

and manipulate this expression to write

$$\dot{x} = f(x) + g(x)\pi_i(x) + g(x)(\pi_b(x) - \pi_i(x)) \tag{2.29}$$

Next, let us introduce Bellman equation that evaluates $\pi_i(x)$ along the trajectory produced by behavior policy $\pi_b(x)$ as follows

$$\begin{aligned} 0 &= Q(x) + \pi_i^T R\pi_i + \nabla_x J^{\pi_i T}\dot{x} \\ 0 &= Q(x) + \pi_i^T R\pi_i + \nabla_x J^{\pi_i T}(f(x) + g(x)\pi_b(x)) \end{aligned} \tag{2.30}$$

and now use (2.29) to write

$$0 = Q(x) + \pi_i^T R\pi_i + \nabla_x J^{\pi_i T}(f(x) + g(x)\pi_i(x)) + \nabla_x J^{\pi_i T} g(x)(\pi_b(x) - \pi_i(x)) \tag{2.31}$$

Now note that $\nabla_x J^{\pi_i T}(f(x) + g(x)\pi_i(x)) = \dot{J}^{\pi_i}$ and therefore

$$0 = Q(x) + \pi_i^T R\pi_i + \dot{J}^{\pi_i} + \nabla_x J^{\pi_i T} g(x)(\pi_b(x) - \pi_i(x)) \tag{2.32}$$

Moreover, from (2.26) we see that $\nabla_x J^{\pi_i T} g(x) = -2R\pi_{i+1}(x)$ and therefore

$$0 = Q(x) + \pi_i^T R\pi_i + \dot{J}^{\pi_i} - 2\pi_{i+1}^T(x)R(\pi_b(x) - \pi_i(x)) \tag{2.33}$$

In order to learn from state trajectories, we integrate (2.33) and write the integral equation

$$0 = \int_t^\infty (Q(x) + \pi_i^T R\pi_i)d\tau + J^{\pi_i} - 2\int_t^\infty \pi_{i+1}^T(x)R(\pi_b(x) - \pi_i(x))d\tau \tag{2.34}$$

This gives us an expression for evaluating J^{π_i}

$$J^{\pi_i}(x(t)) = -\int_t^\infty (Q(x) + \pi_i^T R\pi_i)d\tau + 2\int_t^\infty \pi_{i+1}^T(x)R(\pi_b(x) - \pi_i(x))d\tau \tag{2.35}$$

Moreover, since

$$J^{\pi_i}(x(t+T)) = -\int_{t+T}^{\infty}(Q(x)+\pi_i^T R\pi_i)d\tau + 2\int_{t+T}^{\infty}\pi_{i+1}^T(x)R(\pi_b(x)-\pi_i(x))d\tau \tag{2.36}$$

then we can express $J^{\pi_i}(x(t)) - J^{\pi_i}(x(t+T))$ as

$$\begin{aligned} &J^{\pi_i}(x(t))-J^{\pi_i}(x(t+T)) \\ &\qquad = -\int_{t}^{t+T}(Q(x)+\pi_i^T R\pi_i)d\tau + 2\int_{t}^{t+T}\pi_{i+1}^T(x)R(\pi_b(x)-\pi_i(x))d\tau \end{aligned} \tag{2.37}$$

This brings us to the algorithm.

Algorithm 2.2 ***Continuous Time, Off-Policy - Policy Iteration***

Input: Symmetric cost matrices Q and R

Output: Optimal policy $\pi^*(t) = -R^{-1}g^T(x)\nabla_x J^{\pi^*}$ (see Definition 2).

Initialization: Initialize stabilizing policy π_0.

1: **while** $||\pi_i - \pi_{i-1}|| > \epsilon$ **do**

2: **Policy Evaluation and Policy Improvement.** Update J^{π_i} and π_{i+1} using

$$\begin{aligned} &J^{\pi_i}(x(t))-J^{\pi_i}(x(t+T)) \\ &\qquad = -\int_{t}^{t+T}(Q(x)+\pi_i^T R\pi_i)d\tau + 2\int_{t}^{t+T}\pi_{i+1}^T(x)R(\pi_b(x)-\pi_i(x))d\tau \end{aligned} \tag{2.38}$$

The utility of this algorithm is that the behaviour policy can be independent of learned policy. In other words, behaviour policy can be selected as a safe and stable policy and learned policy does not have to be applied until convergence to the optimal policy. We omit the proof of convergence which can be found in [48].

Chapter 3: INVERSE REINFORCEMENT LEARNING

The optimal control is a well developed field of control [63]. The theory relies on well formulated concepts of optimality of dynamic systems such as dynamic programming [6] and Pontryagin's minimum principle [63]. The general optimal control is considered to be a hard problem to solve analytically, as it requires solution to Hamilton-Jacobi-Bellman (HJB) equation.

The attractive property of optimal control theory is that it provides a clear framework for establishing the stability of resulting control policies. Under certain conditions, the optimal control policy inherits the property of dynamic stability just from the fact that it is optimal. In other words, a control policy that is optimal with respect to selected performance index is also dynamically stable, if and only if the performance index has a certain admissible form [33].

The problem of selecting the performance index in optimal control such that the optimal policy has desired properties is called Inverse Optimal Control. This problem was first proposed in [50] where authors regarded stability as the main property that one should ensure when selecting performance index. The authors provide a solution for scalar case. In [78] authors recommend additional properties like controllability and observability. Well developed content on Inverse Optimal Control is summarized in [33] and [11] where authors used Lyapunov Stability theory to give sufficient conditions for determining performance indices that yield stable optimal policies. More recent results and applications of Inverse Optimal Control can be found in [81], [123], [28], [89]

The family of methods that solve optimal control in iterative real-time data-driven fashion are called Reinforcement Learning [98], Adaptive Dynamic Programming [60], [112] or Approximate Dynamic Programming [8]. In each iteration, the control policy is evaluated based on collected data and then improved. These data-driven iterative procedures are akin to matrix iterations which were developed to solve Algebraic Riccati Equations, [52], [55], [40] and [39]

Recently, Inverse Reinforcement Learning was studied in [75], [1], [76], [70] [90], [34], [2], [126], [125], [92], [26], [18], [100], [118], [32], [66], [30]. Unlike Inverse Optimal Control, where the goal is to find a family of performance indices that satisfy a desirable property, the Inverse

Reinforcement Learning recovers a specific performance index under which the observed actions of an Expert agent are optimal. One of the drawbacks of existing Inverse Reinforcement Learning methods is that they use various heuristics to ensure that the resulting performance index is a meaningful representation of the Expert's objective. Moreover, the policies that are optimal with respect to reconstructed performance index do not have clear stability guarantees.

In this chapter, the theory of Inverse Optimal Control is used to overcome the shortcomings of existing Inverse Reinforcement Learning solutions. The idea is to use Inverse Optimal Control to isolate the family of performance indices that are guaranteed to produce a stabilizing control policy. Firstly, the model-based iterative algorithms are provided along with convergence analysis. Secondly, the data-driven algorithms are derived as approximations to model-based algorithms. Our contributions are summarized as follows.

1) We propose a novel, model-based Inverse Reinforcement Learning problem formulation motivated by well studied Inverse Optimal Control techniques that ensure reconstruction of meaningful performance indices. This clarifies the relation between Inverse Reinforcement Learning and Inverse Optimal Control.

2) It is shown that the optimal cost that generates a given Expert policy may not be unique. We characterize all possible solutions to the Inverse Reinforcement Learning problem.

3) Two novel iterative algorithms for solving model-based Inverse Reinforcement Learning are presented along with the convergence analysis. The first is based on Riccati iterations and the second is based on Lyapunov iterations. We do not assume knowledge of Expert policy. Instead, only the knowledge of Expert dynamics is assumed.

4) A data-driven algorithm is given for implementing Inverse RL in real time without knowing the Expert's dynamics, but only observing its state trajectories. A rigorous derivation is provided.

3.1 Problem Formulation

3.1.1 Set of optimal solutions to LQR

In this subsection, we present the optimality conditions of LQR problem. Consider linear control system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.1}$$

with state $x(t) \in \mathbb{R}^{n\times 1}$ and input $u(t) \in \mathbb{R}^{m\times 1}$. System matrices are defined as $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ and the system (A, B) is assumed to be controllable. The linear feedback policy $u(t) = -Kx(t)$ and the system dynamics in (3.1) together define closed loop dynamics

$$\dot{x}(t) = (A - BK)x(t) \triangleq A_K x(t) \tag{3.2}$$

Consider the cost function

$$J(x(t), u(t)) = \int_t^\infty \{x^T(\tau)Qx(\tau) + u^T(\tau)u(\tau)\}d\tau \tag{3.3}$$

Setting linear policy $u = -Kx$ in the cost function (3.3) defines state dependent value function

$$\begin{aligned} J^K(x(t)) &\triangleq J(x(t), -Kx(t)) \\ &= \int_t^\infty x^T(\tau)\{Q + K^T K\}x(\tau)d\tau \end{aligned} \tag{3.4}$$

The cost $J^K(x(t))$ in (3.4) is commonly referred to as value function [60]. Well known result in optimal control [63] is that the optimal gain K^* that minimizes the performance index in (3.4) is given as $K^* = B^T P$ where P solves

$$\begin{aligned} 0 &= PA + A^T P + Q - PBB^T P \\ Q &= Q^T \end{aligned} \tag{3.5}$$

Definition 1 (Optimality Set)**.** *Given a controllable pair* (A, B)*, the set of all* (Q, P) *that satisfy*

(3.5) is referred to as the Optimality Set.

The Optimality Set identifies the most general set of (Q, P) that are guaranteed to produce stable control policy $u(t) = Kx(t)$. This is the result from Inverse Optimal Control [33] specialized for LQR case.

3.1.2 Expert Policy and Expert Subset

We consider two agents, the Expert and the Learner. The Expert is an agent that employs an optimal control policy with some favorable properties. The Behavior goal is to mimic the Expert by learning its intentions (performance index), rather than learning how the Expert acts in a specific situation (control policy). The technical definition of this problem is discussed later in Subsection 3.1.3. Current subsection is used to formally introduce the concepts of Expert Policy and Expert Subset.

The Expert Policy is denoted as $u^* \triangleq -\mathcal{K}^* x$, where $\mathcal{K}^*$ is referred to as Expert Gain. The Expert Policy u^* and the Expert Gain $\mathcal{K}^*$ are fixed and unknown in our formulation. Note that the closed loop dynamics of Expert is given as $\dot{x}(t) \triangleq A_{\mathcal{K}^*} x(t) = (A - B\mathcal{K}^*)x(t)$. The Expert Policy $u^* = -\mathcal{K}^* x$ is optimal with respect to some cost from the proposed family of quadratic cost functions in (3.3).

Definition 2 (Expert Subset)**.** *The Expert Subset $(\mathcal{Q}, \mathcal{P})$ of the Optimality Set (Q, P), consists of all solutions to (3.5) with the gain K^* fixed to the Expert Gain $\mathcal{K}^*$. That is, the Expert Subset $(\mathcal{Q}, \mathcal{P}) \in (Q, P)$ is given as the set of solutions to*

$$\begin{aligned} 0 &= \mathcal{P}A + A^T\mathcal{P} + \mathcal{Q} - \mathcal{P}BB^T\mathcal{P} \\ \mathcal{Q}^T &= \mathcal{Q} \\ B^T\mathcal{P} &= \mathcal{K}^* \end{aligned} \tag{3.6}$$

It is possible that the Expert Policy is optimal with respect to more than one cost from the family of costs. That is, given $\mathcal{K}^*$, the $(\mathcal{Q}, \mathcal{P})$ that solve (3.6) may not be unique. This non-uniqueness in

the space of solutions to (3.6) is formalized by introducing the Expert Subset in Definition 2. In Theorem 2 of Section 3.2 we identify the Expert Subset analytically.

3.1.3 Inverse Reinforcement Learning Problem Definition

We are now ready to formulate the Inverse Reinforcement Learning problems.

Definition 3 (Model-Based Inverse Reinforcement Learning)**.** *Given the knowledge of the dynamics model* (A, B) *and the closed loop dynamics of the Expert,* $A_{\mathcal{K}^*}$*, solve (3.6). In shorthand notation,* $(A, B, A_{\mathcal{K}^*}) \stackrel{?}{\rightarrow} (\mathcal{Q}, \mathcal{P}, \mathcal{K}^*)$.

Definition 4 (Data-Driven Inverse Reinforcement Learning)**.** *Given Expert trajectories* $x_{\mathcal{K}^*}(t) \triangleq \{x(t) \mid \dot{x}(t) = A_{\mathcal{K}^*}x(t),\ \ t \in (0, \infty)\}$*, solve (3.6). In shorthand notation,* $x_{\mathcal{K}^*}(t) \stackrel{?}{\rightarrow} (\mathcal{Q}, \mathcal{P}, \mathcal{K}^*)$.

Solving the model-based problem in Definition 3 is instrumental for determining the solution to data-driven problem in Definition 4. This is because the Expert trajectories $x_{\mathcal{K}^*}(t)$ are the results of closed loop Expert dynamics $A_{\mathcal{K}^*}$. Therefore, if we can determine the model-based algorithm that solves (3.6), then the data-driven algorithm can be derived as its approximation. Therefore, we first propose the model-based algorithm in Section 3.3 from which the data-driven algorithm emerges naturally in Section 3.4.

Before introducing the algorithms that solve problems from Definition 3 and Definition 4, we need to develop better understanding of the solution space that these algorithms are converging towards. Therefore, next section is used to characterize the solution space - Expert Subset.

3.2 The Expert Subset Analysis

As mentioned earlier, given the Expert Policy $u^*(t) = \mathcal{K}^*x(t)$, the pair $(\mathcal{Q}, \mathcal{P})$ that solves the system of equations in (3.6) may not be unique. Since the objective of this chapter is to recover one such pair, we wish to understand the solution space of all such pairs. In this section we characterize all possible solutions to (3.6), that is, the Expert Subset defined in Definition 2.

3.2.1 Analytical expression for the elements of the Expert Subset

In our first result in Theorem 2, the analytical expression for the elements of the Expert Subset $(\mathcal{Q}, \mathcal{P})$ given the Expert Gain $\mathcal{K}^*$ is provided. Before that, in Lemma 3 we introduce a matrix equation that can be used to determine the Expert Subset. This matrix equation is useful alternative to the matrix equation (3.6) which is originally used to define the Expert Subset in Definition 2.

Lemma 3 (Expert Subset in Terms of $A_{\mathcal{K}^*}$)**.** *The set of solutions to*

$$\begin{aligned} 0 &= \mathcal{P}A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T\mathcal{P} + \mathcal{Q} + \mathcal{P}BB^T\mathcal{P} \\ \mathcal{Q}^T &= \mathcal{Q} \end{aligned} \tag{3.7}$$

is equivalent to set of solutions to (3.6).

Proof. The equation (3.7) can be derived directly from (3.6) and the equivalence of these two equations is well known in Optimal Control literature. □

The equation (3.7) is important for the problem of Inverse Reinforcement Learning because it ties the closed loop dynamics $A_{\mathcal{K}^*}$ with the Expert Subset $(\mathcal{Q}, \mathcal{P})$. This allows us to develop algorithms that update $\mathcal{Q}$ and $\mathcal{P}$ based on $A_{\mathcal{K}^*}$ rather than $\mathcal{K}^*$. Note that assuming the knowledge of $A_{\mathcal{K}^*}$ is less restrictive than assuming the knowledge of $\mathcal{K}^*$. More importantly, the true utility of the algorithm that is driven by the knowledge of $A_{\mathcal{K}^*}$ is that it has a natural data-driven equivalent where $A_{\mathcal{K}^*}$ is replaced by the Expert Trajectory $x_{\mathcal{K}^*}(t)$ as we shall see in Section 3.4. In next definition we introduce the concept of the null-space of input matrix B. As we shall see in Theorem 2, the null-space of the input matrix B provides the shape and determines the dimension of the Expert Subset, which is the set of solutions to problems in Definitions 3 and 4 (see Remark 3.1.2).

Definition 5. *For a matrix B from (3.1), define a null-space projector $B_\perp \in \mathbb{R}^{n\times n}$ with following properties*

1) $B_\perp B = 0$

2) $B_\perp = B_\perp^T$ *(from 1) and 2) follows* $B^T B_\perp = 0$*)*

3) rank($B_\perp$) $= n - m$

The null-space projector can be designed as

$B_\perp = (I - B(B^T B)^{-1} B^T)$.

We are ready to introduce the main result of this section.

Theorem 2 (Determining the Expert Subset)**.** *Let Assumption 3.1.2 hold and let $(\mathcal{Q}^*, \mathcal{P}^*)$ be one solution to (3.6). Then, the Expert Subset is given as*

$$\begin{aligned} \mathcal{P}(L) &= \mathcal{P}^* + \Delta\mathcal{P} \\ where \quad \Delta\mathcal{P} &\triangleq B_\perp L = L^T B_\perp \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} \mathcal{Q}(L) &= \mathcal{Q}^* + \Delta\mathcal{Q} \\ where \;\; \Delta\mathcal{Q} &\triangleq -(\Delta\mathcal{P} A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T \Delta\mathcal{P}) \end{aligned} \tag{3.9}$$

where $B_\perp$ is the projector to the null-space of matrix B^T and L is any matrix of appropriate dimensions that satisfies (3.8). Moreover, $(\mathcal{Q}(L), \mathcal{P}(L))$ is also a solution to (3.7).

Proof The Expert Subset from Definition 2 consists of every $(\mathcal{Q}, \mathcal{P})$ that corresponds to the Expert Policy $\mathcal{K}^*$ in the sense of equation (3.6). Consider two different pairs from the Expert Subset and denote them as $(\mathcal{Q}_1, \mathcal{P}_1)$ and $(\mathcal{Q}_2, \mathcal{P}_2)$. Since both pairs belong to the Expert Subset, then $\mathcal{K}^* = B^T\mathcal{P}_1$ and $\mathcal{K}^* = B^T\mathcal{P}_2$. Subtract these two expressions

$$\begin{aligned} \mathcal{K}^* - \mathcal{K}^* &= B^T\mathcal{P}_1 + B^T\mathcal{P}_2 \\ 0 &= B^T(\mathcal{P}_1 - \mathcal{P}_2) \end{aligned} \tag{3.10}$$

Note that the difference between any two matrices $\mathcal{P}_1$ and $\mathcal{P}_2$ from the Expert Subset is a symmetric matrix that lies in the null-space of matrix B^T . Then the difference can be expressed as

$$\Delta\mathcal{P} \triangleq \mathcal{P}_1 - \mathcal{P}_2 = B_\perp^T L = B_\perp L \tag{3.11}$$

for some matrix L that ensures $B_\perp L = L^T B_\perp$. Therefore, if $\mathcal{P}^*$ is one solution, then any other solution is given by (3.8) for appropriate matrix L. This proves (3.8).

Next, insert (3.8) in the equation (3.7)

$$\begin{aligned}\mathcal{Q} = &- ((\mathcal{P}^* + B_\perp L)^T A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T (\mathcal{P}^* + B_\perp L) \\ &+ (\mathcal{P}^* + B_\perp L)^T B B^T (\mathcal{P}^* + B_\perp L)) \\ = &- (\mathcal{P}^* A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T \mathcal{P}^* + \mathcal{P}^* B B^T \mathcal{P}^* \\ &+ (B_\perp L)^T A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T B_\perp L + \mathcal{P}^* B B^T B_\perp L \\ &+ (B_\perp L)^T B B^T B_\perp L + L^T B_\perp B B^T \mathcal{P}^*)\end{aligned} \tag{3.12}$$

Note that $B^T B_\perp = 0$ and write

$$\begin{aligned}\mathcal{Q} = &- (\mathcal{P}^* A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T \mathcal{P}^* + \mathcal{P}^* B B^T \mathcal{P}^* \\ &+ (B_\perp L)^T A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T B_\perp L)\end{aligned} \tag{3.13}$$

The first three terms constitute a solution Q^* from (3.7)

$$\mathcal{Q}^* = -(\mathcal{P}^* A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T \mathcal{P}^* + \mathcal{P}^* B B^T \mathcal{P}^*) \tag{3.14}$$

Then (3.13) becomes

$$\mathcal{Q} = \mathcal{Q}^* - ((B_\perp L)^T A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T B_\perp L) \tag{3.15}$$

Use $\Delta\mathcal{P} = B_\perp L$ from (3.8) and insert it into (3.15) to prove (3.9).

Finally, $(\mathcal{Q}(L), \mathcal{P}(L))$ is also a solution to (3.7) as a consequence of Lemma 3. □

Let the matrix L satisfy $B_\perp L = L^T B_\perp$. Let the Expert Policy $\mathcal{K}^* = B^T \mathcal{P}$ optimize the performance index (by Assumption 3.1.2)

$$x(t)^T \mathcal{P} x(t) = \min_K \int_t^\infty x^T(\tau)\{\mathcal{Q} + K^T K\} x(\tau) d\tau \tag{3.16}$$

for some $(\mathcal{Q}, \mathcal{P})$ with corresponding Riccati equation

$$A^T\mathcal{P} + \mathcal{P}A + \mathcal{Q} - \mathcal{P}BB^T\mathcal{P} = 0 \tag{3.17}$$

Then, $\mathcal{K}^*$ also optimizes a family of performance indices where a specific element of the family can be obtained by varying L

$$\begin{aligned} &x(t)^T\tilde{\mathcal{P}}x(t) = \\ &\min_K \int_t^\infty x^T(\tau)\{\mathcal{Q} - B_\perp LA - A^TL^TB_\perp + K^TK\}x(\tau)d\tau \end{aligned} \tag{3.18}$$

Moreover, the family of performance indices (3.18) corresponds to a family of Riccati equations

$$A^T\tilde{\mathcal{P}} + \tilde{\mathcal{P}}A + \mathcal{Q} - B_\perp LA - A^TL^TB_\perp - \tilde{\mathcal{P}}BB^T\tilde{\mathcal{P}} = 0 \tag{3.19}$$

Proof. Start by observing that the optimization problem in (3.16) is equivalent to the problem of solving Riccati equation (3.17) and then forming the optimal policy $\mathcal{K}^* = B^T\mathcal{P}$. Add $\pm B_\perp LA \pm A^TL^TB_\perp$ to (3.17) and write

$$\begin{aligned} &A^T(\mathcal{P} + B_\perp L) + (\mathcal{P} + B_\perp L)A \\ &\qquad +\{\mathcal{Q} - B_\perp LA - A^TL^TB_\perp\} - \mathcal{P}BB^T\mathcal{P} = 0 \end{aligned} \tag{3.20}$$

denote $\tilde{\mathcal{P}} = \mathcal{P} + B_\perp L$ and use $\mathcal{P}BB^T\mathcal{P} = \tilde{\mathcal{P}}BB^T\tilde{\mathcal{P}}$ to write the result in (3.19).

By standard optimal control theory, the policy that optimizes (3.18) (let us denote it as $\tilde{\mathcal{K}}^*$) is given by $\tilde{\mathcal{K}}^* = B^T\tilde{\mathcal{P}}$ where $\tilde{\mathcal{P}}$ is the solution to (3.19). However, since $\tilde{\mathcal{P}} = \mathcal{P} + B_\perp L$ we conclude that $\tilde{\mathcal{K}}^* = \mathcal{K}^*$. Therefore, the Expert Policy $\mathcal{K}^*$ is optimal for both performance indices (3.18) and (3.16). This completes the proof. □

3.3 Model-Based Inverse Reinforcement Learning Algorithms

In this section we propose Algorithms 3.1 and 3.2 to solve the Model-Based Inverse RL problem defined in Definition 3. The problem is considered to be solved when the algorithm converges to the Expert Subset given by (3.7), or alternatively (3.6). The first algorithm is based on repeated solutions to Riccati equation and the second is based on Value Iteration, that is repeated solutions of Lyapunov equations.

3.3.1 Model-Based Inverse Reinforcement Learning - Riccati Iterations

In this subsection, we present the first model-based algorithm and analyze its convergence.

Algorithm 3.1 ***Model-Based Inverse Reinforcement Learning - Riccati Iterations***

Input: Symmetric cost matrix Q_0

Output: Symmetric matrix Q_∞ that belongs to the Expert Subset of the Expert Policy $u^*(t) = -\mathcal{K}^* x(t)$ (see Definition 2).

Initialization: Initialize P_0 as the solution to Riccati equation in (3.5) with $Q = Q_0$; $i \leftarrow 0$; Select the convergence tolerance $\epsilon \in \mathbb{R}^+$

1: **while** $||K_i - K_{i-1}|| = ||B^T P_i - B^T P_{i-1}|| > \epsilon$ **do**

2: **Inverse Optimal Control Step.** Estimate Q_i by using equation (3.7)

$$Q_i = -(P_i A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T P_i + P_i B B^T P_i) \tag{3.21}$$

3: **Optimal Control Step -** ***(Riccati equation)*.** Solve (3.5) for P_{i+1} given $Q = Q_i$

$$0 = P_{i+1} A + A^T P_{i+1} + Q_i - P_{i+1} B B^T P_{i+1} \tag{3.22}$$

The algorithm can be expressed in the form of the matrix iteration by inserting (3.21) into (3.22)

$$\begin{aligned} 0 &= P_{i+1} A + A^T P_{i+1} - P_{i+1} B B^T P_{i+1} \\ &+ \{-P_i B B^T P_i - P_i A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T P_i\} \end{aligned} \tag{3.23}$$

Note that the Inverse RL problem is solved in Algorithm 3.1 by interleaved steps based on Inverse Optimal Control and Optimal Control. As such, Algorithm 3.1 clarifies the relation between

Inverse RL and Inverse Optimal Control.

The recursive matrix iteration (3.23) provides useful intuition about the Inverse Reinforcement Learning procedure. First note that the iteration has the form of Riccati Equation (3.5) with $Q = -\{P_i B B^T P_i + P_i A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T P_i\}$. Therefore, model-based Inverse RL procedure in Algorithm 3.1 iteratively solves Riccati equations, where Q_i is adapted to satisfy Riccati equation (3.7) specialized to Expert Dynamics $A_{\mathcal{K}^*} = A - B\mathcal{K}^*$.

Theorem 3 (Fixed Point of Algorithm 3.1)**.** *The fixed point of Algorithm 3.1 belongs to the Expert Subset defined in Definition 2.*

Proof. First note that if the initial P_0 is symmetric, then the update rule (3.23) preserves the symmetry of $P_i, \forall i$.

Next, we analyze the fixed point of (3.23). To do the fixed point analysis, take $P_i = P_{i+1} = P_\infty$ and write (3.23) as

$$\begin{aligned} 0 &= P_\infty A + A^T P_\infty - P_\infty B B^T P_\infty \\ &\quad + \{-P_\infty B B^T P_\infty - P_\infty A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T P_\infty\} \\ 0 &= -P_\infty B B^T P_\infty - P_\infty B B^T P_\infty \\ &\quad + P_\infty B B^T \mathcal{P}^* + \mathcal{P}^* B B^T P_\infty \\ 0 &= \{\mathcal{P}^* - P_\infty\} B B^T P_\infty + P_\infty B B^T \{\mathcal{P}^* - P_\infty\} \end{aligned} \tag{3.24}$$

Then, by inserting

$$P_\infty = \mathcal{P}^* + B_\perp^T L \tag{3.25}$$

into (3.24), we confirm that (3.25) is the fixed point of the algorithm. Moreover, by Theorem 2, (3.25) belongs to the Expert Subset (3.6).

Next, insert (3.25) into steady state form of equation (3.21) to derive

$$Q_\infty(L) = \mathcal{Q}^* - (L^T B_\perp A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T B_\perp^T L) \tag{3.26}$$

which confirms that Q_i in Algorithm 3.1 converges to (3.9), that is the Expert Subset.

Finally, since $P_\infty(L)$ in (3.25) and $Q_\infty(L)$ in (3.26) are in the Expert Subset, then by Definition 3.5, they reconstruct the Expert Policy. This can also be confirmed by writing $K_\infty = B^T P_\infty(L) = B^T(\mathcal{P}^* + B_\perp^T L) = B^T\mathcal{P}^* = \mathcal{K}^*$ □

When Algorithm 3.1 converges, then the final value lies in the Expert Subset $(\mathcal{Q}, \mathcal{P})$ defined in Definition 2. Therefore, Algorithm 3.1 solves the model-based Inverse Reinforcement Learning problem as defined in Definition 3.

3.3.2 Model-Based Inverse Reinforcement Learning - Value Iteration

In this subsection the Algorithm 3.2 is constructed by modifying Algorithm 3.1. The motivation behind the development of this algorithm is that the Optimal Control step in Algorithm 3.1 requires solution to the Riccati equation at each iteration. We wish to simplify that step by replacing the quadratic equation with linear one.

The key idea is to replace second order Riccati equation in (3.22) with linear Lyapunov equation. First note that $K_{i+1} = B^T P_{i+1}$. Let us estimate K_{i+1} using previous P_i as follows

$$K_{i+1} = B^T P_i \tag{3.27}$$

Next, rewrite the equation (3.22) as

$$\begin{aligned} 0 = {} & P_{i+1}(A - BK_{i+1}) + (A - BK_{i+1})^T P_{i+1} \\ & + Q_i + K_{i+1}^T K_{i+1} \end{aligned} \tag{3.28}$$

Finally, approximate K_{i+1} using (3.27) and use $A_{K_i} \triangleq A - BK_i$ to derive the Lyapunov equation

$$0 = P_{i+1}A_{K_i} + A_{K_i}^T P_{i+1} + Q_i + K_i^T K_i \tag{3.29}$$

The final procedure is given in Algorithm 3.2

Algorithm 3.2 ***Model-Based Inverse Reinforcement Learning - Value Iterations***

Input: Symmetric cost matrix Q_0

Output: Symmetric matrix Q_∞ that belongs to the Expert Subset of the Expert Policy $u^*(t) = -\mathcal{K}^* x(t)$ (see Definition 2).

Initialization: Initialize P_0 as the solution to Riccati equation in (3.5) with $Q = Q_0$; $i \leftarrow 0$; Select the convergence tolerance $\epsilon \in \mathbb{R}^+$

1: **while** $||K_i - K_{i-1}|| = ||B^T P_i - B^T P_{i-1}|| > \epsilon$ **do**

2: **Inverse Optimal Control Step.** Estimate Q_i by using equation (3.7)

$$Q_i = -(P_i A_{\mathcal{K}^*} + A_{\mathcal{K}^*}^T P_i + P_i B B^T P_i) \tag{3.30}$$

3: **Optimal Control Step -** ***(Lyapunov equation).*** Solve (3.5) for P_{i+1} given $Q = Q_i$

$$0 = P_{i+1} A_{K_i} + A_{K_i}^T P_{i+1} + Q_i + K_i^T K_i \tag{3.31}$$

Combining two steps of Algorithm 3.2 gives a single-step version of the algorithm

$$P_{i+1} A_{K_i} + A_{K_i}^T P_{i+1} + \{-P_i A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T P_i\} = 0 \tag{3.32}$$

Lemma 4 (Fixed Point of Algorithm 3.2)**.** *The fixed point of the Algorithm 3.2 belongs to the Expert Subset defined in Definition 2.*

Proof. In convergence $P_i \to P_\infty$. Then (3.32) becomes

$$\begin{aligned}
0 =& P_\infty A_\infty + A_\infty^T P_\infty + \{-P_\infty A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T P_\infty\} \\
0 =& P_\infty (A_\infty - A_{\mathcal{K}^*}) + (A_\infty - A_{\mathcal{K}^*})^T P_\infty \\
0 =& P_\infty (A - B K_\infty - (A - B\mathcal{K}^*)) \\
&+ (A - B K_\infty - (A - B\mathcal{K}^*))^T P_\infty \\
0 =& P_\infty (-B K_\infty + B\mathcal{K}^*) + (-B K_\infty + B\mathcal{K}^*)^T P_\infty
\end{aligned} \tag{3.33}$$

Express the Expert gain as $\mathcal{K}^* = B^T \mathcal{P}^*$ where $\mathcal{P}^*$ is one solution from the set of solutions to (3.7)

and insert it in (3.33) to get

$$0 = P_\infty BB^T(\mathcal{P}^* - P_\infty) + (\mathcal{P}^* - P_\infty)^T BB^T P_\infty \tag{3.34}$$

Comparing (3.34) with (3.24) shows that the Algorithm 3.2 has the same fixed point as Algorithm 3.1. Therefore, the statement in Theorem 3 about the fixed-point of Algorithm 3.1 applies to Algorithm 3.2 as well. This proves the statement in Lemma 4. □

In Theorem 4 of this section, we will prove that Algorithm 3.2 converges. But first, we need to introduce few lemmas and definitions that are used in Theorem 4. Next definition introduces the Kronecker algebra notation which is heavily used in the remainder of the chapter.

Definition 6 (Kronecker Algebra Notation)**.**

i) Let $S \in \mathbb{R}^{N\times M}$ be a general matrix of arbitrary size. Then, vertical stacking of columns of matrix S gives a vector $\overrightarrow{S} \in \mathbb{R}^{NM\times 1}$ where symbol $\rightarrow$ is used to denote the operation of vertical stacking and is referred to as "vectorization".

ii) Let $S \otimes Z$ denote the standard Kronecker Product between matrices S and Z.

iii) Let $S \oplus Z = S \otimes I + I \otimes Z$ denote the standard Kronecker Sum between matrices S and Z.

In order to argue about convergence of Algorithm 3.2 we introduce the error matrix in Definition 7.

Definition 7. *Let us define the error matrix $E_i \in \mathbb{R}^{n\times n}$ as*

$$E_i \triangleq P_i - \mathcal{P}^* \ , \quad \overrightarrow{E}_i \triangleq \overrightarrow{P}_i - \overrightarrow{\mathcal{P}}^* \tag{3.35}$$

Moreover, let us introduce normalized error $\xi_i \in \mathbb{R}^{n\times n}$ as

$$\xi_i = \frac{1}{||E_i||} E_i \ , \quad \overrightarrow{\xi}_i = \frac{1}{||E_i||} \overrightarrow{E}_i \tag{3.36}$$

The matrix iteration given in (3.32) can now be expressed using the error matrix by inserting

$P_i = \mathcal{P}^* + E_i$

$$\begin{aligned} 0 =&(E_{i+1} + \mathcal{P}^*)A_{K_i} + A_{K_i}^T(E_{i+1} + \mathcal{P}^*) - \\ &(E_i + \mathcal{P}^*)A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T(E_i + \mathcal{P}^*) \\ =&E_{i+1}A_{K_i} + A_{K_i}^T E_{i+1} - E_i A_{\mathcal{K}^*} - A_{\mathcal{K}^*}^T E_i \\ &-\mathcal{P}^* BB^T E_i - E_i BB^T \mathcal{P}^* \end{aligned} \tag{3.37}$$

Which results in

$$E_{i+1}A_{K_i} + A_{K_i}^T E_{i+1} = E_i A + A^T E_i \tag{3.38}$$

It can also be written in vectorized form

$$\left(A_{K_i}^T \oplus A_{K_i}^T\right)\overrightarrow{E}_{i+1} = \left(A^T \oplus A^T\right)\overrightarrow{E}_i \tag{3.39}$$

Multiplying by $\left(A_{K_i}^T \oplus A_{K_i}^T\right)^{-1}$ on the left yields $\overrightarrow{E}_{i+1}$ as an explicit function of $\overrightarrow{E}_i$. This is formalized in Definition 8.

Definition 8. *Let the operator* $\Psi(\cdot) \ : \ \mathbb{R}^{n^2\times 1} \rightarrow \mathbb{R}^{n^2\times 1}$ *be defined as*

$$\Psi(E_i) \triangleq \left(A_{K_i}^T \oplus A_{K_i}^T\right)^{-1}\left(A^T \oplus A^T\right) \tag{3.40}$$

Then, the error matrix iteration in (3.38) can be succinctly expressed as

$$\overrightarrow{E}_{i+1} = \Psi(\overrightarrow{E}_i)\overrightarrow{E}_i \tag{3.41}$$

Note that $\Psi(E_i)$ in (3.40) depends implicitly on E_i since $A_{K_i} = A - BB^T P_i = A - BB^T(\mathcal{P}^* + E_i)$. We will continue to use the notation in (3.40) in favor of readability. The convergence properties of Algorithm 3.2 can now be studied by analyzing the operator $\Psi(\overrightarrow{E}_i)$. Before that, we need Lemma 5 which is used in that analysis.

Lemma 5. *Consider $\xi \in \mathbb{R}^{n^2 \times 1}$. Then, following inequalities hold*

$$\left|\xi^T \big(I \otimes A_{\mathcal{K}^*}\big)\xi\right| > \left|\xi^T \big(I \otimes A\big)\xi\right| \tag{3.42}$$

$$\left|\xi^T \big(A_{\mathcal{K}^*} \otimes I\big)\xi\right| > \left|\xi^T \big(A \otimes I\big)\xi\right| \tag{3.43}$$

Proof. Proof in the Section 3.7 □

The convergence of Algorithm 3.2 is related to certain properties of operator $\Psi(\cdot)$. Next lemma is used to introduce those properties.

Lemma 6 (Properties of Operator Ψ)**.**

The following statements hold for operator $\Psi(\cdot)$ defined in (3.40)

$$A) \quad \Psi(0) = (A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-T}(A \oplus A)^T \tag{3.44}$$

$$B) \quad \Psi(\overrightarrow{E}_i) = \Big(I - \big((BB^T E_i) \oplus (BB^T E_i)\big) \cdot (A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\Big)^{-T} \Psi(0)$$

Let $\xi \in \mathbb{R}^{n^2 \times 1}$ and $\mu \in \mathbb{R}$, then following implication holds

$$C) \quad \Psi(0)\xi = \mu\xi \;\Rightarrow\; |\mu| < 1 \tag{3.45}$$

Proof. The proof is in the Section 3.7. □

Theorem 4 (Convergence of Algorithm 3.2)**.** *The Algorithm 3.2 converges to the Expert Subset and thereby solves Model Based Reinforcement Learning Problem as defined in Definition 3.*

Proof. We need to show that the sequence of error matrices $\{E_i\}_{i=0}^{\infty}$ produced by iteration (3.41) converges to 0. This is done by studying $\Psi(\overrightarrow{E}_i)$. Start by considering $\omega \in \mathbb{R}$ and E_i such that

$$\Psi(\overrightarrow{E}_i)\overrightarrow{E}_i = \omega \overrightarrow{E}_i \tag{3.46}$$

Use (3.36) to write

$$\big(I - ||E_i||\big((BB^T\xi_i) \oplus (BB^T\xi_i)\big)(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\big)^{-T} \cdot \Psi(0)\overrightarrow{E}_i = \omega\overrightarrow{E}_i \tag{3.47}$$

Multiply by
$\overrightarrow{E}_i^T\big(I - ||E_i||\big((BB^T\xi_i) \oplus (BB^T\xi_i)\big)(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\big)$ on the left side

$$\overrightarrow{E}_i^T\Psi(0)\overrightarrow{E}_i = \omega\overrightarrow{E}_i^T\big(I - ||E_i||\big((BB^T\xi_i) \oplus (BB^T\xi_i)\big) \cdot (A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\big)\overrightarrow{E}_i \tag{3.48}$$

Then we can express $|\omega|$

$$|\omega| = \frac{|\overrightarrow{E}_i^T\Psi(0)\overrightarrow{E}_i|}{\left|\overrightarrow{E}_i^T\Big(I-||E_i||\Big((BB^T\xi_i)\oplus(BB^T\xi_i)\Big)(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\Big)\overrightarrow{E}_i\right|} \tag{3.49}$$

Use (3.36) and cancel $||E_i||^2$ that will appear in the numerator and the denominator to write

$$\begin{aligned}|\omega| &= \frac{|\overrightarrow{\xi}_i^T\Psi(0)\overrightarrow{\xi}_i|}{\left|\overrightarrow{\xi}_i^T\Big(I-||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\Big)\overrightarrow{\xi}_i\right|} \\ &= \frac{|\overrightarrow{\xi}_i^T\Psi(0)\overrightarrow{\xi}_i|}{\left|1-\overrightarrow{\xi}_i^T\Big(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\Big)\overrightarrow{\xi}_i\right|}\end{aligned} \tag{3.50}$$

If $||E_i|| \to 0$ we see that $|\omega| \to |\overrightarrow{\xi}_i^T\Psi(0)\overrightarrow{\xi}_i|$. Since $||\xi_i|| = 1$ and using (3.45) we conclude that $|\omega| < 1$ holds locally.

Moreover, since $|\overrightarrow{\xi}_i^T\Psi(0)\overrightarrow{\xi}_i| < 1$, then there is a margin $\alpha \in (0, 1)$ such that

$$|\overrightarrow{\xi}_i^T\Psi(0)\overrightarrow{\xi}_i| + \alpha = 1 \tag{3.51}$$

Using this margin and using the continuity of operator $\Psi(\cdot)$ we can extend the local result by determining the non-local region of convergence. To do so, we require $|\omega| < 1$ and seek a sufficient condition for this to hold in some non-local region around $E_i = 0$. Insert (3.51) into (3.50) to write

the requirement $|\omega| < 1$ as

$$|\omega| = \frac{1-\alpha}{\left|1-\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i\right|} < 1 \tag{3.52}$$

When $\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i$ is negative, (3.52) directly holds since $1-\alpha < 1$. On the other hand, if $\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i$ is positive, then $|\omega| < 1$ holds true for $||E_i||$ that satisfies

$$\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))\cdot(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i < \alpha \tag{3.53}$$

Therefore, a sufficient condition that ensures $|\omega| < 1$ for the case of positive
$\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i$ is given by

$$||E_i|| < \frac{\alpha}{\overrightarrow{\xi}_i^T((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\overrightarrow{\xi}_i} \tag{3.54}$$

In conclusion, there is a non-local region around $E_i = 0$ where the operator $\Psi(\overrightarrow{E}_i)$ is contractive in the following sense, $|\omega| < 1$, and ω is from (3.46).

To complete the proof, we will show that $\{E_i\}_{i=0}^{\infty} < \infty$. In order to do that, we show that $\Psi(\overrightarrow{E}_i)$ is a contraction for large enough, but necessarily finite E_i. Specifically, we need to show that

$$\exists M\ ,\ M < ||E_i|| < \infty\ \Rightarrow |\omega| < 1 \tag{3.55}$$

where ω is from (3.46). Consider (3.52) and note that for large $||E_i||$ the denominator is dominated by the term
$\overrightarrow{\xi}_i^T\left(||E_i||((BB^T\xi_i)\oplus(BB^T\xi_i))(A_{\mathcal{K}^*}\oplus A_{\mathcal{K}^*})^{-1}\right)\overrightarrow{\xi}_i$.
Formally, if

$$||E_i|| \cdot |\overrightarrow{\xi}_i^T((BB^T\xi_i) \oplus (BB^T\xi_i)) \cdot (A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\overrightarrow{\xi}_i| >> 1 \tag{3.56}$$

then the requirement $|\omega| < 1$ in (3.52) is met. Therefore,

$$|\overrightarrow{\xi}_i^T((BB^T\xi_i) \oplus (BB^T\xi_i))(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-1}\overrightarrow{\xi}_i| << ||E_i|| < \infty \Rightarrow |\omega| < 1 \quad (3.57)$$

We conclude that $\{||E_i||\}_{i=0}^{\infty} < \infty$.

In conclusion, considering the contractive properties of the Algorithm 3.2 as well as the fact that it is globally bounded, it follows that Algorithm 3.2 converges to its fixed point. Moreover, since the fixed point of Algorithm 3.2 is shown to be the element of the Expert Subset by Lemma 4, then Algorithm 3.2 solves the Model Based Inverse Reinforcement Learning Problem as defined in Definition 3. □

In Section 3.5, we show by simulation that Algorithm 3.2 converges.

Model-based Algorithms 3.1 and 3.2 provide a mathematical formalism for learning cost function in continuous time. However, these algorithms can also be understood intuitively as the Expert-Learner interaction. Concretely, in the Inverse Optimal Control steps (3.21) and (3.30), the Learner estimates the cost function by observing the Expert $A_{\mathcal{K}^*}$. After the cost function is estimated, Learner develops a policy which optimizes newly estimated cost function (steps (3.22) and (3.31)). This cycle repeats until Learner develops the Expert Policy by means of optimizing current iterate of cost function.

Often times, the information about the Expert is available in the form of state trajectories $x_{\mathcal{K}^*}(t) \triangleq \{x(t) \mid \dot{x}(t) = A_{\mathcal{K}^*}x(t), \ t \in (0, \infty)\}$ instead of closed loop dynamics $A_{\mathcal{K}^*}$. In next section, we show how the problem can be solved given the Expert Trajectories.

3.4 Data-Driven algorithm

In this section we introduce data-driven equivalent of model-based Algorithm 3.2. The data-driven Inverse Reinforcement Learning problem is defined in Definition 4.

The technique we use to derive the data-driven algorithm is known as Integral Reinforcement Learning [112]. This technique is used in Reinforcement Learning literature to solve Riccati equa-

tion using Expert trajectories. We employ a similar technique to derive data-driven equivalent of model-based Algorithm 3.2 and we call it Integral Inverse Reinforcement Learning (Integral Inverse RL).

3.4.1 Integral Inverse Reinforcement Learning

Model-based Algorithm 3.2 has 2 steps in each iteration. In this subsection, we will convert both steps into a data-driven form using Integral Reinforcement Learning technique [112].

We will start with Inverse Optimal Control step (3.30).

Theorem 5 (Data-Driven Inverse Optimal Control Step)**.** *Assume that P_i, K_i and Expert trajectory $x_{\mathcal{K}^*}(t)$ are given. Then the solution Q_i to equation (3.30) is equivalent to the solution Q_i of the equation*

$$\begin{aligned} &x_{\mathcal{K}^*}^T(t+T)P_i x_{\mathcal{K}^*}(t+T) - x_{\mathcal{K}^*}^T(t)P_i x_{\mathcal{K}^*}(t) \\ &= -\int_t^{t+T} x_{\mathcal{K}^*}^T(\tau)\Big(Q_i + K_i^T K_i\Big)x_{\mathcal{K}^*}(\tau)d\tau \end{aligned} \tag{3.58}$$

Proof. Multiply equation (3.30) by $x_{\mathcal{K}^*}(\tau)$ from both sides

$$\begin{aligned} 0 = &x_{\mathcal{K}^*}^T(\tau)P_i A_{\mathcal{K}^*}x_{\mathcal{K}^*}(\tau) + x_{\mathcal{K}^*}^T(\tau)A_{\mathcal{K}^*}^T P_i x_{\mathcal{K}^*}(\tau) \\ &+ x_{\mathcal{K}^*}^T(\tau)Q_i x_{\mathcal{K}^*}(\tau) + x_{\mathcal{K}^*}^T(\tau)K_i^T K_i x_{\mathcal{K}^*}(\tau) \end{aligned} \tag{3.59}$$

Note that $A_{\mathcal{K}^*}x_{\mathcal{K}^*}(\tau) = \dot{x}_{\mathcal{K}^*}(\tau)$. Use this expression to write (3.59) as

$$\begin{aligned} 0 = &x_{\mathcal{K}^*}^T(\tau)P_i\dot{x}_{\mathcal{K}^*}(\tau) + \dot{x}_{\mathcal{K}^*}^T(\tau)P_i x_{\mathcal{K}^*}(\tau) + x_{\mathcal{K}^*}^T(\tau)Q_i x_{\mathcal{K}^*}(\tau) \\ &+ x_{\mathcal{K}^*}^T(\tau)K_i^T K_i x_{\mathcal{K}^*}(\tau) \end{aligned} \tag{3.60}$$

Next, note that first two terms in (3.60) can be expressed as $\frac{d}{d\tau}\Big(x_{\mathcal{K}^*}^T(\tau)P_i x_{\mathcal{K}^*}(\tau)\Big) = x_{\mathcal{K}^*}^T(\tau)P_i\dot{x}_{\mathcal{K}^*}(\tau) + \dot{x}_{\mathcal{K}^*}^T(\tau)P_i x_{\mathcal{K}^*}(\tau)$. Then (3.60) becomes

$$\begin{aligned} 0 = &\frac{d}{d\tau}\Big(x_{\mathcal{K}^*}^T(\tau)P_i x_{\mathcal{K}^*}(\tau)\Big) \\ &+ x_{\mathcal{K}^*}^T(\tau)Q_i x_{\mathcal{K}^*}(\tau) + x_{\mathcal{K}^*}^T(\tau)K_i^T K_i x_{\mathcal{K}^*}(\tau) \end{aligned} \tag{3.61}$$

Integrate over the interval $[t, t+T]$ to recover (3.58) $\square$

The importance of this result is that the data-driven Inverse Optimal Control step in (3.58) uses the Expert trajectory $x_{\mathcal{K}^*}(t)$ when it updates Q_i in (3.58). This allows us to replace the model of Expert $A_{\mathcal{K}^*}$ in the Inverse Optimal Control Step in (3.30), by its trajectories $x_{\mathcal{K}^*}(\tau)$ as shown in data-driven Inverse Optimal Control Step (3.58).

Next, we need to derive the data-driven Optimal Control step. We show that the data-driven Optimal Control is equivalent to the standard Value Iteration step in the Reinforcement Learning literature. Specifically, we develop off-policy algorithm, where the trajectory used for learning does not necessarily have to be generated by Learner's policy. Instead, any stabilizing policy can be used to collect the data trajectories. The policy selected for data collection will be referred to as the Behavior Policy. We first introduce the corresponding notation in next definition.

Definition 9 (Behavior Policy and Trajectory)**.** *Let the Behavior Dynamics be given as*

$$\dot{x} = Ax(t) + Bu_b(t) \tag{3.62}$$

where $u_b(t)$ is the Behavior Policy. Then, the Behavior Trajectory is given as $x_b(t) \triangleq \{x(t) \mid \dot{x}(t) = Ax(t) + Bu_b(t),\ t \in (0, \infty)\}$.

The Behavior Policy is the policy that is applied to the system during Optimal Control step. As such, it generates Behavior Trajectory which is used in data-driven Optimal Control step of our algorithm. The Learner's Policy $u_i = K_i x(t)$ can be selected as Behavior Policy, but it need not be. In other words, the choice of Behavior Policy is arbitrary as long as it is stabilizing. In the next theorem we derive data-driven version of the Optimal Control Step in (3.31), where data from the Behaviour Trajectory is used to obtain the solution.

Theorem 6 (Data-Driven Optimal Control Step)**.** *Assume that Q_i and K_i are given. Then, the so-*

lutions P_{i+1} and K_{i+1} to the equation (3.31) are equivalent to the solution of data-driven equation

$$\begin{aligned} x_b^T(t+T)P_{i+1}x_b(t+T) - x_b^T(t)P_{i+1}x_b(t) \\ = -\int_t^{t+T} x_b^T(\tau)\big(Q_i + K_i^T K_i\big)x_b(\tau)d\tau \\ + 2\int_t^{t+T} x_b^T(\tau)\big(K_{i+1}^T(u_b(t) + K_i x_b(\tau))\big)d\tau \end{aligned} \quad (3.63)$$

evaluated along the Behavior Trajectory $x_b(t)$.

Proof. Consider the Behavior Dynamics in (3.62) and write the alternative expression as

$$\dot{x}(t) = Ax(t) + Bu_b(t) \pm BK_i x(t) = (A - BK_i)x(t) + B(u_b(t) + K_i x(t)) \quad (3.64)$$

Consider the value function $x^T(t)P_{i+1}x(t)$ and differentiate it along the trajectories generated by Behavior Policy defined in Definition 9.

$$\frac{d}{dt}x^T(t)P_{i+1}x(t) = x^T(t)P_{i+1}\dot{x}(t) + \dot{x}^T(t)P_{i+1}x(t) = 2x^T(t)P_{i+1}\dot{x}(t) \quad (3.65)$$

Inserting (3.64) into (3.65) gives

$$\frac{d}{dt}x^T(t)P_{i+1}x(t) = 2x^T(t)P_{i+1}\big(A_{K_i}x(t) + B(u_b(t) + K_i x(t))\big) \quad (3.66)$$

Next, we use the model-based Optimal Control step in (3.31) to estimate $x(t)^T P_{i+1}A_{K_i}x(t)$ as

$$\begin{aligned} x(t)^T P_{i+1}A_{K_i}x(t) + x(t)^T A_{K_i}^T P_{i+1}x(t) \\ = -x(t)^T Q_i x(t) - x(t)^T K_i^T K_i x(t) \\ \Rightarrow 2x(t)^T P_{i+1}A_{K_i}x(t) = -x(t)^T Q_i x(t) - x(t)^T K_i^T K_i x(t) \end{aligned} \quad (3.67)$$

Insert (3.67) into equation (3.66) to write

$$\frac{d}{dt}x^T(t)P_{i+1}x(t) \;=\; -x(t)^TQ_ix(t) \,-\, x(t)^TK_i^TK_ix(t) \,+\, 2x^T(t)P_iB\big(u_b(t) \,+\, K_ix(t)\big) \quad (3.68)$$

Use approximation in (3.27) to write equation (3.68) as

$$\frac{d}{dt}x^T(t)P_{i+1}x(t) \;=\; -x(t)^TQ_ix(t) - x(t)^TK_i^TK_ix(t) + 2x^T(t)K_{i+1}\big(u_b(t) + K_ix(t)\big) \quad (3.69)$$

Integrate (3.69) along the Behavior trajectory $x_b(t)$, defined in Definition 9, over time horizon of length T to write the result in (3.63). This completes the proof. □

Theorem 6 completes the conversion of model-based Algorithm 3.2 into a data-driven Algorithm 3.3. Now, we introduce this algorithm.

Algorithm 3.3 ***Data-Driven Inverse Reinforcement Learning - Value Iterations***

Input: Symmetric cost matrix Q_0

Output: Symmetric matrix Q_∞ that belongs to the Expert Subset of the Expert Policy $u^*(t) = -\mathcal{K}^* x(t)$ (see Definition 2).

Initialization: Initialize any stabilizing K_0 and $Q_0 = Q_0^T$; $i \leftarrow 0$; Select the convergence tolerance $\epsilon \in \mathbb{R}^+$

1: **while** $||K_i - K_{i-1}|| > \epsilon$ **do**

2: **Data-Driven Optimal Control Step.** Split the Behavior Trajectory $x_b(t)$ and control input $u_b(t)$ into N sequential trajectories that fall into intervals $[0, T], [T, 2T], ...[(N-1)T, NT]$. For each interval, form the scalar equation based on (3.63)

$$\begin{aligned} x_b^T(t+T)P_{i+1}x_b(t+T) - x_b^T(t)P_{i+1}x_b(t) \\ = -\int_t^{t+T} x_b^T(\tau)\big(Q_i + K_i^T K_i\big)x_b(\tau)d\tau \\ + 2\int_t^{t+T} x_b^T(\tau)\big(K_{i+1}^T(u_b(t) + K_i x_b(\tau))\big)d\tau \end{aligned} \quad (3.70)$$

Given Q_i and K_i, solve the system of N equations for P_{i+1} and K_{i+1}.

3: **Data-Driven Inverse Optimal Control.** Split the Expert's Trajectory $x_{\mathcal{K}^*}(t)$ into N sequential trajectories in intervals $[0, T], [T, 2T], ...[(N-1)T, NT]$. For each interval, form the scalar equation based on (3.58)

$$\begin{aligned} x_{\mathcal{K}^*}^T(t+T)P_i x_{\mathcal{K}^*}(t+T) - x_{\mathcal{K}^*}^T(t)P_i x_{\mathcal{K}^*}(t) \\ = -\int_t^{t+T} x_{\mathcal{K}^*}^T(\tau)\Big(Q_i + K_i^T K_i\Big)x_{\mathcal{K}^*}(\tau)d\tau \end{aligned} \quad (3.71)$$

Given P_i and K_i, solve the system of N equations for Q_i.

Note that the Algorithm 3.3 does not impose any rules on the choice of $u_b(t)$. The true power of this design lies in the fact that the Learner's Value function and Policy can be updated based on the trajectory collected by arbitrary Behavior Policy $u_b(t)$. In the Reinforcement Learning literature this is known as the off-policy method.

3.4.2 Data-Driven Algorithm Implementation

In this subsection we show how to implement steps (3.70) and (3.71). We propose the actor-critic structure, whereby the actor is given as Learner's Policy $u_i(t) = K_i x(t)$ and and the critic is given as the Learner's Value function $V_i(x) \triangleq x^T P_i x$.

Start by rewriting (3.71) using Kronecker product

$$\begin{aligned}&\overrightarrow{P_i}^T\big(x_{\mathcal{K}^*}(t+T)\otimes x_{\mathcal{K}^*}(t+T)-x_{\mathcal{K}^*}(t)\otimes x_{\mathcal{K}^*}(t)\big)\\&+(\overrightarrow{Q_i}+\overrightarrow{K_i^TK_i})^T\int_t^{t+T}x_{\mathcal{K}^*}(\tau)\otimes x_{\mathcal{K}^*}(\tau)d\tau=0\end{aligned}\tag{3.72}$$

Note that (3.72) is linear in Q_i and can be expressed as

$$\overrightarrow{Q_i}^T h_t^{t+T}=y_t^{t+T}\tag{3.73}$$

where h_t^{t+T} and y_t^{t+T} are constants that can be evaluated for a given P_i, K_i along the Expert Trajectory $x_{\mathcal{K}^*}(t)$ in time interval $[t,t+T]$. This equation is imposed on every interval from the sequence of N intervals defined in Algorithm 3.3 as follows

$$\begin{aligned}\overrightarrow{Q_i}^T h_0^T&=y_0^T\\\overrightarrow{Q_i}^T h_T^{2T}&=y_T^{2T}\\&\dots\\\overrightarrow{Q_i}^T h_{NT-T}^{NT}&=y_{NT-T}^{NT}\end{aligned}\tag{3.74}$$

The system of N scalar equations can be transformed into a vector equation

$$\overrightarrow{Q_i}^T\begin{bmatrix}h_0^T\\h_T^{2T}\\\dots\\h_{NT-T}^{NT}\end{bmatrix}^T=\begin{bmatrix}y_0^T\\y_T^{2T}\\\dots\\y_{NT-T}^{NT}\end{bmatrix}^T\tag{3.75}$$

Which can be directly solved by gradient descent or, if the size of the matrix permits, by pseudo-inverse. This concludes the implementation of (3.71).

Next, we describe the implementation of (3.70) which is similar to that of (3.71). Concretely,

we express (3.70) as

$$\overrightarrow{P_{i+1}}\big(x_b(t+T)\otimes x_b(t+T)-x_b(t)\otimes x_b(t)\big) + 2\overrightarrow{K_{i+1}}\int_t^{t+T}(u_b(t)+K_i x_b(\tau))\otimes x_b(\tau)d\tau + (\overrightarrow{Q_i}+\overrightarrow{K_i^T K_i})\int_t^{t+T} x_b(\tau)\otimes x_b(\tau)d\tau = 0 \quad (3.76)$$

and note that the resulting scalar equation is linear in $\overrightarrow{P_{i+1}}$ and $\overrightarrow{K_{i+1}}$. Then

$$\begin{bmatrix}\overrightarrow{P_{i+1}} & \overrightarrow{K_{i+1}}\end{bmatrix} g_t^{t+T} = z_t^{t+T} \quad (3.77)$$

where g_t^{t+T} and z_t^{t+T} are constants that can be evaluated for given Q_i and K_i along the Behavior Trajectory $x_b(t)$ in time interval $[t, t+T]$. The system of N scalar equations in this form is given as

$$\begin{bmatrix}\overrightarrow{P_{i+1}} & \overrightarrow{K_{i+1}}\end{bmatrix}^T \begin{bmatrix} g_0^T \\ g_T^{2T} \\ \dots \\ g_{NT-T}^{NT}\end{bmatrix}^T = \begin{bmatrix} z_0^T \\ z_T^{2T} \\ \dots \\ z_{NT-T}^{NT}\end{bmatrix}^T \quad (3.78)$$

Actor and critic are directly updated by the solution to this linear equation.

3.5 Simulation Results

In this section, we simulate Algorithm 2 and Algorithm 3 on a randomly generated system dynamics and expert dynamics matrices.

First, we show simulation results for Algorithm 2. Matrices $A \in \mathbb{R}^{10\times 10}$ in (3.102) and $B = \mathbb{R}^{10\times 3}$ in (3.103) from the Section 3.7 are randomly selected matrices and the cost matrix $\mathcal{Q} = \mathbb{R}^{10\times 10}$, used to construct the Expert Policy (introduced in Assumption 1 and Definition 2), is randomly selected positive definite matrix shown in (3.100). Then, the Expert Policy is given as $\mathcal{K}^* = B^T\mathcal{P}$ where $\mathcal{P}$ solves (3.7). The initial Learner Cost Q_0 is selected as identity matrix and

Learner Value P_0 is calculated using Algebraic Riccati equation. We proceed by iterating over steps of Algorithm 2, until convergence.

The Learner's policy K_i, obtained as the optimal policy with respect to cost Q_i, is shown in Figure 3.1. Since the Learner's policy converges to the Expert's policy, then Q_∞ is in the Expert subset and solves model-based Inverse Reinforcement Learning. The convergence of Q_i is displayed in Figure 3.2.

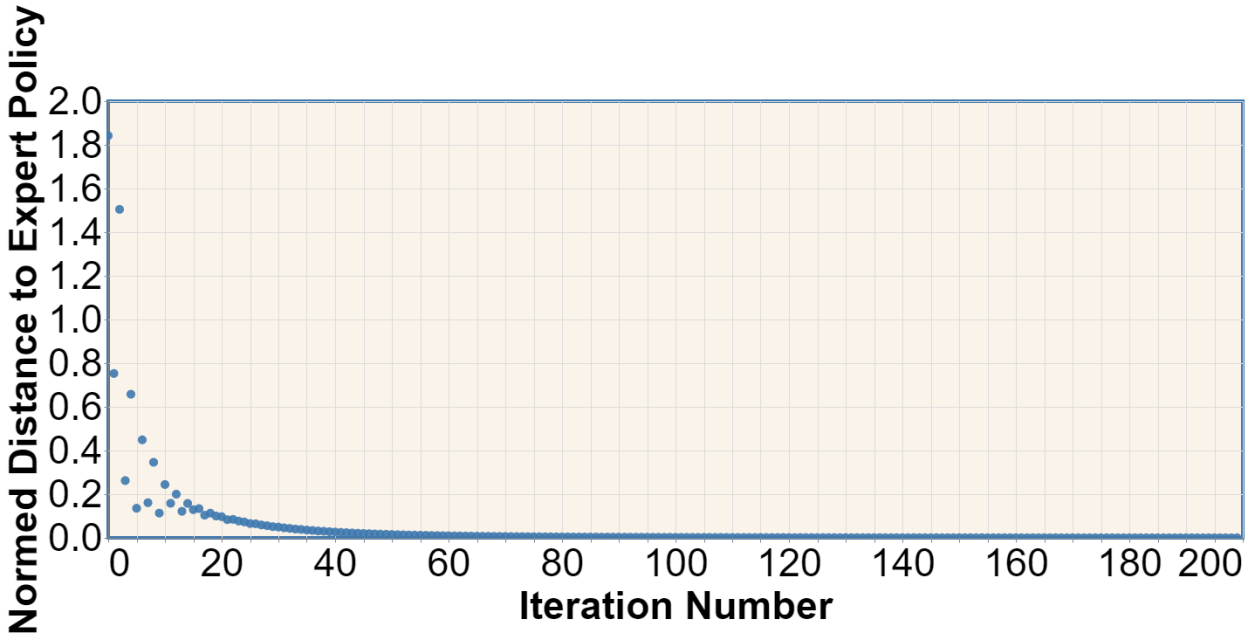

Figure 3.1: Algorithm 2; Normed distance between matrices K_i and $\mathcal{K}^*$; $||K_i - \mathcal{K}^*||$

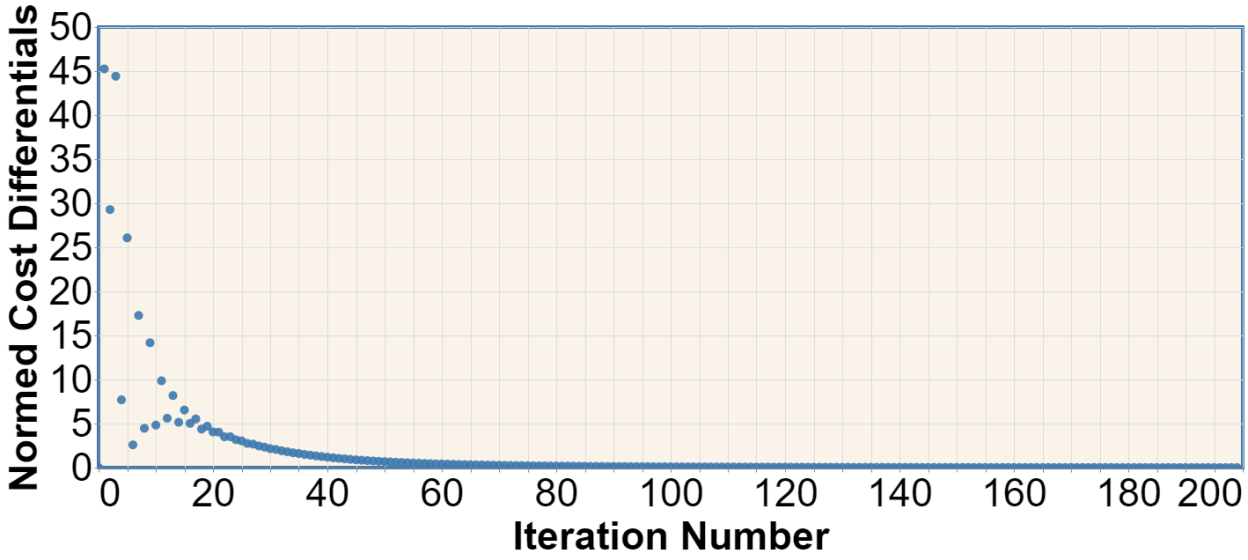

Figure 3.2: Algorithm 2; Normed distance between matrices Q_{i+1} and Q_i; $||Q_{i+1} - Q_i||$

Next, we simulate Algorithm 3.3. We use the same matrices as in model-based simulation. Before running the algorithm, we collect 10 Expert Trajectories starting from randomly selected initial conditions, each $5s$ in length. The trajectories are split into intervals of length $T = 0.05$ which makes $N = 1000$ samples. These trajectories are reused every time in (3.71). The Behaviour Policy is selected as $u_b(t) = K_i x(t) + e(t)$ where $e(t)$ is chirp signal with frequencies between $0.1Hz$ and $500Hz$. The choice of $e(t)$ is important as it excites the system and allows proper exploration of the state space, which is formally known as the persistence of excitation condition. Another option is white noise, but we have found that chirp consistently gives better results. The length of the Reinforcement Learning episode T is $0.05s$, and each step operates on 10 Behaviour Trajectories $5s$ in length. Each trajectory is initialized with randomly selected initial condition. Therefore, in each step, we collect $N = 1000$ samples from $x_b(t)$ and execute (3.70). Convergence tolerance is $\epsilon = 0.01$.

The results from data-driven simulation are shown in Figures 3.3 and 3.4 and they show that the Algorithm 3.3 converges to the Expert Subset and thereby solves data-driven Inverse Reinforcement Learning problem as defined in Definition 4. The Learner's Cost function converges to (3.101) in the Section 3.7. Solving Riccati equation with that cost yields the Expert Policy, which

numerically confirms that Q_∞ is in the Expert Subset.

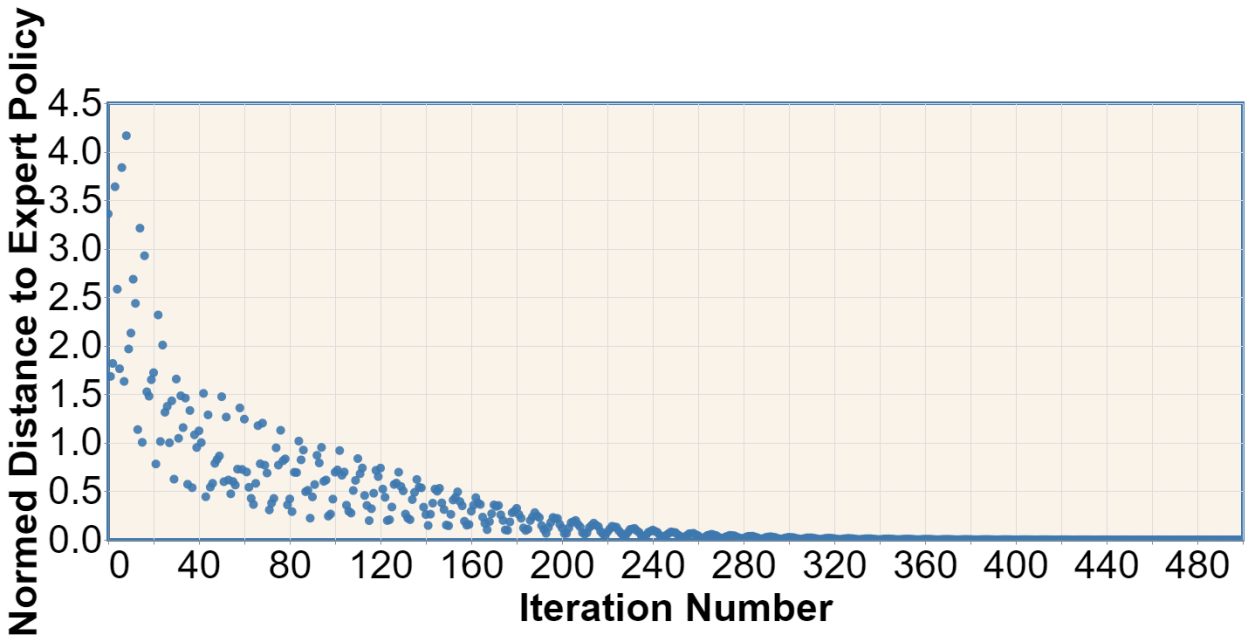

Figure 3.3: Algorithm 3; Normed distance between matrices K_i and $\mathcal{K}^*$; $||K_i - \mathcal{K}^*||$

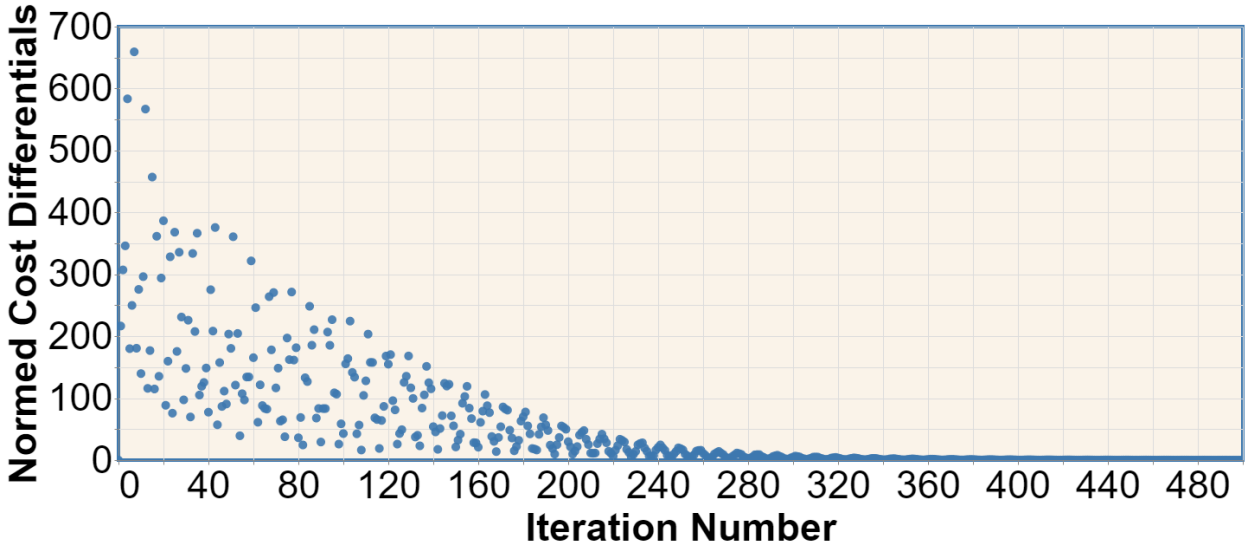

Figure 3.4: Algorithm 3; Normed distance between matrices Q_{i+1} and Q_i; $||Q_{i+1} - Q_i||$

3.6 Conclusion

In this chapter we have developed mathematical framework for solving Inverse Reinforcement Learning problem in continuous time. We analyzed the space of all possible solutions to this prob-

lem, and characterized them analytically. Then, we provided model-based procedure for solving Inverse Reinforcement Learning problem based on matrix iterations. Finally, the iterative model-based procedure is used as the backbone to develop data-driven procedure. Algorithms were tested on higher dimensional random matrices to confirm the correctness of our algorithm as well as robustness to numerical issues.

3.7 Proofs

3.7.1 Proof of Lemma 5

Proof. Let us introduce $\lambda \in \mathbb{R}$ that satisfies $\big(I \otimes A_{\mathcal{K}^*}\big)^{-1}\big(I \otimes A\big)\xi = \lambda\xi$. Then, we can write

$$\begin{aligned} \big(I \otimes A_{\mathcal{K}^*}\big)^{-1}\big(I \otimes A\big)\xi &= \lambda\xi \\ \big(I \otimes A\big)\xi &= \lambda\big(I \otimes A_{\mathcal{K}^*}\big)\xi \end{aligned} \tag{3.79}$$

Take the left multiplication of (3.79) by ξ^T, and express $|\lambda|$

$$\begin{aligned} \xi^T\big(I \otimes A\big)\xi &= \lambda\xi^T\big(I \otimes A_{\mathcal{K}^*}\big)\xi \\ \Rightarrow |\lambda| &= \frac{|\xi^T\big(I \otimes A\big)\xi|}{|\xi^T\big(I \otimes A_{\mathcal{K}^*}\big)\xi|} \end{aligned} \tag{3.80}$$

However, an alternative expression for $|\lambda|$ can be derived by multiplying (3.79) by $\xi^T(I \otimes \mathcal{P}^*)$ from left

$$\begin{aligned} \xi^T\big(I \otimes \mathcal{P}^*\big)\big(I \otimes A\big)\xi &= \lambda\, \xi^T\big(I \otimes \mathcal{P}^*\big)\big(I \otimes A_{\mathcal{K}^*}\big)\xi \\ \xi^T\big(I \otimes \mathcal{P}^*A\big)\xi &= \lambda\, \xi^T\big(I \otimes (\mathcal{P}^*A_{\mathcal{K}^*})\big)\xi \end{aligned} \tag{3.81}$$

Therefore, we can determine $|\lambda|$ as the ratio

$$\big|\lambda\big| = \frac{\big|\xi^T\big(I \otimes (\mathcal{P}^*A)\big)\xi\big|}{\big|\xi^T\big(I \otimes (\mathcal{P}^*A_{\mathcal{K}^*})\big)\xi\big|} \tag{3.82}$$

In order to determine the ratio in (3.82), we first consider Riccati Equation (3.5) and apply

Kronecker multiplication

$$\begin{aligned} I \otimes (\mathcal{P}^* A + A^T \mathcal{P}^* - \mathcal{P}^* B B^T \mathcal{P}^*) &= -I \otimes \mathcal{Q}^* \\ I \otimes (\mathcal{P}^* A + A^T \mathcal{P}^*) - I \otimes \mathcal{P}^* B B^T \mathcal{P}^* &= -I \otimes \mathcal{Q}^* \end{aligned} \tag{3.83}$$

Multiply by ξ on both sides to write

$$\begin{aligned} \xi^T (I \otimes (\mathcal{P}^* A + A^T \mathcal{P}^* - \mathcal{P}^* B B^T \mathcal{P}^*))\xi &= -\xi^T (I \otimes \mathcal{Q}^*)\xi \\ \xi^T (I \otimes (\mathcal{P}^* A + A^T \mathcal{P}^*))\xi - \xi^T (I \otimes \mathcal{P}^* B B^T \mathcal{P}^*)\xi &= -\xi^T (I \otimes \mathcal{Q}^*)\xi \end{aligned} \tag{3.84}$$

Note that $\xi^T (I \otimes (\mathcal{P}^* A))\xi = \xi^T (I \otimes (A^T \mathcal{P}^*))\xi$. Then, we can simplify (3.84)

$$2\xi^T (I \otimes (\mathcal{P}^* A))\xi = -\xi^T (I \otimes \mathcal{Q}^*)\xi + \xi^T (I \otimes \mathcal{P}^* B B^T \mathcal{P}^*)\xi \tag{3.85}$$

Next, we can express $\left|\xi^T (I \otimes (\mathcal{P}^* A))\xi\right|$ as

$$\left|\xi^T (I \otimes (\mathcal{P}^* A))\xi\right| = \frac{1}{2}\left| - \xi^T (I \otimes \mathcal{Q}^*)\xi + \xi^T (I \otimes \mathcal{P}^* B B^T \mathcal{P}^*)\xi\right| \tag{3.86}$$

Similar procedure can be followed to express
$\left|\xi^T (I \otimes (\mathcal{P}^* A_{\mathcal{K}^*}))\xi\right|$ as

$$\left|\xi^T (I \otimes (\mathcal{P}^* A_{\mathcal{K}^*}))\xi\right| = \frac{1}{2}\left| - \xi^T (\mathcal{Q}^* \otimes I)\xi - \xi^T (\mathcal{P}^* B B^T \mathcal{P}^* \otimes I)\xi\right| \tag{3.87}$$

Use $\mathcal{Q}^* > 0$, $\mathcal{P}^* B B^T \mathcal{P}^* \geq 0$ and properties of eigenvalues of Kronecker Product to conclude that $(I \otimes \mathcal{Q}^*) > 0$ and $(I \otimes \mathcal{P}^* B B^T \mathcal{P}^*) \geq 0$. Therefore,

$$\begin{aligned} \left| - \xi^T (I \otimes \mathcal{Q}^*)\xi - \xi^T (I \otimes \mathcal{P}^* B B^T \mathcal{P}^*)\xi\right| > \\ \left| - \xi^T (I \otimes \mathcal{Q}^*)\xi + \xi^T (I \otimes \mathcal{P}^* B B^T \mathcal{P}^*)\xi\right| \end{aligned} \tag{3.88}$$

We can now use (3.88) to compare $\left|\xi^T(I \otimes (\mathcal{P}^* A_{\mathcal{K}^*}))\xi\right|$ in (3.87) with $\left|\xi^T(I \otimes (\mathcal{P}^* A))\xi\right|$ in (3.86)

$$\left|\xi^T(I \otimes (\mathcal{P}^* A_{\mathcal{K}^*}))\xi\right| > \left|\xi^T(I \otimes (\mathcal{P}^* A))\xi\right| \tag{3.89}$$

The inequality in (3.89) allows us to bound the ratio in (3.82). Concretely,

$$\left|\lambda\right| = \frac{\left|\xi^T\big(I \otimes (P^* A)\big)\xi\right|}{\left|\xi^T\big(I \otimes (P^* A_{K^*})\big)\xi\right|} < 1 \tag{3.90}$$

Finally, since $|\lambda| < 1$, we can use (3.80) to write the inequality

$$\frac{|\xi^T(I \otimes A)\xi|}{|\xi^T(I \otimes A_{\mathcal{K}^*})\xi|} < 1 \tag{3.91}$$

and from this inequality the result in (3.42) directly follows. The inequality (3.43) can be derived in a similar fashion. □

3.7.2 Proof of Lemma 6

Proof. First, note that

$$A_{K_i} = A - BB^T P_i = A - BB^T(\mathcal{P}^* + E_i) \tag{3.92}$$

Then, Statement A) can be proved by noticing that for $E_i = 0$, we have $A_{K_i} = A_{\mathcal{K}^*}$. Therefore, the operator (3.40) for $E_i = 0$ is given by (3.44). This proves Statement A).

Next, we prove Statement B). Expand $\Psi(\overrightarrow{E}_i)$ and use (3.92) and (3.36) to write

$$\begin{aligned}\Psi(\overrightarrow{E}_i) &= \big(A_{K_i} \oplus A_{K_i}\big)^{-T}(A \oplus A\big)^T \\ &= \big(A_{K^*} \oplus A_{K^*} - ((BB^TE_i) \oplus (BB^TE_i))\big)^{-T} \cdot \\ &\qquad \cdot (A \oplus A\big)^T \\ &= \Big(I - \big((BB^TE_i) \oplus (BB^TE_i)\big)(A_{K^*} \oplus A_{K^*})^{-1}\Big)^{-T} \cdot \\ &\qquad \cdot (A_{K^*} \oplus A_{K^*})^{-T}(A \oplus A\big)^T \end{aligned} \tag{3.93}$$

Note that the last factor in (3.93) is exactly the operator $\Psi(0)$ evaluated at the solution $\overrightarrow{E}_i = 0$ as given by (3.44). Inserting $\Psi(0)$, directly proves Statement B).

Next, we prove Statement C). Use (3.44) to write

$$\begin{aligned}\Psi(0)\xi &= \mu\xi \\ (A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^{-T}(A \oplus A\big)^T\xi &= \mu\xi\end{aligned} \tag{3.94}$$

Left multiply (3.94) by $\xi^T(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^T$

$$\begin{aligned}\xi^T(A \oplus A\big)^T\xi &= \mu\xi^T(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})^T\xi \\ \xi^T(A \oplus A\big)\xi &= \mu\xi^T(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*})\xi\end{aligned} \tag{3.95}$$

Finally express $\big|\mu\big|$ as

$$\big|\mu\big| = \frac{\big|\xi^T(A \oplus A\big)\xi\big|}{\big|\xi^T\big(A_{\mathcal{K}^*} \oplus A_{\mathcal{K}^*}\big)\xi\big|} \tag{3.96}$$

In order to determine the bound on (3.96), we expand the numerator and denominator using the definition of the Kronecker Sum

$$\begin{aligned}\big|\mu\big| &= \frac{\big|\xi^T(I \otimes A\big)\xi + \xi^T(A \otimes I\big)\xi\big|}{\big|\xi^T(I \otimes A_{\mathcal{K}^*}\big)\xi + \xi^T(A_{\mathcal{K}^*} \otimes I\big)\xi\big|} \\ &= \frac{\big|\xi^T(I \otimes A\big)\xi + \xi^T(A \otimes I\big)\xi\big|}{\big|\xi^T(I \otimes A_{\mathcal{K}^*}\big)\xi + \xi^T(A_{\mathcal{K}^*} \otimes I\big)\xi\big|}\end{aligned} \tag{3.97}$$

and study the terms in denominator and numerator. First, note that $I \otimes A_{\mathcal{K}^*}$ and $A_{\mathcal{K}^*} \otimes I$ are negative definite. This allows us to write the equality $\left|\xi^T(I \otimes A_{\mathcal{K}^*})\xi + \xi^T(A_{\mathcal{K}^*} \otimes I)\xi\right| = \left|\xi^T(I \otimes A_{\mathcal{K}^*})\xi\right| + \left|\xi^T(A_{\mathcal{K}^*} \otimes I)\xi\right|$ and further develop the expression (3.97) as follows

$$|\mu| = \frac{\left|\xi^T(I \otimes A)\xi + \xi^T(A \otimes I)\xi\right|}{\left|\xi^T(I \otimes A_{\mathcal{K}^*})\xi\right| + \left|\xi^T(A_{\mathcal{K}^*} \otimes I)\xi\right|} \tag{3.98}$$

Use the triangle inequality to find the upper bound on $|\mu|$

$$\begin{aligned} |\mu| &= \frac{\left|\xi^T(I \otimes A)\xi + \xi^T(A \otimes I)\xi\right|}{\left|\xi^T(I \otimes A_{\mathcal{K}^*})\xi\right| + \left|\xi^T(A_{\mathcal{K}^*} \otimes I)\xi\right|} \\ &\leq \frac{\left|\xi^T(I \otimes A)\xi\right| + \left|\xi^T(A \otimes I)\xi\right|}{\left|\xi^T(I \otimes A_{\mathcal{K}^*})\xi\right| + \left|\xi^T(A_{\mathcal{K}^*} \otimes I)\xi\right|} \end{aligned} \tag{3.99}$$

From (3.43) and (3.42) in Lemma 5 we see that $|\mu| < 1$. This proves Statement C). □

$$\mathcal{Q} = \begin{bmatrix} 1.088 & 0.827 & 0.996 & 0.449 & 1.222 & 0.848 & 0.693 & 0.863 & 0.805 & 0.539 \\ 0.827 & 1.029 & 0.901 & 0.569 & 1.105 & 0.792 & 0.555 & 0.945 & 0.849 & 0.674 \\ 0.996 & 0.901 & 1.015 & 0.468 & 1.206 & 0.818 & 0.654 & 1.001 & 0.808 & 0.666 \\ 0.449 & 0.569 & 0.468 & 0.656 & 0.594 & 0.575 & 0.413 & 0.633 & 0.598 & 0.42 \\ 1.222 & 1.105 & 1.206 & 0.594 & 1.546 & 0.905 & 0.76 & 1.147 & 0.949 & 0.709 \\ 0.848 & 0.792 & 0.818 & 0.575 & 0.905 & 1.147 & 0.793 & 0.853 & 0.964 & 0.715 \\ 0.693 & 0.555 & 0.654 & 0.413 & 0.76 & 0.793 & 0.983 & 0.984 & 0.9 & 0.706 \\ 0.863 & 0.945 & 1.001 & 0.633 & 1.147 & 0.853 & 0.984 & 1.609 & 1.154 & 1.011 \\ 0.805 & 0.849 & 0.808 & 0.598 & 0.949 & 0.964 & 0.9 & 1.154 & 1.117 & 0.785 \\ 0.539 & 0.674 & 0.666 & 0.42 & 0.709 & 0.715 & 0.706 & 1.011 & 0.785 & 0.789 \end{bmatrix} \tag{3.100}$$

$$Q_\infty = \begin{bmatrix} 1.4 & 0.86 & 1.42 & 1.01 & -0.44 & -0.36 & -0.080.97 & 0.83 & -0.27 \\ 0.86 & -1.77 & -1.01 & -2.09 & 4. & -0.18 & 1.871.26 & 1.91 & 1.67 \\ 1.42 & -1.01 & 2.25 & -0.58 & 3.23 & -1.18 & 1.423.11 & 3.25 & 2.2 \\ 1.01 & -2.09 & -0.58 & -1.04 & 4.71 & -2.11 & 2.22.39 & 3.9 & 2.03 \\ -0.44 & 4. & 3.23 & 4.71 & -6.29 & 4.95 & -3.02-1.69 & -5.3 & -2.47 \\ -0.36 & -0.18 & -1.18 & -2.11 & 4.95 & 2.59 & 2.171.55 & 2.44 & 3.42 \\ -0.08 & 1.87 & 1.42 & 2.2 & -3.02 & 2.17 & -0.09-0.61 & -2.01 & -1.3 \\ 0.97 & 1.26 & 3.11 & 2.39 & -1.69 & 1.55 & -0.612.38 & -0.3 & -0.52 \\ 0.83 & 1.91 & 3.25 & 3.9 & -5.3 & 2.44 & -2.01-0.3 & -2.09 & -2.47 \\ -0.27 & 1.67 & 2.2 & 2.03 & -2.47 & 3.42 & -1.3-0.52 & -2.47 & 0.14 \end{bmatrix} \tag{3.101}$$

$$A = \begin{bmatrix} 0.141 & -0.495 & 0.199 & 0.35 & 0.444 & 0.0140.198 & 0.161 & -0.155 & -0.122 \\ -0.103 & -0.041 & 0.11 & 0.285 & 0.099 & -0.424-0.196 & 0.094 & 0.357 & 0.171 \\ -0.248 & -0.08 & 0.156 & -0.226 & -0.307 & -0.364-0.276 & 0.41 & 0.165 & 0.031 \\ -0.281 & 0.386 & 0.379 & -0.281 & -0.473 & 0.4110.2 & 0.324 & 0.167 & -0.172 \\ -0.332 & 0.207 & 0.472 & -0.003 & 0.229 & 0.3330.118 & 0.311 & 0.478 & 0.448 \\ 0.003 & 0.206 & -0.056 & 0.19 & -0.007 & -0.1080.054 & -0.361 & -0.217 & 0.02 \\ 0.088 & -0.073 & 0.013 & -0.478 & 0.313 & 0.11-0.112 & 0.313 & 0.269 & 0.076 \\ 0.043 & -0.136 & -0.098 & 0.487 & -0.065 & 0.458-0.239 & -0.313 & 0.13 & 0.35 \\ -0.053 & -0.353 & -0.1 & -0.44 & 0.203 & -0.4410.231 & 0.196 & -0.048 & 0.274 \\ -0.386 & 0.378 & -0.387 & -0.291 & -0.376 & -0.0670.015 & 0.316 & -0.004 & -0.248 \end{bmatrix} \tag{3.102}$$

$$B = \begin{bmatrix} 0.345 & -0.123 & 0.46 \\ -0.39 & 0.41 & 0.259 \\ -0.098 & -0.25 & -0.28 \\ 0.213 & -0.27 & 0.023 \\ -0.043 & 0.239 & -0.263 \\ 0.233 & -0.078 & 0.119 \\ 0.371 & -0.466 & 0.444 \\ -0.41 & -0.37 & -0.178 \\ 0.101 & -0.26 & -0.371 \\ -0.417 & -0.499 & 0.192 \end{bmatrix} \tag{3.103}$$

Chapter 4: MODEL BASED REINFORCMENT LEARNING FOR SAFE TRAJECTORY OPTIMIZATION

4.1 Motivation for Model based Reinforcement Learning

In this subsection, we motivate a different family of reinforcement learning algorithms known as model based reinforcement learning. There are 2 main issues in data driven reinforcement learning algorithms that model based reinforcement learning addresses.

First well known issue is that the convergence of all data driven reinforcement learning algorithms rely on the data contained in the state trajectories. It is hard to guarantee that the data contained in the state trajectories is sufficient for learning the value function and optimal policy. This heavy reliance on data is a non-trivial issue that requires the attention of algorithm designer. Additional efforts are needed to guarantee that the data driven algorithms are supplied with proper state trajectories.

The adaptive control community is familiar with this problem for some time. It is known that the convergence of adaptive controllers requires that the data in state trajectories satisfies certain statistical properties. These requirements are formalized through the concept of Persistence of Excitation. Specifically, in order for adaptive controller to converge to the desired solution, the system has to be persistently excited. It is the duty of algorithm designer to ensure that this condition is met. Again, this is a non-trivial problem.

In reinforcement learning community, this issue is generalized to a problem of exploration, stating that the data driven algorithms require the state trajectories to visit the entire admissible part of the state space. The part of state space that has not been explored remains "unexplained by the data". As a result, one cannot guarantee the global optimality of the resulting value function and policy.

Clearly, we are in need of a robust approach to learn optimal policies from the data. An idea that we will study in this chapter is to use models of system dynamics to compactly represent

state trajectories. Instead of learning the value function and optimal policy solely from data, one could learn robust models of system dynamics. This model is then used for learning the value function and optimal policy. The advantage of this approach is that if the model approximator function is rich enough, then all data trajectories can be implicitly represented in that model. That makes the algorithm robust to insufficiently excited state trajectories. In order to make this work, we need a function approximator that is general enough to represent all collected state trajectories simultaneously. In other words, if the function approximator is not a general representation of the data, then the model could be biased towards heavily represented state trajectories, ignoring important corner cases that are underrepresented in the data.

Second problem with data driven reinforcement learning is that it does not scale efficiently with the amount of data available for learning. As a consequence, it might be impossible to use all the data in real time. The experience replay is one solution for this problem. In experience replay, the algorithm randomly selects a batch of data out of all historic data in each step. In this chapter, we take a different approach. Again, we recognize the fact that the dynamics model is a compact way to implicitly store state trajectories.

In conclusion, it seems that the pathologies present in data driven reinforcement learning can be addressed by introducing appropriate model of the system dynamics. The model we need has to be a powerful approximator that has the capacity to take in large amounts of data. There are many different function approximators to choose from, but in this chapter we will focus on Gaussian Process.

4.2 Introduction

Reinforcement learning (RL) is a learning framework that addresses sequential decision-making problems, wherein an 'agent' or a decision maker learns a policy to optimize a long-term reward by interacting with the (unknown) environment. At each step, the RL agent obtains evaluative feedback (called reward or cost) about the performance of its action, allowing it to improve the performance of subsequent actions [99, 113]. Although RL has witnessed huge successes in recent

times [93, 94], there are several unsolved challenges, which restrict the use of these algorithms for industrial systems. In most practical applications, control policies must be designed to satisfy operational constraints, and a satisfactory policy should be learnt in a data-efficient fashion [109].

Model-based reinforcement learning (MBRL) methods [25] learn a model from exploration data of the system, and then exploit the model to synthesize a trajectory-centric controller for the system [57]. These techniques are, in general, harder to train, but could achieve good data efficiency [56]. Learning reliable models is very challenging for non-linear systems and thus, the subsequent trajectory optimization could fail when using inaccurate models. However, modern machine learning methods such as Gaussian processes (GP), stochastic neural networks (SNN), etc. can generate uncertainty estimates associated with predictions [86, 88]. These uncertainty estimates could be used to estimate the confidence set of system states at any step along a given controlled trajectory for the system. The idea presented in this chapter considers the stabilization of the trajectory using a local feedback policy that acts as an attractor for the system in the known region of uncertainty along the trajectory [102].

We present a method for simultaneous trajectory optimization and local policy optimization, where the policy optimization is performed in a neighborhood (local sets) of the system states along the trajectory. These local sets could be obtained by a stochastic function approximator (e.g., GP, SNN, etc.) that used to learn the forward model of the dynamical system. The local policy is obtained by considering the worst-case deviation of the system from the nominal trajectory at every step along the trajectory. Performing simultaneous trajectory and policy optimization could allow us to exploit the modeling uncertainty as it drives the optimization to regions of low uncertainty, where it might be easier to stabilize the trajectory. This allows us to constrain the trajectory optimization procedure to generate robust, high-performance controllers. The proposed method automatically incorporates state and input constraints on the dynamical system.

Contributions. The main contributions of the current chapter are:

1. We present a novel formulation of simultaneous trajectory optimization and time-invariant local policy synthesis for stabilization.

2. We present analysis of the proposed technique that allows us to analytically derive the gradient of the robustness constraint for the optimization problem.

It is noted that this chapter only presents the controller synthesis part for MBRL – a more detailed analysis of the interplay between model uncertainties and controller synthesis is deferred to another publication.

4.3 Related Work

MBRL has raised a lot of interest recently in robotics applications, because model learning algorithms are largely task independent and data-efficient [25, 56, 116]. However, MBRL techniques are generally considered to be hard to train and likely to result in poor performance of the resulting policies/controllers, because the inaccuracies in the learned model could guide the policy optimization process to low-confidence regions of the state space. For non-linear control, the use of trajectory optimization techniques such as differential dynamic programming [43] or its first-order approximation, the iterative Linear Quadratic Regulator (iLQR) [101] is very popular, as it allows the use of gradient-based optimization, and thus could be used for high-dimensional systems. As the iLQR algorithm solves the local LQR problem at every point along the trajectory, it also computes a sequence of feedback gain matrices to use along the trajectory. However, the LQR problem is not solved for ensuring robustness, and furthermore the controller ends up being time-varying, which makes its use somewhat inconvenient for robotic systems. Thus, we believe that the controllers we propose might have better stabilization properties, while also being time-invariant.

Most model-based methods use a function approximator to first learn an approximate model of the system dynamics, and then use stochastic control techniques to synthesize a policy. Some of the seminal work in this direction could be found in [25, 56]. The method proposed in [56] has been shown to be very effective in learning trajectory-based local policies by sampling several initial conditions (states) and then fitting a neural network which can imitate the trajectories by supervised learning. This can be done by using ADMM [12] to jointly optimize trajectories and learn the neural network policies. This approach has achieved impressive performance on several

robotic tasks [56]. The method has been shown to scale well for systems with higher dimensions. Several different variants of the proposed method were introduced later [17, 71, 74]. However, no theoretical analysis could be provided for the performance of the learned policies.

Another set of seminal work related to the proposed work is on the use of sum-of-square (SOS) programming methods for generating stabilizing controller for non-linear systems [102]. In these techniques, a stabilizing controller, expressed as a polynomial function of states, for a non-linear system is generated along a trajectory by solving an optimization problem to maximize its region of attraction [67].

Another set of relevant work could be found in [14, 16] where the idea is to allow constraint satisfaction for the partially-known system by appropriately bounding model errors. Furthermore, authors in [15] present a way to perform approximate dynamic programming using the ideas of invariant sets. However, these methods find the global controller for the system which could be inefficient for high-dimensional systems. lastly, some model-free trajectory-centric approaches could be found in [103, 117].

4.4 Problem Formulation

In this section, we describe the problem studied in the rest of the chapter. To perform trajectory-centric control, we propose a novel formulation for simultaneous design of open-loop trajectory and a time-invariant, locally stabilizing controller that is robust to bounded model uncertainties and/or system measurement noise. As we will present in this section, the proposed formulation is different from that considered in the literature in the sense it allows us to exploit sets of possible deviation of a system to design stabilizing controller.

4.4.1 Trajectory Optimization as Non-linear Program

Consider the discrete-time dynamical system

$$x_{k+1} = f(x_k, u_k) \tag{4.1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$ are the differential states and controls, respectively. The function $f : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ governs the evolution of the differential states. Note that the discrete-time formulation (4.1) can be obtained from a continuous time system $\dot{x} = \hat{f}(x, u)$ by using the *explicit Euler* integration scheme $(x_{k+1} - x_k) = \Delta t \hat{f}(x_k, u_k)$ where Δt is the time-step for integration.

For clarity of exposition we have limited our focus to discrete-time dynamical systems of the form in (4.1) although the techniques we describe can be easily extended to implicit discretization schemes.

In typical applications the states and controls are restricted to lie in sets $:= \{x \in \mathbb{R}^{n_x} \,|\, \underline{x} \leq x \leq \overline{x}\} \subseteq \mathbb{R}^{n_x}$ and $:= \{u \in \mathbb{R}^{n_u} \,|\, \underline{u} \leq u \leq \overline{u}\} \subseteq \mathbb{R}^{n_u}$, *i.e.* $x_k \in, u_k \in$. We use $[K]$ to denote the index set $\{0, 1, \ldots, K\}$. Further, there may exist nonlinear inequality constraints of the form

$$g(x_k) \geq 0 \tag{4.2}$$

with $g : \mathbb{R}^{n_x} \to \mathbb{R}^m$. The inequalities in (4.2) are termed as *path constraints*. The trajectory optimization problem is to manipulate the controls u_k over a certain number of time steps $[T-1]$ so that the resulting trajectory $\{x_k\}_{k\in[T]}$ minimizes a cost function $c(x_k, u_k)$. Formally, we aim to solve the *trajectory optimization problem*

$$\begin{aligned} \min_{x_k, u_k} \; & \sum_{k\in[T]} c(x_k, u_k) \\ \text{s.t. Eq. (4.1)} - \text{(4.2) for } & k \in [T] \\ x_0 = \; & \tilde{x}_0 \\ x_k \in \; & \text{ for } k \in [T] \\ u_k \in \; & \text{ for } k \in [T-1] \end{aligned} \tag{TrajOpt}$$

where $\tilde{x}_0$ is the differential state at initial time $k = 0$. Before introducing the main problem of interest, we would like to introduce some notations.

In the following text, we use the following shorthand notation, $||v||_M^2 = v^T M v$. We denote the

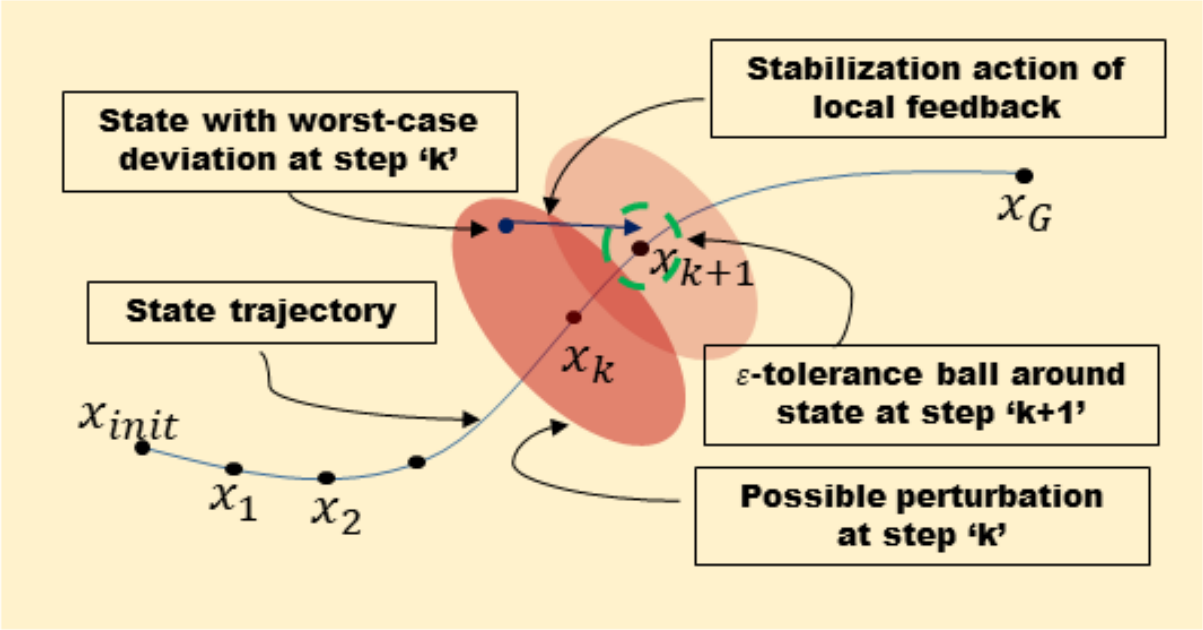

Figure 4.1: A schematic representation of the *robustness constraint* introduced in the chapter. nominal trajectory as $X \equiv x_0, x_1, x_2, x_3, \ldots, x_{T-1}, x_T$, $U \equiv u_0, u_1, u_2, u_3, ..., u_{T-1}$. The actual trajectory followed by the system is denoted as $\hat{X} \equiv \hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_{T-1}, \hat{x}_T$. We denote a local policy as π_W, where π is the policy and W denotes the parameters of the policy. The trajectory cost is also sometimes denoted as $J = \sum_{k \in [T]} c(x_k, u_k)$.

4.4.2 Trajectory Optimization with Local Stabilization

This subsection introduces the main problem of interest in this chapter. A schematic of the problem studied in the chapter is also shown in Figure 4.1. In the rest of this section, we will describe how we can simplify the trajectory optimization and local stabilization problem and turn it into an optimization problem that can be solved by standard non-linear optimization solvers.

Consider the case where the system dynamics, f is only partially known, and the known component of f is used to design the controller. Consider the deviation of the system at any step 'k' from the state trajectory X and denote it as $\delta x_k \equiv x_k - \hat{x}_k$. We introduce a local (time-invariant) policy π_W that regulates the local trajectory deviation δx_k and thus, the final controller is denoted as $\hat{u}_k = u_k + \pi_W(\delta x_k)$. The closed-loop dynamics for the system under this control is then given

by the following:

$$\hat{x}_{k+1} = f(\hat{x}_k, \hat{u}_k) = f(x_k + \delta x_k, u_k + \pi_W(\delta x_k)) \tag{4.3}$$

The main objective of the chapter is to find the time-invariant feedback policy π_W that can stabilize the open-loop trajectory X locally within $\mathbb{R}_k \subset \mathbb{R}^{n_x}$ where $\mathbb{R}_k$ defines the set of uncertainty for the deviation δx_k. The uncertainty region $\mathbb{R}_k$ can be approximated by fitting an ellipsoid to the uncertainty estimate using a diagonal positive definite matrix S_k such that $\mathbb{R}_k = \{\delta x_k : \delta x_k^T S_k \delta x_k \leq 1\}$. The general optimization problem that achieves that is proposed as:

$$\begin{aligned} J^* = &\min_{U,X,W} \ \max_{\delta x_k \in \mathbb{R}_k} [J(X + \delta X, U + \pi_W(\delta x_k)] \\ &x_{k+1} = \hat{f}(x_k, u_k) \end{aligned} \tag{4.4}$$

where $\hat{f}(\cdot, \cdot)$ denotes the known part of the model. Note that in the above equation, we have introduced additional optimization parameters corresponding to the policy π_W when compared to TrajOpt in the previous section. However, to solve the above, one needs to resort to sampling in order to estimate the expected cost. Instead we introduce a constraint that solves for the worst-case cost for the above problem.

Robustness Certificate. The robust trajectory optimization problem is to minimize the trajectory cost while at the same time satisfying a *robust constraint* at every step along the trajectory. This is also explained in Figure 4.1, where the purpose of the local stabilizing controller is to push the max-deviation state at every step along the trajectory to ϵ-tolerance balls around the trajectory.

Mathematically, we express the problem as following:

$$
\begin{aligned}
&\min_{x_k, u_k, W} \sum_{k \in [T]} c(x_k, u_k) \\
&\quad \text{s.t. Eq. (4.1)} - \text{(4.2) for } k \in [T] \\
&\quad\quad x_0 = \tilde{x}_0 \\
&\quad\quad x_k \in \text{ for } k \in [T] \\
&\quad\quad u_k \in \text{ for } k \in [T-1] \\
&\quad\quad \max_{\delta x_k \in \mathbb{R}_k} ||x_{k+1} - f(x_k + \delta x_k, u_k + \pi_W(\delta x_k))||_2 \leq \epsilon_k
\end{aligned}
\tag{RobustTrajOpt}
$$

The additional constraint introduced in RobustTrajOpt allows us to ensure stabilization of the trajectory by estimating parameters of the stabilizing policy π_W. It is easy to see that RobustTrajOpt solves the worst-case problem for the optimization considered in (4.4). However, RobustTrajOpt introduces another hyperparameter to the optimization problem, ϵ_k. In the rest of the chapter, we refer to the following constraint as the *robust constraint*:

$$
\max_{\delta x_k^T S_k \delta x_k \leq 1} ||x_{k+1} - f(x_k + \delta x_k, u_k + \pi_W(\delta x_k))||_2 \leq \epsilon_k \tag{4.5}
$$

Solution of the *robust constraint* for generic non-linear system is out of scope of this chapter. Instead, we linearize the trajectory deviation dynamics as shown in the following Lemma.

Lemma 7. *The trajectory deviation dynamics $\delta x_{k+1} = x_{k+1} - \hat{x}_{k+1}$ approximated locally around the optimal trajectory (X, U) are given by*

$$
\begin{aligned}
\delta x_{k+1} &= A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k) \\
A(x_k, u_k) &\equiv \nabla_{x_k} \hat{f}(x_k, u_k) \\
B(x_k, u_k) &\equiv \nabla_{u_k} \hat{f}(x_k, u_k)
\end{aligned}
\tag{4.6}
$$

Proof. Use Taylor's series expansion to obtain the desired expression. □

To ensure feasibility of the RobustTrajOpt problem and avoid tuning the hyperparameter ϵ_k, we make another relaxation by removing the *robust constraint* from the set of constraints and move it to the objective function. Thus, the simplified robust trajectory optimization problem that we solve in this chapter can be expressed as following (we skip the state constraints to save space).

$$\begin{aligned} &\min_{x_k,u_k,W} \Big(\sum_{k\in[T]} c(x_k,u_k) + \alpha \sum_{k\in[T]} d_{max,k}\Big) \\ &\text{s.t. Eq. (4.1)} - \text{(4.2) for } k \in [T] \end{aligned} \tag{RelaxedRobustTrajOpt}$$

where the term $d_{max,k}$ is defined as following after linearization.

$$\begin{aligned} &d_{max,k} \equiv \\ &\max_{\delta x_k^T S_k \delta x_k \leq 1} ||A(x_k,u_k)\cdot \delta x_k + B(x_k,u_k)\cdot \pi_W(\delta x_k)||_P^2 \end{aligned} \tag{4.7}$$

Note that the matrix P allows to weigh states differently. In the next section, we present the solution approach to compute the gradient for the RelaxedRobustTrajOpt which is then used to solve the optimization problem. Note that this results in simultaneous solution to open-loop and the stabilizing policy π_W.

4.5 Solution Approach

This section introduces the main contribution of the chapter, which is a local feedback design that regulates the deviation of an executed trajectory from the optimal trajectory generated by the optimization procedure.

To solve the optimization problem presented in the last section, we first need to obtain the gradient information of the robustness heuristic that we introduced. However, calculating the gradient of the robust constraint is not straightforward, because the max function is non-differentiable. The gradient of the robustness constraint is computed by the application of Danskins Theorem [9], which is stated next.

Danskin's Theorem: Let $K \subseteq \mathbb{R}^m$ be a nonempty, closed set and let $\Omega \subseteq \mathbb{R}^n$ be a nonempty,

open set. Assume that the function $f : \Omega \times K \to \mathbb{R}$ is continuous on $\Omega \times K$ and that $\nabla_x f(x, y)$ exists and is continuous on $\Omega \times K$. Define the function $g : \Omega \to \mathbb{R} \cup \{\infty\}$ by

$$g(x) \equiv \sup_{y \in K} f(x, y), x \in \Omega$$

and

$$M(x) \equiv \{y \in K \mid g(x) = f(x, y)\}.$$

Let $x \in \Omega$ be a given vector. Suppose that a neighborhood $\mathcal{N}(x) \subseteq \Omega$ of x exists such that $M(x')$ is nonempty for all $x' \in \mathcal{N}(x)$ and the set $\cup_{x' \in \mathcal{N}(x)} M(x')$ is bounded. The following two statements (a) and (b) are valid.

(a) The function g is directionally differentiable at x and

$$g'(x; d) = \sup_{y \in M(x)} \nabla_x f(x, y)^T d.$$

(b) If $M(x)$ reduces to a singleton, say $M(x) = \{y(x)\}$, then g is Gâeaux differentiable at x and

$$\nabla g(x) = \nabla_x f(x, y(x)).$$

Proof See [29], Theorem 10.2.1.

Danskin's theorem allows us to find the gradient of the robustness constraint by first computing the argument of the maximum function and then evaluating the gradient of the maximum function at the point. Thus, in order to find the gradient of the robust constraint (4.5), it is necessary to interpret it as an optimization problem in δx_k, which is presented next. In Section 4.4.2, we presented a general formulation for the stabilization controller π_W, where W are the parameters that are obtained during optimization. However, solution of the general problem is beyond the scope of the current chapter. Rest of this section considers a linear π_W for analysis.

Lemma 8. *Assume the linear feedback $\pi_W(\delta x_k) = W\delta x_k$. Then, the constraint (4.7) is quadratic in δx_k,*

$$\begin{aligned} &\max_{\delta x_k} ||M_k \delta x_k||_P^2 = \max_{\delta x_k} \delta x_k^T M_k^T \cdot P \cdot M_k \delta x_k \\ &s.t. \quad \delta x_k^T S_k \delta x_k \leq 1 \end{aligned} \tag{4.8}$$

where M_k is shorthand notation for

$$M_k(x_k, u_k, W) \equiv A(x_k, u_k) + B(x_k, u_k) \cdot W \tag{4.9}$$

Proof. Write d_{max} from (4.7) as the optimization problem

$$\begin{aligned} &d_{max} = \\ &\max_{\delta x_k} ||A(x_k, u_k) \cdot \delta x_k + B(x_k, u_k) \cdot \pi_W(\delta x_k)||_P^2 \\ &s.t. \quad \delta x_k^T S_k \delta x_k \leq 1 \end{aligned} \tag{4.10}$$

Introduce the linear controller and use the shorthand notation for M_k to write (4.8). □

The next lemma is one of the main results in the chapter. It connects the robust trajectory tracking formulation RelaxedRobustTrajOpt with the optimization problem that is well known in the literature.

Lemma 9. *The worst-case measure of deviation d_{max} is*

$$\begin{aligned} &d_{max} = \\ &\lambda_{max}(S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}) = ||P^{\frac{1}{2}} M_k S_k^{-\frac{1}{2}}||_2^2 \end{aligned}$$

where $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue of a matrix and $||\cdot||_2$ denotes the spectral norm of a matrix. Moreover, the worst-case deviation δ_{max} is the corresponding maximum eigenvector

$$\begin{aligned} &\delta_{max} = \\ &\{\delta x_k : \left[S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}\right] \cdot \delta x_k = d_{max} \cdot \delta x_k\} \end{aligned} \tag{4.10}$$

Proof. Apply coordinate transformation $\delta\tilde{x}_k = S_k^{\frac{1}{2}}\delta x_k$ in (4.8) and write

$$\begin{aligned} &\max_{\delta\tilde{x}_k} \delta\tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta\tilde{x}_k \\ &s.t. \quad \delta\tilde{x}_k \delta\tilde{x}_k \leq 1 \end{aligned} \tag{4.11}$$

Since $S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}$ is positive semi-definite, the maximum lies on the boundary of the set defined by the inequality. Therefore, the problem is equivalent to

$$\begin{aligned} &\max_{\delta\tilde{x}_k} \delta\tilde{x}_k S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}} \delta\tilde{x}_k \\ &s.t. \quad \delta\tilde{x}_k \delta\tilde{x}_k = 1 \end{aligned} \tag{4.12}$$

The formulation (4.12) is a special case with a known analytic solution. Specifically, the maximizing deviation δ_{max} that solves (4.12) is the maximum eigenvector of $S_k^{-\frac{1}{2}} M_k^T \cdot P \cdot M_k S_k^{-\frac{1}{2}}$, and the value d_{max} at the optimum is the corresponding eigenvalue. □

This provides us with the maximum deviation along the trajectory at any step 'k', and now we can use Danskin's theorem to compute the gradient which is presented next.

Theorem 7. *Introduce the following notation, $\mathcal{M}(z) = S_k^{-\frac{1}{2}} M_k^T(z) \cdot P \cdot M_k(z) S_k^{-\frac{1}{2}}$. The gradient of the robust inequality constraint d_{max} with respect to an arbitrary vector z is*

$$\nabla_z d_{max} = \nabla_z \delta_{max}^T \mathcal{M}(z) \delta_{max}$$

Where δ_{max} is maximum trajectory deviation introduced in Lemma 3.

Proof. Start from the definition of gradient of robust constraint

$$\nabla_z d_{max} = \nabla_z \max_{\delta\tilde{x}_k} \delta\tilde{x}_k \mathcal{M}(z) \delta\tilde{x}_k$$

Use Danskin's Theorem and the result from Lemma 9 to write the gradient of robust constraint

with respect to an arbitrary z,

$$\nabla_z d_{max} = \nabla_z \delta_{max}^T \mathcal{M}(z) \delta_{max}$$

which completes the proof. □

The gradient computed from Theorem 7 is used in solution of the RelaxedRobustTrajOpt– however, this is solved only for a linear controller. The next section shows some results in simulation and on a real physical system.

4.6 Experimental Results

In this section, we present some results using the proposed algorithm for an under-actuated inverted pendulum, as well as on a experimental setup for a ball-and-beam system. We use a Python wrapper for the standard interior point solver IPOPT to solve the optimization problem discussed in previous sections. We perform experiments to evaluate the following questions:

1. Can an off-the-shelf optimization solver find feasible solutions to the joint optimization problem described in the chapter?

2. Can the feedback controller obtained by this optimization stabilize the open-loop trajectory in the presence of bounded uncertainties?

3. How good is the performance of the controller on a physical system with unknown system parameters ?

In the following sections, we try to answer these questions using simulations as well as experiments on real systems.

4.6.1 Simulation Results for Underactuated Pendulum

The objective of this subsection is twofold: first, to provide insight into the solution of the optimization problem; and second, to demonstrate the effectiveness of that solution in the stabilization

of the optimal trajectory.

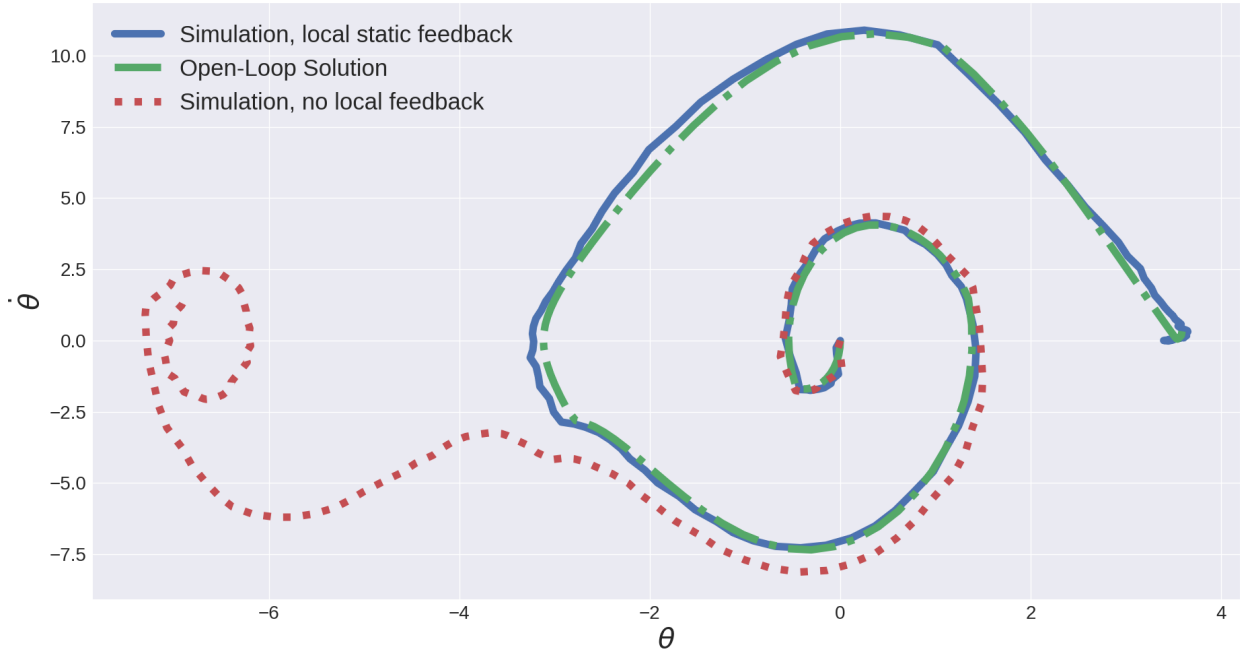

Figure 4.2: State-space representation of the optimal trajectory (green), stable closed-loop system using the obtained solution (blue), and unstable open-loop trajectory without the local feedback (red). Note that the feedback is time-invariant.

For clarity of presentation, we use an underactuated pendulum system, where trajectories can be visualized in state space. The dynamics of the pendulum is modeled as $I\ddot{\theta} + b\dot{\theta} + mgl \cdot sin(\theta) = u$. The continuous-time model is discretized as $(\theta_{k+1}, \dot{\theta}_{k+1}) = f((\theta_k, \dot{\theta}_k), u_k)$. The goal state is $x_g = [\pi, 0]$, and the initial state is $x_0 = [0, 0]$ and the control limit is $u \in [-1.7, 1.7]$. The cost is quadratic in both the states and input. The initial solution provided to the controller is trivial (all states and control are 0). The number of discretization points along the trajectory is $N = 120$, and the discretization time step is $\Delta t = 1/30$. The cost weight on robust slack variables is selected to be $\alpha = 10$. The uncertainty region is roughly estimated as $x_k^T \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 5.5 \end{bmatrix} x_k < 1$ along the trajectory. Detailed analysis on uncertainty estimation based on Gaussian processes is deferred to future work, due to space limits. The optimization procedure terminates in 50 iterations with the static solution $W = [-2.501840, -7.38725]$.

The controller generated by the optimization procedure is then tested in simulation, with noise

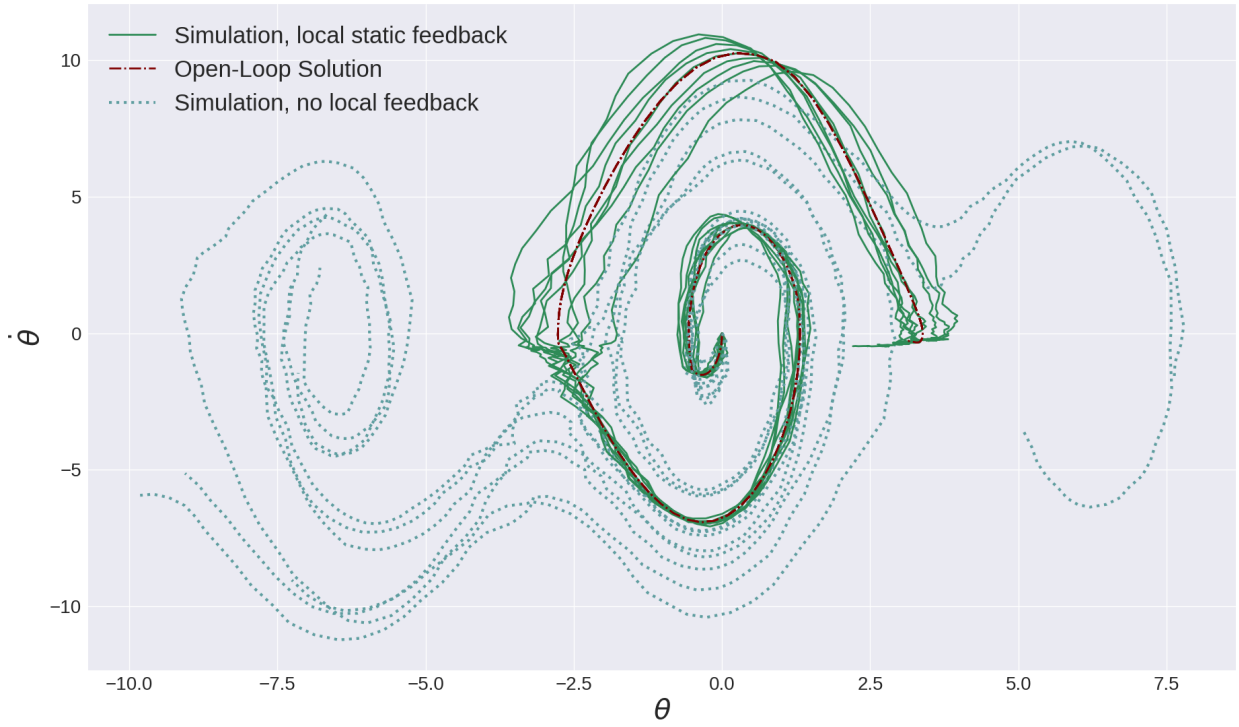

Figure 4.3: State-space representation of the open loop trajectory predicted by optimizer (dotted line), open loop trajectory with stabilizing component closed loop system using the obtained solution (full line), and unstable open-loop trajectory without the local feedback (dashed line). Note that the feedback is time-invariant.

added to each state of the pendulum model at each step of simulation as $x_{k+1} = f(x_k, u_k) + \omega$ with $\omega_\theta \sim \mathcal{U}(-0.2rad, 0.2rad])$ and $\omega_{\dot{\theta}} \sim \mathcal{U}(-0.05rad/s, 0.05rad/s])$.

We tested the controller over several settings and found that the underactuated setting was the most challenging to stabilize. In Figure 4.3, we show the state-space trajectory for the controlled (underactuated) system with additional noise as described above. As seen in the plot, the open-loop controller becomes unstable with the additional noise, while the proposed controller can still stabilize the whole trajectory. In Figure 4.4, we show the control inputs, the time-invariant feedback gains obtained by the optimization problem. We also the time-varying LQR gains obtained along the trajectory to show provide some insight between the two solutions. As the proposed optimization problem is finding the feedback gain for the worst-case deviation from the trajectory, the solutions are different than the LQR-case. Next, in Figure 4.6, we plot the error statistics for the controlled system (in the underactuated setting) over 2 different uncertainty balls using each

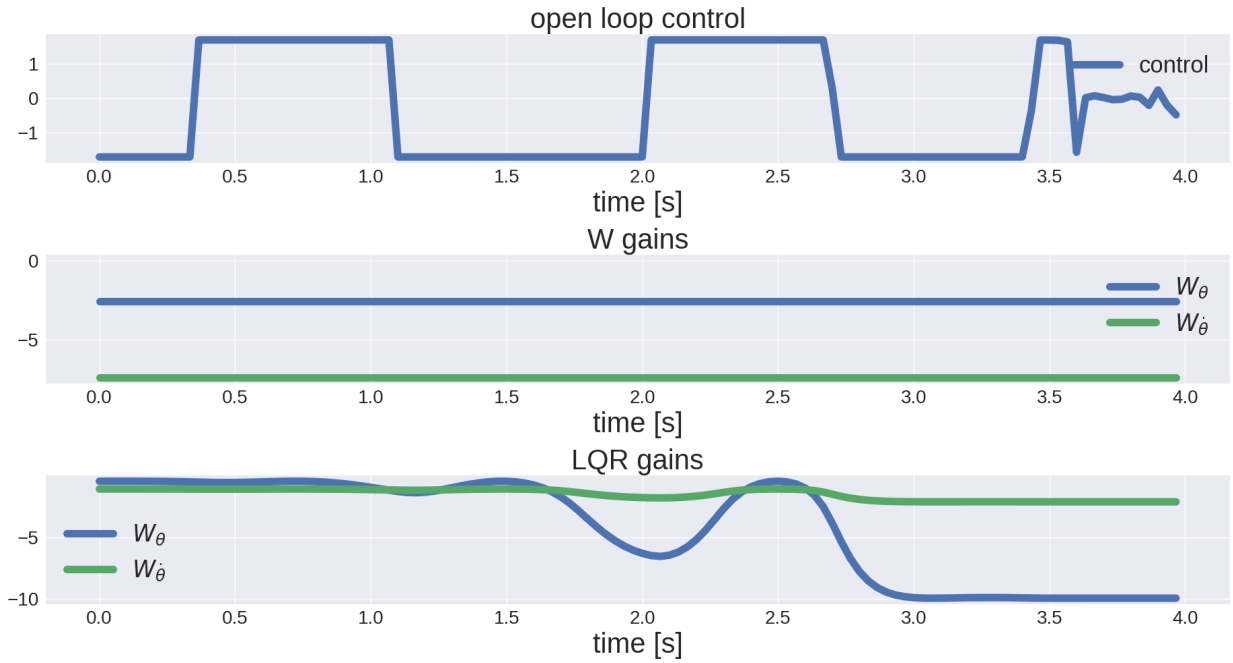

Figure 4.4: a) open-loop control b) static feedback matrix obtained from optimization c) LQR feedback

12 sample for each ball. We observe that the steady-state error goes to zero and the closed-loop system is stable along the entire trajectory. As we are using a linear approximation of the system dynamics, the uncertainty sets are still small, however the results are indicating that incorporating the full dynamics during stabilization could allow to generate much bigger basins of attraction for the stabilizing controller.

4.6.2 Results on Ball-and-Beam System

Next, we implemented the proposed method on a ball-and-beam system (shown in Figure 4.7) [44]. The ball-and-beam system is a low-dimensional non-linear system with the non-linearity due to the dry friction and delay in the servo motors attached to the table (see Figure 4.7). The ball-and-beam system can be modeled with 4 state variables $[x, \dot{x}, \theta, \dot{\theta}]$, where x is the position of the ball, $\dot{x}$ is the ball's velocity, θ is the beam angle in radians, and $\dot{\theta}$ is the angular velocity of the beam. The

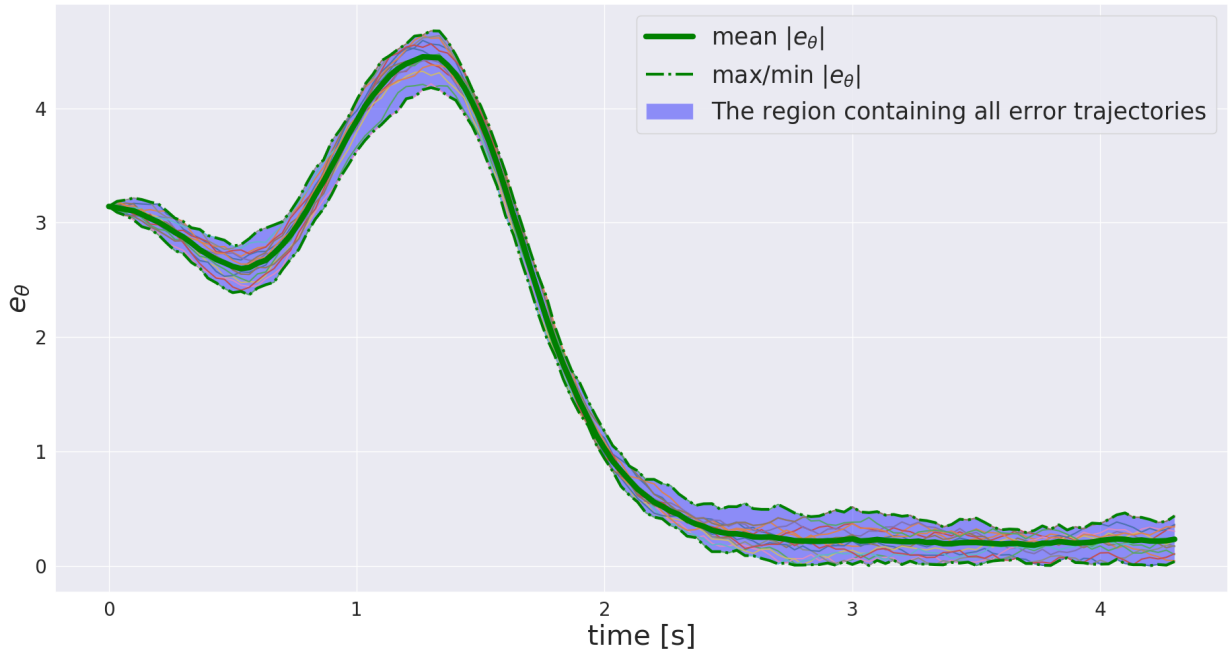

Figure 4.5: Error statistics for the controlled system using the proposed method on the under-actuated pendulum with noise amplitude

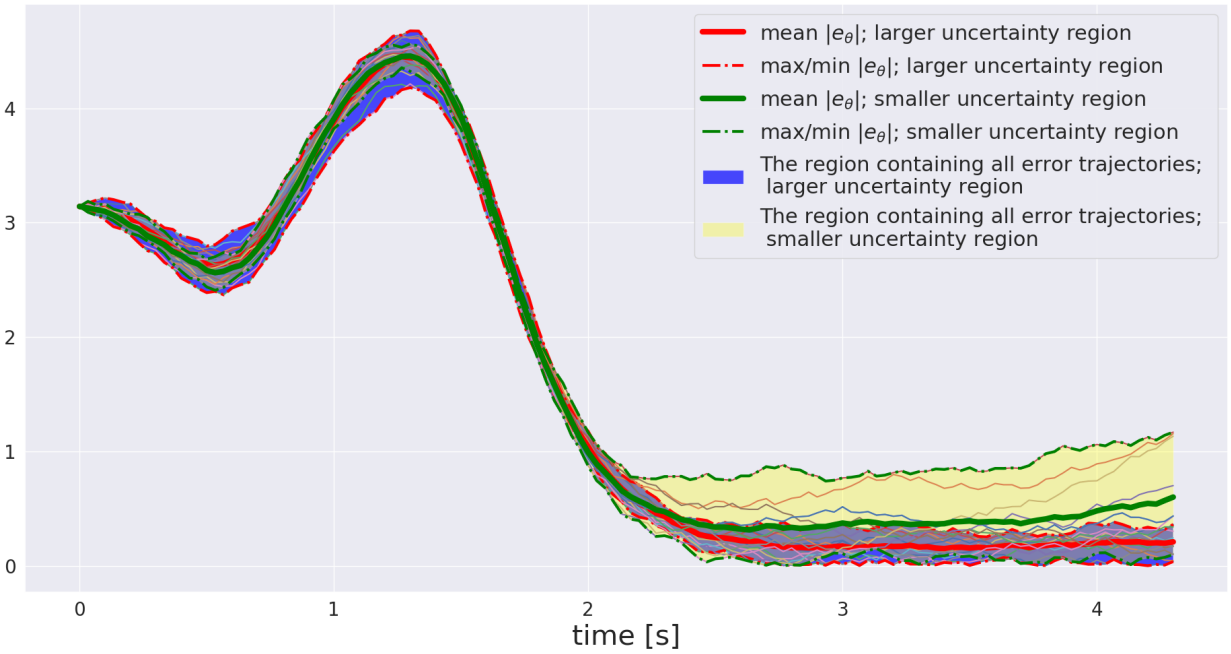

Figure 4.6: Error statistics for the controlled system using the proposed method on the under-actuated pendulum with noise amplitude for uncertainty regions of different sizes

acceleration of the ball, $\ddot{x}$, is given by

$$\ddot{x} = \frac{m_{ball} x \dot{\theta}^2 - b_1 \dot{x} - b_2 m_{ball} g \cos(\theta) - m_{ball} g \sin(\theta)}{\frac{I_{ball}}{r_{ball}^2} + m_{ball}},$$

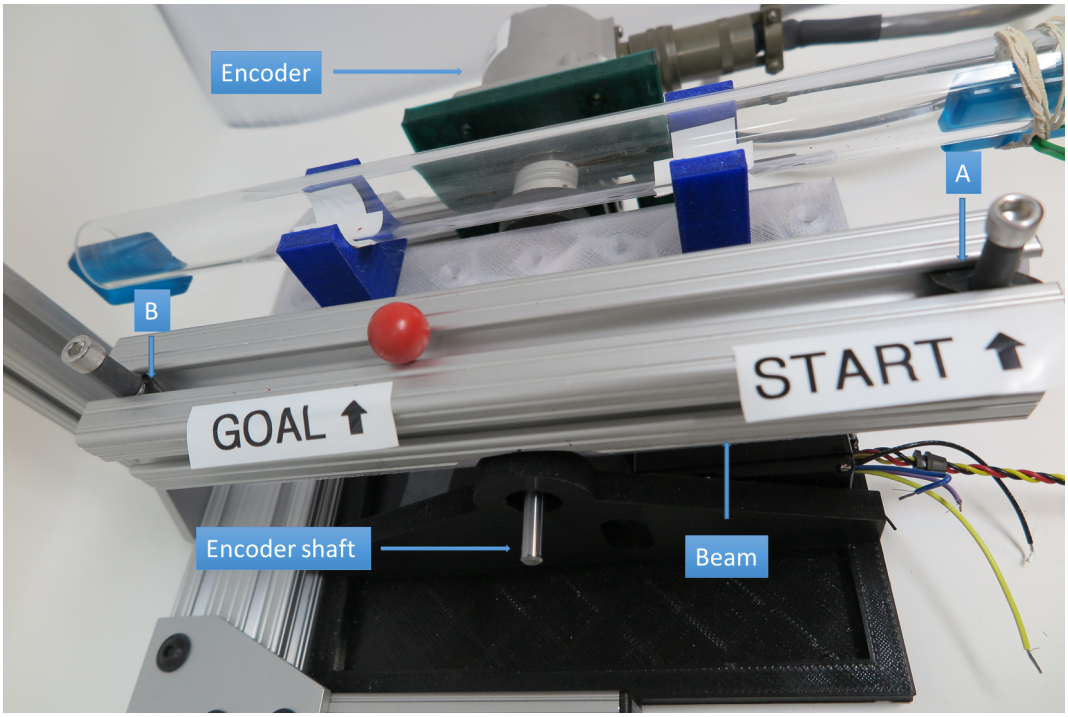

Figure 4.7: The ball-and-beam system used for the experiments. There is an RGB camera above that measures the location of the ball. The encoder (seen in the figure) measures the angular position of the beam.

where m_{ball} is the mass of the ball, I_{ball} is the moment of inertia of the ball, r_{ball} is the radius of the ball, b_1 is the coefficient of viscous friction of the ball on the beam, b_2 is the coefficient of static (dry) friction of the ball on the beam, and g is the acceleration due to gravity. The beam is actuated by a servo motor (position controlled) and an approximate model for its motion is estimated by fitting an auto-regressive model. We use this model for the analysis where the ball's rotational inertia is ignored and we approximately estimate the dry friction. The model is inaccurate, as can be seen from the performance of the open-loop controller in Figure 4.8. However, the proposed controller is still able to regulate the ball position at the desired goal showing the stabilizing behavior for the system (see the performance of the closed-loop controller in Figure 4.8). The plot shows the mean and the standard deviation of the error for 12 runs of the controller. It can be observed that the mean regulation error goes to zero for the closed-loop controller. We believe that the performance of the controller will improve as we improve the model accuracy. In future research, we would like to study the learning behavior for the proposed controller by learning the residual dynamics using

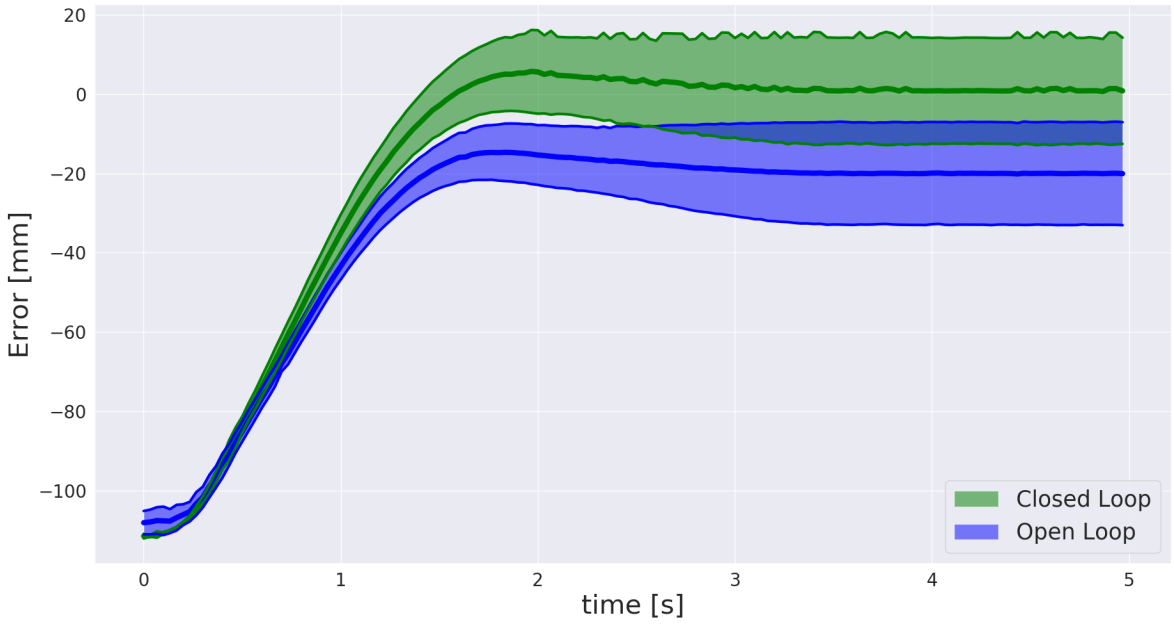

Figure 4.8: Comparison of the performance of the proposed controller on a ball-and-beam system with the open-loop solution. The plot shows the error in the position of the ball from the regulated position averaged over 12 runs.

GP [88].

4.7 Conclusion and Future Work

This chapter presents a method for simultaneously computing an optimal trajectory along with a local, time-invariant stabilizing controller for a dynamical system with known uncertainty bounds. The time-invariant controller was computed by adding a robustness constraint to the trajectory optimization problem. We prove that under certain simplifying assumptions, we can compute the gradient of the robustness constraint so that a gradient-based optimization solver could be used to find a solution for the optimization problem. We tested the proposed approach that shows that it is possible to solve the proposed problem simultaneously. We showed that even a linear parameterization of the stabilizing controller with a linear approximation of the error dynamics allows us to successfully control non-linear systems locally. We tested the proposed method in simulation as well as a physical system. Due to space limitations, we have to skip extra results regarding the behavior of the algorithm.

However, the current approach has two limitations– it makes linear approximation of dynamics for finding the worst-case deviation, and secondly, the linear parameterization of the stabilizing controller can be limiting for a lot of robotic systems. In future research, we will incorporate these two limitations using Lyapunov theory for non-linear control, for better generalization and more powerful results of the proposed approach. We also hope to extend the current formulation to more complex settings for feedback control of manipulation problems [35].

Chapter 5: OPTIMAL DYNAMIC CONTROL ALLOCATION OF INPUT REDUNDANT SYSTEMS

5.1 Introduction

A system is said to be actuator redundant when the number of actuators is greater than the number of high-level control inputs. Consequently, the actuators that gen-erate a certain control input are not unique for actuator redundant systems. The Control Allocation (CA) problem is to select actuators that generate a given control input and satisfy additional criteria, such as constraint satisfac-tion and energy efficiency of actuators.

Comprehensive control allocation surveys are given in [49], [80], [10]. Early attempts to solve the problem were inspired by applications in the control design of robotic manipulators [19], and aerial vehicles [10]. The main motivation for in-troducing control allocation in these applications is to maintain stability in case of actuator failures or/and to re-duce the power requirements.

The common approach to solve the control allocation problem is static optimization. In unconstrained case, the solution can be expressed in closed form, by using the generalized inverse, [49], [80]. If actuator ine-quality constraints are also considered, the problem be-comes a standard quadratic program [11], [82], [41]. In [36], it is shown that the optimality of the entire system is preserved if the con-trol allocation problem is solved separately. An important drawback of the static optimization approach to control allocation is that it disregards the dynamical nature of the problem in the objective.

One interpretation of dynamic allocation considers the problem of control allocation where the dynamics is present naturally in the actuators [38], [106] and [105]. On the other hand, dynamic allocation was de-fined differently in [122] as a novel tech-nique for exploring the subspace of redundant actuators by introducing virtual dynamics in the actuator space. This idea is further developed in [23] and [31]. Model predictive control approach was proposed in [120], [73] that achieves dynamic optimality and adaptive control allocation solution was proposed in [107] where

stability is guaranteed by Lyapunov analysis. However, both methods rely on knowledge of system model and the trajectory.

It is in our interest to apply dynamic optimization pro-cedures to solve the control allocation problem. Rein-forcement learning is a technique for determining solu-tions to dynamic optimization problems by measuring in-put-output data online and without knowing the system dynamics. In the last decade, this approach has attracted an increasing interest in many areas of research due to its ability to determine the optimal policy of a system in re-al-time [69], [64], [45].

The main contribution of this chapter is the novel formulation of control allocation as a dynamic optimization problem. This formulation allows us to: (1) unify and ex-tend the existing static control allocation solutions from the literature under a general solution to dynamic optimization based on robust control design. (2) Apart from dynamic optimality, our solution guarantees the robust-ness to unknown outer loop system dynamics. (3) Develop control allocation with guaranteed actuator constraints, and (4) learn the optimal control allocation online using data-driven model-free reinforcement learning method.

5.2 Mathematical Preliminaries

In this section, the prerequisites required to define our control allocation problem are introduced.

Consider the continuous time dynamic system

$$\dot{x}(t) = f(x(t)) + G(x(t))v \tag{5.1}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $v(t) \in \mathbb{R}^m$ is the control input, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth nonlinear function, and matrix $G \in \mathbb{R}^{n\times m}$ is a map from control space to state space. We will refer to (1) as the outer loop in this chapter. It is assumed that the stabilizing control input v is given by some prior design procedure. Input matrix G is assumed to have full column rank so that the design of v is unique. System (1), for instance, could represent an aircraft dynamics with control input v designed as a pitch rate controller, normal acceleration control augmentation system, etc. [96].

5.2.1 Actuator Redundancy

In this subsection, the basic concepts required to formulate the control allocation problem are defined.

Many controllers found in the literature are usually not directly implementable in practice. For instance, aircraft flight control inputs are generally composed of thrust, and torques in roll, pitch and yaw. To apply these aerodynamic control inputs to the system, it is necessary to find their functional dependence on the actual system actuators. If the number of actuators k is greater than the number of control inputs m, then the actuators are said to be redundant. The following definition formalizes this concept.

Definition 10 (Actuator Redundancy)**.** *Consider the allocation expression*

$$v(t) = Bu(t) \tag{5.2}$$

where $u(t) \in \mathbb{R}^k$ represents the available actuators and $B \in \mathbb{R}^{m\times k}$ is a matrix that can be interpreted as a map from the space of actuators u to the control input v. If $k - m > 0$ then the null space of matrix B is a non-empty set and the system is actuator redundant. The dimension of the nullspace of B is the redundancy $k - m$.

Since we study actuator redundant systems in this chapter, the following assumption is naturally required.

Assumption 1. *The following is assumed throughout the chapter*

i) $k - m > 0$

ii) B *is full row rank*

In order to exploit the redundancy, we introduce pseudoinverse and nullspace projector.

Definition 11 (Pseudoinverse and nullspace projector)**.** *The pseudoinverse B^+ can be calculated as*

$$B^+ = B^T(BB^T)^{-1} \tag{5.3}$$

The nullspace projector $B_0 \in \mathbb{R}^{k\times k}$ can be calculated as $B_0 = I_k - B^+B$, where I_k is the identity matrix, such that $BB_0 = 0$.

The matrix B^+ from (5.3) has full column rank under Assumption 1. Define a full rank matrix $B_\perp \in \mathbb{R}^{k\times(k-m)}$ as a basis for the nullspace of B. Note that $B_\perp$ and B_0 have the same column space, but B_0 is not full rank.

5.3 Optimal Dynamic Allocator

In this section, the control allocation problem is formulated as a dynamic optimal design. A general solution to the problem is provided in Subsection 3.1, and it is shown in Subsection 3.2 that the existing results in the literature [120] and [122] are special cases of our general solution.

The problem of control allocation is to select the actuators u in (2) that guarantee the desired control inputs v. The control inputs v are selected independently by a state feedback design that ensures the stability of outer loop (5.1) (see Stevens Lewis (2003)).

Consider the dynamic control allocation

$$\dot{\omega}(t) = \mu(t) \tag{5.4}$$

$$\begin{aligned} u(t) &= u_c(t) + B_\perp \omega(t) \\ u_c(t) &\triangleq B^+ v(t) \end{aligned} \tag{5.5}$$

where $\omega(t) \in \mathbb{R}^{k-m}$ is the allocator state and $\mu(t) \in \mathbb{R}^{k-m}$ is the allocator input. The actuator vector u in (2) is selected by (5.4) and termed allocator output, while $u_c(t) \in \mathbb{R}^k$ in (5.4) is termed the initial candidate allocation. All possible actuator allocations may be obtained by varying ω.

Note that for $\omega = 0$ equation (5.4) becomes $u(t) = u_c(t)$. Therefore, $u_c(t)$ solves (5.2). However, the goal of control allocation is not only to solve (5.2), but also to select $u(t)$ that efficiently allocates the actuators. This energy efficiency is measured by additional performance index introduced in Definition 3. It is important to note that the cost in Definition 3 is dynamic. The dynamic optimality of actuators is more desirable than static optimality be-cause the outer loop dynamically

changes in the con-trol allocation expression (5.5). Moreover, in Section 5 we additionally require actuators constraints on . In order to dynamically select actuators , we introduce the dy-namics in (5.4). This mechanism allows us to explore the space of valid by modifying in (5.5) and select that is energy efficient and satisfies constraints.

The following lemma formally shows that the control allocator (5.5) decouples the outer loop system (5.1) from allocator dynamics (5.4).

The control allocator (5.5) makes (5.1) independent from (5.4).

Proof. Use (5.2) and (5.5) to write the dynamics (5.1) as $\dot{x} = f(x) + G(x)(BB^{+}v + BB_{\perp}\omega)$. Since $BB_{\perp}$ and $BB^{+} = I$, we see that $\dot{x} = f(x) + G(x)v$. Therefore, the control allocation (5.5) makes x in (5.1) independent from in (5.4). □

Lemma 5.3 is important because it shows that the stability of the outer loop (5.1) is independent of allocator dynamics (5.4) when actuators u are determined by (5.5). In other words, the outer loop state x is independent of ω in (5.5). Therefore, we can safely modify ω in order to explore different actuators u without ever affecting the stability of outer loop (5.1). However, note that Lemma 5.3 does not imply that (5.4) is independent of (5.1). On the contrary, outer loop system is affecting allocator dynamics (5.4) through equation (5.5).

5.3.1 Optimal Dynamic Allocator Problem

The control allocation problem is now formulated as a dynamics optimization problem.

Definition 12 (Optimal Dynamic Allocator (ODA))**.** *The performance index (PI) of the dynamics allocator (5.4),(5.5) is defined as*

$$J^{\mu}(\omega(t), u_c(t)) \triangleq \int_t^{\infty} e^{-\alpha(\tau - t)} \left\{ u^T Q u + ||\mu||_2^2 \right\} \tag{5.6}$$

where $Q = Q^T > 0$ and $\alpha > 0$. The ODA problem is to determine the allocation input policy

$\mu^*(\omega, u_c)$ in (5.4) that minimizes (5.6)

$$\mu^*(\omega, u_c) \triangleq argmin_\mu J^\mu(\omega, u_c) \tag{5.7}$$

The exponential weighting factor α makes the future stage costs approach zero and also bounds the infinite time horizon integral in (5.6).

Use Leibniz differentiation rule to write the differential version of (5.6)

$$\begin{aligned} H^\mu(\omega, u_c) \triangleq ||\mu||_2^2 + u^T Q u - \alpha J^\mu(\omega, u_c) \\ + \dot{u}_c^T \nabla_{u_c} J^\mu(\omega, u_c) + \dot{\omega}^T \nabla_\omega J^\mu(\omega, u_c) \end{aligned} \tag{5.8}$$

where H is Hamiltonian and ∇ denotes the operator of partial derivative. The solution (5.7) can be obtained by solving the corresponding Hamilton-Jacobi-Bellman (HJB) equation

$$0 = -min_\mu H^\mu(\omega, u_c) \tag{5.9}$$

In the following theorem we solve the ODA with the assumption that u_c is constant. It is shown in next subsection that the existing control allocation solutions [120] and ([122] are special cases of this result.

Theorem 1 (Static Allocation)**.** *Consider the non-linear system (5.1) and (5.2) with the allocator design (5.4), (5.5) and let u_c in (5.5) be constant ($\dot{u}_c = 0$). Then, the optimal allocator input (5.7) is given by*

$$\mu^*(\omega, u_c) = -P_{11}\omega - (P_{11} + \alpha)^{-1} B_\perp^T Q u_c \tag{5.10}$$

$$P_{11} = \sqrt{(\alpha/2)^2 + B_\perp^T Q B_\perp} - \alpha/2 \tag{5.11}$$

Proof. Apply stationarity condition on Hamiltonian (8) $\nabla_\omega H^\mu(\omega, u_c) = 0$, to find the expression for the optimal allocation input $\mu^* = -(1/2)\nabla_\omega J^{\mu^*}(\omega, u_c)$. Assume that the optimal cost function has the quadratic form $J^{\mu^*}(\omega, u_c) = [\omega^T \quad u_c^T] P [\omega^T \quad u_c^T]^T$ then the gradient is $\nabla_\omega J^{\mu^*}(\omega, u_c) = 2[P_{11} \quad P_{12}][\omega^T \quad u_c^T]^T$ where P_{12} and P_{11} are upper block matrices of matrix P. The optimal

allocation input is then

$$\mu^*(\omega, u_c) = -P_{11}\omega - P_{12}u_c$$

Insert the cost $J^{\mu^*}(\omega, u_c)$, the gradient of the cost $\nabla_\omega J^{\mu^*}(\omega, u_c)$ and $\mu^*(\omega, u_c)$ in (5.8) to express Hamiltonian. Now use the Hamiltonian in (5.8) and state augmentation

$$\omega_a \triangleq [\omega^T, u_c^T]^T$$

to discover that the equation (5.9) is Algebraic Ricatti Equation (ARE)

$$\begin{aligned} 0 = {} & \omega_a^T [B_\perp \;\; I_k]^T Q [B_\perp \;\; I_k] \omega_a + \omega_a^T [P_{11} \;\; P_{12}]^T [P_{11} \;\; P_{12}] \omega_a \\ & - 2\omega_a^T [P_{11} \;\; P_{12}]^T [P_{11} \;\; P_{12}] \omega_a - \alpha \omega_a^T P \omega_a^T \end{aligned} \tag{5.12}$$

To determine P_{11} and P_{12}, extract the upper block ma-trices in equation (5.12). This gives us matrix equations $0 = B_\perp^T Q B_\perp - P_{11}P_{11} - \alpha P_{11}$ and $0 = B_\perp^T Q - P_{11}P_{12} - \alpha P_{12}$. The first equation is a standard ARE with analytical solution given by (5.11). Since $\alpha > 0$ and P_{11} is solution to the Riccati equation, then $P_{11} + \alpha$ is invertible and $P_{12} = (P_{11} + \alpha)^{-1} B_\perp^T Q$. This confirms that the solution $\mu^*(\omega, u_c) = -P_{11}\omega - P_{12}u_c$ is given by (5.10).

To confirm the stability of allocator dynamics (5.4) for $\mu = \mu^*(\omega, u_c)$, note that the state feedback gain $-P_{11}$ in (5.10) is negative definite for $\alpha > 0$. □

5.3.2 Comparison with Optimal Static Allocators

In this section we show that a special case of the ODA solution in Theorem 1 solves the standard allocation problems in the literature such as [80] and [122].

The problem presented in [122] is to solve

$$\begin{aligned} & min_\omega J_s(u) \triangleq min_\omega u^T Q u \\ & s.t. \;\; u = B^+ v + B_\perp \omega \end{aligned} \tag{5.13}$$

and the solution is

$$\begin{aligned} u_1^* &= B^+ v + B_\perp \omega^* \\ \omega^* &= -(B_\perp^T Q B_\perp)^{-1} B_\perp^T Q u_c \end{aligned} \tag{5.14}$$

A different formulation of the allocation problem that is also common in the literature [120] is given by

$$\begin{aligned} & min_u J_s(u) \triangleq min_u u^T Q u \\ & s.t. \;\; v = Bu \end{aligned} \tag{5.15}$$

with the solution

$$u_2^* = Q^{-1} B^T (B Q^{-1} B^T)^{-1} v \triangleq B^Q v \tag{5.16}$$

Where B^Q is the weighted right inverse of matrix B.

The next two theorems show that these are both special cases of solution (5.10) in Theorem 1. To start, we will now show that the solutions (5.14) and (5.16) are equivalent.

Theorem 2. *The solutions (5.14) and (5.16) are equivalent.*

Proof. First use

$$B_0 B^Q = B^Q - B^+ B B^Q = B^Q - B^+ \tag{5.17}$$

to express B^Q as $B^Q = B^+ + B_0 B^Q$. Now use this form of B^Q to rewrite (5.16) as

$$u_2^* = B^Q v = B^+ v + B_0 B^Q v \tag{5.18}$$

Since B_0 and $B_\perp$ have identical column-space then there is $F_2 \in \mathbb{R}^{(k-m)\times k}$ such that $B_0 = B_{_2}$. The solution u_2^* can then be written as

$$\begin{aligned} u_2^* &= B^+ v + B_\perp \xi \\ \xi &\triangleq F_2 B^Q v = F_2 B^Q B u_c \end{aligned} \tag{5.19}$$

which is in the same form as the solution u_1^* in (5.14) with $\xi \in \mathbb{R}^{k-m}$ corresponding to ω^*. From

convexity of (5.13) and (5.15), identical feasibility domains and identical forms of solutions (5.17) and (5.14) we infer that the two problems have equivalent solutions. □

The next theorem is our main result in this section. It shows that the standard allocation solutions in the literature [80] and [122] are special cases of ODA solution in Theorem 1.

Theorem 3. *Consider the allocator input (5.10) with $\alpha = 0$ and constant u_c. Then the allocator dynamics (5.4) becomes*

$$\begin{aligned} \dot{\omega} &= KB_\perp^T Qu \\ K &= -(B_\perp^T QB_\perp)^{-1/2} \end{aligned} \tag{5.20}$$

Moreover, the allocation output (5.5) converges to the steady state value

$$u(t \to \infty) = (I - B_\perp K^2 B_\perp^T Q)u_c \tag{5.21}$$

which is equivalent to the solution (5.14).

Proof. The allocator dynamics (5.4) with allocator control input (5.10) and $\alpha = 0$ is

$$\dot{\omega} = -P_{11}\omega - P_{11}^{-1}B_\perp^T Qu_c = -P_{11}^{-1}(P_{11}^2\omega + B_\perp^T Qu_c) \tag{5.22}$$

Use (5.11) to express P_{11}^2 and write

$$\dot{\omega} = -P_{11}^{-1}(B_\perp^T QB_\perp\omega + B_\perp^T Qu_c) \tag{5.23}$$

Finally, use (5.5) and (5.11) to show that

$$\dot{\omega} = KB_\perp^T Qu \tag{5.24}$$

which proves (5.20).

The steady state of dynamical equation (5.20) with u in (5.5) is

$$\omega(t \to \infty) = -(B_\perp^T Q B_\perp)^{-1} B_\perp^T Q u_c = -K^2 B_\perp^T Q u_c \tag{5.25}$$

The steady state allocator output (5.5) is then

$$\begin{aligned} u(t \to \infty) &= u_c + B_\perp \omega(t \to \infty) \\ &= (I - B_\perp K^2 B_\perp^T Q) u_c \end{aligned} \tag{5.26}$$

which proves (5.21). It is now easy to see that $u(t \to \infty)$ in (5.21) and u_1^* in (5.14) are equivalent.

□

The allocator dynamics (5.16) was used in ([122], where (5.20) was also derived. Theorem 3 recovers their result as a special case of Theorem 1 where the initial candidate allocation u_c is assumed constant. Both optimization techniques lead to the same steady-state solution (19) that minimizes static optimization cost (13), but our method shows how to select a specific K in (5.20) that also minimizes the dynamic cost functional (6) and not only static objective (13). Moreover, our solution also solves (15) since Theorem 2 shows that solutions (14) and (16) are equivalent.

5.4 Linear Quadratic H_∞ allocation

Theorem 1 solves a specific case of ODA (5.7) assuming $\dot{u}_c = 0$. Theorem 3 shows that existing results are special cases of Theorem 1. In this section, Theorem 4 presents more general result than Theorem 1 that allows us more design freedom, including adding actuator constraints in Section 5.

The assumption $u_c(t) = 0$ in Theorem 1 virtually decouples the Hamiltonian (5.7) from outer loop dynamics. However, in general, the Hamiltonian (5.8) depends on outer loop dynamics so that $\dot{u}_c \neq 0$. Since the optimality of control allocator design (5.4), (5.5) is directly related to Hamiltonian (5.8), it is crucial to cover the general case $\dot{u}_c \neq 0$ to guarantee optimality.

In this section, we generalize the ODA solution to the case $\dot{u}_c \neq 0$. To start, we formally define the problem.

Definition 13 (H^∞ Optimal Dynamic Allocator (ODA)). *Consider the H^∞ performance index*

$$J^\mu(w,d) = \int_t^\infty e^{-\alpha(\tau-t)}(u^TQu + ||u||_2^2 - \gamma^2||d^j||_2^2)d\tau \tag{5.27}$$

where $w \triangleq [\omega^T \quad u_c^T]^T$ is the augmented state, $d = x$ is the disturbance $\gamma > 0$ tunes the amount of disturbance rejection effort. The H^∞ ODA problem is to determine the policy that minimizes (5.27) for the worst case disturbance

$$J^*(w) = J^{\mu^*}(w, d^*(w)) = min_\mu max_d J^{\mu^*}(w,d) \tag{5.28}$$

Define the optimal policy and the worst case disturbance

$$\begin{aligned} \mu^* &\triangleq arg\ min_\mu\ J^\mu(w,d^*) \\ d^* &\triangleq arg\ max_d\ J^\mu(w,d) \end{aligned} \tag{5.29}$$

If solutions (**??**) exist, then they form a zero-sum game saddle point solution for the system (5.4),(5.5) and they are said to be in Nash equilibrium.

Assumption 2. *Let the stabilizing outer loop control $v(t)$ and the outer loop system (5.1) be linear in state x so that*

$$\begin{aligned} \dot{x} &= Ax + Gv \\ v &= Kx \end{aligned} \tag{5.30}$$

This assumption is general and covers, for in- stance, all the cases of linearized aircraft dynamics with linear feedbacks designed using standard flight dynamics control techniques (see, e.g. [96]).

The next result solves the H_∞ ODA for this linear case. The main idea is to construct the explicit expression for dynamics u c assuming the linear feedback control (5.30) in the outer loop, while x in (5.1) is considered as disturbance $d \triangleq x$.

Defining the outer loop state as disturbance $d \triangleq$ allows us to write the actuator dynamics (5.4),

(5.5) in the standard H^∞ control form as shown in Theorem 4.

Theorem 4. *Let assumptions from this chapter hold. Consider the infinite horizon H^∞ cost (5.27) for system (5.4), (5.5) written in the form of augmented state $w \triangleq [\omega^T \quad u_c^T]^T$. Assume that the optimal value function has the form $J^*(w) \triangleq w^T P w$. Then, the optimal control allocation input is*

$$\mu^* = -B_a^T P w \tag{5.31}$$

where P solves the algebraic Riccati equation (ARE)

$$0 = C_a^T Q C_a - P(B_a B_a^T - \gamma^{-2} D_a D_a^T)P + P(A_a - \alpha/2) + (A_a - \alpha/2)^T P \tag{5.32}$$

that corresponds to the dynamics of the augmented state

$$\begin{aligned} \dot{w} &= A_a w + B_a \mu + D_a d \\ u &= C_a w \end{aligned} \tag{5.33}$$

where

$$A_a = \begin{bmatrix} 0 & 0 \\ 0 & B^+ K G B \end{bmatrix} \quad B_a = \begin{bmatrix} I_{k-m} \\ 0 \end{bmatrix} \quad D_a = \begin{bmatrix} 0 \\ B^+ K A \end{bmatrix} \quad C_a = \begin{bmatrix} B_\perp & B^+ \end{bmatrix} \tag{5.34}$$

Proof. Start by differentiating $u_c = B^+ v$ in (5.5) with respect to time and use equations (5.2), (5.5) to time and use Assumption 2 and equations (5.2), (5.5) to write

$$u_c = B^+ K A x + B^+ K G B (u_c + D_\perp \omega) \tag{5.35}$$

Finally,

$$\dot{u}_c = B^+ K A x + B + K G B u_c \tag{5.36}$$

since $B B_\perp = 0$. Use (5.4), (5.5) and (5.36) to write the dynamics of augmented state $\dot{w}$ in (5.33).

To find the solution to H^∞ ODA (5.29) we will follow steps similar to Theorem (1). First, write the Hamiltonian

$$H^\mu(w,d) = u^T Q u + ||\mu||_2^2 - \gamma^2 ||d||_2^2 + \dot{w}^T \nabla_w J^\mu(w,d) - \alpha J^\mu(w,d) \tag{5.37}$$

that corresponds to the cost (5.27) subject to dynamics (5.33). Then, the Hamilton-Jacobi-Isaacs equation is

$$0 = -\underset{d}{max}\ \underset{\mu}{min} H^\mu(w,d) = -H^{\mu^*}(w,d^*(w)) \tag{5.38}$$

Use the stationarity conditions $\nabla_\mu H^\mu(w,d) = 0$, $\nabla_d H^\mu(w,d) = 0$ to find the control input that minimizes (5.38) and the worst case disturbance that maximizes (5.38)

$$\begin{aligned} \mu^* &= -\frac{1}{2} B_a^T \nabla_w J^*(w) \\ d^* &= \frac{1}{2\gamma^2} D_a^T \nabla_w J^*(w) \end{aligned} \tag{5.39}$$

with the gradient of the optimal cost equal to $\nabla_w J^*(w) = \nabla_w w^T P w = 2Pw$. Notice that (5.38) is obtained by inserting (5.39) into (5.37) and it results in ARE (5.32). □

The solution (5.31) in Theorem 4 is a generalization of the solution (5.10) in Theorem 1 since it does not assume $u_c(t) = 0$. Theorem 1, on the other hand, is a generalization of static control allocation results (5.14), (5.16).

Moreover, the H^∞ design provides the allocator (5.5) with robustness against outer loop dynamics (5.1) by treating x as a disturbance. However, the model of the outer loop dynamics (5.30) is needed to determine the solution. In next section, we solve H^∞ ODA in a model-free fashion using machine learning technique.

5.5 Constrained model-free optimal control allocation

In this section, we introduce a cost function specifically designed to ensure constrained actuators while preserving the optimality results. We then develop a Reinforcement Learning (RL) [61]

method that determines the optimal control allocation online by measuring input-output data in real time. Specifically, we employ the off-policy Integral Reinforcement Learning (IRL) algorithm as in [69].

Consider the performance index

$$\begin{aligned} J_c^{\mu}(w,d) &= \int_t^{\infty} e^{-\alpha(\tau-t)}\big(Q(u) + ||\mu||_2^2 - \gamma^2||d||_2^2\big)d\tau \\ Q(u) &= u^T Q u + \beta \sum_{r=1}^{k} (u^r - u_U^r)^{-2l} + (u^r - u_L^r)^{-2l} \end{aligned} \tag{5.40}$$

where β is a positive scalar, u^r denotes r-th element of vector u, $u_{L\,/U}^r$ are design parameters that bound the value of u^r and l is a positive integer. The cost term $Q(u)$ in (5.40) guarantees constrained actuators $u_L^r \leq u^r \leq u_U^r$.

Evaluating the system (5.33) by cost (5.40) induces the Bellman equation

$$H_c^{\mu}(w,d) = Q(u) + ||\mu||_2^2 - \gamma^2||d||_2^2 + \dot{w}^T \nabla_w J_c^{\mu}(w) - \alpha J_c^{\mu}(w) \tag{5.41}$$

The optimal solution $J_c^*(w) = J_c^{\mu}(w, d^*)$ satisfies

$$\min_{\mu} \max_{d} H_c^{\mu}(w,d) = H_c^{\mu^*}(w,d^*) \tag{5.42}$$

5.5.1 Online Reinforcement Learning (RL) algorithm

In this section, we introduce RL to minimize (30). This procedure generates decreasing cost iterates $J_c^{j+1}(w) < J_c^j(w)$ and policy iterates as

$$\begin{aligned} \mu^{j+1} &= -\frac{1}{2} B_a^T \nabla_w J_c^j(w) \\ d^{j+1} &= -\frac{1}{2\gamma^2} D_a^T \nabla_w J_c^j(w) \end{aligned} \tag{5.43}$$

These policies (5.43) are shown to converge to the optimal solution of (5.42), that is

$$\lim_{j\to\infty} H_c^{\mu^j}(w, d^j) = H_c^{\mu^*}(w, d^*) \tag{5.44}$$

The method in [69] consists of an off-policy RL algorithm that determines (5.43) without using any model information. We have used this method to solve the ODA problem, as given in Algorithm 1 below. In this off-policy algorithm, policy updates (5.43) are obtained by solving the Integral Reinforcement Learning (IRL) Bellman Equation (5.45)

$$\begin{aligned} e^{-} J_c^j(w(t+T)) - J_c^j(w(t)) = \int_t^{t+T} e^{-\alpha(\tau-t)}\big(Q(u) + ||\mu^j||_2^2 - \gamma^2||d^j||_2^2 + \\ 2(\mu-\mu^j)^T\mu^{j+1} - 2\gamma^2(d-d^j)^T d^{j+1}\big)d\tau \end{aligned} \tag{5.45}$$

This equation is an integral version of the Bellman equation (5.41) where (5.43) was used. For a complete description of the derivation of (5.45), please refer to [69], equations (58) - (63).

Algorithm 5.1 ***Intelligent control allocator***

Input: Initialize policies $\mu^0(w)$, $d^0(w)$.
Output: Optimal control allocator $\mu^*(w)$
Initialization: Initialize policies $\mu^0(w)$, $d^0(w)$.
while $||K_i - K_{i-1}|| = ||B^T P_i - B^T P_{i-1}|| > \epsilon$ **do**
Step 1 Fix the policies $\mu^j(w)$, $d^j(w)$ and let μ and μ act on the system (5.33). Store i -th sample of (5.45) by collecting data.
Step 2 Determine $J_c^j(w), \mu^{j+1}(w), d^{j+1}(w)$ by solving the system of i IRL equations (5.45)

Theorem 5. *Let $\mu^0(w)$ be stabilizing. Then, Algorithm 5.1 converges to the optimal solution.*

Proof. [58] proved that policy iteration with a non-quadratic positive definite $Q(u)$ converges. [69] proved that off-policy IRL gives the same policy updates and final solution as the policy iteration algorithm. Using both results, we conclude that Algorithm 5.1 converges to the optimal policies. □

5.5.2 Approximate Neural Network Solution to IRL

In order to solve (5.45), the cost and the policies (5.43) are approximated by 3 neural networks (NN)

$$\begin{aligned} J_c^j(w) &= W_J^j \sigma_J(w) \\ \mu^j(w) &= W_\mu^j \sigma_\mu(w) \\ d^j(w) &= W_d^j \sigma_d(w) \end{aligned} \tag{5.46}$$

where W_μ^j, W_d^j, W_J^j are NN weights and σ_μ^j, σ_J^j and σ_d^j are non-linear basis sets.

The i-th sample of IRL equation (5.45) approximated by (5.46) is linear in NN weights and can be written as

$$\begin{aligned} W^j h_i^T &= y_i; \;\; W^j = [W_J^j,\; vec(W_\mu^j)^T,\; vec(W_d^j)^T] \\ h_i &= \begin{bmatrix} e^{-\alpha T}\sigma_J(w(t+T)) - \sigma_J(w(t)) \\ 2\int_t^{t+T} e^{-\alpha(\tau-t)}(\mu - \mu_W) \otimes \sigma_\mu(w) d\tau \\ -2\gamma^2 \int_t^{t+T} e^{-\alpha(\tau-t)}(d - d_W) \otimes \sigma_d(w) d\tau \end{bmatrix} \\ y_i &= -\int_t^{t+T} \left(Q(u) + ||\mu^j||_2^2 - \gamma^2 ||d^j||_2^2 \right) \end{aligned} \tag{5.47}$$

where $\otimes$ denotes Kronecker product and $vec(\cdot)$ is a stack of matrix columns.

In order to solve the IRL equation in step 2 of the algorithm, the number of samples has to be at least equal to number of NN weights $i \geq N = N_j + N_\mu + N_d$. The step 2 of the algorithm can then be executed as a least squares update.

One of the main contributions of this work is that we do not require knowledge of outer loop dynamics (5.1) nor feedback gain K in order to perform Algorithm 5.1 since these do not appear in (5.47). Note also that μ_j and d_j are updated in the algorithm, but are not required to be applied to system (5.1).

5.6 Experimental Validation

In this final section, we experimentally verify that The H^∞-ODA allocation solution (5.31) in Theorem 4 stabilizes the allocator dynamics (5.4),(5.5).

ii) The policy $\mu_j(w)$ in Algorithm 1 without actuator constraints ($\beta = 0$) in (5.40) converges to solution (5.31).

iii) Setting $\beta = 1$ in (5.40) constrains the actuators. The simulation shows that these constraints are obeyed. Consider F16 flight control system in (5.1) to be

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \delta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} v \tag{5.48}$$

where α is the angle of attack (AoA), q is the pitch rate and δ is the elevator deflection angle. This is flight control example in 5.4-1 in Stevens Lewis, (2003). Define x in (5.1) to be $x = [\alpha^T \;\; q^T \;\; \delta^T \;\; (\alpha - \alpha_r)^T]^T$ where α_r is the reference value for AoA. The outer loop feedback gain in (5.30) is selected to stabilize (5.1) as $K = [2.16 \;\; 1.66 \;\; 0.2 \;\; 2.72 \;\; 1.66 \;\; 0.2]$.

The allocation matrix in (5.2) is selected as $B = [1 \;\; 1 \;\; 1 \;\; 1]$. This corresponds to modern aircraft designs where flight controls are distributed into multiple trailing edge actuators to reduce singe-point-of-failure and improve reliability. The cost function parameters in (5.27) and (5.40) are $\gamma = 1.5$, $\alpha = 0.01$ and $Q = diag([1 \;\; 5 \;\; 25 \;\; 125])$

Simulate the outer loop system (5.1) with allocation design (5.4), (5.5) and select the allocation input $\mu(w)$ as (5.31). The simulated actuators are displayed by dashed lines in Figure 1. This experimentally confirms the Statement i). Next, the online Algorithm 1 with no constraints ($\beta = 0$ in 5.40) is run and the results are the solid lines in Figure 1. We introduce function approximators (5.46) as $\sigma_J(w) = w \otimes w$ $\sigma_\mu(w) = w$ and $\sigma_d(w) = w$ and perform Algorithm 5.1 with $N = 250$ episodes of data collection, each $T = 0.15$ seconds long. In first 37.5 seconds of simulation non-optimal behavior policy $\mu^0(w)$ is applied to (5.4). At $t = 37.5s$ the learned policy $\mu^{j=1}(w)$ is applied to (5.4). The policy $\mu^j(w)$ is then continuously updated using new data samples until convergence. After convergence, they conform closely to the dashed lines confirming Statement ii).

Moreover, Figure 2 shows the Frobenius distance between optimal gain $-B_a^T P$ in (5.31) and

W_μ^j in (5.46) for every policy update. The policy weights W_μ^j converge to the optimal gain $-B_a^T P$ in less than 15 iterations with the norm difference of 3.2641. After that, the improvement is slow and the error reaches the value 1.1966 at iteration 60. In conclusion, the policy $\mu^j(w)$ converges to its optimal value, which again confirms the Statement ii).

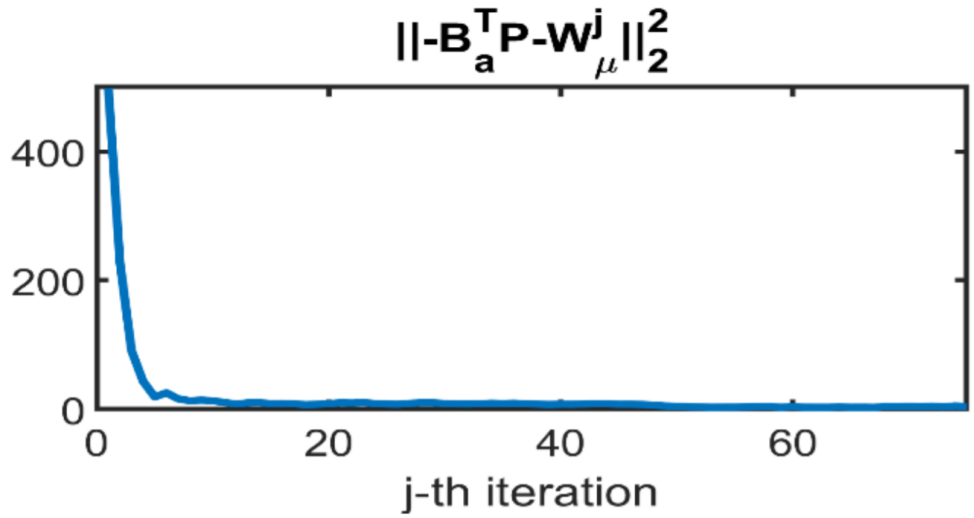

Figure 5.1: The norm distance between the optimal feedback gain matrix $-B_a^T P$ in (5.31) and the learned policy weights W_μ^j in (5.46) updated by least squares.

Finally, Statement iii) is addressed by constraining the first two actuators (u^1, u^2) inside the range $[u_L^{1,2},\ u_U^{1,2}]$ and setting $\beta = 1$, $l = 4$ in (5.40). The features $\sigma_\mu(w)$, $\sigma_d(w)$ are taken as polynomials of all powers up to 5 and $\sigma_J(w)$ of all even powers up to 6. All other parameters and initialization are the same as in previous simulation. The learned policy is applied at $t = 37.5s$.

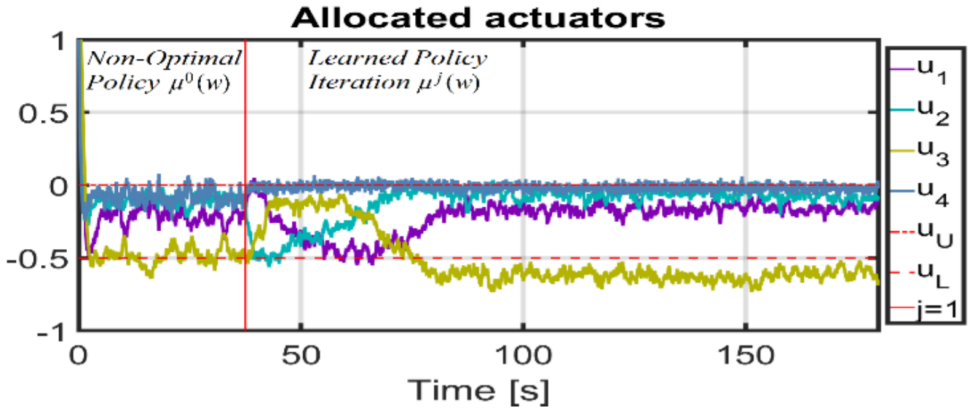

Figure 5.2: Actuators allocated by policies $\mu^j(w)$ that minimize (5.40) for constrained case ($\beta = 1$). Constraints are imposed on actuators (u^1, u^2) selected as $u_L^{1,2}, u_U^{1,2} = (-0.5, 0.0)$.

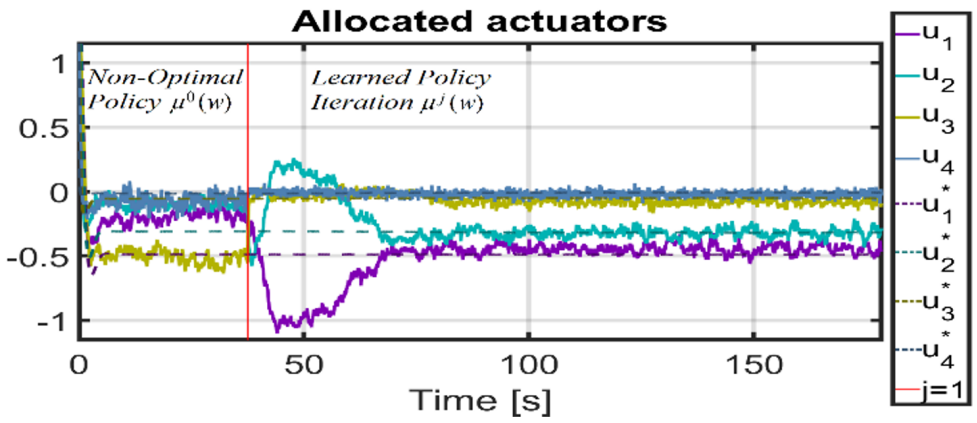

Figure 5.3: Dashed lines show the actuators allocated by optimal control allocation input (5.31). Solid lines show the actuators allocated by policy $\mu^j(w)$ in (5.43). The initial stabilizing policy $\mu^0(w)$ is applied during data collection phase $[0s \ - \ 37.5s]$. The policy $\mu^j(w), \ j = 1, 2, ...$ is then updated by least squares using new data samples.

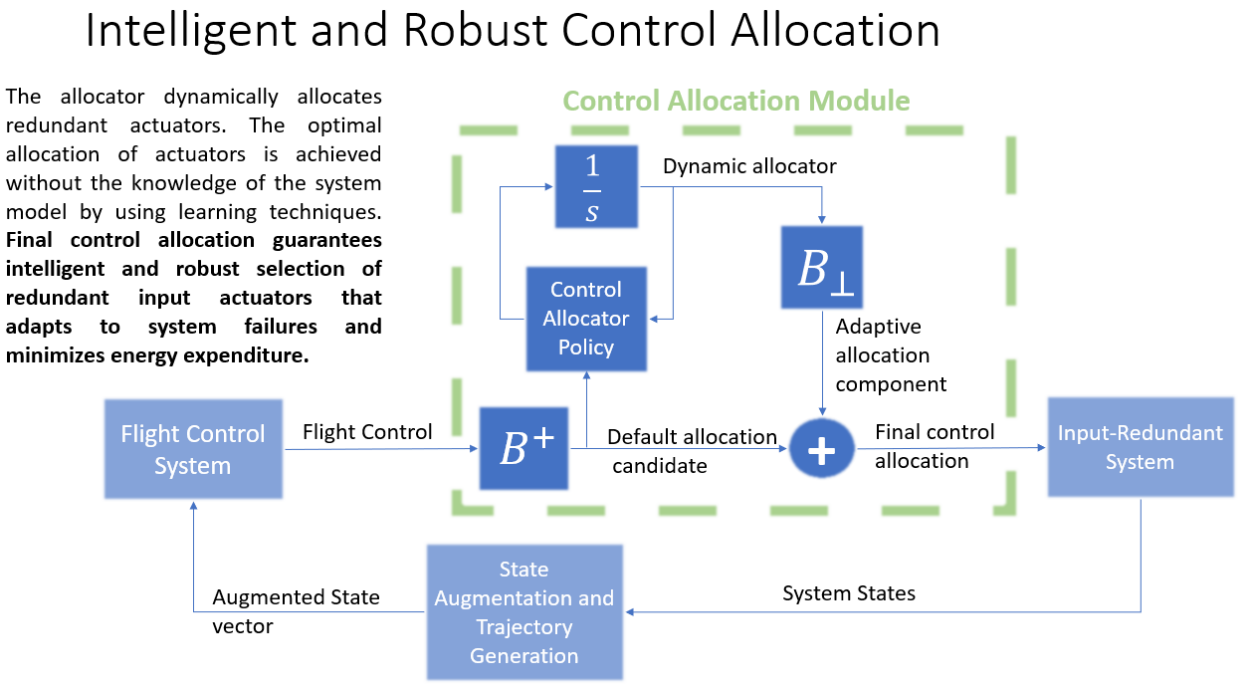

Figure 5.4: Overview of Dynamic Control Allocation Design

Compare Figure 3 with Figure 1 and note that during the learning period the actuators $u^{1,2}$ assume values outside the range $[-0.5, \ 0.0]$ in unconstrained case. On the other hand, the constrained case in Figure 3 shows that the actuators $u^{1,2}$ are successfully constrained at the cost of increase in value of the actuator u^3.

5.7 Conclusion

A novel approach for solving the dynamic control allocation problem is developed. We have proved that our approach is a generalization of standard solutions in the literature. A non-quadratic cost function is introduced that imposes constraints on the actuators. Finally, a reinforcement learning algorithm was successfully used to determine the optimal actuators in real-time.

Chapter 6: DISTRIBUTED FORMATION CONTROL OF MULTI-AGENT SYSTEM OF UAVS

6.1 Introduction

Cooperative control of multiple unmanned aerial vehicles (UAVs) that share a global task has recently drawn a lot of attention. The last decades have witnessed a rapid development of micro UAVs and the idea of multi-agent UAV systems performing tasks grew with it. This is due to the fact that a group of simple individual vehicles may facilitate the execution of a complex task with higher robustness to potential technical faults. Moreover, multi-agent systems also have an economical and logistical advantage, simply because the failure of a single agent does not put the whole mission into jeopardy. That makes a single agent in the system expendable and its design requirements less rigorous.

We are interested in the consensus control of multiple UAVs, especially quadcopters. It is shown in [22] that a two-loop nonlinear scheme using input-output linearisation technique can be employed to satisfy the tracking performance. Moreover, a neural network based backstepping design for UAV is provided in [21], where an inverse kinematics solution is applied akin to that in the manipulator control such as [62]. Non-linear models of UAV and linearization procedure is covered in [115], [85] New control techniques have been proposed to allow multiple vehicles to work cooperatively (see [79], [87], [5], [65]).

More recently, further work is considered to quantify the performances of the whole multi-UAV system and the local UAV. Its formulation falls into a category of cooperative optimal control and game theory. In particular, [124] uses linear quadratic regulator based optimal control to achieve the synchronization of linear systems. Non-zero-sum games are studied for multiple players in [111], [110], [108], where the adaptive algorithm learns online the solution of coupled Riccati equations and coupled HamiltonâJacobi equations. Even though good results were obtained, little research has been carried out to handle the measurement noise imposed on each local UAV.

Note that when the noise is injected into the measurement signals of each local UAV, the imperfect information propagates through distributed control protocols, which makes the multi-UAV system vulnerable to uncertainties and thus severely degrades the cooperative performance. Due to the fact that this work considers actual implementation of the system in the lab environment, it is of great importance to treat the poor information quality in the control law accordingly. To address that problem, [121] proposed an adaptive stabilization control for cyber-physical systems against measurement noises. Moreover, a distributed adaptive leaderless control for multi-agent systems was proposed in [3]. However, current results in ([121], [3]) require that the measurement noises are restricted to certain simple types. More importantly, these papers only theoretically analyzed the measurement noise problem in the cooperative control, but few results have been experimentally validated.

In this chapter, we aim to design the cooperative control of multiple UAVs under measurement noises with theoretical analysis and physical experiments. To do this, we first model the dynamics of each individual UAV and its measurement noises. Based on the sliding mode control design, we propose a distributed control protocol that stabilizes all signals in the closed-loop multi-UAV systems and ensures that the consensus errors are uniformly ultimately bounded. Moreover, we use local information to define the trustworthiness, based on which we tune the edge weights to compensate for the negative effect from noisy measurements. In addition, we bridge the gap between the theoretical framework and the physical experiment and implement our results by using Crazyflie 2.0 platform. In our experiments, we utilize the motion capture system (VICON) to coordinate UAVs through the communication topology based on a master-slave model. Furthermore, a hierarchical motion control state machine is designed in a distributed fashion and implemented as a software solution based on robot operating system environment (ROS). To ensure low latency and smooth control loop, data pipeline is implemented as a multi-threaded routine specifically tailored for the high-speed control requirements of dynamically rich UAVs. Several experiments are designed to validate the effectiveness of our algorithms.

The remaining parts of this chapter are outlined as follows. In Section II, some preliminaries

on graph theory are provided. In Section III, the dynamics of UAV are detailed. In Section IV, the controller for coordinating UAVs is proposed, and its stability analysis is carried out. In Section V, the trustworthiness based approach is proposed to tune the edge weight. The experimental environment is detailed in Section VI, and the experimental results are given in Section VII. Finally, the conclusion is contained in Section VII.

6.2 Preliminaries on Graph Theory

In this section, we briefly give some preliminaries on graph theory that will be used in our analysis and experiments.

Definition 14. *Let graph G be the pair*

$$G = (V, E), \tag{6.1}$$

where V denotes vertices (nodes) of a graph and E denotes the edges of a graph. In that context V can be represented as

$$V = \{v_1, v_2, ..., v_N\}, \tag{6.2}$$

where v_i is vertex (node) i of a graph. Nodes are connected by links (edges). Edge E

$$E = (v_i, v_j), \forall i, j \in [0, N], i \neq j \tag{6.3}$$

is a connection between two nodes i and j.

If edges are not directed (which means that they are bidirectional), then the graph is said to be undirected. Each edge has a weight that can model the strength of the connection between the nodes. The number of edges that enter a node is called in-degree, while the number of edges that exit a node is called out-degree. The graph can be presented in the form of a matrix where column index corresponds to a source of an edge and row index to a sink of an edge. Matrix element a_{ij} indicates the weight of the corresponding edge in the graph. Matrix constructed of elements a_{ij}

is called adjacency matrix ($\mathcal{A}$). $\mathcal{A}$ is very convenient as it enables analysis of graph theory from the perspective of linear algebra. Structural properties of the graph are embedded in matrix $\mathcal{A}$ and can be extracted by examining that matrix. $d_i = \sum_{i=1}^{n} a_{ij}$ is a row sum of $\mathcal{A}$ and it is introduced for convenience in the future analysis. Matrix $\mathcal{D}$ is diagonal matrix of in-degrees, $\mathcal{D} = diag(d_i)$. Therefore, the Laplacian matrix is formulated as $\mathcal{L} = \mathcal{D} - \mathcal{A}$.

6.3 Quadrotor dynamics

When dealing with UAVs, states used for expressing dynamics are position ξ_i, velocity $\dot{\xi}_i$, attitude η_i and angular rate $\dot{\eta}_i$. Quadrotor dynamics are expressed as follows

$$m\ddot{\xi}_i = F_g + F_d(\eta_i), \tag{6.4}$$

$$J(\eta_i)\ddot{\eta}_i = C(\eta_i, \dot{\eta}_i) + \tau_i, \tag{6.5}$$

where F_g is a gravitational force, and F_d is a nonlinear input that depends on the attitude, η are rotational states (roll, pitch and yaw), and τ is the torque generated by propellers.

Note that the focus of this chapter is on efficiency of the cooperative control policy rather than the design of effective dynamical controller for a single UAV. Without the loss of generality, certain simplifications are used. UAVs that we worked with are very lightweight (27 g) which permits us to linearize the dynamical system. Consensus controller will be tested only in the x-y plane. After the linearization around the equilibrium point (similar to [53]), systems (6.4) and (6.5) can be reduced to following form

$$\ddot{x}_i = \begin{bmatrix} \ddot{\xi}_{x,i} \\ \ddot{\xi}_{y,i} \\ \ddot{\xi}_{z,i} \\ \ddot{\eta}_{r,i} \\ \ddot{\eta}_{p,i} \\ \ddot{\eta}_{y,i} \end{bmatrix} = \begin{bmatrix} -g\eta_{p,i} \\ g\eta_{r,i} \\ -\frac{f_{T,i}}{m} \\ \frac{\tau_{x,i}}{I_x} \\ \frac{\tau_{y,i}}{I_y} \\ \frac{\tau_{z,i}}{I_z} \end{bmatrix} \tag{6.6}$$

where the system state x_i is given as

$$x_i = \begin{bmatrix} \xi_{x,i} \\ \xi_{y,i} \\ \xi_{z,i} \\ \eta_{r,i} \\ \eta_{p,i} \\ \eta_{y,i} \end{bmatrix} \tag{6.7}$$

where I_x, I_y and I_z denote the idealized moments of inertia for each axis and m is the mass of the UAV. After separating the input terms from the state transition matrix, we can change (6.6) into

$$\ddot{x}_i = Ax_i + Bv_i, \tag{6.8}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.9}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}, \tag{6.10}$$

and the control input is

$$v_i = \begin{bmatrix} f_{T,i} \\ \tau_{x,i} \\ \tau_{y,i} \\ \tau_{z,i} \end{bmatrix}. \tag{6.11}$$

Considering that there are N UAVs in this chapter, we present the system dynamics in a global form as

$$\ddot{x} = (I_N \otimes A)x + (I_N \otimes B)v. \tag{6.12}$$

To model the group of agents with distributed information we use the graph theory. In the light of that, every link of the graph represents communication capability between connected agents, while nodes represent dynamical agents. The objective of this chapter is to make the follower quadrotors in the graph $\mathcal{G}$ track the desired trajectory based on the distributed information, where the trajectory is generated by the leader.

The objective of this section is to develop distributed cooperative control protocol. In this chapter we study how the distributed flow of information impacts the state of single dynamical agent. In cooperative control, each agent communicates only with its neighbors. The goal is to solve the consensus problem, where the resulting distributed control protocol drives all the states to the same constant steady-state value.

Consensus controller for each agent i

$$u_i = \sum_{j \in N_i} a_{ij}(x_j - x_i) + g_i(x_0 - x_i), \tag{6.13}$$

with a_{ij} the communication graph edge, and g_i is the weight gain describing the communication of the leader with agent i, known as the pinning gain.

6.4 Position consensus controller

We introduce Δ_i as a displacement of UAV i from the center of the formation. Different combinations of Δ_i can now be used to encode various formations. x_0 is the agent leader and is directly linked to those agents that have non-zero pinning gain g_i. The reference is generated by the leader and is effectively transmitted to the whole system through the distributed communication network.

The neighbourhood error is defined as

$$e_i = \sum_{j \in N_i} a_{ij}(x_j - \Delta_j - x_i + \Delta_i) + g_i(x_0 - x_i + \Delta_i), \tag{6.14}$$

and neighbourhood time derivative of the error

$$\dot{e}_i = \sum_{j \in N_i} a_{ij}(\dot{x}_j - \dot{\Delta}_j - \dot{x}_i + \dot{\Delta}_i) + g_i(\dot{x}_0 - \dot{x}_i + \dot{\Delta}_i). \tag{6.15}$$

If we assume that the position of the leader (x_0) and the relative positions between the agents ($\Delta_i, \forall i$) are not changing during the experiment then

$$\dot{e}_i = \sum_{j \in N_i} a_{ij}(\dot{x}_j - \dot{x}_i) + g_i(-\dot{x}_i). \tag{6.16}$$

The alias of ideal positions displaced by Δ_i is introduced

$$x_i^a = x_i - \Delta_i, \tag{6.17}$$

for $i = 1, 2, \ldots, N$. From (6.17), position errors between the two agents are given as

$$\delta_{ij} = x_j^a - x_i^a. \tag{6.18}$$

From (6.18), the neighbourhood error defined in (6.14) now has the form

$$e_i = \sum_{j \in N_i} a_{ij}(x_j^a - x_i^a) + g_i(x_0 - x_i^a), \tag{6.19}$$

$$\dot{e}_i = \sum_{j \in N_i} a_{ij}(\dot{x}_j - \dot{x}_i) + g_i(-\dot{x}_i). \tag{6.20}$$

Define $\underline{1}_N = [1, 1, \ldots, 1]^T \mathbb{R}^N$ with all N elements ones. The global forms of (6.19) and (6.20) are now expressed as

$$e = -((\mathcal{L} + \mathcal{G}) \otimes I_n)(x^a - \underline{1}_N \otimes x_0), \tag{6.21}$$

$$\dot{e} = -((\mathcal{L} + \mathcal{G}) \otimes I_n)\dot{x}. \tag{6.22}$$

In order to further develop the discussion on that hypothesis, the second order error dynamics is first calculated

$$\ddot{e} = -((\mathcal{L} + \mathcal{G}) \otimes I_n)\ddot{x}. \tag{6.23}$$

Plugging (6.12) in (6.23) gives

$$\begin{aligned} \ddot{e} &= -((\mathcal{L} + \mathcal{G}) \otimes I_n)[(I_N \otimes A)x + (I_N \otimes B)v] \\ &= -((\mathcal{L} + \mathcal{G}) \otimes I_n)(A_g x + B_g v), \end{aligned} \tag{6.24}$$

where $A_g = I_N \otimes A$, and $B_g = I_N \otimes B$. Without the loss of generality, further simplification is introduced

$$\ddot{e} = -((\mathcal{L} + \mathcal{G}) \otimes I_n)(A_g x + u) \tag{6.25}$$

where $u = B_g v$ is defined as the global vector of inputs.

To drive both position and velocity to zero, we use the sliding mode control and define the sliding mode error as

$$r = \dot{e} + \Lambda e, \tag{6.26}$$

where Λ is positive definite. From [95], e is bounded as long as r is bounded. Let $\lambda = diag(\lambda_i)$ be N dimensional diagonal matrix with λ_i on the diagonal, each corresponding to one agent. Then $\Lambda = \lambda \otimes I_n$. Taking the time derivative of (6.26) yields

$$\begin{aligned}\dot{r} = \ddot{e} + \Lambda\dot{e} = &- ((\mathcal{L} + \mathcal{G}) \otimes I_n)(A_g x + u) \\ &- \Lambda((\mathcal{L} + \mathcal{G}) \otimes I_n)\dot{x}. \end{aligned} \tag{6.27}$$

By using Kronecker rule $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ (under the assumption that dimensions of A, B, C, D allow multiplications AC and BD), we introduce the following for convenience

$$\begin{aligned} Z =&((\mathcal{L} + \mathcal{G}) \otimes I_n) \\ Z^A =&((\mathcal{L} + \mathcal{G}) \otimes I_n)A_g = ((\mathcal{L} + \mathcal{G}) \otimes A) \\ Z_\lambda =&(\lambda \otimes I_n)((\mathcal{L} + \mathcal{G}) \otimes I_n) = ((\lambda(\mathcal{L} + \mathcal{G})) \otimes I_n) \end{aligned} \tag{6.28}$$

Rewrite (6.27) as

$$\dot{r} = -Z^A x - Z_\lambda \dot{x} - Zu. \tag{6.29}$$

Based on the undirected graph topology, we make some assumptions useful for the control design. $\mathcal{L}$ is irreducibly diagonally dominant matrix. $\mathcal{G}$ has at least one diagonal entry. $\mathcal{G}$ has at least one diagonal entry. It is not restrictive to make Assumptions 1 and 2, since many practical multi-agent systems fall under that category.

We introduce the control law locally and globally as

$$u_i = u_{1i} + K_i r_i \tag{6.30}$$

$$u = u_1 + (I_N \otimes K_i)r = u_1 + Kr \tag{6.31}$$

where matrix $K_i = K_j, \forall i, j$ is used for control design. Assume that u_1 cancels first two terms

in (6.29). u_1 is specified later in the theorem. Under that assumption re-write (6.29) in the global form

$$\dot{r} = -((\mathcal{L} + \mathcal{G}) \otimes K_i)r = -ZKr \tag{6.32}$$

Let Assumptions 1 and 2 hold. If we define

$$W = diag(w_i) = diag(1/q_i), \tag{6.33}$$

where

$$q = (\mathcal{L} + \mathcal{G})^{-1}\underline{1}_N, \tag{6.34}$$

then, both W and $Q = W(\mathcal{L}+\mathcal{G}) + (\mathcal{L}+G)^T W$ are positive definite. Additionally, if Q is positive definite then $Q \otimes I_n$ is also positive definite. $Q = W(\mathcal{L} + \mathcal{G}) + (\mathcal{L} + G)^T W$

Proof. The first assertion follows the results in [65, 83]. For the second assertion, we obtain that eigenvalues of a matrix generated by Kronecker product are cross-products of all possible combinations of eigenvalues from matrix Q and I_n. Both Q and I_n are positive definite and corresponding eigenvalues all positive. This completes the proof. □

Let $\bar{x}$ be the measurement of ideal state vector x

$$\begin{aligned} \bar{x} &= x + \mu_1 \\ \dot{\bar{x}} &= \dot{x} + \mu_2 \\ \bar{x}_0 &= x_0 + \mu_0, \end{aligned} \tag{6.35}$$

then define the bound on sensor noise

$$\begin{aligned} ||\mu_1|| &< \mu_{1,B} \\ ||\mu_2|| &< \mu_{2,B} \\ ||\mu_0|| &< \mu_{0,B}, \end{aligned} \tag{6.36}$$

where $\mu_{0,B}$ denotes the measurement error of leader's position.

Sliding mode error injected with noises is

$$\bar{r}_i = \dot{\bar{e}}_i + \lambda_i \bar{e}_i, \tag{6.37}$$

where

$$\bar{e}_i = \sum_{j \in N_i} a_{ij}(\bar{x}_j - \Delta_j - \bar{x}_i + \Delta_i) + g_i(\bar{x}_0 - \bar{x}_i + \Delta_i), \tag{6.38}$$

$$\dot{\bar{e}}_i = \sum_{j \in N_i} a_{ij}(\dot{\bar{x}}_j - \dot{\bar{x}}_i) + g_i(-\dot{\bar{x}}_i). \tag{6.39}$$

The global form of noisy error is

$$\bar{r} = \dot{\bar{e}} + \Lambda \bar{e}, \tag{6.40}$$

where

$$\begin{aligned}
\bar{e} =& -((\mathcal{L} + \mathcal{G}) \otimes I_n)(\bar{x}^a - \underline{1}_N \otimes \bar{x}_0) \\
=& -((\mathcal{L} + \mathcal{G}) \otimes I_n)(x^a + \mu_1 - \underline{1}_N \otimes x_0 - \underline{1}_N \otimes \mu_0) \\
=& -((\mathcal{L} + \mathcal{G}) \otimes I_n)(\mu_1 - \underline{1}_N \otimes \mu_0) + e, \\
=& -Z(\mu_1 - \underline{1}_N \otimes \mu_0) + e, \qquad (6.41) \\
\dot{\bar{e}} =& -Z\mu_2 + \dot{e}, \qquad (6.42) \\
\ddot{\bar{e}} =& -(A_g \bar{x} + u) \\
=& -ZA_g\mu_1 + \ddot{e}. \qquad (6.43)
\end{aligned}$$

Extracting the ideal sliding mode error from the measured error gives

$$\bar{r} = r - \Lambda Z(\mu_1 - \underline{1}_N \otimes \mu_0) - Z\mu_2. \tag{6.44}$$

Substituting (6.43) and (6.42) into (6.27) yields

$$\dot{\bar{r}} = \dot{r} - ZA_g\mu_1 - \Lambda Z\mu_2. \tag{6.45}$$

Definition 15. *The signal $z(t)$ is said to be uniformly ultimately bounded (UUB) with the ultimate bound b, if given positive constants b and c, for every $d \in (0, c)$, there exists $T(d, b)$, independent of t_0, such that*

$$\|z(t_0)\| \leq d \Rightarrow \|z(t)\| \leq b, \quad \forall t \geq t_0 + T. \tag{6.46}$$

Here, we propose our distributed consensus control for UAVs in the following theorem.

Theorem 6. *Let Assumptions 1–3 hold. Define the sliding mode error dynamics as (6.27). Select the control policy for the local agent as*

$$u_i = K_i\bar{r}_i - A\bar{x}_i - \lambda_i\dot{\bar{x}}_i, \tag{6.47}$$

Assume that $\lambda_i = \lambda_j, \forall i, j$, (6.47). Consider the error dynamics (6.32) and design matrices $\mathcal{Q}$ and $\mathcal{R}$. Pick following control gain

$$K_i = \mathcal{R}^{-1}\mathcal{P} \tag{6.48}$$

$$K = (I_N \otimes K_i) \tag{6.49}$$

where $\mathcal{P}$ is the unique positive definite solution of control algebraic Riccati equation

$$0 = \mathcal{Q} - \mathcal{P}\mathcal{R}^{-1}\mathcal{P}. \tag{6.50}$$

The control law (6.47) with the gain (6.48) guarantees asymptotic stability for (6.32). Moreover it

stabilizes the system (6.12) and makes the ideal tracking error (6.14) UUB.

Proof. Let eigenvalues of $(\mathcal{L}+\mathcal{G})$ be $\nu_i = \alpha_i + j\beta_i$. Matrix $\mathcal{L}+\mathcal{G}$ has $\alpha_i > 0$ because of Assumption 1 and 2. System (6.32) is asymptotically stable if matrices $-\nu_i K_i$ are asymptotically stable (see [65]). Since $\mathcal{P} > 0$, $\mathcal{R} > 0$ and $\alpha_i > 0$ it follows that matrix $-\nu_i K_i$ is Hurwitz and (6.32) is asymptotically stable. From Lyapunov theory, the stability condition for (6.32) is equivalent to

$$(-ZK)^T P + P(-ZK) = Q \tag{6.51}$$

where $Q < 0$ and $P > 0$, $P^T = P$, and K from (6.49). (6.51) is equivalent to

$$PZK = -\frac{1}{2}\mathcal{Q}. \tag{6.52}$$

To prove stability in the presence of noisy measurements, consider following Lyapunov function candidate for multi-agent systems in global form

$$V = \frac{1}{2} r^T P r. \tag{6.53}$$

Differentiating (6.53) with respect to (6.27) yields

$$\begin{aligned}
\dot{V} =& r^T P \dot{r} \\
=& r^T P[((\mathcal{L}+\mathcal{G}) \otimes I_n)(-\Lambda \dot{x} - A_g x - u)] \\
=& r^T P[((\mathcal{L}+\mathcal{G}) \otimes I_n)(-\Lambda \dot{x} - A_g x - K\bar{r} \\
& + A_g \bar{x} + \Lambda \dot{\bar{x}}] \\
=& r^T P \Lambda((\mathcal{L}+\mathcal{G}) \otimes I_n)\mu_2 + \\
& + r^T P((\mathcal{L}+\mathcal{G}) \otimes I_n) A_g \mu_1 \\
& - r^T P((\mathcal{L}+\mathcal{G}) \otimes I_n) K \bar{r}.
\end{aligned} \tag{6.54}$$

For brevity, we use (6.28). Combined with (6.44) gives

$$\begin{aligned}\dot{V} =& r^T P Z_\lambda \mu_2 \qquad (6.55)\\ &+ r^T P Z^A \mu_1 \\ &- r^T P Z K [r - Z_\lambda(\mu_1 - \underline{1}_N \otimes \mu_0) - Z\mu_2] \\ =& - r^T P Z K r + r^T P (Z_\lambda + Z K Z)\mu_2 \\ &+ r^T P (Z^A + Z K Z_\lambda)\mu_1 \\ &- r^T P Z K Z_\lambda (\underline{1}_N \otimes \mu_0). \qquad (6.56)\end{aligned}$$

Note that the upper bound on $\dot{V}$ can be found through the norm algebra and the bounds that are previously assumed to limit the measurement noise vectors. To be specific, from (6.55), we have

$$\begin{aligned}\dot{V} \leq & - min\{||PZK|| \cdot ||r||^2\} \\ &+ max\{||r|| \cdot ||P(Z^A + ZKZ_\lambda)|| \cdot ||\mu_1||\} \\ &+ max\{||r|| \cdot ||P(Z_\lambda + ZKZ)|| \cdot ||\mu_2||\} \\ &+ max\{||r|| \cdot ||PZKZ_\lambda|| \cdot ||\underline{1}_N \otimes \mu_0||\}. \qquad (6.57)\end{aligned}$$

based on which (6.57) is changed into

$$\begin{aligned}\dot{V} \leq & - \sigma_{min}\{PZK\} \cdot ||r||^2 \\ &+ ||r|| \cdot \sigma_{max}\{P(Z^A + ZKZ_\lambda)\} \cdot \mu_{1,B} \\ &+ ||r|| \cdot \sigma_{max}\{P(Z_\lambda + ZKZ)\} \cdot \mu_{2,B} \\ &+ ||r|| \cdot \sigma_{max}\{PZKZ_\lambda\} \cdot \mu_{1,0,B}. \qquad (6.58)\end{aligned}$$

Where $\sigma_{max}(M)$ and $\sigma_{min}(M)$ denote maximum and minimum eigenvalue of matrix M. From

(6.58), it is clear that $\dot{V}$ is always negative for

$$\begin{aligned} ||r|| > & \frac{\sigma_{max}\{P(Z^A + ZKZ_\lambda)\}}{\sigma_{min}\{PZK\}}\mu_{1,B} \\ & + \frac{\sigma_{max}\{P(Z_\lambda + ZKZ)\}}{\sigma_{min}\{PZK\}}\mu_{2,B} \\ & + \frac{\sigma_{max}\{PZKZ_\lambda\}}{\sigma_{min}\{PZK\}}\mu_{1,0,B}. \end{aligned} \tag{6.59}$$

Moreover, if r is outside the bounding set given in (6.59), our controller will eventually drag the error inside the set. In other words, the sliding mode error r is bounded to a set which makes all the signals in the closed-loop multi-agent system UUB. □

The controller drags all aliases x_i^a for $i = 1, \ldots, N$ to the center of the formation. If all aliases are in the center of the formation, no action from the controller is required. Neighbourhood errors quantify the spatial disharmony of the formation. The controller determines an action u_i of the agent i such that it recovers consensus. Λ is a tuning parameter. As given, controller resembles well known standard PD controller.

6.5 Modifying Edge Weights Based on Trust

In this section, it is shown how UAVs in the graph can distributively modify the graph weights based on their measure of trustworthiness of the neighbors. The purpose is to maintain the consensus by eliminating negative influence of stubborn agents or extremely noisy measurement imposed on the local agent.

The trust that agent i has for agent j can be calculated as deviation from the intended average state of neighboring agents

$$\tau_{ij} = \frac{\theta_{ij}}{\|\bar{x}_j^a - \frac{1}{N_i+1}(\bar{x}_i^a + \sum_{j\in N_i}\bar{x}_j^a)\| + \theta_{ij}}. \tag{6.60}$$

where θ_{ij} is a certain positive constant that serves as a threshold value. From (6.60), when the local agent j approaches the average of the neighbourhood agents, the trust value τ_{ij} approaches one;

otherwise, τ_{ij} approaches zero. With this trust metric, we further propose a mechanism to modify the graph weights. To this end, we use the differential equation

$$\dot{d}_{ij} = -\eta d_{ij} + \eta \tau_{ij}, \tag{6.61}$$

where η is a certain positive constant. To take into account the trust value calculated for neighbors, the controller (6.47) is now revisited by weighting all edge weights a_{ij} by d_{ij}. That is all occurrences of a_{ij} in (6.47) are replaced by $d_{ij}a_{ij}$.

6.6 Experimental Environment

Figure 6.1 shows an entire system designed at UTA Research Institute for testing distributed controllers on UAVs. Basic four elements are VICON, Crazyflie 2.0 UAVs, and the workstation that runs master program. Master is implemented as ROS package and it handles consensus controller calculation as well as local manipulation over UAVs.

As already mentioned, Crazyflie 2.0 is used as UAV and it communicates with master workstation using radio transceiver. In order to have reliable feedback information, we use motion capture system VICON. Data is transmitted to master through wireless local area network. Such communication architecture enables consensus controller to send the control commands to UAVs at 200 Hz, which is more than sufficient.

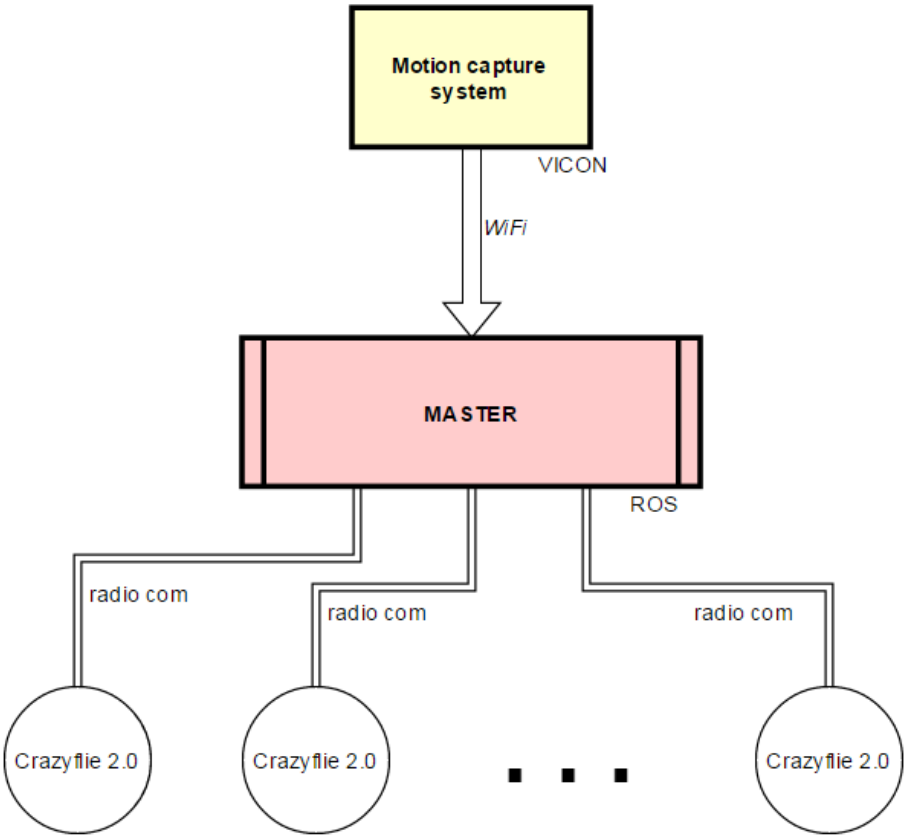

Figure 6.1: Motion capture and communication topology based on master-slave model architecture. *robustness constraint* introduced in the chapter.

6.6.1 Crazyflie 2.0

Crazyflie 2.0 is a commercially available UAV suitable for research and development. It is created by Bitcraze AB. We use their hardware and modify the firmware to implement our control laws. IMU data collection, four motors with corresponding PWM modules and communication with radio (nRF51822) is all handled by micro-controller (STM32F405). Attitude controller is implemented on firmware side rather than on master/server. It uses inputs from outer control loop (commanded velocities in all three dimensions and commanded yaw rate) and IMU data to stabilize UAV and to drive it to the desired position. Desired values are received from nRF51822, previously sent from the master. The output of attitude controller are signals that determine PWM cycle duty and eventually produce propeller thrusts and torques.

6.6.2 Master workstation

Figure 6.2 shows the software architecture that is executed on the master workstation. The master controller is a ROS node that manages general control of all UAVs. Each UAV is controlled by individual outer loop controller.

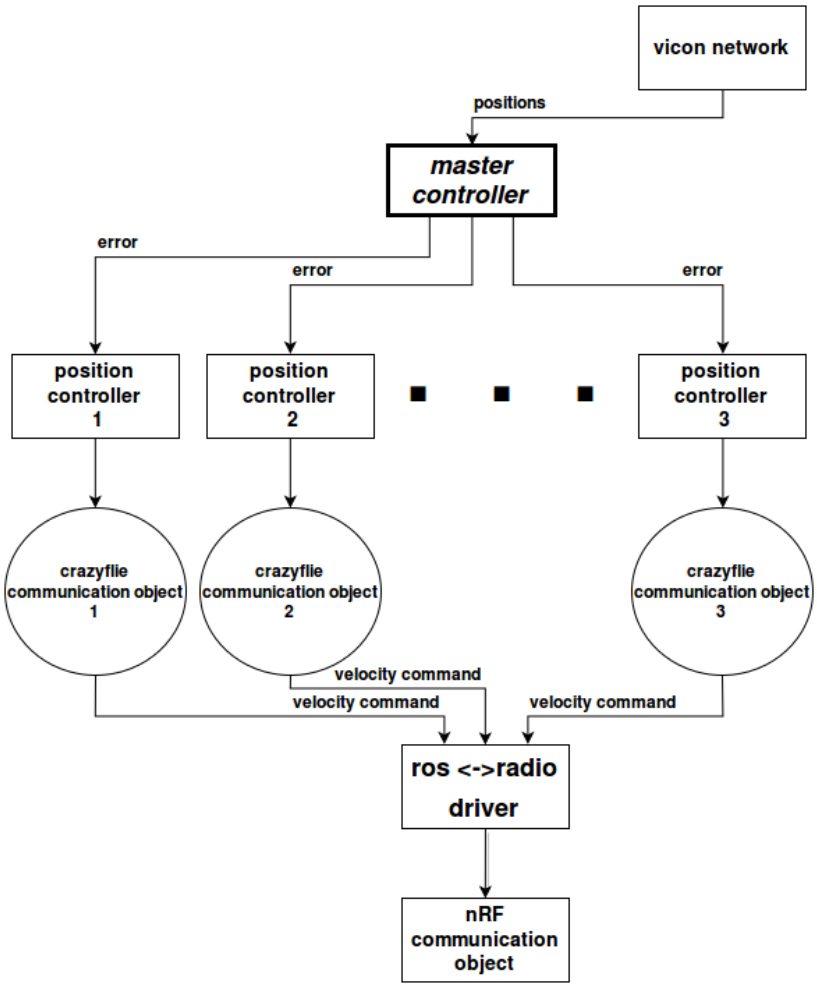

Figure 6.2: Distributed multi-threaded software architecture with data pipeline.

The master controller is, in fact, a mission planner. It commands different tasks to UAVs by communicating with corresponding position controller. It also listens to external user input to allow manual task switching, manual take off, manual landing, and manual emergency landing. State machine seems to be a plausible design solution for master controller, considering its current function of switching states. As Figure 6.3 clearly shows, there are three different states in state machine. In waypoint tracking state, master controller simply sets waypoints to position con-

trollers. Consensus controller is slightly more complex, as generalized formation error for each UAV has to be recalculated at every iteration. The idle state is active when all Crazyflies are off. It is important to notice that user has the control over whole state machine which is a security requirement.

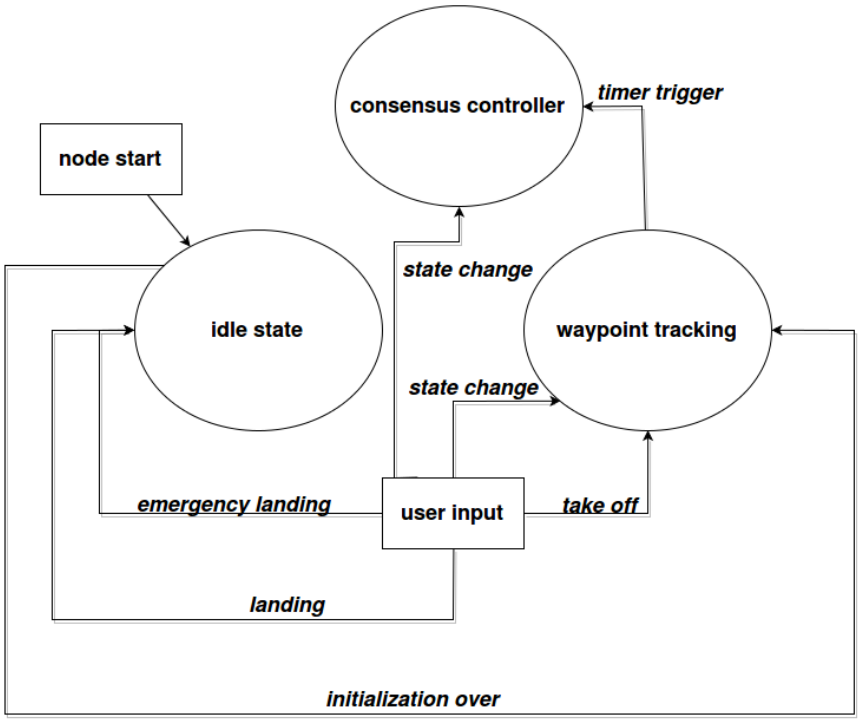

Figure 6.3: Distributed hierarchical motion control statemachine.

The described procedure can be broken down into following main steps

Check the link quality. If link quality is low proceed to Step 7. Otherwise continue to Step 2. Collect data from VICON, distribute information to agents according to adjacency matrix. Update trust values and apply them to modify adjacency matrix. Call local distributed controllers. Send calculated roll, pitch, yaw and thrust commands to each agent. Calculate PWM locally on each agent and generate forces. If no user interrupt, go back to Step 1. Otherwise go to Step 7. Start landing sequence. Land.

Algorithm 6.1 *Software solution pseudocode*

VICON link quality < requirements Switch state to landing Reiterate master loop Collect data from VICON Distribute position data to linked agents Update trust values Call local controllers CF link quality < requirements Send desired R,P,Y,T to UAVs Request landing Calculate PWM based on R,P,Y,T Land and turn off

6.7 Flight Tests

Three flight tests are carried out in this section, where different adjacency matrices are picked. For each experiment, we show two plots, x position and y position for all three UAVs. The desired reference is set $(0, 0, 0.7)$ and disturbance is applied to one of three UAVs to test the behaviour of the group. To be specific, we push the UAV away from the point of global equilibrium imposed by consensus controller. Displacements from the desired reference are $\Delta_1 = (-0.5, 0.5, 0.0)$, $\Delta_2 = (0.5, 0.5, 0.0)$, and $\Delta_3 = (-0.5, -0.5, 0.0)$. Note that in the experiment, we define the virtual input u as

$$u = (I_N \otimes B)v = B_g v. \tag{6.62}$$

Technical control vector v can now be calculated from virtual inputs u through

$$u = (I_N \otimes B)v \tag{6.63}$$

$$v_i = B^T(BB^T)^{-1}u_i \tag{6.64}$$

In fact, the ability to calculate v is not only a matter of convenience, but rather a necessity, since v is the command vector that is transmitted to UAV agents. u_i is a distributed control signal (6.47). In first 5 seconds, altitude control is applied to take off UAVs, and after that our consensus controller is activated.

6.7.1 Experiment 1

In this case, we define the adjacency matrix as

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix} \tag{6.65}$$

and the pinning gains as $g_2 = 3$ and $g_1 = g_3 = 0$. After applying our controller, the experimental results are presented in Figures 4 and 5, where UAVs 1, 2, and 3 are labelled in red, blue, and green. Here, UAV 1 is the agent that is injected with disturbances. In this experiment, UAV 1 moves away from the consensus due to disturbances, while UAVs 2 and 3 try to follow. From (6.65), UAV 3 strictly follows movements of agent 1 in order to keep consensus, since it has direct link to agent 1. Note that UAV 2 is less responsive to position change of agent 1 due to the fact that agent 2 is connected to the leader.

Also notice that UAV 1 is first sent way out of desired position at around 9_{th} second. At the very peak, when UAV 1 is more than 0.5 m away from desired position, UAV 3 does not follow UAV 1 to the extreme deviation because it has lost trust in UAV 1 (a_{13} was close to 0). The fact that trajectories of UAVs 1 and 3 have slightly different shapes at that point confirms that (Figure 5). As UAV 1 returns and distance between UAVs 1 and 3 is back as expected, trust increases and consensus is re-established.

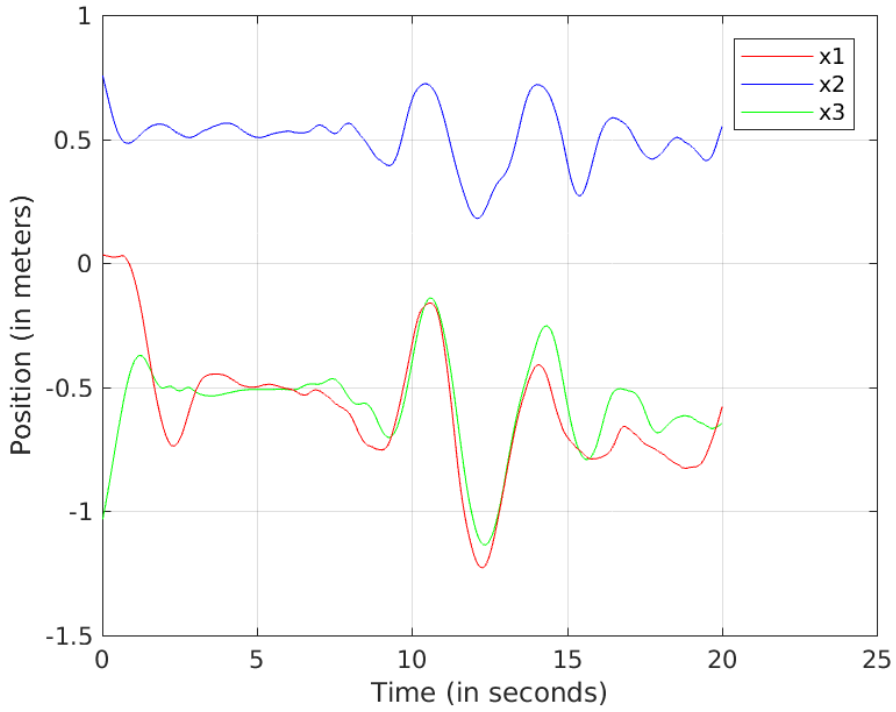

Figure 6.4: Experiment 1, positions x of UAVs

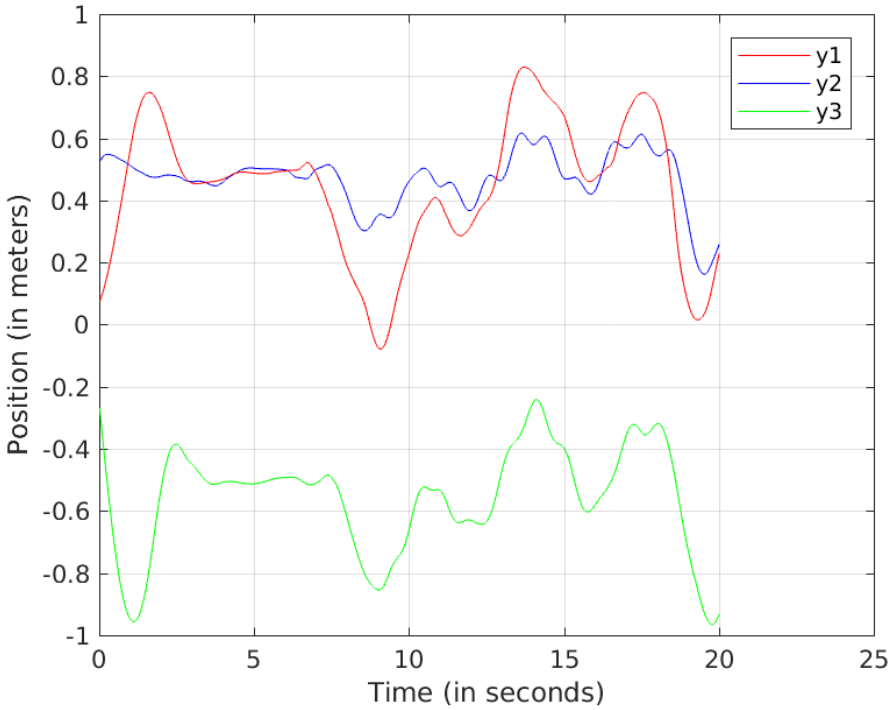

Figure 6.5: Experiment 1, positions y of UAVs

6.7.2 Experiment 2

Here, we define the adjacency matrix as

$$\mathcal{A} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix} \tag{6.66}$$

and the pinning gains as $g_2 = 3$ and $g_1 = g_3 = 0$. Adding a direct link between agents 1 and 2 makes UAV 2 a bit more responsive. Formation control in Figures 6 and 7 is better compared to that in the previous experiment.

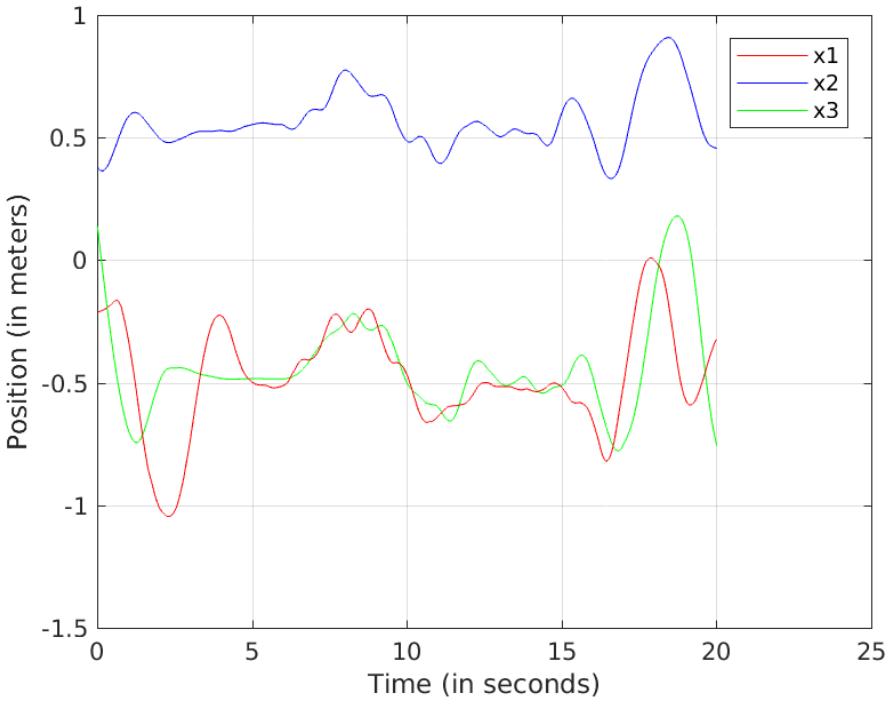

Figure 6.6: Experiment 2, positions x of UAVs

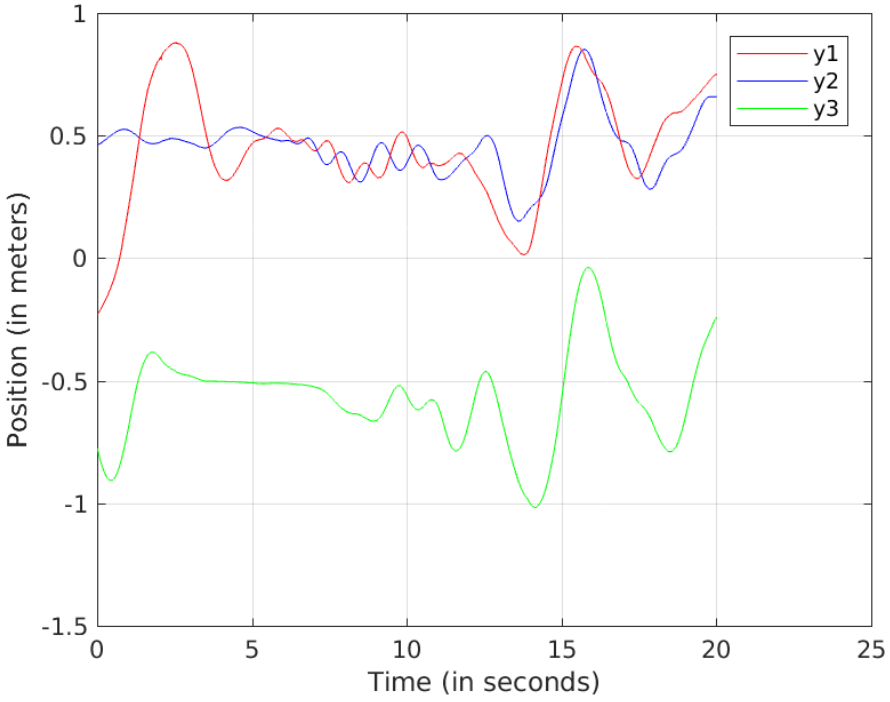

Figure 6.7: Experiment 2, positions y of UAVs

6.7.3 Experiment 3

In the final experiment, the adjacency matrix is designed as

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{6.67}$$

and pinning gains are given as $g_1 = g_2 = g_3 = 1$. Compared to the previous experiments, (6.67) gives the best performance, as shown in Figures 8 and 9. Such results are expected because the graph is dense and weights are not giving preference to any neighbouring agent.

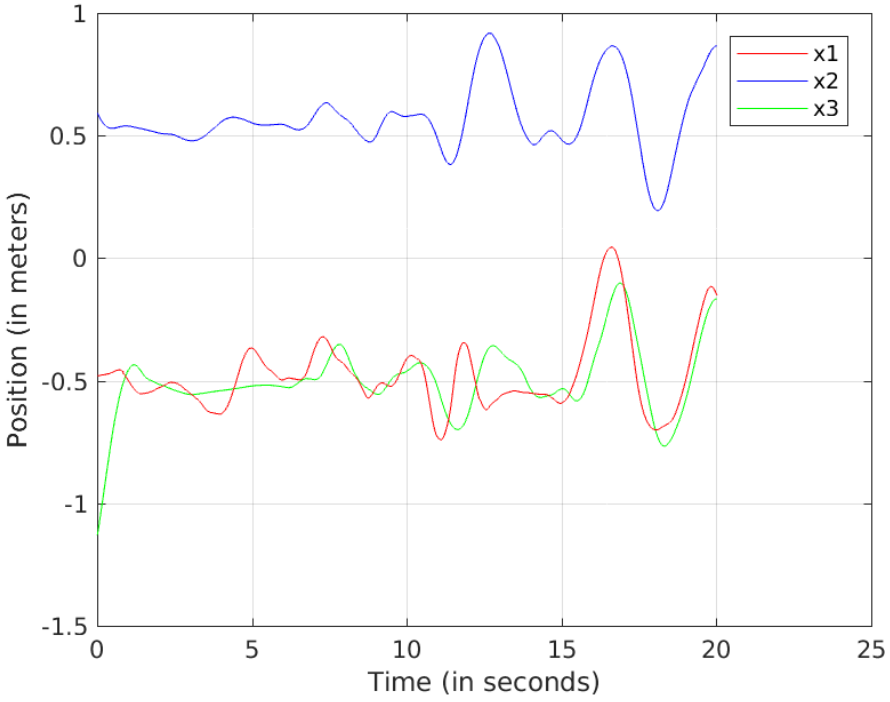

Figure 6.8: Experiment 3, positions x of UAVs

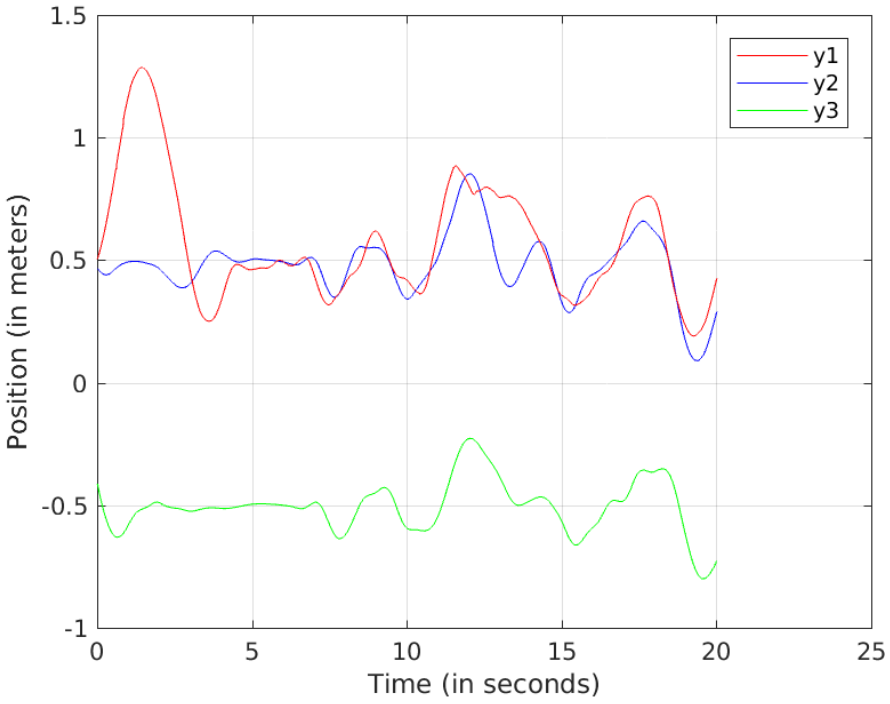

Figure 6.9: Experiment 3, positions y of UAVs

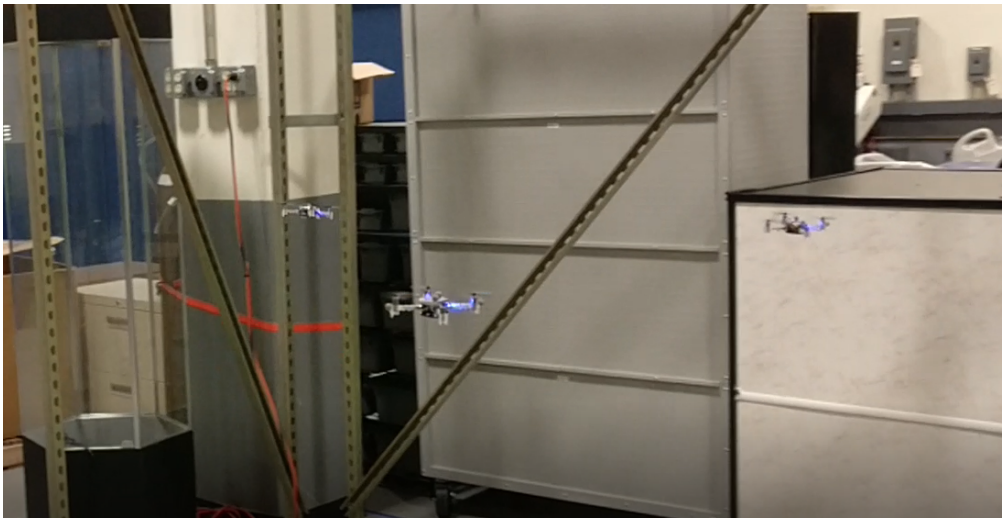

Figure 6.10: Photograph of the experiment during consensus

6.8 Conclusion

Consensus control design introduced in this chapter is effective in keeping a group of UAVs in the formation. Confirmation is given by using Lyapunov theory to prove the stability of the group and later in the experimental environment on the system of multi UAVs.

It is shown that the robustness of consensus depends highly on the graph topology. In that aspect, fully connected graphs will provide better basis for firm consensus control. Unfortunately, it is often the case that practical multi-agent network topology is inherent to the system to which control is applied. In other words, while designing controller, graph is predefined and cannot be changed in favor of better control. However, one thing we can do is to update the values of the links that are present based on the trustworthiness of a neighbouring agent. Our approach achieves that.

Due to certain practical assumptions about dynamics of the nano UAV and by restricting experiments to a slow flight, we were able to make some reasonable simplifications on the dynamical model of the UAV. That allowed for more transparent design of the consensus controller in the light of measurement noises imposed on the system. Future work should extend current design to the domain of non-linear control, allowing for dynamically rich experiments and treat the measurement

noise with advanced tools such as distributed Kalman filter.

Chapter 7: CONCLUSION AND FUTURE DIRECTIONS

The fields of optimal control, reinforcement learning, inverse optimal control and inverse reinforcement learning all deal with different partial aspects of rational objectives and behavior of dynamic agents. None of these different fields deal with the complete picture. Moreover, the relations among these fields are not fully understood since they come from different disciplines including feedback control, computer science, machine learning, game theory among others. In this dissertation, we have worked on 4 important problems in optimal decision making.

First contribution of this dissertation is to provide a unified theoretical framework that joins optimal control, reinforcement learning, inverse RL and inverse OC and shows how they are related. Moreover, rigorous mathematical proofs of convergence, stability and optimality of model-based and model-free Inverse RL algorithms are provided.

All optimal decision making frameworks require that the cost function is given upfront. It is hard to choose a cost function when the performance criteria are subjective and human dependent (e.g. comfort). We have developed iterative algorithm for learning the objectives that an expert agent is optimizing. With our algorithm, the cost of underlying LQR can be attuned to human preferences.

Second important contribution is the development of dynamic intelligent control allocation mechanism that can be used for actuator allocation in redundant robotic systems. This contribution is important as the redundancy in aerial vehicles as well as ground robots is an emerging necessity out of safety reasons. We show that data-driven reinforcement learning algorithm can be used to accomplish this task.

In our third contribution, we developed a mechanism for including uncertainty information from learned model of system dynamics into existing trajectory optimization, state-of-the-art methods. We proposed a robust method for capturing the uncertainty information that is initially estimated by machine learning models such as Gaussian Process.

Finally, we have studied decision-making in multi-agent scenarios and developed a distributed

controller that can navigate a flock of UAVs in a decentralized fashion.

We envision a few research directions that can further improve on our work.

1) Inverse Reinforcement Learning in combination with Optimal Control is a promising framework for capturing human intentions and objectives based on data. In general, all optimal controllers (LQR, MPC, Hinfinity, iLQR, trajectory optimization,...) have associated cost. All these controllers appear in standard robotic systems such as self-driving car or robot manipulator. Optimal controllers are extremely good at accomplishing optimal behaviour if the right cost function is provided. The method developed in this dissertation can be used for data-driven tuning of the cost function. Experimental studies are needed to test the scalability of this method on various real life applications.

2) Optimal decision making (RL) and the inference of objectives and intents (Inverse RL) can be studied in the multi-player setting. The behavior of agents in a network of agents can be best explained by studying the objectives of these agents.

3) Control allocation is another topic that can be further extended. In our work, we take advantage of linear relationship to identify the null-space of allocation matrix as the space that contains all redundant solution. An important extension of this work is to consider nonlinear actuators.

4) Uncertainty informed robust decision making can be improved by considering the hybrid of our solution and sampling based solution. This work can also be extended by considering more powerful machine-learning models such as multi-modal distributions that can model different modes of behaviours.

BIBLIOGRAPHY

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

[2] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. *International Conference on Machine Learning*, 2004.

[3] E. Arabi, T. Yucelen, and W. M. Haddad. Mitigating the effects of sensor uncertainties in networked multiagent systems. In *2016 American Control Conference (ACC)*, pages 5545–5550, 2016.

[4] John Baras and Tao Jiang. *Cooperation, Trust and Games in Wireless Networks*, pages 183–202. 01 2005.

[5] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft*. Princeton University Press, 2012.

[6] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[7] Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas*. Princeton University Press, second edition, 2011.

[8] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. 01 1995.

[9] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[10] K. Bordignon. Constrained control allocation for systems with redundant control effectors. 1996.

[11] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[12] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[13] T. BÃ€chle, K. Graichen, M. Buchholz, and K. Dietmayer. Model predictive control allocation in electric vehicle drive trains. *IFAC-PapersOnLine*, 48(15):335–340, 2015. 4th IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling E-COSM 2015.

[14] A. Chakrabarty, D. K. Jha, and Y. Wang. Data-driven control policies for partially known systems via kernelized lipschitz learning. In *2019 American Control Conference (ACC)*, pages 4192–4197, July 2019.

[15] A. Chakrabarty, R. Quirynen, C. Danielson, and W. Gao. Approximate dynamic programming for linear systems with state and input constraints. In *2019 18th European Control Conference (ECC)*, pages 524–529, June 2019.

[16] Ankush Chakrabarty, Devesh K Jha, Gregery T Buzzard, Yebin Wang, and Kyriakos Vamvoudakis. Safe approximate dynamic programming via kernelized lipschitz estimation. *arXiv preprint arXiv:1907.02151*, 2019.

[17] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

[18] Seungwon Choi and Suseong Kim. Inverse reinforcement learning control for trajectory tracking of a multirotor uav. *International Journal of Control, Automation and Systems*, 15, 07 2017.

[19] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 1989.

[20] Abhijit Das and Frank Lewis. Cooperative adaptive control for synchronization of second-order systems with unknown nonlinearities. *International Journal of Robust and Nonlinear Control*, 21:1509 – 1524, 09 2011.

[21] Abhijit Das, Frank Lewis, and Kamesh Subbarao. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, 56(1-2):127–151, April 2009.

[22] Abhijit Das, Kamesh Subbarao, and F. Lewis. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *Control Theory Applications, IET*, 3:303 – 314, 04 2009.

[23] Gianmaria De Tommasi, Sergio Galeani, Alfredo Pironti, Gianluca Varano, and Luca Zaccarian. Brief paper: Nonlinear dynamic allocator for optimal input/output performance trade-off: Application to the jet tokamak shape controller. *Automatica*, 47(5):981â987, May 2011.

[24] Mao de Yan, Xu Zhu, Xun xun Zhang, and Yao hong Qu. Consensus-based three-dimensionalmulti-UAV formation control strategy with high precision. *Frontiers of Information Technology & Electronic Engineering*, 18(7):968–977, July 2017.

[25] Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[26] Haitham El-Hussieny, Ahmed Ali, Samy Assal, and Said Megahed. Adaptive learning of human motor behaviors: An evolving inverse optimal control approach. *Engineering Applications of Artificial Intelligence*, 50:115–124, 02 2016.

[27] C. M. Elliott, G. Tallant, and A. Dogan. On probability collectives for distributed control allocation. In *2017 IEEE Aerospace Conference*, pages 1–11, 2017.

[28] S. Elvira-Ceja and E. N. Sanchez. Discrete-time inverse optimal control for stochastic nonlinear systems trajectory tracking. In *52nd IEEE Conference on Decision and Control*, pages 2483–2487, 2013.

[29] F. Facchinei and J.-S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems, Volume II*. Springer, New York, NY, 2003.

[30] T. Fernando, S. Denman, S. Sridharan, and C. Fookes. Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion. *IEEE Signal Processing Magazine*, 38(1):87–96, 2021.

[31] Sergio Galeani, Andrea Serrani, Gianluca Varano, and Luca Zaccarian. On input allocation-based regulation for linear over-actuated systems. *Automatica*, 52(C):346â354, February 2015.

[32] Hongbo Gao, Guanya Shi, Guotao Xie, and Bo Cheng. Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making. *International Journal of Advanced Robotic Systems*, 15(6):1729881418817162, 2018.

[33] Wassim Haddad and Vijaysekhar Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton university press, 2008.

[34] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. *Advances in Neural Information Processing Systems*, pages 3909–3917.

[35] Weiqiao Han and Russ Tedrake. Local trajectory stabilization for dexterous manipulation via piecewise affine approximations. *arXiv e-prints*, 2019. `https://arxiv.org/abs/1909.08045`.

[36] O. Harkegard. Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 2, pages 1295–1300 vol.2, 2002.

[37] Ola Harkegard. Dynamic control allocation using constrained quadratic programming. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2002.

[38] Ola Harkegard and S. Torkel Glad. Resolving actuator redundancy-optimal control vs. control allocation. *Automatica*, 41(1):137â144, January 2005.

[39] G.A Hewer. Analysis of a discrete matrix riccati equation of linear control and kalman filtering. *Journal of Mathematical Analysis and Applications*, 42(1):226 – 236, 1973.

[40] N. J. Higham. *Functions of matrices: theory and computation*. Society for Industrial and Applied Mathematics, 2008.

[41] Ola HÃ€rkegÃ¥rd and S. Torkel Glad. Flight control design using backstepping. *IFAC Proceedings Volumes*, 34(6):283–288, 2001. 5th IFAC Symposium on Nonlinear Control Systems 2001, St Petersburg, Russia, 4-6 July 2001.

[42] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[43] David H Jacobson. New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach. *Journal of Optimization Theory and Applications*, 2(6):411–440, 1968.

[44] D. K. Jha, D. Nikovski, W. Yerazunis, and A. Farahmand. Learning to regulate rolling ball motion. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6, Nov 2017.

[45] Devesh Jha, Patrik Kolaric, Diego Romeres, Arvind Raghunathan, Mouhacine Benosman, and Daniel N Nikovski. Robust optimization for trajectory-centric model-based reinforcement learning. In *Workshop on Safety and Robustness in Decision Making at NeurIPS 2019*. NeurIPS, 2019.

[46] Sang-Won Ji, Van Phuoc Bui, B. Balachandran, and Young-Bok Kim. Robust control allocation design for marine vessel. *Ocean Engineering*, 63:105–111, 2013.

[47] T. Jiang and J. S. Baras. Trust evaluation in anarchy: A case study on autonomous networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12, 2006.

[48] Yu Jiang and Zhong-Ping Jiang. Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica*, 48(10):2699–2704, 2012.

[49] Tor A. Johansen and Thor I. Fossen. Control allocationâa survey. *Automatica*, 49(5):1087–1103, 2013.

[50] R. E. Kalman. When Is a Linear Control System Optimal? *Journal of Basic Engineering*, 86(1):51–60, 03 1964.

[51] S. Khoo, L. Xie, and Z. Man. Robust finite-time consensus tracking algorithm for multirobot systems. *IEEE/ASME Transactions on Mechatronics*, 14(2):219–228, 2009.

[52] D. L. Kleinman. On the iterative technique for riccati equation computations. *IEEE Transactions on Automatic Control*, 13:114â115, 1968.

[53] Y. Kuriki and T. Namerikawa. Formation control of uavs with a fourth-order flight dynamics. In *52nd IEEE Conference on Decision and Control*, pages 6706–6711, 2013.

[54] Y. Kuriki and T. Namerikawa. Consensus-based cooperative formation control with collision avoidance for a multi-uav system. In *2014 American Control Conference*, pages 2077–2082, 2014.

[55] P. Lancaster and L. Rodman. *Algebraic Riccati Equations*. Clarendon Press, 1995.

[56] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[57] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[58] F. L. Lewis and M. Abu-Khalaf. A hamilton-jacobi setup for constrained neural network control. In *Proceedings of the 2003 IEEE International Symposium on Intelligent Control*, pages 1–15, 2003.

[59] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[60] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[61] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[62] F W Lewis, S. Jagannathan, and A Yesildirak. *Neural Network Control Of Robot Manipulators And Non-Linear Systems*. CRC Press, August 2020.

[63] Frank Lewis, Draguna Vrabie, and Vassilis Syrmos. *Optimal Control*. Wiley, 2012.

[64] Frank L. Lewis, Draguna L. Vrabie, and Vassilis L. Syrmos. *Optimal Control*. John Wiley & Sons, Inc., January 2012.

[65] Frank L. Lewis, Hongwei Zhang, Kristian Hengster-Movric, and Abhijit Das. *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. Springer Publishing Company, Incorporated, 2014.

[66] X. Lin, P. A. Beling, and R. Cogill. Multiagent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Games*, 10(1):56–68, 2018.

[67] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.

[68] S. Martinez. Uav cooperative decision and control: Challenges and practical approaches (shima, t. and rasmussen, s.; 2008) [bookshelf]. *IEEE Control Systems Magazine*, 30(2):104–107, 2010.

[69] H. Modares, F. L. Lewis, and Z. Jiang. H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2550–2562, 2015.

[70] Timothy L. Molloy, Jason J. Ford, and Tristan Perez. Finite-horizon inverse optimal control for discrete-time nonlinear systems. *Automatica*, 87:442 – 446, 2018.

[71] William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pages 4008–4016, 2016.

[72] Richard M. Murray. Recent Research in Cooperative Control of Multivehicle Systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 05 2007.

[73] Mehdi Naderi, Ali Khaki Sedigh, and Tor Arne Johansen. Guaranteed feasible control allocation using model predictive control. *Control Theory and Technology*, 17(3):252–264, July 2019.

[74] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[75] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth*

International Conference on Machine Learning, ICML â99, page 278â287. Morgan Kaufmann Publishers Inc., 1999.

[76] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[77] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[78] R. W. Obermayer and F. A. Muckler. On the inverse optimal control problem in manual control systems. *National Aeronautics and Space Administration*, 1965.

[79] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[80] M. W. Oppenheimer, D. B. Doman, and M. A. Bolender. Control allocation for over-actuated systems. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–6, 2006.

[81] Yonmook Park. Inverse optimal and robust nonlinear attitude control of rigid spacecraft. *Aerospace Science and Technology*, 28(1):257 – 265, 2013.

[82] J. A. M. Petersen and M. Bodson. Constrained quadratic programming techniques for control allocation. *IEEE Transactions on Control Systems Technology*, 14(1):91–98, 2006.

[83] Z. Qu, J. Wang, and R. A. Hull. Cooperative control of dynamical systems with application to autonomous vehicles. *IEEE Transactions on Automatic Control*, 53(4):894–911, 2008.

[84] Zhihua Qu. Cooperative control of dynamical systems: Applications to autonomous vehicles. *Cooperative Control of Dynamical Systems: Applications to Autonomous Vehicles*, 01 2009.

[85] Guilherme V. Raffo, Manuel G. Ortega, and Francisco R. Rubio. An integral predictive/nonlinear hâ control structure for a quadrotor helicopter. *Automatica*, 46(1):29–39, 2010.

[86] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[87] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

[88] Diego Romeres, Devesh K Jha, Alberto DallaLibera, Bill Yerazunis, and Daniel Nikovski. Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3195–3202. IEEE, 2019.

[89] Ornelas-Tellez F. Sanchez, E.N. *Discrete-Time Inverse Optimal Control for Nonlinear Systems*. CRC Press, 2013.

[90] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233 – 242, 1999.

[91] A. Serrani. Output regulation for over-actuated linear systems via inverse model allocation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 4871–4876, 2012.

[92] Zhifei Shao and Meng Joo Er. A survey of inverse reinforcement learning techniques. *Int. J. Intelligent Computing and Cybernetics*, 5:293–311, 2012.

[93] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[94] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[95] J.J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall International Editions. Prentice-Hall, 1991.

[96] Brian L. Stevens, Frank L. Lewis, and Eric N. Johnson. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, Inc, November 2015.

[97] H. J. Sussmann and J. C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, 1997.

[98] Richard Sutton and Andrew Barto. *Reinforcement learning: An Introduction, 2nd edition*. MIT press.

[99] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction (2nd Edition)*, volume 1. MIT press Cambridge, 2018.

[100] C. Tan, Y. Li, and Y. Cheng. An inverse reinforcement learning algorithm for semi-markov decision processes. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6, 2017.

[101] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[102] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[103] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *journal of machine learning research*, 11(Nov):3137–3181, 2010.

[104] Johan Thunberg, Wenjun Song, Yiguang Hong, and Xiaoming Hu. Distributed attitude synchronization using backstepping and sliding mode control. *Control Theory and Technology*, 12(1):48–55, January 2014.

[105] J. Tjonnas and T. A. Johansen. Optimizing adaptive control allocation with actuator dynamics. In *2007 46th IEEE Conference on Decision and Control*, pages 3780–3785, 2007.

[106] Johannes Tjonnas and Tor A. Johansen. Adaptive control allocation. *Automatica*, 44(11):2754â2765, November 2008.

[107] S.S. Tohidi, Y. Yildiz, and I. Kolmanovsky. Adaptive control allocation for over-actuated systems with actuator saturation **author yildiray yildiz would like to thank the scientific and technological research council of turkey (tubitak) for its financial support through the 2232 reintegration scholarship program. *IFAC-PapersOnLine*, 50(1):5492–5497, 2017. 20th IFAC World Congress.

[108] K. G. Vamvoudakis and F. L. Lewis. Online solution of nonlinear two-player zero-sum games using synchronous policy iteration. In *49th IEEE Conference on Decision and Control (CDC)*, pages 3040–3047, Dec 2010.

[109] K.G. Vamvoudakis, P.J. Antsaklis, W.E. Dixon, J.P. Hespanha, F.L. Lewis, H. Modares, and B. Kiumarsi. Autonomy and machine intelligence in complex systems: A tutorial. In *American Control Conference (ACC), 2015*, pages 5062–5079, July 2015.

[110] Kyriakos G. Vamvoudakis and Frank L. Lewis. Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamiltonâjacobi equations. *Automatica*, 47(8):1556–1569, 2011.

[111] D. Vrabie and F. Lewis. Adaptive dynamic programming algorithm for finding online the equilibrium solution of the two-player zero-sum differential game. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[112] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F.L. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2):477 – 484, 2009.

[113] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis. *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. IET control engineering series. Institution of Engineering and Technology, 2013.

[114] Jing Wang, Xiaohong Nian, and Hai-bo Wang. Consensus and formation control of discrete-time multi-agent systems. *Journal of Central South University of Technology (English Edition)*, 18:1161–1168, 08 2011.

[115] 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)*, pages 498–503, 2016.

[116] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking Model-Based Reinforcement Learning. *arXiv e-prints*, page arXiv:1907.02057, Jul 2019.

[117] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[118] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[119] Xiang Xu, Lu Liu, and Gang Feng. Consensus of single integrator multi-agent systems with directed topology and communication delays. *Control Theory and Technology*, 14(1):21–27, February 2016.

[120] Yu Luo, A. Serrani, S. Yurkovich, D. B. Doman, and M. W. Oppenheimer. Model predictive dynamic control allocation with actuator dynamics. In *Proceedings of the 2004 American Control Conference*, volume 2, pages 1695–1700 vol.2, 2004.

[121] T. Yucelen, W. M. Haddad, and E. M. Feron. Adaptive control architectures for mitigating sensor attacks in cyber-physical systems. In *2016 American Control Conference (ACC)*, pages 1165–1170, 2016.

[122] Luca Zaccarian. Dynamic allocation for input redundant control systems. *Automatica*, 45(6):1431–1438, 2009.

[123] H. Zhang, T. Feng, G. Yang, and H. Liang. Distributed cooperative optimal control for multiagent systems on directed graphs: An inverse optimal approach. *IEEE Transactions on Cybernetics*, 45(7):1315–1326, 2015.

[124] H. Zhang, F. L. Lewis, and A. Das. Optimal design for synchronization of cooperative systems: State feedback, observer and output feedback. *IEEE Transactions on Automatic Control*, 56(8):1948–1952, Aug 2011.

[125] Han Zhang, Jack Umenberger, and Xiaoming Hu. Inverse optimal control for discrete-time finite-horizon linear quadratic regulators. *Automatica*, 110:108593, 2019.

[126] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. pages 1433–1438, 2008.

VITA

Patrik received B.S. in 2013 and M.S. in 2015 from University of Zagreb, both in electrical engineering. He achieved Ph.D. degree at University of Texas, Arlington in 2021. During the time of Ph.D.He worked at UTA Research Institute as a researcher and manager of Autonomous System Lab. He was awarded Enhanced GTA scholarship at UTA. His research insterest include data-driven intelligent control, reinforcement learning, machine learning, robotics and distributed systems.