TRAJECTORY PLANNING UNDER MOTION AND SENSING UNCERTAINTIES: REACHABILITY ANALYSIS AND CONNECTIVITY MAINTENANCE

BY

AKSHAY SHETTY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Assistant Professor Grace Gao, Director of Research
Associate Professor Timothy Bretl, Chair
Professor Jonathan Makela
Research Assistant Professor Huy Tran

ABSTRACT

Recently there has been growing interest in robotic systems with several promising applications such as transportation, delivery of goods, surveillance and cinematography. Additionally, multi-robot systems are being increasingly considered for applications such as exploration, target tracking and formation control. A vital component of these robotic systems is planning trajectories that are collision-safe. Furthermore, for multi-robot systems it is highly desirable to plan trajectories that maintain communication connectivity within the system, thus enabling coordination between robots. For practical robots, trajectory planning is challenging due to the presence of uncertainties in robot motion and sensor measurements. These uncertainties result in the robot deviating from the planned trajectory and can consequently lead to collisions or loss of communication connectivity within multi-robot systems. Thus, it is important to explicitly account for motion and sensing uncertainties while designing trajectory planning algorithms.

Reachability analysis is a popular verification-based tool where reachable sets for the robot are first computed along candidate trajectories and then used to plan collision-safe trajectories. However, previous works do not explicitly account for robot sensing uncertainties in their formulation. While there exist algorithms for trajectory planning under sensing uncertainties, these works model the uncertainties as known Gaussian distributions, which is not always valid. For instance, Global Navigation Satellite System (GNSS) pseudorange measurements may contain additional biases in urban environments due to non-line-of-sight signals or multipath effects. These biases in sensor measurements lead to further deviations from the planned trajectory and thus must be accounted for during planning. On the other hand, for multi-robot systems, the topic of connectivity maintenance has been explored in literature. However, previous works assume simplified robot motion models and do not account for motion and sensing uncertainties in their formulation.

The contribution of this dissertation is to develop trajectory planning algorithms that mitigate the aforementioned limitations in previous works. For planning collision-safe trajectories, we first develop a reachability analysis to predict possible robot deviations under motion and sensing uncertainties. We model the sensing uncertainties as a Gaussian distribution along with an additional bias. Next, we integrate the reachability analysis with an existing trajectory planning framework to plan collision-safe trajectories. Finally, we statistically validate via simulations that the reachability analysis captures the possible robot deviations. The applicability of the trajectory planner is then demonstrated for collision-safe GNSS-based navigation of fixed-wing Unmanned Aerial Vehicles (UAVs). For connectivity maintenance of multi-robot systems, we develop a distributed Alternating Direction Method of Multipliers (ADMM) based trajectory planner that explicitly accounts for motion and sensing uncertainties. We simulate a multi-UAV system and statistically validate that our planner maintains connectivity within the system for multiple scenarios.

To my parents, for their unconditional love and support.

ACKNOWLEDGMENTS

I would first like to thank my advisor, Prof. Grace Xingin Gao for being a continuous source of motivation and guidance throughout my graduate school experience. She has been incredibly patient with me during my academic ups and downs and I feel fortunate to have worked with her.

I am grateful to have interacted with some brilliant people at Urbana-Champaign and at Stanford: Craig Babiarz, Sriramya Bhamidipati, Shubhendra Chauhan, Derek Chen, Daniel Chou, Arthur Chu, Adam Dai, Shubh Gupta, Ashwin Kanhere, Cara Kataria, Derek Knowles, Shreyas Kousik, Athindran Ramesh Kumar, Enyu Luo, Tara Mina, Adyasha Mohanty, Yuting Ng, Matthew Peretic, Andy Lai, Pulkit Rustagi and Victor Zhang. They have generously devoted their time to help me with my research. I am also thankful to Alfredo Bencomo, Sebastian Hening and Kalmanje Krishnakumar from whom I learned a lot during my summer internships at NASA Ames.

I would also like to thank my soccer friends at Urbana-Champaign. I was fortunate to be part of such a lively and diverse group and I enjoyed our time on and off the field. Benjamin and Kensuke, it was a pleasure to be your roommate at Sheswett.

Beatriz Maldonado, I am very fortunate to have had you beside me throughout this journey. You have loved and supported me in countless ways. Your kindness, resilience and curiosity inspires me and I look forward to many more adventures with you.

Finally, I would like to thank my sister Pooja Shetty and my parents Sudha and Prabhakar Shetty. Without your unconditional love, tenacity and numerous sacrifices over the years, I would not have the opportunities I have today. You make me want to be stronger everyday. I dedicate this dissertation to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADMM	Alternating Direction of Multipliers
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
GCM	Global Connectivity Maintenance
iLQG	Iterative Linear Quadratic Gaussian
KF	Kalman Filter
LCM	Local Connectivity Maintenance
LQR	Linear Quadratic Regulator
NLOS	Non-line-of-sight
UAV	Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

Recently there has been growing interest in robotic systems such as Unmanned Aerial Vehicles (UAVs) and autonomous ground vehicles. Several promising applications have emerged in recent years including transportation [1, 2, 3], delivery of goods [4, 5, 6, 7], aerial or ground surveillance [8, 9, 10] and cinematography [11]. The Federal Aviation Administration (FAA) projects that the UAV fleet within the United States will grow to 1.5 million vehicles by 2024 [12], with corresponding traffic management rules being explored [13]. Meanwhile, autonomous ground vehicles are currently being tested in controlled environments for transportation [14] and delivery of goods [7]. Additionally, multi-robot systems are being increasingly considered [15] due to their ability to coordinate (through inter-robot communication) and execute complex missions in a safe [16, 17] and efficient manner. Example applications for multi-robot systems include exploring unknown areas [18, 19, 20], target tracking [21, 22], formation control [23, 24, 25], and cooperative manipulation [26]. Fig. 1.1 shows a few example applications of these robotic systems.

A vital component for safe deployment of the aforementioned robotic systems is trajectory planning. Trajectory planning for a robot typically involves planning its motion over a time horizon while satisfying certain requirements such as maintaining collision-safety. Furthermore, for multi-robot systems it is highly desirable to plan trajectories such that communication connectivity is maintained within the system.

1.1 Trajectory Planning Under Uncertainty

The presence of uncertainties is an inherent characteristic of practical robotic systems [28]. These uncertainties are typically categorized as follows [29]:

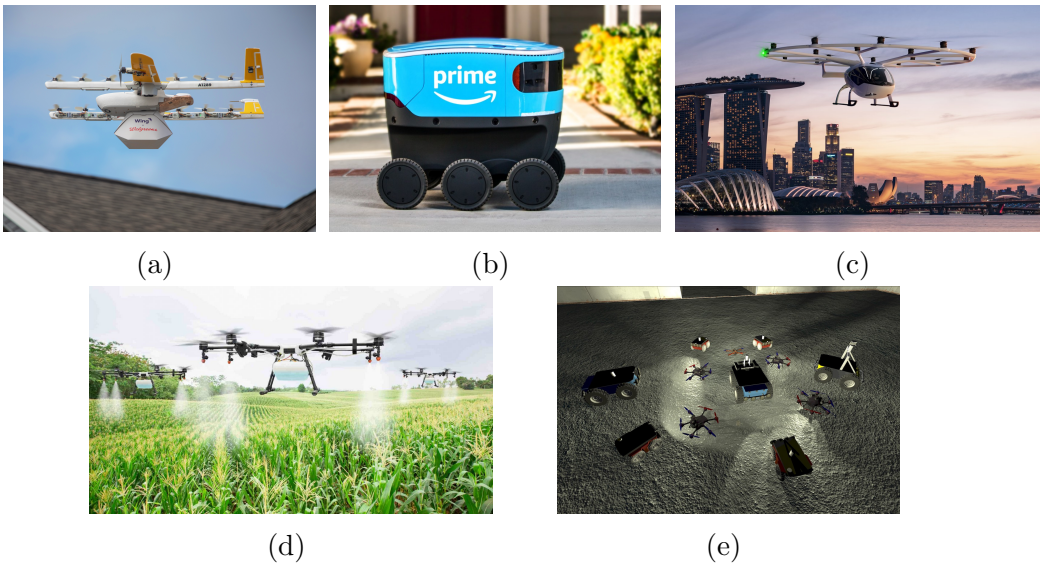
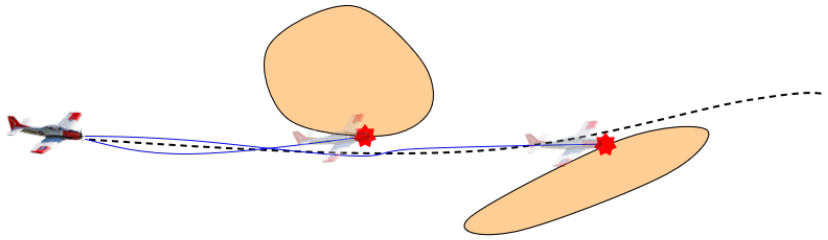


Figure 1.1: Example applications of robotic systems for (a,b) delivery of goods [6, 7], (c) transportation [3], and multi-robot systems for (d) agricultural farming [27] and (e) exploration of unknown areas [20].

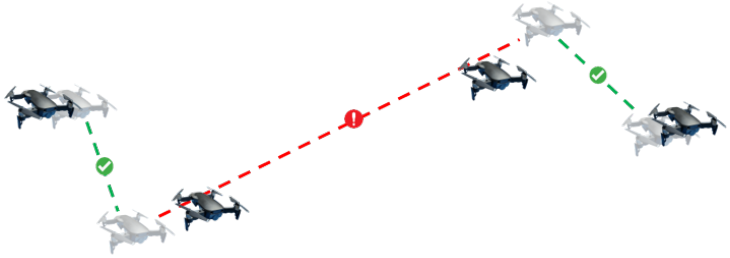
uncertainty in knowledge of the robot motion, uncertainty in the robot's sensor measurements, uncertainty in knowledge of the surroundings such as obstacle locations and uncertainty about the operational environment such as unknown wind disturbances. Here motion uncertainties refer to the errors between the actual robot motion and a mathematical motion model, whereas sensing uncertainties refer to errors in sensor measurements such as errors in localization measurements. These motion and sensing uncertainties result in the robot deviating from the planned trajectory which can consequently lead to undesirable outcomes such as collisions or loss of communication connectivity within multi-robot systems as illustrated in Fig. 1.2. Thus, it is important to explicitly account for motion and sensing uncertainties while designing trajectory planning algorithms.

1.2 Related Work

In this section, we discuss related works on trajectory planning under uncertainty that use reachability analysis and tree-based planners. Additionally, we discuss how relevant state estimation algorithms account for sensing uncertainties. Finally, previous works on connectivity maintenance of multi-robot systems is discussed.



(a)



(b)

Figure 1.2: Presence of motion and sensing uncertainties lead to robots deviating from their planned trajectories, potentially resulting in collisions or loss of communication connectivity. (a) While the planned trajectory (black dashed line) for the UAV is collision-free, deviations (blue line, faded UAVs) lead to collisions (red star) with obstacles (orange). (b) For a system with 4 UAVs, these deviations (faded UAVs) from the planned positions lead to the inter-robot distance becoming larger than the communication range and consequently resulting in loss of communication connectivity (red dashed line) within the system.

1.2.1 Reachability Analysis

Reachability analysis is a powerful formal verification-based tool that is commonly used to provide collision-safety guarantees for robotic systems. The fundamental idea is to compute sets of states (robot positions and orientations) that can be reached when a robot follows a trajectory. These reachable sets capture possible deviations of the robot due to various sources of uncertainty. During trajectory planning these reachable sets are computed along candidate trajectories and then used to evaluate their collision-safety.

Initial works on reachability analysis [30, 31, 32, 33] computed the reachable sets along a single trajectory for linear systems. Extensions to non-linear systems [34, 35, 36, 37, 38] were explored in two primary directions. In [35, 36] the authors linearized a non-linear system, and computed the reachable sets for the linearized system while obtaining conservative bounds

for the linearization errors (referred to as Lagrange remainders). An alternate approach was taken in [37, 38], where the authors used polynomial set representations and computed the reachable sets directly for a non-linear system. However, such set representations come with an additional computational cost [39] which is not desirable for purposes such as trajectory planning.

Different flavors of reachability analysis have been presented for trajectory planning where the reachable sets are computed along a family of trajectories. In [39, 40, 41] the authors use a reachability toolbox [42] to compute the reachable sets and then select trajectories where the reachable sets do not intersect with obstacles. The works in [43, 44, 45] follow a similar approach, where [43] uses a polynomial set representation and [44, 45] use level sets for their reachability analysis. These works do not explicitly account for sensing uncertainties and typically assume that the robot has access to its true state while it follows the planned trajectory. However, sensing uncertainties affect the robot's state estimation and cause the robot to deviate further from the planned trajectory. Thus it is important to account for sensing uncertainties while computing reachable sets used for planning collision-safe trajectories.

1.2.2 Trajectory Planning Under Motion and Sensing Uncertainties

Recent works [46, 47, 48, 49, 50, 51] build upon traditional tree-based planning algorithms [52, 53] to address the presence of motion and sensing uncertainties. While these recent works check for collision-safety by predicting possible robot deviations along candidate trajectories, the authors assume the sensing uncertainties to be represented by known Gaussian distributions. In [46, 47] the authors predict the state (position and orientation) uncertainty for a robot with a linear-quadratic-Gaussian controller, i.e., a combination of a Kalman filter and a linear-quadratic-regulator. The work in [48] predicts a distribution over the robot states along candidate trajectories by considering the distribution in the state estimation error in addition to all possible state estimates that could be realized along the trajectory. In [49, 50] the authors predict the state uncertainty for monocular camera-based navigation. [49] assumes known Gaussian distributions for localization measurements associated with visual features in a stored map, whereas [50] also assumes a known

Gaussian distribution for the localization uncertainty which is obtained using photometric information available in advance. However, the assumption that sensing uncertainties along candidate trajectories can be represented in advance by known Gaussian distributions is not always valid. For instance, for outdoor state estimation robotic systems generally use Global Navigation Satellite System (GNSS) measurements. GNSS pseudorange measurements typically contain an additional bias in urban environments due to signal reflections from nearby buildings [54]. These effects are classified either as multipath, where both the direct and reflected signals from the same satellite are received; or as non-line-of-sight (NLOS), where only the reflected satellite signal is received [55]. Fig. 1.3 illustrates GNSS multipath and NLOS effects. Generally, NLOS effects result in large biases in pseudorange measurements. Various outlier rejection techniques and 3-dimensional (3D) map-based techniques have been proposed to detect and exclude the corresponding measurements [56, 57, 58, 59, 60]. On the other hand, biases due to multipath effects are relatively smaller and more challenging to detect. Previous works [55, 61] have proposed methods to calculate the bounds for these multipath biases using a 3D map and the GNSS receiver architecture. Thus, it is important for a trajectory planner to account for these additional bounded biases in the sensor measurements while finding collision-safe trajectories for the robot.

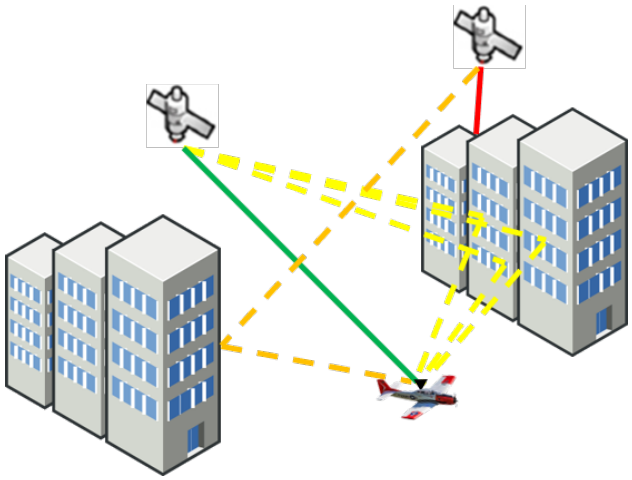


Figure 1.3: GNSS pseudorange measurements contain an additional bias in urban environments due to multipath and NLOS effects. Multipath occurs when both the direct (green) and reflected (yellow) signals from the same satellite are received, whereas NLOS occurs when the direct satellite signal is blocked (red) and only the reflected (orange) signal is received.

The problem of an additional bias in measurements also exists for other commonly used sensors, apart from GNSS receivers. In [62] the authors develop a method to detect biases due to data association errors in visual localization measurements, whereas [63] derive closed-form solutions to determine biases in measurements from an inertial measurement unit. Thus, while we focus on the sensing uncertainties in GNSS measurements, the algorithms developed in this dissertation are also applicable for other sensor configurations.

1.2.3 Trajectory Planning for GNSS-based Navigation

Trajectory planning for GNSS-based navigation in urban environments has been previously explored. In [61] the authors propose an A* planning algorithm to route UAVs through areas with fewer GNSS positioning errors. The work in [64] also uses a similar A*-based planner while accounting for additional sources of GNSS errors such as the receiver thermal noise and reflections from the UAV airframe. In [65] the authors first generate a reliability map for GNSS and cellular signals, and then use a Djikstra planner to find the shortest trajectory that minimizes positioning error and guarantees that the state estimation uncertainty is below a desired threshold. [66] defines GNSS dilution-of-precision layers in the environment and uses them to design trajectories with given positioning accuracy requirements for multiple UAVs. The work in [67] uses particle swarm optimization to plan the shortest trajectory that avoids areas with low GNSS satellite visibility and with a high dilution-of-precision. While the planned trajectories in these works are collision-free, the authors do not check for the collision-safety due to the possible deviations arising from motion and sensing uncertainties.

1.2.4 State Estimation in Presence of Measurement Biases

The presence of an additional bounded bias along with a stochastic component in the sensor measurements has previously been addressed in the field of state estimation. Generally state estimation algorithms assume one of two distinct measurement error models: a stochastic error model or a bounded error model. Typically for stochastic error models the measurement errors are

assumed to have a Gaussian distribution and the system state is estimated using Bayesian filters such as the Kalman filter [28]. On the other hand, set-membership techniques such as the zonotopic Kalman filter [68] have been proposed for bounded error models. In [69, 70] the authors merge the two error models by accounting for the presence of both a bounded bias along with a stochastic component in the sensor measurements. While such a merged approach accounts for additional bounded biases in sensor measurements, these works [69, 70] focus on designing optimal state estimators instead of predicting possible robot deviations and trajectory planning, which is of our primary interest.

1.2.5 Connectivity Maintenance for Multi-robot Systems

The general approach of previous connectivity maintenance algorithms is to synthesize control inputs for each robot in the system such that either local connectivity or global connectivity of the system is maintained [71]. Local Connectivity Maintenance (LCM) methods focus on preserving the initial topology of connections within the multi-robot system [72, 73, 74, 75, 76, 77]. Thus, if two robots are initially connected, the synthesized control inputs for these robots maintain their connection throughout the mission. LCM methods typically consist of relatively simple computations since the control inputs for each robot depend only on local information of the robots to which it is connected. However, the freedom of motion for each robot is restricted since initial connections between robots are not allowed to be broken. Global Connectivity Maintenance (GCM) methods on the other hand allow individual connections to break as long as there exists a (potentially multi-hop) communication path between any two robots in the system [78, 79, 80, 81, 82, 83, 84, 85, 86, 87]. Thus, each robot is afforded a greater freedom of motion in comparison to LCM methods.

A limitation of the previous connectivity maintenance works is that they do not explicitly account for robot motion and sensing uncertainties in their theoretical formulation. As mentioned in Section 1.1, these uncertainties lead to robots deviating from the planned trajectories which directly affects the connectivity within the system. Thus, it is important to explicitly account for robot motion and sensing uncertainties while designing connectivity

maintenance methods.

Additionally, the majority of the previous connectivity maintenance works use a simplified single integrator model to represent the robot motion [79, 80, 81, 82, 83, 84, 85, 86]. This motion model assumes that the robot can instantaneously change its direction of motion and move towards a desired position for connectivity maintenance. Thus, these works derive control inputs in a myopic fashion, i.e., only for the current time instant. However, for most practical robots, such as UAVs, the direction of motion is not instantaneously changeable. The trajectory that the robot follows typically depends on additional quantities (such as previous velocities [88, 89]) and hence the robot might not be able to move towards the desired position instantaneously. Thus, for connectivity maintenance, it is important to derive control inputs in a non-myopic fashion by considering the trajectory of the robot over multiple future time instants.

Local Connectivity Maintenance (LCM)

LCM methods for multi-robot systems are advantageous due to their decentralized nature and their computational simplicity. In [72], the authors introduce a localized notion of connectivity and show that under certain conditions, the global connectivity of the system is also guaranteed. The work in [73] designs a control strategy for the formation control of multi-robot systems while maintaining the initial topology of connections between the robots. In [74], the authors use a potential field-based method for LCM and later extend it to account for robots with bounded control inputs in [75]. The authors in [76] include a repulsive potential field to additionally account for collisions while maintaining local connectivity. [77] builds on the LCM method in [75] to account for critical (close to breaking) initial connections and the presence of leader robots in the system. However, as mentioned above, these works result in less freedom of motion for robots compared to GCM methods.

Global Connectivity Maintenance (GCM)

GCM is widely addressed in literature. Previous methods typically represent the multi-robot system as a weighted undirected graph and use the alge-

braic connectivity of this graph as an indicator of the system connectivity. The algebraic connectivity is defined as the second smallest eigenvalue of the graph Laplacian matrix, as discussed later in Section 2.2. In [80], the authors present a decentralized power iteration algorithm for each robot to estimate the algebraic connectivity of the system. This estimate is then used to design a decentralized gradient-based controller for GCM. [83] and [81] build on the method in [80] by defining a decentralized estimation procedure for algebraic connectivity that is formally guaranteed to be stable. Given the estimation error boundedness, they prove that the proposed control law guarantees GCM if the control parameters are chosen appropriately. Further, in [82] the authors extend their previous GCM works of [83] and [81] by taking into account the presence of an additional (bounded) control input for each robot. [84] extends on [83] and [81] by explicitly accounting for additional inter-robot constraints such as a desired relative distance and collision avoidance. In [85], the authors design a GCM method to account for robots with bounded control inputs. They present a theoretical analysis to evaluate the robustness of the proposed controller to bounded errors in estimate of the system algebraic connectivity. However, the estimation error bound is heuristically obtained without explicitly accounting the for sources of uncertainty such as robot motion and sensing uncertainties.

Another common approach for GCM is to design optimization-based methods without estimating the value of the algebraic connectivity itself. In [78], the authors find optimal positions for vertices of a graph that maximize the algebraic connectivity. [79] follows a similar approach and derives control inputs for a multi-robot system using a decentralized potential field-based method. In [87], the authors develop a differential game-theoretic formulation for maximizing the algebraic connectivity in the presence of a malicious jammer. [86] uses control barrier functions to integrate a GCM requirement with an additional control input for each robot.

While many of these works develop decentralized methods with GCM guarantees, they do not explicitly account for robot motion and sensing uncertainties and a majority of them assume a simplified single integrator robot motion model. Thus, in their simulation/experimental setups they make simplifications; for instance, assuming perfect sensing information such as perfect localization measurements and/or using slow-moving robots that can be reasonably modeled as single integrator systems. However, practical robots are

typically represented by higher-fidelity motion models and use state estimation filters to estimate their positions under motion and sensing uncertainties [28].

1.3 Our Contributions

In this dissertation, we design trajectory planning algorithms for robotic systems that address the limitations in previous works discussed above in Section 1.2. A reachability analysis is developed to account for sensing uncertainties that contain a stochastic component along with an additional bounded bias. We integrate the reachability analysis with an existing planning framework to find collision-safe trajectories. For multi-robot systems, a trajectory planning algorithm is designed to maintain communication connectivity under the presence of motion and sensing uncertainties. The main contributions are summarized as follows:

1. **Reachability Analysis for Linear Systems Under Motion and Sensing Uncertainties**: In order to evaluate collision-safety along candidate trajectories during planning, it is important to predict possible robot deviations due to motion and sensing uncertainties. Previous works on trajectory planning either assume the sensing uncertainties to be known Gaussian distributions (Section 1.2.2) or do not account for sensing uncertainties altogether (Section 1.2.1).

   We develop a reachability analysis for robotic systems represented by linear motion and sensing models, while accounting for motion and sensing uncertainties. Here we model the sensing uncertainties to contain a stochastic component (modeled as a Gaussian distribution) along with an additional bounded bias. The process of mathematical induction is used to derive an expression to compute reachable sets for the robot along a single trajectory. Finally, we discuss simulation results for two linear robotic systems and statistically validate that the computed reachable sets capture the possible deviations due to the motion and sensing uncertainties.

2. **Reachability Analysis for Non-linear Systems Under Motion and Sensing Uncertainties**: Widely used practical robotic systems

such as fixed-wing UAVs are usually represented by non-linear motion models [89]. Additionally, common sensing models such as ranging measurements from beacons or GNSS satellites are also represented by non-linear sensing models. Thus, it is important to analyze the reachability of non-linear robotic systems. While previous non-linear reachability analysis works exist (Section 1.2.1), these do not account for robot sensing uncertainties.

We extend our reachability analysis to robotic systems represented by non-linear motion and sensing models. Similar to the linear reachability analysis, sensing uncertainties are modeled to contain a Gaussian stochastic component along with an additional bounded bias. An expression to compute reachable sets for the robot along a single trajectory is derived and a Gaussian approximation is proposed for the linearization errors (Lagrange remainders) arising in the reachability analysis. Finally, we present simulations for a non-linear robotic system and statistically validate the computed reachable sets.

3. **Collision-safe Trajectory Planning Under Motion and Sensing Uncertainties**: Given that our reachability analysis allows us to predict possible deviations due to motion and sensing uncertainties along a single trajectory, a planning framework is required to find collision-safe trajectories from an initial robot state to a goal state. Planning frameworks typically involve considering multiple candidate trajectories, and thus, it is important to evaluate the collision-safety of these candidate trajectories efficiently. Additionally, for computational tractability, planning frameworks compare candidate trajectories and eliminate undesirable ones.

   We integrate our reachability analysis with an existing planning framework [51] in order to plan collision-safe trajectories under motion and sensing uncertainties. A heuristic approximation of the reachability analysis is proposed which allows us to efficiently evaluate the collision-safety of candidate trajectories during the planning process. In order to compare candidate trajectories during planning, we design a metric for the size of the reachable sets. This metric helps us to identify and eliminate undesirable candidate trajectories. Finally, via simulations, we demonstrate the applicability of the trajectory planner for

GNSS-based navigation of fixed-wing UAVs in an urban environment. Results are discussed for a UAV in a static environment and in a shared airspace with other UAVs. The collision-safety of the UAVs along the planned trajectories is statistically validated.

4. **Trajectory Planning for Connectivity Maintenance of Multi-robot Systems Under Motion and Sensing Uncertainties**: Maintaining communication connectivity within a multi-robot system is highly desirable since it enables robots to coordinate and execute complex tasks safely and efficiently. For connectivity maintenance of practical systems it is important to plan robot trajectories in a non-myopic fashion (over multiple future time instants) while accounting for robot motion and sensing uncertainties.

   We design a trajectory planner for connectivity maintenance of a multi-robot system under motion and sensing uncertainties. We first define a weighted undirected graph to represent the connectivity of the system where we explicitly account for robot motion and sensing uncertainties while formulating the graph edge weights. Next, the algebraic connectivity of the weighted undirected graph is maintained above a specified lower limit using a non-myopic trajectory planner based on a distributed Alternating Direction Method of Multipliers (ADMM) framework. Here we derive an approximation for the Hessian matrices required within the ADMM optimization step to reduce the computational load. Finally, simulation results are discussed to statistically validate the connectivity maintenance of our trajectory planner.

1.4 Dissertation Outline

Chapter 2 presents the preliminaries required for the algorithms described in the following chapters. We describe the set representation used in our reachability analysis and discuss the corresponding set operations. Additionally, relevant background from the field of graph theory is provided which is used in our connectivity maintenance algorithm for multi-robot systems.

In Chapter 3 our reachability analysis for robotic systems represented by linear motion and sensing models is presented. Here the sensing uncertainty

is modeled to contain an additional bounded bias along with a stochastic component (modeled as a Gaussian distribution). We derive an expression to compute the reachable sets for the robot along a single trajectory and statistically validate the computed reachable sets via simulations for two linear robotic systems.

Chapter 4 extends our reachability analysis to robotic systems represented by non-linear motion and sensing models. We follow a similar process as in Chapter 3, to derive an expression to compute the reachable sets for the robot along a single trajectory. An approximation for the linearization errors within the reachability analysis is developed. We simulate a non-linear robotic system and statistically validate the computed reachable sets.

In Chapter 5 we develop the trajectory planning algorithm to plan collision-safe trajectories using our reachability analysis from Chapters 3 and 4. A heuristic approximation of our reachability analysis is provided to efficiently evaluate collision-safety of candidate trajectories during planning. Additionally, a metric for the size of the computed reachable sets is proposed allowing us to compare different candidate trajectories during the planning process. We discuss simulations demonstrating the applicability of the trajectory planner for GNSS-based navigation of fixed-wing UAVs in an urban environment.

Chapter 6 describes our trajectory planning algorithm for connectivity maintenance of a multi-robot system in the presence of motion and sensing uncertainties. We define the weighted undirected graph representing the system, where the graph accounts for uncertain robot positions. A distributed ADMM-based planner is designed that plans trajectories for the robots such that the connectivity is maintained. An approximation for the Hessian matrices required within the ADMM optimization step is derived to reduce the planner computational load. We discuss simulation results to statistically validate the connectivity maintenance of our trajectory planner under multiple scenarios. Finally, Chapter 7 summarizes our contributions.

CHAPTER 2

PRELIMINARIES

This chapter presents the preliminaries required for the trajectory planning algorithms described in the remaining chapters. We first introduce the set representation and corresponding set operations used in our reachability analysis in Chapters 3, 4 and 5. We then provide relevant background from the field of graph theory required for our connectivity maintenance algorithm in Chapter 6.

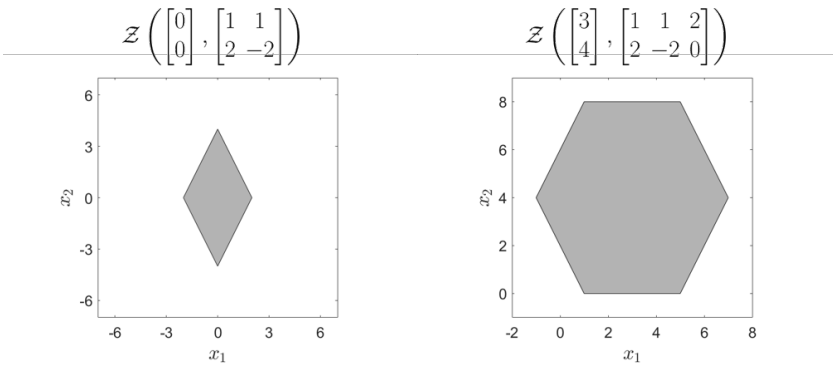
2.1 Set Representations and Operations

Various set representations have been used in previous reachability-based literature to represent reachable sets for the system. Zonotopes and their variants have been commonly used [31, 33, 36, 90, 91] due to their computational efficiency under common set operations such as the Minkowski sum and linear transform operations. Alternate approaches in [37, 38] use polynomial set representations that typically provide tighter sets compared to zonotopes, but come at an additional computational cost. In our reachability analysis, we use the stochastic variant of zonotopes introduced in [90], referred to as probabilistic zonotopes.

2.1.1 Probabilistic Zonotopes

Probabilistic zonotopes are probabilistic hulls that are suitable for enclosing multiple probability distributions. They have been shown to be computationally efficient and closed under the Minkowski sum and linear transform operations [92], which are required in our reachability analysis as shown in Chapters 3 and 4.

To define a probabilistic zonotope, we first begin by defining a n-dimensional



(a) (b)

Figure 2.1: Example visualizations of two 2-dimensional zonotopes with 2 and 3 generators respectively.

zonotope $\mathcal{P}$ as follows:

$$\mathcal{P} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \mathbf{c}_\mathcal{P} + \sum_{i=1}^{r} \beta_i \cdot \mathbf{g}_\mathcal{P}^{\{i\}}, -1 \leq \beta_i \leq 1 \right\}, \tag{2.1}$$

where $\mathbf{c}_\mathcal{P} \in \mathbb{R}^n$ is the center of the zonotope, and $\mathbf{g}_\mathcal{P}^{\{i\}} \in \mathbb{R}^n \ \forall \ i \in [1, r]$ are referred to as generators of the zonotope. The generators determine the shape of the zonotope relative to its center. Thus, a zonotope can be concisely written as $\mathcal{P} = \mathcal{Z}(\mathbf{c}_\mathcal{P}, G_\mathcal{P})$, where $G_\mathcal{P} = \left[\mathbf{g}_\mathcal{P}^{\{1\}}, \ldots, \mathbf{g}_\mathcal{P}^{\{r\}}\right]$ is the corresponding $n \times r$ generator matrix. Fig. 2.1 shows two 2-dimensional zonotopes along with their centers and corresponding generator matrices.

In [90], the authors introduced a stochastic variant of zonotopes, referred to as probabilistic zonotopes. A probabilistic zonotope is a probabilistic hull that contains a bounded component defined by a zonotope along with a stochastic component defined by a Gaussian distribution. Similar to a zonotope, a n-dimensional probabilistic zonotope $\mathscr{P}$ can be concisely written as:

$$\mathscr{P} = \mathscr{Z}(\mathbf{c}_\mathscr{P}, G_\mathscr{P}, \Sigma_\mathscr{P}), \tag{2.2}$$

where $\mathbf{c}_\mathscr{P}$ and $G_\mathscr{P}$ represent the center and the generator matrix for the zonotope defining the bounded component, and $\Sigma_\mathscr{P}$ is the $n \times n$ Gaussian covariance matrix defining the stochastic component.

A probabilistic zonotope can be visualized as a Gaussian distribution that has an uncertain bounded mean defined by a zonotope. Thus, it can be used to enclose multiple Gaussian distributions as illustrated for a 1-dimensional

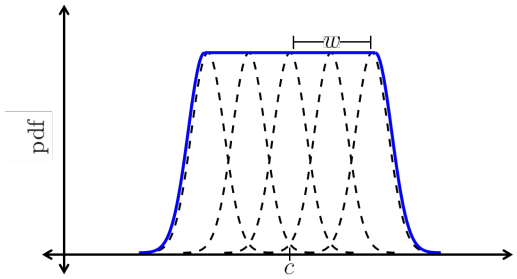
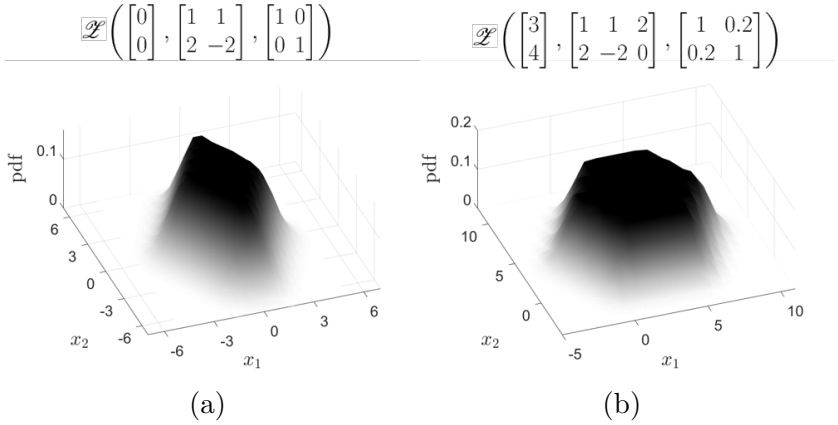


Figure 2.2: Example illustration of a 1-dimensional probabilistic zonotope (blue) enclosing multiple Gaussian distributions (black dashed). Here the Gaussian distributions $\mathcal{N}(b, \Sigma),\ b \in [c - w, c + w]$ are enclosed by the probabilistic zonotope $\mathscr{Z}(c, w, \Sigma)$.



(a) (b)

Figure 2.3: Example visualizations of two 2-dimensional probabilistic zonotopes where the bounded components correspond to the zonotopes visualized in Fig. 2.1.

case in Fig. 2.2. In [92], the authors show how probabilistic zonotopes can also be used to enclose non-Gaussian distributions. However, in this dissertation we use probabilistic zonotopes to only enclose Gaussian distributions. Fig. 2.3 shows two 2-dimensional probabilistic zonotopes with similar bounded components as in Fig. 2.1. Note that unlike regular probability distributions, probabilistic zonotopes do not have a normalized distribution since they enclose multiple distributions. Next, we define the set operations for probabilistic zonotopes required in our reachability analysis.

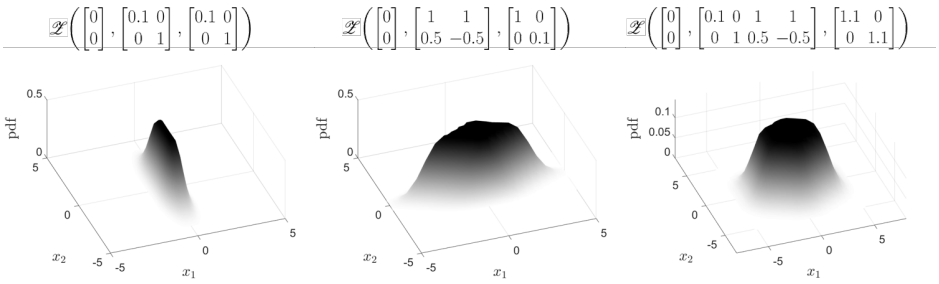


Figure 2.4: Example visualization of the Minkowski sum operation. The Minkowski sum of two probabilistic zonotopes in (a) and (b) is shown in (c).

2.1.2 Minkowski Sum Operation

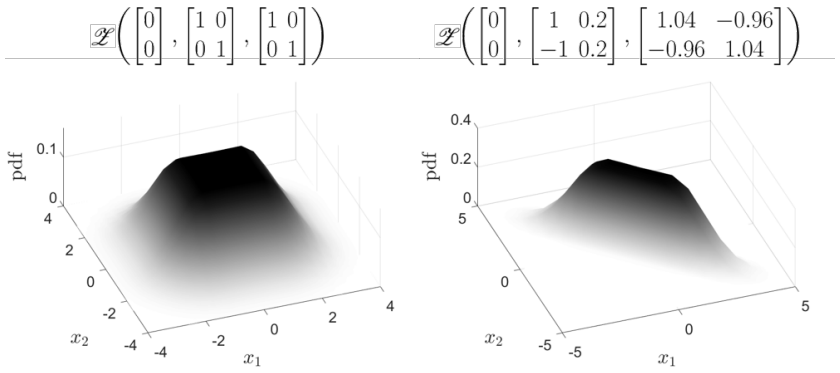
The Minkowski sum of two sets is generally formed by adding each element of the first set to each element in the second set. The Minkowski sum of two zonotopes is defined as:

$$\mathcal{P}_1 \oplus \mathcal{P}_2 = \mathcal{Z}\left(c_{\mathcal{P}_1} + c_{\mathcal{P}_2}, [G_{\mathcal{P}_1}, G_{\mathcal{P}_2}]\right), \tag{2.3}$$

i.e., the new center is the sum of the individual centers and the new generator matrix is a concatenation of the individual generator matrices. In order to account for the additional stochastic component in probabilistic zonotopes, the authors of [90], defined the Minkowski sum operation of two probabilistic zonotopes $\mathscr{P}_1$ and $\mathscr{P}_2$ as:

$$\mathscr{P}_1 \oplus \mathscr{P}_2 = \mathscr{Z}\left(c_{\mathscr{P}_1} + c_{\mathscr{P}_2}, [G_{\mathscr{P}_1}, G_{\mathscr{P}_2}], \Sigma_{\mathscr{P}_1} + \Sigma_{\mathscr{P}_2}\right), \tag{2.4}$$

where the new covariance matrix is the sum of the individual covariance matrices. Fig. 2.4 illustrates an example Minkowski sum operation of two 2-dimensional probabilistic zonotopes. From Equation 2.4 note that the Minkowski sum operation does not consider the correlation between the covariance matrices $\Sigma_{\mathscr{P}_1}$ and $\Sigma_{\mathscr{P}_2}$, and thus by definition assumes $\mathscr{P}_1$ and $\mathscr{P}_2$ to be represent independent quantities. This motivates our approach for computing the system reachable sets later in Chapters 3 and 4.



(a) (b)

Figure 2.5: Example visualization of the linear transform operation. The linear transformation $T = \begin{bmatrix} 1 & 0.2 \\ -1 & 0.2 \end{bmatrix}$ of a probabilistic zonotope in (a) is shown in (b).

2.1.3 Linear Transform Operation

The linear transform of a probabilistic zonotope $\mathscr{P}$ is defined as [90]:

$$T\mathscr{P} = \mathscr{Z}\left(Tc_{\mathscr{P}}, TG_{\mathscr{P}}, T\Sigma_{\mathscr{P}}T^{\top}\right), \tag{2.5}$$

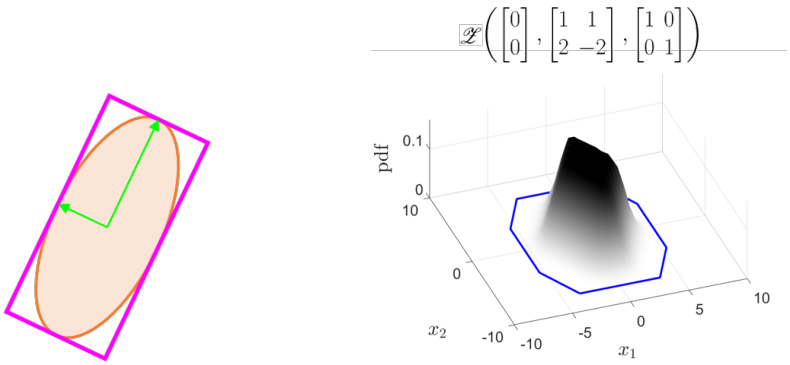
where T is a transformation matrix that maps both the bounded and stochastic components of $\mathscr{P}$. Fig. 2.5 shows an example linear transform operation for a 2-dimensional probabilistic zonotope.

2.1.4 Confidence Set Operation

The objective of the confidence set operation is to output a set that encloses a probabilistic zonotope with a desired confidence level. We begin by defining a zonotope that encloses a zero mean Gaussian distribution with covariance Σ. Let λ_i^{Σ} and $\mathbf{e}_i^{\Sigma}$ represent the i^{th} eigenvalue and the i^{th} eigenvector respectively of the covariance matrix. Then the enclosing zonotope for a $m\sigma$ confidence ellipsoid of the Gaussian distribution can be written as:

$$\texttt{enc}(\Sigma, m) = \mathcal{Z}\left(\mathbf{0}, s\left[\sqrt{\lambda_1^{\Sigma}}\mathbf{e}_1^{\Sigma}, \ldots, \sqrt{\lambda_n^{\Sigma}}\mathbf{e}_n^{\Sigma}\right]\right), \tag{2.6}$$

where s is a scalar factor that follows a chi-square distribution [93, 94] based on the desired confidence level. For our `MATLAB` implementation, we calculate



(a) (b)

Figure 2.6: (a) Enclosing zonotope for 2-dimensional Gaussian distribution. The principal semi-axes (green) of the confidence ellipsoid (orange) are used to define the generators for the enclosing zonotope (magenta). (b) Example visualization of the confidence set operation. The 3σ confidence set (blue) of a probabilistic zonotope is shown.

s as follows:

$$s = \texttt{sqrt}\left(\texttt{chi2inv}\left(\texttt{erf}\left(\frac{m}{\sqrt{2}}\right), n\right)\right), \tag{2.7}$$

where n is the number of dimensions. Fig. 2.6(a) illustrates the enclosing zonotope for 2-dimensional Gaussian distribution. Thus, for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, the enclosing zonotope satisfies the following equation:

$$\Pr(\mathbf{x} \in \texttt{enc}(\Sigma, m)) > 1 - \delta, \tag{2.8}$$

where $\delta = 1 - \texttt{erf}(m/\sqrt{2})$. For $m = 3$, i.e., for a 3σ confidence level the value of $(1 - \delta)$ is 0.9973. Next, since a probabilistic zonotope contains a bounded component in addition to the stochastic Gaussian component, we define the confidence set operation of a probabilistic zonotope $\mathscr{P} = \mathscr{Z}(\mathbf{c}_{\mathscr{P}}, G_{\mathscr{P}}, \Sigma_{\mathscr{P}})$ as:

$$\texttt{conf}\left(\mathscr{P}, m\right) = \mathcal{Z}\left(\mathbf{c}_{\mathscr{P}}, G_{\mathscr{P}}\right) \oplus \texttt{enc}(\Sigma_{\mathscr{P}}, m), \tag{2.9}$$

where the output confidence set is a zonotope. Fig. 2.6 shows the output of the confidence set operation on an example 2-dimensional probabilistic zonotope. Let $\phi(\mathbf{x})$ represent the Gaussian distributions enclosed by the probabilistic zonotope $\mathscr{P}$. Thus, the confidence set obtained in Equation (2.9) satisfies the following equation:

$$\min_{\phi(\mathbf{x})} \Pr(\mathbf{x} \in \texttt{conf}\left(\mathscr{P}, m\right)) > 1 - \delta, \tag{2.10}$$

i.e., for all enclosed distributions $\phi(\mathbf{x})$, the minimum probability of $\mathbf{x}$ belonging to the confidence set is greater than $1-\delta$. In our reachability analysis, we use the confidence set operation on the computed reachable sets for the robot. These provide us with sets containing the robots' position states (with a desired confidence level) which we then use for checking collision-safety.

2.1.5 Projection Operation

The projection operation is used to calculate the magnitude of a probabilistic zonotope along a specified direction. We define the $m\sigma$ confidence level projection of a probabilistic zonotope $\mathscr{P}$ along a direction $\mathbf{e}$ as:

$$\texttt{proj}(\mathscr{P}, \mathbf{e}, m) = \sum_{i=1}^{r} |\mathbf{e}^\top \cdot \mathbf{g}_{\mathscr{P}}^{\{i\}}| + s\sqrt{\mathbf{e}^\top \cdot \Sigma_{\mathscr{P}} \cdot \mathbf{e}}, \tag{2.11}$$

where s is a scalar factor depending on m as computed in Equation (2.7), and $\mathbf{g}_{\mathscr{P}}^{\{i\}}$ and $\Sigma_{\mathscr{P}}$ are the generators and the Gaussian covariance of $\mathscr{P}$. Here, the first term in Equation (2.11) represents the projection of the bounded component of $\mathscr{P}$ whereas the second term represents the projection of the stochastic component. Fig. 2.7 illustrates the projection operation on an example 2-dimensional probabilistic zonotope. We use the projection operation in order to approximate the linearization errors in our non-linear reachability analysis in Chapter 4.

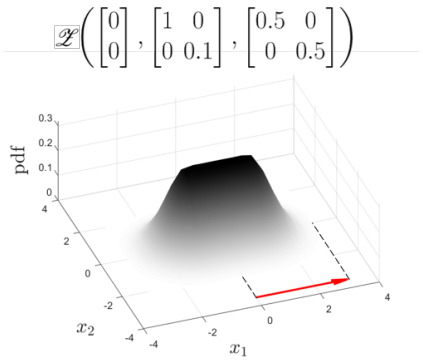


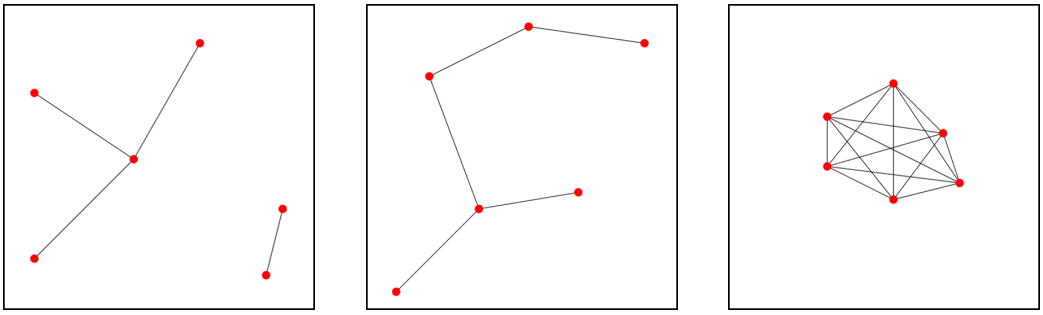
Figure 2.7: Example visualization of the projection operation. The 3σ projection of a probabilistic zonotope along the direction $\mathbf{e} = \begin{bmatrix}1\\0\end{bmatrix}$ is shown. The length of the red arrow shows the magnitude of the projection.

2.2 Graph Theory: Algebraic Connectivity

A multi-robot system can be represented as an undirected graph, where each node represents a robot and each edge represents the communication connectivity between two robots. As discussed in Section 1.2.5, algebraic connectivity of the graph is a commonly used metric to represent the global connectivity of the multi-robot system. To obtain the algebraic connectivity, we begin by defining the adjacency, degree and Laplacian matrices for the graph.

Let N be the number of nodes in the graph. The adjacency matrix $\mathcal{A}$ of the graph is defined as a $N \times N$ binary matrix such that $\mathcal{A}_{ij} = 1$ if nodes i and j are connected and $\mathcal{A}_{ij} = 0$ otherwise [95]. Additionally, the diagonal elements of the adjacency matrix are zeros, i.e., $\mathcal{A}_{ii} = 0$. The degree of a node d_i represents the number of nodes it is connected to, i.e., $d_i = \sum_{j=1}^{N} \mathcal{A}_{ij}$. The vector of node degrees $\mathbf{d}$ is then used to define the degree matrix D of the graph as $D = \texttt{diag}(\mathbf{d})$. Given matrices $\mathcal{A}$ and D the Laplacian matrix $\mathbb{L}$ is defined as $\mathbb{L} = D - \mathcal{A}$ [96].

The algebraic connectivity of the graph is defined as the second-smallest eigenvalue of $\mathbb{L}$, i.e., if $\lambda_1^{\mathbb{L}} \leq \lambda_2^{\mathbb{L}} \leq \cdots \leq \lambda_N^{\mathbb{L}}$ are the eigenvalues of $\mathbb{L}$, then $\lambda_2^{\mathbb{L}}$ is the algebraic connectivity of the graph. An important property of algebraic connectivity is that it not only gives an indication on whether the graph is connected or not, but it also gives an indication of *how* well-connected the graph is. The value of the algebraic connectivity of a graph varies from zero (if the graph is disconnected) to the number nodes (if the



(a) $\lambda_2^{\mathbb{L}} = 0$ (b) $\lambda_2^{\mathbb{L}} = 0.3$ (c) $\lambda_2^{\mathbb{L}} = 6$

Figure 2.8: The algebraic connectivity $\lambda_2^{\mathbb{L}}$ for different configurations of a graph with $N = 6$ nodes. $\lambda_2^{\mathbb{L}}$ varies from zero (disconnected graph) to the number of nodes N (fully-connected graph).

graph is fully connected), i.e., $0 \leq \lambda_2^{\mathbb{L}} \leq N$. Fig. 2.8 illustrates the algebraic connectivity for different configurations of a graph with $N = 6$ nodes. It is important to note here that the algebraic connectivity of a graph remains greater than 0 as long as the graph is globally connected, i.e., as long as there exists a (potentially multi-hop) connection between any two nodes in the graph. We use this property of algebraic connectivity as a constraint in our trajectory planning algorithm for connectivity maintenance in Chapter 6.

CHAPTER 3

LINEAR REACHABILITY ANALYSIS

In this chapter we derive our reachability analysis for robots that are represented by linear motion and sensing models. We first define the robotic system including the motion and sensing models and formulate our reachability problem in Section 3.1. Next, in Section 3.2, we describe the state estimation filter used on-board including our hypothesis to account for the presence of bounded biases in the measurements. We then develop our reachability analysis where we compute reachable sets for the robot along a single trajectory in Section 3.3. The reachable sets are represented by probabilistic zonotopes that enclose all possible state distributions along the trajectory. Finally, in Section 3.4, we discuss simulations for two linear robotic systems and statistically validate the computed reachable sets.

3.1 Problem Formulation

We consider a linear discrete-time system with the following motion model:

$$\mathbf{x}_t = A_{t-1}\mathbf{x}_{t-1} + B_{t-1}\mathbf{u}_{t-1} + \mathbf{w}_t \tag{3.1}$$

where t is the time instant, $\mathbf{x}_t$ is the state vector, $\mathbf{u}_t$ is the control input vector, A_t is state transition matrix, B_t is the control-input matrix and $\mathbf{w}_t$ is the motion model error represented by a Gaussian distribution with covariance Q_t, i.e., $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$. For the robot sensing model, we consider linear measurements of the form:

$$\mathbf{z}_t = C_t\mathbf{x}_t + \mathbf{v}_t, \tag{3.2}$$

where $\mathbf{z}_t$ is the measurement vector, C_t is the system measurement matrix and $\mathbf{v}_t$ is the sensing model error. We account for the presence of a bounded

bias along with a stochastic component in the sensing model error, i.e.:

$$\mathbf{v}_t = \mathbf{v}_t^b + \mathbf{v}_t^s, \tag{3.3}$$

where $\mathbf{v}_t^b$ is the bounded bias component represented by a zonotope $\mathcal{B}_t$, i.e., $\mathbf{v}_t^b \in [\underline{\mathbf{b}}_t, \overline{\mathbf{b}}_t] = \mathcal{B}_t$, and $\mathbf{v}_t^s$ is the stochastic component represented by a Gaussian distribution with covariance R_t, i.e., $\mathbf{v}_t^s \sim \mathcal{N}(\mathbf{0}, R_t)$.

In this chapter we focus on computing the reachable sets for the robot tracking a single nominal trajectory. We assume that this nominal trajectory is provided by a planner, the details of which are presented in Chapter 5. Additionally, we assume that the following information is available along the nominal trajectory:

1. Initial state estimation error covariance P_0.

2. Nominal states $(\check{\mathbf{x}}_0, \check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_T)$ and nominal inputs $(\check{\mathbf{u}}_0, \check{\mathbf{u}}_1, \ldots, \check{\mathbf{u}}_{T-1})$ for the nominal trajectory, where T represents the total number of discrete time-steps. Here we assume that the nominal states and inputs follow the robots linear motion model from Equation (3.1), i.e.:

$$\check{\mathbf{x}}_t = A_{t-1}\check{\mathbf{x}}_{t-1} + B_{t-1}\check{\mathbf{u}}_{t-1} \;\forall\; t \in [1, T]. \tag{3.4}$$

3. Stabilizing linear state feedback control gains $(\check{K}_0, \check{K}_1, \ldots, \check{K}_T)$ along the trajectory, such that the control input is of the form:

$$\mathbf{u}_t = \check{\mathbf{u}}_t - \check{K}_t(\hat{\mathbf{x}}_t - \check{\mathbf{x}}_t), \tag{3.5}$$

   where $\hat{\mathbf{x}}_t$ is the on-board state estimate that the robot obtains during trajectory execution using a Kalman filter as described in Section 3.2. Here the second term is the feedback control input that the robot applies in order to track the nominal trajectory.

4. Information regarding the sensing model error $\mathbf{v}_t$ along the trajectory, i.e., bounds $(\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_{T-1})$ for the bounded bias component $\mathbf{v}_t^b$ and Gaussian covariance matrices $(R_0, R_1, \ldots, R_{T-1})$ for the stochastic component $\mathbf{v}_t^s$.

Let $\phi(\mathbf{x}_t)$ denote the distribution of the robot state $\mathbf{x}_t$ at time instant t along the nominal trajectory. From Equations (3.1) and (3.5) we observe

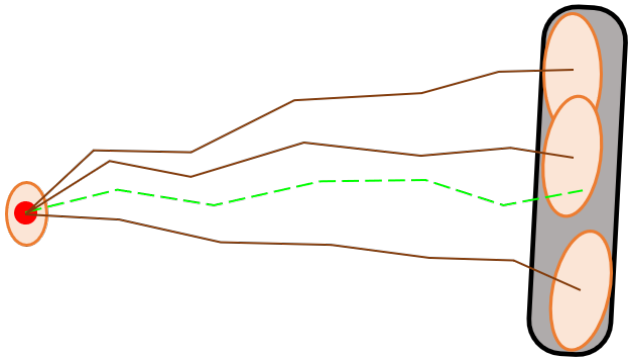


Figure 3.1: Given an initial state (red) and state distribution (orange ellipsoid), biases in sensor measurements along the nominal trajectory (green) lead to biases in future state distributions. The objective of this chapter is to develop a reachability analysis for linear robotic systems in order to predict bounds (black) that enclose the possible future state distributions associated with a desired confidence level.

that the state $\mathbf{x}_t$ depends on the previous state estimate $\hat{\mathbf{x}}_{t-1}$ via the applied feedback control. The state estimate in turn depends on the set of measurements received along the trajectory. Thus, each different set of biases in the measurements $\{\mathbf{v}_0^b, \cdots, \mathbf{v}_{t-1}^b\} \in \mathcal{B}_0 \times \cdots \times \mathcal{B}_{t-1}$ results in a different set of measurement distributions (Equation (3.2)), consequently resulting in a different state distribution $\phi(\mathbf{x}_t)$.

We define the problem for our linear reachability analysis as follows: given a robot with linear motion and sensing models (Equations (3.1) and (3.2)) along with a nominal trajectory including the information specified above, compute sets $\mathcal{R}$ along the trajectory, such that:

$$\min_{\phi(\mathbf{x}_t)} \ \Pr(\mathbf{x}_t \in \mathcal{R}_t) > 1 - \delta, \ \forall \ t \in [1, T], \tag{3.6}$$

i.e., the probability of the robot state $\mathbf{x}_t$ belonging to the computed set $\mathcal{R}_t$ is greater than $1 - \delta$ for all possible state distributions $\phi(\mathbf{x}_t)$ throughout the nominal trajectory. Here δ is a probability value obtained from a desired confidence level. Thus, for a $m\sigma$ confidence level, where $m > 0$, δ is calculated as: $\delta = 1 - \texttt{erf}(m/\sqrt{2})$. Fig. 3.1 illustrates the defined problem.

3.2 On-board State Estimation

During trajectory execution, the true state of the robot is not available due to the presence of uncertainties in the motion and sensing models. Thus, in order to track the nominal trajectory the robot applies control inputs based on an estimate of the state, as shown in Equation (3.5). To obtain a state estimate during trajectory execution, we use a Kalman Filter (KF) on-board the robot. The prediction step of the filter is performed as [97]:

$$\bar{\mathbf{x}}_t = A_{t-1}\hat{\mathbf{x}}_{t-1} + B_{t-1}\mathbf{u}_{t-1}, \tag{3.7}$$

$$\bar{P}_t = A_{t-1}P_{t-1}A_{t-1}^{\top} + Q_t, \tag{3.8}$$

where Q_t is the Gaussian covariance matrix for the motion model error defined in Equation (3.1). The measurements available to the filter can be re-written from Equation (3.2) as follows:

$$\mathbf{z}_t = C_t\mathbf{x}_t + \mathbf{c}_{b_t} + \epsilon_t, \tag{3.9}$$

where $\mathbf{c}_{b_t} = (\underline{\mathbf{b}}_t + \overline{\mathbf{b}}_t)/2$ is the center of bounds $\mathcal{B}_t$, and ϵ_t is a Gaussian distribution with a bounded mean, i.e.:

$$\epsilon_t \sim \mathcal{N}(\mathbf{v}_t^b - \mathbf{c}_{b_t}, R_t), \quad \mathbf{v}_t^b \in \mathcal{B}_t. \tag{3.10}$$

For the KF correction step, we choose an over-bounding hypothesis $\hat{R}_t$ as the measurement covariance matrix in order to account for the presence of the bounded bias component in the measurements. The over-bounding is performed such that $\hat{R}_t$ matches the tail of the possible Gaussian distributions representing ϵ_t in Equation (3.10). Here the distributions are matched at a desired $m\sigma$ confidence level. Thus, the covariance for each measurement $z_t^{(i)}$ in Equation (3.9) is computed as:

$$\hat{R}_t^{(i,i)} = \left[\frac{\left(w_{b_t}^{(i)} + m\sqrt{R_t^{(i,i)}} \right)}{m} \right]^2, \tag{3.11}$$

where $\mathbf{w}_{b_t} = (\overline{\mathbf{b}}_t - \mathbf{c}_{b_t})$ represents the half-width vector of the bounds $\mathcal{B}_t$, and the superscripts (i) and (i,i) refer to the i^{th} and (i,i) element of the

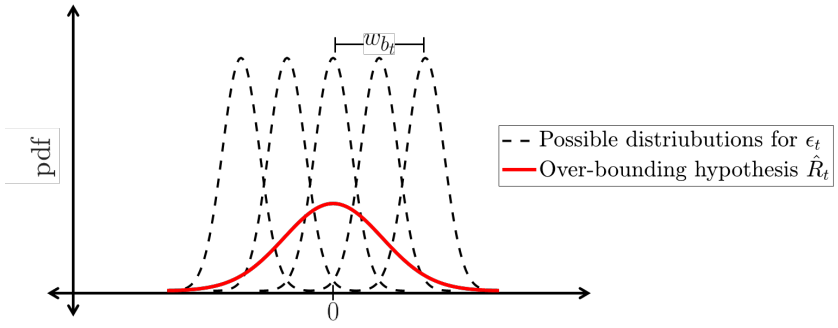


Figure 3.2: Illustration of the over-bounding hypothesis $\hat{R}_t$ used for the measurement covariance matrix in the state estimation filter. The tail of the distribution of $\hat{R}_t$ matches the tail of the possible distributions for ϵ_t at a desired confidence level, such as 3σ confidence.

corresponding vector and matrix respectively. We set the off-diagonal elements in $\hat{R}_t$ to 0. Fig. 3.2 illustrates the over-bounding hypothesis for a single measurement. Note that our reachability analysis does not necessarily require choosing our over-bounding hypothesis $\hat{R}_t$ for the KF measurement covariance matrix. If desired, a different hypothesis can be chosen for the measurement covariance matrix $\hat{R}_t$ and used with the rest of the analysis. We choose the over-bounding hypothesis since it is equivalent to scaling or inflating the covariance matrix, which is a commonly used approach for practical implementation of the KF and its variants [98, 99, 100].

Once measurement covariance matrix $\hat{R}_t$ has been computed using Equation (3.11), the KF correction step is performed as [97]:

$$L_t = \bar{P}_t C_t^\top (C_t \bar{P}_t C_t^\top + \hat{R}_t)^{-1}, \tag{3.12}$$

$$\hat{\mathbf{x}}_t = \bar{\mathbf{x}}_t + L_t(\mathbf{z}_t - C_t \bar{\mathbf{x}}_t - \mathbf{c}_{b_t}), \tag{3.13}$$

$$P_t = \bar{P}_t - L_t C_t \bar{P}_t, \tag{3.14}$$

where L_t is the Kalman gain.

3.3 Computing Reachable Sets for Linear Systems

In this section, we develop our analysis to compute reachable sets for a robot with linear motion and sensing models, such that these sets satisfy the requirement from Equation (3.6). We first derive the equations governing the

growth of the robot state vector and state estimation error vector during trajectory execution, and later transition into a set notation. While transitioning an equation from a vector notation into a set notation, a sum of two vectors transitions into a Minkowski sum of the corresponding sets. As described in Section 2.1.2, the Minkowski sum operation of two probabilistic zonotopes assumes them to be independent. Thus, we derive equations for the robot state and state estimation error vectors as a function of independent quantities along the nominal trajectory and then transition into a set notation.

We begin by subtracting the nominal state $\check{\mathbf{x}}_t$ in Equation (3.4) from the robot motion model in Equation (3.1), giving us:

$$\mathbf{x}_t - \check{\mathbf{x}}_t = A_{t-1}\mathbf{x}_{t-1} + B_{t-1}\mathbf{u}_{t-1} + \mathbf{w}_t - A_{t-1}\check{\mathbf{x}}_{t-1} - B_{t-1}\check{\mathbf{u}}_{t-1}, \tag{3.15}$$

$$= A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\mathbf{u}_{t-1} - \check{\mathbf{u}}_{t-1}) + \mathbf{w}_t. \tag{3.16}$$

As mentioned in Section 3.1, the robot uses linear state feedback control during trajectory execution. Thus, using the definition of the input vector $\mathbf{u}_{t-1}$ from Equation (3.5) we get:

$$\begin{aligned}\mathbf{x}_t = \check{\mathbf{x}}_t + A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\check{\mathbf{u}}_{t-1} - \check{K}_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) - \\ \check{\mathbf{u}}_{t-1})) + \mathbf{w}_t,\end{aligned} \tag{3.17}$$

$$= \check{\mathbf{x}}_t + A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + \mathbf{w}_t. \tag{3.18}$$

Here we define the state estimation error $\tilde{\mathbf{x}}_t$ as the difference between the estimated state $\hat{\mathbf{x}}_t$ and the true state $\mathbf{x}_t$, i.e., $\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t$. Thus, we rewrite Equation (3.18) as:

$$\mathbf{x}_t = \check{\mathbf{x}}_t + A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\mathbf{x}_{t-1} + \tilde{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + \mathbf{w}_t, \tag{3.19}$$

$$\begin{aligned}= \check{\mathbf{x}}_t + A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - \\ B_{t-1}\check{K}_{t-1}\tilde{\mathbf{x}}_{t-1} + \mathbf{w}_t,\end{aligned} \tag{3.20}$$

$$= \check{\mathbf{x}}_t + (A_{t-1} - B_{t-1}\check{K}_{t-1})(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}\tilde{\mathbf{x}}_{t-1} + \mathbf{w}_t. \tag{3.21}$$

In order to obtain the state estimation error $\tilde{\mathbf{x}}_{t-1}$ required in Equation (3.21), we begin with state estimation filter equations in Section 3.2. Using the KF correction step from Equation (3.13) and the robot motion model from

Equation (3.1) we get:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= \hat{\mathbf{x}}_t - \mathbf{x}_t \\ &= \bar{\mathbf{x}}_t + L_t\left(\mathbf{z}_t - C_t\bar{\mathbf{x}}_t - \mathbf{c}_{b_t}\right) - A_{t-1}\mathbf{x}_{t-1} - B_{t-1}\mathbf{u}_{t-1} - \mathbf{w}_t. \end{aligned} \tag{3.22}$$

On replacing the KF predicted state $\bar{\mathbf{x}}_t$ by Equation (3.7) and rearranging the terms, we obtain:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= A_{t-1}\hat{\mathbf{x}}_{t-1} + B_{t-1}\mathbf{u}_{t-1} + L_t(\mathbf{z}_t - C_t(A_{t-1}\hat{\mathbf{x}}_{t-1} + B_{t-1}\mathbf{u}_{t-1}) - \\ &\quad \mathbf{c}_{b_t}) - A_{t-1}\mathbf{x}_{t-1} - B_{t-1}\mathbf{u}_{t-1} - \mathbf{w}_t, \end{aligned} \tag{3.23}$$

$$= A_{t-1}\left(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}\right) + L_t\left(\mathbf{z}_t - \mathbf{c}_{b_t} - C_t\left(A_{t-1}\hat{\mathbf{x}}_{t-1} + B_{t-1}\mathbf{u}_{t-1}\right)\right) - \mathbf{w}_t. \tag{3.24}$$

Furthermore, on replacing the measurement vector $\mathbf{z}_t$ from Equation (3.9) and using the robot motion model for $\mathbf{x}_t$ from Equation (3.1), we get:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= A_{t-1}\left(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}\right) + L_t(C_t\mathbf{x}_t + \mathbf{c}_{b_t} + \epsilon_t - \mathbf{c}_{b_t} - C_t(A_{t-1}\hat{\mathbf{x}}_{t-1} + \\ &\quad B_{t-1}\mathbf{u}_{t-1})) - \mathbf{w}_t, \end{aligned} \tag{3.25}$$

$$\begin{aligned}&= A_{t-1}\left(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}\right) + L_t(C_t\left(A_{t-1}\mathbf{x}_{t-1} + B_{t-1}\mathbf{u}_{t-1} + \mathbf{w}_t\right) + \epsilon_t - \\ &\quad C_t\left(A_{t-1}\hat{\mathbf{x}}_{t-1} + B_{t-1}\mathbf{u}_{t-1}\right)) - \mathbf{w}_t, \end{aligned} \tag{3.26}$$

$$\begin{aligned}&= A_{t-1}\left(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}\right) + L_t(C_tA_{t-1}\mathbf{x}_{t-1} + C_tB_{t-1}\mathbf{u}_{t-1} + C_t\mathbf{w}_t + \\ &\quad \epsilon_t - C_tA_{t-1}\hat{\mathbf{x}}_{t-1} - C_tB_{t-1}\mathbf{u}_{t-1}) - \mathbf{w}_t. \end{aligned} \tag{3.27}$$

The terms with the input vector $\mathbf{u}_{t-1}$ cancel each other, resulting in:

$$\tilde{\mathbf{x}}_t = A_{t-1}(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}) + L_t(C_t(A_{t-1}(\mathbf{x}_{t-1} - \hat{\mathbf{x}}_{t-1}) + \mathbf{w}_t) + \epsilon_t) - \mathbf{w}_t, \tag{3.28}$$

$$\begin{aligned}&= A_{t-1}(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}) - L_tC_t(A_{t-1}(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1})) + L_tC_t\mathbf{w}_t + \\ &\quad L_t\epsilon_t - \mathbf{w}_t, \end{aligned} \tag{3.29}$$

$$= (I - L_tC_t)A_{t-1}(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}) - (I - L_tC_t)\mathbf{w}_t + L_t\epsilon_t, \tag{3.30}$$

$$= (I - L_tC_t)A_{t-1}\tilde{\mathbf{x}}_{t-1} - (I - L_tC_t)\mathbf{w}_t + L_t\epsilon_t, \tag{3.31}$$

where I is an identity matrix of appropriate dimensions.

Note that while Equations (3.21) and (3.31) govern the growth of the state and the state estimation errors respectively, they involve the summation of correlated quantities. For instance, in Equation (3.21) the previous state

$\mathbf{x}_{t-1}$ and the previous state estimation error $\tilde{\mathbf{x}}_{t-1}$, i.e.,

$$\begin{aligned}\mathbf{x}_{t-1} = \check{\mathbf{x}}_{t-1} + (A_{t-2} - B_{t-2}\check{K}_{t-2})(\mathbf{x}_{t-2} - \check{\mathbf{x}}_{t-2}) - B_{t-2}\check{K}_{t-2}\tilde{\mathbf{x}}_{t-2} + \\ \mathbf{w}_{t-1},\end{aligned} \tag{3.32}$$

$$\tilde{\mathbf{x}}_{t-1} = (I - L_{t-1}C_{t-1})A_{t-2}\tilde{\mathbf{x}}_{t-2} - (I - L_{t-1}C_{t-1})\mathbf{w}_{t-1} + L_{t-1}\epsilon_{t-1}, \tag{3.33}$$

both depend on the previous motion model error $\mathbf{w}_{t-1}$. Thus, as explained at the beginning of this section, we cannot directly transition Equations (3.21) and (3.31) into set notations in order to compute the reachable sets for the robot. Instead, we use Equations (3.21) and (3.31) along with the process of mathematical induction to derive the robot state and state estimation error as a function of independent quantities.

We propose that the following expressions of $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ hold true for any time instant t:

$$\mathbf{x}_t = \check{\mathbf{x}}_t + {}^1\Phi_t(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^2\Phi_t\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t} {}^3\Phi_t^n\mathbf{w}_n + \sum_{n=1}^{t} {}^4\Phi_t^n\epsilon_n, \tag{3.34}$$

$$\tilde{\mathbf{x}}_t = {}^2\tilde{\Phi}_t\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t} {}^3\tilde{\Phi}_t^n\mathbf{w}_n + \sum_{n=1}^{t} {}^4\tilde{\Phi}_t^n\epsilon_n, \tag{3.35}$$

where all Φ_t and $\tilde{\Phi}_t$ are matrix coefficients derived from the system and state estimation matrices defined in Sections 3.1 and 3.2. These matrix coefficients can be interpreted as weights that decide how much the corresponding quantities, i.e., $\mathbf{x}_0$, $\tilde{\mathbf{x}}_0$, $\mathbf{w}_n$ and ϵ_n influence the state $\mathbf{x}_t$ and state estimation error $\tilde{\mathbf{x}}_t$ at time instant t. Here $\mathbf{x}_0$ is the initial state of the system, $\tilde{\mathbf{x}}_0$ is the initial state estimation error, $\mathbf{w}_n$ and ϵ_n are motion and sensing model errors from Equations (3.1) and (3.9) that are independently sampled along the trajectory.

We prove that Equations (3.34) and (3.35) hold true for any time instant t by induction. For the induction base case, we show that Equations (3.34) and (3.35) are valid for $t = 0$. The initial state $\mathbf{x}_0$ and state estimation error $\tilde{\mathbf{x}}_0$ can be written in the form of Equations (3.34) and (3.35) as:

$$\begin{aligned}\mathbf{x}_0 &= \check{\mathbf{x}}_0 + (\mathbf{x}_0 - \check{\mathbf{x}}_0), \\ \tilde{\mathbf{x}}_0 &= \tilde{\mathbf{x}}_0,\end{aligned} \tag{3.36}$$

which can be obtained by setting ${}^{1}\Phi_0 = I$, ${}^{i}\Phi_0 = O \; \forall \; i = \{2, 3, 4\}$ in Equation (3.34), and ${}^{2}\tilde{\Phi}_0 = I$, ${}^{i}\tilde{\Phi}_0 = O \; \forall \; i = \{3, 4\}$ in Equation (3.35). Here O is a zero matrix of appropriate dimensions.

For the induction step, we begin by assuming that Equations (3.34) and (3.35) hold true for time instant $t-1$, i.e.:

$$\mathbf{x}_{t-1} = \check{\mathbf{x}}_{t-1} + {}^{1}\Phi_{t-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^{2}\Phi_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^{3}\Phi_{t-1}^{n}\mathbf{w}_n + \sum_{n=1}^{t-1} {}^{4}\Phi_{t-1}^{n}\epsilon_n, \tag{3.37}$$

$$\tilde{\mathbf{x}}_{t-1} = {}^{2}\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^{3}\tilde{\Phi}_{t-1}^{n}\mathbf{w}_n + \sum_{n=1}^{t-1} {}^{4}\tilde{\Phi}_{t-1}^{n}\epsilon_n. \tag{3.38}$$

Next, we first show that Equation (3.35) for the state estimation error vector holds true for time instant t. We replace Equation (3.38) in Equation (3.31) to obtain the following expression for the state estimation error $\tilde{\mathbf{x}}_t$:

$$\begin{aligned}\tilde{\mathbf{x}}_t = {} & (I - L_tC_t)A_{t-1}{}^{2}\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + (I - L_tC_t)A_{t-1}\sum_{n=1}^{t-1} {}^{3}\tilde{\Phi}_{t-1}^{n}\mathbf{w}_n + \\ & (I - L_tC_t)A_{t-1}\sum_{n=1}^{t-1} {}^{4}\tilde{\Phi}_{t-1}^{n}\epsilon_n - (I - L_tC_t)\mathbf{w}_t + L_t\epsilon_t \end{aligned} \tag{3.39}$$

$$\begin{aligned} = {} & \Big((I - L_tC_t)A_{t-1}{}^{2}\tilde{\Phi}_{t-1}\Big)\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1}\Big((I - L_tC_t)A_{t-1}{}^{3}\tilde{\Phi}_{t-1}^{n}\Big)\mathbf{w}_n + \\ & \Big(- (I - L_tC_t)\Big)\mathbf{w}_t + \sum_{n=1}^{t-1}\Big((I - L_tC_t)A_{t-1}{}^{4}\tilde{\Phi}_{t-1}^{n}\Big)\epsilon_n + L_t\epsilon_t. \end{aligned} \tag{3.40}$$

The above equation can then be written in the form of Equation (3.35) where the $\tilde{\Phi}_t$ matrix coefficients can be obtained as:

$$\begin{aligned} {}^{2}\tilde{\Phi}_t &= (I - L_tC_t)A_{t-1}{}^{2}\tilde{\Phi}_{t-1}, \\ {}^{3}\tilde{\Phi}_t^n &= (I - L_tC_t)A_{t-1}{}^{3}\tilde{\Phi}_{t-1}^{n} \; \forall \; n \in [1, t-1], \\ {}^{3}\tilde{\Phi}_t^t &= -(I - L_tC_t), \\ {}^{4}\tilde{\Phi}_t^n &= (I - L_tC_t)A_{t-1}{}^{4}\tilde{\Phi}_{t-1}^{n} \; \forall \; n \in [1, t-1], \\ {}^{4}\tilde{\Phi}_t^t &= L_t. \end{aligned} \tag{3.41}$$

Next, we show that Equation (3.34) for the state vector holds true for time instant t. We replace Equations (3.37) and (3.38) in Equation (3.21) to

obtain the following expression for the state $\mathbf{x}_t$:

$$\begin{aligned}\mathbf{x}_t = \check{\mathbf{x}}_t + (A_{t-1} - B_{t-1}\check{K}_{t-1})\Big({}^1\Phi_{t-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^2\Phi_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^3\Phi_{t-1}^n \mathbf{w}_n + \\ \sum_{n=1}^{t-1} {}^4\Phi_{t-1}^n \epsilon_n\Big) - B_{t-1}\check{K}_{t-1}\Big({}^2\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^3\tilde{\Phi}_{t-1}^n \mathbf{w}_n + \sum_{n=1}^{t-1} {}^4\tilde{\Phi}_{t-1}^n \epsilon_n\Big) + \\ \mathbf{w}_t,\end{aligned} \tag{3.42}$$

$$\begin{aligned}= \check{\mathbf{x}}_t + \Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^1\Phi_{t-1}\Big)(\mathbf{x}_0 - \check{\mathbf{x}}_0) + \\ \Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^2\Phi_{t-1} - B_{t-1}\check{K}_{t-1}{}^2\tilde{\Phi}_{t-1}\Big)\tilde{\mathbf{x}}_0 + \\ \sum_{n=1}^{t-1}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^3\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^3\tilde{\Phi}_{t-1}^n\Big)\mathbf{w}_n + \mathbf{w}_t + \\ \sum_{n=1}^{t-1}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^4\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^4\tilde{\Phi}_{t-1}^n\Big)\epsilon_n.\end{aligned} \tag{3.43}$$

The above equation can then be written in the form of Equation (3.34) where the Φ_t matrix coefficients can be obtained as:

$$\begin{aligned}{}^1\Phi_t &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^1\Phi_{t-1}, \\ {}^2\Phi_t &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^2\Phi_{t-1} - B_{t-1}\check{K}_{t-1}{}^2\tilde{\Phi}_{t-1}, \\ {}^3\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^3\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^3\tilde{\Phi}_{t-1}^n \;\forall\; n \in [1, t-1], \\ {}^3\Phi_t^t &= I, \\ {}^4\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^4\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^4\tilde{\Phi}_{t-1}^n \;\forall\; n \in [1, t-1], \\ {}^4\Phi_t^t &= O.\end{aligned} \tag{3.44}$$

Thus, by the principle of induction, we can claim that Equations (3.34) and (3.35) hold true for all time instants along the trajectory, i.e., $\forall\, t \in [0, T]$. The matrix coefficients Φ_t and $\tilde{\Phi}_t$ required in these equations can be initialized as shown in Equation (3.36) and can be obtained recursively along the trajectory using Equations (3.41) and (3.44).

While Equations (3.34) and (3.35) govern the growth of the robot state $\mathbf{x}_t$ and state estimation error $\tilde{\mathbf{x}}_t$, their exact values cannot be computed since the exact values are not known for the initial robot state $\mathbf{x}_0$, the initial state

estimation error $\tilde{\mathbf{x}}_0$ and the motion and sensing model errors $\mathbf{w}_t$ and ϵ_t along the trajectory. However, as mentioned in Section 3.1, we assume the following information is available: the initial state estimation error covariance P_0, the Gaussian covariance matrices for the motion model error Q_t, the bounds for the bias component in the sensing error $\mathcal{B}_t$, and the Gaussian covariance matrices for the stochastic component in the sensing model error R_t. Thus, we instead transition Equations (3.34) and (3.35) to set notations, where we compute the set of reachable robot states $\mathcal{X}_t$ as a function of the set of initial robot states $\mathcal{X}_0$, the initial state estimation error set $\tilde{\mathcal{X}}_0$ and the sets of motion and sensing uncertainties $\mathcal{W}_t$ and $\mathcal{V}_t$ along the trajectory. The reachable set $\mathcal{X}_t$ and the state estimation error set $\tilde{\mathcal{X}}_t$ at any time instant t are computed as:

$$\mathcal{X}_t = \check{x}_t \oplus {}^1\Phi_t(\mathcal{X}_0 - \check{x}_0) \oplus {}^2\Phi_t\tilde{\mathcal{X}}_0 \oplus \sum_{n=1}^{t} {}^3\Phi_t^n\mathcal{W}_n \oplus \sum_{n=1}^{t} {}^4\Phi_t^n\mathcal{V}_n, \tag{3.45}$$

$$\tilde{\mathcal{X}}_t = {}^2\tilde{\Phi}_t\tilde{\mathcal{X}}_0 \oplus \sum_{n=1}^{t} {}^3\tilde{\Phi}_t^n\mathcal{W}_n \oplus \sum_{n=1}^{t} {}^4\tilde{\Phi}_t^n\mathcal{V}_n, \tag{3.46}$$

where $\oplus$ is the Minkowski sum operation described in Section 2.1.2. As discussed in Section 2.1, we use the probabilistic zonotope set representation for our reachability analysis since it allows us to account for both the bounded and the stochastic components of uncertainties in the system.

We use the initial state estimation error covariance P_0 and the initial nominal state $\check{\mathbf{x}}_0$ to define the set of initial robot states $\mathcal{X}_0$ and the initial state estimation error set $\tilde{\mathcal{X}}_0$ as:

$$\begin{aligned} \mathcal{X}_0 &= \mathscr{Z}(\check{\mathbf{x}}_0, \mathbf{0}, P_0), \\ \tilde{\mathcal{X}}_0 &= \mathscr{Z}(\mathbf{0}, \mathbf{0}, P_0), \end{aligned} \tag{3.47}$$

where the definition of a probabilistic zonotope $\mathscr{Z}$ follows from Equation (2.2) in Section 2.1.1. The sets of motion and sensing uncertainties $\mathcal{W}_t$ and $\mathcal{V}_t$ in Equations (3.45) and (3.46) represent the possible motion and sensing model errors, i.e., $\mathbf{w}_t$ from Equation (3.1) and ϵ_t from Equation (3.9) respectively. As mentioned in Equation (3.1), we assume $\mathbf{w}_t$ to be represented by a Gaussian distribution with covariance Q_t, i.e., $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, Q_t)$. Thus, the

corresponding set $\mathcal{W}_t$ is defined as:

$$\mathcal{W}_t = \mathscr{Z}(\mathbf{0}, \mathbf{0}, Q_t). \tag{3.48}$$

For the sensing model error in Equation (3.10), we assume ϵ_t to be represented by a Gaussian distribution with covariance R_t and with a bounded mean, i.e., $\epsilon_t \sim \mathcal{N}(\mathbf{v}_t^b - \mathbf{c}_{b_t}, R_t), \quad \mathbf{v}_t^b \in \mathcal{B}_t$. Note that the bounded mean for ϵ_t is centered at $\mathbf{0}$ as shown in Fig. 3.2, and has a half-width of $\mathbf{w}_{b_t}$ as mentioned in Equation (3.11). Thus, we define the corresponding set $\mathcal{V}_t$ for ϵ_t as:

$$\mathcal{V}_t = \mathscr{Z}(\mathbf{0}, \texttt{diag}(\mathbf{w}_{b_t}), R_t). \tag{3.49}$$

Given the definition of the sets in Equations (3.47), (3.48) and (3.49) along with Equations (3.41) and (3.44) to update the Φ matrix coefficients, we use Equation (3.45) to compute the reachable sets for the robot along the nominal trajectory. Finally, in order to obtain the sets $\mathcal{R}_t$ required in the problem formulation in Equation (3.6), we use the confidence set operation on the computed reachable sets:

$$\mathcal{R}_t = \texttt{conf}(\mathcal{X}_t, m), \tag{3.50}$$

where m represents the desired confidence level, such as 3σ ($m = 3$) confidence, as specified in Section 3.1. Algorithm 1 summarizes the inputs to our linear reachability analysis followed by the process to obtain the sets $\mathcal{R}_t$ along the nominal trajectory.

3.4 Simulation Results

In this section, we discuss our simulations in order to validate our linear reachability analysis. We simulate two 2-dimensional linear robotic systems commonly used in literature: a single-integrator system and a double-integrator system. For each system we first specify the nominal trajectory along with the inputs listed out in Algorithm 1. We then follow the rest of Algorithm 1 to obtain the sets $\mathcal{R}_t$ using Equation (3.50). In order to validate that these sets satisfy the requirement in Equation (3.6), we simulate a 1000 trajectory rollouts for the robot and check the ratio of trajectories with state

Algorithm 1 Linear Reachability Analysis

0: **Inputs**: Robot motion and sensing model (Equations (3.1) and (3.2)); initial state estimation error covariance P_0; nominal states $(\check{\mathbf{x}}_0, \ldots, \check{\mathbf{x}}_T)$ and nominal inputs $(\check{\mathbf{u}}_0, \ldots, \check{\mathbf{u}}_{T-1})$; measurement bias bounds $(\check{\mathcal{B}}_0, \ldots, \check{\mathcal{B}}_{T-1})$ and measurement Gaussian covariance matrices $(R_0, \ldots, R_{T-1})$; desired confidence level m.
1: Obtain initial robot states $\mathcal{X}_0$ and initial state estimation error set $\tilde{\mathcal{X}}_0$ (Equation (3.47)), motion uncertainty sets $(\check{\mathcal{W}}_0, \ldots, \check{\mathcal{W}}_{T-1})$ (Equation (3.48)), and sensing uncertainty sets $(\check{\mathcal{V}}_0, \ldots, \check{\mathcal{V}}_{T-1})$ (Equation (3.49)).
2: Initialize matrix coefficients Φ_0 and $\tilde{\Phi}_0$ (Equation (3.36)).
3: **for** $t = 1, \ldots, T$ **do**
4: Update matrix coefficients $\tilde{\Phi}_t$ (Equation (3.41)).
5: Update matrix coefficients Φ_t (Equation (3.44)).
6: Compute reachable set $\mathcal{X}_t$ (Equation (3.45)).
7: Obtain confidence sets $\mathcal{R}_t$ (Equation (3.50)).
8: **end for**

$\mathbf{x}_t$ inside the set $\mathcal{R}_t$. Thus, for a 3σ confidence level ($m = 3$) used in our simulations we expect the ratio to be greater than 0.997.

3.4.1 Single-integrator System

For a single-integrator system, the states consist of the robot positions and the control inputs consist of the robot velocities. Thus, the motion model for the robot can be written in the form of Equation (3.1) as:

$$\mathbf{x}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}_{t-1} + \begin{bmatrix} dt & 0 \\ 0 & dt \end{bmatrix} \mathbf{u}_{t-1} + \mathbf{w}_t, \tag{3.51}$$

where $dt = 0.2\,\mathrm{s}$ is the time-step between two time instants and $\mathbf{w}_t$ is the motion model error with covariance Q_t as:

$$\mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 0.01\,\mathrm{m}^2 & 0 \\ 0 & 0.01\,\mathrm{m}^2 \end{bmatrix}\right).$$

For the sensing model, we assume the availability of positioning measurements. Thus, the sensing model can be written in the form of Equation (3.2) as:

$$\mathbf{z}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}_t + \mathbf{v}_t, \tag{3.52}$$

where $\mathbf{v}_t$ is the error in the positioning measurements that contains a bounded bias component $\mathbf{v}_t^b \in \mathcal{B}_t$ and a stochastic component $\mathbf{v}_t^s \sim \mathcal{N}(\mathbf{0}, R_t)$. For the bounded bias component $\mathbf{v}_t^b$, we set the bounds to be $\pm 0.5\,\mathrm{m}$ in each position dimension, i.e., we set $\mathcal{B}_t$ as:

$$\mathcal{B}_t = \left[\begin{bmatrix} -0.5\,\mathrm{m} \\ -0.5\,\mathrm{m} \end{bmatrix}, \begin{bmatrix} 0.5\,\mathrm{m} \\ 0.5\,\mathrm{m} \end{bmatrix} \right]. \tag{3.53}$$

For the stochastic component $\mathbf{v}_t^s$, we set the covariance matrix R_t as:

$$\mathbf{v}_t^s \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 0.1\,\mathrm{m}^2 & 0 \\ 0 & 0.1\,\mathrm{m}^2 \end{bmatrix} \right). \tag{3.54}$$

The state feedback control gains $(\check{K}_0, \check{K}_1, \ldots, \check{K}_T)$ along the trajectory can be obtained using any control design techniques. For our simulations we use a Linear–Quadratic Regulator (LQR) obtained using optimal control theory. Specifically, we use the following function in our `MATLAB` implementation:

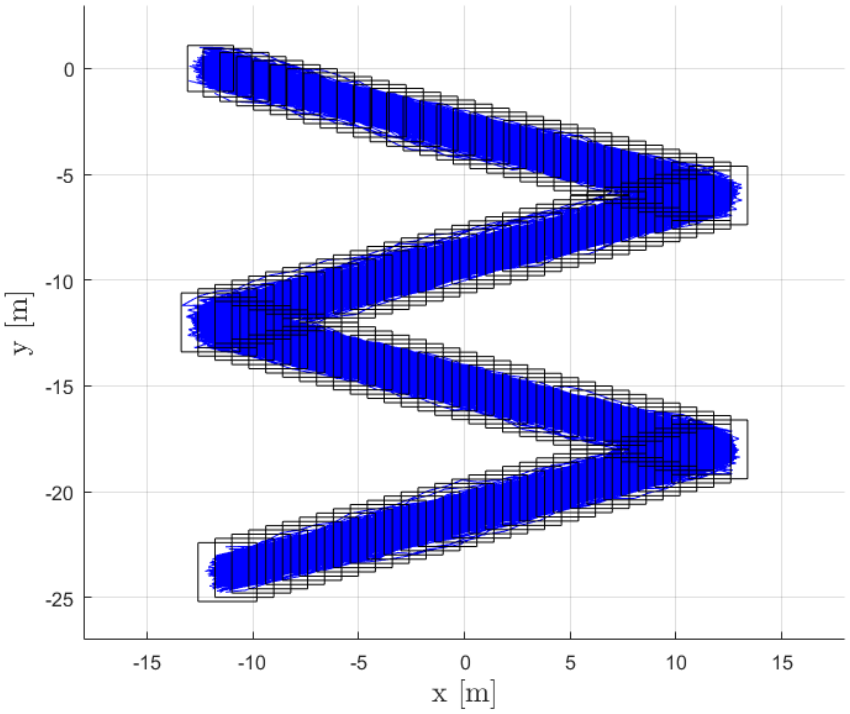
$$\check{K}_t = \texttt{dlqr}\left(A_t, B_t, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right), \tag{3.55}$$

where the identity matrices can be tuned to obtain a desired trajectory tracking performance. For the initial state estimation error covariance, we set P_0 as:

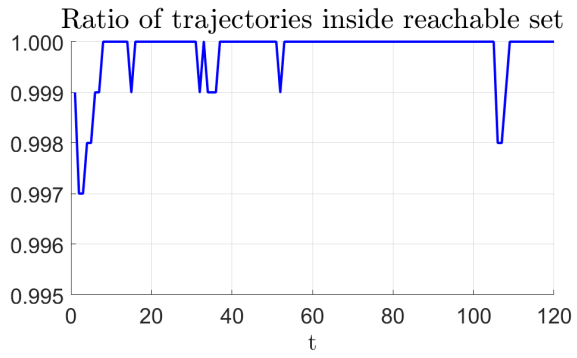
$$P_0 = \begin{bmatrix} 0.1\,\mathrm{m}^2 & 0 \\ 0 & 0.1\,\mathrm{m}^2 \end{bmatrix}. \tag{3.56}$$

Finally, we select a 'zig-zag' nominal trajectory for our simulation, with the nominal states and inputs following the robot motion model according to Equation (3.4).

Given all the aforementioned information, we first obtain the sets $\mathcal{R}_t$ as explained in Algorithm 1 and then validate these sets by simulating 1000 trajectory rollouts for the single-integrator system. Fig. 3.3 shows the results of our simulation. We observe that at least 997 of the 1000 trajectory rollouts remain inside the sets $\mathcal{R}_t \ \forall \ t \in [0, T]$, which reflects the desired 3σ confidence level. Thus, our simulation statistically shows that the requirement in Equation (3.6) is satisfied.



(a)



(b)

Figure 3.3: Reachability analysis for a 2-dimensional single-integrator system receiving positioning measurements. (a) The sets $\mathcal{R}_t$ (black) are obtained using our analysis described in Algorithm 1, which are then validated by simulating 1000 trajectory rollouts (blue). (b) The ratio of trajectories inside the sets $\mathcal{R}_t$ reflects the desired 3σ confidence level.

3.4.2 Double-integrator System

For a double-integrator system, the states consist of the robot positions and velocities whereas the control inputs consist of the robot accelerations. Thus, the motion model for the robot can be written in the form of Equation (3.1) as:

$$\mathbf{x}_t = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{t-1} + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ dt & 0 \\ 0 & dt \end{bmatrix} \mathbf{u}_{t-1} + \mathbf{w}_t, \tag{3.57}$$

where $dt = 0.2\,\mathrm{s}$ is the time-step between two time instants and the motion model error $\mathbf{w}_t$ is modelled as:

$$\mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, 0.1\,\mathrm{m^2 s^{-3}} \begin{bmatrix} \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \\ \frac{dt^2}{2} & 0 & dt & 0 \\ 0 & \frac{dt^2}{2} & 0 & dt \end{bmatrix} \right).$$

For the sensing model, we assume the availability of positioning measurements similar to the single-integrator system. Thus, sensing model can be written in the form of Equation (3.2) as:

$$\mathbf{z}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_t + \mathbf{v}_t. \tag{3.58}$$

where the error $\mathbf{v}_t$ contains a bounded bias component and a stochastic component which we define as done in Equations (3.53) and (3.54). The state feedback control gains are obtained using LQR similar to Equation (3.55) as:

$$\check{K}_t = \texttt{dlqr}\left(A_t, B_t, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right), \tag{3.59}$$

where the identity matrices can be tuned for desired trajectory tracking performance. We set the initial state estimation error covariance P_0 as:

$$P_0 = \begin{bmatrix} 0.1\,\mathrm{m}^2 & 0 & 0 & 0 \\ 0 & 0.1\,\mathrm{m}^2 & 0 & 0 \\ 0 & 0 & 0.001\,\mathrm{m}^2\mathrm{s}^{-2} & 0 \\ 0 & 0 & 0 & 0.001\,\mathrm{m}^2\mathrm{s}^{-2} \end{bmatrix}. \tag{3.60}$$

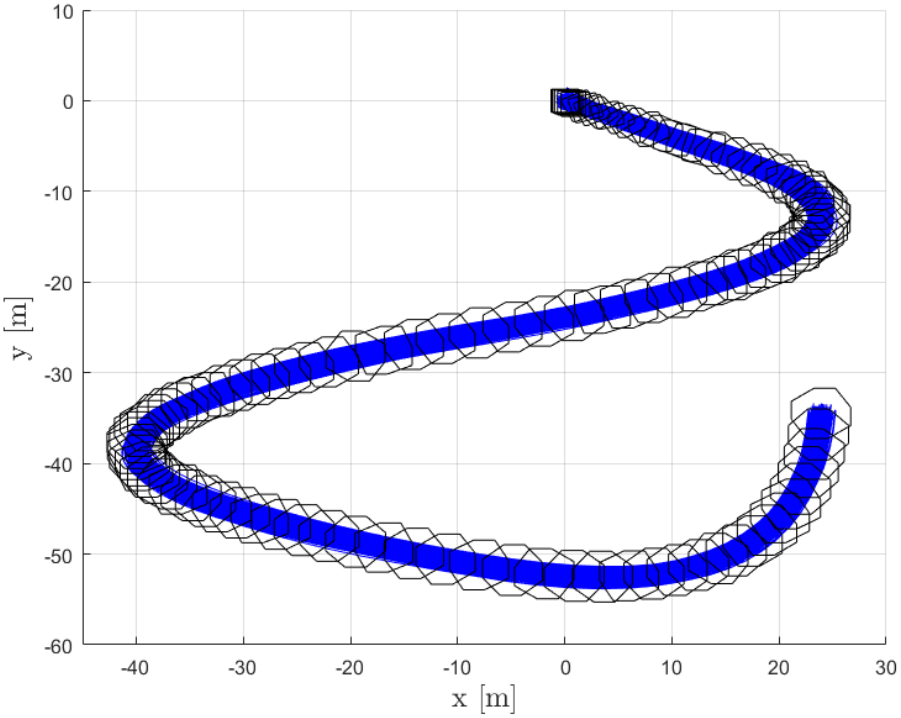
Fig. 3.4 shows the results of our reachability analysis for the double-integrator system. Here we observe that at least 999 of the 1000 trajectory rollouts remain inside the sets $\mathcal{R}_t \ \forall \ t \in [0, T]$, which reflects the desired 3σ confidence level. Thus, similar to Section 3.4.1, our simulation statistically shows that the requirement in Equation (3.6) is satisfied.

Additionally, we validate our reachability analysis in the presence of varying sensing uncertainties along the robot's trajectory. Here, for the region shaded in white we set the bounds for the bias in the positioning measurements to be $0\,\mathrm{m}$ instead of $\pm 0.5\,\mathrm{m}$ in Equation (3.53), whereas the region shaded in red is set to have a larger bias bound of $\pm 1\,\mathrm{m}$. The rest of the parameters are set similarly to the simulation in Fig. 3.4. Fig. 3.5 includes the corresponding results which again statistically show that the requirement in Equation (3.6) is satisfied.

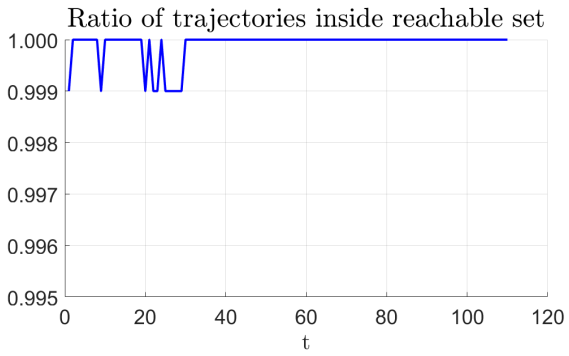
3.5 Chapter Summary

In this chapter we developed our reachability analysis for robots represented by linear motion and sensing models. We first derived equations governing the growth of the robot state and state estimation error along a nominal trajectory using the process of mathematical induction. We then transitioned the equations to set notations to compute the reachable set for the robot states. The equation to compute the reachable sets was as a function of independent quantities along the trajectory in order to satisfy the Minkowski sum definition for the probabilistic zonotope set representation. Algorithm 1 summarized our linear reachability analysis.

To validate our reachability analysis, we simulated a single-integrator and double-integrator system. For each simulation we generated 1000 trajectory rollouts and statistically showed that the requirement stated in the problem formulation was satisfied.

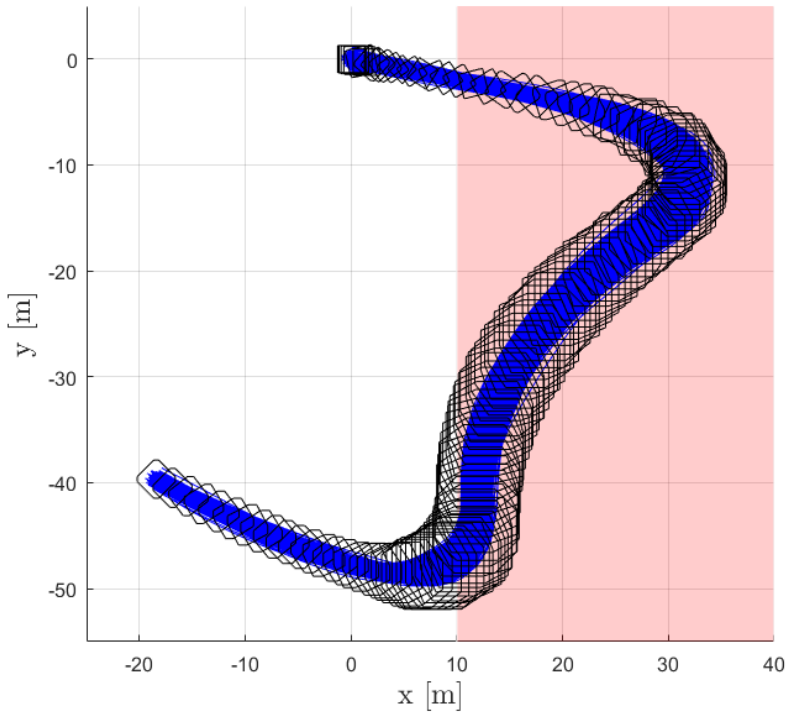


(a)

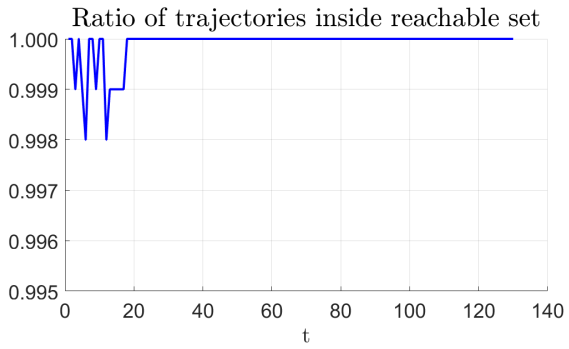


(b)

Figure 3.4: Reachability analysis for a 2-dimensional double-integrator system receiving positioning measurements. (a) The sets $\mathcal{R}_t$ (black) are obtained using our analysis described in Algorithm 1, which are then validated by simulating 1000 trajectory rollouts (blue). (b) The ratio of trajectories inside the sets $\mathcal{R}_t$ reflects the desired 3σ confidence level.



(a)



(b)

Figure 3.5: Reachability analysis in presence of varying sensing uncertainties (larger bounds for measurement bias in red shaded region) for a 2-dimensional double-integrator system receiving positioning measurements. (a) The sets $\mathcal{R}_t$ (black) are obtained using our analysis described in Algorithm 1, which are then validated by simulating 1000 trajectory rollouts (blue). (b) The ratio of trajectories inside the sets $\mathcal{R}_t$ reflects the desired 3σ confidence level.

CHAPTER 4

NON-LINEAR REACHABILITY ANALYSIS

In this chapter we extend our reachability analysis from Chapter 3 to robots that are represented by non-linear motion and sensing models. We follow a similar structure as Chapter 3. First, the non-linear motion and sensing models are defined for the robotic system along with the reachability problem formulation in Section 4.1. Next, in Section 4.2, we describe the state estimation filter used on-board including our hypothesis to account for the presence of bounded biases in the non-linear measurements. Then in Section 4.3 we develop our reachability analysis where we linearize the non-linear models and provide an approximation for the linearization errors referred to as Lagrange remainders (Section 4.4). The computed reachable sets are represented by probabilistic zonotopes that enclose all possible state distributions along the trajectory. Finally, we discuss simulations for a non-linear robotic system and statistically validate the computed reachable sets in Section 4.5.

4.1 Problem Formulation

The following discrete-time non-linear motion model is considered for our system:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathbf{w}_t, \tag{4.1}$$

where t is the time instant, $\mathbf{x}_t$ is the state vector, $\mathbf{u}_t$ is the control input vector, f is a function representing the non-linear motion, $\mathbf{w}_t$ is the motion model error represented by a Gaussian distribution with covariance Q_t, i.e., $\mathbf{w}_t$ $\mathcal{N}(0, Q_t)$. We consider non-linear measurements for the robot sensing model as:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \tag{4.2}$$

where $\mathbf{z}_t$ is the measurement vector, h is a function representing the non-linear sensing model and $\mathbf{v}_t$ is the sensing model error. Again, we account for the presence of a bounded bias along with a stochastic component in the sensing model error, i.e.:

$$\mathbf{v}_t = \mathbf{v}_t^b + \mathbf{v}_t^s, \tag{4.3}$$

where $\mathbf{v}_t^b$ is a bounded error vector represented by a zonotope $\mathcal{B}_t$, i.e., $\mathbf{v}_t^b \in [\underline{\mathbf{b}}_t, \overline{\mathbf{b}}_t] = \mathcal{B}_t$, and $\mathbf{v}_t^s$ is a stochastic error vector represented by a Gaussian distribution with covariance R_t, i.e., $\mathbf{v}_t^s \sim \mathcal{N}(\mathbf{0}, R_t)$.

Similar to Chapter 4, in this chapter we focus on computing the reachable sets for the robot along a single nominal trajectory. We assume that this trajectory is provided by a planner, the details of which are presented in Chapter 5. Additionally, the following information is assumed to be available along the nominal trajectory:

1. Initial state estimation error covariance P_0.

2. Nominal states $(\check{\mathbf{x}}_0, \check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_T)$ and nominal inputs $(\check{\mathbf{u}}_0, \check{\mathbf{u}}_1, \ldots, \check{\mathbf{u}}_{T-1})$ for the trajectory, where T represents the total number of discrete time-steps. Here the nominal states and inputs follow the non-linear motion model from Equation (4.1), i.e.:

$$\check{\mathbf{x}}_t = f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) \; \forall \; t \in [1, T]. \tag{4.4}$$

3. Stabilizing linear state feedback control gains $(\check{K}_0, \check{K}_1, \ldots, \check{K}_T)$ along the trajectory, such that the control input is of the form:

$$\mathbf{u}_t = \check{\mathbf{u}}_t - \check{K}_t(\hat{\mathbf{x}}_t - \check{\mathbf{x}}_t), \tag{4.5}$$

   where $\hat{\mathbf{x}}_t$ is the on-board state estimate that the robot obtains during trajectory execution using a extended Kalman filter as described in Section 4.2. Here the second term is the feedback control input that the robot applies in order to track the nominal trajectory.

4. Information regarding the sensing model error $\mathbf{v}_t$ along the trajectory, i.e., bounds $(\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_{T-1})$ for the bounded bias component $\mathbf{v}_t^b$ and Gaussian covariance matrices $(R_0, R_1, \ldots, R_{T-1})$ for the stochastic component $\mathbf{v}_t^s$.

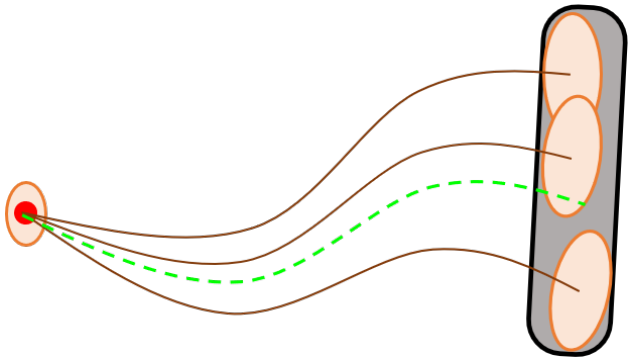


Figure 4.1: Given an initial state (red) and state distribution (orange ellipsoid), biases in sensor measurements along the nominal trajectory (green) lead to biases in future state distributions. The objective of this chapter is to develop a reachability analysis for non-linear robotic systems in order to predict bounds (black) that enclose the possible future state distributions associated with a desired confidence level.

Let $\phi(\mathbf{x}_t)$ denote the distribution of the robot state $\mathbf{x}_t$ at time instant t along the nominal trajectory. From Equations (4.1) and (4.5) we observe that the state $\mathbf{x}_t$ depends on the previous state estimate $\hat{\mathbf{x}}_{t-1}$ via the applied feedback control. The state estimate in turn depends on the set of measurements received along the trajectory. Thus, each different set of biases in the measurements $\{\mathbf{v}_0^b, \cdots, \mathbf{v}_{t-1}^b\} \in \mathcal{B}_0 \times \cdots \times \mathcal{B}_{t-1}$ results in a different set of measurement distributions (Equation (4.2)), consequently resulting in a different state distribution $\phi(\mathbf{x}_t)$.

We define the problem for our non-linear reachability analysis as follows: given a robot with non-linear motion and sensing models (Equations (4.1) and (4.2)) along with a nominal trajectory including the information specified above, compute sets $\mathcal{R}$ along the trajectory, such that:

$$\min_{\phi(\mathbf{x}_t)} \Pr(\mathbf{x}_t \in \mathcal{R}_t) > 1 - \delta, \ \forall\ t \in [1, T], \tag{4.6}$$

i.e., the probability of the robot state $\mathbf{x}_t$ belonging to the computed set $\mathcal{R}_t$ is greater than $1 - \delta$ for all possible state distributions $\phi(\mathbf{x}_t)$ throughout the nominal trajectory. Here δ is a probability value obtained from a desired confidence level. Thus, for a $m\sigma$ confidence level, where $m > 0$, δ is calculated as: $\delta = 1 - \texttt{erf}(m/\sqrt{2})$. Fig. 4.1 illustrates the defined problem.

4.2 On-board State Estimation

During trajectory execution, the true state of the robot is not available due to the presence of uncertainties in the motion and sensing models. Thus, in order to track the nominal trajectory the robot applies control inputs based on an estimate of the state, as shown in Equation (4.5). To obtain a state estimate during trajectory execution, we use an Extended Kalman Filter (EKF) on-board the robot. The prediction step of the filter is performed as [97]:

$$\bar{\mathbf{x}}_t = f(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}), \tag{4.7}$$

$$\bar{P}_t = A_{t-1} P_{t-1} A_{t-1}^\top + Q_t, \tag{4.8}$$

where $A_t = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\check{\mathbf{x}}_t}$ and Q_t is the Gaussian covariance matrix for the motion model error defined in Equation (4.1). The measurements available to the filter can be re-written from Equation (4.2) as follows:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{c}_{b_t} + \epsilon_t, \tag{4.9}$$

where $\mathbf{c}_{b_t} = (\underline{\mathbf{b}}_t + \overline{\mathbf{b}}_t)/2$ is the center of bounds $\mathcal{B}_t$ of the bias component $\mathbf{v}_t^b$, and ϵ_t is a Gaussian distribution with a bounded mean, i.e.:

$$\epsilon_t \sim \mathcal{N}(\mathbf{v}_t^b - \mathbf{c}_{b_t}, R_t), \quad \mathbf{v}_t^b \in \mathcal{B}_t. \tag{4.10}$$

For the EKF correction step, similar to Section 3.2, we choose an over-bounding hypothesis $\hat{R}_t$ as the measurement covariance matrix in order to account for the presence of the bounded bias component in the measurements. The over-bounding is performed such that $\hat{R}_t$ matches the tail of the possible Gaussian distributions representing ϵ_t in Equation (4.10) at a desired $m\sigma$ confidence level. Thus, the covariance for each measurement $z_t^{(i)}$ in Equation (3.9) is computed as:

$$\hat{R}_t^{(i,i)} = \left[\frac{\left(w_{b_t}^{(i)} + m\sqrt{R_t^{(i,i)}} \right)}{m} \right]^2, \tag{4.11}$$

where $\mathbf{w}_{b_t} = (\overline{\mathbf{b}}_t - \mathbf{c}_{b_t})$ represents the half-width vector of the bounds $\mathcal{B}_t$,

and the superscripts (i) and (i,i) refer to the i^{th} and (i,i) element of the corresponding vector and matrix respectively. We set the off-diagonal elements in $\hat{R}_t$ to 0. Note that our reachability analysis does not necessarily require choosing our over-bounding hypothesis $\hat{R}_t$ and if desired, a different hypothesis can be chosen. Once the measurement covariance matrix $\hat{R}_t$ has been computed, the EKF correction step is performed as [97]:

$$L_t = \bar{P}_t C_t^\top (C_t \bar{P}_t C_t^\top + \hat{R}_t)^{-1}, \tag{4.12}$$

$$\hat{\mathbf{x}}_t = \bar{\mathbf{x}}_t + L_t(\mathbf{z}_t - h(\bar{\mathbf{x}}_t) - \mathbf{c}_{b_t}), \tag{4.13}$$

$$P_t = \bar{P}_t - L_t C_t \bar{P}_t, \tag{4.14}$$

where L_t is the Kalman gain and $C_t = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\check{\mathbf{x}}_t}$.

4.3 Computing Reachable Sets for Nonlinear Systems

In this section, we develop our analysis to compute reachable sets for a robot with non-linear motion and sensing models, such that these sets satisfy the requirement from Equation (4.6). We follow a similar approach as done in Section 3.3 where we first derive the equations governing the growth of the robot state vector and state estimation error vector during trajectory execution, and later transition into a set notation. We derive the equations for the robot state and state estimation error vectors as a function of the initial robot state, the initial state estimation error, the motion and sensing uncertainties along the trajectory, and the linearization errors (referred to as Lagrange remainders) along the trajectory.

We begin by defining a combined state and control input vector $\mathbf{s}^\top = [\mathbf{x}^\top, \mathbf{u}^\top]$ in order to keep notations concise. Linearizing the motion model from Equation (4.1) about the nominal trajectory $\check{\mathbf{s}}_{t-1}$ we get:

$$\begin{aligned}\mathbf{x}_t = f(\check{\mathbf{s}}_{t-1}) + \mathbf{w}_t + \frac{\partial f(\mathbf{s})}{\partial \mathbf{s}}\Big|_{\mathbf{s}=\check{\mathbf{s}}_{t-1}} (\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}) + \\ \frac{1}{2}(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1})^\top \frac{\partial^2 f(\mathbf{s})}{\partial \mathbf{s}^2}\Big|_{\mathbf{s}=\check{\mathbf{s}}_{t-1}} (\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}) + \dots .\end{aligned} \tag{4.15}$$

Considering the first-order approximation of the above Taylor series, we get:

$$\mathbf{x}_t = f(\check{\mathbf{s}}_{t-1}) + \mathbf{w}_t + \frac{\partial f(\mathbf{s})}{\partial \mathbf{s}}\Big|_{\mathbf{s}=\check{\mathbf{s}}_{t-1}} (\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}) + \frac{1}{2}(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1})^T \frac{\partial^2 f(\xi)}{\partial \mathbf{s}^2}(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}), \quad (4.16)$$

where $\xi \in \{\check{\mathbf{s}}_{t-1} + \alpha(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}) \mid \alpha \in [0,1]\}$ if $\mathbf{s}_{t-1}$ is restricted to a convex set and if $\mathbf{s}_{t-1}$ and $\check{\mathbf{s}}_{t-1}$ are fixed [36, 101]. Here the last term of Equation (4.16) is the vector of linearization errors, referred to as Lagrange remainders. The remainder vector arises from linearizing the non-linear motion model function f from Equation (4.1) w.r.t. $\mathbf{s}_{t-1}$ about $\check{\mathbf{s}}_{t-1}$. We represent the Lagrange remainder concisely as:

$$\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} = \frac{1}{2}(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1})^T \frac{\partial^2 f(\xi)}{\partial \mathbf{s}^2}(\mathbf{s}_{t-1} - \check{\mathbf{s}}_{t-1}). \quad (4.17)$$

Splitting the combined vector $\mathbf{s}_{t-1}$ into $\mathbf{x}_{t-1}$ and $\mathbf{u}_{t-1}$, Equation (4.16) can be written as:

$$\mathbf{x}_t = A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\mathbf{u}_{t-1} - \check{\mathbf{u}}_{t-1}) + f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t, \quad (4.18)$$

where $A_t = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\check{\mathbf{x}}_t}$ as mentioned in Equation (4.8) and $B_t = \frac{\partial f}{\partial \mathbf{u}}\Big|_{\mathbf{u}=\check{\mathbf{u}}_t}$. Substituting the nominal state from Equation (4.4) and the total control input from Equation (4.5), we get:

$$\begin{aligned} \mathbf{x}_t &= A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\check{\mathbf{u}}_{t-1} - \check{K}_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) - \check{\mathbf{u}}_{t-1}) + \\ &\quad \check{\mathbf{x}}_t + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t \quad (4.19) \\ &= A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + \check{\mathbf{x}}_t + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t. \quad (4.20) \end{aligned}$$

Recall that the state estimation error $\tilde{\mathbf{x}}_t$ is defined as the difference between the estimated state $\hat{\mathbf{x}}_t$ and the true state $\mathbf{x}_t$, i.e., $\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t$. Thus, we

re-write Equation (4.20) as:

$$\begin{aligned}\mathbf{x}_t &= A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\mathbf{x}_{t-1} + \tilde{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + \check{\mathbf{x}}_t + \\ &\quad \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t \qquad (4.21)\\ &= A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}\tilde{\mathbf{x}}_{t-1} + \\ &\quad \check{\mathbf{x}}_t + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t \qquad (4.22)\\ &= \check{\mathbf{x}}_t + (A_{t-1} - B_{t-1}\check{K}_{t-1})(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}\check{K}_{t-1}\tilde{\mathbf{x}}_{t-1} + \\ &\quad \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t. \qquad (4.23)\end{aligned}$$

In order to obtain the state estimation error $\tilde{\mathbf{x}}_{t-1}$ required in Equation (4.23), we begin with state estimation filter equations in Section 4.2. Using the EKF correction step from Equation (4.13) we get:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= \hat{\mathbf{x}}_t - \mathbf{x}_t \\ &= \bar{\mathbf{x}}_t + L_t(\mathbf{z}_t - h(\bar{\mathbf{x}}_t) - \mathbf{c}_{b_t}) - \mathbf{x}_t. \qquad (4.24)\end{aligned}$$

Replacing the measurement vector $\mathbf{z}_t$ from Equation (4.9):

$$\tilde{\mathbf{x}}_t = \bar{\mathbf{x}}_t + L_t(h(\mathbf{x}_t) + \epsilon_t - h(\bar{\mathbf{x}}_t)) - \mathbf{x}_t. \qquad (4.25)$$

The non-linear sensing model function h can be linearized w.r.t. the true state $\mathbf{x}_t$ and the predicted state $\bar{\mathbf{x}}_t$ about the nominal state $\check{\mathbf{x}}_t$, to obtain the following:

$$h(\mathbf{x}_t) = h(\check{\mathbf{x}}_t) + C_t(\mathbf{x}_t - \check{\mathbf{x}}_t) + \mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t}, \qquad (4.26)$$

$$h(\bar{\mathbf{x}}_t) = h(\check{\mathbf{x}}_t) + C_t(\bar{\mathbf{x}}_t - \check{\mathbf{x}}_t) + \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}, \qquad (4.27)$$

where $C_t = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\check{\mathbf{x}}_t}$ as mentioned in Equation (4.12), and $\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t}$ and $\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}$ are the corresponding Lagrange remainder vectors that follow the notations defined in Equation (4.17). Replacing Equations (4.26) and (4.27) in Equa-

tion (4.25), we get the following expression for the state estimation error:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= \bar{\mathbf{x}}_t + L_t\Big(h(\check{\mathbf{x}}_t) + C_t(\mathbf{x}_t - \check{\mathbf{x}}_t) + \mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} + \epsilon_t - h(\check{\mathbf{x}}_t) - \\ &\quad C_t(\bar{\mathbf{x}}_t - \check{\mathbf{x}}_t) - \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}\Big) - \mathbf{x}_t, \qquad (4.28)\\ &= \bar{\mathbf{x}}_t + L_t(C_t(\mathbf{x}_t - \bar{\mathbf{x}}_t) + \mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} - \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t} + \epsilon_t) - \mathbf{x}_t, \qquad (4.29)\\ &= (\bar{\mathbf{x}}_t - \mathbf{x}_t) - L_tC_t(\bar{\mathbf{x}}_t - \mathbf{x}_t) + L_t(\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} - \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}) + L_t\epsilon_t, \qquad (4.30)\\ &= (I - L_tC_t)(\bar{\mathbf{x}}_t - \mathbf{x}_t) + L_t(\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} - \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}) + L_t\epsilon_t, \qquad (4.31)\end{aligned}$$

where I is an identity matrix of appropriate dimensions. Here $\bar{\mathbf{x}}_t$ represents the predicted state within the EKF from Equation (4.7). On linearizing Equation (4.7) about the nominal state $\check{\mathbf{x}}_t$ we obtain:

$$\begin{aligned}\bar{\mathbf{x}}_t &= f(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}),\\ &= A_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\mathbf{u}_{t-1} - \check{\mathbf{u}}_{t-1}) + f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) + \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}}.\end{aligned} \qquad (4.32)$$

where $\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}}$ is the Lagrange remainder following the notations defined in Equation (4.17).

Thus, subtracting the true state $\mathbf{x}_t$ (Equation (4.18)) from the predicted state $\bar{\mathbf{x}}_t$ (Equation (4.32)), we get the error in the predicted state as:

$$\begin{aligned}\bar{\mathbf{x}}_t - \mathbf{x}_t &= A_{t-1}(\hat{\mathbf{x}}_{t-1} - \check{\mathbf{x}}_{t-1}) + B_{t-1}(\mathbf{u}_{t-1} - \check{\mathbf{u}}_{t-1}) + f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) + \\ &\quad \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}} - A_{t-1}(\mathbf{x}_{t-1} - \check{\mathbf{x}}_{t-1}) - B_{t-1}(\mathbf{u}_{t-1} - \check{\mathbf{u}}_{t-1}) - \qquad (4.33)\\ &\quad f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) - \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} - \mathbf{w}_t,\\ &= A_{t-1}(\hat{\mathbf{x}}_{t-1} - \mathbf{x}_{t-1}) + \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}} - \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} - \mathbf{w}_t \qquad (4.34)\\ &= A_{t-1}\tilde{\mathbf{x}}_{t-1} + \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}} - \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} - \mathbf{w}_t. \qquad (4.35)\end{aligned}$$

On substituting the term $(\bar{\mathbf{x}}_t - \mathbf{x}_t)$ from Equation (4.35) in Equation (4.31) we obtain the following equation for the state estimation error:

$$\begin{aligned}\tilde{\mathbf{x}}_t &= (I - L_tC_t)A_{t-1}\tilde{\mathbf{x}}_{t-1} + (I - L_tC_t)(\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}} - \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}}) + \\ &\quad L_t(\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} - \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}) - (I - L_tC_t)\mathbf{w}_t + L_t\epsilon_t.\end{aligned} \qquad (4.36)$$

While Equations (4.23) and (4.36) govern the growth of the state and the state estimation errors respectively, they involve the summation of correlated

quantities similar to Equations (3.32) and (3.33) for the linear system. Thus, in order to transition to set notations, we first derive the growth of the state and the state estimation error as a function of independent quantities. We use Equations (4.23) and (4.36) along with the process of mathematical induction to derive the required equations.

For our non-linear system, we propose that the following expressions of $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ hold true for any time instant t:

$$\begin{aligned}\mathbf{x}_t = \check{\mathbf{x}}_t + {}^1\Phi_t(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^2\Phi_t\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t} {}^3\Phi_t^n\mathbf{w}_n + \sum_{n=1}^{t} {}^4\Phi_t^n\epsilon_n + \\ \sum_{n=0}^{t-1} {}^5\Phi_t^n\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-1} {}^6\Phi_t^n\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-1} {}^7\Phi_t^n\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + \sum_{n=0}^{t-1} {}^8\Phi_t^n\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n},\end{aligned} \tag{4.37}$$

$$\begin{aligned}\tilde{\mathbf{x}}_t = {}^2\tilde{\Phi}_t\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t} {}^3\tilde{\Phi}_t^n\mathbf{w}_n + \sum_{n=1}^{t} {}^4\tilde{\Phi}_t^n\epsilon_n + \sum_{n=0}^{t-1} {}^5\tilde{\Phi}_t^n\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \\ \sum_{n=0}^{t-1} {}^6\tilde{\Phi}_t^n\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-1} {}^7\tilde{\Phi}_t^n\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + \sum_{n=0}^{t-1} {}^8\tilde{\Phi}_t^n\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n},\end{aligned} \tag{4.38}$$

where all Φ_t and $\tilde{\Phi}_t$ are matrix coefficients derived from system models and state estimation matrices defined in Sections 4.1 and 4.2. These matrix coefficients can be interpreted as weights that determine how much the corresponding quantities, i.e., $\mathbf{x}_0$, $\tilde{\mathbf{x}}_0$, $\mathbf{w}_n$, ϵ_n, $\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n}$, $\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n}$, $\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n}$ and $\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}$ influence the state $\mathbf{x}_t$ and state estimation error $\tilde{\mathbf{x}}_t$ at time instant t. Here $\mathbf{x}_0$ is the initial state of the system, $\tilde{\mathbf{x}_0}$ is the initial state estimation error, $\mathbf{w}_n$ and ϵ_n are motion and sensing model errors independently sampled along the trajectory, and $\mathcal{L}$ are the corresponding Lagrange remainders.

For the induction base case, we show that Equations (4.37) and (4.38) hold true for $t = 0$. The initial state $\mathbf{x}_0$ and state estimation error $\tilde{\mathbf{x}}_0$ can be written in the form of Equations (4.37) and (4.38) as:

$$\begin{aligned}\mathbf{x}_0 &= \check{\mathbf{x}}_0 + (\mathbf{x}_0 - \check{\mathbf{x}}_0), \\ \tilde{\mathbf{x}}_0 &= \tilde{\mathbf{x}}_0,\end{aligned} \tag{4.39}$$

which can be obtained by setting ${}^1\Phi_0 = I$, ${}^i\Phi_0 = O \;\forall\; i = [2, 8]$ in Equation (4.37), and ${}^2\tilde{\Phi}_0 = I$, ${}^i\tilde{\Phi}_0 = O \;\forall\; i = [3, 8]$ in Equation (4.38). Here O is a zero matrix of appropriate dimensions.

For the induction step, we begin by assuming that Equations (4.37) and (4.38) hold true for time instant $t-1$, i.e.:

$$\mathbf{x}_{t-1} = \check{\mathbf{x}}_{t-1} + {}^{1}\Phi_t(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^{2}\Phi_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^{3}\Phi_{t-1}^{n}\mathbf{w}_n + \sum_{n=1}^{t-1} {}^{4}\Phi_{t-1}^{n}\epsilon_n + \sum_{n=0}^{t-2} {}^{5}\Phi_{t-1}^{n}\mathcal{L}_{[\mathbf{s},\check{\mathbf{s}}]_n}^{f} + \sum_{n=0}^{t-2} {}^{6}\Phi_{t-1}^{n}\mathcal{L}_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n}^{f} + \sum_{n=0}^{t-2} {}^{7}\Phi_{t-1}^{n}\mathcal{L}_{[\mathbf{x},\check{\mathbf{x}}]_n}^{h} + \sum_{n=0}^{t-2} {}^{8}\Phi_{t-1}^{n}\mathcal{L}_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}^{h}, \tag{4.40}$$

$$\tilde{\mathbf{x}}_{t-1} = {}^{2}\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^{3}\tilde{\Phi}_{t-1}^{n}\mathbf{w}_n + \sum_{n=1}^{t-1} {}^{4}\tilde{\Phi}_{t-1}^{n}\epsilon_n + \sum_{n=0}^{t-2} {}^{5}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\mathbf{s},\check{\mathbf{s}}]_n}^{f} + \sum_{n=0}^{t-2} {}^{6}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n}^{f} + \sum_{n=0}^{t-2} {}^{7}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\mathbf{x},\check{\mathbf{x}}]_n}^{h} + \sum_{n=0}^{t-2} {}^{8}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}^{h}, \tag{4.41}$$

Next, we show that Equations (4.37) and (4.38) hold true for time instant t. We begin by replacing Equation (4.41) in Equation (4.36). This gives us the following expression for the state estimation error $\tilde{\mathbf{x}}_t$:

$$\begin{aligned}
\tilde{\mathbf{x}}_t = {} & (I - L_tC_t)A_{t-1}\,{}^{2}\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + (I - L_tC_t)A_{t-1}\sum_{n=1}^{t-1} {}^{3}\tilde{\Phi}_{t-1}^{n}\mathbf{w}_n - \\
& (I - L_tC_t)\mathbf{w}_t + (I - L_tC_t)A_{t-1}\sum_{n=1}^{t-1} {}^{4}\tilde{\Phi}_{t-1}^{n}\epsilon_n + L_t\epsilon_t + \\
& (I - L_tC_t)A_{t-1}\sum_{n=0}^{t-2} {}^{5}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\mathbf{s},\check{\mathbf{s}}]_n}^{f} - (I - L_tC_t)\mathcal{L}_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}}^{f} + \\
& (I - L_tC_t)A_{t-1}\sum_{n=0}^{t-2} {}^{6}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n}^{f} + (I - L_tC_t)\mathcal{L}_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}}^{f} + \\
& (I - L_tC_t)A_{t-1}\sum_{n=0}^{t-2} {}^{7}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\mathbf{x},\check{\mathbf{x}}]_n}^{h} + L_t\mathcal{L}_{[\mathbf{x},\check{\mathbf{x}}]_t}^{h} + \\
& (I - L_tC_t)A_{t-1}\sum_{n=0}^{t-2} {}^{8}\tilde{\Phi}_{t-1}^{n}\mathcal{L}_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}^{h} - L_t\mathcal{L}_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}^{h},
\end{aligned} \tag{4.42}$$

(4.43)

$$
\begin{aligned}
\tilde{\mathbf{x}}_t = &\Big((I - L_t C_t) {A_{t-1}}^2 \tilde{\Phi}_{t-1}\Big)\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1}\Big((I - L_t C_t) {A_{t-1}}^3 \tilde{\Phi}^n_{t-1}\Big)\mathbf{w}_n + \\
&\Big(-(I - L_t C_t)\Big)\mathbf{w}_t + \sum_{n=1}^{t-1}\Big((I - L_t C_t) {A_{t-1}}^4 \tilde{\Phi}^n_{t-1}\Big)\epsilon_n + L_t\epsilon_t + \\
&\sum_{n=0}^{t-2}\Big((I - L_t C_t) {A_{t-1}}^5 \tilde{\Phi}^n_{t-1}\Big)\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \Big(-(I - L_t C_t)\Big)\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \\
&\sum_{n=0}^{t-2}\Big((I - L_t C_t) {A_{t-1}}^6 \tilde{\Phi}^n_{t-1}\Big)\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \Big((I - L_t C_t)\Big)\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_{t-1}} + \\
&\sum_{n=0}^{t-2}\Big((I - L_t C_t) {A_{t-1}}^7 \tilde{\Phi}^n_{t-1}\Big)\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + L_t\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_t} + \\
&\sum_{n=0}^{t-2}\Big((I - L_t C_t) {A_{t-1}}^8 \tilde{\Phi}^n_{t-1}\Big)\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n} + \Big(-L_t\Big)\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_t}.
\end{aligned}
\tag{4.44}
$$

The above equation can then be written in the form of Equation (4.38) where the $\tilde{\Phi}_t$ matrix coefficients can be obtained as:

$$
\begin{aligned}
{}^2\tilde{\Phi}_t &= (I - L_t C_t) {A_{t-1}}^2 \tilde{\Phi}_{t-1}, \\
{}^3\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^3 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [1, t-1], \\
{}^3\tilde{\Phi}^t_t &= -(I - L_t C_t), \\
{}^4\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^4 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [1, t-1], \\
{}^4\tilde{\Phi}^t_t &= L_t, \\
{}^5\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^5 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [0, t-2], \\
{}^5\tilde{\Phi}^{t-1}_t &= -(I - L_t C_t), \\
{}^6\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^6 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [0, t-2], \\
{}^6\tilde{\Phi}^{t-1}_t &= (I - L_t C_t), \\
{}^7\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^7 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [0, t-2], \\
{}^7\tilde{\Phi}^{t-1}_t &= L_t, \\
{}^8\tilde{\Phi}^n_t &= (I - L_t C_t) {A_{t-1}}^8 \tilde{\Phi}^n_{t-1} \ \forall\ n \in [0, t-2], \\
{}^8\tilde{\Phi}^{t-1}_t &= -L_t.
\end{aligned}
\tag{4.45}
$$

Similarly for state vector $\mathbf{x}_t$, we replace Equations (4.40) and (4.41) in Equa-

tion (4.23), to get the following expression:

$$\begin{aligned}\mathbf{x}_t = \check{\mathbf{x}}_t + (A_{t-1} - B_{t-1}\check{K}_{t-1})\Big({}^1\Phi_t(\mathbf{x}_0 - \check{\mathbf{x}}_0) + {}^2\Phi_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^3\Phi_{t-1}^n \mathbf{w}_n + \\ \sum_{n=1}^{t-1} {}^4\Phi_{t-1}^n \epsilon_n + \sum_{n=0}^{t-2} {}^5\Phi_{t-1}^n \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-2} {}^6\Phi_{t-1}^n \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-2} {}^7\Phi_{t-1}^n \mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + \\ \sum_{n=0}^{t-2} {}^8\Phi_{t-1}^n \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}\Big) - B_{t-1}\check{K}_{t-1}\Big({}^2\tilde{\Phi}_{t-1}\tilde{\mathbf{x}}_0 + \sum_{n=1}^{t-1} {}^3\tilde{\Phi}_{t-1}^n \mathbf{w}_n + \sum_{n=1}^{t-1} {}^4\tilde{\Phi}_{t-1}^n \epsilon_n + \\ \sum_{n=0}^{t-2} {}^5\tilde{\Phi}_{t-1}^n \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-2} {}^6\tilde{\Phi}_{t-1}^n \mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \sum_{n=0}^{t-2} {}^7\tilde{\Phi}_{t-1}^n \mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + \\ \sum_{n=0}^{t-2} {}^8\tilde{\Phi}_{t-1}^n \mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}\Big) + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \mathbf{w}_t.\end{aligned} \tag{4.46}$$

$$\begin{aligned}\mathbf{x}_t = \check{\mathbf{x}}_t + (A_{t-1} - B_{t-1}\check{K}_{t-1})^1\Phi_{t-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) + \\ \Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^2\Phi_{t-1} - B_{t-1}\check{K}_{t-1}{}^2\tilde{\Phi}_{t-1}\Big)\tilde{\mathbf{x}}_0 + \\ \sum_{n=1}^{t-1}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^3\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^3\tilde{\Phi}_{t-1}^n\Big)\mathbf{w}_n + \mathbf{w}_t + \\ \sum_{n=1}^{t-1}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^4\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^4\tilde{\Phi}_{t-1}^n\Big)\epsilon_n + \\ \sum_{n=0}^{t-2}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^5\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^5\tilde{\Phi}_{t-1}^n\Big)\mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} + \mathcal{L}^f_{[\mathbf{s},\check{\mathbf{s}}]_{t-1}} + \\ \sum_{n=0}^{t-2}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^6\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^6\tilde{\Phi}_{t-1}^n\Big)\mathcal{L}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} + \\ \sum_{n=0}^{t-2}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^6\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^6\tilde{\Phi}_{t-1}^n\Big)\mathcal{L}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} + \\ \sum_{n=0}^{t-2}\Big((A_{t-1} - B_{t-1}\check{K}_{t-1})^7\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^7\tilde{\Phi}_{t-1}^n\Big)\mathcal{L}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}.\end{aligned} \tag{4.47}$$

The above equation can then be written in the form of Equation (4.37) where

the Φ_t matrix coefficients can be obtained as:

$$
\begin{aligned}
{}^{1}\Phi_t &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{1}\Phi_{t-1}, \\
{}^{2}\Phi_t &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{2}\Phi_{t-1} - B_{t-1}\check{K}_{t-1}{}^{2}\tilde{\Phi}_{t-1}, \\
{}^{3}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{3}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{3}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [1, t-1], \\
{}^{3}\Phi_t^t &= I, \\
{}^{4}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{4}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{4}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [1, t-1], \\
{}^{4}\Phi_t^t &= O, \\
{}^{5}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{5}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{5}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [0, t-2], \\
{}^{5}\Phi_t^{t-1} &= I, \\
{}^{6}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{6}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{6}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [0, t-2], \\
{}^{6}\Phi_t^{t-1} &= O, \\
{}^{7}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{7}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{7}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [0, t-2], \\
{}^{7}\Phi_t^{t-1} &= O, \\
{}^{8}\Phi_t^n &= (A_{t-1} - B_{t-1}\check{K}_{t-1})^{8}\Phi_{t-1}^n - B_{t-1}\check{K}_{t-1}{}^{8}\tilde{\Phi}_{t-1}^n \;\forall\; n \in [0, t-2], \\
{}^{8}\Phi_t^{t-1} &= O.
\end{aligned}
\tag{4.48}
$$

Thus, by the principle of induction, we can claim that Equations (4.37) and (4.38) hold true for all time instants along the trajectory, i.e., $\forall\, t \in [0, T]$. The matrix coefficients Φ_t and $\tilde{\Phi}_t$ required in these equations can be initialized as shown in Equation (4.39) and can be obtained recursively along the trajectory using Equations (4.45) and (4.48).

While Equations (4.37) and (4.38) govern the growth of the robot state $\mathbf{x}_t$ and state estimation error $\tilde{\mathbf{x}}_t$, their exact values cannot be computed since the exact values are not known for the initial robot state $\mathbf{x}_0$, the initial state estimation error $\tilde{\mathbf{x}}_0$, the motion and sensing model errors $\mathbf{w}_t$ and ϵ_t along the trajectory and the Lagrange remainders $\mathcal{L}$ along the trajectory. Thus, similar to the linear reachability analysis in Section 3.3, we instead transition Equations (4.37) and (4.38) to set notations. The set of initial robot states $\mathcal{X}_0$, the initial state estimation error set $\tilde{\mathcal{X}}_0$ and the sets of motion and sensing uncertainties $\mathcal{W}_t$ and $\mathcal{V}_t$ can be obtained using information available in Section 4.1. However, modeling the Lagrange remainders $\mathcal{L}$ is not trivial, and thus we develop a method to approximate the remainders with a Gaussian distribution $\widehat{\mathcal{L}}$ as explained later in Section 4.4. The reachable set $\mathcal{X}_t$ and

the state estimation error set $\tilde{\mathcal{X}}_t$ at any time instant t are computed as:

$$\begin{aligned}\mathcal{X}_t = \check{\mathbf{x}}_t \oplus {}^1\Phi_t(\mathcal{X}_0 - \check{\mathbf{x}}_0) \oplus {}^2\Phi_t\tilde{\mathcal{X}}_0 \oplus \sum_{n=1}^{t} {}^3\Phi_t^n\mathcal{W}_n \oplus \sum_{n=1}^{t} {}^4\Phi_t^n\mathcal{V}_n \oplus \\ \sum_{n=0}^{t-1} {}^5\Phi_t^n\widehat{\mathcal{L}}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} \oplus \sum_{n=0}^{t-1} {}^6\Phi_t^n\widehat{\mathcal{L}}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} \oplus \sum_{n=0}^{t-1} {}^7\Phi_t^n\widehat{\mathcal{L}}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} \oplus \sum_{n=0}^{t-1} {}^8\Phi_t^n\widehat{\mathcal{L}}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n},\end{aligned} \tag{4.49}$$

$$\begin{aligned}\tilde{\mathcal{X}}_t = {}^2\tilde{\Phi}_t\tilde{\mathcal{X}}_0 \oplus \sum_{n=1}^{t} {}^3\tilde{\Phi}_t^n\mathcal{W}_n \oplus \sum_{n=1}^{t} {}^4\tilde{\Phi}_t^n\mathcal{V}_n \oplus \sum_{n=0}^{t-1} {}^5\tilde{\Phi}_t^n\widehat{\mathcal{L}}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} \oplus \\ \sum_{n=0}^{t-1} {}^6\tilde{\Phi}_t^n\widehat{\mathcal{L}}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} \oplus \sum_{n=0}^{t-1} {}^7\tilde{\Phi}_t^n\widehat{\mathcal{L}}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} \oplus \sum_{n=0}^{t-1} {}^8\tilde{\Phi}_t^n\widehat{\mathcal{L}}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}.\end{aligned} \tag{4.50}$$

where $\oplus$ is the Minkowski sum operation described in Section 2.1.2. As discussed in Section 2.1, we use the probabilistic zonotope set representation for our reachability analysis since they allow us to account for both the bounded and the stochastic components of uncertainties in the system. The sets $\mathcal{X}_0$, $\tilde{\mathcal{X}}_0$, $\mathcal{W}_t$ and $\mathcal{V}_t$ are defined similar to Equations (3.47), (3.48) and (3.49) in Section 3.3 as:

$$\begin{aligned}\mathcal{X}_0 &= \mathscr{Z}(\check{\mathbf{x}}_0, \mathbf{0}, P_0), \\ \tilde{\mathcal{X}}_0 &= \mathscr{Z}(\mathbf{0}, \mathbf{0}, P_0), \\ \mathcal{W}_t &= \mathscr{Z}(\mathbf{0}, \mathbf{0}, Q_t), \\ \mathcal{V}_t &= \mathscr{Z}(\mathbf{0}, \mathtt{diag}(\mathbf{w}_{b_t}), R_t),\end{aligned} \tag{4.51}$$

where P_0 is the initial state estimation error covariance, Q_t is the Gaussian covariance matrix for the motion model error, R_t is the Gaussian covariance matrix for the stochastic component in the sensing model error and $\mathbf{w}_{b_t}$ is the half-width of the bounds $\mathcal{B}_t$ defined in Equation (4.11).

Finally, in order to obtain the sets $\mathcal{R}_t$ required in the problem formulation in Equation (4.6), we use the confidence set operation on the computed reachable sets:

$$\mathcal{R}_t = \mathtt{conf}(\mathcal{X}_t, m), \tag{4.52}$$

where m represents the desired confidence level, such as 3σ ($m = 3$) confidence, as specified in Section 4.1. Algorithm 2 summarizes the inputs to our non-linear reachability analysis followed by the process to obtain the sets $\mathcal{R}_t$ along the nominal trajectory.

Algorithm 2 Non-linear Reachability Analysis

0: **Inputs**: Robot motion and sensing model (Equations (4.1) and (4.2)); initial state estimation error covariance P_0; nominal states $(\check{\mathbf{x}}_0, \ldots, \check{\mathbf{x}}_T)$ and nominal inputs $(\check{\mathbf{u}}_0, \ldots, \check{\mathbf{u}}_{T-1})$; measurement bias bounds $(\check{\mathcal{B}}_0, \ldots, \check{\mathcal{B}}_{T-1})$ and measurement Gaussian covariance matrices $(R_0, \ldots, R_{T-1})$; desired confidence level m.
1: Obtain initial robot states $\mathcal{X}_0$ and initial state estimation error set $\tilde{\mathcal{X}}_0$ (Equation (3.47)), motion uncertainty sets $(\check{\mathcal{W}}_0, \ldots, \check{\mathcal{W}}_{T-1})$ (Equation (3.48)), and sensing uncertainty sets $(\check{\mathcal{V}}_0, \ldots, \check{\mathcal{V}}_{T-1})$ (Equation (3.49)).
2: Initialize matrix coefficients Φ_0 and $\tilde{\Phi}_0$ (Equation (4.39)).
3: **for** $t = 1, \ldots, T$ **do**
4: Update matrix coefficients $\tilde{\Phi}_t$ (Equation (4.45)).
5: Update matrix coefficients Φ_t (Equation (4.48)).
6: Obtain Lagrange remainder approximations $\widehat{\mathcal{L}}$ (Equation (4.58)).
7: Compute reachable set $\mathcal{X}_t$ (Equation (4.49)).
8: Obtain confidence sets $\mathcal{R}_t$ (Equation (4.52)).
9: **end for**

4.4 Lagrange Remainder Approximation

The Lagrange remainders in our non-linear reachability analysis arise due to linearization of the motion and sensing models in Equations (4.15), (4.26), (4.27) and (4.32). In this section, we explain the steps that we take for approximating a general Lagrange remainder vector of the form $\mathcal{L}^{q}_{[\mathbf{p},\check{\mathbf{p}}]}$ resulting from linearizing a function q w.r.t. $\mathbf{p}$ about a point $\check{\mathbf{p}}$. The i^{th} element of the Lagrange remainder vector is defined as:

$$\begin{aligned} \mathcal{L}^{q_{(i)}}_{[\mathbf{p},\check{\mathbf{p}}]} &= \frac{1}{2}(\mathbf{p}-\check{\mathbf{p}})^\top J^{q_{(i)}}_{\mathbf{p}}(\xi)(\mathbf{p}-\check{\mathbf{p}}), \\ \xi &\in \{\check{\mathbf{p}} + \alpha(\mathbf{p}-\check{\mathbf{p}}) |\ \alpha \in [0,1]\}, \end{aligned} \tag{4.53}$$

where $J^{q_{(i)}}_{\mathbf{p}}(\xi) = \frac{\partial^2 q_{(i)}(\xi)}{\partial \mathbf{p}^2}$. In our reachability analysis the vector $\mathbf{p}$ belongs to a probabilistic zonotope, thus resulting in a stochastic distribution for the Lagrange remainder $\mathcal{L}^{q_{(i)}}_{[\mathbf{p},\check{\mathbf{p}}]}$. Based on Equation (4.53), this distribution for the remainder is not trivial to compute. Thus, we approximate the Lagrange remainder as a Gaussian distribution by following the steps listed below:

1. Let $\mathscr{P}$ denote the probabilistic zonotope that $\mathbf{p}$ belongs to. We first generate a $m\sigma$ projection zonotope $\mathcal{P}^{m\sigma}$ for $\mathbf{p}$. Here m denotes the desired confidence level specified from Equation (4.6) in the problem

formulation. The generators of the projection zonotope $\mathcal{P}^{m\sigma}$ are obtained using the projection operation from Section 2.1.5 along each dimension of $\mathscr{P}$ as:

$$\bar{p}^{(i)} = \texttt{proj}(\mathscr{P}, \mathbf{e}_i, m), \tag{4.54}$$

where $\mathbf{e}_i$ represents a unit column vector with the i^{th} element set to 1 and the remaining elements set to 0. The projection zonotope $\mathcal{P}^{m\sigma}$ is then obtained as:

$$\mathcal{P}^{m\sigma} = \mathcal{Z}(c_{\mathscr{P}}, \texttt{diag}(\bar{\mathbf{p}})), \tag{4.55}$$

where $c_{\mathscr{P}}$ is the center of the probabilistic zonotope $\mathscr{P}$.

2. Using the approach presented in [36], we calculate the maximum value of each element of the Lagrange remainder vector when $\mathbf{p}$ belongs to $\mathcal{P}^{m\sigma}$. The maximum value for the i^{th} element is calculated as:

$$\bar{\mathcal{L}}^{q_{(i)}}_{[\mathbf{p},\check{\mathbf{p}}]^{m\sigma}} = \frac{1}{2}\gamma^{\top} \max(|J^{q_{(i)}}_{\mathbf{p}}(\xi(\mathbf{p}))|)\gamma, \tag{4.56}$$

where $\gamma = |\mathbf{c}_{\mathscr{P}} - \check{\mathbf{p}}| + |\bar{\mathbf{p}}|$. The expression $\max(|J^{q_{(i)}}_{\mathbf{p}}(\xi(\mathbf{p}))|)$ is obtained by first deriving the matrix $J^{q_{(i)}}_{\mathbf{p}}(\xi(\mathbf{p}))$ for the system and then finding the maximum of the absolute value of each element in the matrix, assuming $\mathbf{p} \in \mathscr{P}$.

3. The Gaussian covariance matrix for the Lagrange remainder approximation is then chosen such that it reflects the confidence level $m\sigma$ of the maximum values $\bar{\mathcal{L}}^{q}_{[\mathbf{p},\check{\mathbf{p}}]^{m\sigma}}$ from Equation (4.56). Thus, the covariance of the i^{th} element $\Sigma_{\mathcal{L}^{q_{(i)}}_{[p,\check{p}]}}$ is determined as:

$$\Sigma_{\mathcal{L}^{q_{(i)}}_{[\mathbf{p},\check{\mathbf{p}}]}} = \left(\frac{\bar{\mathcal{L}}^{q_{(i)}}_{[\mathbf{p},\check{\mathbf{p}}]^{m\sigma}}}{m}\right)^2 \tag{4.57}$$

Finally, the approximations of the Lagrange remainders required in Equations (4.49) and (4.50) are obtained as:

$$\widehat{\mathcal{L}}^{q}_{[\mathbf{p},\check{\mathbf{p}}]} = \mathscr{Z}\left(\mathbf{0}, \mathbf{0}, \texttt{diag}\left(\left[\Sigma_{\mathcal{L}^{q_{(1)}}_{[\mathbf{p},\check{\mathbf{p}}]}}; \cdots; \Sigma_{\mathcal{L}^{q_{(n)}}_{[\mathbf{p},\check{\mathbf{p}}]}}\right]\right)\right), \tag{4.58}$$

where n is the number of dimensions of $\mathbf{p}$.

4.5 Simulation Results

In this section, we discuss our simulations in order to validate our non-linear reachability analysis. We simulate a 2-dimensional Dubins model [28] which is commonly used in literature for ground vehicles and fixed-wing UAVs. We first specify the nominal trajectory along with the inputs as required by Algorithm 2. We then follow the rest of Algorithm 2 to obtain the sets $\mathcal{R}_t$ using Equation (4.52). In order to validate that these sets satisfy the requirement in Equation (4.6), we simulate a 1000 trajectory rollouts for the robot and check the ratio of trajectories with state $\mathbf{x}_t$ inside the set $\mathcal{R}_t$. Thus, for a 3σ confidence level ($m = 3$) used in our simulations we expect the ratio to be greater than 0.997.

The state vector for the Dubins model consists of the robot positions (x, y) and the heading angle θ. The control inputs to the system are the forward velocity V and the angular velocity ω. Thus, the motion model of the robot can be written in the form of Equation (4.1) as:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} V_{t-1}\cos(\theta_{t-1})dt \\ V_{t-1}\sin(\theta_{t-1})dt \\ \omega_{t-1}dt \end{bmatrix}}_{f(\mathbf{x}_{t-1},\mathbf{u}_{t-1})} + \mathbf{w}_t, \tag{4.59}$$

where $dt = 0.2\,\mathrm{s}$ is the time-step between two time instants and $\mathbf{w}_t$ is the motion model error with covariance Q_t as:

$$\mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 0.01\,\mathrm{m}^2 & 0 & 0 \\ 0 & 0.01\,\mathrm{m}^2 & 0 \\ 0 & 0 & 0.001\,\mathrm{rad}^2 \end{bmatrix}\right).$$

For the sensing model, we assume the availability of ranging measurements from beacons near the robot along with heading measurements from an on-board compass. The range measurement from the i^{th} beacon can be expressed as:

$$r_t^{(i)} = \left\|\mathbf{x}_t - \mathbf{x}_t^{b_i}\right\|_2 + v_t^{b_{(i)}} + v_t^{s_{(i)}}, \tag{4.60}$$

where $\|\cdot\|_2$ represents the distance between the true robot position $\mathbf{x}_t$ and the i^{th} beacon position $\mathbf{x}_t^{b_i}$, $v_t^{b_{(i)}}$ is a bias component of the ranging measurement error for which we set the bounds as $\pm 1\,\mathrm{m}$ and $v_t^{s_{(i)}}$ is a stochastic component

of the ranging measurement error modeled as a Gaussian distribution $v_t^{s_{(i)}} \sim \mathcal{N}(0, 0.1\,\mathrm{m}^2)$. Thus, given N ranging beacons, the sensing model can be written in the from of Equation (4.2) as:

$$\underbrace{\begin{bmatrix} z_t^{(1)} \\ \vdots \\ z_t^{(N)} \\ z_t^{(N+1)} \end{bmatrix}}_{\mathbf{z}_t} = \underbrace{\begin{bmatrix} \left\|\mathbf{x}_t - \mathbf{x}_t^{b_1}\right\|_2 \\ \vdots \\ \left\|\mathbf{x}_t - \mathbf{x}_t^{b_N}\right\|_2 \\ \theta_t \end{bmatrix}}_{h(\mathbf{x}_t)} + \underbrace{\begin{bmatrix} v_t^{b_{(1)}} \\ \vdots \\ v_t^{b_{(N)}} \\ 0 \end{bmatrix}}_{\mathbf{v}_t^b} + \underbrace{\begin{bmatrix} v_t^{s_{(1)}} \\ \vdots \\ v_t^{s_{(N)}} \\ v_t^{s_{(N+1)}} \end{bmatrix}}_{\mathbf{v}_t^s}, \tag{4.61}$$

where $z_t^{(i)} = r_t^{(i)}$ from Equation (4.60) $\forall\ i = 1$ to N, and $z_t^{(N+1)}$ represents the heading measurement from the on-board compass for which we model the error as a Gaussian distribution $v_t^{s_{(N+1)}} \sim \mathcal{N}(0, 0.001\,\mathrm{rad}^2)$.

For the state feedback control gains $(\check{K}_0, \check{K}_1, \ldots, \check{K}_T)$ we use a locally optimal LQR similar to Equation (3.55). Thus, we use the following function in our `MATLAB` implementation:

$$\check{K}_t = \texttt{dlqr}\left(A_t, B_t, \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}\right), \tag{4.62}$$

where $A_t = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\check{\mathbf{x}}_t}$ and $B_t = \frac{\partial f}{\partial \mathbf{u}}\Big|_{\mathbf{u}=\check{\mathbf{u}}_t}$ mentioned in Equation (4.18), and the other matrices can be tuned for desired trajectory tracking performance. For the initial state estimation error covariance, we set P_0 as:

$$P_0 = \begin{bmatrix} 0.1\,\mathrm{m}^2 & 0 & 0 \\ 0 & 0.1\,\mathrm{m}^2 & 0 \\ 0 & 0 & 0.01\,\mathrm{rad}^2 \end{bmatrix}. \tag{4.63}$$

Next, we obtain the double derivative matrices of the non-linear motion model $J_{\mathbf{s}}^{f_1}$ and $J_{\mathbf{s}}^{f_2}$ required for the Lagrange remainder approximation in Section 4.4:

$$J_{\mathbf{s}}^{f_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -V\cos(\theta)dt & -\sin(\theta)dt & 0 \\ 0 & 0 & -\sin(\theta)dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.64}$$

$$J_{\mathbf{s}}^{f_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -V\sin(\theta)dt & \cos(\theta)dt & 0 \\ 0 & 0 & \cos(\theta)dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.65}$$

$$J_{\mathbf{s}}^{f_3} = O, \tag{4.66}$$

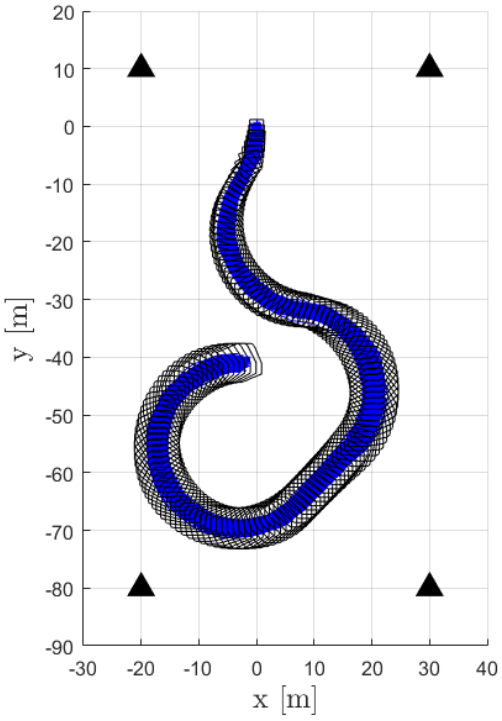
where $\mathbf{s}$ is the combined state and control input vector, i.e., $\mathbf{s}^\top = [\mathbf{x}^\top, \mathbf{u}^\top]$, and O is a zero matrix of appropriate dimensions. Similarly, the double derivative matrices for the non-linear sensing model are obtained as:

$$J_{\mathbf{x}}^{h_{(i)}} = \begin{bmatrix} \frac{(y-y^{b_i})^2}{\left\|\mathbf{x}-\mathbf{x}^{b_i}\right\|_2^3} & -\frac{(x-x^{b_i})(y-y^{b_i})}{\left\|\mathbf{x}-\mathbf{x}^{b_i}\right\|_2^3} & 0 \\ -\frac{(x-x^{b_i})(y-y^{b_i})}{\left\|\mathbf{x}-\mathbf{x}^{b_i}\right\|_2^3} & \frac{(x-x^{b_i})^2}{\left\|\mathbf{x}-\mathbf{x}^{b_i}\right\|_2^3} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.67}$$

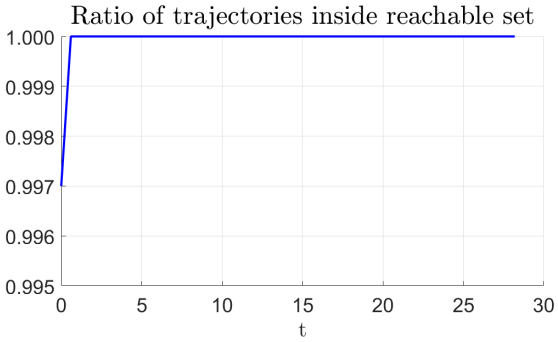
$$J_{\mathbf{x}}^{h_{(N+1)}} = O. \tag{4.68}$$

Finally, given all the aforementioned information, we first obtain the sets $\mathcal{R}_t$ as explained in Algorithm 2 and then validate these sets by simulating 1000 trajectory rollouts for the Dubins system. Fig. 4.2 shows the results of our simulation. We observe that at least 997 of the 1000 trajectory rollouts remain inside the sets $\mathcal{R}_t \ \forall \ t \in [0, T]$, which reflects the desired 3σ confidence level. Thus, our simulation statistically shows that the requirement in Equation (4.6) is satisfied.

Additionally, we validate our reachability analysis in the presence of varying sensing uncertainties as shown in Fig. 3.5(a). Here for the region shaded in white we set the bounds for the bias in the ranging measurements to be 0 m (i.e., no biases in the measurements), whereas the region shaded in red is set to have a larger bias bound of ± 1 m. The rest of the parameters are set similarly to the simulation in Fig. 4.2. We simulate a 1000 trajectory rollouts for the system and statistically show in Fig. 3.5(b) that the requirement in Equation (4.6) is satisfied.

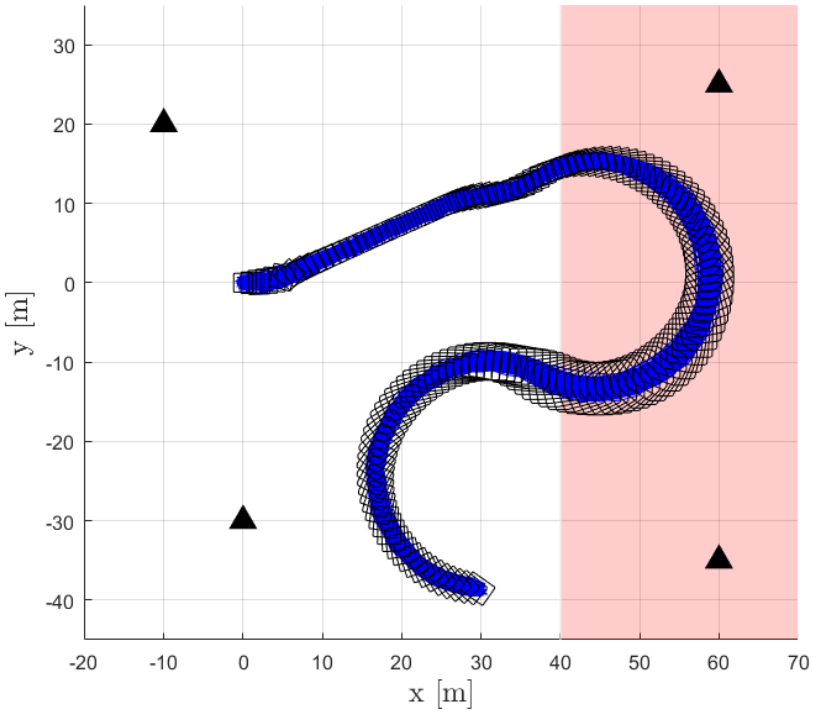


(a)

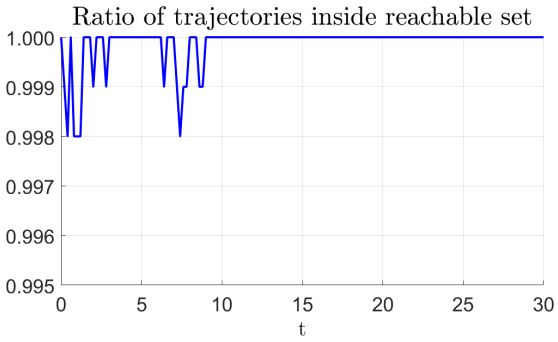


(b)

Figure 4.2: Reachability analysis for a 2-dimensional Dubins model receiving ranging measurements from beacons (black triangles) and heading measurements from an on-board compass. (a) The sets $\mathcal{R}_t$ (black) are obtained using our analysis described in Algorithm 2, which are then validated by simulating 1000 trajectory rollouts (blue). (b) The ratio of trajectories inside the sets $\mathcal{R}_t$ reflects the desired 3σ confidence level.



(a)



(b)

Figure 4.3: Reachability analysis in presence of varying sensing uncertainties (larger bounds for measurement bias in red shaded region) for a 2-dimensional Dubins model receiving ranging measurements from beacons (black triangles) and heading measurements from an on-board compass. (a) The sets $\mathcal{R}_t$ (black) are obtained using our analysis described in Algorithm 2, which are then validated by simulating 1000 trajectory rollouts (blue). (b) The ratio of trajectories inside the sets $\mathcal{R}_t$ reflects the desired 3σ confidence level.

4.6 Chapter Summary

In this chapter we developed our reachability analysis for robots represented by non-linear motion and sensing models. We first derived equations governing the growth of the robot state and state estimation error along a nominal trajectory using the process of mathematical induction. We then transitioned the equations to set notations in order to compute the reachable set for the robot states. Additionally, we developed an approximation for the linearization errors (referred to as Lagrange remainders) required to compute the reachable sets. Algorithm 2 summarized our non-linear reachability analysis.

To validate our reachability analysis, we simulated a 2-dimensional Dubins model receiving ranging measurements from beacons and heading measurements from an on-board compass. For each simulation we generated 1000 trajectory rollouts and statistically showed that the requirement stated in the problem formulation was satisfied.

CHAPTER 5

TRAJECTORY PLANNING USING REACHABILITY ANALYSIS

In this chapter we show how our reachability analysis from Chapters 3 and 4 can be integrated with existing trajectory planning frameworks in order to plan collision-safe trajectories for a robot. We first formulate the trajectory planning problem including specifications of the robot motion and sensing models in Section 5.1. Next, in Section 5.2, we integrate our reachability analysis with the planning framework designed in [51]. Here we provide a heuristic approximation of our reachability analysis in order to efficiently evaluate the collision-safety of candidate trajectories (Section 5.2.2). Additionally, we propose a metric for the size of the computed reachable sets which allows us to compare different candidate trajectories during the planning process (Section 5.2.3). Finally, via simulations in Section 5.3, we demonstrate the applicability of the trajectory planner for GNSS-based navigation of fixed-wing UAVs in an urban environment. Planning results are discussed for a UAV in a static environment and in a shared airspace with other UAVs. We statistically validate the collision-safety of the UAVs by simulating multiple trajectory rollouts along the planned trajectories, similar to the process followed in Sections 3.4 and 4.5.

5.1 Problem Formulation

We formulate the planning problem for a non-linear system as done in Chapter 4. The problem can be formulated similarly for a linear system from Chapter 3 if desired. The discrete-time robot motion model is written as:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathbf{w}_t, \tag{5.1}$$

where t is the time instant, $\mathbf{x}_t$ is the state vector, $\mathbf{u}_t$ is the control input vector, f is a non-linear function representing the robot motion, and $\mathbf{w}_t$ is

the motion model error represented by a Gaussian distribution $\mathcal{N}(0, Q_t)$. For measurements, the following non-linear sensing model is considered:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \tag{5.2}$$

where $\mathbf{z}_t$ is the measurement vector, h is a function representing the non-linear sensing model and $\mathbf{v}_t$ is the sensing model error. As mentioned in Equation (4.3), $\mathbf{v}_t$ contains a bounded bias component $\mathbf{v}_t^b \in [\underline{\mathbf{b}}_t, \overline{\mathbf{b}}_t] = \mathcal{B}_t$ and a stochastic component $\mathbf{v}_t^s \sim \mathcal{N}(\mathbf{0}, R_t)$.

Similar to prior trajectory planning work [48], we assume that there exists a low-level `CONNECT` function that connects a trajectory between two states. Thus, given two states $\mathbf{x}^i$ and $\mathbf{x}^j$, the `CONNECT` function provides us the following:

$$\left(\check{X}^{i,j}, \check{U}^{i,j}, \check{K}^{i,j}\right) = \texttt{CONNECT}(\mathbf{x}^i, \mathbf{x}^j), \tag{5.3}$$

where $\check{X}^{i,j}$ is the sequence of nominal states $(\check{\mathbf{x}}_{\tau_i}, \check{\mathbf{x}}_{\tau_i+1}, \cdots, \check{\mathbf{x}}_{\tau_j-1})$ and $\check{U}^{i,j}$ is the sequence of nominal inputs $(\check{\mathbf{u}}_{\tau_i}, \check{\mathbf{u}}_{\tau_i+1}, \cdots, \check{\mathbf{u}}_{\tau_j-1})$. These nominal states and inputs satisfy the nominal motion model from Equation (5.1):

$$\begin{gathered} \check{\mathbf{x}}_t = f(\check{\mathbf{x}}_{t-1}, \check{\mathbf{u}}_{t-1}) \;\forall\; t \in [\tau_i + 1, \tau_j - 1], \\ \mathbf{x}^i = \check{\mathbf{x}}_{\tau_i},\; \mathbf{x}^j = f(\check{\mathbf{x}}_{\tau_j-1}, \check{\mathbf{u}}_{\tau_j-1}), \end{gathered} \tag{5.4}$$

where $\check{K}^{i,j}$ is the sequence of stabilizing linear state feedback control gains $(\check{K}_{\tau_i}, \check{K}_{\tau_i+1}, \cdots, \check{K}_{\tau_j-1})$. We assume that the robot uses an EKF as defined in Section 4.2 for on-board state estimation. Given the feedback control gain and an on-board state estimate $\hat{\mathbf{x}}_t$, the total control input during execution is of the form:

$$\mathbf{u}_t = \check{\mathbf{u}}_t - \check{K}_t(\hat{\mathbf{x}}_t - \check{\mathbf{x}}_t). \tag{5.5}$$

The availability of such a `CONNECT` function along with the computation of the corresponding sequences $\check{X}$, $\check{U}$, and $\check{K}$ has been widely addressed in literature [28, 53] and is beyond the scope of this work. For instance, for a Dubins vehicle the optimal nominal states and inputs can be easily obtained in closed form [52] and the feedback control gains can be obtained using a locally optimal Linear-Quadratic Regulator (LQR) design.

Let $\mathcal{X}_t^u$ denote the set of unsafe (or undesirable) states for the robot at time instant t. Here we assume $\mathcal{X}_t^u$ to be comprised of two components:

$\mathcal{X}^{u,s}$ that represents static obstacles in the environment such as buildings and no-operation zones (no-fly zones for UAVs), and $\mathcal{X}_t^{u,d}$ that represents the sets occupied by other dynamic robots operating in the environment at time instant t.

Thus, we assume that the following information is available for trajectory planning:

1. The covariance matrices Q_t for the motion model error in Equation (5.1).
2. The covariance matrices R_t and the bounds $\mathcal{B}_t$ for the sensing model error in Equation (5.2).
3. A `CONNECT` function as defined in Equation (5.3).
4. The set of unsafe (or undesirable) states $\mathcal{X}_t^u$.

Given the above information along with an initial state $\mathbf{x}^{\text{init}}$ and a desired goal state $\mathbf{x}^{\text{goal}}$, the problem for the trajectory planner is defined as:

$$\begin{aligned} &\min_{(\check{X},\check{U},\check{K})} \text{cost}(\check{X},\check{U},\check{K}), \\ &\text{subject to: } \Pr(\mathbf{x}_t \in \mathcal{X}_t^u) < \delta, \end{aligned} \tag{5.6}$$

where δ is a specified threshold for the probability of collision, and $(\check{X},\check{U},\check{K})$ are concatenated sequences representing the nominal trajectory from $\mathbf{x}^{\text{init}}$ to $\mathbf{x}^{\text{goal}}$ as follows:

$$(\check{X},\check{U},\check{K}) = \left(\texttt{CONNECT}(\mathbf{x}^{\text{init}},\mathbf{x}^1), \texttt{CONNECT}(\mathbf{x}^1,\mathbf{x}^2), \cdots, \texttt{CONNECT}(\mathbf{x}^l,\mathbf{x}^{\text{goal}})\right). \tag{5.7}$$

The probability value δ is typically obtained from a desired confidence level of collision-safety. Thus, for a $m\sigma$ confidence level, where $m > 0$, δ is calculated as: $\delta = 1 - \texttt{erf}(m/\sqrt{2})$. This value of δ is usually provided as a parameter to the trajectory planning algorithm. Smaller values of δ translate to a stricter collision-safety requirement and could potentially result in the planner being unable to find a trajectory, whereas larger values of δ result in a more collision-risky trajectory being planned. For our simulations in Section 5.3 we choose δ based on a 3σ confidence level ($m = 3$) for collision-safety.

Note that the trajectory planner described next in Section 5.2 is applicable for any general robot motion model (in Equation (5.1)) and sensing model (in Equation (5.2)) and any general cost function (in Equation (5.6)).

5.2 Trajectory Planning Algorithm

In this section we describe the details of the trajectory planner used for solving the problem defined in Equation (5.6). We first provide an overview of the planning framework in [51] and how we use our reachability analysis within the framework. Next, we develop a heuristic approximation of the reachable set computation from Chapters 3 and 4. This approximation allows us to efficiently evaluate the collision safety of candidate trajectories during the planning process. Finally, we propose a metric to compare the sizes of the approximate reachable sets which is required for comparing and eliminating undesirable candidate trajectories in [51].

5.2.1 Overview

The objective of the trajectory planner is to explore the environment and find a solution for the problem defined in Equation (5.6). We choose the planning framework from [51] given its highly parallelizable structure, which is desirable for real-world applications. However, note that our reachability analysis can also be integrated with other planning frameworks [48, 50] if desired. The planning framework from [51] can be summarized as follows:

1. The planner begins with a graph constructing phase. Multiple states are sampled in the environment (including $\mathbf{x}^{\text{init}}$ and $\mathbf{x}^{\text{goal}}$) and kinematically feasible trajectories obeying the nominal motion model from Equation 5.4 are obtained between nearby states using the `CONNECT` function defined in Equation (5.3). The trajectory between two states $\mathbf{x}^i$ and $\mathbf{x}^j$ is added to the graph if the corresponding sequence of nominal states $\check{X}^{i,j}$ is collision-free with respect to static set of unsafe (or undesirable) states in the environment, i.e., with respect to $\mathcal{X}^{u,s}$.

2. Next, the planner explores candidate trajectories in the graph in order to find a complete trajectory from $\mathbf{x}^{\text{init}}$ to $\mathbf{x}^{\text{goal}}$. Here in order to evalu-

ate the collision-safety of a trajectory, we use our reachability analysis from Chapters 3 and 4. Since exploring the graph requires evaluating the collision-safety of numerous candidate trajectories, we develop a heuristic approximation of the reachable set computation to speed up the planning process. The details of evaluating the collision-safety along with the approximation are discussed below in Section 5.2.2.

3. For computational tractability during the graph exploration phase, the planner eliminates undesirable candidate trajectories by comparing trajectories that arrive at the same state. Here the comparison is done in terms of the trajectory costs from $\mathbf{x}^{\text{init}}$ to the current state and the sizes of the reachable sets at the current state. We present the details of comparing candidate trajectories along with our proposed metric for the size of the reachable set below in Section 5.2.3.

4. Once a collision-safe trajectory to the goal state $\mathbf{x}^{\text{goal}}$ is found, the planner stops exploring the graph and outputs $(\check{X}, \check{U}, \check{K})$, i.e., the sequence of nominal states, nominal control inputs and feedback control gains for the robot to follow during trajectory execution.

Note that the planner works in an *offline* manner, i.e., first the solution to the planning problem from Equation (5.6) is found and then the planned trajectory is executed by the robot. Fig. 5.1 illustrates the planning graph constructing and exploration phases. For additional details of the planning framework, we refer the readers to [51].

5.2.2 Evaluating Collision-Safety

An important component of the graph exploration phase during planning is to evaluate if candidate trajectories are collision-safe for the robot. The evaluation for a candidate trajectory is done in two steps: first we compute approximate reachable sets for the robot along the trajectory using our analysis from Chapters 3 and 4, and second we check if these approximate reachable sets intersect with the set of unsafe states.

From Equation (4.49), we have the following expression for computing the

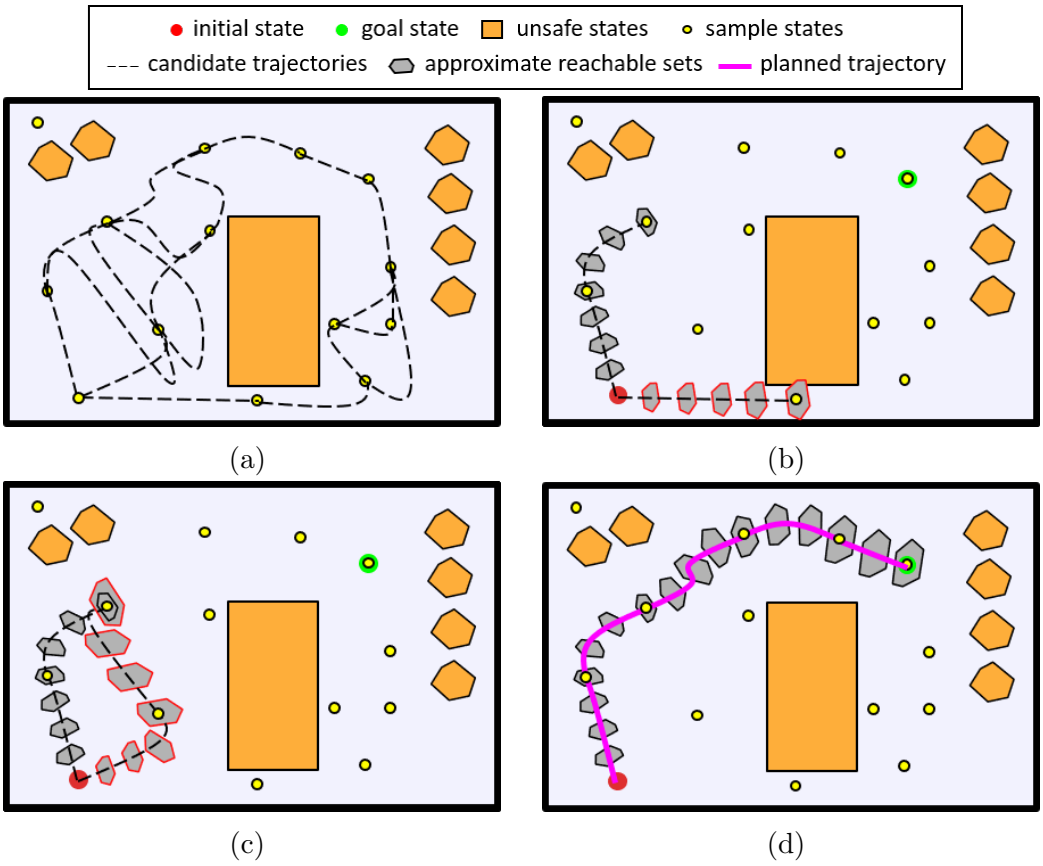


Figure 5.1: Overview of the trajectory planning framework [51] integrated with our reachability analysis. (a) The planner first constructs a graph in the environment by randomly sampling states and using the `CONNECT` function defined in Equation (5.3). (b) Next, the planner begins exploring the graph and evaluates the collision-safety of candidate trajectories. Approximate reachable sets are computed along the trajectories using Equation (5.9) and are used to detect and eliminate collision-unsafe trajectories (red outline) from the graph exploration phase. (c) For computational tractability, undesirable candidate trajectories (red outline) are eliminated from the graph exploration phase. Here candidate trajectories are compared based on their costs (such as trajectory lengths in this illustration) and the size of the approximate reachable sets (from Equation (5.12)). (d) Finally, the planner stops exploring once a collision-safe trajectory from the initial state to the goal state is found.

reachable set $\mathcal{X}_t$ for a general non-linear system described in Section 5.1:

$$\begin{aligned}\mathcal{X}_t = \check{\mathbf{x}}_t \oplus {}^1\Phi_t(\mathcal{X}_0 - \check{\mathbf{x}}_0) \oplus {}^2\Phi_k\tilde{\mathcal{X}}_0 \oplus \sum_{n=1}^{t} {}^3\Phi_t^n\mathcal{W}_n \oplus \sum_{n=1}^{t} {}^4\Phi_t^n\mathcal{V}_n \oplus \\ \sum_{n=0}^{t-1} {}^5\Phi_t^n\widehat{\mathcal{L}}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} \oplus \sum_{n=0}^{t-1} {}^6\Phi_t^n\widehat{\mathcal{L}}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} \oplus \sum_{n=0}^{t-1} {}^7\Phi_t^n\widehat{\mathcal{L}}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} \oplus \sum_{n=0}^{t-1} {}^8\Phi_t^n\widehat{\mathcal{L}}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n},\end{aligned} \tag{5.8}$$

where $\check{\mathbf{x}}_t$ is the nominal state along the candidate trajectory, $\mathcal{X}_0$ and $\tilde{\mathcal{X}}_0$ are the initial set of robot states and the initial state estimation error set respectively at $\mathbf{x}^{\text{init}}$, $\mathcal{W}_n$ and $\mathcal{V}_n$ are the sets of motion and sensing uncertainties along the trajectory, ${}^i\Phi_t$ are matrix coefficients as obtained in Equation (4.48) and $\widehat{\mathcal{L}}$ are approximations of the Lagrange remainders as obtained in Equation (4.58). Note that as the trajectory grows (i.e., as t increases), the number of Minkowski sum operations in Equation (5.8) increases, consequently increasing the computation load of Equation (5.8). Given that the graph exploration phase requires evaluating the collision-safety of numerous candidate trajectories, it is desirable to speed up the process of computing reachable sets. Thus, we provide a heuristic approximation to bound the number of Minkowski sum operations in Equation (5.8) and efficiently compute approximate reachable sets for the robot.

From Equations (4.45) and (4.48) we observe that updating the matrix coefficients involve contractive terms $(A_{t-1} - B_{t-1}\check{K}_{t-1})$ and $(I - L_tC_t)$, where A_t, B_t and C_t are partial derivatives as defined in Equations (4.18) and (4.27) of the motion and sensing models f and h, $\check{K}_t$ is the feedback control gain and L_t is the Kalman gain from the on-board EKF. Consequently, some matrix coefficients become negligible and this results in some quantities in Equation (5.8) having a negligible contribution in the computation of $\mathcal{X}_t$. The intuition behind this is that as the trajectory grows, the reachable set $\mathcal{X}_t$ depends more on recent quantities (such as recent sets of motion and sensing uncertainties) as opposed to quantities from earlier in the trajectory (such as initial sets of motion and sensing uncertainties). In order to check which quantities in Equation (5.8) negligibly contribute to $\mathcal{X}_t$, we check the Frobenius norm of the matrix coefficients ${}^i\Phi_t$. Only quantities whose corresponding matrix coefficients ${}^i\Phi_t$ have a Frobenius norm higher than a specified threshold ${}^i\zeta$ are considered. This gives us the following expression for the approximate

reachable set $\underline{\mathcal{X}}_t$:

$$\underline{\mathcal{X}}_t = \check{\mathbf{x}}_t \oplus {}^1\Phi_t(\mathcal{X}_0 - \check{\mathbf{x}}_0) \oplus {}^2\Phi_t\tilde{\mathcal{X}}_0 \oplus \sum_{n\in{}^3\mathcal{N}_t} {}^3\Phi_t^n\mathcal{W}_n \oplus \sum_{n\in{}^4\mathcal{N}_t} {}^4\Phi_t^n\mathcal{V}_n \oplus \sum_{n\in{}^5\mathcal{N}_t} {}^5\Phi_t^n\widehat{\mathcal{L}}^f_{[\mathbf{s},\check{\mathbf{s}}]_n} \oplus \sum_{n\in{}^6\mathcal{N}_t} {}^6\Phi_t^n\widehat{\mathcal{L}}^f_{[\hat{\mathbf{s}},\check{\mathbf{s}}]_n} \oplus \sum_{n\in{}^7\mathcal{N}_t} {}^7\Phi_t^n\widehat{\mathcal{L}}^h_{[\mathbf{x},\check{\mathbf{x}}]_n} \oplus \sum_{n\in{}^8\mathcal{N}_t} {}^8\Phi_t^n\widehat{\mathcal{L}}^h_{[\bar{\mathbf{x}},\check{\mathbf{x}}]_n}, \tag{5.9}$$

where ${}^i\mathcal{N}_t$ contains the following set of elements:

$${}^i\mathcal{N}_t = \left\{ n \in [0,t] \mid \left\| {}^i\Phi_t^n \right\|_F \geq {}^i\zeta \right\}, \tag{5.10}$$

where $\|.\|_F$ represents the Frobenius norm.

Next, in order to evaluate the collision-safety of the candidate trajectory, we need to check if the constraint from Equation (5.6) is satisfied, i.e., if $\Pr(\mathbf{x}_t \in \mathcal{X}_t^u) < \delta$ along the trajectory. We first obtain confidence set $\underline{\mathcal{R}}_t$ corresponding to the approximate reachable set $\underline{\mathcal{X}}_t$ as:

$$\underline{\mathcal{R}}_t = \texttt{conf}(\underline{\mathcal{X}}_t, m), \tag{5.11}$$

where `conf` is the confidence set operation defined in Section 2.1.4 and m denotes the desired confidence level corresponding to the probability value δ. Finally, given $\underline{\mathcal{R}}_t$, we consider the candidate trajectory to be collision-safe only if all $\underline{\mathcal{R}}_t$ along the trajectory do not intersect with the corresponding set of unsafe states $\mathcal{X}_t^u$. Trajectories detected as collision-unsafe are eliminated from the graph exploration phase, as illustrated in Fig. 5.1(b).

5.2.3 Comparing Candidate Trajectories

As the size of the constructed planning graph increases, it becomes computationally intractable to explore all the candidate trajectories in the graph. Thus, in [51] the authors propose a method to compare candidate trajectories and eliminate undesirable ones during the graph exploration phase. Trajectories arriving at the same state in the graph are compared with each other based on two criteria: the cost of the trajectories from the initial state $\mathbf{x}^{\text{init}}$ to the current state and the size of the reachable sets at the current state. A candidate trajectory is considered to be undesirable (and consequently elimi-

nated from the graph exploration phase) if the there exists another trajectory that arrives at the same state with a lower cost and a smaller reachable set.

Thus, we need a metric to measure the size of the approximate reachable sets $\underline{\mathcal{X}}_t$ computed using Equation (5.9). For this purpose we propose the following metric for the size of $\underline{\mathcal{X}}_t$ using the corresponding confidence set $\underline{\mathcal{R}}_t$ obtained in Equation (5.11):

$$\texttt{size}(\underline{\mathcal{X}}_t) = \texttt{trace}(G_{\underline{\mathcal{R}}_t}^{\top} G_{\underline{\mathcal{R}}_t}), \tag{5.12}$$

where $G_{\underline{\mathcal{R}}_t}$ represents the generator matrix of $\underline{\mathcal{R}}_t$. Here $G_{\underline{\mathcal{R}}_t}^{\top} G_{\underline{\mathcal{R}}_t}$ is referred to as the covariation matrix [68] of $\underline{\mathcal{R}}_t$, and is analogous to the covariance matrix of a Gaussian distribution. Thus, for comparing the size of the reachable sets during the graph exploration phase, we use the scalar value obtained using Equation (5.12).

5.3 Simulations for GNSS-based UAV Navigation

In this section we demonstrate the applicability of the trajectory planner for GNSS-based navigation of fixed-wing UAVs in an urban environment. We first describe the UAV motion model and GNSS-psuedorange-based sensing model used in the simulations. Next, we provide details of the 3-dimensional (3D) environment setup along with the process followed to obtain the bias bounds in the pseudorange measurements. Finally, we discuss planning results in a static environment and in a shared airspace with other operating UAVs. The collision-safety of the planned trajectories is statistically validated by simulating 1000 trajectory rollouts and considering a 3σ confidence level ($m = 3$).

5.3.1 Motion and Sensing Models

For simplicity we restrict the fixed-wing UAV motion to a horizontal plane and represent it by a 2-dimensional Dubins model, similar to the model used in Equation (4.59). The state vector consists of the UAV positions (x, y) and the heading angle θ, whereas the control inputs are the forward velocity V and the angular velocity ω. Thus, the motion model of the robot can be

written in the form of Equation (5.1) as:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} V_{t-1}\cos(\theta_{t-1})dt \\ V_{t-1}\sin(\theta_{t-1})dt \\ \omega_{t-1}dt \end{bmatrix}}_{f(\mathbf{x}_{t-1},\mathbf{u}_{t-1})} + \mathbf{w}_t, \tag{5.13}$$

where $dt = 0.2\,\mathrm{s}$ and $\mathbf{w}_t$ is modeled as:

$$\mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 0.01\,\mathrm{m}^2 & 0 & 0 \\ 0 & 0.01\,\mathrm{m}^2 & 0 \\ 0 & 0 & 0.001\,\mathrm{rad}^2 \end{bmatrix}\right).$$

For the sensing model, we assume the availability of GNSS pseudorange measurements along with heading measurements from an on-board compass. Here we assume the pseudorange measurements have been corrected for atmospheric effects [55] and consider the following commonly used model:

$$\rho_t^{(i)} = \left\|\mathbf{x}_t - \mathbf{x}_t^{s_i}\right\|_2 + c\delta t + v_t^{b_{(i)}} + v_t^{s_{(i)}}, \tag{5.14}$$

where $\|\cdot\|_2$ represents the true range between the receiver position $\mathbf{x}_t$ and satellite position $\mathbf{x}_t^{s_i}$, $c\delta t$ is the clock bias error, and $v_t^{b_{(i)}}$ is an additional bias component due to multipath/NLOS discussed further below in Section 5.3.2, $v_t^{s_{(i)}}$ is the stochastic component modeled as a Gaussian distribution. Here the satellite positions are simulated from publicly available almanac data. We model the covariance of the stochastic component $v_t^{s_{(i)}}$ with an elevation-based factor [102, 103] as: $R_t^{(i)} = \Sigma_\rho / \sin^2(\mathrm{el}^{(i)})$, where $\mathrm{el}^{(i)}$ is the elevation angle of the i^{th} satellite and Σ_ρ is set to be $5\,\mathrm{m}^2$. Since we are primarily concerned with the UAV position states, we assume for simplicity that the receiver clock and the satellite clocks are perfectly synced, i.e., there is zero clock bias error ($\delta t = 0$). However, if desired, clock bias states can also be included in the state vector for the trajectory planner. Thus, given N GNSS

satellites, the sensing model for the fixed-wing UAV looks as follows:

$$\underbrace{\begin{bmatrix} z_t^{(1)} \\ \vdots \\ z_t^{(N)} \\ z_t^{(N+1)} \end{bmatrix}}_{\mathbf{z}_t} = \underbrace{\begin{bmatrix} \|\mathbf{x}_t - \mathbf{x}_t^{s_1}\|_2 \\ \vdots \\ \|\mathbf{x}_t - \mathbf{x}_t^{s_N}\|_2 \\ \theta_t \end{bmatrix}}_{h(\mathbf{x}_t)} + \underbrace{\begin{bmatrix} v_t^{b_{(1)}} \\ \vdots \\ v_t^{b_{(N)}} \\ 0 \end{bmatrix}}_{\mathbf{v}_t^b} + \underbrace{\begin{bmatrix} v_t^{s_{(1)}} \\ \vdots \\ v_t^{s_{(N)}} \\ v_t^{s_{(N+1)}} \end{bmatrix}}_{\mathbf{v}_t^s}, \tag{5.15}$$

where $z_t^{(i)} = \rho_t^{(i)}$ from Equation (5.14) $\forall\ i = 1$ to N, and $z_t^{(N+1)}$ represents the heading measurement from the on-board compass for which we model the error as a Gaussian distribution $v_t^{s_{(N+1)}} \sim \mathcal{N}(0, 0.001\,\text{rad}^2)$. The remaining setup for the UAV is similar to the setup in Section 4.5 from Equation (4.62) to (4.68).

5.3.2 3D Environment and Pseudorange Bias Bounds

For our simulations, we setup a large 3D urban environment in the Unity game engine [104]. The environment dimensions are 1 km $\times$ 1 km wide, and contain buildings up to 120 m tall, whereas we set the UAV flight to be 65 m.

As discussed in Section 1.2.2, we only consider multipath signals and assume the NLOS signals to be detected and excluded by available outlier rejection methods [56, 57, 58, 59, 60]. However, if desired additional modeling can be performed to simulate the pseudorange bias due to NLOS signals [105, 61].

For the GNSS receiver architecture, we assume a quarter-chip spacing between the early and late correlators [55]. We consider the possibility of strong signal reflections from nearby buildings, and assume that the reflected signal strength can be as strong as the direct LOS signal. A positive pseudorange bias occurs when the direct and reflected signals are in-phase (constructive interference), whereas a negative pseudorange bias occurs when the signals are out-of-phase (destructive interference). It is non-trivial to estimate the phase difference by ray-tracing, thus we consider a phase difference of 0° and 180° to obtain an upper bound of the multipath noise envelope [61]. Fig. 5.2 shows the multipath noise envelope used for our simulations.

In order to obtain the bounds of the pseudorange bias, i.e., $\mathcal{B}_t$ in Equation (5.2), we need to estimate the possible differential path lengths before using

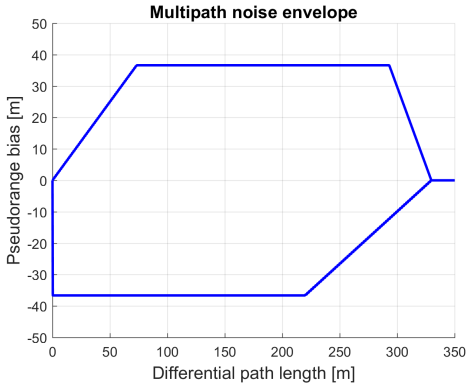


Figure 5.2: Multipath noise envelope for a GNSS receiver with a quarter-chip spacing between the early and late correlators.

Fig. 5.2. Thus, we perform the following steps to ray-trace the possible differential path lengths between a satellite and a receiver positions, and consequently to obtain the bounds $\mathcal{B}_t$:

1. Given the satellite and receiver position, we identify possible reflecting surfaces as illustrated in Fig. 5.3. Here we assume that the receiver is able to detect and exclude any signals arriving from below the receiver [106, 107]. Thus, reflections from the ground and from buildings lower than the receiver altitude are ignored.

2. Next, we ray-trace the possible differential path lengths for each surface (including diffuse reflections) and calculate the overall minimum and maximum differential path lengths. Here we assume that the 3D environment information is accurate, i.e., the building dimensions are exactly known. However, if desired, uncertainty in the building reflecting surfaces can be accounted for while calculating the minimum and maximum differential path lengths.

3. Finally, given the minimum and maximum differential path lengths, we use the multipath noise envelope in Fig. 5.2 to obtain $\mathcal{B}_t$ as the maximum bounds for the pseudorange bias.

Note that the ray-tracing procedure described in this section is performed offline during the planning graph exploration phase. The bounds $\mathcal{B}_t$ obtained using the ray-tracing are used in the reachability analysis to compute the sets $\underline{\mathcal{X}}_t$ from Equation (5.9), which are consequently used to evaluate the collision-safety of candidate trajectories.

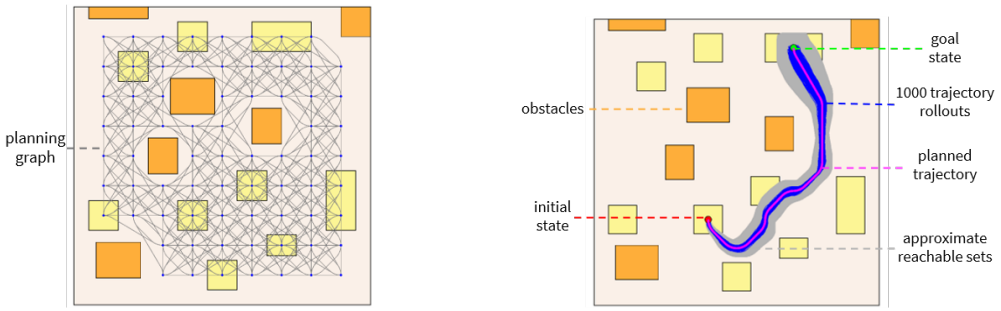
Figure 5.3: Simulated urban environment where we perform ray-tracing to compute the differential path lengths between the direct signal (green) and the multipath signals (yellow).

While we have made simplifying assumptions in our ray-tracing procedure, the main purpose is to demonstrate the applicability of our reachability analysis-based trajectory planner for GNSS-based navigation. In practice commercially available high-fidelity ray-tracing software [108, 109] can be used to obtain more accurate bounds for the pseudorange biases.

5.3.3 Simulation Results

Given the above motion and sensing models and the 3D environment setup, we validate the trajectory planner in two different scenarios: planning for a UAV navigating in the static 3D environment, and planning for a UAV navigating in the 3D environment in the presence of other operating UAVs. Note that all the trajectories for both the scenarios are planned offline, i.e., before any UAV begins flight. The results for the first scenario are shown in Fig. 5.4. Buildings taller than the flight altitude of 65 m are colored orange, whereas buildings shorter than the flight altitude are colored yellow. The taller buildings are considered to be static obstacles $\mathcal{X}^{u,s}$, whereas $\mathcal{X}_t^{u,d}$ is set to be empty for this scenario.

As described in Section 5.2.1, the planner first uses the `CONNECT` function defined in Equation (5.3) to construct a graph of kinematically feasible and collision-free trajectories. Here we use a grid of states with 100 m spacing as shown in Fig. 5.4(a). Once the graph is constructed, the planner explores candidate trajectories in the graph and finds a trajectory between the given initial state and goal state. In order to statistically validate the collision-safety of the planned trajectory, we simulate 1000 trajectory rollouts for the



(a) (b)

Figure 5.4: Planning results for a fixed-wing UAV in a static 1 km $\times$ 1 km wide environment. (a) A graph of kinematically feasible trajectories is constructed by the planner. Here we sample the states in a 100 m-spaced grid and obtain the trajectories using the `CONNECT` function defined in Equation (5.3). (b) The trajectory planner finds a trajectory from the initial state to the goal state such that the 3σ confidence sets of the approximate reachable sets do not intersect with the obstacles. We simulate 1000 trajectory rollouts for the UAV in order to statistically validate the collision-safety constraint from Equation (5.6).

UAV along the planned trajectory. Fig. 5.4(b) shows the planned trajectory, the confidence sets of the approximate reachable sets obtained using Equation (5.11) and the 1000 trajectory rollouts for the UAV. We observe that all 1000 simulated trajectories remain within the 3σ confidence sets and thus satisfy the collision-safety constraint from Equation (5.6).

For the second scenario, we sequentially plan trajectories for five UAVs in the 3D environment as shown in Fig. 5.5. Given that the UAV motion models and the environment are the same as the first scenario, we can re-use the planning graph constructed in Fig. 5.4(a). Additionally, we begin with the same set of unsafe states $\mathcal{X}_t^u$ as the first scenario, i.e., $\mathcal{X}^{u,s}$ contains the buildings, whereas $\mathcal{X}_t^{u,d}$ is empty. Given an initial state and a goal state for UAV-1, we plan a trajectory to maintain collision-safety with respect to $\mathcal{X}_t^u$. Before planning the trajectory for UAV-2, we update $\mathcal{X}_t^u$ such that $\mathcal{X}_t^{u,d}$ now contains the confidence sets of UAV-1. This is done in order to maintain collision-safety between UAV-1 and UAV-2 at any given time instant t. Thus, note that while a shorter candidate trajectory existed for UAV-2, it was eliminated by the planner since it could not maintain collision-safety with respect to UAV-1. Similarly, we update $\mathcal{X}_t^{u,d}$ to additionally contain the confidence sets of UAV-2. Thus, the trajectory for UAV-3 is planned while

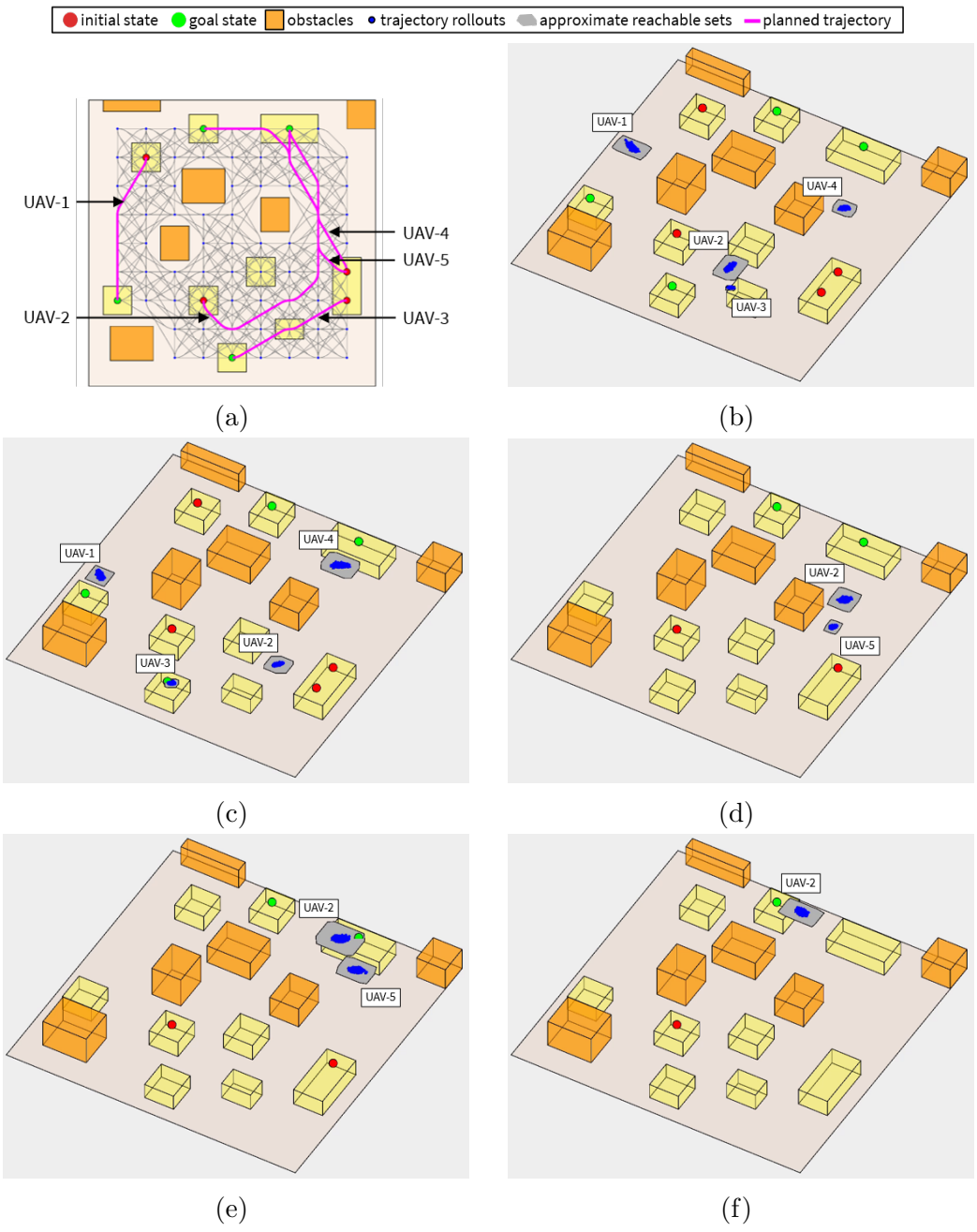


Figure 5.5: Sequential planning results for fixed-wing UAVs in the presence of other operating UAVs. (a) Planned trajectories for all the UAVs along with the planning graph. UAV-1 to UAV-4 are set to begin flying at the same time, whereas UAV-5 is set to begin flying 40 s later. (b)-(f) Snapshots of the approximate reachable sets for the UAVs at different time instants (in order of time). We simulate 1000 trajectory rollouts for each UAV in order to statistically validate the collision-safety constraint from Equation (5.6).

maintaining collision-safety with respect to the buildings and UAV-1 and UAV-2. A similar process is followed for planning the trajectory for UAV-4. For UAV-4 the planner finds a trajectory that seems to collide with the planned trajectory for UAV-2. However, the trajectories do not intersect temporally, thus maintaining collision-safety. In order to simulate situations when UAVs might begin flights at different times, we set UAV-5 to begin flying 40 s later than the other UAVs with the same initial state and goal state as UAV-4. Again the planned trajectory does not intersect temporally with the planned trajectories for UAV-2 and UAV-4, and maintains collision-safety with respect to the buildings and other UAVs.

In order to statistically validate the collision-safety of the planned trajectories for the five UAVs, we simulate 1000 trajectory rollouts for each UAV. Figs. 5.5(b)-(f) show snapshots of the trajectory rollouts and the 3σ confidence zonotopes of the approximate reachable sets at different time instants. All 1000 trajectory rollouts for all five UAVs remain within the confidence zonotopes and do not collide with buildings or other UAVs, thus satisfying the collision-safety constraint from Equation (5.6). The complete video for the second scenario can be found at `https://youtu.be/eyA3vEojdnQ`.

5.4 Chapter Summary

In this chapter, we designed a trajectory planner for a general non-linear system where the sensing model error contains a stochastic component (modeled as a Gaussian distribution) along with an additional bounded bias component. We utilized the highly parallelizable planning framework from [51] where a graph of candidate trajectories is first constructed and then explored. In order to efficiently evaluate the collision-safety of multiple candidate trajectories during the graph exploration phase, we developed a heuristic approximation for our reachability analysis from Chapters 3 and 4. We then proposed using the trace of the covariation matrix of the approximate reachable sets as a metric to compare their sizes during the graph exploration phase. Finally, we demonstrated the applicability of the planner for fixed-wing UAVs navigation in urban areas using GNSS pseudoranges. We validated the planner in two scenarios: planning for a UAV in a static environment, and sequentially planning for UAVs in a shared airspace with

other operating UAVs. 1000 trajectory rollouts were simulated for each UAV to statistically validate that the planned trajectories were collision-safe with respect to a 3σ confidence level.

CHAPTER 6

CONNECTIVITY MAINTENANCE

Inter-robot communication enables multi-robot systems to coordinate and execute complex missions efficiently. Thus, maintaining connectivity of the communication network between robots is essential for many multi-robot systems. In this chapter, we design a trajectory planner for connectivity maintenance of a multi-robot system. We first formulate the connectivity maintenance problem in Section 6.1. Then in Section 6.2 we define a weighted undirected graph to represent the connectivity of the system. Unlike previous connectivity maintenance works, we explicitly account for robot motion and sensing uncertainties while formulating the graph edge weights. These uncertainties result in uncertain robot positions which directly affect the connectivity of the system. Next, in Section 6.3, the algebraic connectivity of the weighted undirected graph is maintained above a specified lower limit using a trajectory planner based on a distributed Alternating Direction Method of Multipliers (ADMM) framework. Here we derive an approximation for the Hessian matrices required within the ADMM optimization step to reduce the computational load (Section 6.3.4). Finally, in Section 6.4, simulation results are discussed to statistically validate the connectivity maintenance of our trajectory planner.

6.1 Problem Formulation

6.1.1 Robot Description

For each robot i in a multi-robot system with N robots, we consider the following linear discrete-time motion and sensing models:

$$\mathbf{x}_{i,t} = A_{i,t-1}\mathbf{x}_{i,t-1} + B_{i,t-1}\mathbf{u}_{i,t-1} + \mathbf{w}_{i,t}, \tag{6.1}$$

$$\mathbf{z}_{i,t} = C_{i,t}\mathbf{x}_{i,t} + \mathbf{v}_{i,t}, \tag{6.2}$$

where t is the time instant, $\mathbf{x}_{i,t}$ is the state vector, $\mathbf{u}_{i,t}$ is the input vector, $\mathbf{z}_{i,t}$ is the sensed measurement vector, $A_{i,t}$ is is state transition matrix, $B_{i,t}$ is the control-input matrix, $C_{i,t}$ is the system measurement matrix, and $\mathbf{w}_{i,t}$ and $\mathbf{v}_{i,t}$ are Gaussian-distributed error vectors with covariance matrices $Q_{i,t}$ and $R_{i,t}$ respectively, such that $\mathbf{w}_{i,t} \sim \mathcal{N}(\mathbf{0}, Q_{i,t})$ and $\mathbf{v}_{i,t} \sim \mathcal{N}(\mathbf{0}, R_{i,t})$. We assume that each robot implements a Kalman filter (KF) on board to obtain an estimate of its state $\hat{\mathbf{x}}_i$. The prediction step of the KF is performed as:

$$\bar{\mathbf{x}}_{i,t} = A_{i,t-1}\hat{\mathbf{x}}_{i,t-1} + B_{i,t-1}\mathbf{u}_{i,t-1}, \tag{6.3}$$

$$\bar{P}_{i,t} = A_{i,t-1}P_{i,t-1}A_{i,t-1}^{\top} + Q_{i,t}, \tag{6.4}$$

where $P_{i,t}$ is the state estimation covariance matrix such that $\mathbf{x}_{i,t} \sim \mathcal{N}(\hat{\mathbf{x}}_{i,t}, P_{i,t})$. The KF correction step is performed as:

$$L_{i,t} = \bar{P}_{i,t}C_{i,t}^{\top}(C_{i,t}\bar{P}_{i,t}C_{i,t}^{\top} + R_{i,t})^{-1}, \tag{6.5}$$

$$\hat{\mathbf{x}}_{i,t} = \bar{\mathbf{x}}_{i,t} + L_{i,t}(\mathbf{z}_{i,t} - C_{i,t}\bar{\mathbf{x}}_{i,t}), \tag{6.6}$$

$$P_{i,t} = \bar{P}_{i,t} - L_{i,t}C_{i,t}\bar{P}_{i,t}, \tag{6.7}$$

where L_i is the Kalman gain. Here the second term in Equation (6.6) is referred to as the *innovation* term and is distributed according to $\mathcal{N}(\mathbf{0}, L_{i,t}C_{i,t}\bar{P}_{i,t})$.

Thus, each robot can be represented in the belief space with the belief vector defined as [47]:

$$\mathbf{b}_{i,t} = \begin{bmatrix} \hat{\mathbf{x}}_{i,t} \\ \mathbf{vec}(P_{i,t}) \end{bmatrix}, \tag{6.8}$$

where $\mathbf{vec}(P_{i,t})$ denotes a column vector containing the elements of the covariance matrix $P_{i,t}$. Furthermore, the belief dynamics for the robot can be summarized as [47]:

$$\mathbf{b}_{i,t+1} = \mathbf{g}_i(\mathbf{b}_{i,t}, \mathbf{u}_{i,t}) + M_i(\mathbf{b}_{i,t}, \mathbf{u}_{i,t})\mathbf{m}_{i,t}, \tag{6.9}$$

where:

$$\mathbf{g}_i(\mathbf{b}_{i,t}, \mathbf{u}_{i,t}) = \begin{bmatrix} \bar{\mathbf{x}}_{i,t} \\ \mathbf{vec}(\bar{P}_{i,t} - L_{i,t}C_{i,t}\bar{P}_{i,t}) \end{bmatrix},$$

$$M_i(\mathbf{b}_{i,t}, \mathbf{u}_{i,t}) = \begin{bmatrix} \sqrt{L_{i,t}C_{i,t}\bar{P}_{i,t}} \\ 0 \end{bmatrix},$$

$$\mathbf{m}_{i,t} \sim \mathcal{N}(\mathbf{0}, \mathcal{I}),$$

where $\mathcal{I}$ represents an identity matrix.

6.1.2 Connectivity Maintenance

The state vector of a robot $\mathbf{x}_i$ typically contains the position of the robot $\mathbf{p}_i$ along with additional quantities such as robot velocity. Similar to most previous connectivity maintenance works [80, 81, 82, 83, 84, 85], we assume a disk communication model. Thus, two robots are considered to be connected only if the distance between them is smaller than a specified communication range Δ. Let $l_{ij,t} = \|\mathbf{p}_{i,t} - \mathbf{p}_{j,t}\|_2$ be the euclidean distance between the true positions of two robots i and j, where $\|\cdot\|_2$ represents the L2-norm. The corresponding edge weight $\mathcal{A}_{ij,t}$ for the adjacency matrix is computed as:

$$\mathcal{A}_{ij,t} = \begin{cases} 1 & 0 \leq l_{ij,t} \leq \Delta \\ 0 & l_{ij,t} > \Delta \end{cases}. \tag{6.10}$$

Given the edge weights, the true algebraic connectivity $\lambda_2^{\mathbb{L}_t}$ of the multi-robot system is obtained as described earlier in Section 2.2. Note that since the robot positions are stochastic in nature, the algebraic connectivity of the system is also stochastic. Thus, given a lower limit ϵ for the algebraic connectivity, we state the following connectivity maintenance requirement for our trajectory planning algorithm:

$$\Pr(\lambda_2^{\mathbb{L}_t} > \epsilon) \geq 1 - \delta \;\forall\; t \in [0, T], \tag{6.11}$$

i.e., the planner should maintain $\lambda_2^{\mathbb{L}_t}$ above ϵ with a minimum probability value of δ for the planning time horizon T. We specify the values of ϵ and δ chosen for our simulations later in Section 6.4.1.

6.1.3 Trajectory Planning

The objective of the trajectory planner is to plan nominal trajectories for each robot such that they perform local tasks while maintaining connectivity within the multi-robot system. Here the local tasks can represent objectives such as tracking a target, minimizing the control input effort, avoiding collisions, reaching a desired position for exploration, coverage or formation

control, etc. We assume that the following information is available to each robot in the system:

1. The initial beliefs of all robots in the system, i.e., $\mathbf{b}_{i,\text{init}}\ \forall\ i\ \in\ [1, N]$. As defined in Equation (6.8), the initial belief vector consists of the initial state estimate and the initial estimation covariance.

2. The belief dynamics associated with all robots in the system as defined in Equation (6.9).

3. The cost functions representing the local tasks for all robots in the system, i.e., $J_{i,t}(\mathbf{b}_{i,t}, \mathbf{u}_{i,t})\ \forall\ i\ \in\ [1, N], \forall\ t\ \in\ [0, T]$.

The nominal trajectory for each robot i can be represented as a series of nominal beliefs and nominal control inputs $(\check{\mathbf{b}}_{i,0}, \check{\mathbf{u}}_{i,0}, \ldots, \check{\mathbf{b}}_{i,T-1}, \check{\mathbf{u}}_{i,T-1}, \check{\mathbf{b}}_{i,T})$ [47], such that:

$$\check{\mathbf{b}}_{i,t+1} = \mathbf{g}_i(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t})\ \forall\ t \in [0, T-1]. \tag{6.12}$$

We define a concatenated nominal input matrix $\check{U}$, consisting of nominal input vectors of all robots in the system, over the entire planning time horizon:

$$\check{U} = \begin{bmatrix} \check{\mathbf{u}}_{1,0} & \ldots & \check{\mathbf{u}}_{1,T-1} \\ \vdots & \vdots & \vdots \\ \check{\mathbf{u}}_{N,0} & \ldots & \check{\mathbf{u}}_{N,T-1} \end{bmatrix}. \tag{6.13}$$

Note that given the initial beliefs $\mathbf{b}_{i,\text{init}}\ \forall\ i \in [1, N]$, it is sufficient to represent the nominal trajectories for the multi-robot system by $\check{U}$ since the nominal beliefs for each robot can be calculated recursively using Equation (6.12).

Thus, we state the overall objective of the trajectory planner as following:

$$\begin{aligned} \check{U} &= \text{argmin} \sum_{t=0}^{T} \sum_{i=1}^{N} J_{i,t}(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}), \\ &\text{subject to:} \\ &\Pr(\lambda_2^{\mathbb{L}_t} > \epsilon) \geq 1 - \delta\ \forall\ t \in [0, T], \\ &\check{\mathbf{b}}_{i,0} = \mathbf{b}_{i,\text{init}}\ \forall\ i \in [1, N], \\ &\check{\mathbf{b}}_{i,t+1} = \mathbf{g}_i(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t})\ \forall\ i \in [1, N], \forall\ t \in [0, T-1], \end{aligned} \tag{6.14}$$

where the first constraint is the connectivity maintenance requirement stated in Equation (6.11). We use a distributed ADMM-based trajectory planning

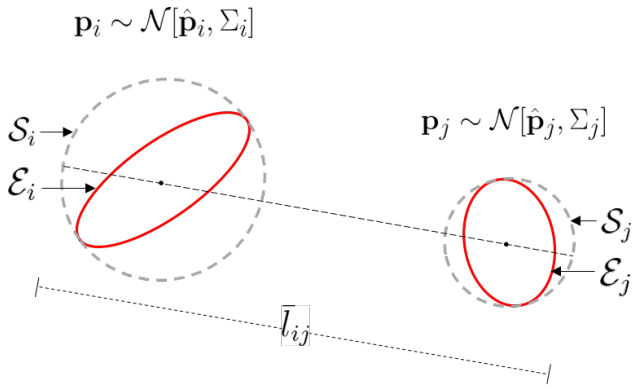


Figure 6.1: Distance measure $\bar{l}_{ij}$ between two robots with Gaussian-distributed positions $\mathbf{p}_i \sim \mathcal{N}(\hat{\mathbf{p}}_i, \Sigma_i)$ and $\mathbf{p}_j \sim \mathcal{N}(\hat{\mathbf{p}}_j, \Sigma_j)$. $\bar{l}_{ij}$ is the maximum distance between the boundaries of the circular regions $\mathcal{S}_i$ and $\mathcal{S}_j$ which overbound the confidence ellipsoids $\mathcal{E}_i$ and $\mathcal{E}_j$ respectively.

algorithm to solve the above planning problem as presented later in Section 6.3.

6.2 Weighted Undirected Graph for Uncertain Robot Positions

In order to address the connectivity maintenance requirement from Equation (6.11), we first define a weighted undirected graph that accounts for uncertain robot positions arising due to the presence of motion and sensing uncertainties. The algebraic connectivity of this graph is then used in our trajectory planning algorithm in Section 6.3. Since the graph definition is applicable for any time instant $t \in [0, T]$, for simplicity we omit the time notations in this section.

Given that the position for each robot $\mathbf{p}_i$ is Gaussian-distributed as $\mathbf{p}_i \sim \mathcal{N}(\hat{\mathbf{p}}_i, \Sigma_i)$, we begin by considering a confidence ellipse $\mathcal{E}_i$ centered at $\hat{\mathbf{p}}_i$ such that:

$$\Pr(\mathbf{p}_i \in \mathcal{E}_i) = 1 - \delta_{\mathcal{E}}, \tag{6.15}$$

where $\delta_{\mathcal{E}}$ is a probability value that decides the size of the confidence ellipse. We derive the value for $\delta_{\mathcal{E}}$ used in our algorithm later in Equation (6.25). Let $\bar{\lambda}^{\Sigma_i}$ represent the largest eigenvalue of the covariance matrix Σ_i. Thus, the length of the semi-major axis of $\mathcal{E}_i$ is $s\sqrt{\bar{\lambda}^{\Sigma_i}}$, where s is a scalar factor that

follows a chi-square distribution [93, 94] based on the value of $\delta_{\mathcal{E}}$. We then define a circular region $\mathcal{S}_i$ centered at $\hat{\mathbf{p}}_i$ with radius $s\sqrt{\bar{\lambda}^{\Sigma_i}}$. This circular region overbounds $\mathcal{E}_i$ and thus, contains $\mathbf{p}_i$ with a probability greater than or equal to $\delta_{\mathcal{E}}$, i.e.:

$$\Pr(\mathbf{p}_i \in \mathcal{S}_i) \geq 1 - \delta_{\mathcal{E}}. \tag{6.16}$$

We then define a distance measure between the boundaries of the overbounding circular regions of two robots i and j as follows:

$$\bar{l}_{ij} = \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2 + s\sqrt{\bar{\lambda}^{\Sigma_i}} + s\sqrt{\bar{\lambda}^{\Sigma_j}}. \tag{6.17}$$

Fig. 6.1 illustrates the confidence ellipses, the overbounding circular regions and the distance measure between two robots.

Given the communication range of Δ between two robots, we introduce a new parameter Δ_0, such that $0 < \Delta_0 < \Delta$. Based on the edge weight defined in [84], we use Δ_0 to define a non-binary edge weight between two robots i and j as:

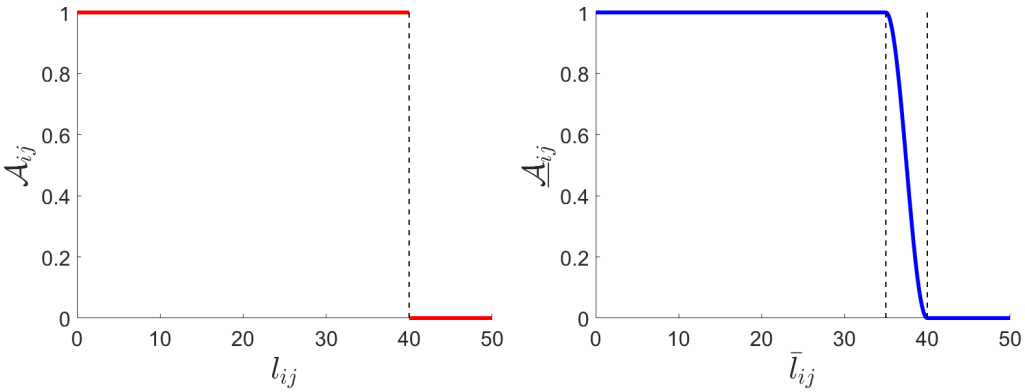
$$\underline{\mathcal{A}}_{ij} = \begin{cases} 1 & 0 \leq \bar{l}_{ij} \leq \Delta_0 \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi(\bar{l}_{ij} - \Delta_0)}{\Delta - \Delta_0}\right) & \Delta_0 < \bar{l}_{ij} \leq \Delta \\ 0 & \bar{l}_{ij} > \Delta \end{cases}. \tag{6.18}$$

Fig. 6.2 compares $\underline{\mathcal{A}}_{ij}$ with $\mathcal{A}_{ij}$ from Equation (6.10). We then proceed to define the corresponding degree matrix $\underline{D} = \text{diag}(\underline{d}_i)$ with $\underline{d}_i = \sum_{j=1}^{n} \underline{\mathcal{A}}_{ij}$ and the corresponding Laplacian matrix $\underline{\mathbb{L}} = \underline{D} - \underline{\mathcal{A}}$. Finally, we use the algebraic connectivity of this constructed weighted undirected graph $\underline{\lambda}_2^{\mathbb{L}}$ as an indicator for the connectivity of the system with uncertain positions.

Next, we proceed to derive the value of $\delta_{\mathcal{E}}$ required in Equation (6.15). We define the following events:

$$\begin{aligned} \mathbb{E}_1 &: \ \mathbf{p}_i \in \mathcal{S}_i \ \forall \ i \in [1, N], \\ \mathbb{E}_2 &: \ \bar{l}_{ij} \geq l_{ij} \ \forall \ \{i, j\} \in [1, N] \times [1, N] \mid i \neq j, \\ \mathbb{E}_3 &: \ \underline{\mathcal{A}}_{ij} \leq \mathcal{A}_{ij} \ \forall \ \{i, j\} \in [1, N] \times [1, N] \mid i \neq j, \\ \mathbb{E}_4 &: \ \lambda_2^{\mathbb{L}} \geq \underline{\lambda}_2^{\mathbb{L}}. \end{aligned}$$

Here $\mathbb{E}_1$ represents the event that all robot positions lie within their corre-



(a) (b)

Figure 6.2: Edge weights between two robots assuming a communication range of $\Delta = 40\,\text{m}$. (a) The binary edge weight $\mathcal{A}_{ij}$ (Equation 6.10) is defined as $\mathcal{A}_{ij} = 1$ if robots i and j are connected, else $\mathcal{A}_{ij} = 0$. (b) For our proposed weighted undirected graph, we define a non-binary edge weight $\underline{\mathcal{A}}_{ij}$ (Equation 6.18) that gradually goes to 0 as the distance measure $\bar{l}_{ij}$ goes from $\Delta_0 = 35\,\text{m}$ to $\Delta = 40\,\text{m}$.

sponding circular regions. We assume that the true positions $\mathbf{p}_i$ of the robots in the system are independent of each other. Thus, using Equation (6.16) we express the probability of event $\mathbb{E}_1$ as:

$$\Pr(\mathbb{E}_1) = \prod_{i=1}^{N} \Pr(\mathbf{p}_i \in \mathcal{S}_i) \geq \prod_{i=1}^{N} (1 - \delta_{\mathcal{E}}) = (1 - \delta_{\mathcal{E}})^N, \tag{6.19}$$

where N is the number of robots in the system.

$\mathbb{E}_2$ represents the event that the distance measures $\bar{l}_{ij}$ between any two robots i and j will always be greater than or equal to the true distance l_{ij}. We proceed to derive the probability of event $\mathbb{E}_2$ as:

$$\begin{aligned} \Pr(\mathbb{E}_2) &= \Pr(\mathbb{E}_2 \mid \mathbb{E}_1) \cdot \Pr(\mathbb{E}_1) + \Pr(\mathbb{E}_2 \mid \mathbb{E}_1') \cdot \Pr(\mathbb{E}_1') \\ &\geq \Pr(\mathbb{E}_2 \mid \mathbb{E}_1) \cdot \Pr(\mathbb{E}_1) = \Pr(\mathbb{E}_1), \end{aligned} \tag{6.20}$$

where $\Pr(\mathbb{E}_2 \mid \mathbb{E}_1) = 1$ since for any two robots i and j if $\mathbf{p}_i \in \mathcal{S}_i$ and $\mathbf{p}_j \in \mathcal{S}_j$, then $\bar{l}_{ij} \geq l_{ij}$ as shown in Fig. 6.1.

$\mathbb{E}_3$ represents the event that the non-binary edge weight $\underline{\mathcal{A}}_{ij}$ from Equation (6.18) is less than the edge weight $\mathcal{A}_{ij}$ from Equation (6.10). Similar to

Equation (6.20), we derive the probability of event $\mathbb{E}_3$ as:

$$\begin{aligned}\Pr(\mathbb{E}_3) &= \Pr(\mathbb{E}_3 \mid \mathbb{E}_2) \cdot \Pr(\mathbb{E}_2) + \Pr(\mathbb{E}_3 \mid \mathbb{E}_2') \cdot \Pr(\mathbb{E}_2') \\ &\geq \Pr(\mathbb{E}_3 \mid \mathbb{E}_2) \cdot \Pr(\mathbb{E}_2) = \Pr(\mathbb{E}_2), \end{aligned} \tag{6.21}$$

where $\Pr(\mathbb{E}_3 \mid \mathbb{E}_2) = 1$ since for any two robots i and j if $\bar{l}_{ij} \geq l_{ij}$, then $\underline{\mathcal{A}}_{ij} \leq \mathcal{A}_{ij}$ as shown in Fig. 6.2.

Finally, $\mathbb{E}_4$ represents the event that the algebraic connectivity of our weighted undirected graph $\underline{\lambda}_2^{\mathbb{L}}$ is less than or equal to the true algebraic connectivity $\lambda_2^{\mathbb{L}}$. The probability of event $\mathbb{E}_4$ is derived as:

$$\begin{aligned}\Pr(\mathbb{E}_4) &= \Pr(\mathbb{E}_4 \mid \mathbb{E}_3) \cdot \Pr(\mathbb{E}_3) + \Pr(\mathbb{E}_4 \mid \mathbb{E}_3') \cdot \Pr(\mathbb{E}_3') \\ &\geq \Pr(\mathbb{E}_4 \mid \mathbb{E}_3) \cdot \Pr(\mathbb{E}_3) = \Pr(\mathbb{E}_3), \end{aligned} \tag{6.22}$$

where $\Pr(\mathbb{E}_4 \mid \mathbb{E}_3) = 1$ since by definition the algebraic connectivity of a graph monotonically increases as the edge weights in the graph increase [80, 84]. Thus, from Equations (6.19)-(6.22) we have:

$$\Pr(\lambda_2^{\mathbb{L}} \geq \underline{\lambda}_2^{\mathbb{L}}) \geq (1 - \delta_{\mathcal{E}})^N, \tag{6.23}$$

which shows that $\underline{\lambda}_2^{\mathbb{L}}$ lower-bounds $\lambda_2^{\mathbb{L}}$ with a minimum probability value of $(1 - \delta_{\mathcal{E}})^N$. If the value of $\underline{\lambda}_2^{\mathbb{L}}$ is maintained above the specified lower limit ϵ from Equation (6.14), i.e., if $\underline{\lambda}_2^{\mathbb{L}} > \epsilon$, then from Equation (6.23) we get:

$$\Pr(\lambda_2^{\mathbb{L}} > \epsilon) \geq (1 - \delta_{\mathcal{E}})^N. \tag{6.24}$$

In order to satisfy the connectivity maintenance requirement described in Equation (6.11), we set $(1 - \delta_{\mathcal{E}})^N = 1 - \delta$, which finally gives us the following value for $\delta_{\mathcal{E}}$:

$$\delta_{\mathcal{E}} = 1 - (1 - \delta)^{(1/N)}, \tag{6.25}$$

where δ is the probability value representing the desired confidence level in Equation (6.11).

To summarize, setting the value of $\delta_{\mathcal{E}}$ from Equation (6.25) and ensuring that our metric $\underline{\lambda}_2^{\mathbb{L}}$ is maintained above ϵ results in satisfying the connectivity maintenance requirement from Equation (6.11). Note that the weighted undirected graph defined in this section can be directly extended to three-

dimensions, where $\mathcal{E}_i$ (Equation (6.15)) would represent a confidence ellipsoid for robot i and $\mathcal{S}_i$ (Equation (6.16)) would represent the corresponding over-bounding spherical region.

6.3 Trajectory Planning For Connectivity Maintenance

In this section, we present the details of our trajectory planning algorithm for solving the problem stated in Equation (6.14). First, we define a connectivity cost function based on the algebraic connectivity $\underline{\lambda}_2^{\mathbb{L}_t}$ of the weighted undirected graph defined in Section 6.2. We incorporate this cost function with the cost in Equation (6.14) to obtain a transformed planning problem. Next, we propose a distributed ADMM setup in order to solve the transformed problem and plan nominal trajectories for the multi-robot system. We then describe the method used for performing the optimization step within the ADMM setup and analyze the complexity of its computational bottleneck. Finally, we develop an approach to reduce the computational load of this optimization step by deriving an approximation for the required Hessian matrices.

6.3.1 Connectivity Cost and Transformed Planning Problem

As discussed earlier in Section 6.2, maintaining $\underline{\lambda}_2^{\mathbb{L}_t}$ above the specified lower limit ϵ enables us to satisfy the connectivity maintenance requirement for the multi-robot system as described in Equation (6.11). Thus, in order to maintain $\underline{\lambda}_2^{\mathbb{L}_t}$ above ϵ, we define a connectivity cost function that grows to infinity as $\underline{\lambda}_2^{\mathbb{L}_t}$ approaches ϵ. Various cost functions with the above property have been proposed in the related works [81] and [84]. For a distributed ADMM setup, it has been shown that the ADMM iteration complexity is inversely proportional to the algebraic connectivity of the system [110]. Thus, we choose to define the connectivity cost function for any time instant t as following:

$$J_t^c = \frac{k_c}{(\underline{\lambda}_2^{\mathbb{L}_t} - \epsilon)} \quad \forall\ \underline{\lambda}_2^{\mathbb{L}_t} > \epsilon, \tag{6.26}$$

where k_c is a parameter that determines the magnitude of the cost function. Fig. 6.3 illustrates the connectivity cost function. In order to incorporate the

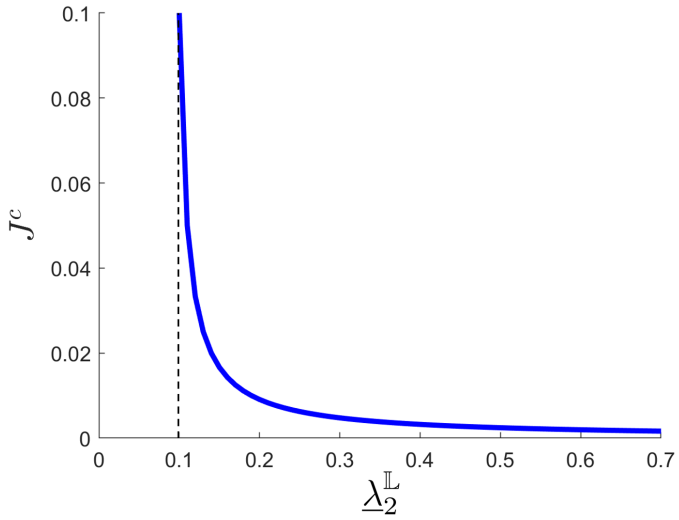


Figure 6.3: The connectivity cost function J^c as a function of the algebraic connectivity $\underline{\lambda}_2^{\mathbb{L}}$ (here $k_c = 0.001$). The cost grows to infinity as $\underline{\lambda}_2^{\mathbb{L}}$ approaches a specified lower limit of $\epsilon = 0.1$ as shown in Equation (26).

connectivity cost function with Equation (6.14), we update original planning problem as follows:

$$\check{U} = \text{argmin} \left(\sum_{t=0}^{T} \sum_{i=1}^{N} J_{i,t}(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}) + \sum_{t=0}^{T} J_t^c \right) \tag{6.27}$$

$$= \text{argmin} \sum_{t=0}^{T} \sum_{i=1}^{N} \left(J_{i,t}(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}) + \frac{1}{N} J_t^c \right) \tag{6.28}$$

$$= \text{argmin} \sum_{t=0}^{T} \sum_{i=1}^{N} \tilde{J}_{i,t}, \tag{6.29}$$

where:

$$\tilde{J}_{i,t} = J_{i,t}(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}) + \frac{1}{N} J_t^c.$$

Thus, we write the transformed planning problem as:

$$\check{U} = \text{argmin} \sum_{t=0}^{T} \sum_{i=1}^{N} \tilde{J}_{i,t}$$

$$\text{subject to:} \tag{6.30}$$

$$\check{\mathbf{b}}_{i,0} = \mathbf{b}_{i,\text{init}} \ \forall \ i \in [1, N],$$

$$\check{\mathbf{b}}_{i,t+1} = \mathbf{g}_i(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}) \ \forall \ i \in [1, N], \forall \ t \in [0, T-1].$$

The difference between Equation (6.14) and the transformed planning problem is that the connectivity maintenance constraint has been incorporated in the cost function. The main reason behind transforming the planning problem from Equation (6.14) to Equation (6.30) is to allow us to use existing optimization tools [47] within the ADMM setup, as will be discussed in Section 6.3.3.

6.3.2 Distributed ADMM Setup

In order to solve the transformed planning problem from Equation (6.30), we implement a distributed ADMM setup [111] that iteratively plans nominal trajectories for the multi-robot system. In each ADMM iteration, each robot optimizes only a subset of the robot trajectories in order to reduce the computational load of the optimization step. The optimized trajectories are then communicated with the rest of the system. After the communication step, each robot updates its local ADMM consensus and dual variables before moving on to the next ADMM iteration. Finally, when the stopping criteria is satisfied, the updated local ADMM consensus variable is used as the planned nominal trajectories for the multi-robot system.

Each robot i begins by generating an initial guess for the nominal trajectories of the multi-robot system $\check{U}^{(i,1)}$, where the superscript denotes that the variable is stored locally on robot i and is for the first ADMM iteration. The initial guess is typically generated based on the local tasks for each robot. We assume that the process used to generate the initial guess maintains $\underline{\lambda}_2^{\mathbb{L}_t}$ above $\epsilon \ \forall \ t \in [0, T]$. Later in Section 6.4.1, we describe our method for obtaining the initial guess when the local task for each robot involves reaching a desired position. Once the initial guess has been generated, the robot proceeds to initialize its local copy of the consensus variable as $\bar{U}^{(i,1)} = \check{U}^{(i,1)}$. The ADMM dual variable $Y^{(i,1)}$ is initialized as a zero matrix.

Next, the robot begins the ADMM iterations. In each ADMM iteration k, the robot first obtains a subset $\mathcal{V}^{(i,k)}$ containing indices of the robot trajectories to optimize. Different strategies can be deployed for obtaining $\mathcal{V}^{(i,k)}$. For example, setting $\mathcal{V}^{(i,k)} = \{i\}$ results in a greedy optimization where the robot optimizes its own trajectory; setting $\mathcal{V}^{(i,k)}$ to contain neighboring robot indices focuses more on the local connectivity rather than the global connec-

tivity of the system. In our algorithm, we obtain $\mathcal{V}^{(i,k)}$ such that it contains i and cycles through the indices of other robots in the system. Let η represent the number of elements in $\mathcal{V}^{(i,k)}$. Table 6.1 shows an example of the subsets $\mathcal{V}^{(i,k)}$ for four ADMM iterations in a system with four robots and with $\eta = 3$. We observe that this strategy for obtaining $\mathcal{V}^{(i,k)}$ avoids the problem of greedy optimizations and eventually results in nominal trajectories for the system with lower overall costs as shown in Section 6.4.

	$\mathcal{V}^{(1,k)}$	$\mathcal{V}^{(2,k)}$	$\mathcal{V}^{(3,k)}$	$\mathcal{V}^{(4,k)}$
$k=1$	$\{1,2,3\}$	$\{2,3,4\}$	$\{3,4,1\}$	$\{4,1,2\}$
$k=2$	$\{1,3,4\}$	$\{2,4,1\}$	$\{3,1,2\}$	$\{4,2,3\}$
$k=3$	$\{1,4,2\}$	$\{2,1,3\}$	$\{3,2,4\}$	$\{4,3,1\}$
$k=4$	$\{1,2,3\}$	$\{2,3,4\}$	$\{3,4,1\}$	$\{4,1,2\}$

Table 6.1: Example of subsets $\mathcal{V}$ for a system with four robots, where $\eta = 3$ and up to $k = 4$ ADMM iterations are considered.

Once the subset $\mathcal{V}^{(i,k)}$ has been obtained, the robot performs the optimization step. In this step, we first initialize $\check{U}^{(i,k+1)} = \bar{U}^{(i,k)}$ and then only update the trajectories for robots in the subset $\mathcal{V}^{(i,k)}$. Based on the cost function in Equation (6.30), the robot optimizes the following augmented cost [111] in order to obtain the optimized trajectories for robots in $\mathcal{V}^{(i,k)}$:

$$\begin{aligned}\check{U}^{(i,k+1)}_{\mathcal{V}^{(i,k)}} = \underset{\check{U}_{\mathcal{V}^{(i,k)}}}{\operatorname{argmin}} \sum_{t=0}^{T} \Bigg\{ & \sum_{j \in \mathcal{V}^{(i,k)}} \tilde{J}_{j,t} \\ & + \mathbf{y}^{(i,k)\top}_{\mathcal{V}^{(i,k)},t} \left(\check{\mathbf{u}}_{\mathcal{V}^{(i,k)},t} - \bar{\mathbf{u}}^{(i,k)}_{\mathcal{V}^{(i,k)},t} \right) \\ & + (\rho/2) \left\| \check{\mathbf{u}}_{\mathcal{V}^{(i,k)},t} - \bar{\mathbf{u}}^{(i,k)}_{\mathcal{V}^{(i,k)},t} \right\|_2^2 \Bigg\},\end{aligned} \tag{6.31}$$

subject to:

$$\check{\mathbf{b}}_{i,0} = \mathbf{b}_{i,\text{init}} \ \forall \ i \in [1, N],$$
$$\check{\mathbf{b}}_{i,t+1} = \mathbf{g}_i(\check{\mathbf{b}}_{i,t}, \check{\mathbf{u}}_{i,t}) \ \forall \ i \in [1, N], \forall \ t \in [0, T-1],$$

where $\rho > 0$ is the ADMM penalty weight, $\check{\mathbf{u}}$ and $\bar{\mathbf{u}}^{(i,k)}$ represent the corresponding vectors from matrices $\check{U}$ and $\bar{U}^{(i,k)}$ respectively, and $\mathbf{y}^{(i,k)}$ represents the corresponding vector from dual variable matrix $Y^{(i,k)}$. We discuss the method used to solve this optimization step later in Section 6.3.3.

After the optimization step, the robot proceeds to the communication step where each robot i shares the optimized trajectories $\check{U}^{(i,k+1)}_{\mathcal{V}^{(i,k)}}$ that it obtained from Equation (6.31). In this step, each robot receives the optimized trajectories from all other robots in the system, potentially via multi-hop communication. Later in Section 6.4.3, we account for a delay due to the communication step while evaluating our planner under time constraints. Note that our planner is distributed since each robot optimizes with respect to a subset of trajectories. However, it is not decentralized since each robot shares information with all other robots instead of only its neighbors.

Once the communication step is complete, each robot i receives the optimized trajectories from all other robots. Note that the trajectory for each robot has been optimized η times across the system. For example, in Table 6.1 at ADMM iteration $k = 3$, the trajectory for robot 4 was optimized by robots $1, 3$ and 4, i.e., $\eta = 3$ times. Thus, based on the consensus update step in [111], the robot calculates an average optimized trajectory for each robot j as follows:

$$\tilde{U}^{(i,k+1)}_j = \frac{1}{\eta} \sum_{l=1}^{N} \check{U}^{(l,k+1)}_j \cdot \mathbb{1}_{\mathcal{V}^{(l,k)}}(j), \tag{6.32}$$

where $\mathbb{1}_{\mathcal{V}^{(l,k)}}(j)$ is an indicator function equal to 1 if $j \in \mathcal{V}^{(l,k)}$ and equal to 0 otherwise.

After the averaging step, it is possible that $\tilde{U}^{(i,k+1)}$ might result in a trajectory for the multi-robot system that does not maintain $\underline{\lambda}_2^{\mathbb{L}_t}$ above the specified lower limit of ϵ. Thus, in order to ensure that the consensus variable $\bar{U}$ always results in trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$, we use a line search algorithm [112] to update $\bar{U}$. We limit the change to $\bar{U}$ between iterations as follows:

$$\bar{U}^{(i,k+1)} = \bar{U}^{(i,k)} + \beta \cdot (\tilde{U}^{(i,k+1)} - \bar{U}^{(i,k)}), \tag{6.33}$$

where β is a parameter that determines the amount of change in $\bar{U}$. We begin with $\beta = 1$ and check if the corresponding $\bar{U}^{(i,k+1)}$ results in trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$. If $\underline{\lambda}_2^{\mathbb{L}_t}$ is not maintained above ϵ, we reduce β by a factor γ as: $\beta = \gamma \cdot \beta$, where $0 < \gamma < 1$. We then calculate the new $\bar{U}^{(i,k+1)}$ using Equation (6.33) and repeat the process until $\bar{U}^{(i,k+1)}$ results in trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$. Since our initial nominal trajectory guess maintains $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$, the line search in Equation (6.33) ensures that $\bar{U}$ always results in

trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$.

Note that the averaging step in Equation (6.32) and the consensus update in Equation (6.33) can be performed on a *central* robot, similar to the approach in [113]. However, dependence on a *central* robot makes the algorithm susceptible to potential single points of failures in the multi-robot system. Additionally, the computational load for Equations (6.32) and (6.33) are relatively low since they do not require any optimizations and hence can be quickly performed on every robot. Finally, each robot i updates its ADMM dual variable as follows:

$$Y^{(i,k+1)} = Y^{(i,k)} + \rho \cdot (\check{U}^{(i,k+1)} - \bar{U}^{(i,k+1)}), \tag{6.34}$$

where ρ is the ADMM penalty weight defined in Equation (6.31).

Before beginning the next ADMM iteration, the robot checks if the stopping criterion has been satisfied. The stopping criteria can be either convergence-based, which is typical for applications when the planning is done in an *offline* manner, or time-based, which is typical for *online* planning applications. If the stopping criteria is satisfied, the robots set the last updated value of $\bar{U}$ as the planned nominal trajectories for the multi-robot system. Since $\bar{U}$ always results in trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$, the output from our trajectory planning algorithm always results in trajectories that maintain $\underline{\lambda}_2^{\mathbb{L}_t} > \epsilon$. Thus, our algorithm satisfies the connectivity maintenance requirement from Equation (6.11). Algorithm 3 summarizes our trajectory planning algorithm.

6.3.3 ADMM Trajectory Optimization and Complexity Analysis

In order to obtain the optimized nominal trajectories $\check{U}_{\mathcal{V}^{(i,k)}}^{(i,k+1)}$ in Equation (6.31), we use the belief-space iterative Linear Quadratic Gaussian (belief-space iLQG) method [47]. Since the analysis in the remainder of this section is applicable for a general subset of robot trajectories, we simply represent $\mathcal{V}^{(i,k)}$ as $\mathcal{V}$. We first extend Equation (6.9) to define concatenated belief dynamics for the subset $\mathcal{V}$ as follows:

$$\mathbf{b}_{\mathcal{V},t+1} = \mathbf{g}_{\mathcal{V}}(\mathbf{b}_{\mathcal{V},t}, \mathbf{u}_{\mathcal{V},t}) + M_{\mathcal{V}}(\mathbf{b}_{\mathcal{V},t}, \mathbf{u}_{\mathcal{V},t})\mathbf{m}_{\mathcal{V},t}, \tag{6.35}$$

Algorithm 3 Trajectory planner

for $i = 1, \ldots, N$ **do** in parallel
  Generate initial nominal trajectory guess $\check{U}^{(i,1)}$
  Initialize consensus variable $\bar{U}^{(i,1)} = \check{U}^{(i,1)}$, dual variable $Y^{(i,1)}$ as zero matrix, and ADMM iteration $k = 1$
  while stopping criterion is not satisfied **do**
    Obtain subset $\mathcal{V}^{(i,k)}$ of trajectories to optimize
    Perform the optimization step (Equation (6.31)) to obtain $\check{U}^{(i,k+1)}_{\mathcal{V}^{(i,k)}}$
    Communicate $\check{U}^{(i,k+1)}_{\mathcal{V}^{(i,k)}}$ to (and from) other robots
    Calculate average optimized trajectories (Equation (6.32)) to obtain $\tilde{U}^{(i,k+1)}$
    Update consensus variable $\bar{U}^{(i,k+1)}$ using line search algorithm (Equation (6.33))
    Update dual variable $Y^{(i,k+1)}$ (Equation (6.34))
    Update ADMM iteration $k = k + 1$
  end while
  Set planned nominal trajectories as $\check{U} = \bar{U}^{(i,k)}$
end for

where $\mathbf{b}_{\mathcal{V}}$ is the concatenated belief vector of robots in the subset $\mathcal{V}$. Next, similar to Equation (6.12), the concatenated nominal trajectory for the subset $\mathcal{V}$ is represented as $(\check{\mathbf{b}}_{\mathcal{V},0}, \check{\mathbf{u}}_{\mathcal{V},0}, \ldots, \check{\mathbf{b}}_{\mathcal{V},T-1}, \check{\mathbf{u}}_{\mathcal{V},T-1}, \check{\mathbf{b}}_{\mathcal{V},T})$, such that:

$$\check{\mathbf{b}}_{\mathcal{V},t+1} = \mathbf{g}_{\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}) \; \forall \; t \in [0, T-1]. \tag{6.36}$$

Additionally, we rewrite the ADMM optimization step in Equation (6.31) as:

$$\begin{aligned} \check{U}^{(i,k+1)}_{\mathcal{V}} &= \underset{\check{U}_{\mathcal{V}}}{\operatorname{argmin}} \sum_{t=0}^{T} c_t(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}), \\ &\text{subject to:} \\ &\check{\mathbf{b}}_{\mathcal{V},0} = \mathbf{b}_{\mathcal{V},\text{init}}, \\ &\check{\mathbf{b}}_{\mathcal{V},t+1} = \mathbf{g}_{\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}) \; \forall \; t \in [0, T-1], \end{aligned} \tag{6.37}$$

where c_t is the cost at time instant t. While c_t depends on the belief and input vectors of the entire multi-robot system, for simplicity we write c_t to be a function of only $\check{\mathbf{b}}_{\mathcal{V}}$ and $\check{\mathbf{u}}_{\mathcal{V}}$ since only the trajectories for subset $\mathcal{V}$ are being optimized. The belief-space iLQG method [47] begins with an initial guess for the nominal trajectory and computes a locally optimal solution

for Equation (6.37) by performing backward value iteration. The value iteration process involves quadratizing the cost function c_t along the nominal trajectory as follows:

$$c_t \approx \frac{1}{2}\begin{bmatrix}\delta\mathbf{b}\\ \delta\mathbf{u}\end{bmatrix}^\top \begin{bmatrix}\check{c}_{\mathbf{bb},t} & \check{c}_{\mathbf{bu},t}^\top\\ \check{c}_{\mathbf{bu},t} & \check{c}_{\mathbf{uu},t}\end{bmatrix}\begin{bmatrix}\delta\mathbf{b}\\ \delta\mathbf{u}\end{bmatrix} + \begin{bmatrix}\delta\mathbf{b}\\ \delta\mathbf{u}\end{bmatrix}^\top \begin{bmatrix}\check{\mathbf{c}}_{\mathbf{b},t}\\ \check{\mathbf{c}}_{\mathbf{u},t}\end{bmatrix} + \check{c}_t, \tag{6.38}$$

where:

$$\begin{aligned}
\delta\mathbf{b} &= \mathbf{b}_{\mathcal{V},t} - \check{\mathbf{b}}_{\mathcal{V},t}, & \delta\mathbf{u} &= \mathbf{u}_{\mathcal{V},t} - \check{\mathbf{u}}_{\mathcal{V},t},\\
\check{c}_{\mathbf{bb},t} &= \frac{\partial^2 c_t}{\partial\mathbf{b}_\mathcal{V}\partial\mathbf{b}_\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}), & \check{c}_{\mathbf{bu},t} &= \frac{\partial^2 c_t}{\partial\mathbf{b}_\mathcal{V}\partial\mathbf{u}_\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}),\\
\check{c}_{\mathbf{uu},t} &= \frac{\partial^2 c_t}{\partial\mathbf{u}_\mathcal{V}\partial\mathbf{u}_\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}), & \check{\mathbf{c}}_{\mathbf{b},t} &= \frac{\partial c_t}{\partial\mathbf{b}_\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}),\\
\check{\mathbf{c}}_{\mathbf{u},t} &= \frac{\partial c_t}{\partial\mathbf{u}_\mathcal{V}}(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}), & \check{c}_t &= c_t(\check{\mathbf{b}}_{\mathcal{V},t}, \check{\mathbf{u}}_{\mathcal{V},t}).
\end{aligned}$$

As discussed in previous works related to belief-space iLQG [47, 22], one of the primary sources of a computational bottleneck lies in the computation of the Hessian $\check{c}_{\mathbf{bb},t}$. In our trajectory planner, from Equations (6.27)-(6.31) and (6.37), note that the cost function c_t consists of the connectivity cost function J_t^c. Thus, computation of $\check{c}_{\mathbf{bb},t}$ involves computing the Hessian of the connectivity cost function $\check{J}^c_{\mathbf{bb},t} = \frac{\partial^2 J_t^c}{\partial\mathbf{b}_\mathcal{V}\partial\mathbf{b}_\mathcal{V}}(\underline{\lambda}_2^{\mathbb{L}_t}(\check{\mathbf{b}}_{\mathcal{V},t}))$. For our belief-space iLQG implementation, we observe that computing $\check{J}^c_{\mathbf{bb},t}$ is the primary computational bottleneck. Thus, we analyze its complexity below.

For simplicity, we assume that the state vector $\mathbf{x}_{i,t}$ has the same dimension n for all robots in the system throughout the planning horizon. In this case, the dimension of the belief vector for each robot, as defined in Equation (6.8), is $O(n^2)$ since it contains elements from the state estimation covariance matrix. Thus, the dimension of the concatenated belief vector $\mathbf{b}_\mathcal{V}$ is $O(\eta n^2)$, since $\mathcal{V}$ contains η elements. Given the dimension of $\mathbf{b}_\mathcal{V}$, the Hessian $\check{J}^c_{\mathbf{bb},t}$ contains $O(\eta^2 n^4)$ entries. The typical approach to compute the required Hessian in previous belief-space iLQG implementations is to use numerical differentiation (central differences) [47, 22]. Using numerical differentiation would require $O(\eta^2 n^4)$ evaluations of J_t^c, and consequently $O(\eta^2 n^4)$ evaluations of $\underline{\lambda}_2^{\mathbb{L}_t}$. Considering the entire planning horizon, this results in $O(\eta^2 n^4 T)$ evaluations of $\underline{\lambda}_2^{\mathbb{L}_t}$ per iteration of belief-space iLQG.

Evaluating $\underline{\lambda}_2^{\mathbb{L}_t}$ requires obtaining the Laplacian matrix $\underline{\mathbb{L}}_t$ whose elements depend on the distance measure as shown in Equation (6.18). For obtaining the distance measures between all robots in the system, we need to perform eigendecompositions of the covariance matrices $\Sigma_{i,t}\ \forall\ i \in [1, N]$ as shown in Equation (6.17). Assuming the dimension of the position vector $\mathbf{p}_{i,t}$ to be ϱ, each eigendecomposition can be evaluated in $O(\varrho^3)$ time [114]. Thus, the complexity to obtain $\underline{\mathbb{L}}_t$ is of $O(\varrho^3 N)$. Once we obtain $\underline{\mathbb{L}}_t$, we need to perform another eigendecomposition with complexity of $O(N^3)$ to obtain $\underline{\lambda}_2^{\mathbb{L}_t}$. Thus, the complexity of a single evaluation of $\underline{\lambda}_2^{\mathbb{L}_t}$ is of $O(\max(\varrho^3 N, N^3))$. Given that we need $O(\eta^2 n^4 T)$ evaluations of $\underline{\lambda}_2^{\mathbb{L}_t}$, we finally have a complexity of $O(\eta^2 n^4 T \cdot \max(\varrho^3 N, N^3))$ per iteration of belief-space iLQG.

Since the belief-space iLQG method is used for the optimization step within each ADMM iteration, using numerical differentiation to compute $\check{J}^c_{\mathbf{bb},t}$ results in a prohibitively large computational load. Thus, in the next subsection we develop an approach to approximate $\check{J}^c_{\mathbf{bb},t}$ and consequently reduce the required computational load for the belief-space iLQG method.

6.3.4 Hessian Approximation for Complexity Reduction

In this subsection, we drop the time notation for simplicity since our approximation is applicable $\forall\ t \in [0, T]$. As discussed in Section 6.3.3, the primary computational bottleneck in our implementation of belief-space iLQG arises in computing $\check{J}^c_{\mathbf{bb},t}$. Thus, in this subsection we derive an analytical expression to approximate $\check{J}^c_{\mathbf{bb},t}$ and show that it significantly reduces the required computational load.

We begin by obtaining the gradient of our metric $\underline{\lambda}_2^{\mathbb{L}}$ with respect to the belief vector $\mathbf{b}_i$ as follows [80]:

$$\frac{\partial \underline{\lambda}_2^{\mathbb{L}}}{\partial \mathbf{b}_i} = (\underline{\mathbf{e}}_2^{\mathbb{L}})^\top \frac{\partial \underline{\mathbb{L}}}{\partial \mathbf{b}_i} (\underline{\mathbf{e}}_2^{\mathbb{L}}) = \sum_{j=1}^{N} \frac{\partial \underline{\mathcal{A}}_{ij}}{\partial \mathbf{b}_i} \left(\underline{e}_2^{\mathbb{L},(i)} - \underline{e}_2^{\mathbb{L},(j)} \right)^2, \tag{6.39}$$

where $\underline{\mathbf{e}}_2^{\mathbb{L}}$ is the eigenvector of $\underline{\mathbb{L}}$ corresponding to the eigenvalue $\underline{\lambda}_2^{\mathbb{L}}$, and $\underline{e}_2^{\mathbb{L},(i)}$ is the i^{th} element of $\underline{\mathbf{e}}_2^{\mathbb{L}}$. From Equation (6.18), we obtain the gradient of $\underline{\mathcal{A}}_{ij}$

with respect to the belief vector $\mathbf{b}_i$ as:

$$\frac{\partial \underline{\mathcal{A}}_{ij}}{\partial \mathbf{b}_i} = -\frac{\pi}{2(\Delta - \Delta_0)} \sin\left(\frac{\pi(\bar{l}_{ij} - \Delta_0)}{\Delta - \Delta_0}\right) \frac{\partial \bar{l}_{ij}}{\partial \mathbf{b}_i}. \tag{6.40}$$

Note that while the belief vector $\mathbf{b}_i$ contains the state estimate and the estimation covariance (Equation (6.8)), the distance measure $\bar{l}_{ij}$ depends only on the position estimate $\hat{\mathbf{p}}_i$ and the position estimation covariance Σ_i (Equation (6.17)). Thus, in order to obtain $\frac{\partial \bar{l}_{ij}}{\partial \mathbf{b}_i}$ in Equation (6.40), we only need the gradient of $\bar{l}_{ij}$ with respect to each element of $\hat{\mathbf{p}}_i$ and with respect to each element of Σ_i. From Equation (6.17), the gradient of $\bar{l}_{ij}$ with respect to $\hat{p}_i^{(m)}$, i.e., the m^{th} element of $\hat{\mathbf{p}}_i$, is computed as:

$$\frac{\partial \bar{l}_{ij}}{\partial \hat{p}_i^{(m)}} = \frac{\left(\hat{p}_i^{(m)} - \hat{p}_j^{(m)}\right)}{\|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2^2}, \tag{6.41}$$

and the gradient of $\bar{l}_{ij}$ with respect to element (m, b) of Σ_i is computed as [115]:

$$\frac{\partial \bar{l}_{ij}}{\partial \Sigma_i^{(m,b)}} = \left(\frac{s}{2\sqrt{\bar{\lambda}^{\Sigma_i}}}\right) \mathtt{tr}\left(\left(\frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \Sigma_i}\right)^\top \left(\frac{\partial \Sigma_i}{\partial \Sigma_i^{(m,b)}}\right)\right), \tag{6.42}$$

where $\mathtt{tr}(\cdot)$ represents the trace of a matrix. Furthermore, from [115], we get:

$$\frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \Sigma_i} = \frac{(\bar{\mathbf{e}}^{\Sigma_i})(\bar{\mathbf{e}}^{\Sigma_i})^\top}{(\bar{\mathbf{e}}^{\Sigma_i})^\top (\bar{\mathbf{e}}^{\Sigma_i})}, \tag{6.43}$$

where $\bar{\mathbf{e}}^{\Sigma_i}$ is the eigenvector of Σ_i corresponding to the largest eigenvalue $\bar{\lambda}^{\Sigma_i}$. Thus, Equation (6.42) simplifies to:

$$\frac{\partial \bar{l}_{ij}}{\partial \Sigma_i^{(m,b)}} = \left(\frac{s}{2\sqrt{\bar{\lambda}^{\Sigma_i}}}\right) \bar{e}^{\Sigma_i,(m)} \bar{e}^{\Sigma_i,(b)}. \tag{6.44}$$

Equations (6.41) and (6.44) allow us to construct the gradient $\frac{\partial \bar{l}_{ij}}{\partial \mathbf{b}_i}$, which is required to obtain $\frac{\partial \bar{\lambda}_2^{\mathbb{L}}}{\partial \mathbf{b}_i}$ using Equations (6.39) and (6.40). By concatenating the gradients $\frac{\partial \bar{\lambda}_2^{\mathbb{L}}}{\partial \mathbf{b}_i}$ $\forall\ i \in \mathcal{V}$, we obtain the gradient $\frac{\partial \bar{\lambda}_2^{\mathbb{L}}}{\partial \mathbf{b}_\mathcal{V}}$.

Next, in order to approximate $\check{J}^c_{\mathbf{bb}}$, we begin by writing the second-order

Taylor expansion of J^c about $\check{\mathbf{b}}_\mathcal{V}$:

$$J^c(\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V})) \approx \frac{1}{2}\check{J}^c_{\lambda\lambda}(\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V}) - \underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V}))^2 + \check{J}^c_\lambda(\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V}) - \underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V})) + \check{J}^c, \quad (6.45)$$

where:

$$\check{J}^c_{\lambda\lambda} = \frac{\partial^2 J^c}{\partial \underline{\lambda}_2^{\mathbb{L}} \partial \underline{\lambda}_2^{\mathbb{L}}}(\underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V})), \quad \check{J}^c_\lambda = \frac{\partial J^c}{\partial \underline{\lambda}_2^{\mathbb{L}}}(\underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V})),$$
$$\check{J}^c = J^c(\underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V})).$$

Here $\check{J}^c_{\lambda\lambda}$ is obtained from Equation (6.26) as:

$$\check{J}^c_{\lambda\lambda} = \frac{2k_c}{(\underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V}) - \epsilon)^3}. \quad (6.46)$$

We then approximate the term $(\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V}) - \underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V}))$ in Equation (6.45) using a first-order Taylor expansion about $\check{\mathbf{b}}_\mathcal{V}$ as follows:

$$\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V}) - \underline{\lambda}_2^{\mathbb{L}}(\check{\mathbf{b}}_\mathcal{V}) \approx (\mathbf{b}_\mathcal{V} - \check{\mathbf{b}}_\mathcal{V})^\top \mathbf{a}, \quad (6.47)$$

where $\mathbf{a} = \left(\frac{\partial \underline{\lambda}_2^{\mathbb{L}}}{\partial \mathbf{b}_\mathcal{V}}(\check{\mathbf{b}}_\mathcal{V})\right)^\top$. By substituting Equation (6.47) in Equation (6.45), we get:

$$J^c(\underline{\lambda}_2^{\mathbb{L}}(\mathbf{b}_\mathcal{V})) \approx \frac{1}{2}(\mathbf{b}_\mathcal{V} - \check{\mathbf{b}}_\mathcal{V})^\top (\check{J}^c_{\lambda\lambda}\mathbf{a}\mathbf{a}^\top)(\mathbf{b}_\mathcal{V} - \check{\mathbf{b}}_\mathcal{V}) + (\mathbf{b}_\mathcal{V} - \check{\mathbf{b}}_\mathcal{V})^\top (\check{J}^c_\lambda \mathbf{a}) + \check{J}^c, \quad (6.48)$$

where $(\check{J}^c_{\lambda\lambda}\mathbf{a}\mathbf{a}^\top)$ is an approximation for $\check{J}^c_{\mathbf{bb}}$. Note that in order to compute the above approximation for $\check{J}^c_{\mathbf{bb}}$, we require only a single evaluation of $\underline{\lambda}_2^{\mathbb{L}}$ in Equation (6.46). Considering the entire planning horizon, this results in only T evaluations of $\underline{\lambda}_2^{\mathbb{L}}$ per iteration of belief-space iLQG. This is in contrast to using numerical differentiation which requires $O(\eta^2 n^4 T)$ evaluations as discussed in Section 6.3.3. Thus, approximating $\check{J}^c_{\mathbf{bb}}$ with $(\check{J}^c_{\lambda\lambda}\mathbf{a}\mathbf{a}^\top)$ significantly reduces the computational load of the belief-space iLQG method.

6.4 Simulation Results

In this section, we evaluate our trajectory planning algorithm by simulating multiple missions for a multi-UAV system. We first describe the simulation setup which includes the UAV motion and sensing models, the cost functions representing the local tasks, the method used for generating initial trajectory guesses, and the values used for parameters within the planner. We then discuss the performance of our planner for two types of planning applications: *offline* and *online*. For *offline* planning we focus on the convergence of the planner, whereas for *online* planning we focus on the planner performance under time constraints. Additionally, for both applications we statistically validate the connectivity maintenance of our trajectory planning algorithm. All simulations in this section are performed on a 2.80 GHz Quad-core Intel™ i7 machine.

6.4.1 Simulation Setup

For each UAV in the multi-UAV system, we consider a 2-dimensional (2D) double integrator model as the motion model. The UAV state vector contains the 2D position and velocity, i.e, $\mathbf{x}_{i,t} = \begin{bmatrix} \mathbf{p}_{i,t}^\top & \dot{\mathbf{p}}_{i,t}^\top \end{bmatrix}^\top$, and the input vector is the UAV accelerations. The motion model for the UAV can be written in the form of Equation (6.1) as:

$$\mathbf{x}_{i,t} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{i,t-1} + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ dt & 0 \\ 0 & dt \end{bmatrix} \mathbf{u}_{i,t-1} + \mathbf{w}_{i,t}, \tag{6.49}$$

where dt is the time-step between two time instants and

$$\mathbf{w}_{i,t} \sim \mathcal{N} \left(\mathbf{0}, 0.1\,\mathrm{m^2 s^{-3}} \begin{bmatrix} \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \\ \frac{dt^2}{2} & 0 & dt & 0 \\ 0 & \frac{dt^2}{2} & 0 & dt \end{bmatrix} \right).$$

It is important to choose an appropriately small time-step dt such that the system connectivity along the discretized trajectory represents the system

connectivity in continuous time. For our simulations we choose $dt = 0.2\,\mathrm{s}$ which also reflects the rate of common sensor measurements such as global positioning system or visual-based localization measurements. Additionally, we set a maximum limit of $5\,\mathrm{m\,s^{-2}}$ on the magnitude of input accelerations. Note that in contrast to a majority of previous connectivity maintenance works, the above motion model does not assume that the UAVs can instantaneously change their direction of motion. For the sensing model in Equation (6.2), we consider position measurements:

$$\mathbf{z}_{i,t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_{i,t} + \mathbf{v}_{i,t}, \tag{6.50}$$

where

$$\mathbf{v}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1\,\mathrm{m}^2 & 0 \\ 0 & 1\,\mathrm{m}^2 \end{bmatrix}\right).$$

Additionally, we assume that each UAV operates at a different altitude, similar to [22]. We make this assumption in order to alleviate inter-UAV collision constraints and focus on the connectivity maintenance of the system.

As mentioned in Section 6.1.3, each UAV i is considered to have a local task represented by a cost function $J_{i,t}$. For our simulations we consider the local task of reaching a desired position along with minimizing the control input effort. Thus, we use the following cost functions:

$$J_{i,T} = (\hat{\mathbf{x}}_{i,T} - \mathbf{x}_{i,\mathrm{des}})^\top W_i^{\mathbf{x}} (\hat{\mathbf{x}}_{i,T} - \mathbf{x}_{i,\mathrm{des}}), \tag{6.51}$$

$$J_{i,t} = \mathbf{u}_{i,t}^\top W_i^{\mathbf{u}} \mathbf{u}_{i,t}, \tag{6.52}$$

where $\mathbf{x}_{i,\mathrm{des}} = \begin{bmatrix} \mathbf{p}_{i,\mathrm{des}}^\top & \dot{\mathbf{p}}_{i,\mathrm{des}}^\top \end{bmatrix}^\top$ contains the desired position $\mathbf{p}_{i,\mathrm{des}}$ and desired velocity $\dot{\mathbf{p}}_{i,\mathrm{des}}$ for the UAV, and $W_i^{\mathbf{x}}$ and $W_i^{\mathbf{u}}$ are used to set the relative importance of the different costs. For our simulations we set $\dot{\mathbf{p}}_{i,\mathrm{des}} = \mathbf{0}$.

We specify an initial position $\mathbf{p}_{i,\mathrm{init}}$ for each UAV i and set their initial velocities to be zeros, i.e., the initial state vector is $\mathbf{x}_{i,\mathrm{init}} = \begin{bmatrix} \mathbf{p}_{i,\mathrm{init}}^\top & \mathbf{0}^\top \end{bmatrix}^\top$.

The initial state estimation covariance $P_{i,\text{init}}$ is set as:

$$P_{i,\text{init}} = \begin{bmatrix} 0.1\,\text{m}^2 & 0 & 0 & 0 \\ 0 & 0.1\,\text{m}^2 & 0 & 0 \\ 0 & 0 & 0.001\,\text{m}^2\text{s}^{-2} & 0 \\ 0 & 0 & 0 & 0.001\,\text{m}^2\text{s}^{-2} \end{bmatrix}. \tag{6.53}$$

For our simulations, we consider the system to be comprised of two types of UAVs: *primary* and *bridge*. The local task for *primary* UAVs includes both reaching the desired position and minimizing control input effort, thus, we set:

$$W_i^{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, \quad W_i^{\mathbf{u}} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix},$$

for each UAV i that is a *primary* UAV. On the other hand, *bridge* UAVs are not assigned a desired position and hence focus on arranging themselves in order to maintain connectivity within the system. Thus for each UAV i that is a *bridge* UAV, we set:

$$W_i^{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, \quad W_i^{\mathbf{u}} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

For the initial trajectory guess, we require a computationally inexpensive method of generating a trajectory based on the local tasks for each UAV. For example, sampling-based planners such as rapidly-exploring random trees (RRTs) can be used in order to quickly plan trajectories around obstacles [47, 116]; in a multiple target tracking application, each UAV can be randomly assigned to track a separate target [22]. In our algorithm, we use a Linear-Quadratic-Regulator (LQR) to obtain an initial trajectory guess for each *primary* UAV i from $\mathbf{x}_{i,\text{init}}$ to $\mathbf{x}_{i,\text{des}}$. For *bridge* UAVs, we simply set the initial trajectory guess to be hovering at the initial position. If $\underline{\lambda}_2^{\mathbb{L}_t}$ is not maintained above ϵ for the resulting trajectory guess of the multi-UAV system, we consider new desired states (only for the initial trajectory guess and not for the rest of the planner) midway between $\mathbf{x}_{i,\text{init}}$ and the $\mathbf{x}_{i,\text{des}}$ for

all *primary* UAVs and repeat the process.

We consider a communication range of $\Delta = 40\,\text{m}$ and set the parameter $\Delta_0 = 35\,\text{m}$ in Equation (6.18). For the connectivity maintenance requirement in Equation (6.11), we specify the lower algebraic connectivity limit $\epsilon = 0.1$ and the corresponding probability value $\delta = 0.003$ to reflect a 3σ confidence level. We set $T = 250$, which results in a planning time horizon of 50 s. In Equation (6.26), we set the parameter for the magnitude of the connectivity cost as $k_c = 0.001$. We set $\eta = 2$ for the number of elements in subsets $\mathcal{V}$ obtained in the trajectory planner. Finally, for the line search algorithm used to update the ADMM consensus variable in Equation (6.33), we set $\gamma = 0.8$.

6.4.2 Offline Planning

For *offline* planning applications, we simulate two missions for a system with five UAVs. For the first *offline* mission we consider all five UAVs to be *primary* UAVs. Here we set the desired positions for the UAVs at a sufficient relative distance such that the system connectivity would not be maintained if each UAV reaches its desired position. Thus, the planner attempts to find trajectories such that each UAV reaches as close as possible to its desired position while still satisfying the connectivity requirement from Equation (6.11). Fig. 6.4 shows the results for the first *offline* mission.

Figs. 6.4(a)-(d) show the process of convergence of our trajectory planning algorithm. The planner begins by generating an initial trajectory guess using the LQR-based method discussed in Section 6.4.1. Fig. 6.4(d) shows the final planned trajectories obtained after the convergence-based stopping criteria has been satisfied. Additionally, we simulate 1000 trajectory rollouts for the multi-UAV system where each UAV tracks the planned trajectory using a LQR controller. The simulated rollouts are later used to statistically validate the system connectivity. In Fig. 6.4(e), we show the convergence of the cost in the transformed optimization problem from Equation (6.30).

Fig. 6.4(f) shows the connectivity maintenance of the system along the planned trajectories. Given that we incorporated the connectivity cost function (Equation (6.26)) in the transformed optimization problem (Equation (6.30)), the planned trajectories maintain $\underline{\lambda}_2^{\mathbb{L}}$ above the lower limit ϵ. As discussed in Section 6.2, maintaining $\underline{\lambda}_2^{\mathbb{L}} > \epsilon$ results in satisfying the connectivity main-

tenance requirement described in Equation (6.11). In order to statistically validate that this requirement is satisfied, we compare the true algebraic connectivity $\lambda_2^{\mathbb{L}}$ of the 1000 simulated rollouts against the lower limit ϵ in Fig. 6.4(f). Only one (colored red) of the 1000 rollouts results in $\lambda_2^{\mathbb{L}}$ dropping below ϵ. This statistically validates that the connectivity maintenance requirement from Equation (6.11) is satisfied.

For the second *offline* mission, we consider the multi-UAV system to consist of three *primary* UAVs and two *bridge* UAVs. Each *primary* UAV is tasked with reaching a desired position, whereas the *bridge* UAVs are not assigned any desired positions. Fig. 6.5 shows the results for the second *offline* mission. The convergence of our planning algorithm is shown in Figs. 6.5(a)-(d). For the initial trajectory guess, we use the LQR-based method for the *primary* UAVs, whereas for the *bridge* UAVs we set them to be hovering at their initial positions. We observe that the final planned trajectories for the *primary* UAVs reach their desired positions, while the *bridge* UAVs arrange themselves in order to maintain connectivity within the system. Similar to the first *offline* mission, we simulate 1000 trajectory rollouts where each UAV uses LQR to track the planned trajectories. Fig. 6.5(e) shows the convergence of the cost in the transformed optimization problem from Equation (6.30).

In Fig. 6.5(f) we show the connectivity maintenance of the system. The planned trajectories maintain $\underline{\lambda}_2^{\mathbb{L}}$ above the lower limit ϵ. For only two (colored magenta) of the 1000 rollouts the true algebraic connectivity $\lambda_2^{\mathbb{L}}$ drops below $\underline{\lambda}_2^{\mathbb{L}}$, whereas none drop below ϵ. This statistically validates that the connectivity maintenance requirement from Equation (6.11) is satisfied.

6.4.3 Online Planning

For *online* planning applications, we evaluate our planning algorithm on two missions under time constraints. We consider an *online* mission to be comprised of multiple segments and allot a maximum planning time for each segment. This resembles applications such as exploration, coverage, or formation control, where only a limited amount of computation time is available for planning trajectories while the remaining time is required for other purposes such as analyzing sensor data or decision making. Given that we consider multiple segments in the mission, we plan the trajectories for the

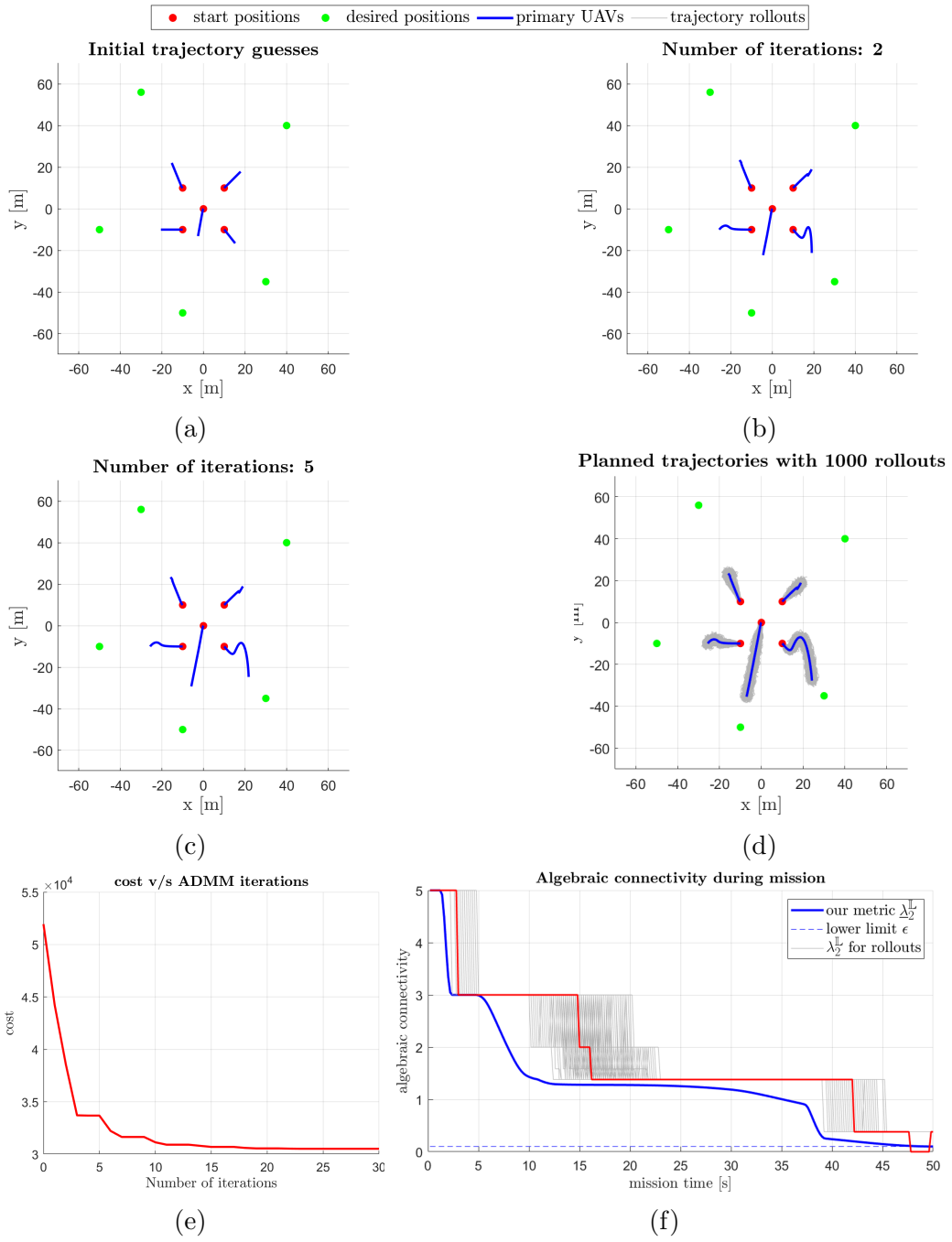


Figure 6.4: Trajectory planning and connectivity maintenance for *offline* mission with five *primary* UAVs. (a)-(d) Convergence of the planned trajectories. (e) Convergence of the cost in the transformed optimization problem (Equation (6.30)). (f) Statistical validation of connectivity maintenance throughout the mission. Only one (red) of the 1000 trajectory rollouts results in true algebraic connectivity $\lambda_2^{\mathbb{L}}$ less than the lower limit ϵ.

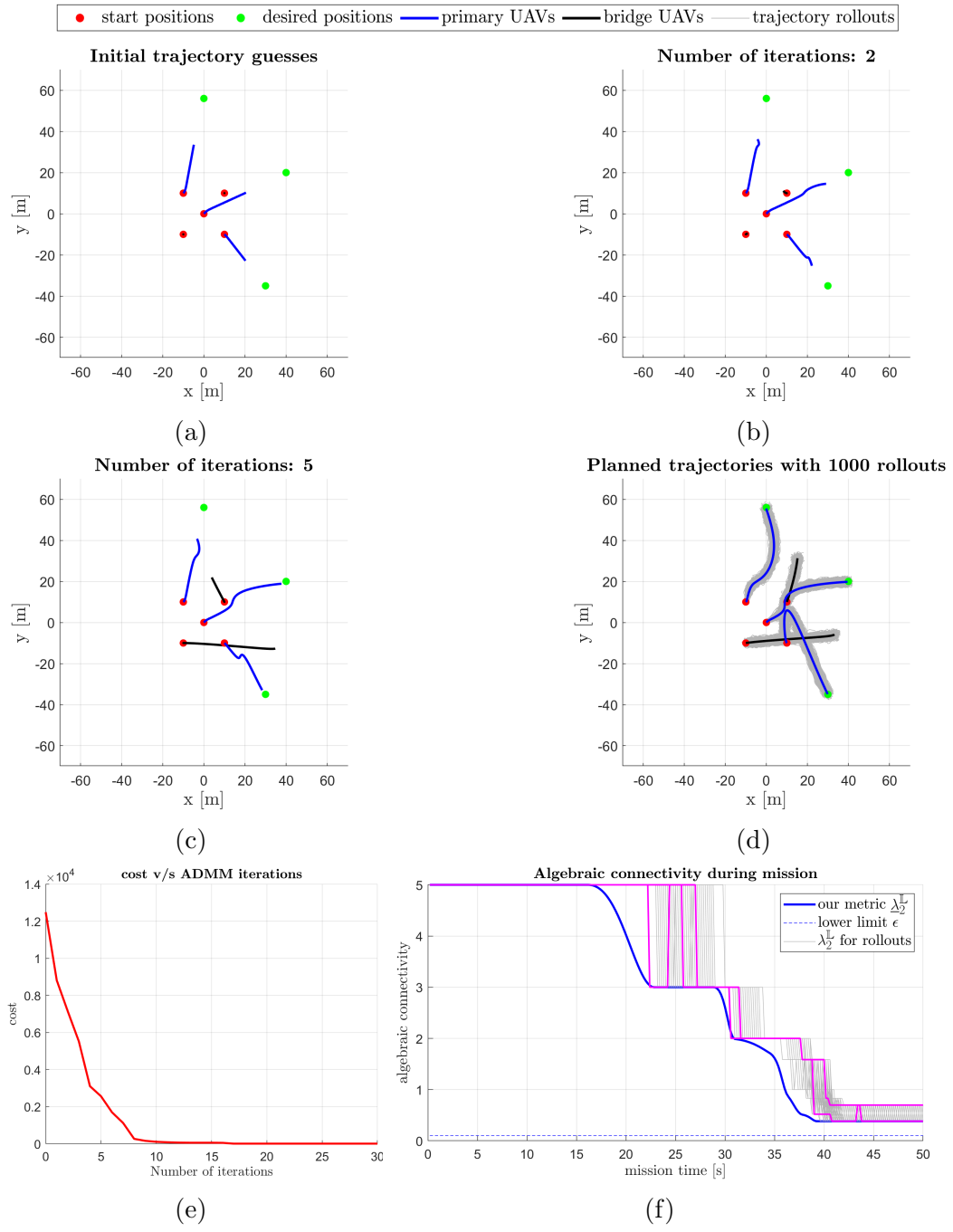


Figure 6.5: Trajectory planning and connectivity maintenance for *offline* mission with three *primary* UAVs and two *bridge* UAVs. (a)-(d) Convergence of the planned trajectories. (e) Convergence of the cost in the transformed optimization problem (Equation (6.30)). (f) Statistical validation of connectivity maintenance throughout the mission. Only two (magenta) of the 1000 trajectory rollouts results in true algebraic connectivity $\lambda_2^{\mathbb{L}}$ less than $\underline{\lambda}_2^{\mathbb{L}}$, whereas none drop below the lower limit ϵ.

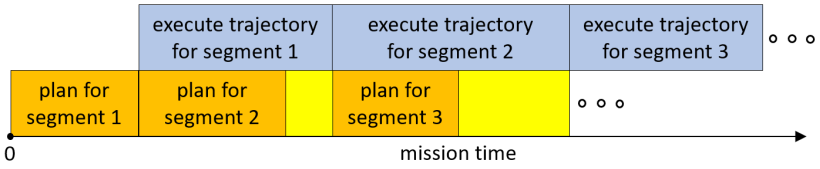


Figure 6.6: Order of trajectory planning (orange) and execution (blue) for *online* planning applications. We allot a maximum planning time for each segment since the remaining time (yellow) could be required for other purposes such as analyzing sensor data or decision making.

upcoming mission segment while executing the planned trajectories for the current mission segment. Fig. 6.6 shows the order of planning and trajectory execution for an *online* mission. We set a maximum planning time of 25 s and a trajectory duration of 50 s for each mission segment. Additionally, since our planner potentially requires multi-hop communication, we account for a delay of 0.2 s in each ADMM iteration of our planning algorithm (Algorithm 3).

For the first *online* mission, we consider a multi-UAV system with three *primary* and two *bridge* UAVs. The *primary* UAVs are tasked with reaching different desired positions in each segment. Fig. 6.7 shows the results for the first *online* mission. Here, the desired positions for the first segment are selected similar to the desired positions in the second *offline* mission (Fig. 6.5). Note that due to a limited planning time, the planned trajectories for the first segment do not converge to the desired positions for all UAVs. However, as discussed in Section 6.3.2, the output from our trajectory planning algorithm always satisfies the connectivity maintenance requirement from Equation (6.11). Fig. 6.7(a) shows the planned trajectories for the first segment along with 1000 trajectory rollouts of the system. The final UAV positions from the first segment are then used as initial positions while planning for the second segment. Figs. 6.7(b) and 6.7(c) show the planned trajectories and rollouts for the second and third segments. The convergence of the cost from Equation (6.30) for the first three segments is shown in Figs. 6.7(d)-(f). Note that the order of trajectory planning is as shown in Fig. 6.6. Fig 6.7(g) shows the connectivity maintenance results for the complete mission. $\underline{\lambda}_2^{\mathbb{L}}$ is maintained above the lower limit ϵ throughout the mission. From the 1000 simulated rollouts, we observe that the true algebraic connectivity $\lambda_2^{\mathbb{L}}$ for only two (colored magenta) rollouts drop below $\underline{\lambda}_2^{\mathbb{L}}$, while none drop below

the limit ϵ.

For the second *online* mission, in order to evaluate our planner for a larger multi-UAV system, we consider ten UAVs: six *primary* and four *bridge*. Additionally, as shown in Fig. 6.8 we consider the mission to consist of six segments in order to show the planning results for a variety of configurations of the desired UAV positions. Figs. 6.8(a)-(f) show the planned trajectories for the six segments and Figs. 6.8(g)-(l) show the convergence of the corresponding costs. In Fig. 6.8, we show the connectivity maintenance throughout the mission. Similar to the first *online* mission, $\underline{\lambda}_2^{\mathbb{L}}$ is maintained above the lower limit ϵ. The true algebraic connectivity of only one (colored magenta) of the 1000 simulated rollouts drops below $\underline{\lambda}_2^{\mathbb{L}}$, while none drop below ϵ. Thus, the connectivity maintenance requirement from Equation (6.11) is satisfied for both *online* missions.

6.5 Chapter Summary

We have designed a trajectory planning algorithm for global connectivity maintenance of multi-robot systems. The planner address two limitations in previous connectivity maintenance works: it accounts for robot motion and sensing uncertainties, and it considers robot motion models which do not necessarily have a instantaneously changeable direction of motion. To address the connectivity maintenance requirement in the planner, we first define a weighted undirected graph to represent a system with uncertain robot positions. The algebraic connectivity of this graph is then used to define a transformed trajectory planning problem which is solved by a distributed ADMM setup. We analyze the complexity of the optimization step within our ADMM setup and develop an approach to reduce its computational load by approximating the required Hessian matrices. Finally, we evaluate the planner on simulated *offline* and *online* missions of multi-UAV systems. Our algorithm plans trajectories to complete the local tasks for each UAV while maintaining connectivity within the system. We simulate a 1000 trajectory rollouts for each mission and statistically validate the connectivity maintenance.

While we have demonstrated the utility of our planner in addressing the aforementioned limitations in previous works, multiple future directions of

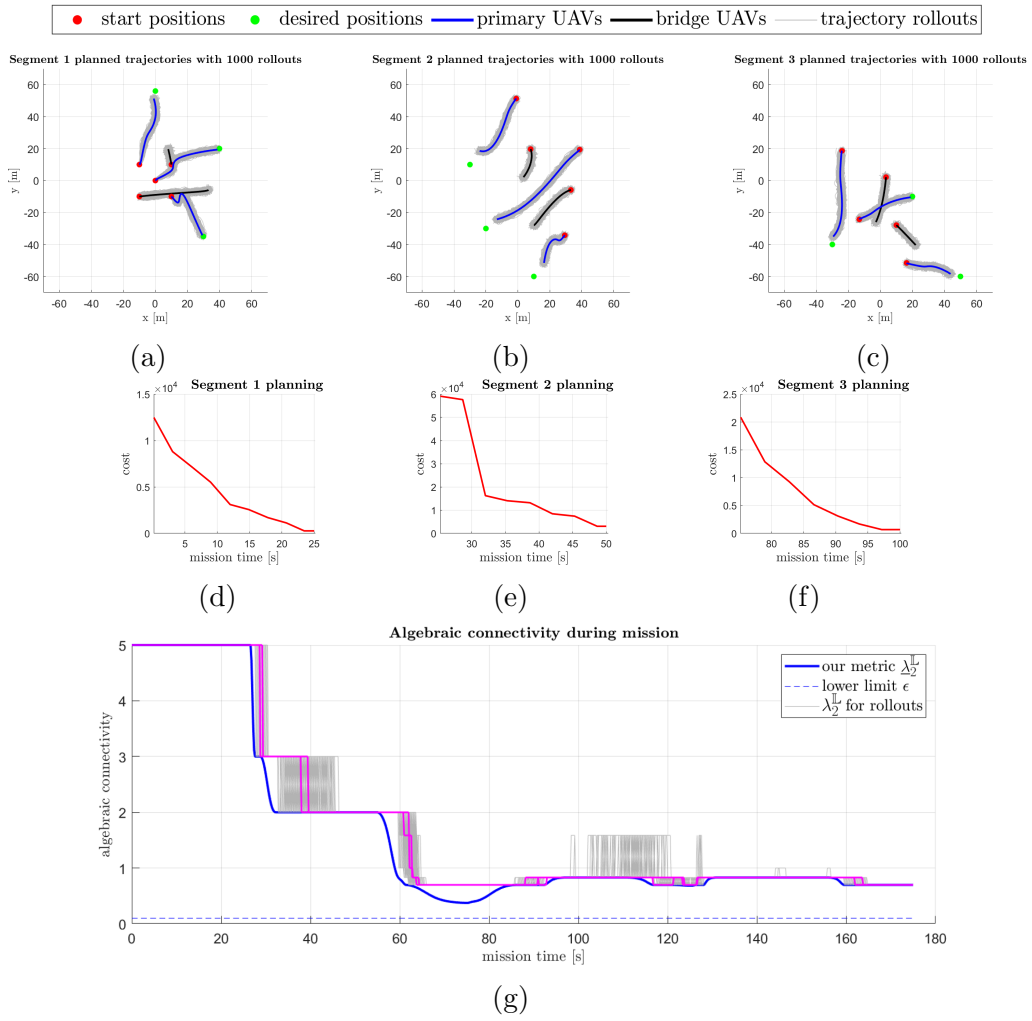


Figure 6.7: Trajectory planning and connectivity maintenance for *online* mission with three *primary* and two *bridge* UAVs. (a)-(c) The final planned trajectories and rollouts for each segment of the mission. (d)-(f) Convergence of the cost in the transformed optimization problem for a maximum planning time of 25 s. (g) Statistical validation of connectivity maintenance throughout the mission. Only two (magenta) of the 1000 trajectory rollouts results in true algebraic connectivity $\lambda_2^{\mathbb{L}}$ less than $\underline{\lambda}_2^{\mathbb{L}}$, whereas none drop below the lower limit ϵ.

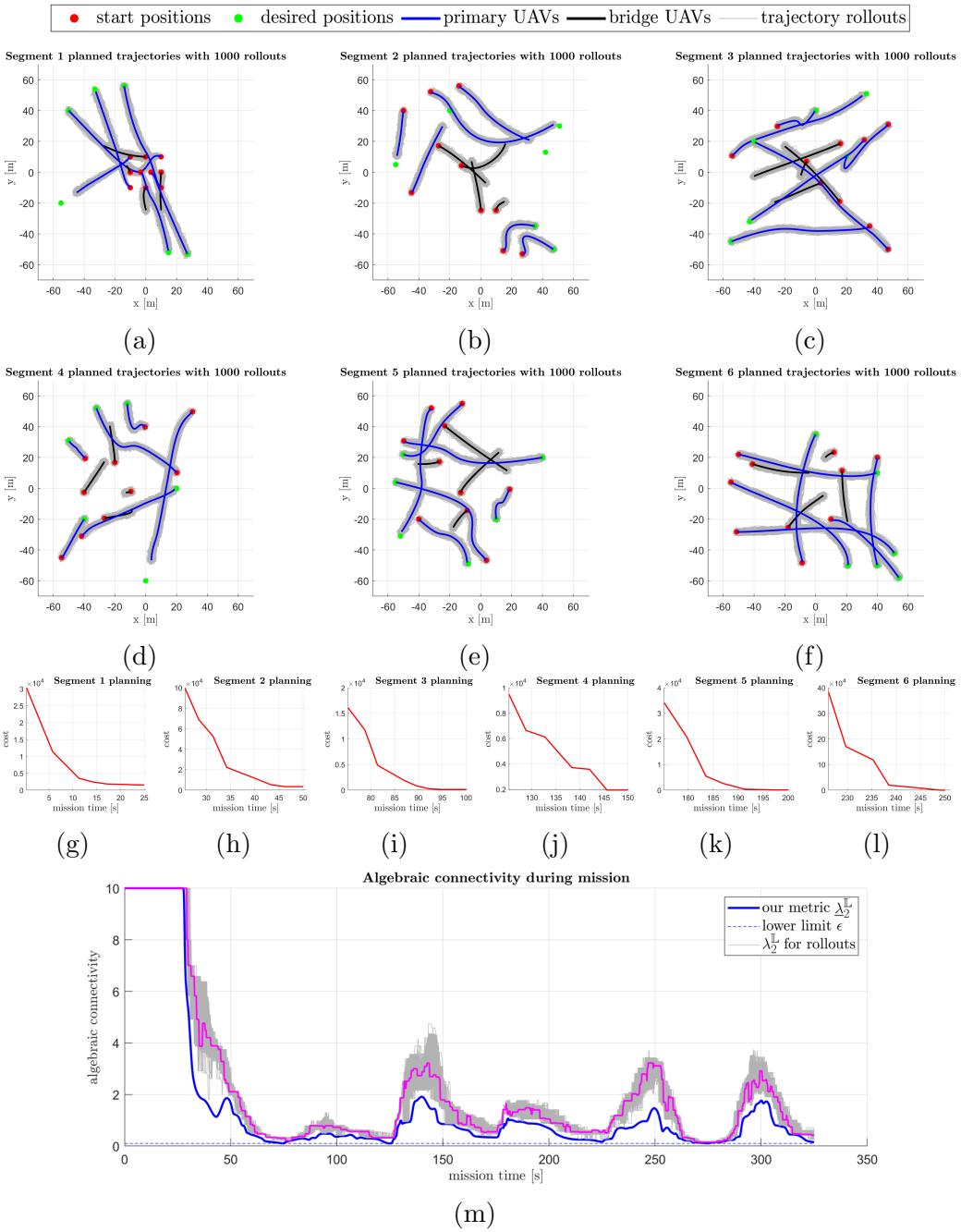


Figure 6.8: Trajectory planning and connectivity maintenance for *online* mission with six *primary* and four *bridge* UAVs. (a)-(f) The final planned trajectories and rollouts for each segment of the mission. (g)-(l) Convergence of the cost in the transformed optimization problem for a maximum planning time of 25 s. (m) Statistical validation of connectivity maintenance throughout the mission. Only one (magenta) of the 1000 trajectory rollouts results in true algebraic connectivity $\lambda_2^{\mathbb{L}}$ less than $\underline{\lambda}_2^{\mathbb{L}}$, whereas none drop below the lower limit ϵ.

work exist. First, a natural extension includes exploring decentralized architectures in order to improve the scalability of our planner with respect to communication and computational load. Second, we plan to evaluate the performance of our planner for nonlinear robot motion and sensing models where an approximate state distribution from an extended Kalman filter is used for defining the graph in Section 6.2. Third, it is desirable to include additional realistic constraints for the multi-robot system such as line-of-sight communication and collision avoidance. Finally, we also plan to test the planner on a real-world hardware multi-robot platform.

CHAPTER 7

CONCLUSIONS

This dissertation designed trajectory planning algorithms for robotic systems under motion and sensing uncertainties. We developed a reachability analysis to predict possible robot deviations along trajectories while accounting for a stochastic uncertainty and an additional bounded bias in the sensor measurements. The reachability analysis was integrated with an existing trajectory planning framework to plan collision-safe trajectories. For multi-robot systems, we designed a distributed trajectory planner that maintains system connectivity under motion and sensing uncertainties. Simulation results under various scenarios were shown to statistically validate that the trajectory planners maintained collision-safety and connectivity within the system. The contributions are summarized as follows:

- Chapter 2 presented required preliminaries from the fields of set theory and graph theory. We described the probabilistic zonotope set representation used in our reachability analysis. Set operations including the Minkowski sum, linear transform, confidence set and projection operations were defined. We discussed the topic of algebraic connectivity of a graph which is a commonly used metric to represent the global connectivity of a multi-robot system.

- Chapter 3 described our reachability analysis for robotic systems with linear motion and sensing models. The reachability analysis problem was formulated where we modeled the sensing uncertainties to contain a Gaussian stochastic component along with a bounded bias. We then used mathematical induction to derive the expression to compute reachable sets along a candidate trajectory. Here the reachable sets were computed as a function of the initial set of states, the initial state estimation error set, and the sets of motion and sensing uncertainties along the trajectory. We statistically validated that the computed

reachable sets captured the possible robot deviations for two simulated linear systems (a single-integrator and a double-integrator system) with noisy positioning measurements.

- Chapter 4 extended our reachability analysis to robotic systems with non-linear motion and sensing models. Here the reachable sets were computed as a function of the initial set of states, the initial state estimation error set, the sets of motion and sensing uncertainties along the trajectory, and the sets of linearization errors (Lagrange remainders) along the trajectory. We proposed a Gaussian approximation of these linearization errors where the projection operation from Chapter 2 was used. A 2D Dubins model with noisy ranging measurements was simulated to statistically validate the computed reachable sets.

- Chapter 5 designed a trajectory planning algorithm that used our reachability analysis from Chapters 3 and 4 to plan collision-safe trajectories. The planning framework from [51] was used due to its highly parallelizable structure which is desirable for real-world applications. We proposed a heuristic approximation to speed up the reachable set computation by discarding quantities that have a negligible contribution. A metric for the size of the reachable sets (which are represented by probabilistic zonotopes) was designed, which allowed us to compare candidate trajectories and eliminate undesirable ones. We demonstrated the applicability of the planner for fixed-wing UAVs navigating using GNSS pseudorange measurements. Simulation results were represented for planning collision-safe trajectories in a static environment and in a shared airspace with other operating UAVs. The collision-safety of the planned trajectories was statistically validated.

- Chapter 6 addressed trajectory planning for connectivity maintenance of multi-robot systems under motion and sensing uncertainties. We first defined a weighted undirected graph that accounts for uncertain robot positions arising due to the motion and sensing uncertainties. We showed that the algebraic connectivity of this graph is a probabilistic lower-bound (with a desired confidence level) for the true algebraic connectivity of the system. Next, a distributed ADMM-based trajectory planner was described that maintained the algebraic connectivity

of the proposed graph above a specified lower limit. The planning was done in a non-myopic fashion, i.e., by considering the trajectory of the robots over multiple future time instants. We analyzed the computational load of the optimization step within the ADMM framework and derived an approximation of required Hessian matrices to reduce the computational load. Various multi-UAV missions were simulated to validate the connectivity maintenance performance of the trajectory planner under motion and sensing uncertainties.

REFERENCES

[1] S. Rajendran and S. Srinivas, "Air Taxi Service for Urban Mobility: A Critical Review of Recent Developments, Future Challenges, and Opportunities," *Transportation research part E: logistics and transportation review*, vol. 143, p. 102090, 2020.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] Volocopter, "Welcome to the Urban Air Mobility Pioneer," https://www.volocopter.com/en/urban-mobility/, accessed: 2021-03-22.

[4] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle Routing Problems for Drone Delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2016.

[5] M. Jaller, C. Otero-Palencia, and A. Pahwa, "Automation, Electrification, and Shared Mobility in Urban Freight: Opportunities and Challenges," *Transportation Research Procedia*, vol. 46, pp. 13–20, 2020.

[6] G. Wing, "The Official Wing Blog," https://wing-prod.blogspot.com/, accessed: 2021-03-22.

[7] S. Scott, "Field Testing a New Delivery System with Amazon Scout," https://www.aboutamazon.com/news/transportation/meet-scout?linkCode=w50&tag=w050b-20&imprToken=ZeMIelf0qQWgVzt0u3nPig&slotNum=0, accessed: 2021-03-17.

[8] R. Ke, Z. Li, S. Kim, J. Ash, Z. Cui, and Y. Wang, "Real-time Bidirectional Traffic Flow Parameter Estimation from Aerial Videos," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 890–901, 2016.

[9] Y. Xu, G. Yu, X. Wu, Y. Wang, and Y. Ma, "An Enhanced Viola-Jones Vehicle Detection Method from Unmanned Aerial Vehicles Imagery," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1845–1856, 2016.

[10] P. Gonzalez-De-Santos, R. Fernández, D. Sepúlveda, E. Navas, and M. Armada, "Unmanned Ground Vehicles for Smart Farms," *Agronomy-Climate Change & Food Security*, p. 73, 2020.

[11] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in Autonomous UAV Cinematography: An Overview," in *2018 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2018, pp. 1–6.

[12] FAA, "FAA Aerospace Forecast Fiscal Years 2020-2040," www.faa.gov, 2020.

[13] A. S. Aweiss, B. D. Owens, J. Rios, J. R. Homola, and C. P. Mohlenbrink, "Unmanned Aircraft Systems (UAS) Traffic Management (UTM) National Campaign II," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1727.

[14] M. Schwall, T. Daniel, T. Victor, F. Favaro, and H. Hohnhold, "Waymo Public Road Safety Performance Data," *arXiv preprint arXiv:2011.00038*, 2020.

[15] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative Heterogeneous Multi-robot Systems: A Survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[16] A. Alanwar, H. Said, and M. Althoff, "Distributed Secure State Estimation Using Diffusion Kalman Filters and Reachability Analysis," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4133–4139.

[17] H. Park and S. Hutchinson, "Robust Rendezvous for Multi-robot System with Random Node Failures: An Optimization Approach," *Autonomous Robots*, vol. 42, no. 8, pp. 1807–1818, 2018.

[18] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot Tree and Graph Exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, Aug 2011.

[19] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ et al., "DARPA Subterranean Challenge: Multi-robotic Exploration of Underground Environments," in *International Conference on Modelling and Simulation for Autonomous Systesm*. Springer, 2019, pp. 274–290.

[20] "Teams complete subt challenge virtual tunnel circuit."

[21] K. Zhou and S. I. Roumeliotis, "Multirobot Active Target Tracking With Combinations of Relative Observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678–695, 2011.

[22] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, "A Distributed ADMM Approach to Non-Myopic Path Planning for Multi-Target Tracking," *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.

[23] M. Ji and M. Egerstedt, "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, Aug 2007.

[24] M. M. Zavlanos, H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Hybrid Control for Connectivity Preserving Flocking," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2869–2875, dec 2009.

[25] H. Wang and M. Rubenstein, "Shape Formation in Homogeneous Swarms Using Local Task Swapping," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 597–612, 2020.

[26] P. B. G. Dohmann and S. Hirche, "Distributed Control for Cooperative Manipulation With Event-Triggered Communication," *IEEE Transactions on Robotics*, pp. 1–15, 2020.

[27] G. Desmarais, "Smart Farming," https://www.agritechmauritius.org/smart-farming/, accessed: 2021-03-22.

[28] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series), ser. Intelligent Robotics and Autonomous Agents," 2005.

[29] N. Dadkhah and B. Mettler, "Survey of Motion Planning Literature in the Presence of Uncertainty: Considerations for UAV Guidance," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 233–246, 2012.

[30] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2000, pp. 20–31.

[31] A. Girard, "Reachability of Uncertain Linear Systems Using Zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305.

[32] A. Girard and C. Le Guernic, "Efficient Reachability Analysis for Linear Systems Using Support Functions," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 8966–8971, 2008.

[33] M. Althoff, O. Stursberg, and M. Buss, "Reachability Analysis of Linear Systems with Uncertain Parameters and Inputs," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 726–732.

[34] E. Asarin, T. Dang, and A. Girard, "Reachability Analysis of Nonlinear Systems Using Conservative Approximation," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 20–35.

[35] B. Noack, V. Klumpp, N. Petkov, and U. D. Hanebeck, "Bounding Linearization Errors with Sets of Densities in Approximate Kalman Filtering," in *2010 13th International Conference on Information Fusion*. IEEE, 2010, pp. 1–8.

[36] M. Althoff, O. Stursberg, and M. Buss, "Reachability Analysis of Nonlinear Systems with Uncertain Parameters Using Conservative Linearization," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4042–4048.

[37] M. Jones and M. M. Peet, "Using SOS and Sublevel Set Volume Minimization for Estimation of Forward Reachable Sets," *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 484–489, 2019.

[38] V. Shia, R. Vasudevan, R. Bajcsy, and R. Tedrake, "Convex Computation of the Reachable Set for Controlled Polynomial Hybrid Systems," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1499–1506.

[39] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the Gap Between Safety and Real-time Performance in Receding-horizon Trajectory Design for Mobile Robots," *arXiv preprint arXiv:1809.06746*, 2018.

[40] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, Aggressive Quadrotor Flight via Reachability-based Trajectory Design," in *Dynamic Systems and Control Conference*, vol. 59162. American Society of Mechanical Engineers, 2019, p. V003T19A010.

[41] S. Vaskov, U. Sharma, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, "Guaranteed Safe Reachability-based Trajectory Design for a High-fidelity Model of an Autonomous Passenger Vehicle," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 705–710.

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

[43] A. Majumdar and R. Tedrake, "Funnel Libraries for Real-time Robust Feedback Motion Planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[44] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A Time-dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[45] I. M. Mitchell and C. J. Tomlin, "Overapproximating Reachable Sets by Hamilton-Jacobi Projections," *journal of Scientific Computing*, vol. 19, no. 1-3, pp. 323–346, 2003.

[46] J. Van Den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[47] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion Planning Under Uncertainty Using Iterative Local Optimization in Belief Space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[48] A. Bry and N. Roy, "Rapidly-Exploring Random Belief Trees for Motion Planning Under Uncertainty," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 723–730.

[49] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion and Uncertainty Aware Path Planning for Micro Aerial Vehicles," *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[50] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware Path Planning," *arXiv preprint arXiv:1605.04151*, 2016.

[51] B. Ichter, E. Schmerling, A.-a. Agha-mohammadi, and M. Pavone, "Real-time Stochastic Kinodynamic Motion Planning via Multiobjective Search on GPUs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5019–5026.

[52] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[53] S. Karaman and E. Frazzoli, "Sampling-Based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[54] B. W. Parkinson and P. Axelrad, "Autonomous GPS Integrity Monitoring Using the Pseudorange Residual," *NAVIGATION, Journal of the Institute of Navigation*, vol. 35, no. 2, pp. 255–274, 1988.

[55] P. Misra and P. Enge, "Global Positioning System: Signals, Measurements and Performance Second Edition," *Massachusetts: Ganga-Jamuna Press*, 2006.

[56] S. Peyraud, D. Bétaille, S. Renault, M. Ortiz, F. Mougel, D. Meizel, and F. Peyret, "About Non-Line-of-Sight Satellite Detection and Exclusion in a 3D Map-Aided Localization Algorithm," *Sensors*, vol. 13, no. 1, pp. 829–847, 2013.

[57] Z. Jiang and P. D. Groves, "NLOS GPS Signal Detection Using a Dual-Polarisation Antenna," *GPS solutions*, vol. 18, no. 1, pp. 15–26, 2014.

[58] R. Yozevitch, B. B. Moshe, and A. Weissman, "A Robust GNSS LOS/NLOS Signal Classifier," *NAVIGATION, Journal of the Institute of Navigation*, vol. 63, no. 4, pp. 427–440, 2016.

[59] A. Shetty and G. X. Gao, "Covariance Estimation for GPS-lidar Sensor Fusion for UAVs," in *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, 2017, pp. 2919–2923.

[60] W. Wen, G. Zhang, and L.-T. Hsu, "Exclusion of GNSS NLOS Receptions Caused by Dynamic Objects in Heavy Traffic Urban Scenarios Using Real-time 3D Point Cloud: An Approach Without 3D Maps," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2018, pp. 158–165.

[61] G. Zhang and L.-T. Hsu, "A New Path Planning Algorithm Using a GNSS Localization Error Map for UAVs in an Urban Area," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 1, pp. 219–235, 2019.

[62] G. Duenas Arana, O. Abdul Hafez, M. Joerger, and M. Spenko, "Integrity Monitoring for Kalman Filter-based Localization," *The International Journal of Robotics Research*, vol. 39, no. 13, pp. 1503–1524, 2020.

[63] A. Martinelli, "Vision and IMU Data Fusion: Closed-form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2011.

[64] S. Bijjahalli, R. Sabatini, and A. Gardi, "GNSS Performance Modelling and Augmentation for Urban Air Mobility," *Sensors*, vol. 19, no. 19, p. 4209, 2019.

[65] S. Ragothaman, M. Maaref, and Z. M. Kassas, "Autonomous Ground Vehicle Path Planning in Urban Environments Using GNSS and Cellular Signals Reliability Maps: Models and Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[66] F. Causa and G. Fasano, "Multiple UAVs Trajectory Generation and Waypoint Assignment in Urban Environment Based on DOP Maps," *Aerospace Science and Technology*, vol. 110, p. 106507, 2021.

[67] N. Nanos, O. K. Isik, R. Verdeguer Moreno, I. Petrunin, D. Panagiotakopoulos, and A. Tsourdos, "UAV Path Planning Optimization Based on GNSS Quality and Mission Requirements," in *AIAA Scitech 2021 Forum*, 2021, p. 0710.

[68] C. Combastel, "Zonotopes and Kalman Observers: Gain Optimality Under Distinct Uncertainty Paradigms and Robust Convergence," *Automatica*, vol. 55, pp. 265–273, 2015.

[69] C. Combastel, "Merging Kalman Filtering and Zonotopic State Bounding for Robust Fault Detection Under Noisy Environment," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 289–295, 2015.

[70] C. Combastel, "An Extended Zonotopic and Gaussian Kalman Filter (EZGKF) Merging Set-membership and Stochastic Paradigms: Toward Non-linear Filtering and Fault Detection," *Annual Reviews in Control*, vol. 42, pp. 232–243, 2016.

[71] K. Khateri, M. Pourgholi, M. Montazeri, and L. Sabattini, "A Comparison Between Decentralized Local and Global Methods for Connectivity Maintenance of Multi-robot Networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 633–640, 2019.

[72] D. P. Spanos and R. M. Murray, "Robust Connectivity of Networked Vehicles," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 3. IEEE, 2004, pp. 2893–2898.

[73] Z. Lin, B. Francis, and M. Maggiore, "Necessary and Sufficient Graphical Conditions for Formation Control of Unicycles," *IEEE Transactions on Automatic Control*, vol. 50, no. 1, pp. 121–127, 2005.

[74] D. V. Dimarogonas and K. J. Kyriakopoulos, "On the Rendezvous Problem for Multiple Nonholonomic Agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 916–922, 2007.

[75] D. V. Dimarogonas and K. H. Johansson, "Decentralized Connectivity Maintenance in Mobile Networks with Bounded Inputs," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1507–1512.

[76] D. V. Dimarogonas and K. J. Kyriakopoulos, "Connectedness Preserving Distributed Swarm Aggregation for Multiple Kinematic Robots," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1213–1223, 2008.

[77] A. Ajorlou, A. Momeni, and A. G. Aghdam, "A Class of Bounded Distributed Control Strategies for Connectivity Preservation in Multi-agent Systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2828–2833, 2010.

[78] Y. Kim and M. Mesbahi, "On Maximizing the Second Smallest Eigenvalue of a State-dependent Graph Laplacian," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 99–103.

[79] M. C. De Gennaro and A. Jadbabaie, "Decentralized Control of Connectivity for Multi-agent Systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 3628–3633.

[80] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized Estimation and Control of Graph Connectivity for Mobile Sensor Networks," *Automatica*, vol. 46, no. 2, pp. 390–396, 2010.

[81] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized Connectivity Maintenance for Cooperative Control of Mobile Robotic Systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, oct 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913499085

[82] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed Control of Multirobot Systems with Global Connectivity Maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[83] L. Sabattini, N. Chopra, and C. Secchi, "On Decentralized Connectivity Maintenance for Mobile Robotic Systems," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 988–993.

[84] P. Robuffo Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A Passivity-based Decentralized Strategy for Generalized Connectivity Maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[85] A. Gasparri, L. Sabattini, and G. Ulivi, "Bounded Control Law for Global Connectivity Maintenance in Cooperative Multirobot Systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 700–717, 2017.

[86] B. Capelli and L. Sabattini, "Connectivity Maintenance: Global and Optimized Approach Through Control Barrier Functions," *arXiv preprint arXiv:2003.10178*, 2020.

[87] S. Bhattacharya and T. Başar, "Graph-theoretic Approach for Connectivity Maintenance in Mobile Networks in the Presence of a Jammer," in *Proceedings of the IEEE Conference on Decision and Control*, 2010, pp. 3560–3565.

[88] H. Fernando, A. De Silva, M. De Zoysa, K. Dilshan, and S. Munasinghe, "Modelling, Simulation and Implementation of a Quadrotor UAV," in *2013 IEEE 8th International Conference on Industrial and Information Systems*. IEEE, 2013, pp. 207–212.

[89] M. Owen, R. Beard, and T. McLain, "Implementing Dubins Airplane Paths on Fixed-wing UAVs," *Contributed chapter to the Handbook of Unmanned Aerial Vehicles*, pp. 1677–1701, 2014.

[90] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment for stochastic linear systems using enclosing hulls of probability density functions," in *2009 European Control Conference (ECC)*. IEEE, 2009, pp. 625–630.

[91] M. Althoff and B. H. Krogh, "Zonotope Bundles for the Efficient Computation of Reachable Sets," in *2011 50th IEEE conference on decision and control and European control conference*. IEEE, 2011, pp. 6814–6821.

[92] M. Althoff, "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars," Ph.D. dissertation, Technische Universität München, 2010.

[93] W. E. Hoover, "Algorithms for Confidence Circles and Ellipses," *NOAA Technical Report*, 1984.

[94] B. Wang, W. Shi, and Z. Miao, "Confidence Analysis of Standard Deviational Ellipse and its Extension into Higher Dimensional Euclidean Space," *PloS one*, vol. 10, no. 3, p. e0118537, 2015.

[95] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The Laplacian Spectrum of Graphs," *Graph theory, Combinatorics, and Applications*, vol. 2, no. 871-898, p. 12, 1991.

[96] R. Grone, R. Merris, and V. S. Sunder, "The Laplacian Spectrum of a Graph," *SIAM Journal on matrix analysis and applications*, vol. 11, no. 2, pp. 218–238, 1990.

[97] G. Welch, G. Bishop et al., "An Introduction to the Kalman Filter," 1995.

[98] M. Susi, M. Andreotti, M. Aquino, and A. Dodson, "Tuning a Kalman Filter Carrier Tracking Algorithm in the Presence of Ionospheric Scintillation," *GPS Solutions*, vol. 21, no. 3, pp. 1149–1160, 2017.

[99] P. N. Raanes, M. Bocquet, and A. Carrassi, "Adaptive Covariance Inflation in the Ensemble Kalman Filter by Gaussian Scale Mixtures," *Quarterly Journal of the Royal Meteorological Society*, vol. 145, no. 718, pp. 53–75, 2019.

[100] Y. Yang and W. Gao, "An Optimal Adaptive Kalman Filter," *Journal of Geodesy*, vol. 80, no. 4, pp. 177–183, 2006.

[101] M. Berz and G. Hoffstätter, "Computation and Application of Taylor Polynomials with Interval Remainder Bounds," *Reliable Computing*, vol. 4, no. 1, pp. 83–97, 1998.

[102] S. Tay and J. Marais, "Weighting Models for GPS Pseudorange Observations for Land Transportation in Urban Canyons," 2013.

[103] H. Hartinger and F. Brunner, "Variances of GPS Phase Observations: The SIGMA-E Model," *GPS solutions*, vol. 2, no. 4, pp. 35–43, 1999.

[104] I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. Baskaraca, H. Karim, and A. A. Rahman, "3D Modelling and Visualization Based on the Unity Game Engine - Advantages and Challenges." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 4, 2017.

[105] L.-T. Hsu, "Analysis and Modeling GPS NLOS Effect in Highly Urbanized Area," *GPS solutions*, vol. 22, no. 1, pp. 1–12, 2018.

[106] L. Zhao, J. Xu, J. Ding, A. Liu, and L. Li, "Direction-of-Arrival Estimation of Multipath Signals Using Independent Component Analysis and Compressive Sensing," *Plos one*, vol. 12, no. 7, p. e0181838, 2017.

[107] M. H. Keshvadi, A. Broumandan, and G. Lachapelle, "Analysis of GNSS Beamforming and Angle of Arrival Estimation in Multipath Environments," in *Proceedings of International Technical Meeting (ITM'11)*, vol. 1, 2011, pp. 427–435.

[108] Spirent, "SimGen Multi-GNSS Simulator Software Suite," https://www.spirent.com/assets/simgen, accessed: 2021-03-16.

[109] F. V. Diggelen, "Google to Improve Urban GPS Accuracy for Apps," https://www.gpsworld.com/google-to-improve-urban-gps-accuracy-for-apps/, accessed: 2021-03-16.

[110] A. Makhdoumi and A. Ozdaglar, "Convergence Rate of Distributed ADMM over Networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[111] S. Boyd, N. Parikh, E. Chu, J. Eckstein, S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends R in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[112] J. J. Moré and D. J. Thuente, "Line Search Algorithms with Guaranteed Sufficient Decrease," *ACM Transactions on Mathematical Software (TOMS)*, vol. 20, no. 3, pp. 286–307, 1994.

[113] B. E. Jackson, T. A. Howell, K. Shah, M. Schwager, and Z. Manchester, "Scalable Cooperative Transport of Cable-Suspended Loads With UAVs Using Distributed Trajectory Optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3368–3374, 2020.

[114] V. Y. Pan and Z. Q. Chen, "The Complexity of the Matrix Eigenproblem," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 507–516.

[115] E. Stump, A. Jadbabaie, and V. Kumar, "Connectivity Management in Mobile Robot Teams," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1525–1530.

[116] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal Sampling-based Planning for Linear-quadratic Kinodynamic Systems," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2429–2436.