Applied Safety Critical Control

Thesis by
Thomas Gurriet

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2020
Defended April 27th, 2020

To Amandine...

ABSTRACT

There is currently a clear gap between control-theoretical results and the reality of robotic implementation, in the sense that it is very difficult to transfer analytical guarantees to practical ones. This is especially problematic when trying to design safety-critical systems where failure is not an option. While there is a vast body of work on safety and reliability in control theory, very little of it is actually used in practice where safety margins are typically empiric and/or heuristic. Nevertheless, it is still widely accepted that a solution to these problems can only emerge from rigorous analysis, mathematics, and methods. In this work, we therefore seek to help bridge this gap by revisiting and expanding existing theoretical results in light of the complexity of hardware implementation.

To that end, we begin by making a clear theoretical distinction between systems and models, and outline how the two need to be related for guarantees to transfer from the latter to the former. We then formalize various imperfections of reality that need to be accounted for at a model level to provide theoretical results with better applicability. We then discuss the reality of digital controller implementation and present the mathematical constraints that theoretical control laws must satisfy for them to be implementable on real hardware. In light of these discussions, we derive new realizable set-invariance conditions that, if properly enforced, can guarantee safety with an arbitrary high levels of confidence. We then discuss how these conditions can be rigorously enforced in a systematic and minimally invasive way through convex optimization-based Safety Filters. Multiple safety filter formulations are proposed with varying levels of complexity and applicability. To enable the use of these safety filters, a new algorithm is presented to compute appropriate control invariant sets and guarantee feasibility of the optimization problem defining these filters. The effectiveness of this approach is demonstrated in simulation on a nonlinear inverted pendulum and experimentally on a simple vehicle. The aptitude of the framework to handle a system's dynamics uncertainty is illustrated by varying the mass of the vehicle and showcasing when safety is conserved. Then, the aptitude of this approach to provide guarantees that account for controller implementation's constraints is illustrated by varying the frequency of the control loop and again showcasing when safety is conserved.

In the second part of this work, we revisit the safety filtering approach in a way that addresses the scalability issues of the first part of this work. There are two main approaches to safety-critical control. The first one relies on computation of control invariant sets and was presented in the first part of this work. The second approach draws from the topic of optimal control and relies on the ability to realize Model-Predictive-Controllers online to guarantee the safety of a system. In that online approach, safety is ensured at a planning stage by solving the control problem subject for some explicitly defined constraints on the state and control input. Both approaches have distinct advantages but also major drawbacks that hinder their practical effectiveness, namely scalability for the first one and computational complexity for the second one. We therefore present an approach that draws from the advantages of both approaches to deliver efficient and scalable methods of ensuring safety for nonlinear dynamical systems. In particular, we show that identifying a backup control law that stabilizes the system is in fact sufficient to exploit some of the set-invariance conditions presented in the first part of this work. Indeed, one only needs to be able to numerically integrate the closed-loop dynamics of the system over a finite horizon under this backup law to compute all the information necessary for evaluating the regulation map and enforcing safety. The effect of relaxing the stabilization requirements of the backup law is also studied, and weaker but more practical safety guarantees are brought forward. We then explore the relationship between the optimality of the backup law and how conservative the resulting safety filter is. Finally, methods of selecting a safe input with varying levels of trade-off between conservativeness and computational complexity are proposed and illustrated on multiple robotic systems, namely: a two-wheeled inverted pendulum (Segway), an industrial manipulator, a quadrotor, and a lower body exoskeleton.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Juan-Pablo Afman, John Franklin, Mark L. Mote, Thomas Gurriet, and Eric Feron. On the design and optimization of an autonomous microgravity enabling aerial robot. *arXiv preprint arXiv:1611.07650*, 2016. T.G. participated in the conception of the project, developed the sizing tool, and participated in the writing of the manuscript.

[2] Juan-Pablo Afman, Laurent Ciarletta, Eric Feron, John Franklin, Thomas Gurriet, and Eric N. Johnson. Towards a new paradigm of uav safety. *arXiv preprint arXiv:1803.09026*, 2018. T.G. participated in the conception of the project, designed and built the Optimal Active Breaking Braking System, and participated in the writing of the manuscript.

[3] Thomas Gurriet and Laurent Ciarletta. Towards a generic and modular geofencing strategy for civilian uavs. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 540–549. IEEE, 2016. doi: 10.1109/ICUAS.2016.7502603. T.G. developed the mathematical methods and algorithms use in this work and participated to the writing of the manuscript.

[4] Thomas Gurriet, Mark L. Mote, Aaron D. Ames, and Eric Feron. Establishing trust in remotely reprogrammable systems. In *Proceedings of the International Conference on Human-Computer Interaction in Aerospace*, pages 1–4, 2016. doi: 10.1145/2950112.2964573. T.G. participated in the writing of the manuscript.

[5] Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, Ayonga Hereid, Omar Harib, Matthieu Masselin, Jessy Grizzle, and Aaron D. Ames. Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2804–2811. IEEE, 2018. doi: 10.1109/ICRA.2018.8460647. T.G. participated in the experiments as well as the writing of the manuscript.

[6] Thomas Gurriet, Mark L. Mote, Aaron D. Ames, and Eric Feron. An online approach to active set invariance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3592–3599. IEEE, 2018. doi: 10.1109/CDC.2018.8619139. T.G. developed the mathematical methods and algorithms use in this work and participated in the writing of the manuscript.

[7] Thomas Gurriet, Andrew Singletary, Jacob Reher, Laurent Ciarletta, Eric Feron, and Aaron D. Ames. Towards a framework for realizable safety critical control through active set invariance. In *2018 ACM/IEEE 9th*

International Conference on Cyber-Physical Systems (ICCPS), pages 98–106. IEEE, 2018. doi: 10.1109/ICCPS.2018.00018. T.G. developed the mathematical methods and algorithms use in this work and participated in the experiments as well as the writing of the manuscript.

[8] Thomas Gurriet, Mark L. Mote, Andrew Singletary, Eric Feron, and Aaron D. Ames. A scalable controlled set invariance framework with practical safety guarantees. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 2046–2053, Dec 2019. doi: 10.1109/CDC40024.2019.9030159. T.G. developed the mathematical methods and algorithms use in this work and participated in the experiments as well as the writing of the manuscript.

[9] Thomas Gurriet, Petter Nilsson, Andrew Singletary, and Aaron D. Ames. Realizable set invariance conditions for cyber-physical systems. In *2019 American Control Conference (ACC)*, pages 3642–3649. IEEE, 2019. doi: 10.23919/ACC.2019.8815332. T.G. developed the mathematical methods and algorithms use in this work and participated in the writing of the manuscript.

[10] Thomas Gurriet, Maegan Tucker, Alexis Duburcq, Guilhem Boeris, and Aaron D. Ames. Towards variable assistance for lower body exoskeletons. *IEEE Robotics and Automation Letters*, 5(1):266–273, 2019. doi: 10.1109/LRA.2019.2955946. T.G. developed the mathematical methods and algorithms use in this work and participated in the experiments as well as the writing of the manuscript.

[11] Thomas Gurriet, Maegan Tucker, Claudia Kann, Guilhem Boeris, and Aaron D. Ames. Stabilization of exoskeletons through active ankle compensation. *arXiv preprint arXiv:1909.11848*, 2019. T.G. developed the mathematical methods and algorithms use in this work and participated in the experiments as well as the writing of the manuscript.

[12] Omar Harib, Ayonga Hereid, Ayush Agrawal, Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, M. Eva Mungai, Mattieu Masselin, Aaron D. Ames, et al. Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking. *IEEE Control Systems Magazine*, 38(6):61–87, 2018. doi: 10.1109/MCS.2018.2866604. T.G. participated in the conception of the project, participated in the experiments, and participated in the writing of the manuscript.

[13] Andrew Singletary, Petter Nilsson, Thomas Gurriet, and Aaron D. Ames. Online active safety for robotic manipulators. 2019. doi: 10.1109/IROS40897.2019.8968231. T.G. participated in the conception of the project, coded the Safety Filtering algorithm, and participated in the writing of the manuscript.

[14] Andrew Singletary, Thomas Gurriet, Petter Nilsson, and Aaron D. Ames. Enabling rapid aerial exploration of unknown environments. In *To appear in the proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020. T.G. participated in the conception of the project, setup the simulation environment, coded the Performance Filter, Safety Filter, and Velocity Controller, and participated in the writing of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

FOUNDATIONS

1.1 Introduction

Safety is arguably one of the most critical issues hindering the democratization of autonomous systems. Even though safety is fairly well understood at a theoretical level, solutions that are both rigorous and practical have yet to be realized [17, 52, 70]. In this work, we will explore approaches to help us design and build safer systems.

At the core of System Design is the concept of requirements and specifications. There are many approaches to requirements [54], but they all converge on the idea of a hierarchy between requirements and specifications. A design always begins with "high-level" requirements expressing the purpose of the system, and how it shall behave. Finding a physical realization of the system that satisfies requirement is what the bulk of the design work is about. The precise nature of this realization is captured into specifications about the different physical components constituting the system. Controller Design constitutes one level in the requirement/specification pyramid. The idea of controlling a system arises when the system's behaviors are dynamic. This is, for example, the case with autonomous cars, as their behaviors are highly dynamic and it is necessary to have a way to regulate these behaviors to avoid potential disasters. Closed-loop control—simply referred to as control from now on—leverages interactions between parts of the system and its environment to achieve desired behaviors. Figuring out what this coupling needs to be in order to achieve the desired behaviors is what Control Theory tries to achieve by relying on mathematical abstractions of reality and rigorous deduction to translate high level behavior requirements into precise specification on the coupling between parts of the system. Ways to physically realize this coupling can vary widely and mechanical solutions have progressively made way for electro-mechanical and digital ones. With the advent of general purpose computing, the possibilities for what this coupling can be are becoming almost endless. One must however be careful and realize that the mathematical descriptions of this coupling—namely **control laws**—bare weight on the entire physical realization of the system. As we will see later, there is a tight coupling between the knowledge of the system's dynamics, the control law implementation (the **controller**), and the control law itself.

It is possible to classify requirements about the behavior of a system into two categories: performance and safety. Performance requirements address what a system shall do, and safety what it shall **not** do. The naming of these two categories can be controversial as safety is usually associated with the idea of not causing harm to living beings. But for simplicity's sake, we will use this term here in a more generic way and describe any undesirable behavior as unsafe. To design safe systems, the most common approach is to choose performance requirements that are intrinsically safe—meet the safety requirements. Then all one has to focus on is make sure the performance requirements are met. This approach is satisfactory when the desired behavior of the system is fixed and known at the design state. This is for example the case of most industrial manipulators that only have to repeatedly follow prescribed trajectories that have been designed ahead of time by an expert so as to avoid any collision with its surroundings. But what about autonomous cars for which the trajectory and the environment is not known at design time? There lies the issue of this approach: if the desired behavior of the robot changes, these trajectories have to be re-designed and re-validated to be safe, which is difficult and time consuming. This approach is therefore not viable when trying to design dynamic systems where desired behaviors and environments change constantly, hence the current confinement of most autonomous robots to repetitive tasks in factories. To be able to design and operate robotic systems in truly dynamic contexts, we have no choice but to decouple the enforcement of safety from the quest for performance. Instead of finding a special case of safe behaviors satisfying the performance requirement, we need to characterize the essence of all safe behaviors for a given system and find a way to enforce safety on the fly, in a way that maximizes the performances of the system—follow the desired trajectory as closely as possible, for example. This approach to safety of controlled systems is known as **Run-time Assurance**.

At a mathematical level, this approach is associated with the concept of set invariance [14]—given a desired safety set, ensuring safe system operation is equivalent to making sure that this system remains inside a safety set at all times. This is usually done in two stages. First, the parts of the safety set from where future safety violations are unavoidable are identified, which is done by computing a control invariant subset of the safety set [8, 13, 14]. Many approaches have been proposed for finding such a subset: discretized solutions of Hamilton-Jacobi equations [64], SOS optimization [88], sampling [28], and many others [62]. Secondly, a control law that renders the control invariant subset forward invariant is designed. In this second step, two different approaches are usually considered. In [10, 58, 74], a control structure is proposed that switches between a nominal controller designed for performance and a "safe" controller designed for set-invariance. Whereas in [4, 5, 85], continuous optimization-based filtering of the control input is performed so as to enforce set invariance in a minimally invasive way. Regardless of the chosen approach, all strategies realise, at a fundamental level, the enforcement of Nagumo's subtangentiality condition [8] with varying degrees of conservatism.

What we set to achieve is fundamentally deductive: derive a specification that, if satisfied, guarantees the satisfaction of the safety requirements. However, we know that one can only reason inductively about reality—predictions can only be extrapolation of observations. The hard truth is that we cannot prove anything about reality. We can only observe reality over a small region of space-time and extrapolate what it most likely will be in a region of space-time we have not explored yet. This is the idea of creating a **model** of reality. Such models are therefore fundamentally uncertain, and the confidence in a theoretical prediction made from a model is fundamentally correlated to the confidence we have in the validity of the model itself. Despite solid theoretical underpinnings, the practical effectiveness of the body of work on controlled invariance is hindered by the use of idealistic models and assumptions. Inclusion of more realistic representations in the context of set invariance is a hard problem and has not yet been fully addressed. Additive uncertainty has been considered [68, 72], as well as generic parametric uncertainty [38], including for hybrid models [20], whereas [21, 65] deals with sampled data models. But even with these *robust* approaches, systematic and unfailing run-time assurance has yet to be demonstrated on real hardware, although there have been promising cases of experimental success through reasonable usage of the aforementioned methods [38, 84].

To achieve the goal of translating theoretical guarantees to hardware, we argue that these aforementioned approaches must be revisited in light of the complexity of reality and the constraints it imposes on control laws for them to be realizable. To that end, we begin by making a clear conceptual distinction between a system and a model, and outline how the two need to be related for guarantees to transfer from the latter to the former.

1.2 Modeling Cyber-Physical Systems

While physical systems, control systems, and models of systems are treated interchangeably in much of the existing literature, at the core of this work is the clear distinction between these fundamentally different concepts.

1.2.1 Control Systems as Abstractions of Physical Reality

Before talking about models of systems, we need to understand what we mean by system. It all begins with the Physical Reality. Its essence is inaccessible to us—at least for now, and probably forever—but we can nonetheless interact with it, and from these experiences, we can create representations of the Physical Reality that make sense to us: Homo Sapiens. We will talk about the **physical system** to refer to the part of the physical reality we are focusing on.

Science strives to provide a coherent representation and description of this physical reality. At the core of this enterprise is the notion of measurement, which is our only rigorous access point to Reality. A measurement is the process of mapping elements of Reality to mathematical objects. Depending on the physical system, some mathematical representations are more relevant than others. Most of the physical systems we care about in engineering are of macroscopic scale, where reality is mostly Euclidean and deterministic, and can therefore be represented fairly well with real numbers and their evolution with time. But at the microscopic level, probability distributions are better suited to represent phenomena as the field of quantum mechanics illustrates in the case of elementary particles. When a measurement is sufficiently representative of reality we talk about a **physical quantity**. Let us now define the notion of a Control System.

Terminology 1. A **Classical Control System** $\Sigma = (\bar{x}, \bar{u})$ is a collection of values $\bar{x}$ and $\bar{u}$ resulting from the perfect and continuous measurements of some physical quantities.

Terminology 2. We will call **state of the system** the function $\bar{x} : \mathbb{R} \to \mathbb{R}^n$

and **input of the system** the function $\bar{u} : \mathbb{R} \to \mathbb{R}^n$.

The word classical is used here to imply that the state and input are finite dimensional vectors of real numbers and that they can be defined uniquely with respect to time. Indeed, when measuring a physical quantity, the result of this measurement is always tainted with noise and uncertainty. But if the measurement is representative of the physical reality, successive measurements of the same reality tend to aggregate around a unique value. It is this value we refer to as the result of a **perfect measurement**. The notion of time is also fundamental in this definition. The word classical is therefore also used to imply the continuity and monotonicity of time in the physical system we are trying to represent, hence the notion of **continuous measurement** as the limit of a series of individual measurements when the time interval between them becomes infinitesimally small.

The split between states and inputs can seem somewhat arbitrary as they both are of the same nature: physical quantities. The choice of this split is driven by the physical reality of the system we want to control. Controlling a system is choosing a coupling between the states and input in such a way that the system will exhibit the desired behavior. However, some couplings are easier to realize than others, hence the importance of the choice of states and inputs. Some physical quantities are also more fundamental than others, and most of the physical quantities at a macro-level tend to be aggregate of micro-level ones. The quality of a system's physical quantities to be representative of reality will be referred to as the **representativity of the system**. Understanding the representativity of a system is the first key in translating mathematical results into practical ones.

1.2.2 Models of Control Systems

Once a control system has been defined, one is then naturally interested in the **dynamics** of that system, i.e. how it evolves with time and how these variations are coupled with the values of the state and input. We will therefore define a **model of a system** as a mathematical representation of the evolution of the state of that system.

Their exist many types of models, depending on the system we are trying to describe. In this work, we focus on engineering systems whose behaviors unfold on fairly uniform timescales. For such systems, the dynamics are usually described by an Ordinary Differential Equation (ODE). However, such ideal representation can only **approximate** the true system's behavior and are therefore not suited to effectively translate mathematical guarantees into practical ones. We will therefore focus on models that can **capture** the system's behavior.

Terminology 3. A **Classical Model** $\mathcal{M} = (\mathcal{X},\mathcal{U},F)$ of a control system Σ consists of sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ and a set-valued function (also called map) $F : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ describing the effect of the state and input over the variation of the state with respect to time. We refer the reader to [9] for more details on set-valued analysis.

Terminology 4. The map F will be called the **dynamics** of the control system.

Terminology 5. The set $\mathcal{X} \times \mathcal{U}$ will be called the **domain of definition** of model $\mathcal{M}$. It represents the values of the states and inputs for which the dynamics is well defined from a mathematical standpoint.

We will use the term of **solutions** of the model to refer to the possible trajectories described by the model.

Terminology 6. A **practical solution** of a model $\mathcal{M}$ is a trajectory pair $(x,u) : \mathbb{R} \to \mathcal{X} \times \mathcal{U}$, with x piecewise differentiable and u piecewise locally Lipschitz continuous, that satisfies the differential inclusion $\frac{dx}{dt} \in F(x,u)$ for almost all $t \in \mathbb{R}$.

We will denote by $\mathcal{S}(\mathcal{M})$ the set of practical solutions of model $\mathcal{M}$.

Terminology 7. Functions x and u that are solutions of the model will be respectively called a **state and input of the model**.

Remark 1. If their is no ambiguity on whether we are talking about model's state/input or system's state/input, we will abbreviate and talk about state/input.

Solutions of a model are abstract mathematical objects and have no intrinsic value to describe the system. We will therefore use the notion of **validity** to describe the quality of a model in describing the behavior of a control system.

Terminology 8. A model $\mathcal{M}$ is a **valid** representation of control system Σ over $X \times U$ if for all system's state $\bar{x} \in X$ and inputs $\bar{u} \in U$ the following holds:

$$\frac{d\bar{x}}{dt} \in F(\bar{x}, \bar{u}). \tag{1.1}$$

Remark 2. The set $X \times U$ will be called the **domain of validity** of model $\mathcal{M}$.

The validity of a model is a property that can only be verified experimentally through measurements. But like any experimental result, it is uncertain, and the validity can only be verified probabilistically. One can therefore never be certain about the validity of a model and can only have a finite level of confidence in this validity.

Note also that the ability to determine the validity of a model is fundamentally coupled with the quality of the chosen states to represent the complexity of the reality of the system. We will use the term of **relevance** to describe the quality of a system to represent a given reality in the sense that the choice of states and inputs is sufficiently rich to allow for precise models to be derived with high levels of validity.

To summarize, a control system is a mathematical representation of a physical system, and a model is a mathematical description of the behavior of a control system. A classical control system has a **unique** state and input given by the perfect and continuous measurements of the physical quantities of interest, whereas a model can have many possible states and inputs. A model is valid if the system's state and input are a solution of the model. In the rest of this work, we will use the notation $\bar{x}$ to refer to the system's state, $\tilde{x}$ to refer to an estimate of the system's state, and x to refer to a state of a model of the system (and similarly for u).

1.2.3 Uncertainty in Models

1.2.3.1 Precision of models

Since we can not hope to obtain an exact mathematical description of the behavior of control systems, we resort to the idea of finding models that *over-approximate* it. Constructing an over-approximating model for a real system is of course a very challenging task that involves a trade-off between the confidence in the the validity of the model and the **precision** of the model.

Definition 1. A model $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ is less **precise** than (or equivalently over-approximates) a model $\mathcal{M}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, written $\mathcal{M}_1 \succeq \mathcal{M}_2$, if there exist (projection) maps $\Gamma^{\mathcal{X}}$ and $\Gamma^{\mathcal{U}}$ such that $\mathcal{X}_1 \supseteq \Gamma^{\mathcal{X}}(\mathcal{X}_2)$, $\mathcal{U}_1 \supseteq \Gamma^{\mathcal{U}}(\mathcal{U}_2)$ and for all $x \in \mathcal{X}_2$ and $u \in \mathcal{U}_2$,

$$F_1\left(\Gamma^{\mathcal{X}}(x), \Gamma^{\mathcal{U}}(u)\right) \supseteq T_x\Gamma^{\mathcal{X}}\left(F_2\left(x, u\right)\right), \tag{1.2}$$

where $T_x\Gamma^X$ is the tangent map of Γ^X at x.

Combined, these conditions imply that the dynamics of $\mathcal{M}_2$ via the projection maps can be embedded in the space of $\mathcal{M}_1$ in a way so that all behaviors of $\mathcal{M}_2$ are captured also by $\mathcal{M}_1$.

Terminology 9. A model will be called **exact** when its dynamics F are single valued, and **inexact** when they are not.

Terminology 10. Models that are either exact or inexact but valid will be called respectively **perfect** and **imperfect** models.

1.2.3.2 White-box modeling

Often in engineering, models can be derived from domain specific laws. These laws are usually exact models that have been validated up to high enough levels of confidence. However, models derived this way require parameters that represent physical properties of the system in question. But even if assuming that there exists such an exact description of the behavior of the system, finding the exact values of these parameters is fundamentally impossible. So the first step in building a model that has a chance of being valid is to acknowledge the uncertainty about these model parameters and consider their probable values.

In particular, let us consider an exact dynamical model of the form $\frac{dx}{dt} = f\left(x, u, p\right)$, with p the parameters of the model. As it is hard if not impossible to find values for the parameters that make this model valid, we can consider a model of the form $\frac{dx}{dt} \in F\left(x, u\right)$ where:

$$\forall x \in X, \forall u \in U,\ F\left(x, u\right) = \left\{f\left(x, u, p\right) \mid p \in P\right\}. \tag{1.3}$$

It is important to understand that the size of P can be driven by both uncertainty and design choices. In the context of a commercial aircraft, for example, the parameters could be the mass of the aircraft, but also the drag coefficient of the aircraft or the force and direction of the wind. The first one is variable as the aircraft burns fuel, the second is fixed but uncertain, and the third one is variable and uncertain. In the first case, the choice of P would come from an educated knowledge about the system and its operation. In the second case, P could come from testing in wind tunnels and associated measurement uncertainty. And in the third case, the choice of P would come from a decision on the allowed operational conditions for this aircraft. The larger P is, the more likely the model is to be valid, but the more conservative we will have to be in the control strategy as we will discuss in more detail.

When the model dynamics is described in such a way, we will denote the resulting model by $\mathcal{M} = (X, U, P, f)$. Note that we use X instead of $\mathcal{X}$ to indicate that we consider the model over its domain of validity only.

1.2.4 Digital Controller Implementation

Control system models describe the effect of the state and input on the behavior of the system. Our goal being to regulate this behavior, we are interested in finding how we can couple the input to the state in a way that achieve the desired behaviors. A mathematical relation between the model input and state will be called a **control law**. When the desired behavior is not fixed at design time, it needs to be considered as an external input of the control law itself. We will call it the **command**, and denote it by $cmd(t)$. The command cmd is a mathematical representation of a desired behavior, and in most cases will be a vector of real numbers representing a desired state to be reached. The control law then needs to be designed such that the behavior of the system corresponds to the one represented by the command. We usually work with static control laws of the form $u(t) = k(x(t), cmd(t))$, but ones depending on the history of the system can also be considered.

Designing a control law consists in characterizing what the coupling between state and input need to be to achieve the desired behavior. To make use of a control law on the physical system, this control law has to be **implemented**. The result of this implementation is a **controller**, i.e. a physical realization of the control law. In this sense, control laws are just models of controllers, even though we do not usually think of them in this way. Nevertheless, it is fundamental, like with system models, to understand the relation between control laws and their physical counterparts. From a control design perspective, this is captured by the idea that not all control laws are **realizable**, and it is important to be aware of the technological reality of controller implementation and how it translates to realizability constraints for the control laws.

1.2.4.1 Sensing and Actuation Uncertainty

The first constraint on control law realizability is the fundamental one about uncertainty. If state and input are defined by perfect measurements, getting access to the values of these perfect measurements is impractical, if not impossible. The best we can hope for is to be able to capture the values of the states and inputs. For the purpose of providing results applying to the control system, we will consider the following implementation assumptions.

Assumption 1. If the state of the system is $\bar{x}(t) \in S$, we have access to an estimate $\tilde{x}(t)$ of that state such that:

$$\bar{x}(t) \in \tilde{x}(t) \oplus \mathit{\Delta}_x, \quad \mathit{\Delta}_x \subseteq \mathbb{R}^n. \tag{1.4}$$

Remark 3. Here $\oplus$ will be used to denote the Minkowski addition of two sets:

$$A \oplus B = \{a + b \mid \forall a \in A,\ \forall b \in B\}, \tag{1.5}$$

and $\ominus$ will denote the Minkowski difference of two sets:

$$A \ominus B = \{a - b \mid \forall a \in A,\ \forall b \in B\}. \tag{1.6}$$

If the set on the left is a singleton, we will write $\{a\} \oplus B$ as $a \oplus B$ for simplicity.

This assumption relates directly to the accuracy of the sensors, as well as to the accuracy of the state estimation method. And similarly to model validity, these assumptions can only be met probabilistically. For linear systems, it is known how this type of robust state-valued observers can be synthesized [76], but optimal observers can be arbitrarily complex whereby more practical alternatives have been proposed [15]. In this setting, the resulting probability distribution can be converted into a chance constraint set $\mathit{\Delta}_x$ with $\mathbb{P}\left[\tilde{x}(t) - \bar{x}(t) \in \mathit{\Delta}_x\right] \geq 1 - \delta$, and $\mathit{\Delta}_x$ can be taken as the state estimate. This results in guarantees that hold with high probability.

The next assumption reflects noise and unmodeled dynamics between the controller output $\tilde{u}(t)$ and the actual value of the system input $\bar{u}(t)$.

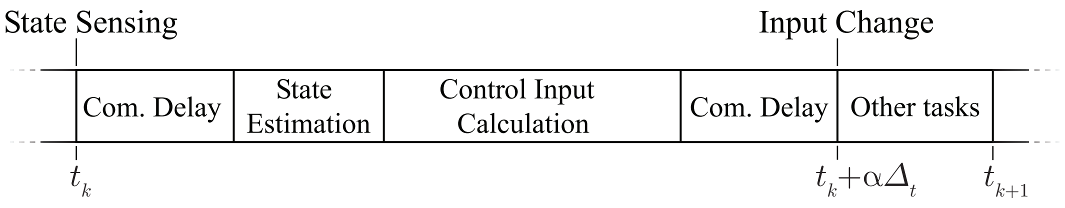


Figure 1.1: Typical digital control loop timing structure.

Assumption 2. For a desired input $\tilde{u}(t)$ to the system, the actual system's input $\bar{u}(t)$ is such that:

$$\bar{u}(t) \in \tilde{u}(t) \oplus \Delta_u, \quad \Delta_u \subseteq \mathbb{R}^m. \tag{1.7}$$

There are many reasons as to why a desired input might not be perfectly achievable on the physical system. For instance, in many cases the computed input is passed to an Electronic Control Unit (ECU) that acts as a closed-loop controller around the property of interest (e.g. servo angle, torque, or speed). In that case, transients in the ECU control loop might cause the input to deviate from the desired value. In low-powered embedded systems there may also be quantization effects that result in small but sometime significant discrepancies between the desired and actual inputs.

1.2.4.2 Time Quantization and Delays in Digital Computing

If one were to try to implement a control law of the form $\tilde{u}(t) = k\left(\tilde{x}(t)\right)$, he would need to have access to continuous measurements of the state and have the capability to continuously modify the inputs. This is certainly possible through means of analog electronics, however, the flexibility and performances provided by digital electronics overwhelmingly outweigh the continuity benefits of such an analog controller. With digital controllers, estimates of the states are available only at discrete time instances, and the input is modified at discrete points in time. Furthermore, the time between receiving a state estimate and applying a computed control input is non-negligible. Therefore, we need to account for this constraint when designing control laws that have to be digitally implemented. For that, we consider the typical timing structure of digital control systems depicted in Fig. 1.1 and assume the following.

Assumption 3. The digital controller implementation operates at a fixed loop frequency $1/\Delta_t$. Each control loop starts at time $t_k,\ k \in \mathbb{N}$ with an estimate

$\tilde{x}_k = \tilde{x}(t_k)$ of the state as in Assumption 1. This estimate is then used to compute a subsequent control action $\tilde{u}_k$ that is realized at time $t_k + \alpha\Delta_t$ for $\alpha \in [0,1]$.

With this approach, the continuous time assumption is replaced by a more realistic validity assumption on the timing of the control loop. Rather than incorporating these discrete components into the model of the system, we choose to address them at the control law level and provide results with respect to continuous-time control systems (cf. Sampled Data Systems). We refer the reader to the real-time computing literature for more details on the complex issue of guaranteeing a bounded computing time [24].

1.3 Outline

This work is divided into two main chapters. In the first part, we derive new realizable set-invariance conditions that, if properly enforced, can guarantee safety with arbitrarily high levels of confidence. We then discuss how these conditions can be rigorously enforced in a systematic and minimally invasive way through convex optimization based Safety Filters. Multiple safety filter formulations are proposed with varying levels of complexity and applicability. To enable the use of these safety filters, a new algorithm is presented to compute appropriate control invariant sets and guarantee feasibility of the optimization problem defining these filters. Finally, the proposed framework effectiveness is demonstrated in simulation on an inverted pendulum and experimentally on a Segway vehicle. The aptitude of the framework to handle system's dynamics uncertainty is illustrated by varying the inertial proprieties of the vehicle and verifying that safety is maintained. Then, the aptitude of the framework to provide guarantees that account for controller implementation's constraints is illustrated by varying the frequency of the control loop and observing that safety is also conserved.

In the second part, we propose to unify set-based and trajectory-based safety critical control approaches and show that they are really just two sides of the same coin. We present a safety critical control framework that combines the strengths of both approaches to deliver efficient and scalable methods of ensuring safety for complex dynamical systems. First, we show how it is possible to systematically define a control-invariant subset of the safety set, namely a Safe Backward Image (SBI) of the backup set. This set is defined implicitly from a backup control law and a backup set, and the implementation details of a safety filter in that context are then presented. We then show that one can relax the stability requirement on the backup control law and still get meaningful albeit weaker safety guarantees. In a following section, we explore the relation between optimality of the backup control law and size of the Safe Backward Image. In the case of linear systems, we show that it is possible to implement and couple a Model Predictive Controller and a Safety Filter to obtain the largest possible Safe Backward Image. Finally, methods of selecting a safe input with varying levels of trade-off between conservativeness and computational complexity are proposed and illustrated on relevant systems and applications, namely: a two-wheeled inverted pendulum (Segway), an industrial manipulator, a quadrotor, and a lower body exoskeleton.

Chapter 2

EXPLICIT SAFETY FILTERING

2.1 Set Invariance Conditions

We are now properly set to start reasoning rigorously about the system and its behavior. In this work, we will focus on **safety** and define it as follow:

Definition 2. Given a safety set $\bar{S} \in \mathbb{R}^n$, a control system $\Sigma = (\bar{x}, \bar{u})$ is safe at a time t if its state at that time t is in the safety set, i.e. $\bar{x}(t) \in \bar{S}$.

As discussed in the introduction, our goal is to characterize what needs to be done for the system to stay safe during its operation. Therefore, we first need to derive sufficient conditions under which the system is safe.

2.1.1 Control Invariant Sets

In order to understand the nature of these conditions, we need to understand the structure of safety sets. The following results will be given without proof and we refer the reader to [8, 14] for more details.

We will denote by $\mathcal{L}_U$ the set of piecewise locally Lipschitz continuous scalar functions taking values in set U and start by introducing these ideas in the case where we have access to an exact model of the system and without any consideration about realizability. Let us therefore consider an exact model $\mathcal{M} = (X, U, F)$ with single valued control affine dynamics:

$$F(x, u) = f(x) + g(x)\, u. \tag{2.1}$$

For any given input $u \in \mathcal{L}_U$, there is a unique state x forming a practical solution to $\mathcal{M}$ and defined for all time [26].

Remark 4. In this work, we will derive all results for control affine models, but the results can be extended to more generic models.

If the safety set $\bar{S}$ is chosen arbitrarily, then there will almost always exist a subset S_d of $\bar{S}$ that has finite escape time, i.e. if $x(t_0) \in S_u, \implies \forall u \in \mathcal{L}_U,\ \exists t_e \geq t_0,\ x(t) \notin \bar{S}$. In other words, if the system finds itself in S_u, then no control policy can constrain solutions of $\mathcal{M}$ to remain inside $\bar{S}$ for all times

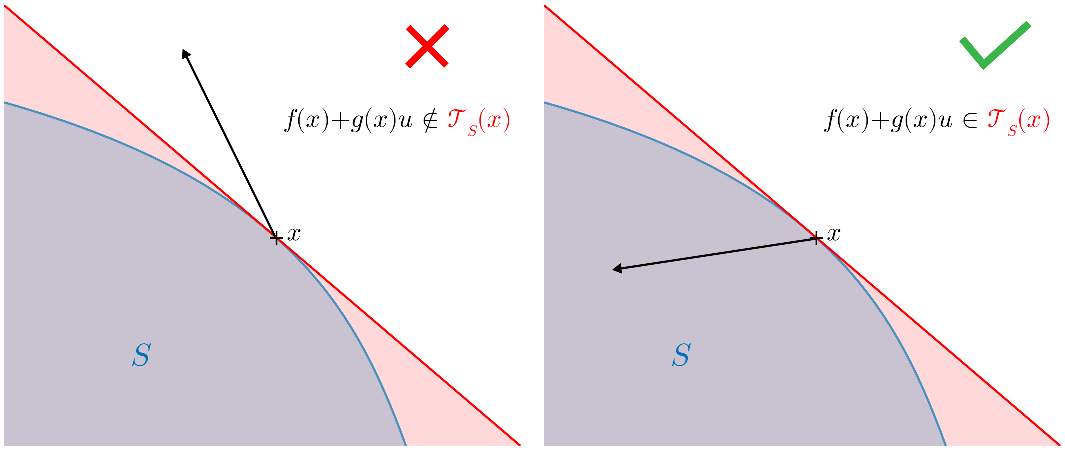

$f(x){+}g(x)u \notin \mathcal{T}_S(x)$
x
S
$f(x){+}g(x)u \in \mathcal{T}_S(x)$
x
S

Figure 2.1: Illustration of the subtangentiality condition for exact models.

in the future. The main difficulty in controlling the system to remain in $\bar{S}$ is therefore to avoid this dangerous subset of the safety set.

The largest subset of a given safety set that does not contain any such dangerous region is usually referred to as the **viability kernel** [8]. This property of a set to not contain dangerous states is called controlled invariance. Remaining in the safety set for all time is therefore equivalent to staying inside the viability kernel for all time.

Definition 3. A closed set S is **control invariant** for an exact model $\mathcal{M}$ if for any $x\left(t_0\right) \in S$, their exist a control input $u \in \mathcal{L}_U$ such that the associated state solution to $\mathcal{M}$ satisfies: $\forall t \geq t_0,\ x(t) \in S$.

Definition 4. A closed set S is **invariant** for an exact model $\mathcal{M}$ under a policy $u \in \mathcal{L}_U$ if for any $x\left(t_0\right) \in S$, the associated state solution to $\mathcal{M}$ satisfies: $\forall t \geq t_0,\ x(t) \in S$.

Staying inside the viability kernel of the safety set is necessary to remain safe, but staying inside a control invariant subset of the safety set is sufficient. And as we will now see, having access to a control invariant subset of the safety set is actually the key that will allow us to characterize safe input law.

2.1.2 Regulation Map for Exact Models

Let $\mathcal{T}_S(x)$ denote the contingent cone to a set S at x as defined in [8, 14]:

Definition 5 (Contingent Cone)**.** Given a closed set S, the contingent cone to S at x is given by:

$$\mathcal{T}_S(x) = \left\{ z \in \mathbb{R}^n \;\middle|\; \liminf_{\tau \to 0} \frac{\text{dist}\,(x + \tau z, S)}{\tau} = 0 \right\}, \tag{2.2}$$

with:

$$\text{dist}\,(y, S) = \inf_{w \in S} \|y - w\|_*, \tag{2.3}$$

for any relevant norm $\| \cdot \|_*$.

The contingent cone is therefore the set of directions that "point towards the inside" of set S. It is empty outside of S, equal to $\mathbb{R}^n$ in the interior of S, and only non-trivial on the boundary of S. With that in mind, we can present **Nagumo's theorem**.

Theorem 1. *Given model* (2.1) *and a control input $u \in \mathcal{L}_U$, a set S is invariant if and only if for any $x(t_0) \in S$ and almost all $t \geq t_0$:*

$$f(x(t)) + g(x(t))u\,(t) \in \mathcal{T}_S(x(t)). \tag{2.4}$$

This subtangentiality condition is actually fairly intuitive as it essentially means that the dynamics has to be "pointing" towards the "inside" of the set S (cf. Fig. 2.1). Nagumo's theorem therefore gives us a local condition relating state and input and can hence be used to design safe control laws. Indeed, if the set S is a control invariant subset of the safety set, then it is possible to find inputs that regulate a safe solution for model (2.1), and these inputs are the one satisfying (2.4). It naturally follows the definition of a regulation map (the term *regulation map* being borrowed from [8] and specialized for our application).

Definition 6. The **regulation map** of a closed set S for a model $\mathcal{M}$ is defined by:

$$\overline{U_S}\,(x) \triangleq \{u \in U \mid F(x) + G(x)u \subseteq \mathcal{T}_S(x)\}\,. \tag{2.5}$$

In the exact model case, it is given by:

$$\overline{U_S}\,(x) = \{u \in U \mid f(x) + g(x)u \in \mathcal{T}_S(x)\}\,. \tag{2.6}$$

Therefore, as long as the input takes values in the regulation map of subset of the safety set, the state is guaranteed to remain inside that set. This regulation map is however non-empty in all of S only if S is a control invariant set, hence the necessity of considering control invariant subsets of the safety set. The notion of **safe input** follows naturally.

Definition 7. Let $\mathcal{M}$ be a model of a control system Σ and S be a closed subset of the safety set $\bar{S}$. Then u is a **safe input** at time t if $u(t) \in \overline{U_S}\left(x(t)\right)$ for some solution (x, u) of $\mathcal{M}$.

Remark 5. If $\bar{u}(t) \in \overline{U_S}\left(\bar{x}(t)\right)$, we will say that the system's input is safe at time t.

Finally, if the model is valid, safe inputs can be used to guarantee safety of the system itself.

Proposition 1. *Let $\mathcal{M}$ be an exact model of a control system Σ and S be closed subset of the safety set $\bar{S}$. If $\mathcal{M}$ is valid, $\bar{x}(t_0) \in S$ and for all $t \geq t_0$, the system's input $\bar{u}$ is safe, then for all $t \geq t_0$, Σ will be safe.*

Proof. From the Def. 7 of a safe input, and Thm. 1, we know that state solutions to $\mathcal{M}$ starting in S at t_0 and whose associated input is safe, are themselves safe for all $t \geq t_0$. Since $\mathcal{M}$ is valid, $(\bar{x}, \bar{u})$ is a solution of $\mathcal{M}$, so because $\bar{x}(t_0) \in S$ and for all $t \geq t_0$, the system's input $\bar{u}$ is safe, then $\bar{x}$ is safe for all $t \geq t_0$. □

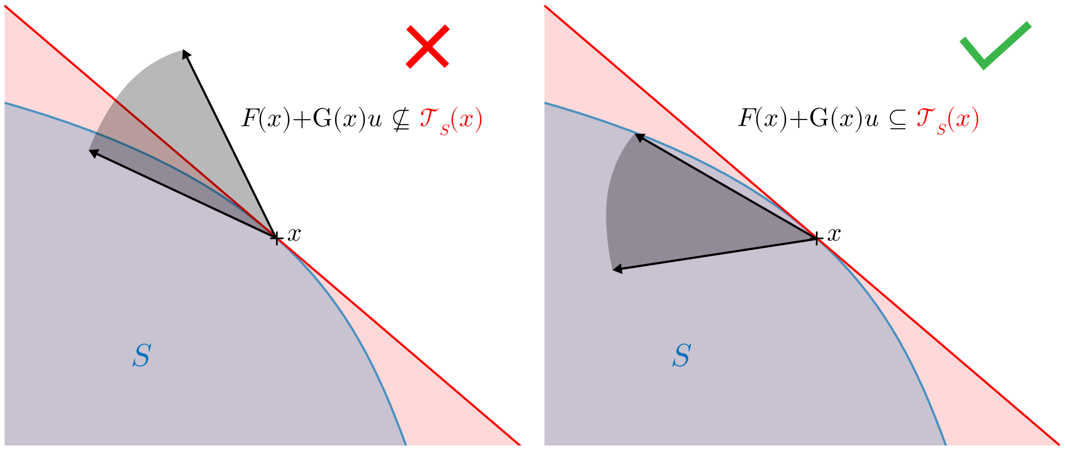
We now understand why it is important for S to be a control invariant subset of the safety, as otherwise $\bar{u}$ can never be chosen safe and the conditions of Prop. 1 can never be met. Similarly and as we have already discussed, it is essentially impossible to verify that an exact model is valid. Therefore, we need to revisit this set invariance condition in the more relevant case of inexact model.

2.1.3 Robust Invariant Sets

Let us consider an inexact model $\mathcal{M} = (X, U, F)$ with control affine dynamics:

$$F(x, u) = F\left(x\right) + G\left(x\right) u. \tag{2.7}$$

Definition 8. A closed set S is **robust control invariant** for a model $\mathcal{M}$ if for any $x\left(t_0\right) \in S$, there exist a control input $u \in \mathcal{L}_U$ such that **all** the associated state solutions to $\mathcal{M}$ satisfies: $\forall t \geq t_0,\ x(t) \in S$.



$F(x)+\mathrm{G}(x)u \nsubseteq \mathcal{T}_S(x)$
x
S
$F(x)+\mathrm{G}(x)u \subseteq \mathcal{T}_S(x)$
x
S

Figure 2.2: Illustration of the subtangentiality condition for inexact models.

Definition 9. A closed set S is **robust invariant** for a model $\mathcal{M}$ under a policy $u \in \mathcal{L}_U$ if for any $x(t_0) \in S$, **all** the associated state solutions to $\mathcal{M}$ satisfies: $\forall t \geq t_0,\ x(t) \in S$.

2.1.4 Regulation Map for Inexact Models

We can now present the robust counterpart of Nagumo's theorem (cf. [8] for more details and proof).

Theorem 2. *Given model (2.7) and a control input $u \in \mathcal{L}_U$, then a set S is robust invariant if and only if for any $x(t_0) \in S$ and almost all $t \geq t_0$:*

$$F(x(t)) + G(x(t))u(t) \subseteq \mathcal{T}_S(x(t)). \tag{2.8}$$

Again, this robust subtangentiality condition is fairly intuitive and can be visualized in Fig. 2.2. It essentially means that all the possible single valued dynamics captured by the model have to be "pointing" towards the "inside" of the set S. Similarly to the exact model case, if the model is valid, these safe input laws can be used to guarantee safety of the system.

Definition 7 remains valid in this imperfect model case and Prop. 1 becomes :

Proposition 2. *Let $\mathcal{M}$ be a model of a control system Σ and S be a closed subset of the safety set $\bar{S}$. If $\mathcal{M}$ is valid, $\bar{x}(t_0) \in S$ and for all $t \geq t_0$, the system's input $\bar{u}$ is safe, then for all $t \geq t_0$, Σ will be safe.*

Proof. From the Def. 7 of a safe input, and Thm. 2, we know that state solutions to $\mathcal{M}$ starting in S at t_0 and whose associated input is safe, are

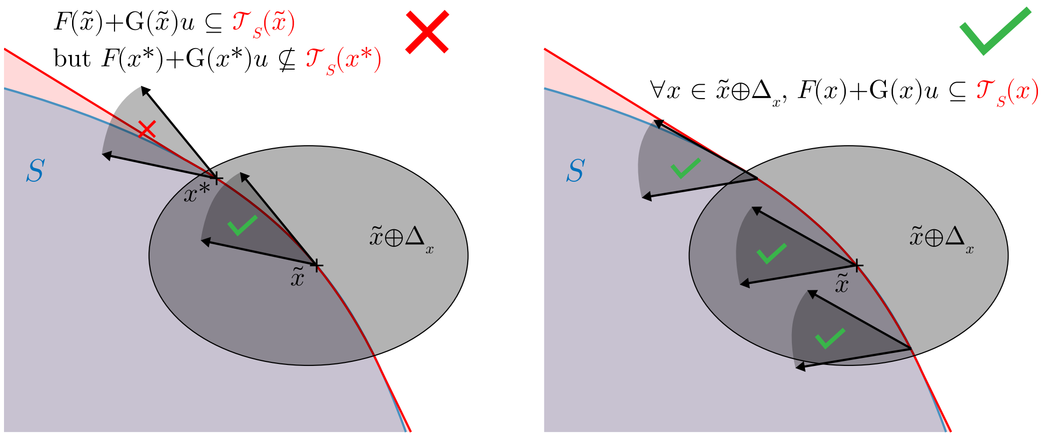

$F(\tilde{x})+G(\tilde{x})u \subseteq \mathcal{T}_S(\tilde{x})$
but $F(x^*)+G(x^*)u \nsubseteq \mathcal{T}_S(x^*)$
S
x^*
$\tilde{x}$
$\tilde{x}\oplus\Delta_x$
$\forall x \in \tilde{x}\oplus\Delta_x,\ F(x)+G(x)u \subseteq \mathcal{T}_S(x)$
S
$\tilde{x}$
$\tilde{x}\oplus\Delta_x$

Figure 2.3: Illustration of the realizable subtangentiality condition for inexact models.

themselves safe for all $t \geq t_0$. Since $\mathcal{M}$ is valid, $(\bar{x}, \bar{u})$ is a solution of $\mathcal{M}$, so because $\bar{x}(t_0) \in S$ and for all $t \geq t_0$, the system's input $\bar{u}$ is safe, then $\bar{x}$ is safe for all $t \geq t_0$. $\square$

The confidence we can have in the safety of a system is therefore no lower than the confidence we have in the validity of its model.

2.2 Realizable Set Invariance for Cyber Physical System

2.2.1 Realizable Regulation Map

The safety guarantees provided by Prop. 2 are contingent on the system's input being safe. But as discussed in Sec. 1.2.4, guaranteeing safety of the input is not that simple, and one has to account for many physical and technological constraints.

So to benefit from the safety guarantees claimed by Prop. 2, one needs to design control laws such that once implemented:

$$\bar{u}(t) \in \overline{U_S}\left(\bar{x}(t)\right),\ \forall t \geq t_0, \tag{2.9}$$

which is essentially a validity constraint on the control law as a model of the controller. We therefore need to define a new type of regulation map in light of the implementation constraint discussed in Sec. 1.2.4 to make it realizable. In particular, we do not have access to the state and input of the system, but only to imperfect measurements of these physical quantities. Let us therefore design a regulation map that is only expressed as function of such estimates $\tilde{x}$ and $\tilde{u}$.

Definition 10. The **realizable regulation map** of a closed set S for a model $\mathcal{M}$ with state uncertainty Δ_x and input uncertainty Δ_u is defined by:

$$\widetilde{U}_{S,\Delta_x,\Delta_u}\left(x\right) \triangleq \left\{u \in U \mid \forall \varepsilon_x \in \Delta_x,\ F\left(x+\varepsilon_x\right) + G\left(x+\varepsilon_x\right) u_\Delta \subseteq \mathcal{T}_S\left(x+\varepsilon_x\right)\right\}, \quad (2.10)$$

where $u_\Delta \triangleq u \oplus \Delta_u$.

We can now use this realizable regulation map to ensure the system's input safety from state estimate and desired inputs.

Proposition 3. *If $\tilde{u}(t) \in \widetilde{U}_{S,\Delta_x,\Delta_u}\left(\tilde{x}(t)\right)$, $\mathcal{M}$ is valid, and assumptions 1 and 2 hold, then the system's input is safe, i.e. $\bar{u}(t) \in \overline{U_S}\left(\bar{x}(t)\right)$ for all time.*

Proof. The proof follows directly from definition 10 of realizable regulation map and assumptions 1 and 2. Indeed, if $\tilde{u}(t) \in \widetilde{U}_{S,\Delta_x,\Delta_u}\left(\tilde{x}(t)\right)$, then for all $\varepsilon_x \in \Delta_x$ and $\varepsilon_u \in \Delta_u$:

$$F\left(\tilde{x}+\varepsilon_x\right) + G\left(\tilde{x}+\varepsilon_x\right)\left(\tilde{u}+\varepsilon_u\right) \subseteq \mathcal{T}_S\left(\tilde{x}+\varepsilon_x\right). \quad (2.11)$$

If assumptions 1 and 2 hold, then there exists $\bar{\varepsilon}_x \in \Delta_x$ and $\bar{\varepsilon}_u \in \Delta_u$ such that $\bar{x} = \tilde{x} + \bar{\varepsilon}_x$ and $\bar{u} = \tilde{u} + \bar{\varepsilon}_u$. So in particular:

$$F\left(\tilde{x}+\bar{\varepsilon}_x\right) + G\left(\tilde{x}+\bar{\varepsilon}_x\right)\left(\tilde{u}+\bar{\varepsilon}_u\right) \subseteq \mathcal{T}_S\left(\tilde{x}+\bar{\varepsilon}_x\right), \quad (2.12)$$

hence:

$$F\left(\bar{x}\right) + G\left(\bar{x}\right)\bar{u} \subseteq \mathcal{T}_S\left(\bar{x}\right), \quad (2.13)$$

and $\bar{u}(t) \in \overline{U_S}\left(\bar{x}(t)\right)$. □

Ensuring that the desired inputs always belong to the realizable regulation map is therefore sufficient for the system's input to belong to the regulation map and to get the safety guarantees provided by Prop. 2. This result is also fairly intuitive and illustrated in Fig. 2.3, the idea being to ensure safety of the system's input for all possible values of the state at a given instant.

Note however that for this realizable regulation map to be usable, it must never be empty, which constrains what the set S can be. We will call the sets for which the associated realizable regulation map is never empty **realisable robust control invariant set**. We will see in Sec. 2.4 how one can compute such a set.

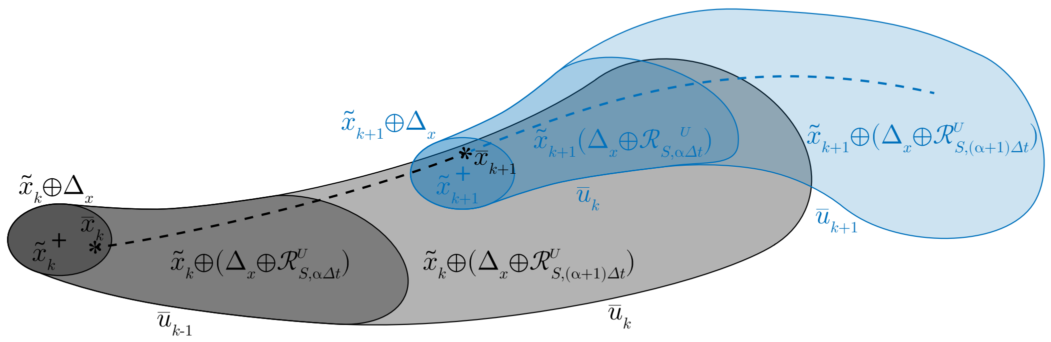

$\tilde{x}_{k+1} \oplus \Delta_x$
$\tilde{x}_{k+1}(\Delta_x \oplus \mathcal{R}^U_{S,\alpha\Delta t})$
$\tilde{x}_{k+1} \oplus (\Delta_x \oplus \mathcal{R}^U_{S,(\alpha+1)\Delta t})$
$\overline{x}_{k+1}$
$\tilde{x}_{k+1}$
$\overline{u}_k$
$\overline{u}_{k+1}$
$\tilde{x}_k \oplus \Delta_x$
$\overline{x}_k$
$\tilde{x}_k$
$\tilde{x}_k \oplus (\Delta_x \oplus \mathcal{R}^U_{S,\alpha\Delta t})$
$\tilde{x}_k \oplus (\Delta_x \oplus \mathcal{R}^U_{S,(\alpha+1)\Delta t})$
$\overline{u}_{k\text{-}1}$
$\overline{u}_k$

Figure 2.4: Local reachable sets to consider for the realizable subtangentiality condition.

2.2.2 State Uncertainty for Digital Implementation

As discussed in Sec. 1.2.4.2, when control laws are implemented using digital computers, the state estimates are only known at discrete times, and changes to the input can only be made with a non-zero delay after the state sample times. Therefore, it is impractical to directly enforce that $\tilde{u}(t) \in \widetilde{U}_{S,\Delta_x,\Delta_u}(\tilde{x}(t))$ for all $t \geq t_0$.

The issue in terms of input safety is that decisions taken at discrete instants have to carry over these safety guarantees to the time intervals between the discrete state sampling instants. In other words, the input has to be safe for all states along the system's trajectory between these discrete instants.

In order to achieve such guarantees, we will consider the realizable regulation map formulation (10) with an added state uncertainty corresponding to the possible variations of the system's state due to its evolution between the discrete sampling instants.

To characterise these possible variations, we will utilize reachable sets (see [61] for methods to compute such sets).

Definition 11. The **reachable set** for a model $\mathcal{M}$ with input $u \in \mathcal{L}_U$, denoted $\mathcal{R}^U_{=t}(\mathit{\Delta})$, is the set of states reachable from $\mathit{\Delta}$ in time t:

$$\mathcal{R}^U_{=t}(\mathit{\Delta}) \triangleq \{x(t_0 + t) \mid \forall t_0 \in \mathbb{R},\ \forall (x, u) \in \mathcal{S}(\mathcal{M}) \text{ s.t. } x(t_0) \in \mathit{\Delta}\}. \quad (2.14)$$

Similarly:

Definition 12. The **interval reachable set** for a model $\mathcal{M}$, denoted $\mathcal{R}^U_{\leq t}(\mathit{\Delta})$, is the set of states reachable in time up to t, i.e.:

$$\mathcal{R}^U_{\leq t}\left(\varDelta\right) \triangleq \bigcup_{\tau\in[0,t]} \mathcal{R}^U_{=t}\left(\varDelta\right). \tag{2.15}$$

Let us now introduce the idea of the local reachable set.

Definition 13. The **Local Reachable Set** over a set S for a model $\mathcal{M}$ with input $u \in \mathcal{L}_U$ and for a time horizon $\varDelta_t$ is the set:

$$\mathcal{R}^U_{S,\varDelta_t} \triangleq \bigcup_{x\in S} \left(\mathcal{R}^U_{\leq \varDelta_t}\left(x\right) \ominus x\right). \tag{2.16}$$

In other words, the local reachable set is the largest neighborhood that can be visited by the system over an interval $\varDelta_t$ for any states in S. It is therefore easy to see that the following proposition holds.

Proposition 4. *If given a valid model $\mathcal{M}$ such that assumptions 1, 2, and 3 hold, and for some $k \geq 0$:*

$$\tilde{u}_k \in \widetilde{U}_{S,\varDelta_x^+,\varDelta_u}\left(\tilde{x}_k\right), \tag{2.17}$$

with:

$$\varDelta_x^+ \triangleq \varDelta_x \oplus \mathcal{R}^U_{S,(\alpha+1)\varDelta_t}, \tag{2.18}$$

then for all $t \in [t_k + \alpha\varDelta_t, t_{k+1} + \alpha\varDelta_t]$:

$$\bar{u}(t) \in \overline{U_S}\left(\bar{x}(t)\right). \tag{2.19}$$

Proof. From assumption 1 and 3, and from Def. 13, we know that for all $t \in [t_k, t_{k+1} + \alpha\varDelta_t]$, $\bar{x}(t) \in \tilde{x}_k \oplus \varDelta_x^+$, hence this holds in particular for all $t \in [t_k + \alpha\varDelta_t, t_{k+1} + \alpha\varDelta_t]$. So if $\tilde{u}_k \in \widetilde{U}_{S,\varDelta_x^+,\varDelta_u}\left(\tilde{x}_k\right)$, then for all $\varepsilon_x \in \varDelta_x^+$ and $\varepsilon_u \in \varDelta_u$:

$$F\left(\tilde{x}_k + \varepsilon_x\right) + G\left(\tilde{x}_k + \varepsilon_x\right)\left(\tilde{u}_k + \varepsilon_u\right) \subseteq \mathcal{T}_S\left(\tilde{x}_k + \varepsilon_x\right), \tag{2.20}$$

and therefore for all $t \in [t_k + \alpha\varDelta_t, t_{k+1} + \alpha\varDelta_t]$ and $\varepsilon_u \in \varDelta_u$:

$$F\left(\bar{x}(t)\right) + G\left(\bar{x}(t)\right)\left(\tilde{u}_k + \varepsilon_u\right) \subseteq \mathcal{T}_S\left(\bar{x}(t)\right). \tag{2.21}$$

From assumptions 2 and 3, we know that for all $t \in [t_k + \alpha\varDelta_t, t_{k+1} + \alpha\varDelta_t]$:

$$\bar{u}(t) = \tilde{u}_k + \bar{\varepsilon}_u \tag{2.22}$$

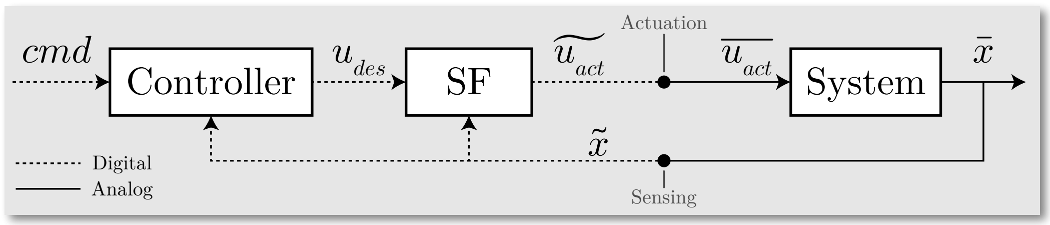


Figure 2.5: Safety filtering control structure.

for some $\bar{\varepsilon}_u \in \varDelta_u$, so for all $t \in [t_k + \alpha\varDelta_t, t_{k+1} + \alpha\varDelta_t]$:

$$F\left(\bar{x}(t)\right) + G\left(\bar{x}(t)\right)\bar{u}(t) \subseteq \mathcal{T}_S\left(\bar{x}(t)\right). \tag{2.23}$$

□

This proposition (illustrated in Fig. 2.4) gives a way to ensure safety of the system's input over the entire trajectory of the system by only having to change the desired input at discrete instants. It also characterizes the necessary trade-off between controller frequency and added conservatism as if $\varDelta_t$ increases, so does the size of $\mathcal{R}^U_{S,\varDelta_t}$ which in turn makes the size of $\widetilde{U}_{S,\varDelta_x^+,\varDelta_u}$ decrease.

2.3 Safety Filtering

The controller design task generally consists of finding a control policy that maximizes some performance criteria while ensuring safety of the system. Finding a high performing policy that is safe by construction is difficult. System performance and safety are often conflicting goals, and guarantees on safety of the system generally become much more challenging to get as complexity of the controller increases. As an alternate paradigm, we consider the control structure depicted in Fig. 2.5. The idea here is to decouple performance from the enforcement of hard safety constraints in such a way that prioritizes the latter over the former. Given a nominal controller that processes commands and focuses on performing the desired task, a safety filter can be used to preempt these desired inputs in a way that ensures safety of the system when necessary (see [16, 38, 85] for application examples). Ideally, the filter is minimally invasive to the desired input, i.e. u_{des} is left unmodified as long as the signal is not compromising to system safety. We will now present a way to realize such a filter.

2.3.1 Sub-Regulation Map for Exact Models

As discussed in the previous section, for a safety filter to be effective at guaranteeing safety of the system, it has to constrain the system's input to remain safe, i.e. to be inside the regulation map. Depending on the type of set considered, there are different ways of expressing the regulation map. Practical sets—as defined in [14]—are suitable for most realistic cases and make it convenient to express the contingent cone. To describe such sets, one only needs to consider N_s continuously differentiable functions $h_i : \mathbb{R}^n \to \mathbb{R}$ such that [1] [2]:

$$\begin{aligned} S &= \{x \in \mathbb{R}^n \mid \forall i \in [\![1, N_s]\!]\,,\ h_i\,(x) \geq 0\} \\ \partial S &= \{x \in S \mid \exists i \in [\![1, N_s]\!]\,,\ h_i\,(x) = 0\}\,. \end{aligned} \tag{2.24}$$

For such sets, the contingent cone can be expressed as:

$$\mathcal{T}_S(x) = \{z \in \mathbb{R}^n \mid \forall i \in \mathrm{Act}(x),\ \nabla h_i(x).z \geq 0\}\,, \tag{2.25}$$

with:

$$\mathrm{Act}(x) \triangleq \{i \in [\![1, N_s]\!] \mid h_i(x) = 0\}\,. \tag{2.26}$$

In that case, the subtangentiality condition (2.4) can be written as

$$TC_i\,(x, u) \triangleq L_f h_i(x) + L_g h_i(x) u, \tag{2.27}$$

for all $x \in \partial S$, and $i \in \mathrm{Act}(x)$. Here, $L_f h$ and $L_g h$ denote the Lie derivatives of h along f and g, respectively. The regulation map then becomes:

$$\overline{U_S}\,(x) = \{u \in U \mid \forall i \in \mathrm{Act}(x),\ TC_i\,(x, u) \geq 0\}\,. \tag{2.28}$$

The subtangentiality condition is however not very practical as it only defines a non-trivial set of admissible inputs when the system is on the boundary of S, i.e. $\mathrm{Act}(x) = \emptyset$ if $x \notin \partial S$, which leads to a discontinuous regulation of the control input. Furthermore, it is not defined outside of S which makes any real implementation essentially impossible. One solution to smooth out the regulated input and make the safety filter implementation possible is explored in [8]—further developed in [5]. It consists of considering a strengthening term in (2.27) and of imposing **barrier conditions**:

$$BC_i\,(x, u) \triangleq L_f h_i(x) + L_g h_i(x) u + \alpha_i\,(h_i(x))\,, \tag{2.29}$$

[1] See [14, p. 103] for all conditions under which S is practical.

[2] $[\![m, n]\!]$ will denote the set of integers from n to m

for all $x \in S$, $i \in [\![1, N_s]\!]$ with the extended class $\mathcal{K}$ **strengthening** functions $\alpha_i : \mathbb{R} \to \mathbb{R}$. This barrier condition defines a **sub-regulation map** U_S of admissible inputs:

$$U_S(x) \triangleq \{u \in U \mid \forall i \in [\![1, N_s]\!],\ BC_i(x, u) \geq 0\}. \tag{2.30}$$

In the rest of this work, we will use the term *sub-regulation map* (borrowed again from [8]) to refer to any map whose image is contained inside a particular regulation map.

Because for all $x \in S$, $U_S(x) \subseteq \overline{U_S}(x)$, enforcing that the input stays inside a sub-regulation map is therefore sufficient to ensure safety of the input, and this holds for any sub-regulation map, i.e. map whose image is a subset of the regulation map.

Note, however, that special care must be take to ensure that a sub-regulation map does not take empty value, otherwise it will not be usable for filtering desired inputs. In this particular case, one has to choose the strengthening functions α_i accordingly, as discussed in [8, 38], which is always possible under the present assumptions.

2.3.2 Safety Filter as a Quadratic Program

In light of these results, it is now possible to realize a safety filter that can enforce input's safety, hence system's safety, synergistically with the performance goals of the system. Given a desired input u_{des} provided by a nominal controller (cf. Fig. 2.5), enforcing safety in a **minimally invasive** way can be naturally formulated as the following quadratic program:

Safety Filter QP

$$\begin{aligned} \overline{u_{\text{act}}}(t, x) = \underset{u \in U}{\operatorname{argmin}}\ & \|u_{\text{des}}(t) - u\|^2 \\ \text{s.t. } & BC_i(x, u) \geq 0,\ \forall i \in [\![1, N_s]\!]. \end{aligned} \tag{2.31}$$

If S is control invariant, $\overline{U_S}$ has non-empty compact convex values, but it is not necessarily the case for U_S. As discussed before, this can be addressed by either choosing a control invariant set S and then finding the α_i functions accordingly so as to ensure that U_S is never empty, or by first choosing the α_i functions and then computing a set S over which U_S is never empty.

Alternatively, given a control invariant set S, barrier conditions can be used along with their smoothing action without having to carefully choose the α_i by using the following safety filter formulation. Given $\alpha_{\text{des},i}$ desired smoothing factors and a relaxation penalty factor $M \gg 0$, we can define a relaxed safety filter:

Relaxed Safety Filter QP

$$\begin{aligned} \overline{u_{\text{act}}}(t,x) = \underset{\substack{u \in U \\ \alpha_i \geq \alpha_{\text{des},i}}}{\operatorname{argmin}} \ & \|u_{\text{des}}(t) - u\|^2 + M \sum_{i=1}^{N_s} (\alpha_{\text{des},i} - \alpha_i)^2 \\ \text{s.t. } & L_f h_i(x) + L_g h_i(x) u + \alpha_i h_i(x) \geq 0 \\ & \forall i \in [\![1, N_s]\!]. \end{aligned} \tag{2.32}$$

This way, the *slowing down* effect of a barrier condition can be chosen and followed if possible, but otherwise relaxed in a way that still ensures that the resulting filtered input is in $\overline{U_S}$. Note also that it is possible to only consider a single α_i to relax all of the constraints at once, this way reducing the size of the resulting QP which can be beneficial on embedded systems with limited computational capabilities.

2.3.3 Safety Filter for Inexact Models

Let us now look at how to express the sub-regulation map for inexact models, and in particular for parametric models as defined in (1.3). In this case, a robust barrier condition can be defined of the form:

$$RBC_i\,(x,u,p) \triangleq L_f h_i(x,p) + L_g h_i(x,p) u + \alpha_i(h_i(x)), \tag{2.33}$$

and the associated sub-regulation map is:

$$U_S\,(x) = \{u \in U \mid \forall i \in [\![1, N_s]\!]\,,\ \forall p \in P,\ RBC_i\,(x,u,p) \geq 0\}\,. \tag{2.34}$$

Therefore, an associated robust safety filter can be formulated as:

Robust Safety Filter

$$\begin{aligned}\overline{u_{\text{act}}}(t,x) = \underset{u\in U}{\operatorname{argmin}}\ & \left\|u_{\text{des}}\left(t\right)-u\right\|^2 \\ \text{s.t. } & RBC_i\left(x,u,p\right)\geq 0, \\ & \forall i\in \llbracket 1,N_s\rrbracket\,,\ \forall p\in P.\end{aligned} \tag{2.35}$$

This optimization problem, however, belongs to the class of robust optimization problems[11, 12]. In general, the dependency in the uncertainty terms is **nonlinear** making the problem numerically intractable. The fundamental idea that will allow us to overcome this issue is that by over-approximating the effect of uncertainty on the barrier condition, we can under-approximate the sub-regulation map.

The key to this approach is the following Lemma.

Lemma 1. *Let $f:\mathbb{R}^m\longrightarrow\mathbb{R}$ and $g:\mathbb{R}^n\longrightarrow\mathbb{R}^m$ be any two continuous functions. Let $A\in\mathbb{R}^n$ be a compact set and:*

$$\begin{aligned}B &\triangleq \left\{g\left(x\right)\in\mathbb{R}^m \mid x\in A\right\}\\ C &\triangleq \left\{f\left(y\right)\in\mathbb{R} \mid y\in B'\supseteq B\right\}\\ D &\triangleq \left\{f\circ g\left(x\right)\in\mathbb{R} \mid x\in A\right\},\end{aligned}$$

where B' is compact, then:

$$\left[\max_{y\in B'}\left(f\left(y\right)\right)\right]\leq 0 \implies \left[\max_{x\in A}\left(f\circ g\left(x\right)\right)\right]\leq 0. \tag{2.36}$$

Since:

$$\begin{gathered}\forall p\in P,\ RBC_i\left(x,u,p\right)\geq 0\\ \Longleftrightarrow\\ \left[\max_{p\in P}\left(-L_gh_i(x,p)u-L_fh_i(x,p)-\alpha_i(h_i(x))\right)\right]\leq 0,\end{gathered} \tag{2.37}$$

given maps A_i^P and B_i^P such that for all $x\in S$:

$$\begin{aligned}A_i^P\left(x\right) &\supseteq \left\{-L_gh_i(x,p)^\top \;\middle|\; \forall p\in P\right\}\\ B_i^P\left(x\right) &\supseteq \left\{-L_fh_i(x,p)-\alpha_i(h_i(x)) \mid \forall p\in P\right\},\end{aligned} \tag{2.38}$$

it follows from Lemma 1 that:

$$\begin{gathered}\left[\max_{\substack{a_i \in A_i^P(x)\\ b_i \in B_i^P(x)}} \left(a_i^\top u + b_i\right)\right] \leq 0,\\ \Longrightarrow \\ \left[\max_{p\in P} \left(-L_g h_i(x,p)u - L_f h_i(x,p) - \alpha_i(h_i(x))\right)\right] \leq 0.\end{gathered} \tag{2.39}$$

The key to making a robust and tractable safety filter is therefore to find A_i^P and B_i^P maps that have polytopic images. If one is able to provide such an over-approximation, then the difficult optimization problem (2.35) can be transformed into a QP with polytopic constraints that is easy to solve using available numerical solvers as in such case, it is possible to rewrite this linear program as:

$$\begin{aligned} c_i^*(x) = \max_{c_i} \quad & \left(c_i^\top v\right)\\ \text{s.t.} \quad & D_i\left(x\right) c_i \leq d_i\left(x\right), \end{aligned} \tag{2.40}$$

where $c_i \triangleq \left[a_i^\top | b_i\right]^\top$ and $v \triangleq [u|1]^\top$, and with $d_i\left(x\right)$ and $D_i\left(x\right)$ encoding the polytopic constraints $a_i \in A_i^P(x)$ and $b_i \in B_i^P(x)$.

Assuming (2.40) is a feasible linear program, strong duality holds and:

$$\begin{aligned} c_i^*(x) = \min_{\lambda_i \geq 0} \quad & \left(\lambda_i^\top d_i\left(x\right)\right)\\ \text{s.t.} \quad & \lambda_i^\top D_i\left(x\right) = v^\top. \end{aligned} \tag{2.41}$$

Therefore, we have that:

$$\left(\begin{aligned} \min_{u\in U} \quad & \left\|u_{\text{des}}\left(t\right) - u\right\|^2\\ \text{s.t.} \quad & c_i^*\left(x\right) \leq 0,\\ & \forall i \in [\![1, N_s]\!] \end{aligned}\right) = \left(\begin{aligned} \min_{\substack{u\in U\\ \lambda_i \geq 0}} \quad & \left\|u_{\text{des}}\left(t\right) - u\right\|^2\\ \text{s.t.} \quad & \lambda_i^\top d_i\left(x\right) \leq 0\\ & \lambda_i^\top D_i\left(x\right) = v^\top\\ & \forall i \in [\![1, N_s]\!], \end{aligned}\right), \tag{2.42}$$

which allows us to define the following robust safety filter that guarantees input safety and is tractable:

Robust Safety Filter QP

$$\begin{aligned}\overline{u_{\text{act}}}(t,x) = \underset{\substack{u\in U\\ \lambda_i\geq 0}}{\operatorname{argmin}} \ & \left\|u_{\text{des}}\left(t\right)-u\right\|^2 \\ \text{s.t. } & \lambda_i^\top d_i\left(x\right) \leq 0 \\ & \lambda_i^\top D_i\left(x\right) = v^\top \\ & \forall i \in [\![1, N_s]\!] \,.\end{aligned} \tag{2.43}$$

One must be careful now since the polytopic sub-regulation map defined by (2.43):

$$\underline{U_S}\left(x\right) = \left\{u \in U \;\middle|\; \forall i \in [\![1, N_s]\!]\,,\ \forall a_i \in A_i^P(x),\ \forall b_i \in B_i^P(x),\ \ a_i^\top u + b_i \leq 0\right\} \tag{2.44}$$

has images strictly smaller than the ones of the regulation map not just in the interior of S, but also on the boundary of S, i.e. for all $x \in \partial S$, $\underline{U_S}(x) \subset U_S(x) = \overline{U_S}(x)$. It is therefore impossible in general to find α_i functions that make (2.43) feasible everywhere in S.

To ensure feasibility of (2.43), one needs to find a subset of the safety set S where there exists function α_i such that the sub-regulation map is non-empty. We will call such a set a **Regulation Kernel**, and it immediately follows that regulation kernels are also robust control invariant sets. We will see in Sec. 2.4 how to compute such sets when using interval arithmetic [66] to define the A_i^P and B_i^P maps.

Finally, note that a relaxed version of (2.43) can be formulated without any added difficulty. For that, let us define modified versions of the maps (2.38):

$$\begin{aligned}\mathcal{A}_i^P\left(x\right) &\supseteq \left\{-L_g h_i(x,p)^\top \;\middle|\; \forall p \in P\right\} \\ \mathcal{B}_i^P\left(x\right) &\supseteq \left\{-L_f h_i(x,p) \mid \forall p \in P\right\}.\end{aligned} \tag{2.45}$$

In this case, the top inequality in (2.39) can be written as:

$$\left[\max_{\substack{a_i\in\mathcal{A}_i^P(x)\\ b_i\in\mathcal{B}_i^P(x)}} \left(a_i^\top u + b_i\right)\right] \leq \alpha_i h_i(x), \tag{2.46}$$

and so (2.43) can be written as:

Relaxed Robust Safety Filter QP

$$\begin{aligned}\overline{u_{\text{act}}}(t,x) = \underset{\substack{u\in U\\ \lambda_i\geq 0\\ \alpha_i\geq\alpha_{\text{des},i}}}{\operatorname{argmin}}\ & \left\|u_{\text{des}}\left(t\right)-u\right\|^2 + M\sum_{i=1}^{N_s}\left(\alpha_{\text{des},i}-\alpha_i\right)^2\\ \text{s.t. }\ & \lambda_i^\top\delta_i\left(x\right)\leq\alpha_i h_i(x)\\ & \lambda_i^\top\mathcal{D}_i\left(x\right)=v^\top\\ & \forall i\in[\![1,N_s]\!]\end{aligned} \tag{2.47}$$

which is indeed a QP where δ_i and $\mathcal{D}_i$ now encode the polytopic constraints $a_i\in\mathcal{A}_i^P(x)$ and $b_i\in\mathcal{B}_i^P(x)$.

2.3.4 Realizable Safety Filter

Let us now look at how to express the realizable sub-regulation map (2.10). The difficulty in this case comes from expressing the uncertainty in the contingent cone at a given point of the regulation map resulting from state uncertainty: $\mathcal{T}_S\left(\tilde{x}+\varDelta_x\right)$. In particular, for a given state measurement $\tilde{x}$, the dynamics have to be contained in the intersection of all the possible tangent cones in the neighborhood of $\tilde{x}$:

$$\begin{aligned}&\widetilde{U}_{S,\varDelta_x,\varDelta_u}\left(\tilde{x}\right)=\\ &\qquad\{u\in U\mid\forall x\in\tilde{x}\oplus\varDelta_x,\ \forall i\in\operatorname{Act}(x),\ \forall p\in P,\ \forall\varepsilon_u\in\varDelta_u,\\ &\qquad\qquad L_f h_i(x,p)+L_g h_i(x,p)(u+\varepsilon_u)\geq 0\}\,.\end{aligned}\tag{2.48}$$

Similarly to the inexact model case, this sub-regulation map is in general not polytopic because of the nonlinear dependence in parameters p, but moreover because of the nonlinear dependence in state x. We, therefore, define the following polytopic realizable sub-regulation map:

$$\begin{aligned}&\underline{\widetilde{U}}_{S,\varDelta_x,\varDelta_u}\left(\tilde{x}\right)=\\ &\qquad\left\{u\in U\,\middle|\,\forall a\in\widetilde{A}^{P,\varDelta_{x,u}}(\tilde{x}),\ \forall b\in\widetilde{B}^{P,\varDelta_{x,u}}(\tilde{x}),\ a^\top u+b\leq 0\right\},\end{aligned}\tag{2.49}$$

with:

$$\begin{aligned}\widetilde{A}^{P,\varDelta_{x,u}}\left(\tilde{x}\right)\supseteq\big\{-L_g h_i(x,p,\varepsilon_u)^\top\,\big|\,&\forall x\in\tilde{x}\oplus\varDelta_x,\\ &\forall i\in\operatorname{Act}(x),\ \forall p\in P,\ \forall\varepsilon_u\in\varDelta_u\big\}\\ \widetilde{B}^{P,\varDelta_{x,u}}\left(\tilde{x}\right)\supseteq\big\{-L_f h_i(x,p,\varepsilon_u)\,\big|\,&\forall x\in\tilde{x}\oplus\varDelta_x,\\ &\forall i\in\operatorname{Act}(x),\ \forall p\in P,\ \forall\varepsilon_u\in\varDelta_u\big\}.\end{aligned}\tag{2.50}$$

The resulting safety filter is given by:

Realizable Safety Filter QP

$$\begin{aligned} \overline{u_{\text{act}}}(t,\tilde{x}) = \underset{\substack{u\in U \\ \lambda\geq 0}}{\text{argmin}} \ & \left\|u_{\text{des}}\left(t\right)-u\right\|^2 \\ \text{s.t.}\ & \lambda^\top \widetilde{d}\left(\tilde{x}\right)\leq 0 \\ & \lambda^\top \widetilde{D}\left(\tilde{x}\right) = v^\top \end{aligned} \tag{2.51}$$

---

which is a QP with a single constraint where $\widetilde{d}$ and $\widetilde{D}$ now encode the polytopic constraints $a \in \widetilde{A}^{P,\Delta_{x,u}}$ and $b \in \widetilde{B}^{P,\Delta_{x,u}}$.

Contrary to the regulation map (2.28) for ideal models, (2.48) and (2.49) are non-trivial on a neighborhood of ∂S and not just on the boundary itself, which makes (2.51) practical. However, the region where the sub-regulation map is non-trivial is essentially of size Δ_x, which one would want to keep as small as possible for minimum conservatism, which conflicts with having a large region where the regulation map is non-trivial for maximum robustness of implementation.

Therefore, we propose a more practical realizable safety filter that provides more control over the behavior of the system inside of S:

Relaxed Realizable Safety Filter QP

$$\begin{aligned} \overline{u_{\text{act}}}(t,\tilde{x}) = \underset{\substack{u\in U \\ \lambda\geq 0 \\ \epsilon_i\geq 0}}{\text{argmin}} \ & \left\|u_{\text{des}}\left(t\right)-u\right\|^2 + M\sum_{i=1}^{N_s}\epsilon_i^2 \\ \text{s.t.}\ & \lambda^\top \widetilde{d}\left(\tilde{x}\right)\leq 0 \\ & \lambda^\top \widetilde{D}\left(\tilde{x}\right) = v^\top \\ & BC_i^m\left(\tilde{x},u\right) \geq -\epsilon_i,\ \forall i \in [\![1,N_s]\!] \end{aligned} \tag{2.52}$$

---

where:

$$BC_i^m\left(x,u\right) \triangleq L_f h_i(x,p^m) + L_g h_i(x,p^m)(u+\varepsilon_u^m) + \alpha_{\text{des},i}\left(h_i(x)\right), \tag{2.53}$$

for some mean values of the uncertainties $p^m \in P$ and $\varepsilon_u^m \in \Delta_u$. Indeed, in this safety filter formulation, the constraints $BC_i^m\left(\tilde{x},u\right) \geq -\epsilon_i$ can be arbitrarily relaxed, so if $\underline{\widetilde{U}}_{S,\Delta_x,\Delta_u}$ is never empty in S, (2.52) is always feasible. Therefore, by choosing M and $\alpha_{\mathrm{des},i}$ properly, it is possible to control the behavior of the system inside of S and not just near its boundary, while guaranteeing feasibility of the safety filter.

2.3.5 Scalable Safety Filter

Even though the sub-regulation map (2.29) has many advantages over the regulation map (2.28), as we talked about, it has one major inconvenient over the latter, namely that when the number N_s of parts of S gets large, the QP (2.31) becomes more time consuming to solve. Indeed, with this formulation, all the parts of S are considered in the QP, even though all that is necessary for safety is to consider the active set of parts of S when the system is on the boundary of S, which in practice consist of much fewer parts.

To address this issue, let $\widehat{N_s}$ be a number greater than the maximum number of active constraints for any given point in ∂S:

$$\widehat{N_s} \geq \widehat{N_{s,\max}} \triangleq \max_{x \in \partial S} |\mathrm{Act}(x)| \,. \tag{2.54}$$

Then:

$$\widehat{U_S}(x) \triangleq \left\{ u \in U \;\middle|\; \forall i \in \mathrm{Act}_{\widehat{N_s}}(x),\; BC_i(x,u) \geq 0 \right\}, \tag{2.55}$$

with $\mathrm{Act}_{\widehat{N_s}}(x)$ being the set of indices of the $\widehat{N_s}$-th smallest elements in the set $\{h_1(x), \ldots, h_{N_s}(x)\}$, is a sub-regulation map. Therefore, the advantages of (2.55) are that it is defined everywhere and piecewise smooth like (2.29), and also that it leads to a safety filter QP with a fixed and small number of constraints $\widehat{N_s}$ which makes its implementation very efficient:

Scalable Safety Filter QP

$$\begin{aligned} \overline{u_{\mathrm{act}}}(t,x) = \underset{u \in U}{\operatorname{argmin}} \;& \|u_{\mathrm{des}}(t) - u\|^2 \\ \text{s.t. } & BC_i(x,u) \geq 0,\; \forall i \in \mathrm{Act}_{\widehat{N_s}}(x). \end{aligned} \tag{2.56}$$

The same approach can be carried out without additional difficulty for a relaxed safety filter (2.32), a robust safety filter (2.43), a relaxed robust safety Filter (2.47) and a relaxed realizable safety filter (2.52).

2.4 Regulation Kernel Algorithm

2.4.1 The Viability Kernel

As discussed in the previous section, in order to use a safety filter, we need to find control invariant subsets of the safety set that are as large as possible, the largest possible subset being the Viability Kernel. As explained in [8], states that are inside the safety set but outside the viability kernel have finite escape time, i.e. for each of these dangerous states, all the possible emerging trajectories leave the safety set after some finite time.

The viability kernel can therefore be characterize as the solution to a reachability problem [64]. In particular, it is a solution to a continuous differential game where the player is trying to reach the complement of the safety set and an adversary is trying to keep the system inside the safety set through the system's control input. In this same paper, it is shown that the viability kernel can therefore be described as the solution to an Hamilton-Jacobi partial differential equation [64] in terms of the sublevel set of a function $v : X \times \mathbb{R} \to \mathbb{R}$:

$$\text{Viab} = \lim_{t \to -\infty} \{x \in X \mid v(x,t) \leq 0\}, \tag{2.57}$$

that is solution to the terminal value HJI PDE:

$$\frac{\partial v(x,t)}{\partial t} + \min \left[0, H\left(x, \frac{\partial s(x,t)}{\partial x}\right)\right] = 0, \tag{2.58}$$

with terminal condition:

$$v(x,0) = \overline{h}(x), \tag{2.59}$$

for any function $\overline{h}$ describing the safety set:

$$\overline{S} = \left\{x \in X \mid \overline{h}(x) \geq 0\right\}. \tag{2.60}$$

Here, the Hamiltonian for (2.1) is given by:

$$H(x,z) = \min_{u \in U} z^{\top} \left(f(x) + g(x)u\right). \tag{2.61}$$

In this next section, existing approaches for computing the viability kernel, or at least a large control invariant subsets of the safety set, are reviewed.

2.4.2 Existing Computational Approaches

2.4.2.1 Level Set Methods

A level set toolbox for MATLAB® was designed to directly find solutions to (2.58), hence allowing it to approximate viability kernels for generic non-linear systems [63]. The main drawback of this method is its exponential scaling with the dimension of the system, due to the required gridding of the state-space.

2.4.2.2 Parametric Methods

More recent algorithms based on parametric set representations, shown in [62], have also been used to approximate viability kernels of higher-dimensional linear systems. The general idea behind the algorithm is to begin with an over-approximation $\mathcal{K}_{\prime}$ of the viability kernel $\mathcal{K}$, and iteratively decrease the size through the following:

$$\mathcal{K}_{n+1} = K_0 \cap \{x_0 \mid \exists u, x(\rho) \in \mathcal{K}_n\}. \tag{2.62}$$

2.4.2.3 SOS Optimization Methods

Another recent approach to compute control invariant sets is SOS optimization. Interested readers can find more details in [47, 59, 80]. A polynomial p is a **sum of squares (SOS)** if there exist polynomials $\{q_i\}_{i=1}^{N}$ such that $p = \Sigma_{i=1}^{N} q_i^2$. Using the Lyapunov asymptotic stability theorem, we can characterize a region of attraction by:

$$B \triangleq \{x \in \mathbb{R}^n \mid V(x) \leq \gamma, \text{ and } \nabla V(x) f(x) \leq 0\}, \tag{2.63}$$

with $V(x)$ a positive definite function. Therefore, given a polynomial Lyapunov function and dynamics, one can formulate an SOS program that checks if:

$$\forall x \in \{x \mid V(x) \leq \gamma\}, \ \nabla V(x) f(x) \leq 0, \tag{2.64}$$

for a given $\gamma > 0$. The positivstellensatz characterization of polynomials (or S-procedure) tells us that if there exists a polynomial s that is SOS such that:

$$-(\nabla V(x) \cdot f + s(\gamma - V(x))) \text{ is SOS,}$$

then the contractiveness condition (2.64) is satisfied. Checking that a polynomial is SOS being equivalent to solving an SDP program, the problem is therefore a convex program. MATLAB® toolboxes such as SOSTOOLS and SPOT exist for these methods, which convert SOS or modified SOS constraints into an SDP problem. However, SDP solvers are still relatively slow and better-scaling method of checking SOS based on diagonally-dominant sum of squares optimization are presented in [60].

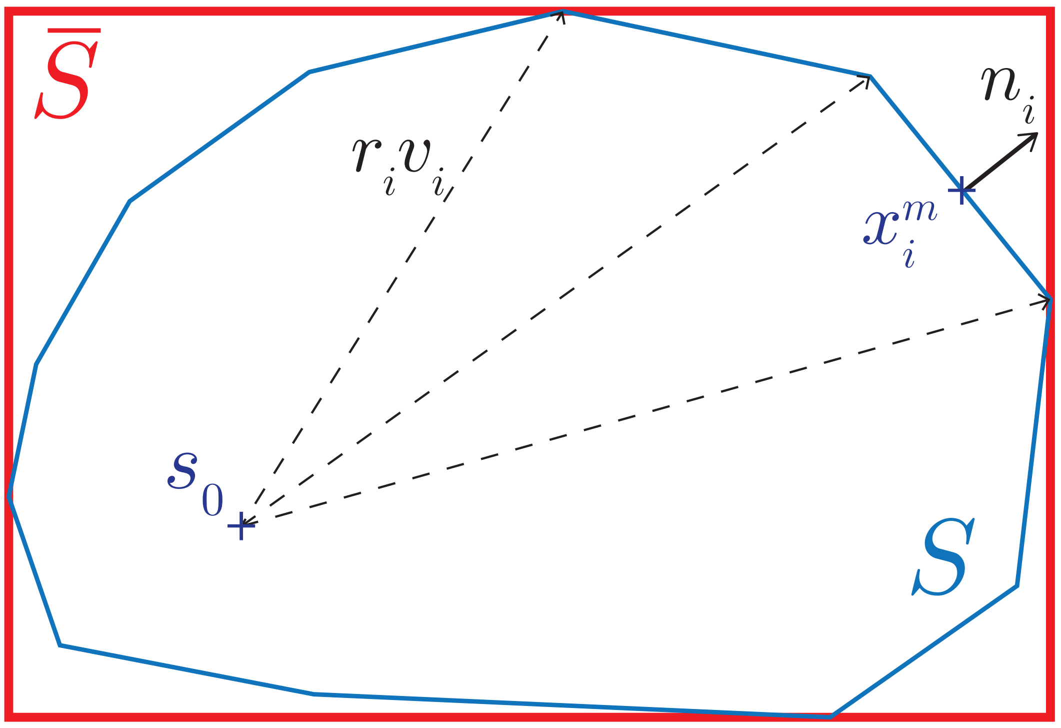


Figure 2.6: Illustration of the polytopic parameterization of S for the regulation kernel algorithms.

Similarly to viability algorithms, it is interesting to try and find the largest ROA possible. In that case, the problem becomes non-linear, but iterative approaches exist to try and overcome this increase in complexity by sequentially searching for a Lyapunov function and a region of attraction [47].

2.4.3 Regulation Kernel Algorithm

As discussed above, we need to find the largest subset of the safety set such that the sub-regulation map (2.44) is never empty. The main issue with all these approaches is that they do not factor in the added conservativeness of sub-regulation maps. They therefore cannot be used in the context of a robust safety filter for example. To find a regulation kernel, we propose a nonlinear optimization problem formulation that directly accounts for sub-regulation maps relying on interval arithmetic [66].

To that end, we first need to choose a type of set representation. In this work, we choose to work with polytopic sets S that are compact and convex (cf. Fig. 2.6). This representation makes it easy to express S as in (2.24) where:

$$h_i(x) = 1 - n_i^\top (x - s_0)\,, \tag{2.65}$$

for all facets of S and with $n_i^\top$ being an outward facing normal of that facet satisfying $n_i^\top (x - s_0) = 1$ for any x on that facet and given a chosen *center* for S: $s_0 \in \mathbb{R}^n$ (cf. Fig. 2.6).

As explained in the previous section, the barrier condition constraints that make S a regulation kernel only matter on the boundary of S, as the α_i functions can be chosen to arbitrarily relax the barrier conditions in the interior of S. Therefore, a regulation kernel is a set for which the sub-regulation map (2.44) is non-empty on the boundary of S.

The benefit of such a set representation is that it is easy to divide ∂S into finitely many elements where the associated active constraints set 2.26 is constant. In particular, these elements are all the faces of the polytope and the associated constraint sets correspond to their respective incident facets.

Let:

$$FT^S \triangleq [\![1, N_{ft}]\!] \,, \tag{2.66}$$

be the set of facet indices of S and:

$$F^S \triangleq [\![1, N_f]\!] \,, \tag{2.67}$$

be the set of face indices of S. Let furthermore:

$$FT_i^S \triangleq \{x \in i\text{-th facet of } S\} \,, \tag{2.68}$$

denote the i-th facet of S and:

$$F_i^S \triangleq \{x \in i\text{-th face of } S\} \,, \tag{2.69}$$

denote the i-th face of S. Let finally V_i^S denote the set of vertices of face i and VT_j^S denote the set of vertices of facet j.

Then for all face indices $i \in F^S$ and for all $x \in F_i^S$:

$$\mathrm{Act}(x) = \left\{ j \in FT^S \;\middle|\; FT_j^S \cap F_i^S \neq \emptyset \right\} , \tag{2.70}$$

which does not depend on x along that face. We will therefore denote by Act_i the set of active constraints for face i. Note that for all $i \in F^S$ and for all $j \in \mathrm{Act}_i$:

$$V_i^S \subseteq VT_j^S . \tag{2.71}$$

Finding a regulation kernel can be achieved by finding a set for which for all face indices $i \in F^S$, there exist an input $u_i \in U$ such that for all $j \in \mathrm{Act}_i$:

$$\max_{\substack{x \in F_i^S \\ a_j \in A_j^P(x) \\ b_j \in B_j^P(x)}} \left(a_j^\top u_i + b_j \right) \leq 0, \tag{2.72}$$

in the context of (2.38). In particular, for S defined by (2.65) we have that for all $x \in F_j^S$:

$$\begin{aligned} a_j\left(x,p\right)^\top &= n_j^\top g(x,p) \\ b_j\left(x,p\right) &= n_j^\top f(x,p). \end{aligned} \tag{2.73}$$

To formulate the associated optimization problem, it is necessary to choose a parameterization of S. There exist multiple ways to parameterize polytopes, but in this work we will focus on a vertex-based parameterization for reasons that will become clear later. For the rest of this work, N_v will denote the chosen number of vertices of S.

One could at first be tempted to use the cartesian coordinates of the vertices as optimization variables. However, such a parameterization not only scales poorly with the state-space dimension, but also requires the constant re-computation of a triangulation of the vertices to determine what are the faces and facets of S. To avoid such repeated costly computations, we limit ourselves to a simpler parameterization where each vertex is defined by a fixed unit vector $v_i \in \mathbb{R}^n$ and a variable radius $r_i \geq 0$ (cf. Fig. 2.6). This way, a triangulation of the vertices for $r_i = 1$ for all $i \in [\![1, N_v]\!]$ is still valid for any combination of $r_i \geq 0$. Let us now present the different steps of the algorithm.

2.4.3.1 Algorithm Inputs

The inputs to the algorithm are as follow:

- Input bounds vectors: $u_{\text{min}} \in \mathbb{R}^m$ and $u_{\text{max}} \in \mathbb{R}^m$
- Safety set state bounds vectors: $x_{\text{min}}^{\bar{S}} \in \mathbb{R}^n$ and $x_{\text{max}}^{\bar{S}} \in \mathbb{R}^n$
- Dynamics of the system and their gradients: f, g, Df, and Dg.
- Description functions of the safety set and their gradients: $h_i^{\bar{S}}$, and $Dh_i^{\bar{S}}$, $i \in [\![1, N_{\bar{S}}]\!]$.

2.4.3.2 Initialization

- Our algorithm therefore starts by generating the set of unit vectors $v_i \in \mathbb{R}^n$. There exist multiple ways to do so, the most basic one being to randomly pick vectors on the unit sphere.

- Then, upper bounds $r_{\max,i}$ are computed for the optimization variables r_i such that for all $i \in [\![1, N_v]\!]$:

$$\begin{gathered} 0 \leq r_i \leq r_{\max,i} \\ \Longrightarrow \\ x_{\min}^{\bar{S}} \leq s_0 + r_i v_i \leq x_{\max}^{\bar{S}} \end{gathered} \tag{2.74}$$

- Then, the face lattice is generated [49]. This is accomplished by first generating a triangulation of the vertices, and then enumerating the polytope faces of lower and lower dimensions recursively. Several pieces of information are precomputed at this stage for later algorithmic efficiency like the face to incident facets map and some matrices inverse to compute the normal vector of the facets for given radiuses.

- Upper bounds $r_{\max,i}$ on the radiuses r_i are also computed from $x_{\min}^{\bar{S}}$ and $x_{\max}^{\bar{S}}$ to accelerate the optimization.

- Convexity constraints are prepared. In particular, for every facet, the indices of the vertices exactly one edge away in the face lattice are determined.

- The radius optimization variables r_i are initialized to the same value: the smallest $r_{\max,i}$.

- Finally, all the vertices of the input bounds box are explicitly determined and the input optimization variables u_i are initialized so as to minimize the left side of (2.72).

2.4.3.3 Nonlinear Optimization Problem

Given the chosen setup, finding the largest regulation kernel can be formulated as the following conceptual nonlinear optimization problem:

$$\begin{aligned} \underset{\substack{r_1,\ldots,r_{N_v}\geq 0\\ u_1,\ldots,u_{N_f}\in U}}{\text{maximize}} \quad & \sum_{i=1}^{N_v} r_i^2 \\ \text{s.t. all vertices of } S \text{ are in } \bar{S} \quad & \\ S \text{ is convex} \quad & \\ (2.72) \text{ satisfied on all faces.} \quad & \end{aligned} \tag{2.75}$$

Containment Constraint. Constraining S to be a subset of S is imposed by the bounds on r_i (2.74) and by the constraints:

$$h_j^{\bar{S}}(s_0 + r_i v_i) \geq 0, \tag{2.76}$$

for all $i\, [\![1, N_s]\!]$ and $j \in [\![1, N_{\bar{s}}]\!]$. The gradient of this constraint with respect to r_i is:

$$Dh_j^{\bar{S}}(s_0 + r_i v_i) v_i. \tag{2.77}$$

Note that these constraints are sufficient to ensure that $S \subseteq \bar{S}$ only if $\bar{S}$ is convex. Otherwise, the right side of (2.76) has to be a strictly positive value carefully chosen.

Convexity Constraint. Constraining the set S to be convex can be achieved by ensuring that for all facets of S, all vertices of S be on the inner side of that facet, as any points in S is a convex combination of its vertices. However, this formulation leads to a very high number of constraints. Fortunately, it can be showed that imposing for every facet that vertices exactly one edge away from the facet be on the inner side of that facet is sufficient. More formally, S must be such that for all $i \in FT_S$ and for all j in the set of neighbouring vertices of facet i:

$$n_i^\top r_j v_j \leq 1. \tag{2.78}$$

This way, the number of constraints necessary to express convexity is drastically reduced. The gradient of this constraint with respect to r_i is $n_i^\top v_j$.

Subtangentiality Constraint. The key constraint imposing that S be a regulation kernel is (2.72). As discussed in Sec. 2.3, it is fundamental for the maps A_j and B_j to have polytopic images. Furthermore, it is important that these maps can be evaluated explicitly quickly. To that end, a tool of choice is Interval Arithmetic [66] (abbreviated IA). IA is a numerical approach that allows one to evaluate an over-approximation of the image of an interval through any function. In particular, IA tools allow to quickly compute a hyper-box containing the image of any nonlinear function. For our algorithm, we will use Affine Arithmetic (abbreviated AA), which is a variation of IA that reduces some of the conservativeness of IA at the expense of slightly longer computation times (see [22]).

One other benefit of IA is that it makes it easy to find an upper bound on the left side of (2.72). Therefore, at the cost of added conservatism, it is possible to enforce constraints (2.72) by enforcing that:

$$STC_{i,j}(r, u_i) \leq 0, \tag{2.79}$$

where:

$$STC_{i,j}(r, u_i) \geq \max_{\substack{x \in F_i^S \\ a_j \in A_j^P(x) \\ b_j \in B_j^P(x)}} \left(a_j^\top u_i + b_j\right) \tag{2.80}$$

is computed quickly using AA for all faces $i \in F_S$ and $j \in \mathrm{Act}_i$. Note that we will denote the vector of radiuses by:

$$r \triangleq [r_1, \dots, r_{N_s}]^\top. \tag{2.81}$$

In order to compute $STC_{i,j}(r, u_i)$, it is however necessary to express the sets F_i^S in terms of intervals. Indeed, the basic definition of a face i is:

$$F_i^S = \left\{ s_0 + \sum_{j \in V_i^S} \lambda_j r_j v_j \;\middle|\; \forall j \in V_i^S,\ \lambda_j \in [0,1] \right\}, \tag{2.82}$$

which is not suited for computing $STC_{i,j}(r, u_i)$ using IA or AA. Therefore, we choose to represent a face i as:

$$F_i^S = \left\{ s_0 \oplus XF^i_{|V_i^S|-1} \;\middle|\; XF^i_1 = \left\{ v_{\left(V_i^S\right)_1} \right\}, \text{ and} \right.$$
$$\left. XF^i_{j+1} = \lambda_j XF^i_j \oplus (1 - \lambda_j)\, v_{\left(V_i^S\right)_j},\ \forall \lambda_j \in [0,1] \right\}. \tag{2.83}$$

One issue with this approach though is that neither (2.72) nor (2.79) are smooth with respect to the decision variables r_i and u_i. Therefore, (2.75) is not a smooth optimization problem. This is not fundamentally an issue as there exist numerical solvers that can handle such non-smooth nonlinear problems. As an alternative, we also propose a sequential solving approach that has been observed to work well for this algorithm.

2.4.3.4 Sequential Solving

In order to solve the non-smooth problem (2.75), we propose to solve a sequence of smooth nonlinear optimization problems (NLP) whose solutions converge to the solution of (2.75). In particular, we substitute the sub-tangentially constraint (2.72) for the following smooth one where for all face indices $i \in F^S$, $j \in \text{Act}_i$:

$$STC^m_{i,j}(r, u_i) \triangleq n_j^\top f(x_i^m, p^m) + n_j^\top g(x_i^m, p^m) u_i \leq -\varepsilon_{i,j}, \tag{2.84}$$

where $p^m \in P$ is a constant value of the parameters in the parameter set P (for example the center of P), $\varepsilon_{i,j}$ is a constant value whose purpose will become clear later, and where x_i^m is the middle of face i:

$$x_i^m = s_0 + \frac{1}{|V_i^S|} \sum_{j \in V_i^S} r_j v_j. \tag{2.85}$$

Because for any $j \in FT^S$:

$$n_j = M_j^{-1} R_j, \tag{2.86}$$

where:

$$R_j = \left[1/r_{(VT_j^S)_1}, \ldots, 1/r_{(VT_j^S)_{|VT_j^S|}} \right], \tag{2.87}$$

and:

$$M_j^\top = \left[v_{(VT_j^S)_1}, \ldots, v_{(VT_j^S)_{|VT_j^S|}} \right], \tag{2.88}$$

the gradient of (2.84) can be expressed as:

$$\frac{\partial STC^m_{i,j}(r, u_i)}{\partial u_i} = n_j^\top g(x_i^m, p^m), \tag{2.89}$$

and for all $k \in VT_j^S$:

$$\frac{\partial STC^m_{i,j}(r, u_i)}{\partial r_k} = F_i^m(r, u_j)^\top \frac{\partial n_j}{\partial r_k} + n_j^\top \frac{\partial F_i^m(r, u_j)}{\partial r_k}, \tag{2.90}$$

where:

$$F_i^m(r, u_j) \triangleq f(x_i^m, p^m) + g(x_i^m, p^m) u_i. \tag{2.91}$$

Therefore, for all $k \in V_i^S$:

$$\frac{\partial F_i^m(r, u_j)}{\partial r_k} = \frac{1}{|V_i^S|} v_j^\top \left(Df(x_i^m, p^m) + Dg(x_i^m, p^m) u_i\right), \tag{2.92}$$

and for all $k \in VT_j^S$:

$$\frac{\partial n_j}{\partial r_k} = -\left(M^{-1}\right)_{*,K}^\top \frac{1}{r_k^2}, \tag{2.93}$$

where $\left(M^{-1}\right)_{*,K}$ is the K-th column of M^{-1} for K being such that $\left(VT_j^S\right)_K = k$.

It is therefore easy to solve (2.75) where (2.72) is replaced by (2.84):

$$\begin{aligned} \underset{\substack{r_1,\ldots,r_{N_v} \geq 0 \\ u_1,\ldots,u_{N_f} \in U}}{\text{maximize}} \quad & \sum_{i=1}^{N_v} r_i^2 \\ \text{s.t. all vertices of } S \text{ are in } \bar{S} & \\ S \text{ is convex} & \\ \text{(2.84) satisfied on all faces} & \end{aligned} \tag{2.94}$$

as it is now a smooth optimization problem.

Given a solution:

$$S^* \triangleq \left\{r_1^*, \ldots, r_{N_v}^*, u_1^*, \ldots, u_{N_f}^*\right\} \tag{2.95}$$

to (2.94), it is now possible to evaluate $STC_{i,j}(S^*)$, which will evidently be strictly positive for some i and j if $\varepsilon_{i,j} = 0$.

The idea is therefore to begin with $\varepsilon_{i,j}^0 = 0$, compute S_0^*, and then let:

$$\varepsilon_{i,j}^{\iota+1} = STC_{i,j}(S_\iota^*) - STC_{i,j}^m(S_\iota^*) + \frac{\epsilon}{2}, \tag{2.96}$$

where $\epsilon > 0$ is a chosen tolerance for the convergence of the algorithm. By proceeding this way recursively, $STC_{i,j}(S_\iota^*)$ will in practice converge to values $\varepsilon_{i,j}^* \in [-\epsilon, 0)$, although we do not provide any guarantees it will do so. The algorithm is considered to have successfully converged when all $\varepsilon_{i,j}^\iota \in [-\epsilon, 0)$.

Finally, note that in some cases, the convergence of this scheme has been observed to be more reliable by choosing $\varepsilon_{i,j}^0$ to be strictly positive and gradually decreasing these values over the first couple of iterations until resuming the recurrence relationship (2.96).

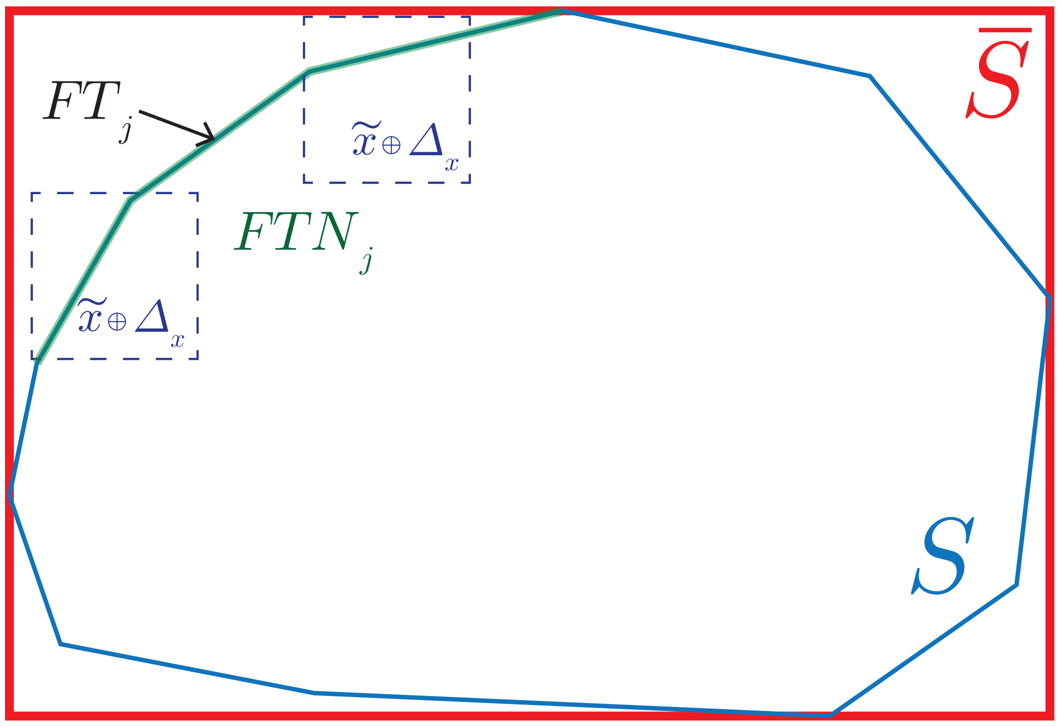

$\overline{S}$
FT_j
$\widetilde{x} \oplus \Delta_x$
FTN_j
$\widetilde{x} \oplus \Delta_x$
S

Figure 2.7: Illustration of the sets of neighboring facets to consider for the realizable regulation kernel algorithm.

2.4.3.5 Algorithm Output

The output of this algorithm is the normal vectors n_i describing a regulation kernel S as in (2.65). The interest of the safety filter formulation (2.56) becomes apparent in this case where one would want to use as many vertices as possible to describe S.

2.4.3.6 C++ Implementation

This algorithm and the next one have been implemented in C++ to produce the results presented in the following sections. It uses the CGAL library [81] to perform the random sampling of points on the unit sphere and the initial triangulation. The nonlinear solver used is IPOPT [83]. The affine arithmetic library used is Libaffa [27]. The linear algebra library used is Eigen [29].

2.4.4 Realizable Regulation Kernel Algorithm

The algorithm presented above can be extended to find realizable regulation kernels, but first we need to choose expressions of $\widetilde{A}^{P,\Delta_{x,u}}$ and $\widetilde{B}^{P,\Delta_{x,u}}$ that can be evaluated in real time for the safety filter. In this case, we will define them as follow. Let $F^S_{\Delta_x}(\tilde{x})$ be the set of facet indices intersecting the state uncertainty set for a given state measurement $\tilde{x}$:

$$F^S_{\Delta_x}(\tilde{x}) \triangleq \left\{ i \in F^S \mid (\tilde{x} \oplus \Delta_x) \cap F^S_i \neq \emptyset \right\}, \tag{2.97}$$

and:

$$\begin{aligned} \widetilde{A}^{P,\Delta_{x,u}}(\tilde{x}) = \big\{ -L_g h_j(x,p,\varepsilon_u)^\top \bigm| & \forall i \in F^S_{\Delta_x}(\tilde{x}), \\ & \forall x \in FT^S_i,\ \forall j \in \mathrm{Act}FT_i,\ \forall p \in P,\ \forall \varepsilon_u \in \Delta_u \big\}, \end{aligned} \tag{2.98}$$

$$\begin{aligned} \widetilde{B}^{P,\Delta_{x,u}}(\tilde{x}) = \big\{ -L_f h_j(x,p,\varepsilon_u) \bigm| & \forall i \in F^S_{\Delta_x}(\tilde{x}), \\ & \forall x \in FT^S_i,\ \forall j \in \mathrm{Act}FT_i,\ \forall p \in P,\ \forall \varepsilon_u \in \Delta_u \big\}, \end{aligned} \tag{2.99}$$

where:

$$\mathrm{Act}FT_i = \left\{ j \in FT^S \bigm| FT^S_i \cap FT^S_j \neq \emptyset \right\} \tag{2.100}$$

is the set of possible active constraints for states on facet i.

The main difficulty in this extension resides in the fact that enumerating all possible combinations $F^S_{\Delta_x}(\tilde{x})$ for all $\tilde{x} \in S$ is totally impractical given how big this number is even for sets with a modest number of facets. We will therefore formulate the regulation kernel algorithm in a way that is practical, at the obviously expense of added conservatism.

Let us restrict ourselves to N_{ft} sets FTN_i comprised of all facet indices $j \in FT^S$ for which there exist $\tilde{x} \in S$ such that (cf. Fig. 2.7):

$$\begin{gathered} FT^s_i \cap (\tilde{x} \oplus \Delta_x) \neq \emptyset \\ \text{and} \\ FT^s_j \cap (\tilde{x} \oplus \Delta_x) \neq \emptyset. \end{gathered} \tag{2.101}$$

Computing these sets is achieved by checking, for each facet i, the feasibility of the $N_{ft} - 1$ linear programs for all facets $j \neq i$:

$$\begin{aligned} \min_{\substack{0 \leq \lambda^1_k \leq 1 \\ 0 \leq \lambda^2_k \leq 1}} \quad & 0 \\ \text{s.t.} \quad & \sum_{k=1}^{n} \lambda^1_k = 1 \\ & \sum_{k=1}^{n} \lambda^2_k = 1 \\ & \left(\sum_{k=1}^{n} \lambda^1_k v_{\left(V^S_i\right)_k} - \sum_{k=1}^{n} \lambda^2_k v_{\left(V^S_j\right)_k} \right) \in 2\Delta_x, \end{aligned} \tag{2.102}$$

where $2\varDelta_x = \varDelta_x \oplus \varDelta_x$.

It is easy to see that for any $\tilde{x} \in S$ such that $F^S_{\varDelta_x}(\tilde{x}) \neq \emptyset$, there exist $i \in FTN^S$ such that $F^S_{\varDelta_x}(\tilde{x}) \subseteq FTN_i$. We can therefore find a regulation kernel for a realizable sub-regulation map defined by (2.98) by imposing that for $i \in FT^S$, there exist an input $u_i \in U$ such that for all $j \in FTN_i$ and for all $k \in \mathrm{Act}FT_j$:

$$\max_{\substack{x \in FT^S_j \\ a_k \in A^{P,\varDelta_u}_k(x) \\ b_k \in B^{P,\varDelta_u}_k(x)}} \left(a_k^\top u_i + b_k\right) \leq 0, \tag{2.103}$$

where:

$$\begin{aligned} A^{P,\varDelta_u}_k(x) &= \left\{ -L_g h_k(x,p,\varepsilon_u)^\top \,\middle|\, \forall p \in P,\ \forall \varepsilon_u \in \varDelta_u \right\} \\ B^{P,\varDelta_u}_k(x) &= \left\{ -L_f h_k(x,p,\varepsilon_u) \,\middle|\, \forall p \in P,\ \forall \varepsilon_u \in \varDelta_u \right\}. \end{aligned} \tag{2.104}$$

Therefore, it is possible to use the same approach as above, and by replacing (2.72) with (2.103), compute a realizable regulation kernel. The derivation of $STC^m_{i,j,k}(r, u_i)$ is in this case identical to (2.84) and below, only with different sets of facets and constraints to consider.

An important point to address in this realizable version of the regulation kernel algorithm is that the FTN_i sets depend on the values of the radiuses r_i defining S. It is therefore impossible to precompute the FTN_i sets as they change during the optimization process and so the constraint bound update law (2.96) cannot be implemented as is. To address this issue, the sequential solving approach of Algo. 1 is proposed.

Note finally that by considering $\varDelta_x = \{0\}$, this version of the algorithm can be used to reduce the computation time for computing a simple regulation kernel (i.e. not realizable) at the obvious expense of added conservatism.

2.5 Simulation of an Inverted Pendulum

2.5.1 Problem Setup

We will now apply this framework to a simple example of nonlinear inverted pendulum. This system is defined by the state $x = \left[\theta, \dot{\theta}\right]^\top$ and the dynamics:

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \sin(\theta) + p \cdot u \end{bmatrix}, \tag{2.105}$$

with a saturated input $u \in [u_{min}, u_{max}]$ with $u_{min} = -1.5$ and $u_{max} = 1.5$. Here, $p \in P$ is a constant parameter. This pendulum is upright at an unstable

Algorithm 1: Sequential NLP solving

```
Data: $\epsilon$ and $\iota_{\max}$
Result: Radiuses $r_i^*$ of Realizable Regulation Kernel $S^*$
1  Initialize $r_i$, $u_i$ and $FTN_i$;
2  $\iota \leftarrow 0$;
3  while true do
4      $\varepsilon_{i,j,k} \leftarrow 0$;
5      while true do
6          $r_i$, $u_i \leftarrow$ NLP (2.94) solution;
7          Compute $STC_{i,j,k}$ from $r_i$ and $u_i$;
8          Update $\varepsilon_{i,j,k}$ as in (2.96);
9          if $-\epsilon \leq max(\varepsilon_{i,j,k}) \leq 0$ then
10             Determine new $FTN_i'$ from new $r_i$;
11             if $\forall i,\ FTN_i' \neq FTN_i$ then
12                 Compute $STC_{i,j,k}$ from $r_i$ and $u_i$;
13                 if $-\epsilon \leq max(\varepsilon_{i,j,k}) \leq 0$ then
14                     Return $r_i$;
15                 end
16                 $FTN_i \leftarrow FTN_i'$;
17             end
18             go to 21;
19         end
20     end
21     $\iota \leftarrow \iota + 1$;
22     if $\iota > \iota_{max}$ then
23         Return Failure;
24     end
25 end
```

equilibrium when $\theta = 0$. The safety set $\overline{S}$ is chosen to be a box centered at the origin and of edge size 2π (cf. Fig. 2.8).

In the following simulations, the safety filters are implemented in C++ using the OSQP solver [79] and run on an Intel i7-6820HQ processor.

2.5.2 Exact Model Results

For this example, we first assume an exact model: $P = [1.0, 1.0]$. The algorithm is initialized as discussed in Sec. 2.4.3.2 with a set S_0 centered at the origin and made up of 150 vertices uniformly spread over the unit circle. The result of this initialization phase is showed in Fig. 2.8. As expected, not all $STC_{i,j}(S_0)$ are negative at this stage. The sequential optimization approach

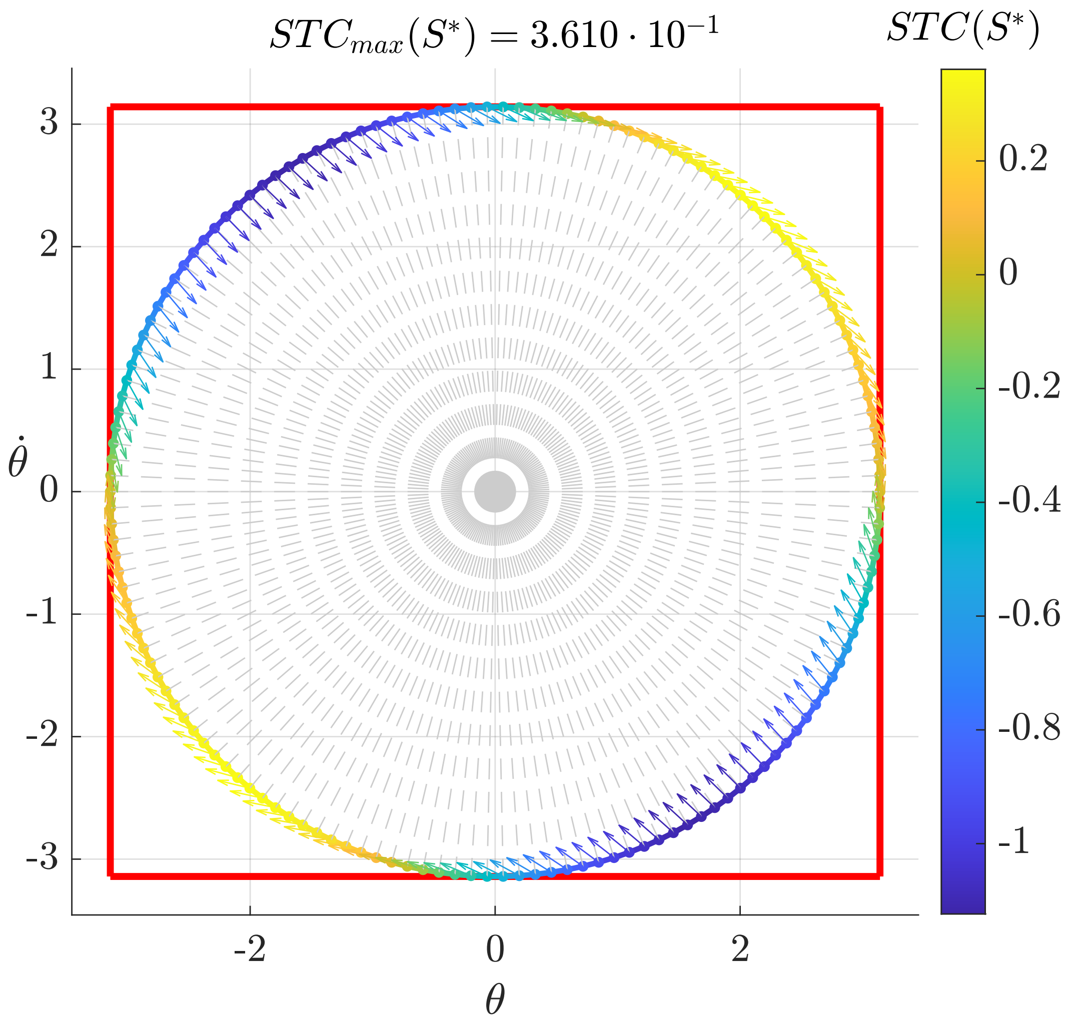


Figure 2.8: Set S for an exact model (2.105) after the initialization phase. The safety set is in red. The facets and vertices of S are displayed with colors corresponding to the maximum value of the subtangentiality condition for that face. The arrows correspond to the directions of the dynamics at the vertices for the initial inputs u_i.

discussed in 2.4.3.4 is performed with $\epsilon = 10^{-6}$ and the result of this process after convergence can be seen in Fig. 2.9. As expected, all $STC_{i,j}(S*)$ are sticky between 0 and $-\epsilon$, and the resulting set is close to the largest possible convex set contained inside the viability kernel for this setup. Note that for this particular example, the algorithm takes on average $2s$ and 9 sequential NLP solving steps to converge.

Given this control invariant set, relaxed safety filters (2.32) are implemented with a single relaxation term α_{des} and different values of α_{des}. The relaxation

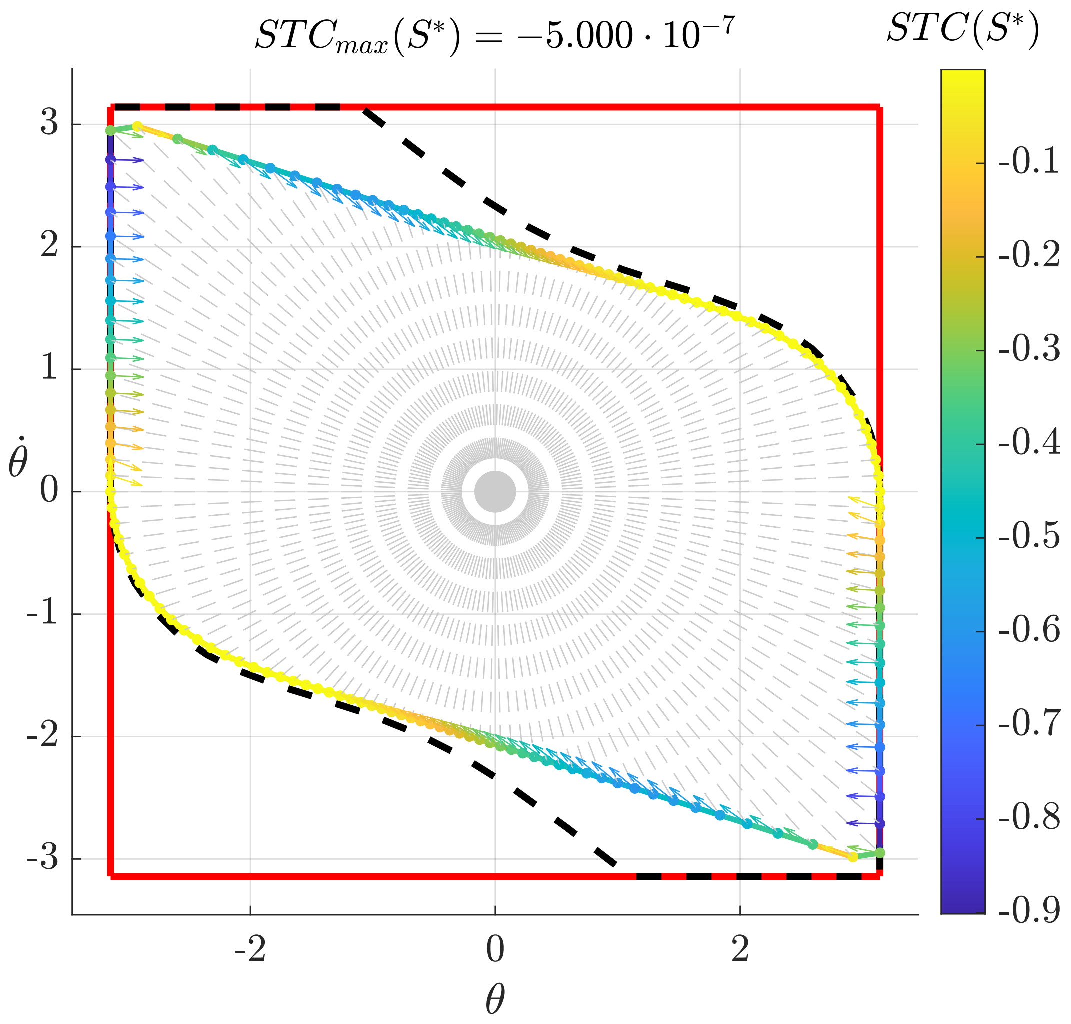


Figure 2.9: Regulation kernel for an exact model (2.105) after convergence of the proposed algorithm. The safety set is in red. The viability kernel is the black dashed line. The facets and vertices of S are displayed with colors corresponding to the maximum value of the subtangentiality condition for that face. The arrows correspond to the directions of the dynamics at the vertices for the optimal inputs u_i^*.

penalty factor is chosen to be $M = 1$. The pendulum is initialized at $x(t_0) = [0.5, 0]^\top$ and left to fall freely: $u_{des} = 0$. The resulting system trajectories are shown in Fig. 2.10. As expected, the system remains inside the safety set, and the smaller α_{des} is, the more invasive the resulting filtering is. It can also be seen that α is also relaxed at times because of the small chosen value of M. Higher values of M would result in less relaxation, but still guarantee feasibility of the filter inside the regulation kernel.

2.5.3 Inexact Model Results

In a second phase, the uncertainty associated with p is increased and the resulting regulation kernels can be seen in Fig. 2.11 for various sizes of that uncertainty. As expected, the higher the uncertainty is, the smaller the resulting kernel becomes. Note also that increasing the uncertainty does not have any perceptible effect on the convergence time of the algorithm.

Given the regulation kernel associated with $P = 1.0 \pm 0.2$, a relaxed robust safety filter (2.47) is implemented with a single relaxation term $\alpha_{des} = 5$. The relaxation penalty factor is chosen to be $M = 50$. The pendulum is initialized at $x(t_0) = [0.5, 0]^\top$ and left to fall freely: $u_{des} = 0$. The resulting system trajectories are shown in Fig. 2.12 for different actual values of p. As expected, the system remains inside the safety set for all values $p \in [0.8, 1.2]$.

Then, given this same regulation kernel associated with $P = 1.0 \pm 0.2$, relaxed robust safety filters (2.47) are implemented with different uncertainty associated with p, while the actual value of the parameter is set to $p = 1.0$. Again, the relaxation term is chosen to be $\alpha_{des} = 5$, the relaxation penalty factor is chosen to be $M = 50$, and the pendulum is initialized at $x(t_0) = [0.5, 0]^\top$ and left to fall freely: $u_{des} = 0$. The resulting system trajectories are shown in Fig. 2.13. As expected, the regulation kernel associated with $P = 1.0 \pm 0.2$ is also a regulation kernel for smaller size of uncertainty associated with P. The system therefore remains inside the safety set for all the different values of the uncertainty associated with p while being more conservative when this uncertainty set is large.

Finally, given this same regulation kernel associated with $P = 1.0 \pm 0.2$, scalable relaxed robust safety filters (cf. Sec. 2.3.5) are implemented with different values of $\widehat{N_s} \geq 2$ as in this 2D case, the maximum number of active constraint is $\widehat{N_{s,\max}} = 2$. Again, the parameter is set to $p = 1.0$, $\alpha_{des} = 5$, $M = 50$, and the pendulum is initialized at $x(t_0) = [0.5, 0]^\top$ and left to fall freely: $u_{des} = 0$. The resulting system trajectories are shown in Fig. 2.14. As expected, the system remains inside the safety set for all the different values of $\widehat{N_s}$, and larger values of $\widehat{N_s}$ yield smoother input filtering, but at the expense of computational time.

2.5.4 Realizable Results

Finally, the effect of state uncertainty on safety filtering is showcased. In particular, state uncertainty $\mathit{\Delta}_x$ is introduced and the resulting realizable regulation kernels for $P = 1.0 \pm 0.1$ can be seen in Fig. 2.15 for various sizes of that uncertainty. As expected, the higher the uncertainty is, the smaller the resulting kernel becomes. Note also that this time, increasing the uncertainty **does** have a noticeable effect on the overall convergence time of the algorithm as the higher the uncertainty is, the more constraints the successive NLPs have.

As discussed previously, the realizable kernel algorithm 2.4.4 can be used to more quickly compute simple regulation kernels at the expense of added conservatism. This is showcased in Fig. 2.15 where the black set computed with the simple algorithm 2.4.3 is larger than the blue set computed with the realizable algorithm 2.4.4 for a state uncertainty set $\mathit{\Delta} = 0$.

An example of realizable safety filter using these computed sets can be seen in Fig. 2.16. In this case, the inverted pendulum is controlled by a realizable safety filter based on an inexact measurement of the state. In particular, the measurement $\tilde{x}(t)$ is smooth, but different from $x(t)$ by at most ± 0.2. As can be seen in these graphs, a realizable formulation of the safety filter is effective at keeping the system safe inside the regulation kernel.

In order to evaluate the $\widetilde{A}^{P,\mathit{\Delta}_x,u}$ and $\widetilde{B}^{P,\mathit{\Delta}_x,u}$ maps online, one need to be able to evaluate $F^S_{\mathit{\Delta}_x}(\tilde{x})$. This is done in 2 steps. First, facets that potentially constitute $F^S_{\mathit{\Delta}_x}(\tilde{x})$ are determined by checking for intersection between $\tilde{x} + \mathit{\Delta}_x$ and the bounding boxes of all the facets of S. Then, the actual $F^S_{\mathit{\Delta}_x}(\tilde{x})$ set is determined by checking the feasibility of the following LP for all of these pre-selected facets i:

$$\begin{aligned} \min_{0 \le \lambda_k \le 1} \quad & 0 \\ \text{s.t.} \quad & \sum_{k=1}^{n} \lambda_k = 1 \\ & \sum_{k=1}^{n} \lambda_k v_{\left(V_i^S\right)_k} \in \tilde{x} + \mathit{\Delta}_x. \end{aligned} \tag{2.106}$$

The importance of the realizability of the filter is highlighted in Fig. 2.17 where trajectories of the inverted pendulum presented in the previous section can be observed for a robust safety filter and a realizable safety filter working with a realizable safety kernel. In this scenario, the control loop rate is $\Delta_t = 0.1$s and as expected, a simple robust safety filter fails in preventing the system from leaving the regulation kernel. On the other hand, a realizable safety kernel is effective at keeping the system safe despite this low control loop rate.

The behavior of realizable safety filters with different control loop rates can be seen in Fig. 2.18 for a given realizable regulation kernel.

2.6 Hardware Implementation

2.6.1 Hardware Setup

The Segway platform used here began as a Ninebot E+. All of the electronics, including sensors and motor controllers, were removed, as well as the steering column. The steering column and human rider were replaced by a steel column with adjustable weights and height at its top. The original motor controller was replaced by a set of Elmo Gold Solo Twitters, which provide direct current control of the motors (cf. Fig. 3.10). The onboard state estimation is performed using wheel encoders and a VectorNav VN-100 IMU.

A Teensy 3.5 reads and processes the sensor data and sends the state information to the main computer onboard, a Jetson TX2, which computes the control action that is sent to the motor controllers. The TX2 runs Ubuntu 18.04 LTS and the ERIKA3 real-time operating system concurrently through the Jailhouse hypervisor. The Linux OS runs ROS, which allows external communication and logging of all of the necessary data. The real-time operating system handles the communication with the Teensy and the computation of the state observer and control actions. These two operating systems are able to share information through a shared memory interface. The safety filters implementation is the same as the one used for the simulations of the inverted pendulum.

2.6.2 System Identification

In order to translate these simulation results to hardware, confidence in the validity of the model is fundamental as discussed presently. To that end, we developed two test beds to accurately identify the inertial parameters of the segway vehicle (cf. Fig. 2.20). The first test bed (on the right in Fig. 2.20) is able to measure the position of the center of mass of any object in a given plane by using 3 load-cells whose positions are accurately known. The second test bed (on the left in Fig. 2.20) is able to measure the inertia of any object along a given axis by measuring the oscillations of that object when attached to a rotational spring. Then, by combining several such measurements for various orientations of the object on the test beds, it is possible to estimate all the inertial properties of this object (mass, position of the center of mass, inertia matrix) with a high level of accuracy.

This process was therefore carried out for the segway vehicle in order to derive the models that will be used in the rest of this work (cf. Fig. 2.21 and Fig. 2.22).

2.6.3 Model and Formulation

For this demonstration, the Segway is fitted with a 3rd wheel and only controlled longitudinally so that it can be modeled as a double integrator with viscous friction. The state is therefore defined by longitudinal position and velocity of the vehicle $x = [p, v]$ with input $u \in [-20, 20]$ being the total motor current. The identified model is:

$$\dot{x} = \begin{bmatrix} v \\ \frac{k}{m \cdot R} u - \frac{f}{m} \cdot v \end{bmatrix}, \tag{2.107}$$

where $m \in 73 \pm 1$kg is the equivalent mass of the vehicle (accounting for wheel inertia), $R \in 0.2 \pm 0.01$m is the wheel radius, $k \in 1.2 \pm 0.1$Nm/A the wheel level torque constant, and $f \in 23 \pm 2$N/(m/s) the global friction coefficient.

The safety set is defined as $p \in [-3, 3]$m and $v \in [-3, 3]$m/s.

2.6.4 Robustness to Mass Variability

First, the effectiveness of the robust approach is demonstrated by computing a regulation kernel assuming the identified measurements and bounds $m \in [70, 75]$kg and running a scalable relaxed robust safety filter (2.47) assuming these same bounds. Two runs are performed, one with the vehicle empty, and one with the vehicle loaded with a 57kg weight (cf. Fig. 2.19). For each run, the vehicle is started at the origin of the state space and the desired input is set to a constant $u_{\text{des}} = 20$A. The results of these experiments are reported in Fig. 2.23. As expected, when the vehicle is empty and the identified model is (most likely) valid, the vehicle remains inside the regulation kernel and therefore the safety set. However, when the model is not valid, the system cannot be kept inside the regulation kernel and safety set. Note that when the safety filter becomes infeasible in this latter case, the input is set to $u = -20 \cdot \tanh(10v)$.

When a safety kernel is computed assuming $m \in [70, 130]$kg, the safety filter is effective at keeping the system safe with or without additional mass as reported in Fig. 2.24. For the case with no additional mass, two safety filters assuming respectively 75kg and 135kg maximum vehicle mass are tested on the $m \in [70, 130]$kg safety kernel, and as expected, both perform well with minimal performance difference. Note that for all these experiments, the safety filter is ran at 800Hz.

2.6.5 Robustness to Controller Frequency

Now, the effectiveness of the proposed framework to address state uncertainty is showcased with the same experimental setup and scenario as previously. In particular, the capability of this framework to explicitly account for time sampled sensing and control is highlighted by altering the frequency at which the safety filter is ran.

At first, a realizable safety kernel is computed for the identified model (2.107) with $\varDelta_t = 0.01$s which correspond to $\varDelta_x^+ = \{\pm 0.031, \pm 0.028\}^\top$. Scalable realizable safety filters (2.52) are implemented with a variable loop rate satisfying $\varDelta_t < 0.01$s, and with a fixed loop rate $\varDelta_t = 0.1$ in its relaxed and non-relaxed version.

As can be seen in Fig. 2.25, the system is only kept inside the regulation kernel when $\varDelta_t < 0.01$s, as anticipated. The result of the same experiments but for a realizable safety kernel corresponding to $\varDelta_t = 0.1$s can be seen in Fig. 2.26. As expected, all the trajectories remain safe in this case.

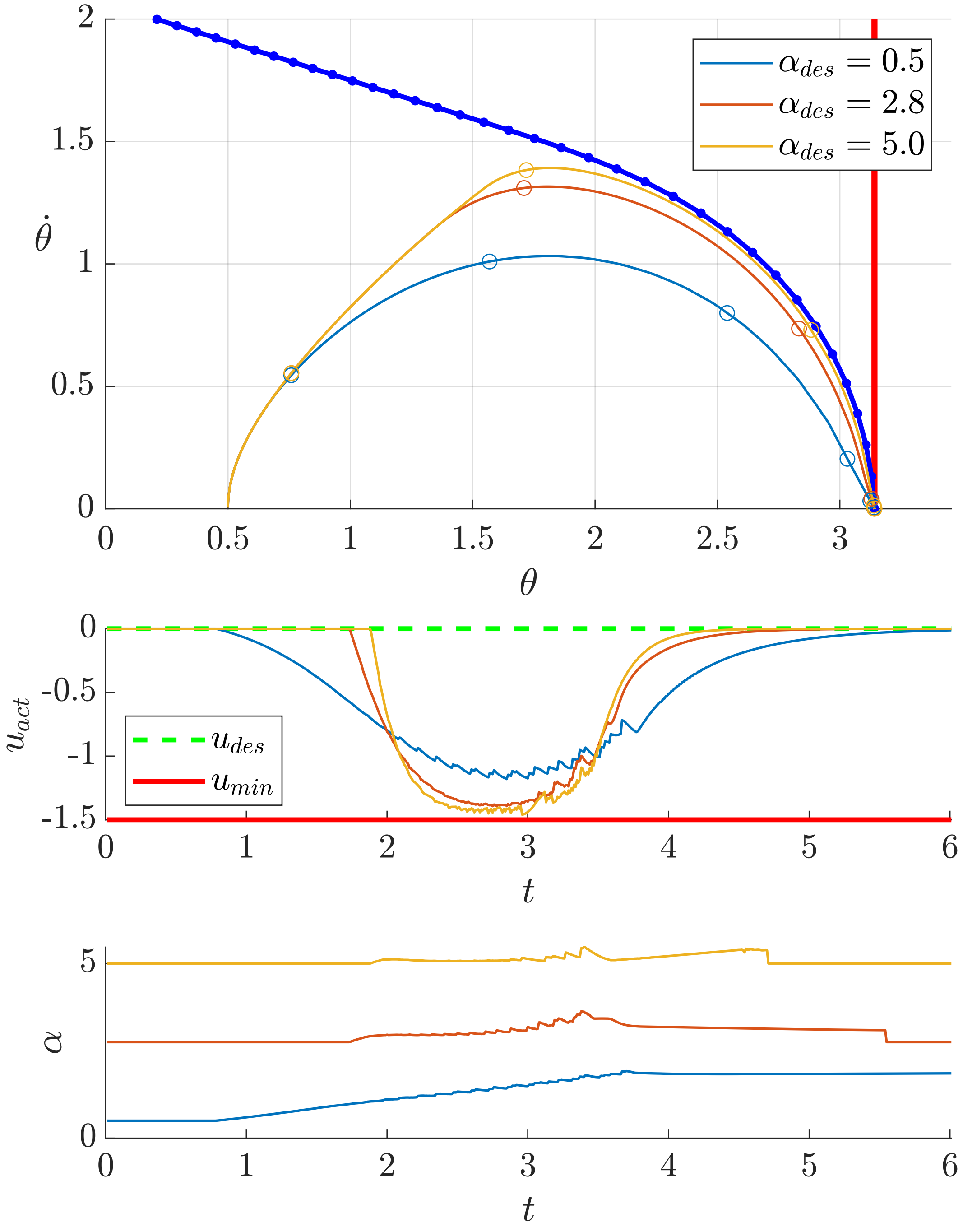


Figure 2.10: Relaxed safety filter for an exact model (2.105) with $M = 1$ and various values of desired relaxation factor α_{des}. The circular markers along the trajectories are spaced 1s apart. The safety set is in red and the regulation kernel in blue.

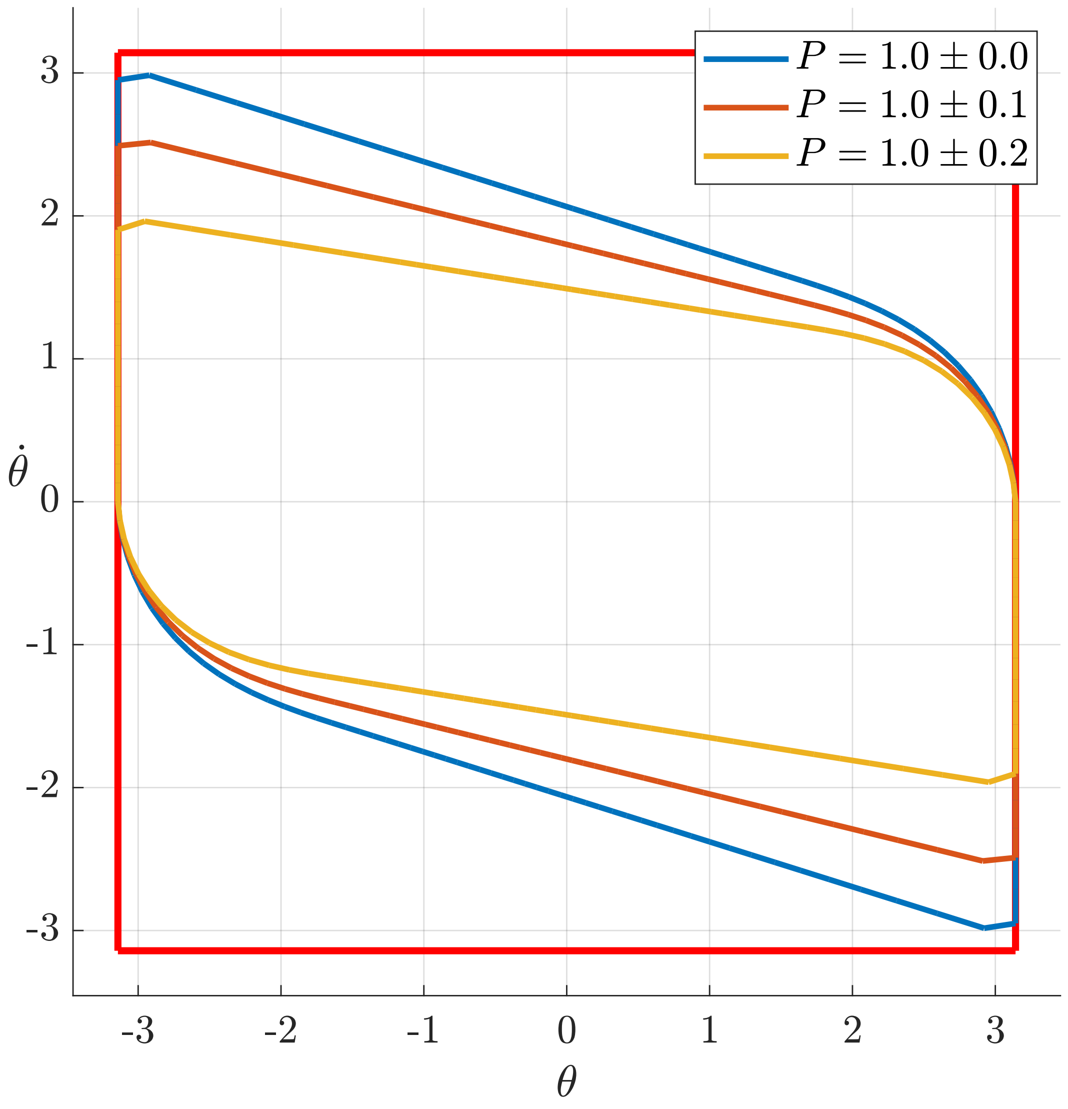


Figure 2.11: Regulation kernels for various inexact models (2.105) after convergence of the proposed algorithm. The safety set is in red.

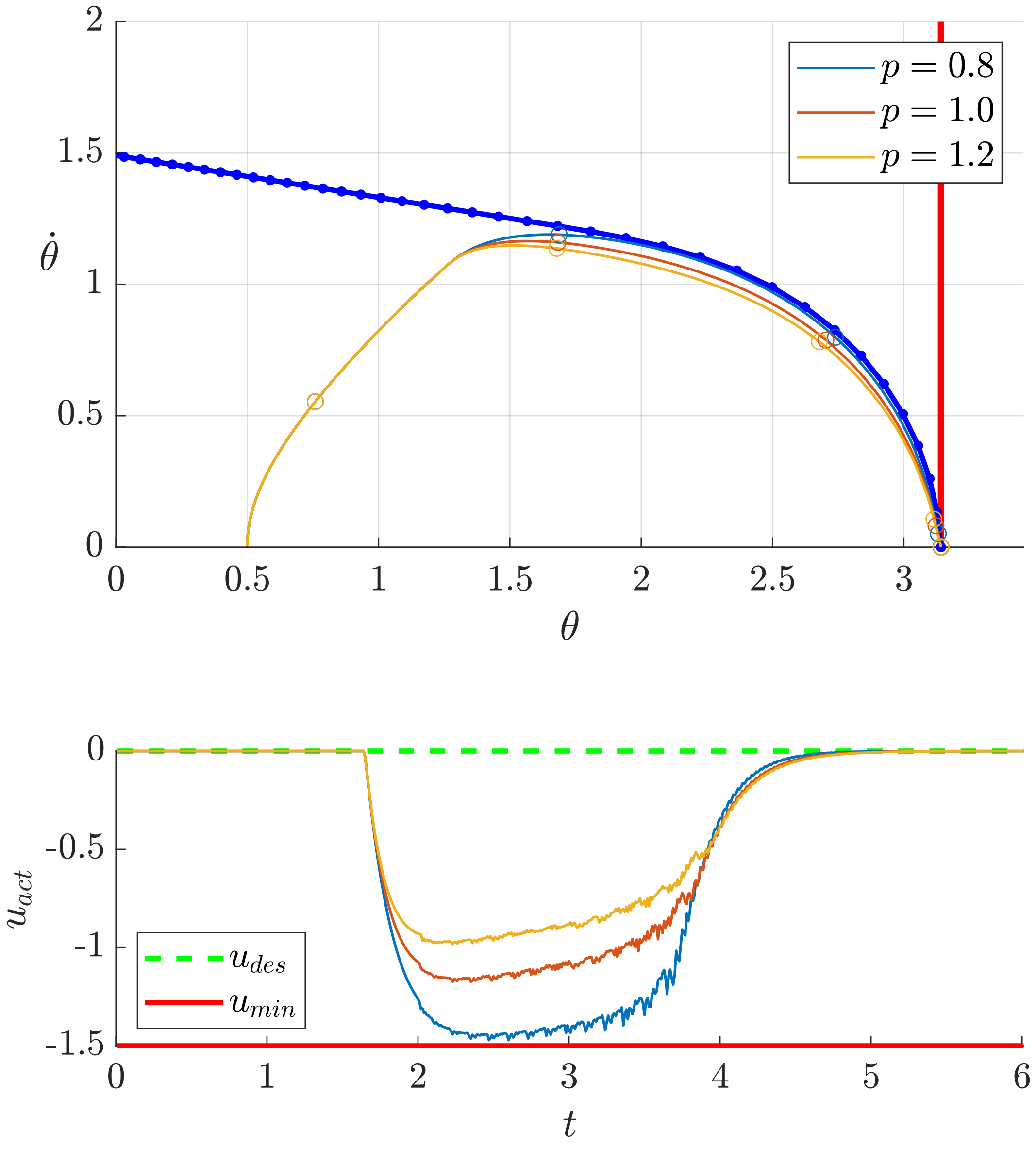


Figure 2.12: Relaxed robust safety filter for an inexact model (2.105) assuming $P = 1.0 \pm 0.2$, and with $M = 50$ and $\alpha_{des} = 5$. The regulation kernel in blue corresponds to $P = 1.0 \pm 0.2$. The different trajectories correspond to various actual values of the parameter p. The circular markers along the trajectories are spaced 1s apart. The safety set is in red.

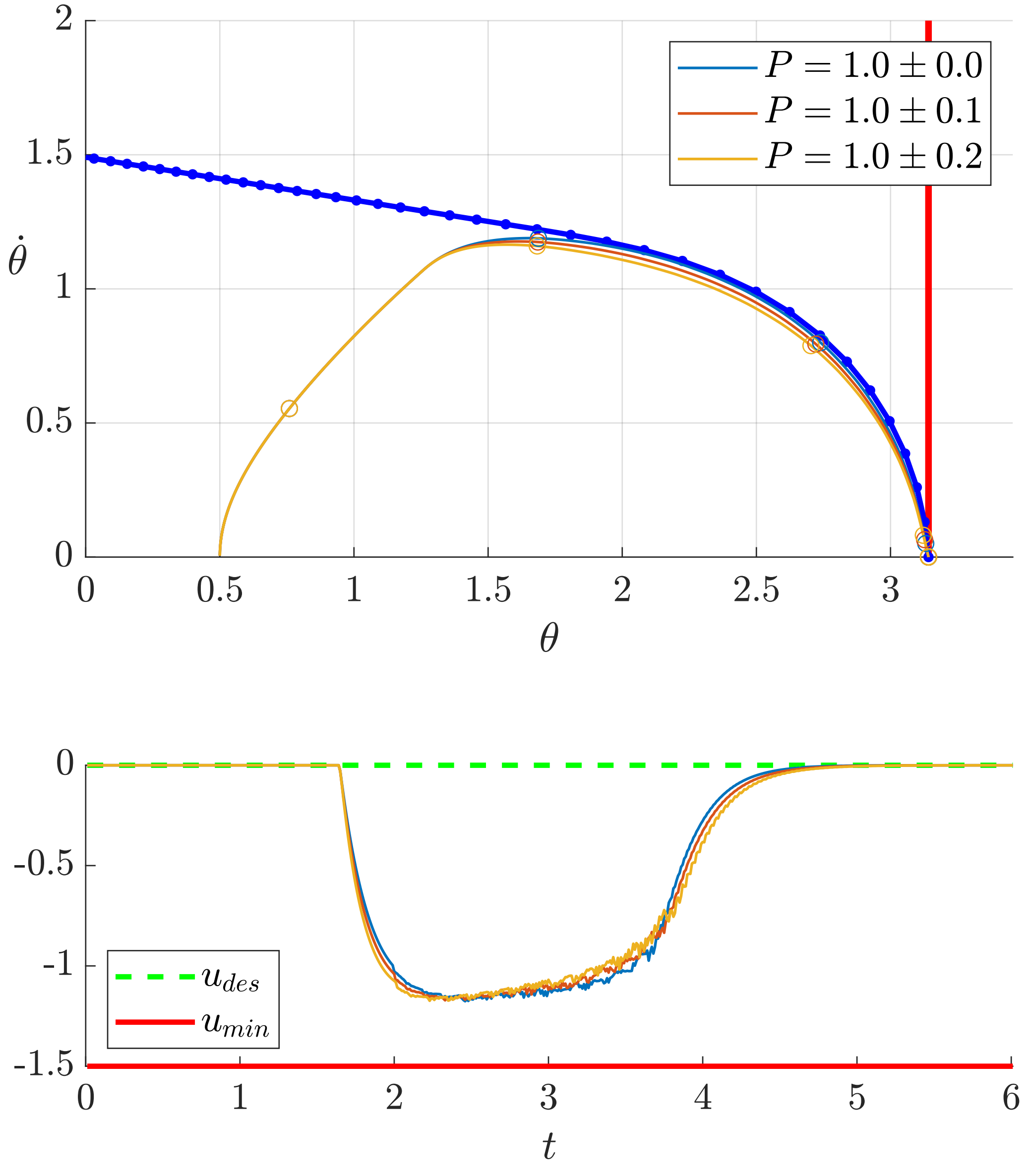


Figure 2.13: Relaxed robust safety filters for an inexact model (2.105) assuming different values of P, and with $M = 50$ and $\alpha_{des} = 5$. The regulation kernel in blue corresponds to $P = 1.0 \pm 0.2$. The actual value of the parameter is $p = 1.0$. The circular markers along the trajectories are spaced 1s apart. The safety set is in red.

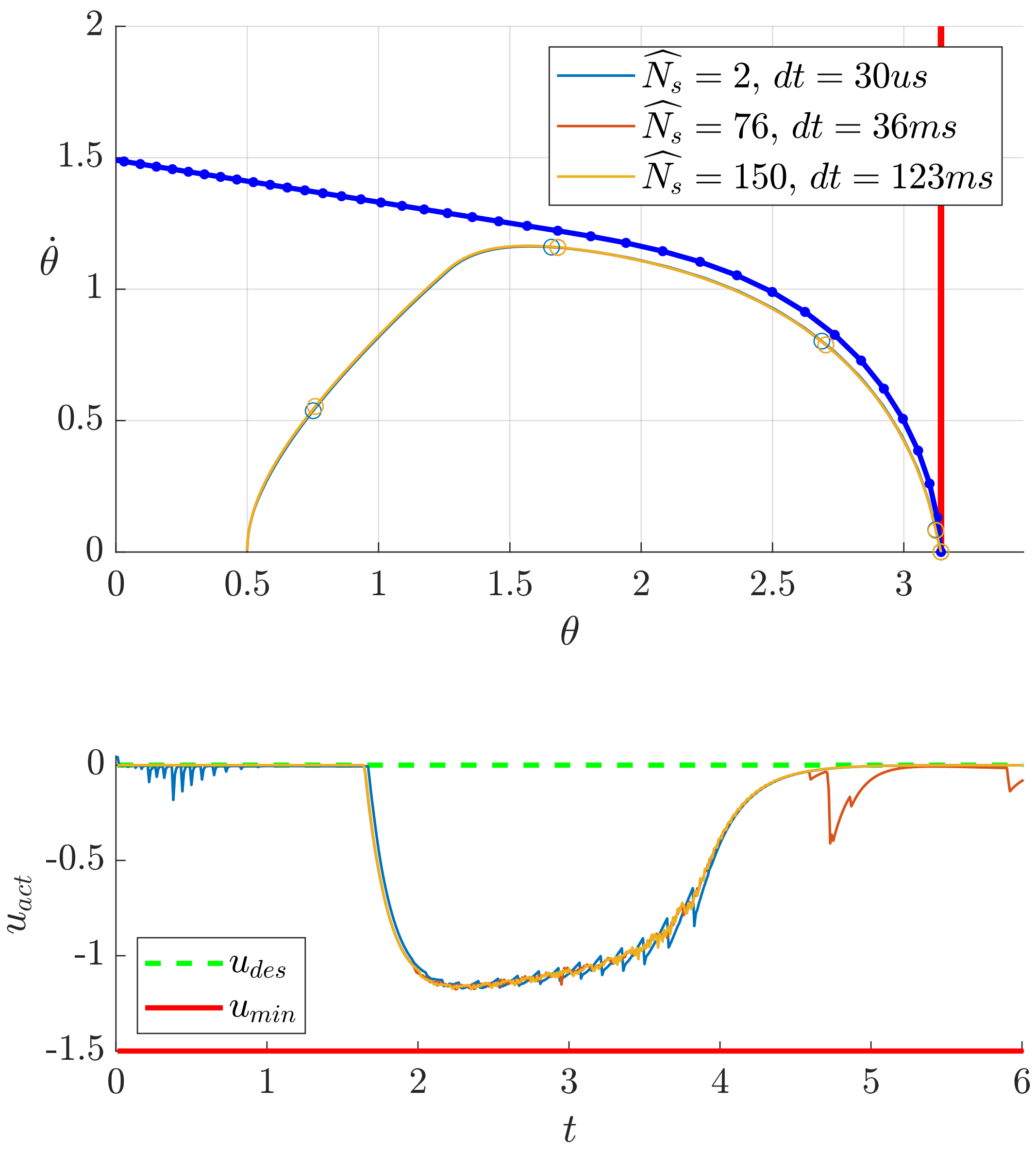


Figure 2.14: Scalable relaxed robust safety filter for an inexact model (2.105) assuming $P = 1.0 \pm 0.2$, and with $M = 50$, $\alpha_{des} = 5$ and $p = 1$. The regulation kernel in blue corresponds to $P = 1.0 \pm 0.2$. The different trajectories correspond to various values of $\widehat{N_s}$. The reported dt correspond to the average computation time for the safety filters on an Intel i7-6820HQ. The circular markers along the trajectories are spaced 1s apart. The safety set is in red.

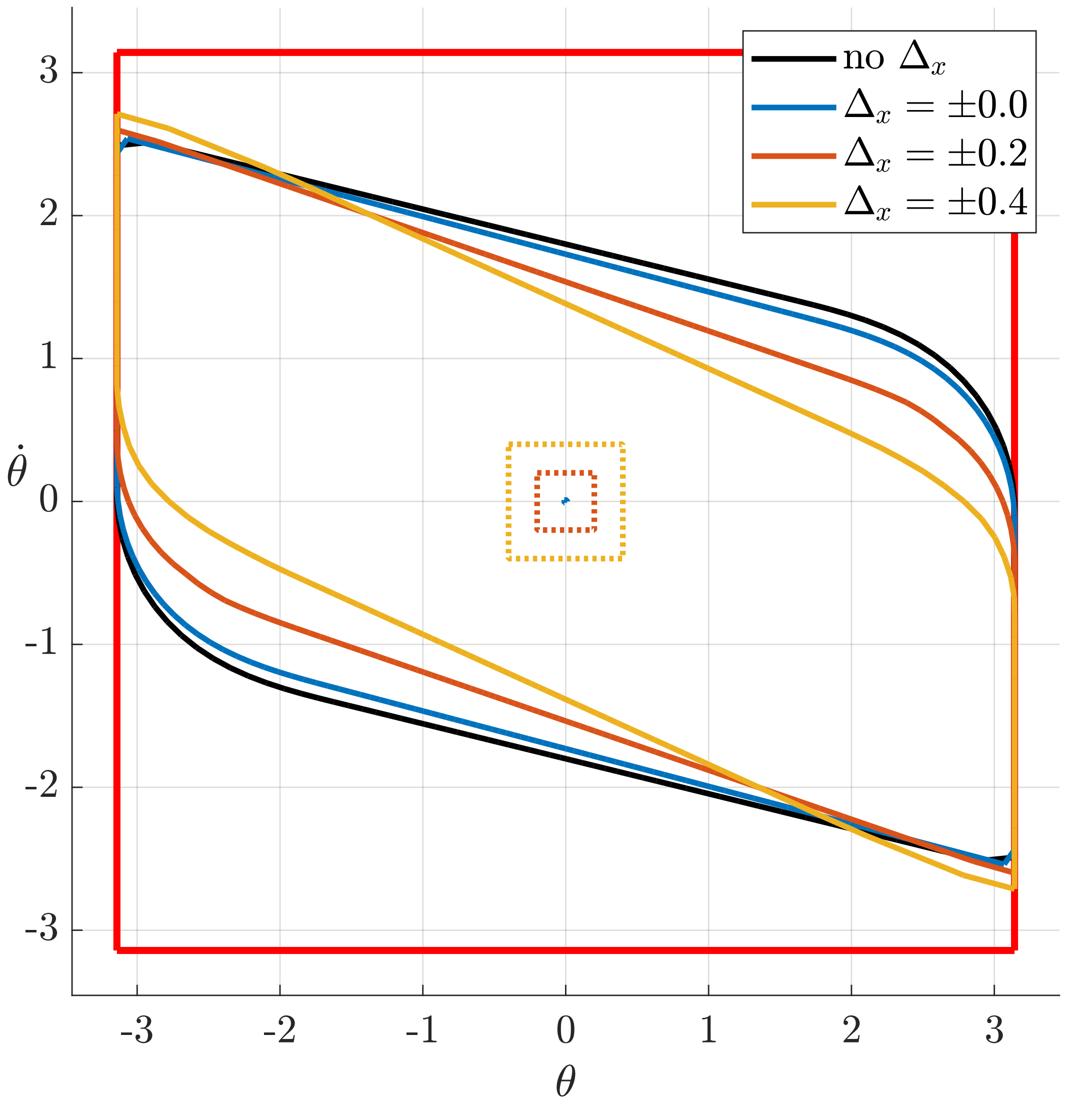


Figure 2.15: Realizable regulation kernels of inexact models (2.105) with $P = 1.0 \pm 0.1$ and various value of state uncertainty Δ_x. The safety set is in red. In black is the regulation kernels computed with the regulation kernel algorithm 2.4.3. The dotted squares in the middle correspond to the size of the state uncertainty set Δ_x

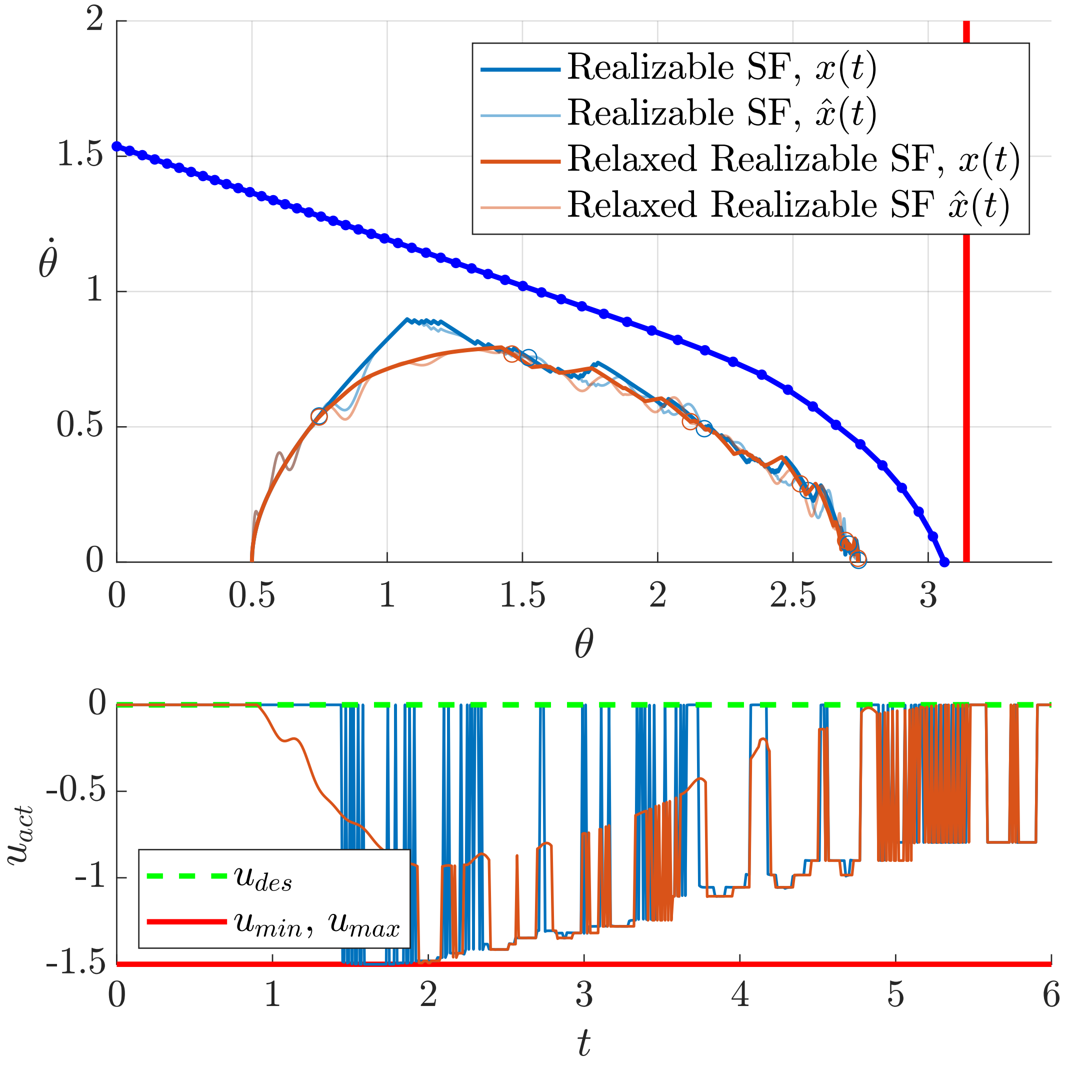


Figure 2.16: Realizable safety filter and scalable relaxed realizable safety filter with $varDelta_x = \pm 0.2$, $\widehat{N_s} = 5$, $M = 50$ and $\alpha_{des} = 1$.

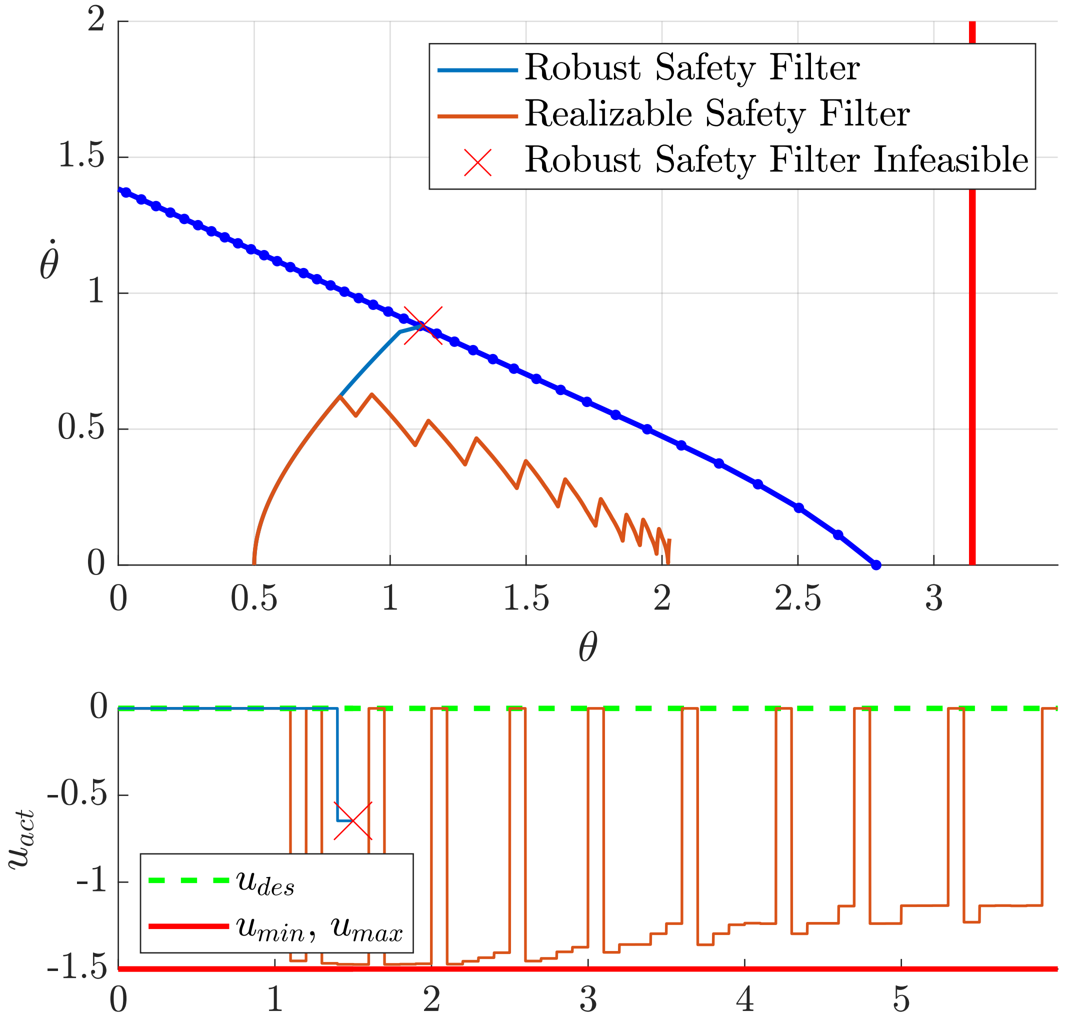


Figure 2.17: Realizable vs Robust-only safety filters.

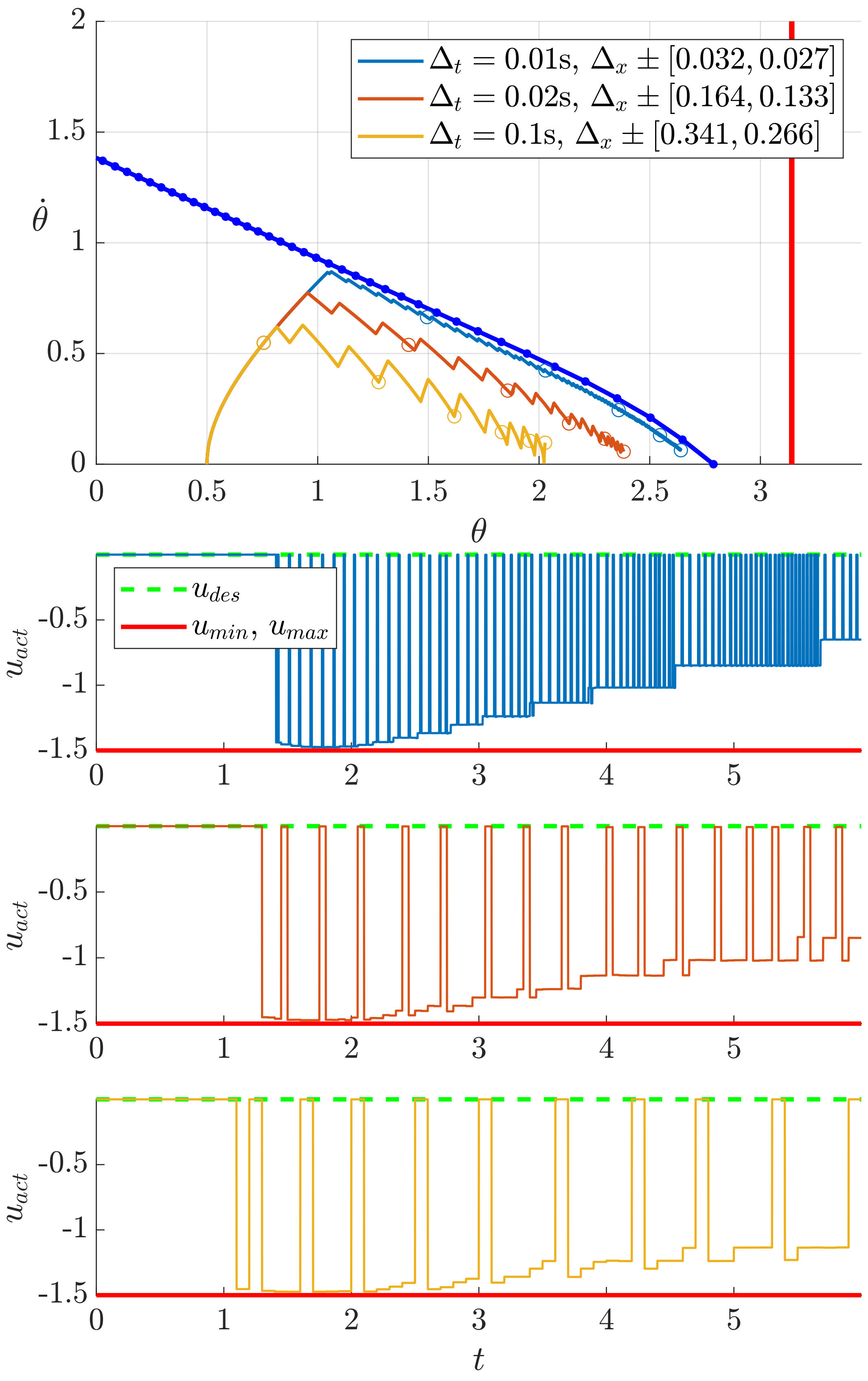


Figure 2.18: Trajectories of an inverted pendulum under various realizable safety filters and with various sampling frequencies.

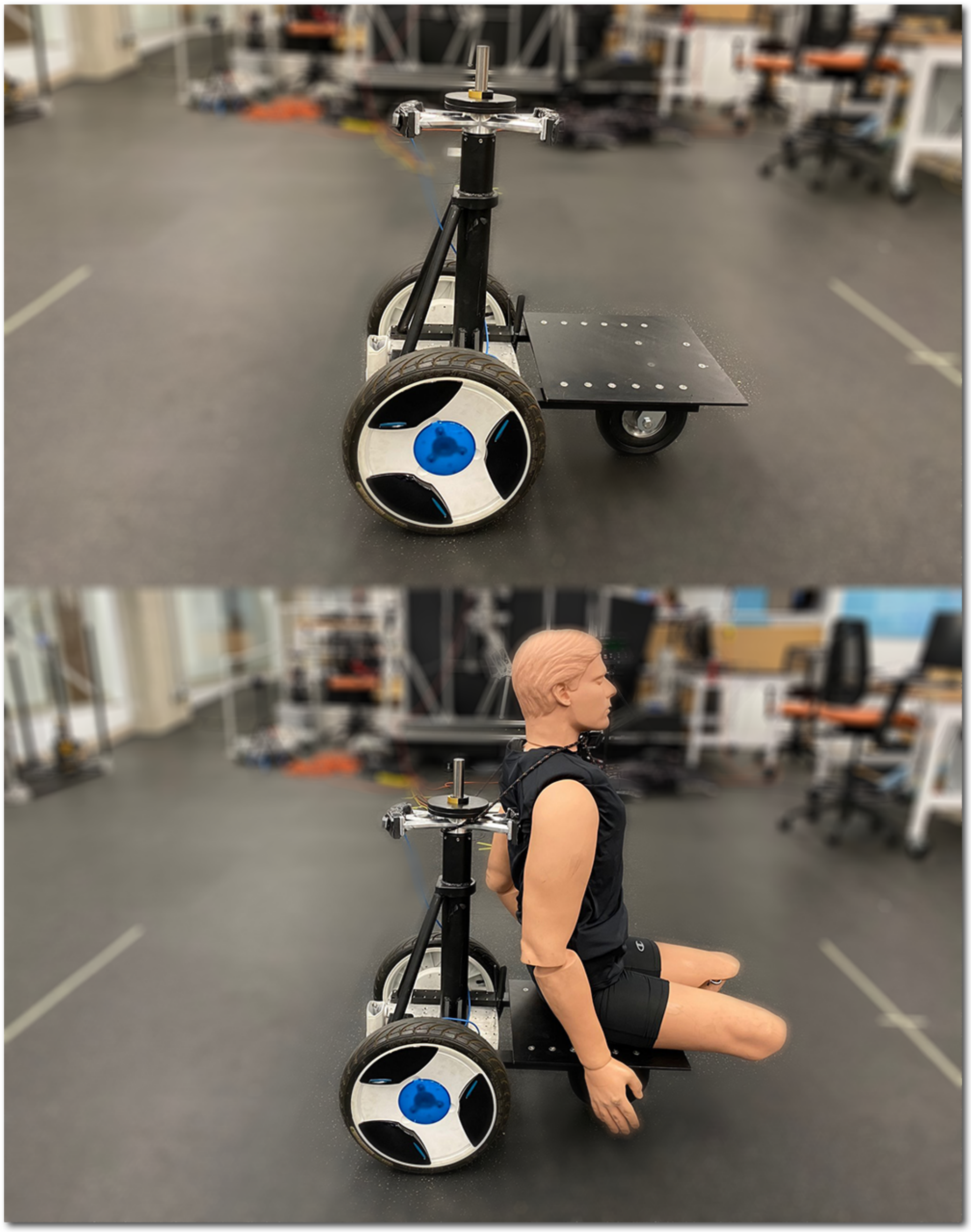


Figure 2.19: Picture of the segway vehicle used for implementation.

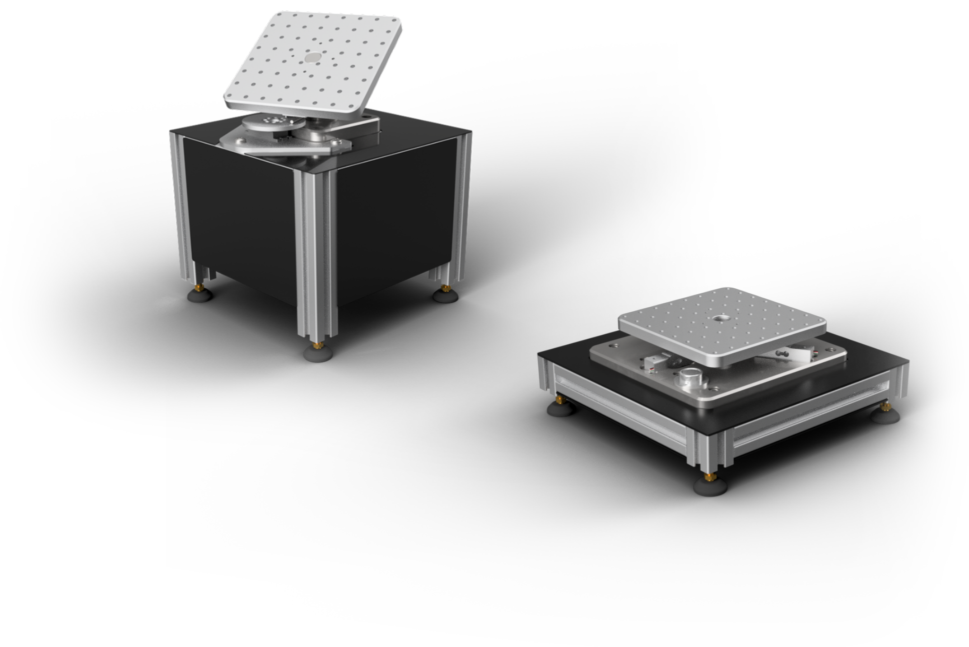


Figure 2.20: CAD models of the inertial measurement test beds.

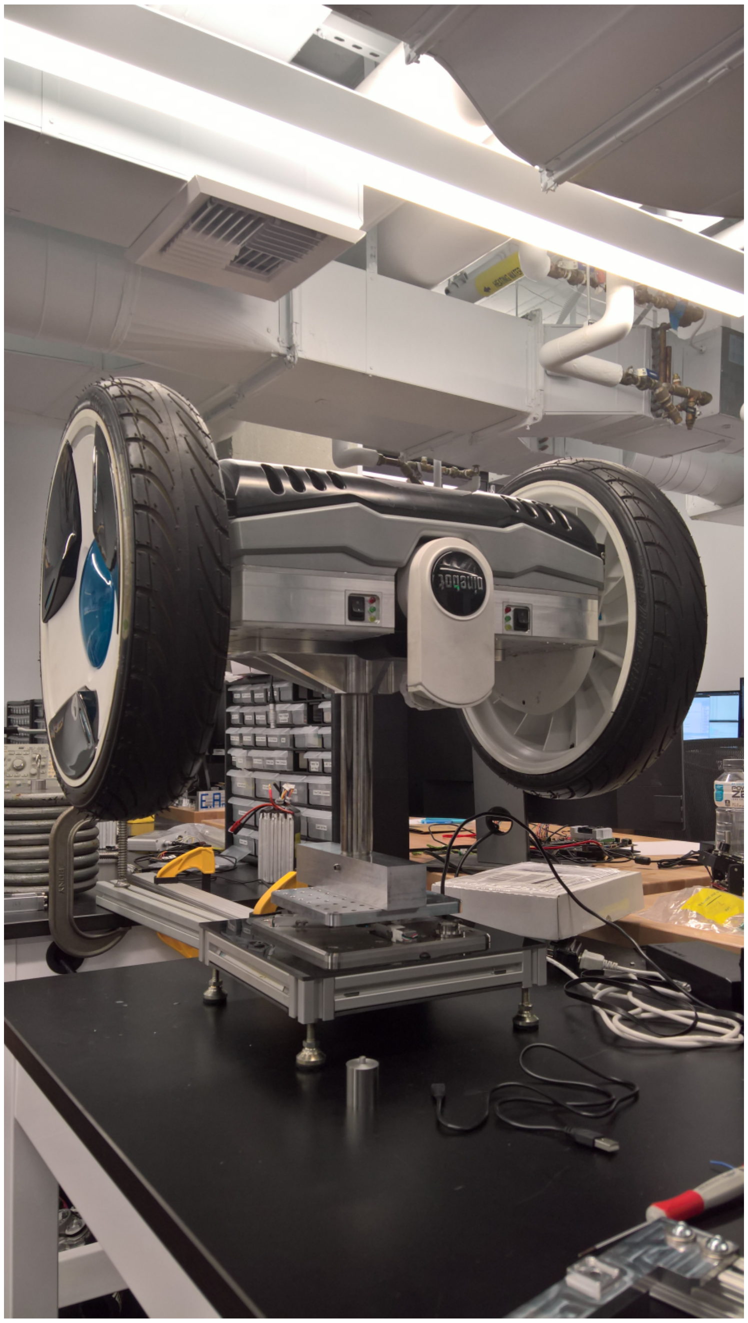


Figure 2.21: Picture of the segway vehicle on the center of mass test bed.

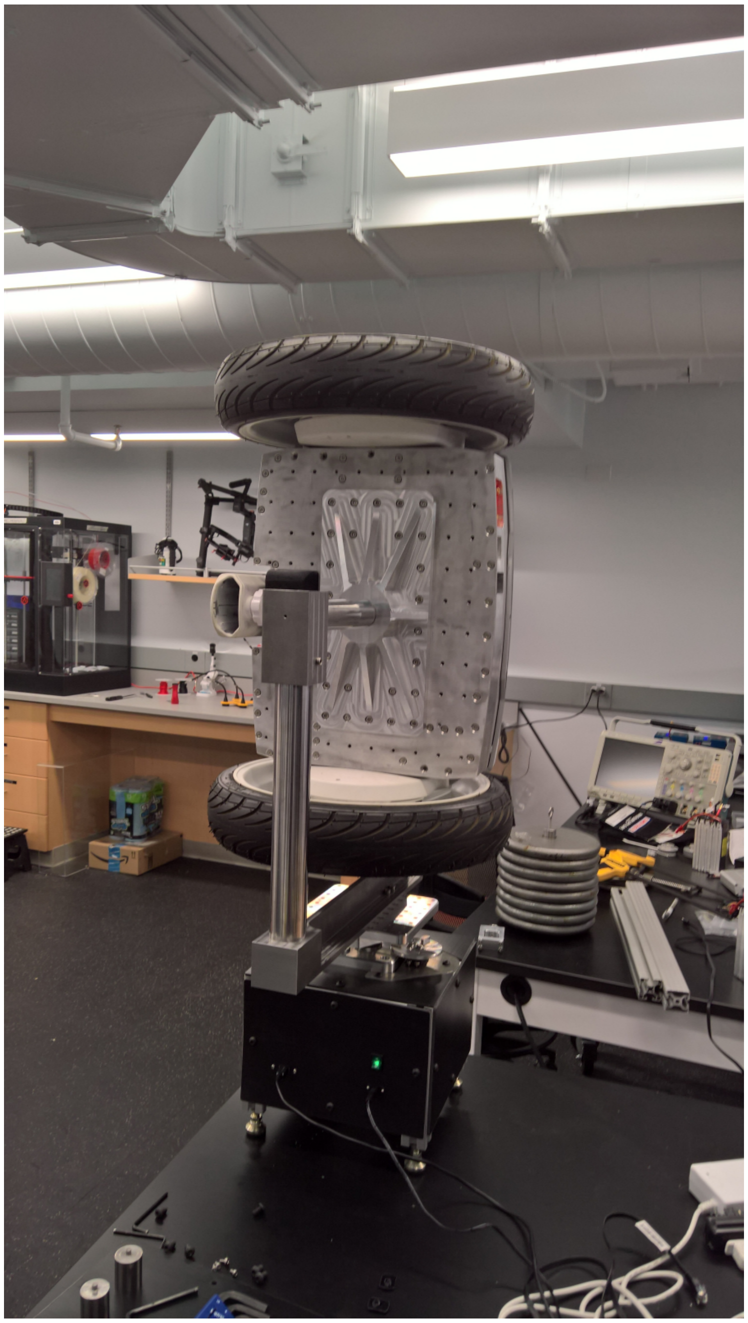


Figure 2.22: Picture of the segway vehicle on the inertia test bed.

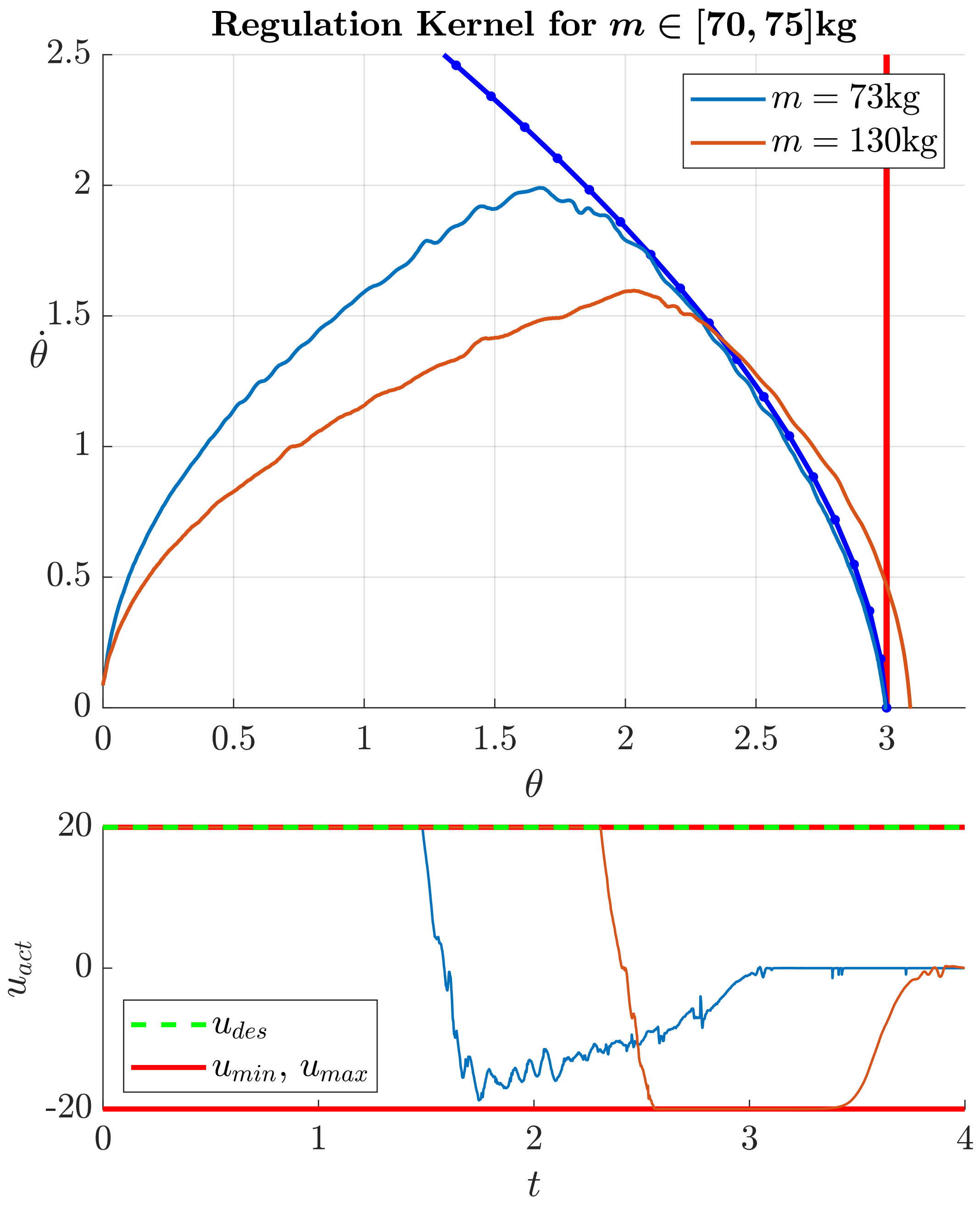


Figure 2.23: Implementations of a Scalable relaxed robust safety filter with valid models and non valid models for 2 different system. $\widehat{N_s} = 5$, $M = 50$, and $\alpha_{des} = 5$.

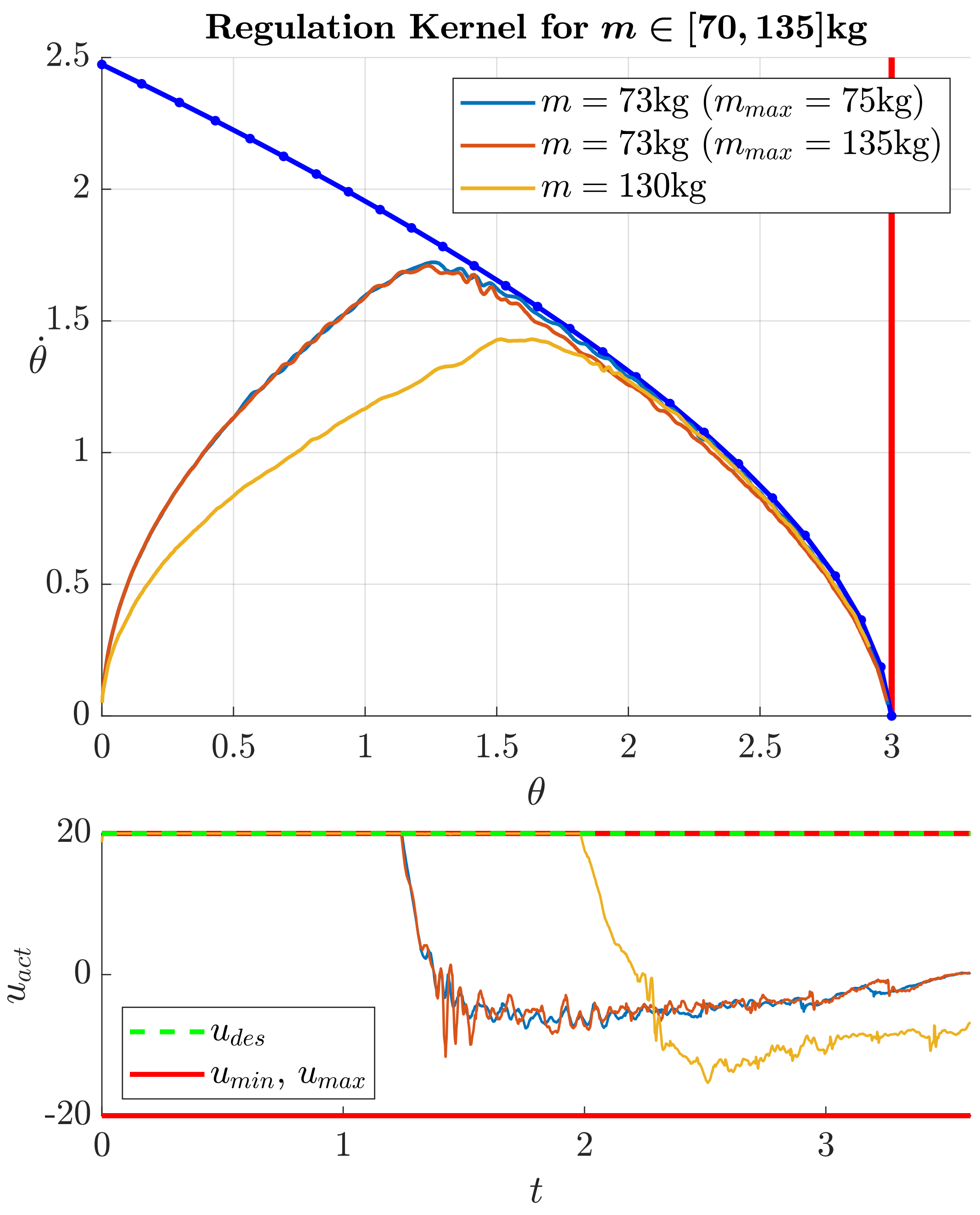


Figure 2.24: Implementations of a scalable relaxed robust safety filter with valid models for 2 different systems. The value of m_{max} correspond the model used for the safety filter, $\widehat{N_s} = 5$, $M = 50$, and $\alpha_{des} = 5$.

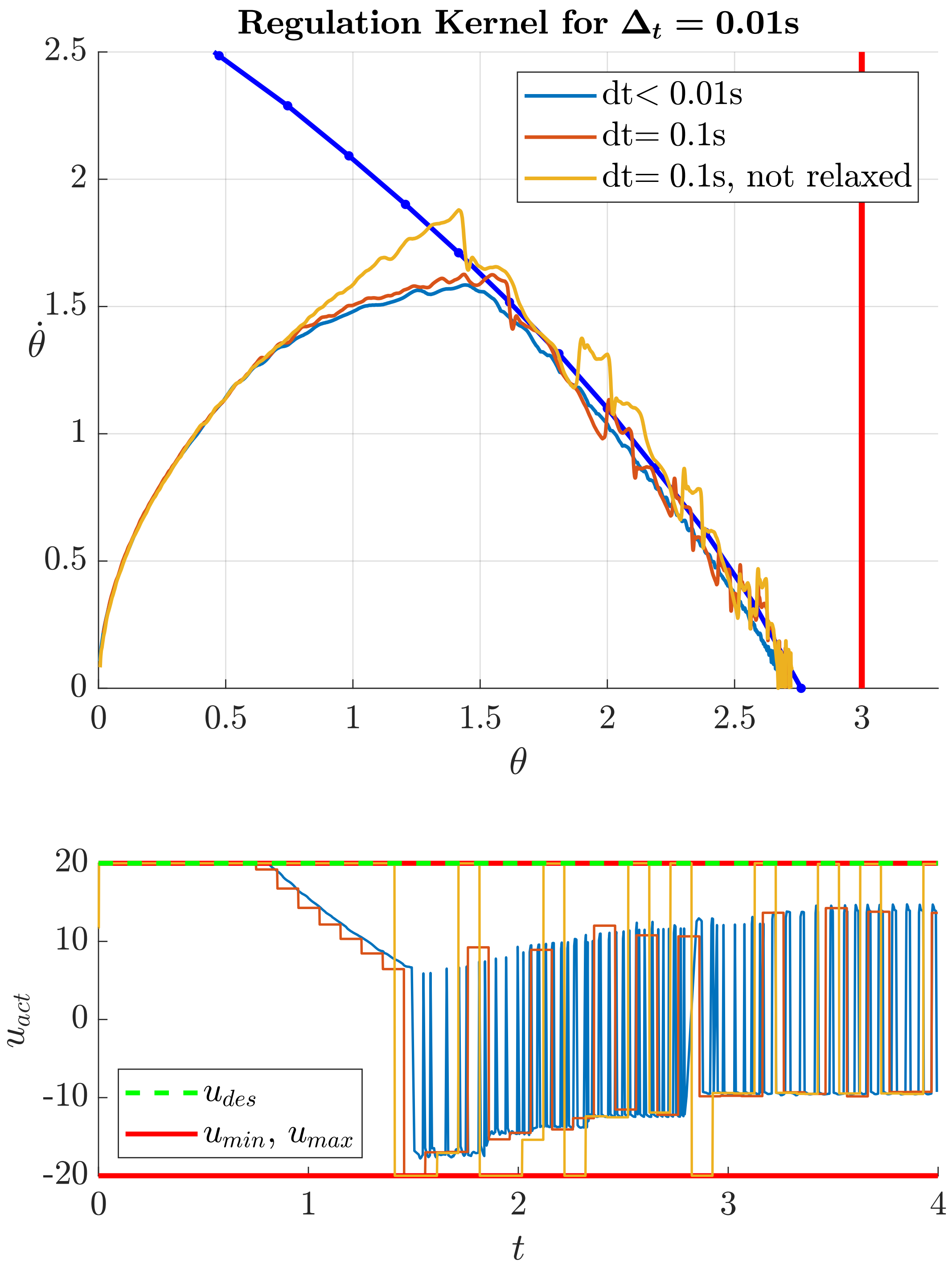


Figure 2.25: Implementations of a scalable realizable safety Filter with various control loop timing resulting in valid and non valid control law realizations. $\widehat{N_s} = 2$, $M = 100$, and $\alpha_{des} = 15$.

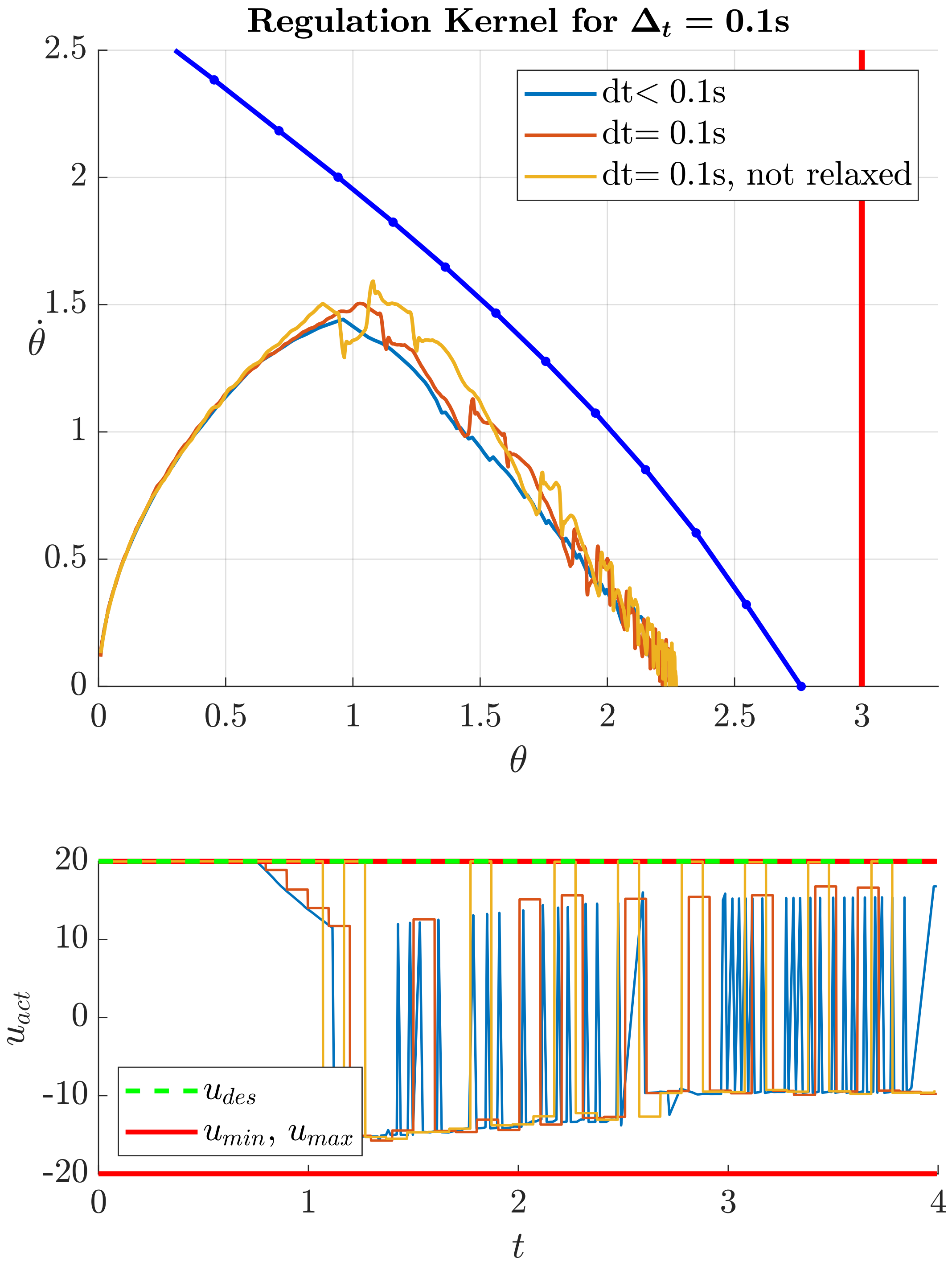


Figure 2.26: Implementations of a scalable realizable safety filter with various control loop frequencies resulting in valid control law realizations. $\widehat{N_s} = 2$, $M = 100$, and $\alpha_{des} = 15$.

Chapter 3

IMPLICIT SAFETY FILTERING

3.1 Introduction

As discussed in the previous chapter, finding explicit representations of large control invariant sets makes it easy to then filter unsafe inputs. Unfortunately, these algorithms take substantial time to run and can only handle high dimensional systems at the expense of conservative results, leading to small operational regions and degraded performances for the system. However, computing viable sets is only one possible approach for ensuring safe operation of a system. Another popular class of methods relies on predicting systems' trajectories to guarantee safety. A backup strategy is chosen, and the trajectory of the system under that backup control law is computed online at every instant. Guaranteeing safety of the system can then be achieved by switching between the nominal and backup controller intelligently based on the safety of the backup trajectory [10, 42]. Multiple backup strategies can also be chosen from on the fly with a similar underlying switching strategy [57, 67]. Furthermore, the backup strategy can be determined on the fly so as to adapt to the situation and be as minimally invasive as possible [44, 71, 74]. This last methodology is at the heart of path planning and optimal control research, but even though it potentially yields the best system performances, its complexity makes practical applications favor the simpler alternatives discussed before.

In this chapter, we propose to unify both set-based and trajectory-based approaches and show that they are really just two sides of the same coin. We present a safety critical control framework that combines the strengths of both approaches to deliver efficient and scalable methods of ensuring safety for complex dynamical systems. First, we will present the safety-filtering methodology from a set-based perspective. We will then show how it is possible to systematically define a control-invariant subset of the safety set, namely a Safe Backward Image (SBI) of the backup set. This set is defined implicitly from a backup control law and a backup set, and the implementation details of a safety filter in that context are then presented. We then show that one can relax the stability requirement on the backup control law and still get meaningful albeit weaker safety guarantees. In the following section, we explore the relation between optimality of the backup control law and size of the Safe Backward Image. In the case of linear systems, we show that it is possible to implement and couple a Model Predictive Controller and a Safety Filter to obtain the largest possible Safe Backward Image. Finally, methods of selecting a safe input with varying levels of trade-off between conservativeness and computational complexity are proposed and illustrated on relevant systems and applications, namely: a two-wheeled inverted pendulum (Segway), an industrial manipulator, a quadrotor, and a lower body exoskeleton.

3.2 Implicit Safety Filtering

The key to our approach is to realise that, in practice, an explicit representation of a control invariant set S is not necessary. If S is practical and can be defined as in (2.24), then one only needs to be able to numerically evaluate $h_i(x)$ and $\nabla h_i(x)$ for any given state $x \in S$ quickly enough for the safety filter to run in real-time. We therefore propose a way to systematically define a control invariant subset of $\overline{S}$.

3.2.1 Implicit Control Invariant Set

Our approach for defining such a set is inspired by [46]. The idea is to start with a "seed of safety": the **backup set**, that is easy to compute explicitly and provide infinite time horizon guarantees, i.e it is control invariant. Ideally, this backup set would be big enough so that we can use it directly for safety filtering, but as discussed previously, explicit safe sets are hard to compute which leads to conservative results and poor performance for high-dimensional systems. Therefore, we chose to "implicitly expand" the backup set over an additional finite time horizon through the flow of the system under a carefully chosen **backup control law**. This way, if this implicit expansion is constructed properly, we get access to a larger control invariant subset of the safety set without going through the time-consuming process of computing an explicit representation of this large set. We will call this implicit expansion of the backup set the **safe backward image** of the backup set. Indeed, as we are about to see, the key for defining a set that is control invariant is to consider all the states that can safely reach the backup set.

For the exposition of this framework, we will only address the perfect model case. Let therefore start with some further assumptions about the perfect model:

Assumption 4. The functions f and g defined on a compact set $X \subset \mathbb{R}^n$ are continuously differentiable. The control policies are restricted to be functions $u : \mathbb{R}^+ \times X \longrightarrow \mathbb{R}^n$ Lipschitz continuous in state over X and piecewise continuous in time over $\mathbb{R}^+$. We furthermore define by $U \subset \mathbb{R}^m$ the compact and convex set of admissible inputs for this system, i.e. $\forall x \in X$ and $\forall t \in \mathbb{R}^+$, $u\left(t, x\right) \in U$.

Assumption 5. Let $\mathcal{U}$ be the set of all continuously differentiable backup control laws taking values in the set of admissible inputs: $u_b : \mathbb{R}^n \rightarrow U$. Under Assumptions 4, we know that for all $u_b \in \mathcal{U}$, there exists a solution to (2.1) that is unique and defined for all times when solutions to (2.1) stay in X. Therefore, one can define $\phi^{u_b} : [0, T_X] \times X \rightarrow \mathbb{R}^n$ to be the **flow** of (2.1) under the control law u_b. Under all these assumptions, the map $\phi_t^{u_b} : X \rightarrow \mathbb{R}^n$ defined by $\phi_t^u\left(x\right) \triangleq \phi^u\left(t, x\right)$ is a homeomorphism of X (cf. [53]) for all $t \in [0, T_X]$. We will denote by $S_b \subseteq \overline{S}$ the non-empty compact **backup set** as depicted in Fig. 3.1. We furthermore assume that S_b is practical and can be represented as the super level set of a smooth function $h_b : \mathbb{R}^n \rightarrow \mathbb{R}$.

Definition 14. We define the **safe backward image** of the backup set to be the set $S_T^{u_b} \triangleq R_T^{u_b} \cap \Omega_T^{u_b}$ where:

$$R_T^{u_b} \triangleq \{x \in X \mid \phi_T^{u_b}\left(x\right) \in S_b\} \tag{3.1}$$

is a set encoding the **reachability** of the backup set under the backup control law and:

$$\Omega_T^{u_b} \triangleq \left\{x \in \overline{S} \mid \forall t \in [0, T]\,,\ \phi_t^{u_b}\left(x\right) \in \overline{S}\right\} \tag{3.2}$$

is a set encoding the safety of the corresponding backup trajectory.

Because $R_T^{u_b} = \left(\phi_T^{u_b}\right)^{-1}\left(S_b\right)$ and $\phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$, the definition of forward invariance can be reformulated in terms of the flow. Indeed, $x\left(t_0\right) \in S \Rightarrow \forall t \geq t_0,\ x\left(t\right) \in S$ is equivalent to $\forall t \geq 0,\ \phi_t^{u_b}\left(S\right) \subseteq S$. Hence the following propositions.

Proposition 5. *If S_b is forward invariant under $u_b \in \mathcal{U}$, then for all $T \geq 0$, $R_T^{u_b}$ is forward invariant.*

Proof. Let us reason by contraction and assume that $R_T^{u_b}$ is not forward invariant. That means there exist $x^* \in R_T^{u_b}$ and $t^* \geq 0$ such that $\phi_{t^*}^{u_b}(x^*) \notin R_T^{u_b}$. Let $\tilde{x} \triangleq \phi_{t^*}^{u_b}(x^*)$. By property of the flow, $\phi_T^{u_b}(\tilde{x}) = \phi_{t^*}^{u_b}\left(\phi_{T-t^*}^{u_b}(\tilde{x})\right)$. But $\phi_{T-t^*}^{u_b}(\tilde{x}) = \phi_T^{u_b}(x^*) \triangleq x_b \in S_b$. So $\phi_T^{u_b}(\tilde{x}) = \phi_{t^*}^{u_b}(x_b) \in S_b$ since S_b is forward invariant. This implies that $\tilde{x} \in R_T^{u_b}$, hence the contradiction that proves the proposition. □

Proposition 6. *The set S_b is forward invariant under $u_b \in \mathcal{U}$ if and only if $S_b \subseteq R_T^{u_b}$ for all $T \geq 0$.*

Proof. Let us first assume that for all $T \geq 0$, $S_b \subseteq R_T^{u_b}$. By definition of $R_T^{u_b}$ and because $\phi_T^{u_b}$ is a homeomorphism, $\forall T \geq 0$, $\phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$. So for all $T \geq 0$, $\phi_T^{u_b}(S_b) \subseteq S_b$ which proves the necessity of forward invariance. The sufficiency of forward invariance follows directly from Prop. 5, as for all $T \geq 0$, $\phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$ and $\phi_T^{u_b}\left(R_T^{u_b}\right) \subseteq R_T^{u_b}$ so $S_b \subseteq R_T^{u_b}$. □

These two propositions are actually fairly intuitive as they indicate that if the backup control law stabilizes the backup set, then the backward reachable set of the backup set is an invariant set larger than the backup set. Let us now see how we can analytically describe this set.

Proposition 7. *Given $u_b \in \mathcal{U}$ and a forward invariant set $S_b = \{x \in \mathbb{R}^n \mid h_b(x) \geq 0\}$, then:*

$$R_T^{u_b} = \left\{x \in \mathbb{R}^n \mid h_b \circ \phi_T^{u_b}(x) \geq 0\right\}. \tag{3.3}$$

Proof. Consider $x \in R_T^{u_b}$, then $\phi_T^{u_b}(x) \in S_b$. So $h_b\left(\phi_T^{u_b}(x)\right) \geq 0$, hence $x \in \left\{x \in \mathbb{R}^n \mid h_b\left(\phi_T^{u_b}(x)\right) \geq 0\right\}$. Let us now consider $x \in \left\{x \in \mathbb{R}^n \mid h_b\left(\phi_T^{u_b}(x)\right) \geq 0\right\}$, then $\phi_T^{u_b}(x) \in S_b$, hence $x \in R_T^{u_b}$. □

The description for $\Omega_T^{u_b}$ is similar.

Proposition 8. *Given $u_b \in \mathcal{U}$ and $\overline{S}$ described as in (2.24), then:*

$$\Omega_T^{u_b} = \bigcap_{t \in [0,T]} \left\{x \in X \mid h_i \circ \phi_t^{u_b}(x) \geq 0,\ i \in [\![1, N_s]\!]\right\}, \tag{3.4}$$

or equivalently:

$$\Omega_T^{u_b} = \left\{x \in X \mid h_{S_T^{u_b}}(x) \geq 0\right\}, \tag{3.5}$$

with:

$$h_{S_T^{u_b}}(x) \triangleq \min_{\substack{t \in [0,T] \\ i \in [\![1,N_s]\!]}} h_i \circ \phi_t^{u_b}(x). \tag{3.6}$$

Proof. This follows directly from the definition of $\Omega_T^{u_b}$ and the expression of $\overline{S}$ as a practical set. $\square$

We can now state the core theoretical proposition of this work.

Proposition 9. *If $S_b \subseteq \overline{S}$ is forward invariant under $u_b \in \mathcal{U}$, then for all $T \geq 0$, the safe backward image $S_T^{u_b}$ is a subset of S that is forward invariant under that control law.*

Proof. The fact that $S_T^{u_b} \subseteq \overline{S}$ follows trivially from the definition of $\Omega_T^{u_b}$. Let us reason by contradiction and assume that $S_T^{u_b}$ is not forward invariant. This means that there exist $x^* \in S_T^{u_b}$ and $t^* \geq 0$ such that $\phi_{t^*}^{u_b}(x^*) \notin S_T^{u_b}$. But from Prop. 5, we know that $R_T^{u_b}$ is forward invariant so $\phi_{t^*}^{u_b}(x^*) \in R_T^{u_b}$ and $\phi_{t^*}^{u_b}(x^*) \notin \Omega_T^{u_b}$. This implies that there exists $t^\# \geq 0$ such that $\phi_{t^\#}^{u_b}(x^*) \notin S$. But $x^* \in \Omega_T^{u_b}$, so $t^\# > T$, i.e. there exists $t' > 0$ such that $\phi_{t'}^{u_b}(\phi_T^{u_b}(x^*)) \notin S$. But $x^* \in R_T^{u_b}$, so $\phi_T^{u_b}(x^*) \in S_b$ and because S_b is forward invariant, $\phi_{t'}^{u_b}(\phi_T^{u_b}(x^*)) \in S_b$, which contradicts $S_b \subseteq S$. $\square$

It trivially follows from this last proposition that $S_T^{u_b}$ is a control invariant subset of $\overline{S}$, so it can be used to define a non-empty sub-regulation map $U_{S_T^{u_b}}$. The challenge now is to be able to evaluate this regulation map, as it only has an implicit expression inherited from the implicit nature of the construction of $S_T^{u_b}$. Let us now see how we can tackle this issue.

3.2.2 Implicit Safety Filter

3.2.2.1 Implicit Sub-Regulation Map

In order to realize a safety filter, one must be able to evaluate the regulation map $U_S(x)$. This is easy if S has an explicit representation, which is why so much effort has been focused on finding such a representation. In our framework however, $S_T^{u_b}$ is defined as a function of $\phi_t^{u_b}$ for which we do not have an explicit representation, hence the implicit nature of $S_T^{u_b}$.

The sub-regulation map $U_{S_T^{u_b}}$ evaluated at a current state x_0 is equal to the set of $u \in U$ such that:

$$\begin{cases} \nabla h_b(x_T)\, D\phi_T^{u_b}(x_0)\, f_0(u) + \alpha_0(h_b(x_T)) \geq 0 \\ \nabla h_i(x_{t_b})\, D\phi_{t_b}^{u_b}(x_0)\, f_0(u) + \alpha_i(h_i(x_{t_b})) \geq 0 \end{cases}, \tag{3.7}$$

for all $i \in [\![1, N_s]\!]$, all $t_b \in [0, T]$, and with $x_{t_b} \triangleq \phi_{t_b}^{u_b}(x_0)$ and $f_0(u) \triangleq f(x_0) + g(x_0)u$.

Two issues arise at the sight of this expression of the sub-regulation map. First, the gradient of the flow—whose existence is guaranteed from the smoothness assumptions in 4 and 5—needs to be computed. Second, because t_b lives in the interval $[0, T]$, the images of $U_{S_T^{u_b}}$ are formed by an **uncountable** set of constraints, which elevates the safety filter's underlying optimization problem into the class of **robust optimization problems**, which is hard, if not impossible to solve in real-time. Let us therefore see how we can address these issues.

3.2.2.2 Numerical Approximation of the Sub-Regulation Map

The practical solution to these issues is to recourse to numerical integration tools. By numerically integrating (2.1) forward in the interval $[0, T]$ under the backup law u_b, $\phi_{t_b}^{u_b}(x_0)$ can be numerically evaluated a discrete times $\{t_{b,0}, \ldots, t_{b,N_b}\}$. It is therefore possible to **approximate** $U_{S_T^{u_b}}(x_0)$ by considering the countable set of constraints at these different times $t_{b,k}$; the more points being considered, the tighter the approximation becomes.

Then, to compute $D\phi_{t_b}^{u}(x_0)$, one only needs to integrate along with (2.1) a **sensitivity** matrix $Q(t_b, x_0)$. As explained in [75], the square matrix $Q(t_b, x_0)$ solution of the following differential equation:

$$\frac{dQ(t, x_0)}{dt_b} = Df_{cl}\left(\phi_{t_b}^{u_b}(x_0)\right) Q(t_b, x_0), \tag{3.8}$$

with $Q(0, x_0) = I$ and where $f_{cl}(x) \triangleq f(x) + g(x)u_b(x)$ is exactly the Jacobian of the flow $\phi_{t_b}^{u_b}$ at x_0:

$$Q(t_b, x_0) = D\phi_{t_b}^{u_b}(x_0). \tag{3.9}$$

Note that it is important for the backup control law used to smooth, which means that in the case of finite input bounds, a smooth saturation function has to be used for the expression of $u_b(x)$.

3.2.2.3 Under-Approximation of the Sub-Regulation Map

We now turn to the issue of having an infinite number of functions defining the set. In practice, we can only enforce positivity of a finite number of the functions in (3.7), and therefore propose a safety filter that enforces positivity of a finite subset of ϵ-tightened constraints evenly spaced in time:

$$\nabla h_b(x_T) D\phi_T^{u_b}(x_0) f_0(u) + \alpha_0(h_b(x_T)) \geq 0, \tag{3.10a}$$

$$\nabla h_i(x_{t_{b,k}}) D\phi_{t_{b,k}}^{u_b}(x_0) f_0(u) + \alpha_k(h_b(x_{t_{b,k}}) - \epsilon_i) \geq 0, \tag{3.10b}$$

for $i \in [\![1, N_s]\!]$ and $k \in [\![0, N_b]\!]$.

Although this just enforces positivity of a finite number of constraints, under some regularity conditions and appropriate margins ϵ_i, we expect that this should be sufficient to guarantee positivity of the whole family of functions. We make this more precise below via the following lemma.

Lemma 2. *Let L_h be the Lipschitz constant of a function h with respect to the Euclidean norm and let:*

$$L_\phi = \sup_{x \in \overline{S}} \|f(x) + g(x) u_b(x)\|_2, \tag{3.11}$$

be the maximal velocity of the backup vector field. Then:

$$\left| h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x) \right| \leq L_h L_\phi |t - s|. \tag{3.12}$$

Proof. Assume WLOG that $t \geq s$ and let $y = \phi_s^{u^B}(x)$. Then:

$$\begin{aligned} \left| h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x) \right| &\leq L_h \left\| \phi_t^{u^B}(x) - \phi_s^{u^B}(x) \right\|_2 \\ &= L_h \left\| \phi_{t-s}^{u^B}(y) - y \right\|_2 \leq L_h L_\phi |t - s|, \end{aligned}$$

since L_ϕ is the maximal velocity of the vector field. □

It follows that invariance of $S_T^{u_b}$ can be enforced via the finite subset of constraints in (3.10) provided that the times $t_{b,k}$ are spaced tightly enough.

Theorem 3. *Let $\eta_i = \max_{k \in [\![1, N_i - 1]\!]} t_{b,k+1} - t_{b,k}$ be the granularity of the time discretization. If for all i it holds that $\epsilon_i \geq L_{h_i} L_\phi \frac{\eta_i}{2}$, then (3.10) enforce invariance of $S_T^{u_b}$.*

Proof. By the same reasoning as before, the constraints (3.10) imply that $h_i(x_{t_{b,k}}) = h_j \circ \phi_{t_{b,k}}^{u_b}(x) \geq \epsilon$ for all $k \in [\![1, N_i]\!]$ for all times. Therefore, by Lemma 2, we can for each $t \in [0, T]$ find a k^* such that:

$$\left| h_i(x_t) - h_i(x_{t_{b,k^*}}) \right| \leq L_{h_i} L_\phi \frac{\eta_i}{2}, \tag{3.13}$$

meaning that:

$$h_i(x_t) \geq \epsilon - L_{h_i} L_\phi \frac{\eta_i}{2} \geq 0. \tag{3.14}$$

Thus all the functions defining $S_T^{u_b}$ are positive, and hence $S_T^{u_b}$ is invariant. $\square$

3.2.3 Numerical Example

We now illustrate these ideas on a simple example of nonlinear inverted pendulum. This system is defined by the state $x = \left[\theta, \dot{\theta}\right]^\top$ and the dynamics:

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \sin(\theta) + u \end{bmatrix} \tag{3.15}$$

with a saturated input $u \in [-u_{max}, u_{max}]$ with $u_{max} = 1.5$. This pendulum is upright at an unstable equilibrium when $\theta = 0$. The safety set is chosen to be a box centered at the origin and of edge size 2π (cf. Fig. 3.2).

The first step is to choose a backup control law and a backup set. In this example, we consider linear backup laws that stabilize the system to the origin:

$$u_b(x) = -K.x \tag{3.16}$$

for some gain vector K. A backup set can then be carefully chosen as a level set of a quadratic Lyapunov function of the linearized dynamics of the system around the origin. As hinted at before, higher backup gains yield in general a larger SBI, as illustrated in Fig. 3.2. It is therefore important to choose a *good* backup law as we will discuss further in Sec. 3.4.

Given a backup control law, a backup horizon T has to be chosen. As one can expect, the larger this time horizon is, the larger the resulting SBI also is, as illustrated in Fig. 3.3.

Given a choice of backup law, set, and horizon, a safety filter can be implemented using numerical integration as explained in Sec. 3.2.2.2. At each safety filter iteration, the dynamics of the system are integrated under the backup law over a time horizon T. From the discrete values of the state and sensitivity matrix over this backup trajectory, a set of linear constraints approximating the regulation map can be constructed. Finally, a quadratic program can be solved to find the best input satisfying the safety constraint encoded by the regulation map. Some trajectories of the system under this safety filter can be seen in Fig. 3.4. In this illustration, the pendulum is started from various initial angles with zero velocity.

The value of the backup horizon therefore has a crucial impact on the applicability of the method as one has to be able to do the numerical integration and the QP solving fast enough for the safety filter to run in real time. Note also that the computational complexity of the backup law and the dynamics of the system have a non negligible impact on the speed of the safety filter computations as they have to be evaluated numerous times for the integration of the backup trajectory.

Finally, it is important to note that, although we compute the SBI explicitly in this example for the purpose of illustration, at no point it is required to do so for the safety filter to operate. Through the numerical scheme we propose, we can render the SBI forward invariant without having to find an explicit representation of it. This makes this framework applicable to nonlinear systems and does not require any particular analytical structure for the dynamics of the system. As we will see in Sec. 3.6, not having to explicitly compute a control invariant set has important advantages in practice.

3.3 Finite Time Safety Guarantees

So far, the backup policy and the backup set cannot be chosen arbitrarily as the former has to be invariant under the latter, which still hinders the scalability of the proposed approach. This comes from our desire for the safe backward image to be a control invariant set, i.e. a set in which the system can remain and evolve forever. In practice however, safety requirements often do not necessitate that the system be able to run forever, but only that the system be able to safely stop or terminate. In this section, we discuss an extension to the proposed approach that allows a safety filter to be used to enforce such safety requirements. As we will see, this extension allows for the backup controller and backup set to be chosen independently which makes our approach truly scalable in this case.

3.3.1 Reformulation of the Reachability Constraint

The idea here is to relax the reachability constraint of $R_T^{u_b}$. For that, we will restrict ourselves to backup sets that can be represented as the upper level-set of a single function h_b **twice differentiable**. In that context, we define the notion of **time to safety**.

Definition 15. Given a backup control law $u_b \in \mathcal{U}$, the **time to safety**

$T_b : X \to \mathbb{R}$ is given by:

$$T_b(x) = \min\left\{t \geq 0 : h_b\left(\phi_t^{u_b}(x)\right) = 0\right\}. \tag{3.17}$$

When $x \in S_b$, we choose $T_b(x) = 0$, and when $x \notin S_b$ and a solution to (3.17) does not exist, we choose $T_b(x) = +\infty$.

Let us now consider the set $\overline{R_T^{u_b}}$ given by the closure:

$$\overline{R_T^{u_b}} \triangleq \overline{\{x \in X \mid 0 < T_b(x) \leq T\}}, \tag{3.18}$$

with $T > 0$ and $u_b \in \mathcal{U}$.

The interest of considering such a set becomes clear when realizing that if S_b is not invariant under u_b, S_b is not a subset of $R_T^{u_b}$ (cd. Fig. (3.1c)). This makes the set $R_T^{u_b}$ unusable as it is not even guaranteed that $R_T^{u_b} \cap \overline{S} \neq \{\emptyset\}$ and that $S_T^{u_b}$ is not empty. The set $\overline{R_T^{u_b}}$ on the other hand will at least contain part of the boundary S_b (provided the backup set is not completely repulsive) and to grow monotonically with T (cf. Fig. (3.1d)).

3.3.2 Augmented Regulation Map

Similarly to Sec. 72, we would like to regulate safe solutions using the sub-regulation map U_S while utilizing this new set $\overline{R_T^{u_b}}$. The barrier condition (2.29) for set $\overline{R_T^{u_b}}$ evaluated at a state $x_0 \in \overline{R_T^{u_b}}$ for a given backup control law $u_b \in \mathcal{U}$ is given by

$$-\nabla T_b(x_0)\, \tilde{f}(x_0, u) + \alpha\left(T - T_b(x_0)\right) \geq 0, \tag{3.19}$$

where $\tilde{f}(x_0, u) \triangleq f(x_0) + g(x_0)u$. As demonstrated in [41]:

$$\nabla T_b(x_0) = -\frac{\nabla h_b(\overline{x}) \cdot D\overline{x}(x_0)}{\nabla h_b(\overline{x}) \cdot \tilde{f}\left(\overline{x}, u_b(\overline{x})\right)}, \tag{3.20}$$

with $\overline{x} \triangleq \phi_{T_b(x_0)}^{u_b}$. One will immediately notice that this gradient is only defined when $\nabla h_b(\overline{x}) \cdot \tilde{f}(\overline{x}, u_b(\overline{x}))$ does not vanish. States for which this happens should therefore be avoided to allow the regulation of safe solutions using U_S with $\overline{R_T^{u_b}}$. Let us therefore consider the following extension of the safe backward image of the backup set—the safe backward reachable set of the backup set:

$$\overline{S_T^{u_b}} = \overline{R_T^{u_b}} \cap \Omega_T^{u_b} \cap C_T^{u_b}, \tag{3.21}$$

with

$$C_T^{u_b} \triangleq \left\{ x_0 \in \overline{R_T^{u_b}} \mid \widetilde{cos}\left(x_0\right) \geq \varepsilon_b \right\}, \tag{3.22}$$

a *small* constant $\varepsilon_b > 0$, and

$$\widetilde{cos}\left(x_0\right) \triangleq \frac{\nabla h_b\left(\overline{x}\right) \tilde{f}\left(\overline{x}, u_b\left(\overline{x}\right)\right)}{\left\|\nabla h_b\left(\overline{x}\right)\right\| \left\|\tilde{f}\left(\overline{x}, u_b\left(\overline{x}\right)\right)\right\|}. \tag{3.23}$$

Proposition 10. *If $\overline{R_T^{u_b}}$ is not empty, then for all $x_0 \in \overline{S_T^{u_b}} \setminus S_b$, there exist strengthening functions α_i such that $U_{\overline{S_T^{u_b}}}\left(x_0\right) \neq \{\emptyset\}$.*

Proof. First, note that ∇T_b is not defined on ∂S_b, hence considering only states in $\overline{S_T^{u_b}} \setminus S_b$. Then, $U_{\overline{S_T^{u_b}}}$ evaluated at a given state $x_0 \in \overline{S_T^{u_b}} \setminus S_b$ is equal to the set of $u \in U$ such that

$$\begin{cases} \nabla h_i\left(x_{t_b}\right) D\phi_{t_b}^{u_b}\left(x_0\right) \tilde{f}\left(x_0, u\right) + \alpha_i\left(h_i\left(x_{t_b}\right)\right) \geq 0 \\ -\nabla T_b\left(x_0\right) \tilde{f}\left(x_0, u\right) + \beta\left(T - T_b\left(x_0\right)\right) \geq 0 \\ \nabla \widetilde{cos}\left(x_0\right) \tilde{f}\left(x_0, u\right) + \gamma\left(\widetilde{cos}\left(x_0\right) - \varepsilon_b\right) \geq 0 \end{cases} \tag{3.24}$$

with $i \in \{1, \ldots, N_s\}$, $t_b \in [0, T_b\left(x_0\right)]$, $x_t \triangleq \phi_t^{u_b}\left(x_0\right)$, β and γ extended class $\mathcal{K}$ functions, and $\tilde{f}\left(x_0, u\right) \triangleq f(x_0) + g(x_0)u$. Firstly, as $D\overline{x}\left(x_0\right) \tilde{f}\left(x_0, u_b\left(x_0\right)\right) = \tilde{f}\left(\overline{x}, u_b\left(\overline{x}\right)\right)$, we have

$$\nabla T_b\left(x_0\right) \tilde{f}\left(x_0, u_b\left(x_0\right)\right) = -1. \tag{3.25}$$

This is fairly intuitive as (3.25) is the time derivative of the time to safety when the system is evolving along the backup trajectory. Secondly, because for all $t_b \in [0, T_b\left(x_0\right)]$, $\overline{x}\left(\phi_{t_b}^{u_b}\left(x_0\right)\right) = \overline{x}\left(x_0\right)$,

$$\nabla \widetilde{cos}\left(x_0\right) \tilde{f}\left(x_0, u_b\left(x_0\right)\right) = 0. \tag{3.26}$$

Finally, because $x_0 \in \overline{S_T^{u_b}} \supset \Omega_T^{u_b}$, for all $t_b \in [0, T_b\left(x_0\right)]$, $\phi_{t_b}^{u_b}\left(x_0\right) \in \overline{S}$. So by continuity of all the functions involved, the α_i can be chosen (cf. [5, 38]) such that for all $x_0 \in \overline{S_T^{u_b}}$:

$$\nabla h_i\left(x_{t_b}\right) D\phi_{t_b}^{u_b}\left(x_0\right) \tilde{f}\left(x_0, u_b\right) + \alpha_i\left(h_i\left(x_{t_b}\right)\right) \geq 0.$$

So all conditions in (3.24) can be simultaneously satisfied in $\overline{S_T^{u_b}}$ by choosing $u\left(x_0\right) = u_b\left(x_0\right)$, hence $U_{\overline{S_T^{u_b}}}$ is well defined and non-empty over all of $\overline{S_T^{u_b}}$. □

Remark 6. The extended class $\mathcal{K}$ functions β and γ can be chosen arbitrarily.

Remark 7. Computing $\nabla\widetilde{cos}\left(x_0\right)$ requires the evaluation of the hessian of h_b at x_0, but that otherwise $U_{\overline{S_T^{u_b}}}$ can be evaluated using the same technique as in Sec. 3.2.2 without any change in complexity of the algorithms. Also note that this new formulation of the sub-regulation map creates additional algorithmic challenges in reliably finding $\overline{x}$ but the details of these numerical issues are outside the scope of this paper.

3.3.3 Weaker but Practical Safety Guarantees

Let us now study what guarantees we get when regulating the system in $\overline{S_T^{u_b}}$ using $U_{\overline{S_T^{u_b}}}$.

Theorem 4. *If $\overline{R_T^{u_b}}$ is not empty, $x\left(0\right) \in \overline{S_T^{u_b}}$, and for almost all $t \in \mathbb{R}^+$, $u\left(t, x\left(t\right)\right) \in U_{\overline{S_T^{u_b}}}$, then there exist $T_s \in [0, +\infty]$ such that for all $t \in [0, T_s)$, $\phi_t^{u(t,x(t))}\left(x\right) \in \overline{S_T^{u_b}}$. Furthermore, if $T_s < +\infty$, $\phi_{T_s}^{u(t,x(t))}\left(x\right) \in S_b$.*

Proof. From Prop. 10, we know that $U_{\overline{S_T^{u_b}}}$ is non-empty on $\overline{S_T^{u_b}} \setminus S_b$ so the sub-tangentiality condition is satisfied on the boundary of that set. Hence from [8, Prop. 4.3.7], we know that the system will never cross the boundary of $\overline{S_T^{u_b}} \setminus S_b$. So if the system ever leaves the compact set $\overline{S_T^{u_b}}$ it will be through $\overline{S_T^{u_b}} \cap S_b$ which is never empty $\overline{R_T^{u_b}}$. □

Concretely, this means that regulating the inputs using $U_{\overline{S_T^{u_b}}}$ (with safety filter (2.31) for example) will guarantee that—if the system starts in $\overline{S_T^{u_b}}$ but outside of S_b, it will either stay in $\overline{S}$ and within *reach* of the backup set S_b within a finite time T, or reach the backup set in finite time (cf. Fig. 3.1d). These guarantees may seem weak compared to the ones in (3.2), but they are actually very relevant in practice.

For autonomous systems for example, the priority is (almost) always given to the avoidance of human casualty over the integrity of the system. Being able to safely *terminate* the system is often all that is requested (cf. [1] for more details in the case of UAVs). For commercial aviation and transoceanic flights, being able to safely reach an airfield within a set amount of time is the safety criterion used by the Federal Aviation Administration (cf. ETOPS). In this case, the modality of landing the plane—what happens once the system has reached the backup set—can be handled separately.

Finally, it will not be proven here but, it is easy to verify that $\left(\overline{R_T^{u_b}} \cup S_b\right) \supseteq R_T^{u_b}$ and that under some mild assumptions, when S_b is forward invariant under u_b, $\left(\overline{R_T^{u_b}} \cup S_b\right) = R_T^{u_b}$ (cf. Fig. 3.1b). Therefore, when S_b is forward invariant (and ε_b small enough), the present approach yields the original safety guarantees of Sec. (3.2), i.e. the system remains in $\tilde{S}$ for all times. Hence the soundness of this approach that provides *weak but practical* safety guarantees without the challenge of having to verify the forward invariance of the backup set under u_b, but also provides *strong* safety guarantees when the backup set is actually forward invariant under u_b (cf. Fig. 3.1).

3.3.4 Numerical Example

We now illustrate this approach on the nonlinear inverted pendulum of Sec. 3.2.3. In this case, we assume that the pendulum shall not go past $\theta = -\frac{\pi}{2}$ on one side, but that there is a hard stop on the other side at $\theta = \frac{\pi}{2}$ that the system can run into to safely stop. The backup set is therefore chosen to be a narrow band around $\theta = \frac{\pi}{2}$, and the safety set is a rectangle with edges $\theta = -\frac{\pi}{2}$ and $\dot{\theta} = \pm\frac{\pi}{3}$, so opened towards the edge $\theta = \frac{\pi}{2}$ (cf. Fig. 3.5). The backup policy is chosen to be $u_b(x) = 10 * (\frac{\pi}{10} - \dot{\theta})$ such as to drive the system back towards the hard stop.

In this case, the backup set is certainly not invariant under the backup policy, so it is not the safe backward image of the backup set as illustrated in Fig. 3.5. The safe backward reachable set on the other end is well suited for this scenario where only finite time safety is required. Indeed, as can be seen in Fig. 3.5, the safe backward reachable set fully captures the set of safe states and allows for safe operations. As illustrated in Fig. 3.6, if the pendulum is left to fall towards negative θs, the associated safety filter slows it down so that it stops before $\theta = -\frac{\pi}{2}$, but if it is left to fall towards positive θs, the filter makes sure the system safely reaches the backup set.

3.4 Optimality of the Backup Controller

3.4.1 Model Predictive Backup Controller

In light of the results illustrated in Fig. 3.2, a natural question that arises is "what is the *best* backup control law to choose", that is, a control law that maximizes the size of the safe backward image for a given backup set and backup horizon.

To address this question, let us consider the control law u^* given by

$$u^*(x_0) = u^*_{x_0}(0) \tag{3.27}$$

with $u^*_{x_0}$ being the control policy solution to the following optimal control problem:

Backup MPC

$$\begin{aligned} u^*_{x_0} \triangleq \underset{u \in \mathcal{U}_T}{\operatorname{argmax}} \quad & J(x(t), u(t)) \\ \text{s.t.} \quad & \dot{x} = f(x) + g(x)u \\ & x(0) = x_0 \\ & u(t) \in U, \ \forall t \in [0, T] \\ & x(t) \in \overline{S}, \ \forall t \in [0, T] \\ & h_b(x(T)) \geq 0 \end{aligned} \tag{3.28}$$

---

for some cost function J and where $\mathcal{U}_T$ denotes the set of all piecewise continuous *control policies*.

Remark 8. The time in (3.28) is shifted so that $x_0 = x(t_0) = x(0)$ as only time-invariant control systems are considered in this section.

Remark 9. Solutions to (3.28) might not be unique for a given x_0, in which case the right-hand side of (3.28) is chosen to be any element in the set of solutions.

It immediately follows that

$$S_T^{u^*} = \left\{ x_0 \in X \mid \exists u \in \mathcal{U}_T, \text{ s.t. } \begin{array}{l} x_{x_0}(t) \in \overline{S}, \ \forall t \in [0, T] \\ \text{and } x_{x_0}(T) \in S_b \end{array} \right\}, \tag{3.29}$$

where $x_{x_0}(t)$ is the solution to (2.1) starting at x_0 at $t = 0$ under the time based control policy $u(t)$.

Therefore, for all $u_b \in \mathcal{U}$, $S_T^{u_b} \subseteq S_T^{u_*}$. We can therefore conclude that u_* is an optimal backup law in the sense that it yields an upper bound on the largest possible safe backward image for a given backup set, as illustrated in Fig. 3.7. However, u_* is not well suited for the current framework.

A couple of points are important to address if one wants to use the u^* law in the proposed framework. First, the cost function J has to be chosen preferably so as to stabilise S_b (see [46] for such conditions). Secondly, u^* must be smooth (or at least continuous if one uses the filtering approach of Sec. 3.5), which is rarely the case in general. Finally, if one is able to make u^* smooth, solving a nonlinear MPC online is not an easy task, and is even harder when one has to compute the gradient of the flow under u^* along with the optimal trajectory.

In some cases, these issues can be successfully addressed and an MPC backup controller can be used directly as showcased in [37]. Nevertheless, in the general case, it is not possible to use an MPC backup controller online, and we have to default to a more offline approach. In particular, it is possible to get near-optimal safe backward images by finding a smooth explicit approximation of the optimal backup control law.

3.4.2 Neural Network Approximation of the MPC

In practice, one can use any functional basis of choice to fit the optimal backup policy, provided that it is fast enough to numerically compute along with its gradient. A smooth functional basis for approximating complex functions can be found with feedforward neural networks, whose recent popularity has made the associated tools very efficient. In particular, evaluating gradients of feedforward neural network is computationally easy.

Indeed, for simple neural networks with recursion law:

$$y_{i+1} = f\left(w_i y_i + b_i\right), \tag{3.30}$$

the gradient of the entire neural network can be computed in the same forward pass with the following recursion law:

$$dy_{i+1} = \operatorname{diag}\left(\frac{df(y_{i+1})}{dx}\right) w_i dy_i, \tag{3.31}$$

with dy_0 being the identity matrix of size n (state dimension).

Referring back to the nonlinear inverted pendulum of Sec. 3.2.3, we can see in Fig. 3.7 that the optimal safe backward image is larger than the largest SBI we found using linear feedback for the backup controller (cf. Fig. 3.2).

We first solve the OCP (3.28) with the cost function:

$$J\left(x(t), u(t)\right) = \int_0^T \left(0.1u(t)^2 + 10\theta(t)^2 + \dot{\theta}(t)^2\right) dt \tag{3.32}$$

over a grid of size 200×200 using GPOPS-II [69]. We then fit a neural network with 2 hidden layers, each of size 35, and with a hyperbolic tangent activation function over the generated data. The resulting safe backward image of this approximately optimal policy, as can be seen in Fig. 3.7, captures most of the optimally of the OCP, but, is actually usable in the proposed framework. This method is applied on a larger dimensional system in Sec. 3.6 and similarly strong results are observed.

3.5 Scalable Implicit Safety Filter

So far, we have been able to use an optimization-based safety filter by relying on our ability to numerically evaluate the sub-regulation map at any given state. Although this approach is optimal in terms of filtering, it comes at the cost of having to compute the gradient of the flow along the backup trajectories. The dimension of the system to integrate is therefore $n + n^2$, which can become a computational bottleneck for higher dimensional systems. It is however, possible to sacrifice optimally of the safety filter for better scalability. The key is that **by construction**, the backup law evaluated at the current state is always an element of the image of the regulation map for that state. In other words, if the backup policy is followed for the initial conditions inside $S_T^{u_b}$, the system will remain in the safety set for all time.

3.5.1 Smooth Switching to the Backup Control Law

This idea is at the core of a lot of alternative approaches to safety filtering [10, 42, 44, 57, 67, 71, 74]. In most of these methods though, the safety filter just operates a simple switch between nominal and backup controller until the system has reached the backup set, which in practice is fairly intrusive.

The natural evolution of this idea is to implement a smooth transition between desired and backup inputs. To that end, let us look at the following proposition.

Proposition 11. *Given a nonlinear control system (2.1) with a corresponding backup controller $u_b \in \mathcal{U}$ and a continuous function $\alpha : \mathbb{R}^+ \times \mathbb{R}^m \times X \times \mathbb{R} \to U$, the control law defined by*

$$u_f(t, x) \triangleq \alpha\left(t, u_{des}(t), x, h_{S_T^{u_b}}(x)\right) \tag{3.33}$$

is a continuous selection of $\overline{U_{S_T^{u_b}}}$ if for all $t \geq 0$ and $x \in S_T^{u_b}$, u_{des} is continuous and:

$$\alpha\left(t, u_{des}(t), x, 0\right) = u_b(x). \tag{3.34}$$

Proof. This follows trivially from the fact that for all $x \in S_T^{u_b}$, $u_b(x) \in \overline{U_{S_T^{u_b}}}$ and that $\overline{U_{S_T^{u_b}}}$ is non-trivial only when $h_{S_T^{u_b}}(x) = 0$. □

This means that by choosing a **switching function** α appropriately, it is possible to *mimic* the behavior of a QP-based safety filter without the added computational complexity. Note, however, that this approach is fundamentally more conservative than with a QP-based filter as when $h_{S_T^{u_b}}(x) = 0$, it enforces $u_{act}(x) = u_b(x)$ whereas with a QP-based filter, $u_{act}(x) \in U_{S_T^{u_b}}$, which is in general larger than the singleton $u_b(x)$. Nonetheless, it is possible with a proper choice of filtering function to get good performances in practice. Especially since on a significant part of the boundary of the viability kernel of $\overline{S}$, $U_{S_T^{u_b}}$ is actually reduced to a singleton [64].

3.5.2 Numerical Example

We now illustrate this approach on the nonlinear inverted pendulum of Sec. 3.2.3. The safety specifications are the same as in Sec. 3.2.3. The switching function is first chosen to be a basic ramp up of the backup input near the boundary of the safe backward image:

$$\alpha\left(t, x, h_{S_T^{u_b}}(x)\right) = \left(1 - h_{S_T^{u_b}}(x)\right)^6 u_b(x). \tag{3.35}$$

Some trajectories of the system under this scalable safety filter can be seen in Fig. 3.8. As expected, this safety filter is more conservative than the QP-based one of Fig. 3.4, which often translates into an oscillatory behavior near the boundary of the safe backward image. However, it is possible to mitigate this behavior with a better choice of filtering function. For example, let us consider the switching function defined by:

$$\alpha\left(t, x, h_{S_T^{u_b}}(x)\right) = \sigma_0^1\left(\lambda(x) + (1 - \lambda(x))\lambda_d(t)\right) u_b(x) \tag{3.36}$$

with

$$\lambda(x) \triangleq \left(1 - h_{S_T^{u_b}}(x)\right)^6, \tag{3.37}$$

$$\lambda_d(t) \triangleq -\zeta \frac{dh_{S_T^{u_b}}(x(t))}{dt}, \tag{3.38}$$

and σ_0^1 the saturation function between 0 and 1. As illustrated in Fig. 3.9, adding such a damping term can help reduce the oscillatory nature of such scalable filters. A further approach for enhancing the performance and applicability of these scalable safety filters will be presented in Sec. 3.6.4.

3.6 Applications

3.6.1 Two-Wheeled Inverted Pendulum (Segway)

3.6.1.1 Hardware Setup

The Segway platform used here began as a Ninebot E+. All of the electronics, including sensors and motor controllers, were removed, as well as the steering column. The steering column and human rider were replaced by a steel column with weights near the top, to simulate the mass and inertia of a human. The original motor controller was replaced by a set of Elmo Gold Solo Twitters, which provide direct current control of the motors. The onboard sensing is performed using wheel incremental encoders and a VectorNav VN-100 IMU. A Teensy 3.5 reads the sensor data and acts as the state observer for the robot.

The Teensy sends the state information to the main computer onboard, a Jetson TX2, which computes the control action that is sent to the motor controllers. The TX2 runs standard Linux and the ERIKA3 real-time operating system concurrently through the Jailhouse hypervisor. The Linux OS runs ROS, which allows external communication and logs all of the necessary data. The real-time operating system handles the communication with the Teensy and the computation of the control actions. These two operating systems are able to share information through a shared memory interface. All of the code running on the Segway is written in C++.

3.6.1.2 Segway Test

In order to test the implicit safety filter on the Segway, a model of the dynamics is required. The equations of motion are derived via Newton-Euler method, treating the Segway as a two-wheeled inverted pendulum with torque inputs at each wheel. For this experiment, the planar model is used, consisting of four states: position (p), velocity ($\dot{p}$), pitch angle (ψ), and angular rate ($\dot{\psi}$) (cf. Fig. 3.10a). Since the motor controllers command current, the motor torque constant is estimated via system identification. The other necessary parameters, including the mass and inertia properties of the Segway frame and wheels, were measured using various testbeds.

The next step is to define the safety set for the test. This is simply defined as bounds on all of the states, with $p \in [-1, 1]$ m, and $\psi \in [-\frac{\pi}{6}, \frac{\pi}{6}]$ rad. The input bounds are $u \in [-20, 20]$A.

After identifying the system dynamics and determining a safety set, a backup set and backup controller must be generated. The backup set must be a subset of the safety set that is invariant under the backup controller. Thus, we choose a backup controller that stabilizes the Segway to the origin at its equilibrium angle. This is achieved with a simple LQR controller. To obtain the backup set, we compute a small region of attraction for this controller about the origin in the form of a level set of a quadratic Lyapunov function of the linearized dynamics of the system.

To implement the safety filter, a C++ implementation of the proposed safety filter was developed. The library requires an expression for the dynamics, the backup controller, the gradient of the closed-loop dynamics, the backup set, the safety set, and the gradient of the safety set. The library integrates the dynamics using an Euler scheme. The resulting quadratic program is solved using a modified version of OSQP [79] that can be compiled on the real-time operating system.

To showcase the effectiveness of the proposed approach, a simple scenario is executed on the Segway with and without an implicit safety filter. The nominal controller is a simple LQR that can be commanded a desired position. A sequence of desired positions outside of the safety set are commanded, and as can be seen in Fig. 3.12 and Fig. 3.11, without safety filter, the system blithely breaches the safety set to the point where it falls to the ground when the command is too aggressive. On the other hand, with the proposed implicit safety filter, the system stably remains inside the safety set despite the unsafe desired inputs. Note that for this experiment, the filter ran at 800Hz on the embedded hardware with a backup horizon $T = 1$s and an integration time-step of 0.01s. A video of this experiment can be found in [31].

3.6.2 Industrial Manipulator in Time Varying Environment

Next, we apply the infinite-time implicit safety method described in Section 3.2 to the problem of collision avoidance in a dynamic environment. The global industrial robot market has more than doubled in the past five years, and the International Federation of Robotics expected almost two million new robot installations in factories by 2020 [45]. However, concern for the safety of their human counterparts grows along with the density of robots in factories. As a result, in heavy manufacturing, machines and humans are mostly separated.

This makes the process rigid: it becomes spatially constrained and manual intervention in the vicinity of a robot may require halting the process altogether. To reduce downtime and allow for more human-robot interaction, we would like to be able to ensure that these robots cannot collide with human operators under any circumstances while also avoiding having to stop the robot altogether when a worker is in its vicinity. More information about this work can be found in [77].

3.6.2.1 Problem Formulation

This problem is complex because human workers can move around the robot in a somewhat unpredictable manner. In order to guarantee that no collision can occur with such dynamic and uncertain targets, their is no choice but to assume all possible movements of the workers and avoid all of them. In other words, the safety set in this scenario has to be the complement of the forward reachable set of the human workers. Such a set spans space **and** time, so the state of the system has to be augmented to include this time dependency. Furthermore, the forward reachable set, and so the safety set, has to be periodically updated to account for the actual movement of the worker and avoid very punitive conservativeness.

A major advantage of our implicit safety filtering over explicit safety filtering is that the safety set can be changed on the fly without any additional computations, whereas with an explicit safe set approach, if the safety set changes, a new safe set has to be computed, which is time consuming and represent a real bottleneck that hinder the capability of these explicit approaches to handle practical scenarios such as this one.

Let us consider the 6-link IRB 6640 manipulator from ABB depicted in Figure 3.13 that has six degrees of freedom. The dynamics of this robotic arm can be written in a classic manipulator equations form:

$$M(q)\ddot{q} + C\left(q, \dot{q}\right)\dot{q} + G(q) = \tau, \tag{3.39}$$

where q describes the joint angles and τ is a vector of applied torques.

For manipulators with many degrees of freedom, the explicit expressions for $M(q)$, $C(q, \dot{q})$ and $G(q)$ are very complicated. As an alternative, they can be evaluated at given points via the Articulated Body Algorithm (ABA) that steps over links of the manipulator in a recursive fashion [25]. Only having "black-box" access to the equations of motion would pose a problem for most methods for finding invariant sets, but the implicit method proposed in this paper only requires access to the numerical values of the dynamics and its derivatives. As discussed above, the state of the system is extended to contain time which makes the overall system's state 13-dimensional:

$$x = [q, \dot{q}, t]^T. \tag{3.40}$$

A point in the Cartesian and time space will be denoted by:

$$k = [a, b, c, t]^T. \tag{3.41}$$

3.6.2.2 Safety and Backup Sets

The backup set is considered to be a vertical tube around the robot. In practice, this would be a small closed-off area that is inaccessible to the human. For this implementation, it is described by angle constraints on the second and third joints (the joints being enumerated from the base to the end effector):

$$S_b = \left\{ x \in \mathbb{R}^{13} \;\middle|\; \begin{array}{l} q_2 \in \left[-\frac{\pi}{12}, \frac{\pi}{12}\right] \\ q_3 \in \left[-\frac{7\pi}{12}, -\frac{5\pi}{12}\right] \end{array} \right\}. \tag{3.42}$$

The safety set is then simply the union of the backup set and complement of the reachable set of the human in space-time over the duration of the backup maneuver. For the purpose of this demonstration, the human is modeled as a single integrator with a maximum velocity $v_{\max}$, meaning that the size of its reachable set grows linearly in time. By adding time as a state, we prevent the filter from being overly conservative, which would be the result if we only used the reachable set of the human over the time horizon of the backup controller.

If (a_0, b_0) is the current horizontal Cartesian position of the human, the reachable set of the human can be simply expressed as an n-cylinder [51] centered at $(a_0, b_0, H/2)$ in Cartesian space, where H is the height of the human. We can then write this set as the lower-level set of a time-dependent differentiable function $h_r : \mathbb{R}^4 \to \mathbb{R}$ defined by:

$$\begin{aligned} h_r(k) = {} & (a - a_0)^2 + (b - b_0)^2 \\ & + \frac{(c - \sqrt{H})^2 (r_0 + v_{\max} t)^2}{H} \\ & - (r_0 + v_{\max} t)^2. \end{aligned} \tag{3.43}$$

Thus, for the robot to not come in contact with the human, $h_r(k)$ must be positive for all physical points k along the robot. However, because the dynamics of the robot are defined in joint space and the safety set is defined in Cartesian space, one must be careful when using the h_r. Let us denote our forward kinematics function that takes a point from joint and time space to Cartesian and time space, by $K : \mathbb{R}^{13} \to \mathbb{R}^4$. The gradient of h_r with respect to the states is:

$$\frac{\partial h_r(k)}{\partial x} = \frac{\partial h_r(K(x))}{\partial x} = \frac{\partial h_r(K(x))}{\partial k}\frac{\partial K(x)}{\partial x}, \tag{3.44}$$

where:

$$\frac{\partial K(x)}{\partial x} = \begin{bmatrix} \frac{\partial K}{\partial q} & \frac{\partial K}{\partial \dot{q}} & \frac{\partial K}{\partial t} \end{bmatrix} = \begin{bmatrix} J & \vec{0} & J\dot{q} \\ \vec{0} & \vec{0} & 1 \end{bmatrix}, \tag{3.45}$$

with the kinematic Jacobian J being computed numerically.

The safe set h is then defined as the union of h_r and the backup set, S_b, in order to avoid issues with the reachable set of the human intersecting with the backup set, which is not possible in reality.

Finally, as the system evolves, the reachable set of the worker gets larger in the Cartesian space. It is therefore important to **update** this reachable set when a new measurement (a_0, b_0) is available so as to avoid the reachable set filling the entire work envelope of the robot. A fundamental constraint for this update to be possible is that the resulting safety set must be larger after the update than before. In our case, this is guaranteed by the fact that any new position of the human will be contained in its reachable set (cf. Fig. 3.14), so the safety set grows in the full state space, even though it does not when only looking at Cartesian space.

3.6.2.3 Backup Controller

For the backup controller, we leverage the power of the Recursive Newton-Euler Algorithm (RNEA) [50], which provides the necessary joint torques to generate desired joint accelerations. The flexibility of this method is again showcased by the fact that we do not need an analytic expression for the backup controller, as long as we know its gradient.

There are only two joints that require actuation to reach the backup set. A simple PD controller is used to obtain desired joint accelerations for these

Table 3.1: Computation time for IRB 6640 in Pinocchio

Expression	Time (μs)
Affine forward dynamics ($f(x)$ and $g(x)$)	4
Gradient of closed-loop forward dynamics	42
Backup controller	5
Gradient of backup controller	31

joints, which is fed into the RNEA that generates the control inputs, as well as their gradient. The controller is of the form:

$$\begin{aligned} a_{\text{des}}(q,\dot{q}) &= -k_p(q-q_d) - k_d(\dot{q}), \\ u_{\text{b}}(q,\dot{q}) &= \text{RNEA}(q,\dot{q},a_{\text{des}}(q,\dot{q})). \end{aligned} \tag{3.46}$$

The gradient of this backup controller, which is required to evaluate the sub-regulation map, is described by:

$$\begin{aligned} \frac{\partial u_b}{\partial q} &= \frac{\partial \text{RNEA}}{\partial q} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}\frac{\partial a_{\text{des}}}{\partial q} = \frac{\partial \text{RNEA}}{\partial q} - k_p \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}, \\ \frac{\partial u_b}{\partial \dot{q}} &= \frac{\partial \text{RNEA}}{\partial \dot{q}} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}\frac{\partial a_{\text{des}}}{\partial \dot{q}} = \frac{\partial \text{RNEA}}{\partial \dot{q}} - k_d \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}, \\ \frac{\partial u_b}{dt} &= 0. \end{aligned}$$

Since the RNEA provides the exact torques needed to achieve desired joint accelerations, the forward invariance of the backup controller is guaranteed under the proper choice of desired joint accelerations.

3.6.2.4 Simulation

The rigid body algorithm library used for this simulation is Pinocchio [19]. This C++ library has been shown to be the fastest of its kind, with the Table 3.1 showcasing the average computation times of each necessary expression for the robot.

A ROS environment was created to simulate the system, with V-REP used as a visualizer. The ROS package consisted of five nodes: the robotic arm (PLANT), the task giver (TASK), a nominal controller (CONT), the human (HUMAN), and the safety filter (ASIF), connected as shown in Figure 3.15. Each component of the system ran at 200 Hz on a desktop PC with an Intel 8700k processor. The dynamics were integrated in the plant node via the

ODEint C++ library, with the `runge_kutta_dopri5` scheme over a timestep of 5 ms.

The controller node tracked a sequence of desired end-effector positions given to it by the task giver node. Once the system reached the desired position, the task giver would send a new desired location to the system to mimic a typical operational cycle for such robot. The RNEA approach is also used for this tracking controller. The human node allowed the user to joystick a human, modeled as a single integrator, around the factory floor.

Lastly, the safety filter node handles safety for the system. It takes in the state from the robot and the desired inputs from the controller, and outputs the actual inputs that is used for integration by the plant.

The ASIF uses an adaptive-step RK4 scheme for integration under the backup controller, and the resulting quadratic program is solved by the OSQP library [79].

Figure 3.16 shows the value of the ASIF when a human attempts to pass through the working area of the arm. This image well illustrates the minimally invasive property of the ASIF, as the filter keeps the value of $h(x)$ just barely above zero. For a video demonstration of the filter's capabilities, please see [32].

3.6.3 Variable Assistance for Lower Body Exoskeletons

Next, we apply this framework, and more specifically the approach of Sec. 3.5, to the problem of variable assistance for lower body exoskeletons. Contrary to most common applications of this framework, this particular one is not about safety. The main focus of this application is exoskeleton technology aimed at restoring locomotion for people with a leg pathology. A general review of control strategies for lower-limb assistive devices is given in [6, 48, 82]. Most current approaches to control powered leg devices are driven by finite-state machines with each phase defined using heuristic parameters. These approaches typically require the use of additional stability aids such as arm-crutches. Recently, dynamically stable crutch-less exoskeleton walking has been demonstrated for patients with paraplegia by leveraging the full nonlinear dynamics of the system and generating dynamically stable gaits [36]. While this approach enables crutch-less exoskeleton walking, it is no longer optimal when exoskeleton technology is extended to patients who are recovering muscle functionality.

For patients who are trying to strengthen recovering muscles, partial assistance would be more appropriate than full assistance. A previous study showed that permitting partial assistance and variability during step training enhanced stepping recovery after a complete spinal cord transection in adult mice [18]. As we are about to see, the framework presented in this paper can be used to enable assist-as-needed strategies while guaranteeing coherence of the walking pattern. The method presented here allows users to control their own motions when they are performing well, but intervene when they are not, so as to maintain a functional walking pattern. More details about this method can be found in [30].

3.6.3.1 Atalante Exoskeleton Design

The exoskeleton used for this work, named Atalante, was developed by the French startup company Wandercraft and has already demonstrated its ability to perform crutch-less dynamic walking with patients with paraplegia [36]. As shown in Fig. 3.18, this lower-body exoskeleton has 12 actuated joints. Each leg of the exoskeleton consists of three actuated joints controlling the spherical motion of the hip, a single actuated joint for the flexion/extension motion of the knee, and two actuated joints for the hinge motion (inversion/eversion, dorsiflexion/plantarflexion) at the ankle. The joints controlling the motion of the hip and the knee are each actuated by a brushless DC motor. The ankle joints have a more complex actuation mechanism that provides rotation in the sagittal plane and about the henke axis. The position and velocity of each actuated joint is measured using a digital encoder. Additionally, the exoskeleton has four Inertial Measurement Units (IMUs) that are positioned on the torso, the pelvis, the left shank, and the right shank. These IMUs are used to provide additional information about the attitude of the robot with respect to the world. To detect ground contact, four 3-axis force sensors are attached to the bottom of each foot. All of the actuators and sensors are controlled by an embedded computer unit running a real-time operating system.

Other components of the exoskeleton include secure loops for mounting the exoskeleton to an overhead hoist, buttons to change the operating mode of the exoskeleton, a connection port to connect the exoskeleton to a computer, handles on either side of the exoskeleton for the operator to assist the exoskeleton if needed, thigh and shank harnesses to secure a patient to the exoskeleton, thigh and shank length adjustments to change the dimensions of the exoskeleton to match that of a patient, and a torso harness that a patient wears to secure their torso to the exoskeleton (cf. Fig. 3.17).

3.6.3.2 Patient-Exoskeleton Model Generation

A model of each patient is generated to account for each person's unique physical characteristics. Key measurements of each patient are made such as: height, mass, thigh length, and shank length. The thigh length is approximated by the measurement between the gluteus maximus and the patella when the patient is in a seated position. The shank length is approximated as the measurement between the femoral condyles and the ground when the patient is in a seated position. The measured thigh length and shank length are used to adjust the leg lengths of the exoskeleton to match that of the patient. The patient model is then created as follows. First, the total height of the patient is used to extrapolate the length of each segment of the patient model. The segments were chosen to be: Head, Arms and Trunk (HAT); Pelvis; Left Thigh; Left Shank; Left Foot; Right Thigh; Right Shank; Right Foot. These extrapolations were first derived by Drillis and Contini [23]. Using these segment lengths, the center of mass (COM) and inertial for each body segment are then calculated using anthropometric data [87]. The inertia and COM of each segment is given with respect to the proximal end of that segment. The inertia and COM of each segment are then combined with those of the corresponding segments of the rigid body exoskeleton model to form the patient-exoskeleton system.

This combined human-exoskeleton system can be mathematically represented as a rigid body system. A floating-base generalized coordinate system is constructed as $q = (p, \phi, q_b) \in Q \in \mathbb{R}^{18}$, where $p \in \mathbb{R}^3$ and $\phi \in \mathbb{SO}^3$ denote the position and orientation of the exoskeleton's base frame with respect to the world frame. The relative angles of the actuated joints are denoted by $q_b \in \mathbb{R}^{12}$. In total, the system has 18 degrees of freedom, and is fully actuated when one foot is flatly in contact with the ground.

3.6.3.3 Variable Assistance Framework

As discussed in [18], the correct muscle activation pattern is an important criterion for the spinal learning process. To that end, we utilise the proposed set invariance framework to precisely control how much freedom is granted to the user, as the better the motricity of the patient is, the more he or she can be relied on to execute a stable walking pattern. First, we choose joints that we want to let the user control: the assisted joints. All the other joints will be rigidly controlled. In this work, we choose to only assist the sagittal hip and sagittal knee of the swing leg (cf. Fig. 3.18).

The architecture of the variable assistance framework, as shown in Fig. 3.19, contains four main components. First, a nominal gait is obtained from a neural network based library built from PHZD trajectories (cf. [2, 3, 36, 39, 40, 86]). This trajectory is modulated by a deadbeat mechanism. This deadbeat mechanism is critical in this case because the nominal joint trajectory will not be followed very accurately when the user is in control of the assisted joints.

The filtered trajectory $q_{des}(\cdot)$ is then fed into two separate controllers. One is the **baseline controller** that plays back the trajectory and generates position and velocity targets $q_{des}(t - t_i)$ and $q'_{des}(t - t_i)$ for the PID controllers that in turn generate tracking torques $u_t(t)$. The flatfoot ankle controller separately computes targets for the swing leg ankle that are then substituted in place of the nominal ones.

The other controller is the **variable assistance controller**. This controller is the heart of this variable assistance approach and leverages the proposed controlled invariance framework. The variable assistance controller has three subcomponents: joint idealization, feedforward assistance, and virtual guide filter.

The **joint idealization** component computes the torques required to compensate for gravity and friction in the assisted joints. The goal is to make these joints as transparent as possible such that when there is no assistance, the user does not feel any resistance that would impede his ability to walk freely. This joint idealization component is, however, not sufficient to make the exoskeleton fully transparent as the inertia of the exoskeleton is not compensated for, which makes the user's legs harder to move. The **feedforward assistance** component therefore provides feedforward torques $u_f(t)$ – calculated during the PHZD gait generation process [36] – to obtain a first-order level of compensation for the inertia of the assisted joints. This does not truly compensate for inertia, but at least provides enough assistance for the user to move the exoskeleton legs along the desired trajectory. The intensity of both idealization and feedforward components can be adjusted to produce varying levels of user effort.

The **virtual guide filter** computes the joint torques $u_v(t)$ required to limit the discrepancy between the actual and desired trajectory of the assisted joints. To that end, we will explore two approaches. First, the discrepancy limit is described by a tube around the desired trajectory: a virtual guide. The shapes and sizes of the virtual guides can be chosen almost arbitrarily. In a second time, the discrepancy will be characterised by the position of the swing foot with respect to a nominal trajectory.

Finally, an impact detection block also records which leg of the exoskeleton is in stance or swing, and generates an "assisted joints selection matrix" that controls which joints are being assisted at a given instant. Only these joints are assigned the assistive torques. The remaining joints are assigned the baseline tracking torques. The merging of these torques comprises the final joint torques $u(t)$ that are commanded to the exoskeleton.

3.6.3.4 Joint-Based Virtual Guide Filter

Formulation. In this first approach, each joint is idealized so that it can be handled independently of the rest of the system. We therefore consider the following dynamics for each joint:

$$J\ddot{q} = u_v + u_f(t - t_i) + u_{ext}, \tag{3.47}$$

where J is the inertia at the joint, u_v is the torque the virtual guide filter can apply, $u_f(t)$ the feedfoward torque applied to the joint, and u_{ext} the torque applied by the exoskeleton user on the joint. The state of the system is therefore $x = [q, \dot{q}]^\top$.

The virtual guide $\overline{S}$ we want to constrain the joint to stay in is characterized by:

$$h(t, x) = 1 - \left(\frac{q_{des}(t - t_i) - q(t)}{q_{bound}(t - t_i)}\right)^2 \tag{3.48}$$

for some properly chosen q_{bound} to achieve the desired shape of the guide (cf. Fig. 3.20 for examples of shapes). Note that this is a time varying set, so the propositions of Sec. 3.5 have to be modified as done in [30].

Because u_{ext} is not known ahead of time, a robust version of the method presented in 3.5 has to be used. The key is that system (3.47) is monotone [7]. In this case, the safe backward image is characterized by:

$$h_T^{\Omega}(t,x) = \min_{\substack{\tau\in[0,T-t] \\ u_{ext}\in\left\{u_{ext}^{min},\ u_{ext}^{max}\right\}}} h \circ \phi_{\tau}^{u_b,u_{ext}}(q)\,, \tag{3.49}$$

where u_{ext}^{min} and u_{ext}^{max} are the extreme values of the disturbance the user can generate. So in order to evaluate $h_T^{\Omega}(t,q)$, the numerical integration of the dynamics only has to be performed twice each time assuming the extremal values of the disturbance. The backup policy is chosen to be:

$$u_b(t,x) = K_p(q_{des}(t-t_i) - q) + K_d(\dot{q}_{des}(t-t_i) - \dot{q}) \tag{3.50}$$

for some properly chosen gains K_p and K_d. For this work, these gains were chosen to be the same as the one used for the PIDs of the baseline controller.

Finally, the filtering law is given by:

$$u_v(t,x) = (\lambda(t,q) + (1-\lambda(t,q))\,\lambda_d(t,q))\,u_b(t,x), \tag{3.51}$$

where $\lambda(t,q) = \left(1 - h_T^{\Omega}(t,q)\right)^3$ and $\lambda_d(t,q) = \zeta\frac{dh_T^{\Omega}(t,q(t))}{dt}$ for some derivative gain ζ. The usage of this derivative term helps dampen the behavior of the safety filter.

Experiments. The validation experiments were performed on the empty exoskeleton as it hung in the air in an effort to show the behavior of the filter without user perturbations and without feedforward torque. The plots of the experimental results, shown in Fig. 3.20, illustrate the actual joint angles over 30 steps with each step overlaid on top of each other. It can be seen that for all tube shapes, the actual joint angles remained inside of the bounds and the filter only acts when necessary.

Then, the variable assistance framework was tested with a subject inside the exoskeleton and walking on a treadmill. The required assistive torque as well as the trajectory tracking are presented in Fig. 3.21. It can be observed that when the subject is passive under partial assistance, the joint trajectories tend to group near the virtual guides as expected. Alternatively, when the subject is active under partial assistance, the actual joint trajectories tend to span more of the virtual guide as the subject is actively trying to avoid hitting the bounds of the guide. In all cases, the trajectories stay contained within the virtual guides. For a video of these experiments please see [33].

3.6.3.5 Feature-Based Virtual Guide Filter

One of the issues with this first approach is that the subject has to follow the nominal trajectory which in practice is not particularly anthropomorphic. This makes it difficult for the subject to perform well. So to make the variable assistance more permissive while still ensuring coherent walking if the subject is not performing well, we propose to extend the previous approach to a feature-based constraint instead of a joint-based one. Here, we propose to constrain the trajectory of the swing foot to ensure a correct ground clearance and forward motion. This way, the user has more freedom on individual joints trajectories which allows him to perform more natural steps.

For this approach, the controller architecture remains the same as in Fig. 3.19. Only the formulation of the virtual guide filter is changed from the previous section.

Formulation. This time, we consider a simplified model of the swing leg as a whole. In particular, we model the swing leg as a double pendulum whose joints corresponds the sagittal hip and knee, the ankle joints being assumed locked (cf. Fig. 3.22). The state of the system is now 4-dimensional: $x = [q, \dot{q}]^\top$ with $q = [q_1, q_2]^\top$, and both hip and knee joint torques are being filtered concurrently. For simplicity, the torso pitch angle q_0 is fixed at the desired angle chosen to generate the nominal trajectory. The dynamics can be found using classical Euler-Lagrange formalism and is of the form:

$$M(q)\ddot{q} = u - C(q, \dot{q}) - g(q), \tag{3.52}$$

where $u = [u_1, u_2]^\top$ is the vector of joint torques.

The virtual guide is now characterized by the forward position of the swing foot **with respect to the nominal trajectory** (cf. Fig. 3.22):

$$h(t, x) = p_x\left(q(t)\right) - p_x\left(q_{des}(t - t_i)\right) + \epsilon_x. \tag{3.53}$$

In other words, the swing foot is constrained to move forward at least as fast as it does in the nominal trajectory.

The backup policy is chosen to be the same as for the joint-based virtual guides—PID tracking of the nominal trajectory—and similarly with the filtering law. The constant ϵ_x is chosen so as to allow the backup law to steer the system away from the boundary of the virtual guide.

Experiments. The validation experiments were performed on the empty exoskeleton as it hung in the air. The plots of the experimental results, shown in Fig. 3.23, illustrate the foot position and constraints over several steps. It can be seen that in the absence of user effort, the proposed formulation results in the actual foot trajectory following closely the nominal one.

Then, this approach was tested with a subject inside the exoskeleton and walking on a treadmill. The results are presented in Fig. 3.24 and 3.25. It can be observed that in this case, the subject can freely execute a gait with longer step length if he desires (cf. Fig. 3.25), while still guaranteeing that a minimum step stride is respected if the subject is not able to perform longer steps (cf. Fig. 3.24).

3.6.4 Safe Exploration of Unknown Environments with a Quadcopter UAV

Finally, we apply this controlled set invariance framework to the problem of safe exploration of an unknown environment (note that more information about this work can be found in [78]). This task is particularly relevant for drones whose usage is becoming prevalent for tasks such as autonomous deliveries, aerial surveillance, or disaster relief. Most of these missions involve navigating through unknown or uncertain environments. Due to the altitude of the vehicles and their often exposed propellers, collisions are catastrophic for the drone and might also be dangerous for its surroundings. For this reason, collision avoidance techniques are crucial to further the use of these systems in everyday life.

In typical drone flight, collision avoidance is the topic of navigating safely through an environment. This usually translates into creating and tracking trajectories that take the drone through the surrounding free space and that avoid occupied or uncertain space. While this approach to collision avoidance can be effective in practice, as evidenced in [44, 55, 56], its computational complexity necessitates simplified abstractions of the model and obstacles. Couple to that the inherent uncertainty associated with mapping an environment and it is easy to see why such approaches are typically conservative, which can lead to slow mobility, while still lacking guarantees of collision-free tracking of the trajectories.

The authors therefore believe that trajectory planning is not the most effective layer in which to enforce safety. Planner updates are too infrequent, and there is too much uncertainty stemming from the aforementioned hurdles to be able to provide rigorous guarantees of safety with such a method. Instead, we propose an approach leveraging the proposed set invariance framework to ensure collision avoidance enforced at a control level while relying solely on local sensing information (i.e. no mapping is required). This approach can be applied in conjunction with any planning algorithm (or even a human operator), which means that it allows for planning algorithms that are more aggressive, since they do not need to guarantee collision avoidance or dynamic feasibility.

The planner used for this work is designed to work in tandem with the Octomap mapping library [43] to represent the environment map. The path planning is a basic implementation of the A* algorithm that searches for a path to a nearby frontier cluster. The search algorithm runs directly on the octomap, which is possible via implementation of algorithms that enumerate neighbors in a 3D octree [73]. Special heuristics encourages the planner to visit large clusters that are close to the current position of the drone. After a global plan is achieved, the local planner creates a spline and continuously updates a target point on this spline in front of the UAV. The position of this target point in the UAV frame is then used as the desired velocity V_{des} for the performance filter (cf. Fig. 3.27).

3.6.4.1 Collision Avoidance Framework

The goal of this Collision Avoidance Framework is to allow the vehicle to safely navigate around any unknown environment. As discussed above, we want to only rely on local sensing information, i.e. no mapping will be performed. It is therefore fundamental that the sensing method used allows full 360 degrees coverage of the vehicle surroundings. The goal is to define the safety set as an envelope around the UAV in which we know there is no obstacles.

For this scenario, the environment is assumed to be static, and we will use a point-cloud representation of the environment. We will assume that this point-cloud is obtained from a ray-based method of sensing such that all segments between the vehicle and the points of the point-clouds corresponds to unoccupied space (i.e. does not go through any obstacles). Points therefore correspond to the boundary between free space and either physical obstacles or unknown space (cf. Fig. 3.31). The safety set will hence be described as in (2.24) with:

$$h_i = \|p - p_i\|^2 - \Delta_h^2, \tag{3.54}$$

where p is the position of the UAV in cartesian space, p_i are the points of the point-cloud, and Δ_h a *hard margin* introduced to account for the size of the vehicle (cf. Fig. 3.28).

Most ray-tracing sensors available to date do not enjoy the same high update rate as the sensors necessary for low level control (i.e. IMU, visual odometry, etc...). Therefore, the position of the UAV only has to be tracked between each environment sensor update. This is one of the main advantages of the proposed framework over the ones requiring global planning and positioning. Only local positioning over a duration corresponding to the update period of the environment sensor is required.

Similarly to Sec. 3.6.2, the safety set is continuously changing as the environment gets explored. However, contrary to Sec. 3.6.2, the time dependency of the safety set is not known ahead of time and so the safety set does not necessarily always grow inside the state space (cf. Fig. 3.31). Furthermore, it is not trivial to find an efficient backup policy for all the possible shapes of safety set that can arise during exploration. Finally, embedded systems for UAVs do not quite have the computational capabilities to run a QP-based safety filter as for Sec. 3.6.2, especially at the control rates necessary for such agile vehicles.

To tackle this problem, we will therefore take the same approach as in Sec. 3.6.3 and use the scalable safety filter of Sec. 3.5. The filtering law is chosen to be:

$$v_{safe}(x) = \lambda(x) v_{perf}(x) + (1 - \lambda(x))\, v_b(x), \tag{3.55}$$

with

$$\lambda(x) = 1 - e^{-3h_{S_T^{u_b}}(x)/(\Delta_s - \Delta_h)}, \tag{3.56}$$

where Δ_s is the *soft margin* (cf. Fig. 3.28).

A key difference with the work presented so far though is that the safety filter is placed **before** a velocity controller (cf. Fig. 3.27). This velocity controller closes the loop around desired velocities in the world frame. In this work, it is a simple Velocity-Attitude-Rates cascade PID controller, but any controller that can track a desired velocity would work in this framework. This has two effects: first it makes the overall system more robust to model uncertainty. Secondly, it allows us to think about backup policies in a space that is simpler to grasp than the actuator space (in that case a 3D velocity space). Therefore, efficient backup policies can be constructed more easily.

The backup policy chosen for this application is inspired by previous work on a geofencing for civilian UAVs [35]. The idea is to slow down to a halt while steering the vehicle away when getting close to obstacles. This can be achieved by commanding $V_{bak} = 0$ unless the vehicle is between the soft and hard margins, in which case the V_{bak} vector is pointed away from the nearest obstacles as depicted in Fig. 3.29.

The effect of that backup policy in conjunction with the filtering law is depicted in Fig. 3.28. However, even though this approach guarantees safety of the system, it yields poor performances in some circumstances. In particular, it does not allow the system to *smoothly glide* along obstacles as the backup policy is intermittently switched to and from, leading to very oscillatory behaviors. Therefore, in order to increase the overall performance of the system, another component is used to filter the velocity inputs commanded by the planner in order to minimize the interventions from the safety filter: the performance filter.

The base of this performance filter is the same as the one of the backup policy (cf. [35]). The desired velocity is altered in a way that slows down convergence to the obstacles when getting close to it, and incentivizes divergence when past the soft margin. However, the closest point considered for the filtering of V_{des} is not based on the distance between the current drone position and the obstacles, but the shortest distance between the drone position **along the backup trajectory** and the obstacles (cf. Fig. 3.30). Furthermore, the distance to this closest point is the shortest distance between the drone position along the backup trajectory and the obstacles. This way, the performance filter is able to better anticipate incoming obstacles which leads to less intrusions of the backup trajectory into the soft margin, which in turn leads to less interventions of the safety filter. In the end, these 3 components work together to provide a filter with minimal conservativeness and with guaranteed collision avoidance (cf. Fig. 3.27).

3.6.4.2 Recursive Feasibility

As discussed before, at each update of the environment sensors, a new safety set is redefined. This means that after an update, the system could end up outside of $S_T^{u_b}$ as illustrated in Fig. 3.31. To address this use, there are two approaches.

A first approach it to check for each update whether or not the system would end up inside of $S_T^{u_b}$ with this new safety set. If it does, then the update can be carried out, otherwise this new safety set is discarded and the current one continues to be used until the next update. This approach carries a non-negligible computational weight as the safety filter has to be ran twice when an update is not successful. One must also be careful because when safety set updates are skipped, the system only relies on localisation data whose drift can become substantial.

A second approach is to rely on the backup policy to bring the system to a stop if the system ends up outside of $S_T^{u_b}$ after a safety set update. Indeed, even though the system is outside of $S_T^{u_b}$, it is not before the update, which means that the backup policy can safely bring the system to a stop. As the slowing down occurs, the safety set can continue to be updated in hope to regain feasibility of the safety filter. In practice, feasibility is regained quickly and this is the approach we take for the following simulations.

3.6.4.3 Simulation

The simulation environment is a ROS-based C++ environment. The point cloud data is obtained from a modified Velodyne LIDAR sensor inside of the Gazebo simulator at a frequency of 10 hz. The simulation, including visualization in Gazebo and RVIZ, was able to run at a frequency of 300 Hz on a modern laptop computer.

The cave environment to explore was a large 240m by 460m structure with one entrance and one exit (cf. Fig. 3.32 and Fig. 3.26). The cave height is constant at roughly 3m, but the width is constantly changing, and gets as small as 0.5m with several protruding areas.

The quadrotor was able to explore the entire 240m by 460m cave in just under 28 minutes (cf. Fig. 3.32). The maximum allowable speed from the planner was 5 m/s, which the drone reached during open areas of the cave. The average desired speed sent from the planner was 4.09 m/s, and the average speed of the drone after the safety filter was 3.28 m/s.

A positive value of the barrier functions was maintained throughout, meaning the quadrotor never went closer than the minimum allowed distance from a point in the point cloud, which was set at $\Delta_h = 0.2m$ meters. For a video of these simulations please see [34].

3.6.4.4 Restricted Field of View

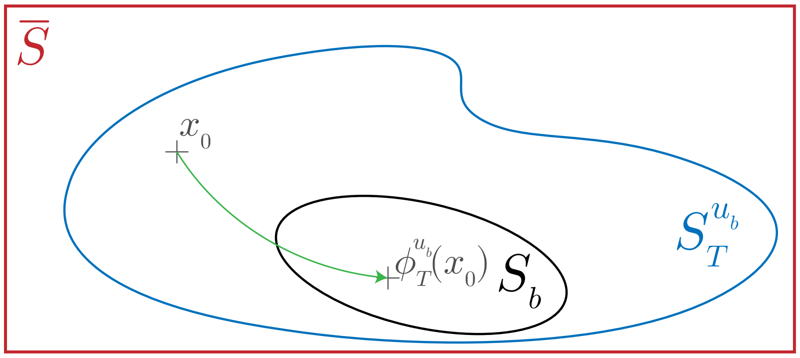
So far, we required that the vehicle be equipped with sensors that cover all of the UAV surroundings. Even though this is possible with available technology, such a sensing capability would be fairly expensive and would require a large enough vehicle to carry the sensors as well as an embedded computer capable of processing that much data. Therefore, we propose to extend the proposed approach for vehicle equipped with only partial sensing of the vehicle surrounding. This is for example the case of most UAVs that are equipped with a single forward-facing depth-sensing camera.

In that case, each individual sensor point-cloud is not sufficient to create a safety set that envelops the vehicle as illustrated in Fig. 3.33. It is therefore necessary to implement a mapping strategy to create a safety set envelope that the vehicle can evolve in. Note, however, that for the first iteration of the algorithm, the vehicle is not inside the safety set, so an assumption must be made that the immediate surroundings of the vehicle are safe for this 1st iteration (cf. Fig. 3.33). During subsequent iterations, mapping can be performed based only on new sensor data and a safety set can be progressively built as illustrated in Fig. 3.34.

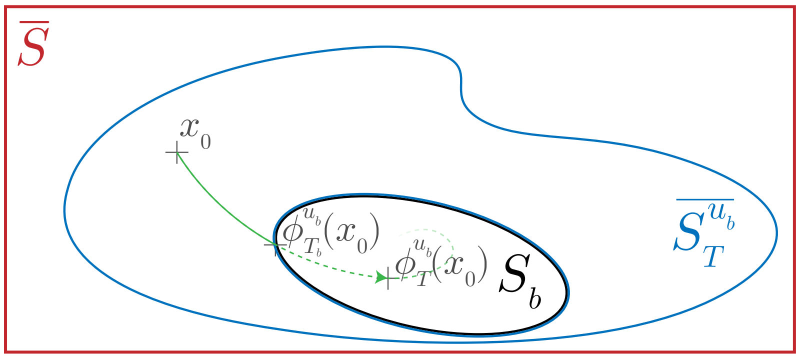
To implement this approach in a way that leverages the precision and efficiency of a point-cloud representation of the safety set, we propose an algorithm that combines point-cloud and voxel representations of the environment. A naive approach would be to just fusion the point-clouds given by the sensor, however this would only define the boundary between free and occupied space, but not between free and unknown space. One could therefore rely on a voxel-based representation of the environment, but this would come at the cost of conservatism on the location of the obstacle. One way to address this issues is therefore to combine both approaches.

Point-cloud data is therefore used to generate a voxel map of the environment, but point-cloud data is also conserved to refine the position of the frontier between free and occupied space (cf. Fig, 3.35). Successive point-clouds are merged together and the resulting point-cloud is down-sampled using the generated voxel map such that each voxel contains only one point. The position of this point is then associated with the corresponding voxel. A point-cloud of the safety set envelope is then generated by using the down-sampled point-cloud augmented by another point-cloud of the centers of all voxels on the frontier between free and unknown spaces. The rest of the safety filtering algorithm is the same as in the previous section.

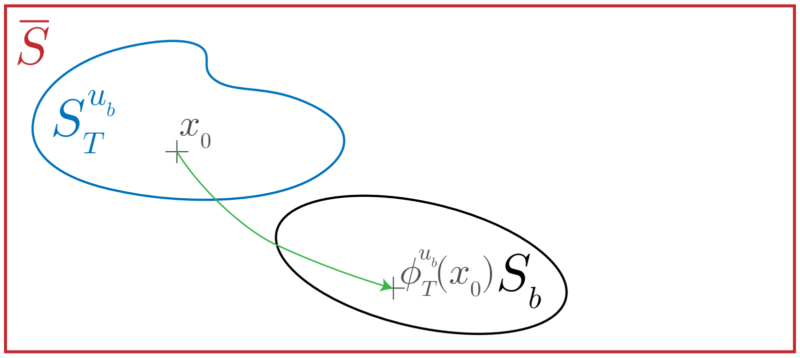
The simulated cave environment is explored again using an Intel Realsense as the only source of environmental data. This time, the yaw of the vehicle is controlled so as to point forward with respect to the path. As expected, the UAV is able to explore the cave safely, but does so more slowly overall.



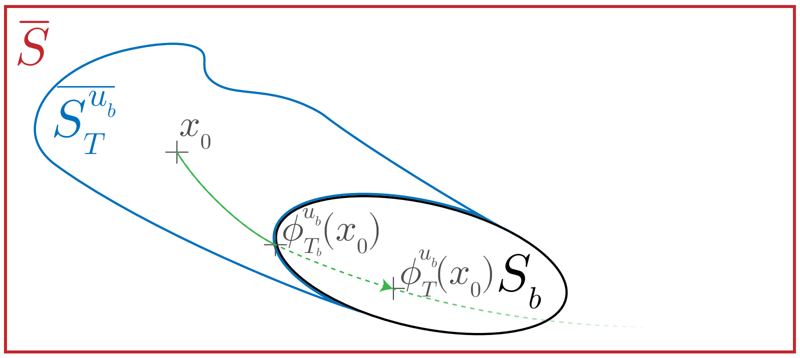
(a) The backup set is forward invariant under u_b.



(b) The backup set is forward invariant under u_b.



(c) The backup set is **not** forward invariant under u_b.



(d) The backup set is **not** forward invariant under u_b.

Figure 3.1: Safety Set in red, Backup Set in black, and Implicit Control Invariant set in blue. The backup trajectory under the backup control law u_b is in green.

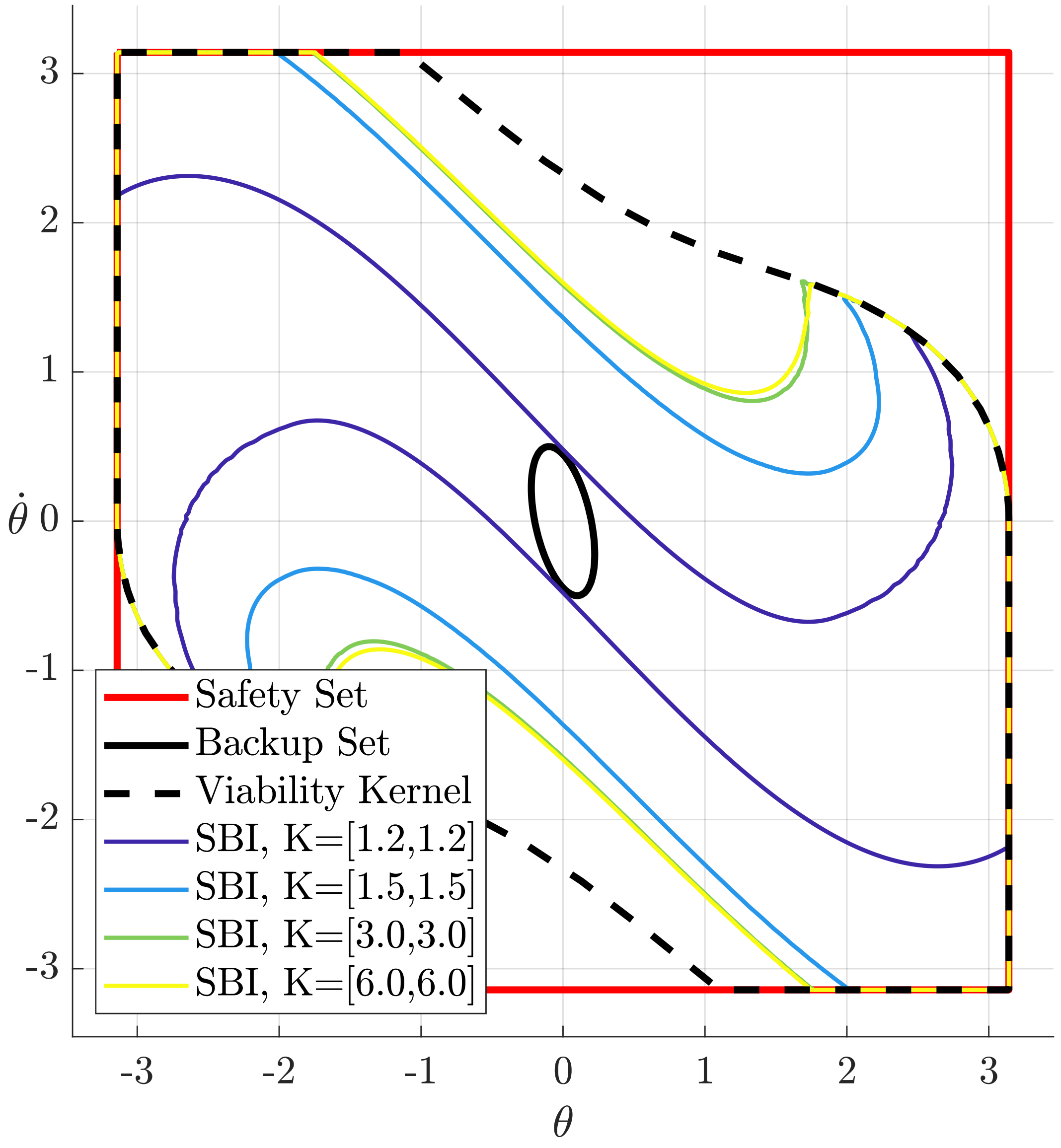


Figure 3.2: Plot of the SBI for different backup gains K with $T = 5$.

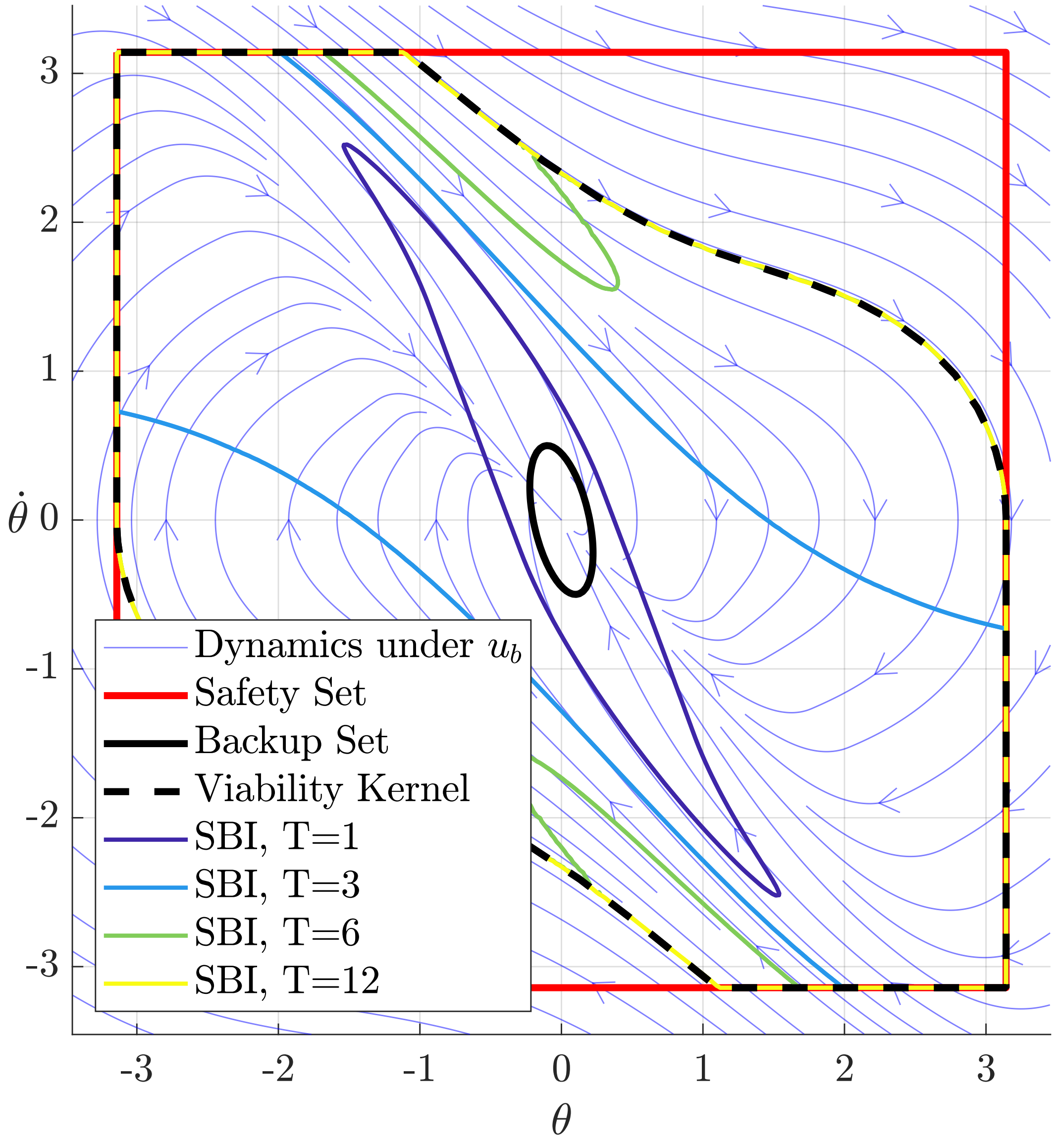


Figure 3.3: Plot of the SBI for different time horizons T.

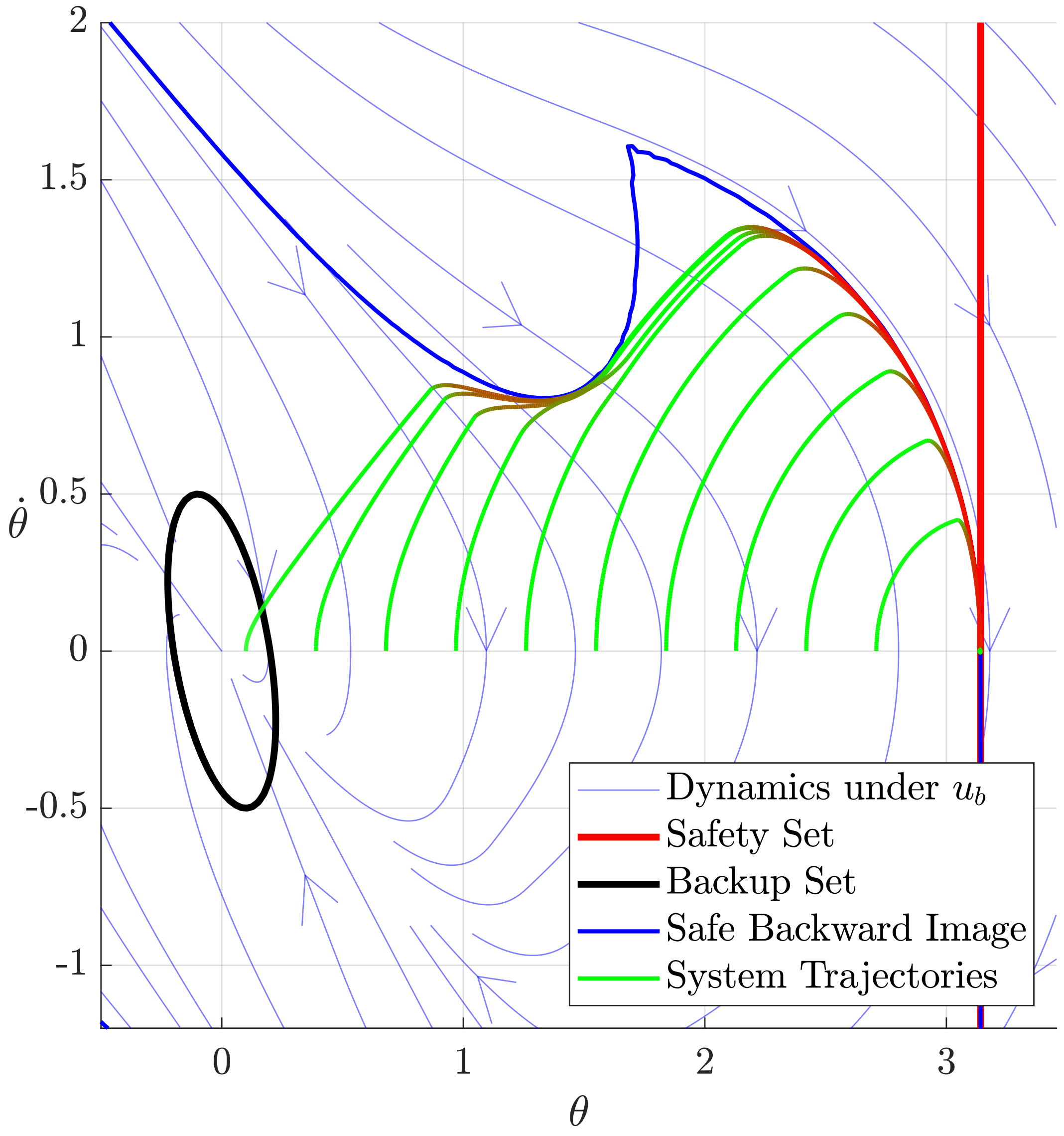


Figure 3.4: Trajectories of the system with $u_{des} = 0, T = 5$ and $K = [3, 3]$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

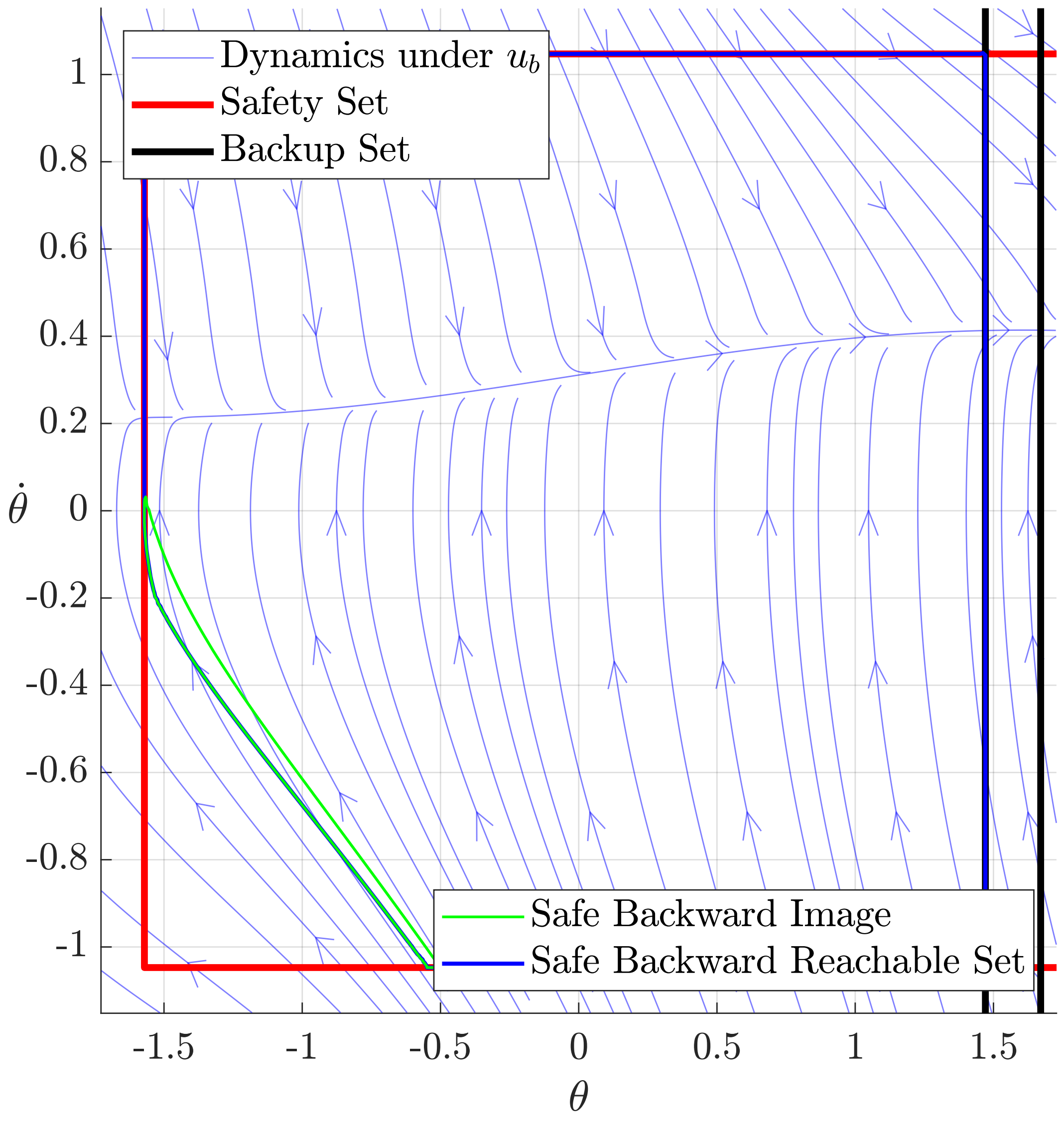


Figure 3.5: Comparison between safe backward image and safe backward reachable set for $T = 11$ and $u_b(x) = 10 * (\frac{\pi}{10} - \dot{\theta})$.

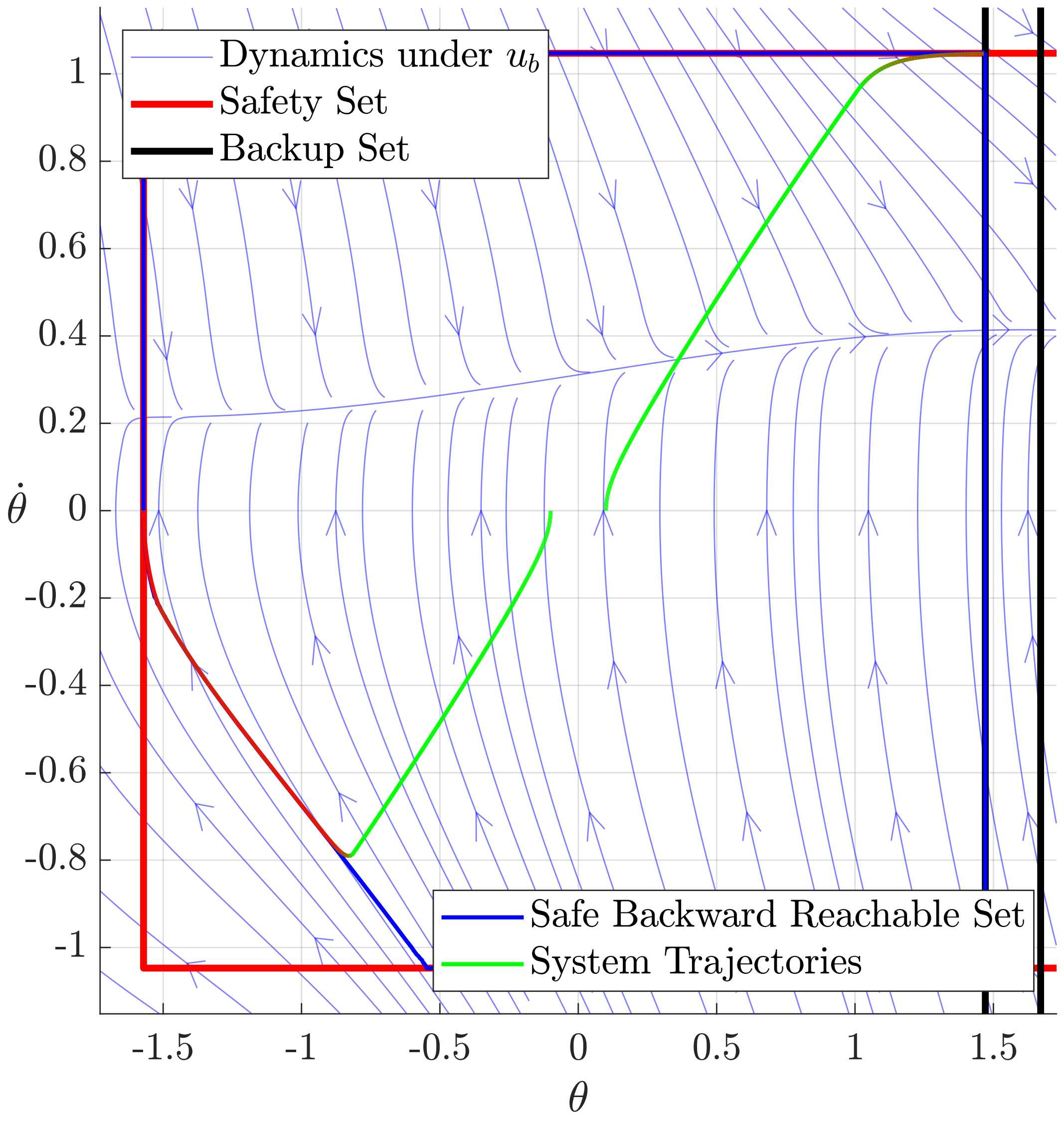


Figure 3.6: Trajectories of the system with $u_{des} = 0$, $T = 11$ and $u_b(x) = 10 * (\frac{\pi}{10} - \dot{\theta})$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

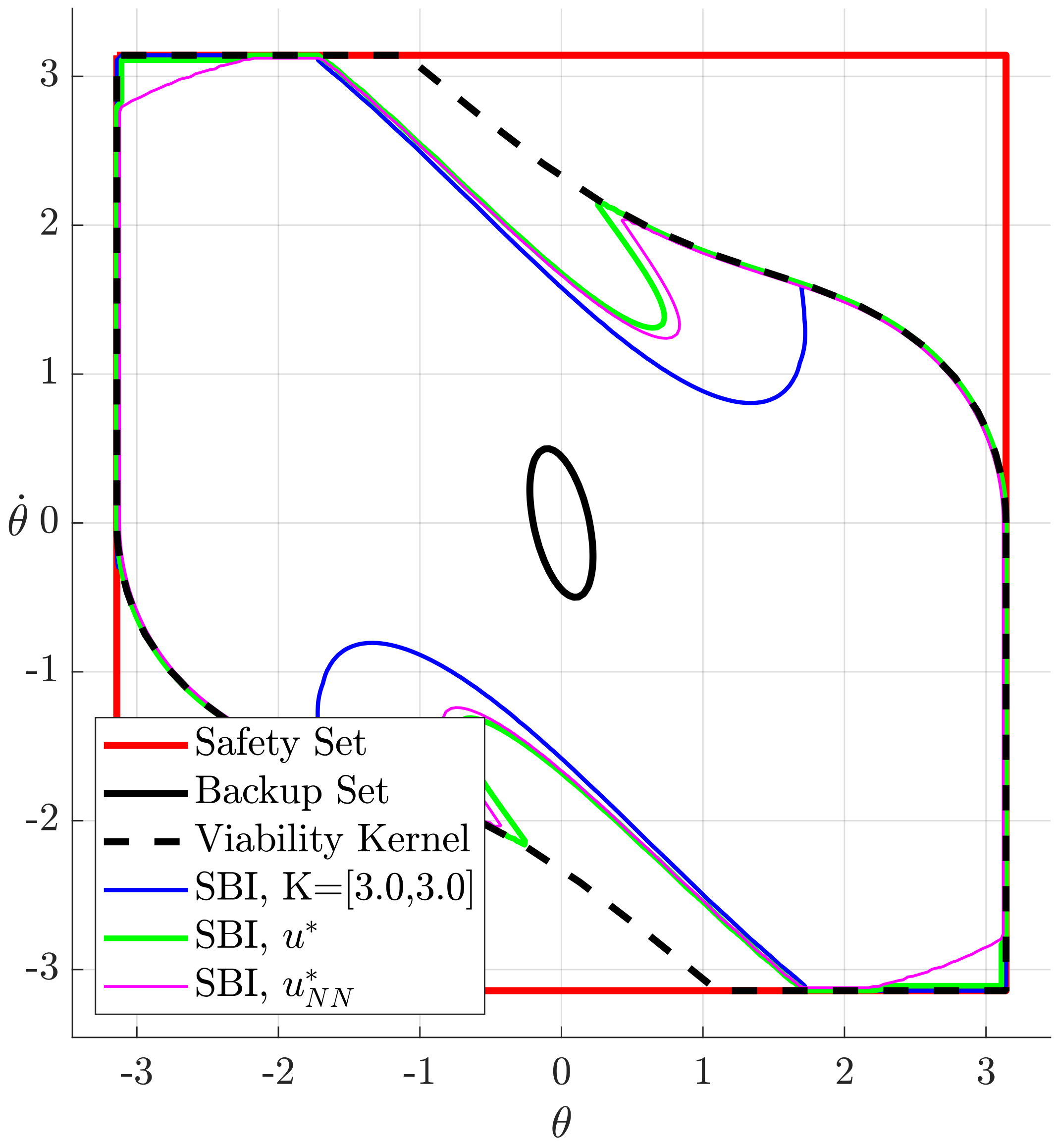


Figure 3.7: Comparison between the safe backward images of a linear backup control law (3.16), the optimal control law (3.27), and a Neural Network approximation of it, with $T = 5$.

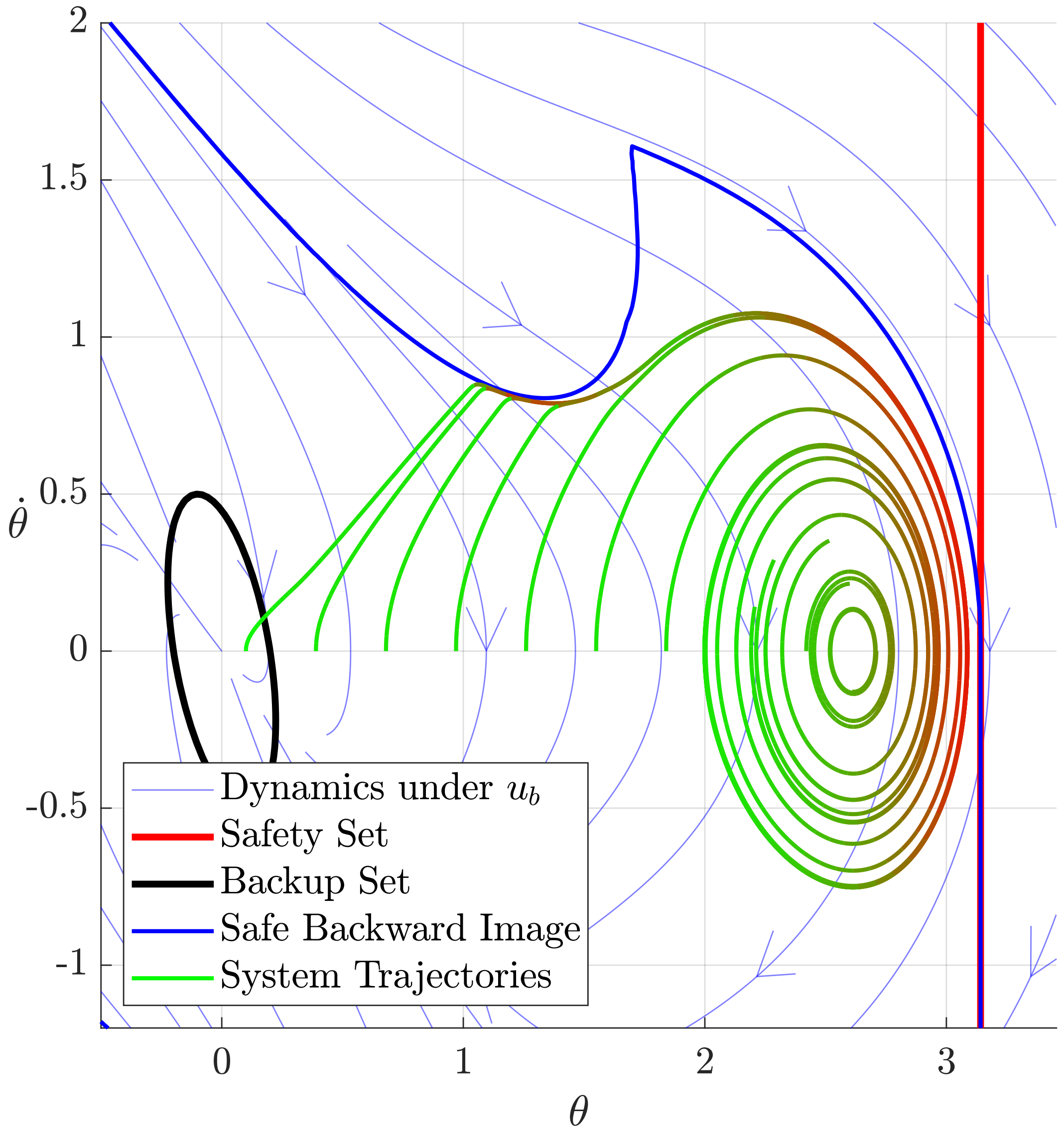


Figure 3.8: Trajectories of the system under a scalable implicit safety filter with $u_{des} = 0$, $T = 5$ and $K = [3, 3]$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

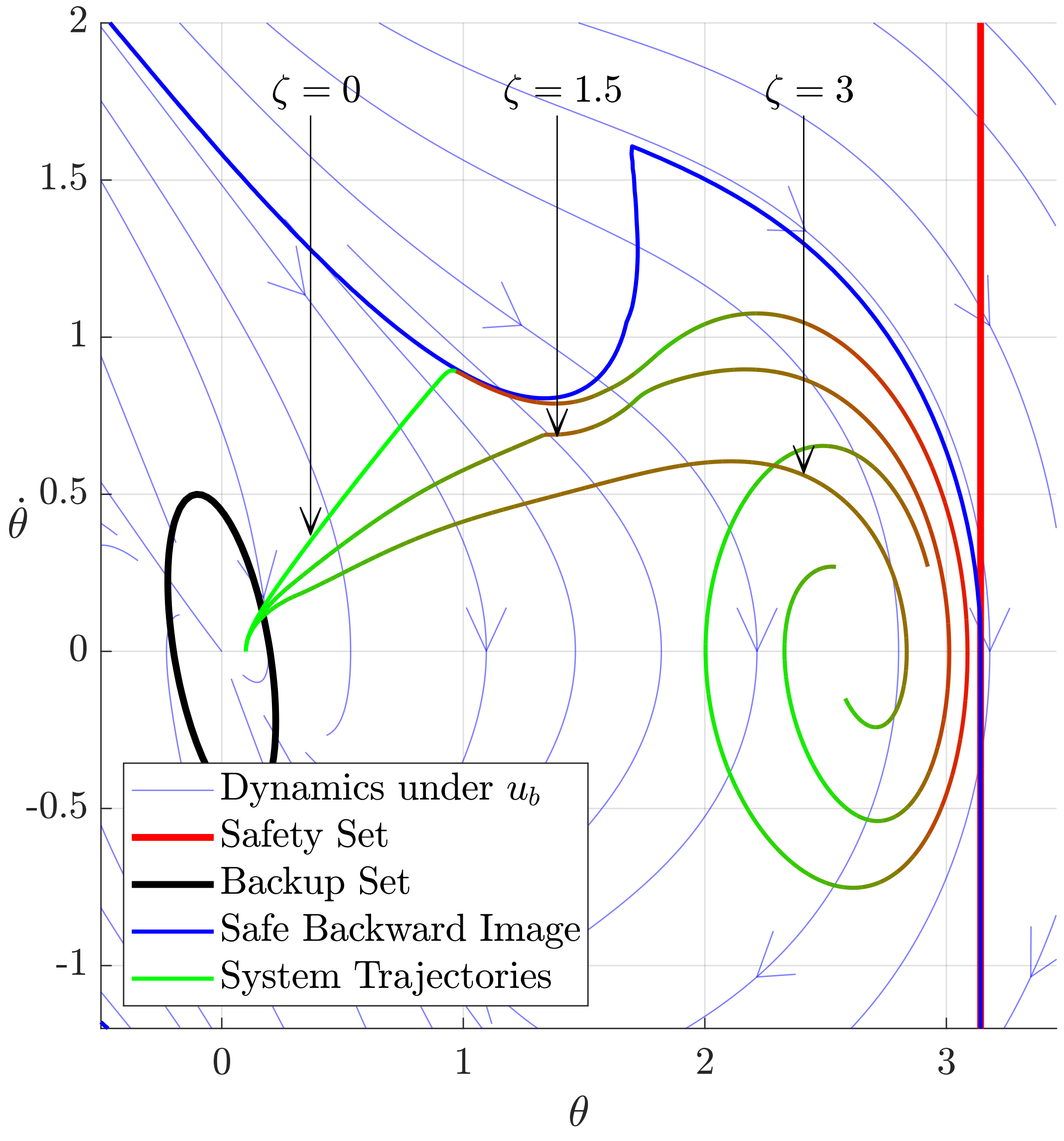
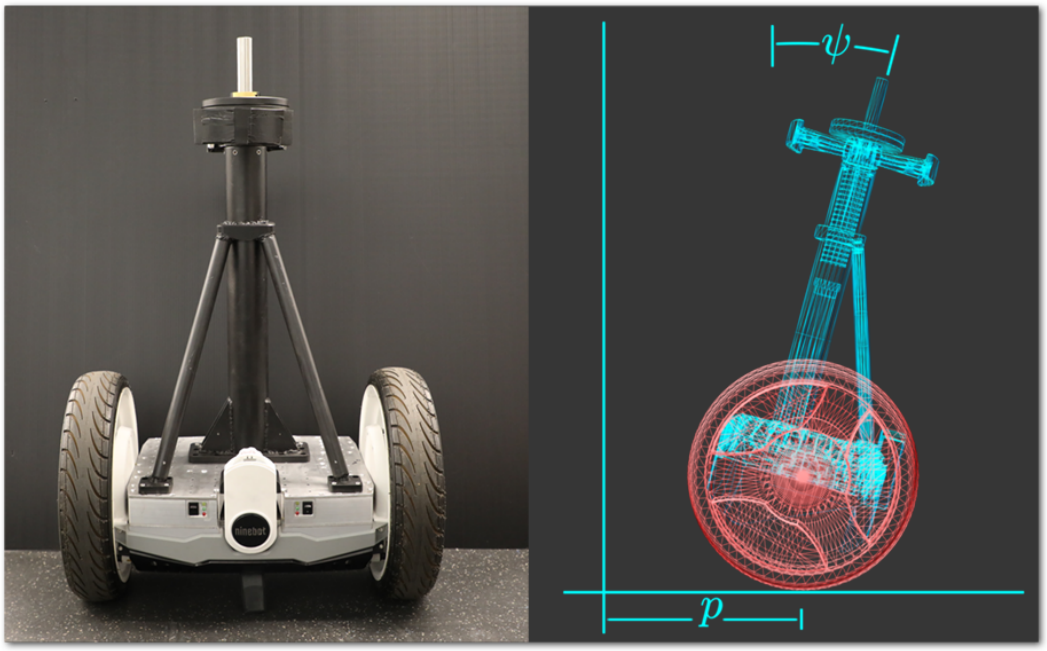
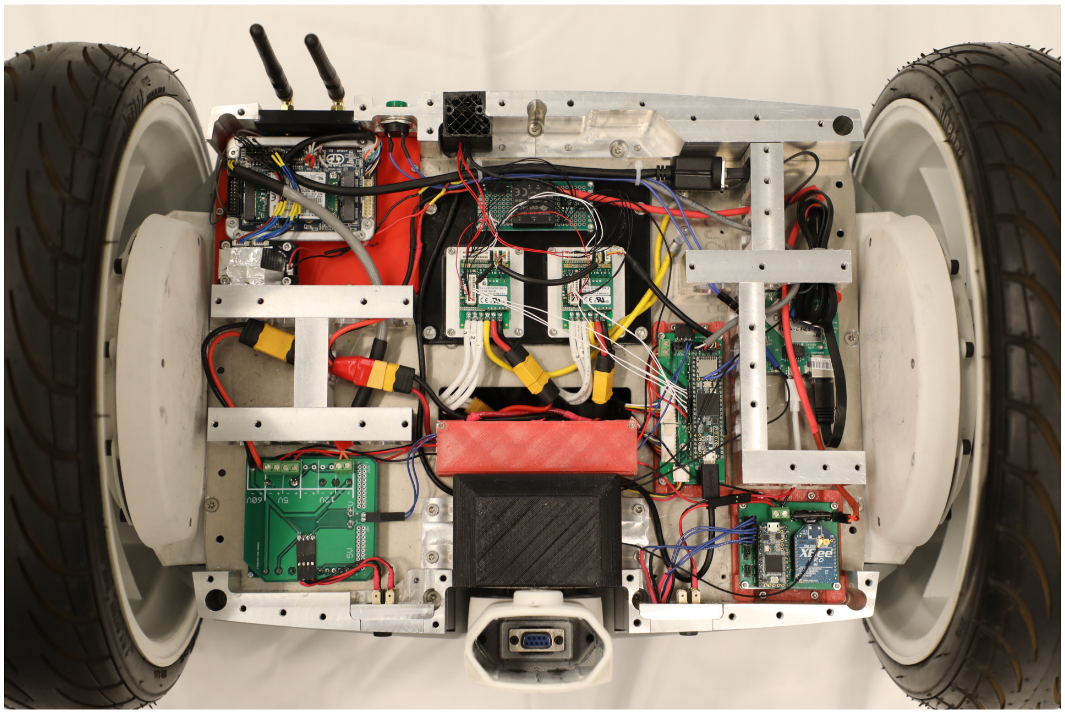


Figure 3.9: Trajectories of the system under a scalable implicit safety filter with $u_{des} = 0$, $T = 5$, $K = [3, 3]$ and different switching functions. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.



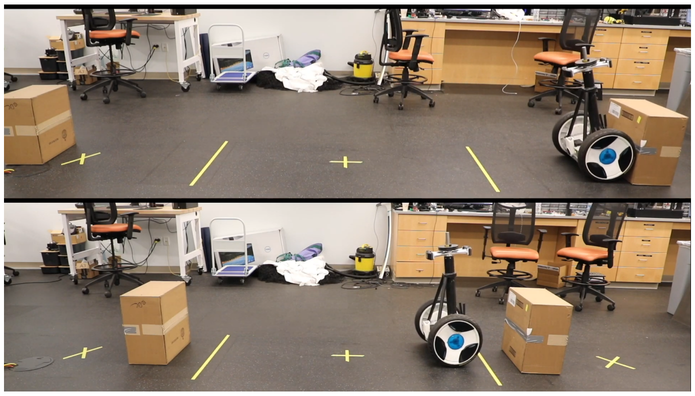
(a) General view of the hardware and parameterization of the state

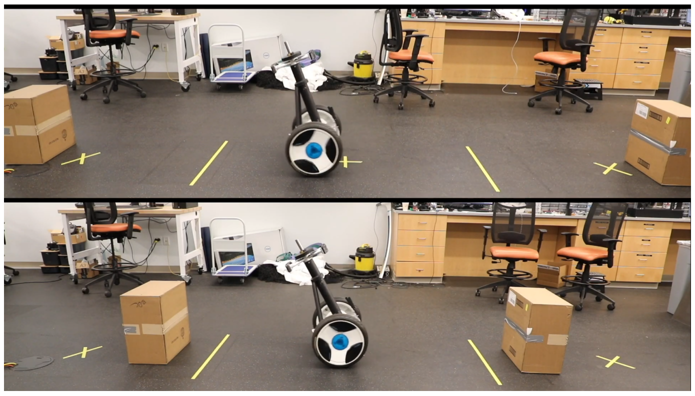


(b) Custom made electronics

Figure 3.10: Segway vehicle used for experiments.

(a) $t = 3$


(b) $t = 9$


(c) $t = 12.5$

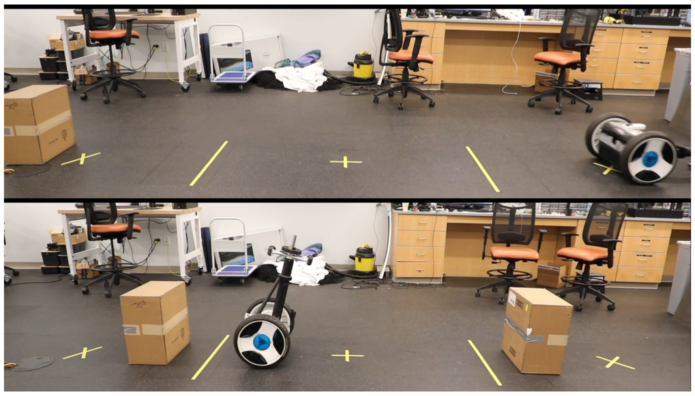

(d) $t = 15$

Figure 3.11: Pictures of the implicit filtering Segway experiment.

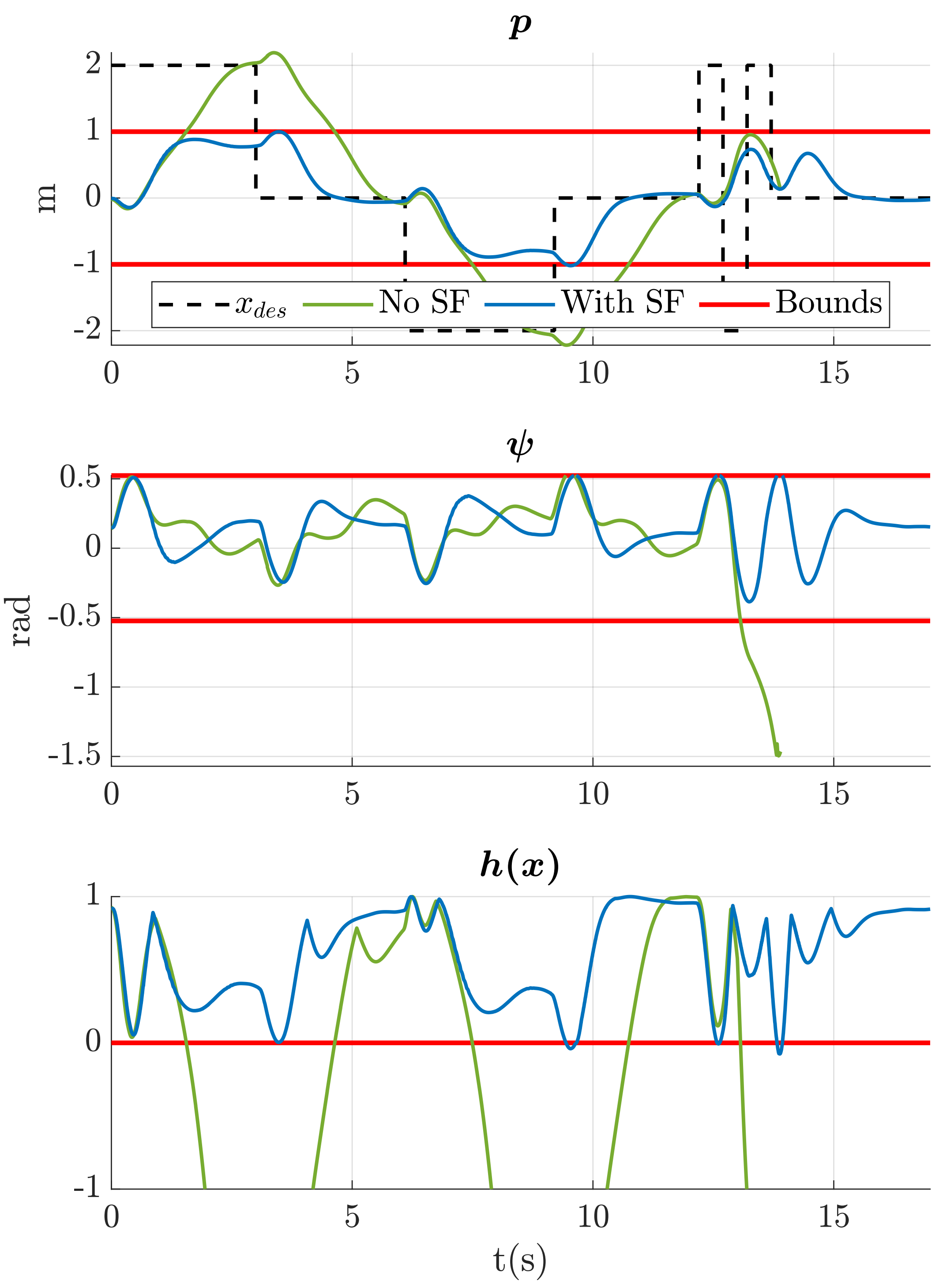


Figure 3.12: Results of the implicit filtering Segway experiments with and without safety filter. The nominal controller is an LQR driven by a desired position x_{des}.

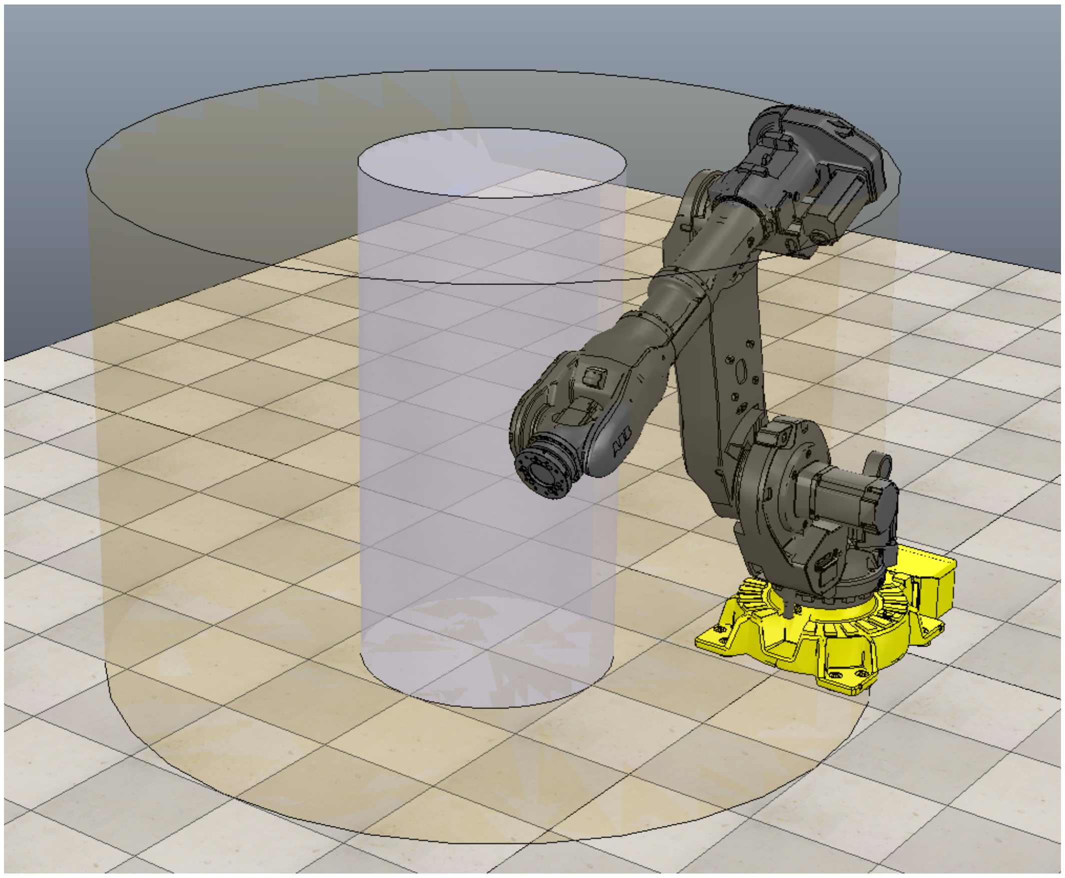


Figure 3.13: The IRB 6640 robotic arm along with the reachable sets for a human worker at $t = 0s$ in purple and $t = 1s$ in yellow.

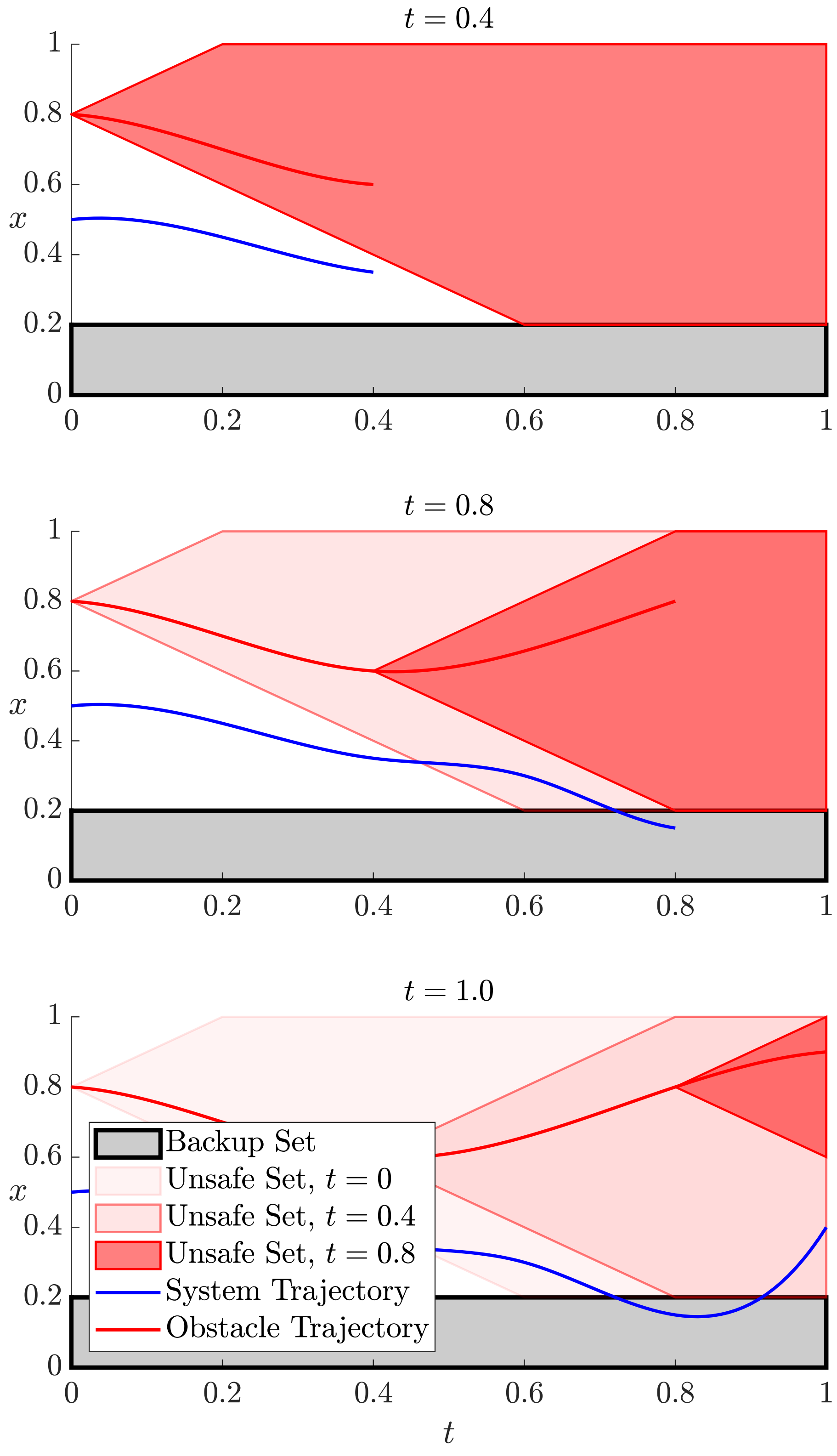


Figure 3.14: Illustration of time varying uncertain environment. The obstacle moves with time (red line) but its position is only measured at specific times (0.4,0.8,1.0). The safety set is the complement of the unsafe set, which is the forward reachable set of the obstacle minus the backup set. The system has to remain outside of the unsafe set, which shrinks for each obstacle measurement.

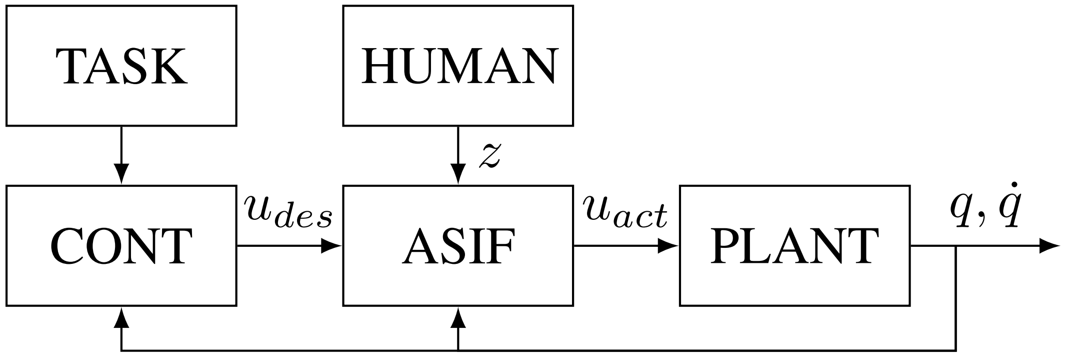


Figure 3.15: Block diagram of the ROS nodes used in the simulations.

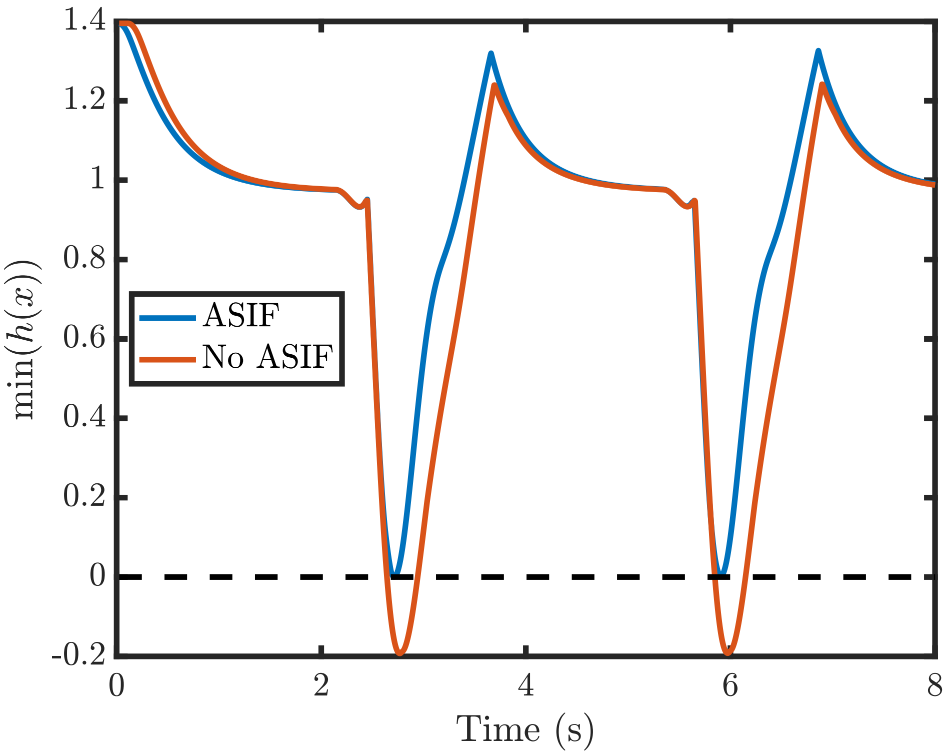


Figure 3.16: Value of the Barrier Function with and without ASIF engaged.

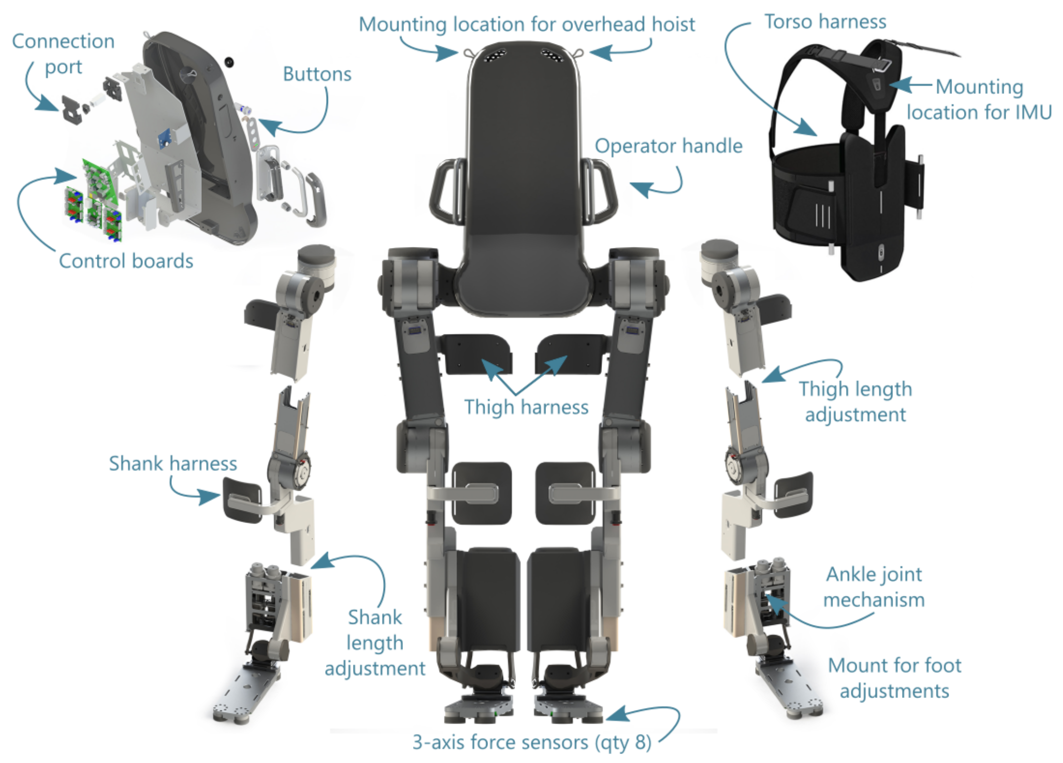


Figure 3.17: Exploded view of the Atalante exoskeleton.

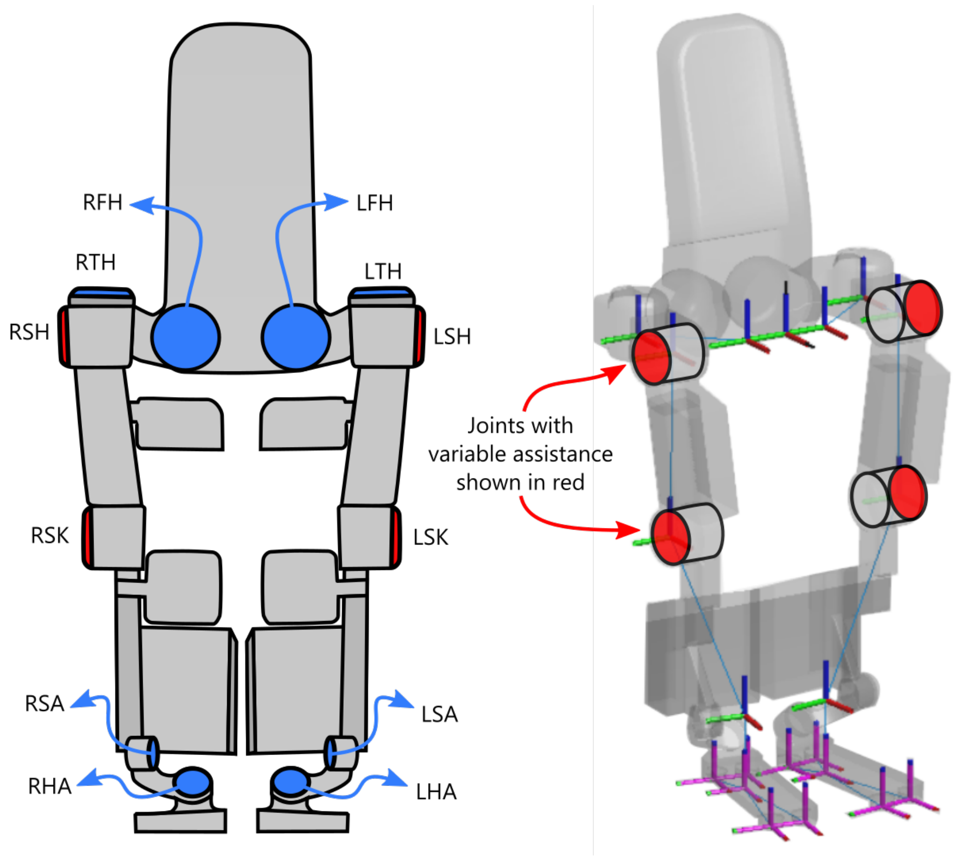


Figure 3.18: Schematic representation of the Atalante exoskeleton. In red are the joints that will be used for variable assistance.

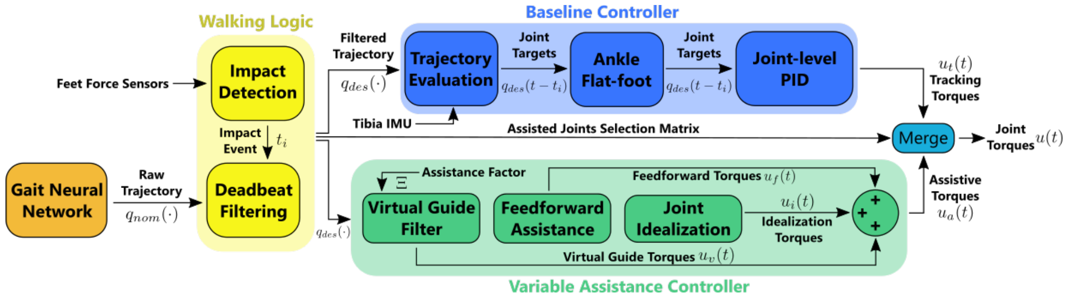


Figure 3.19: Architecture of the variable assistance framework.

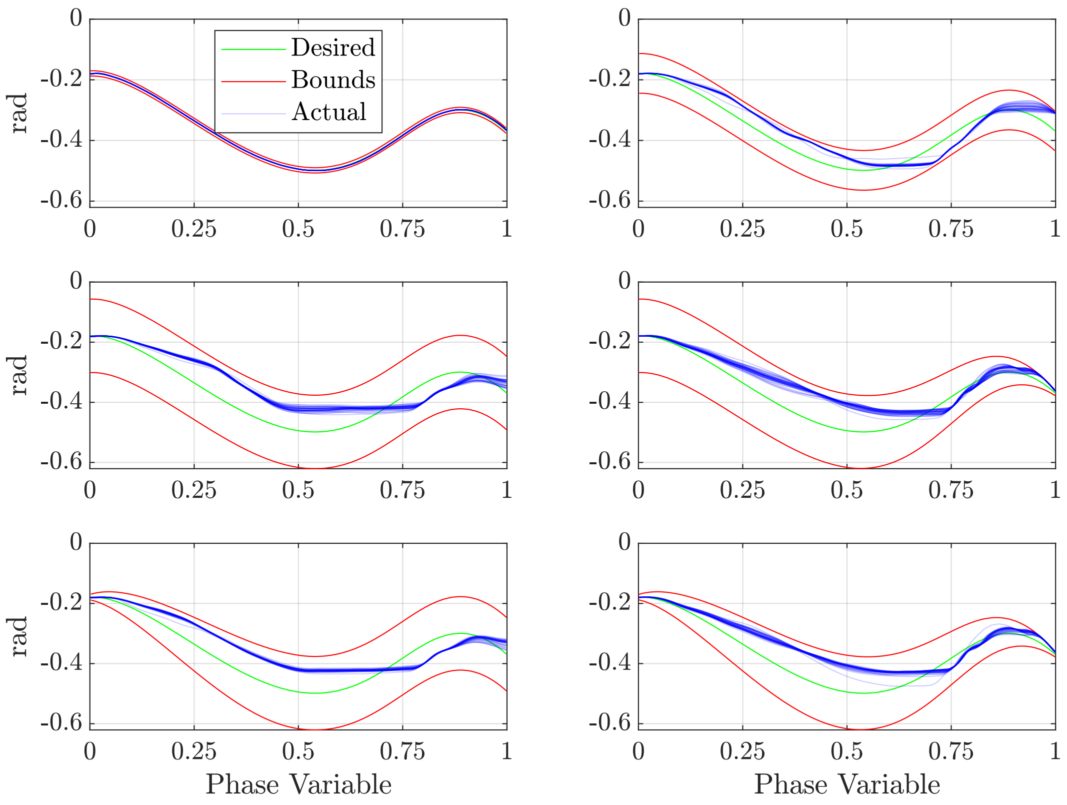


Figure 3.20: Left hip positions joint angles for the Exoskeleton empty and hanging in the air. The plots corresponds to 30 right steps and each subplot correspond to a different virtual guide shape.

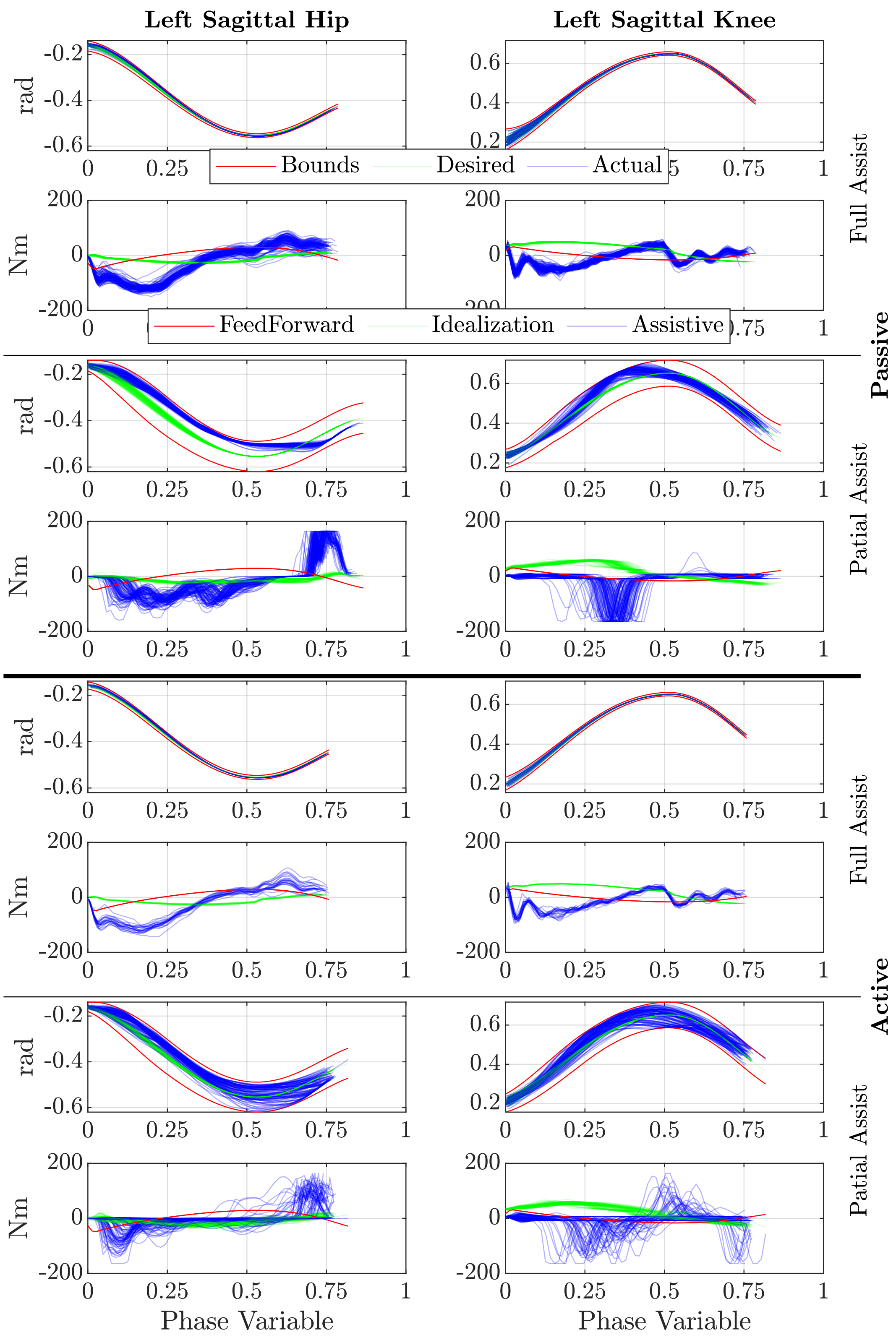


Figure 3.21: Comparison at the joint level between a subject actively trying to walk and being passive under both full and partial assistance of the exoskeleton. Because of early striking, most steps ended before the phase variable reached 1, unlike in Fig. 3.20 where the exoskeleton was in the air.

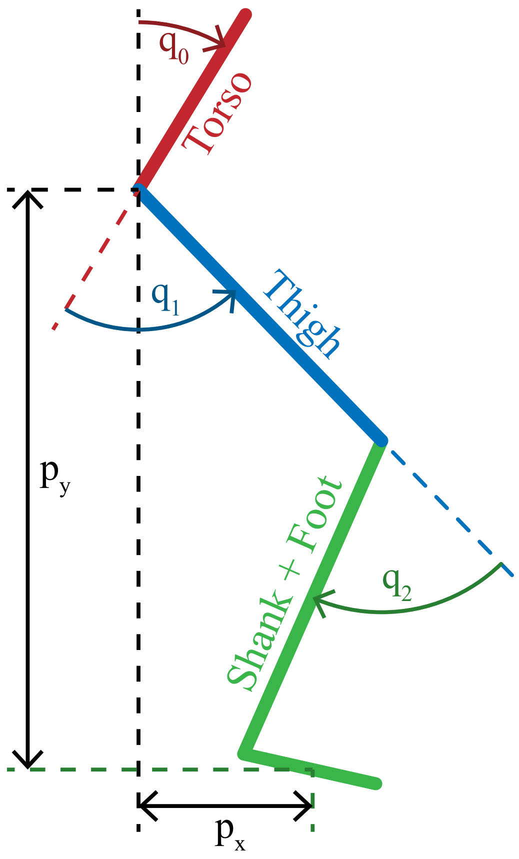


Figure 3.22: Simplified model of the exoskeleton and its swing leg.

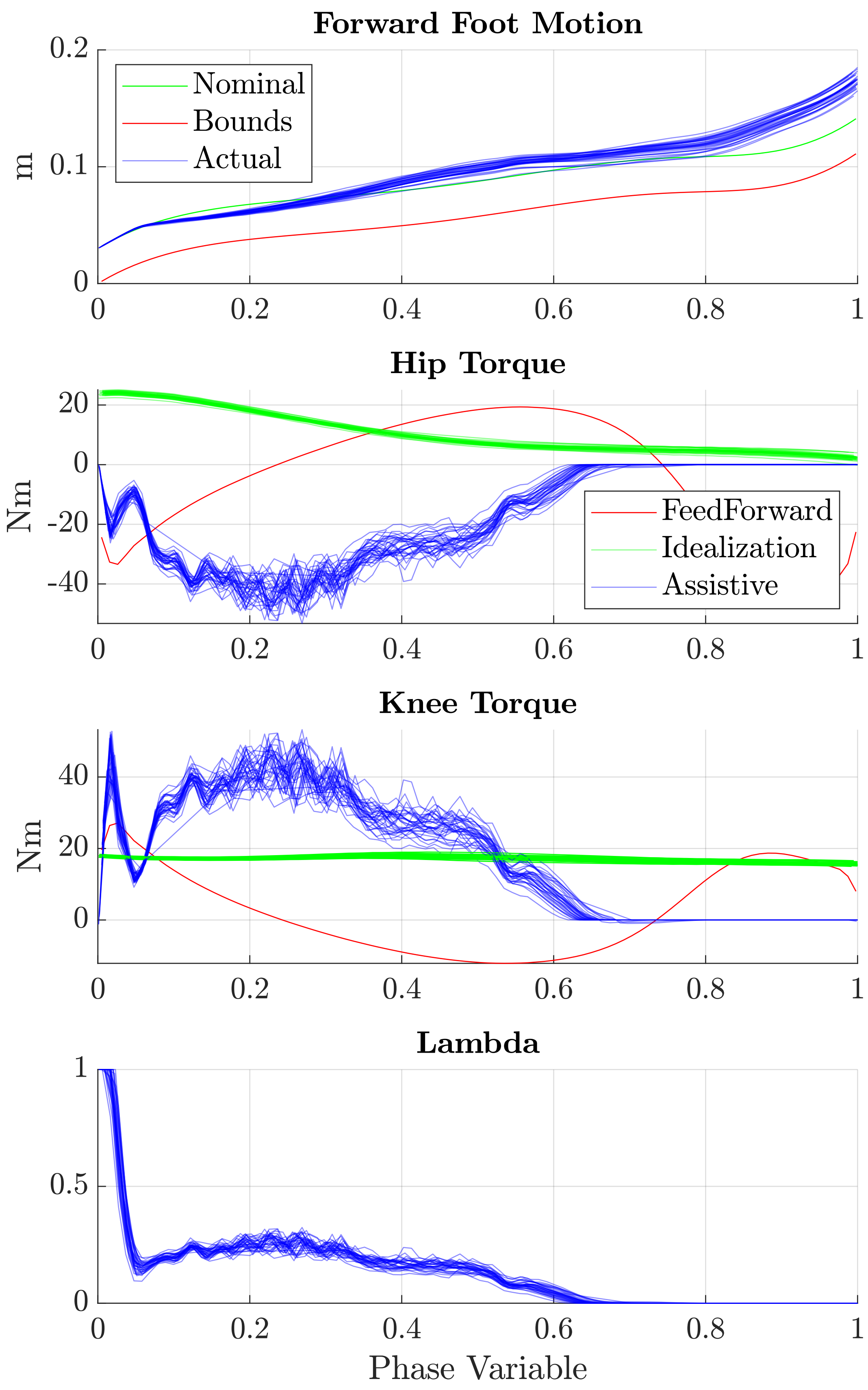


Figure 3.23: Foot level variable assistance for the Exoskeleton empty and hanging in the air.

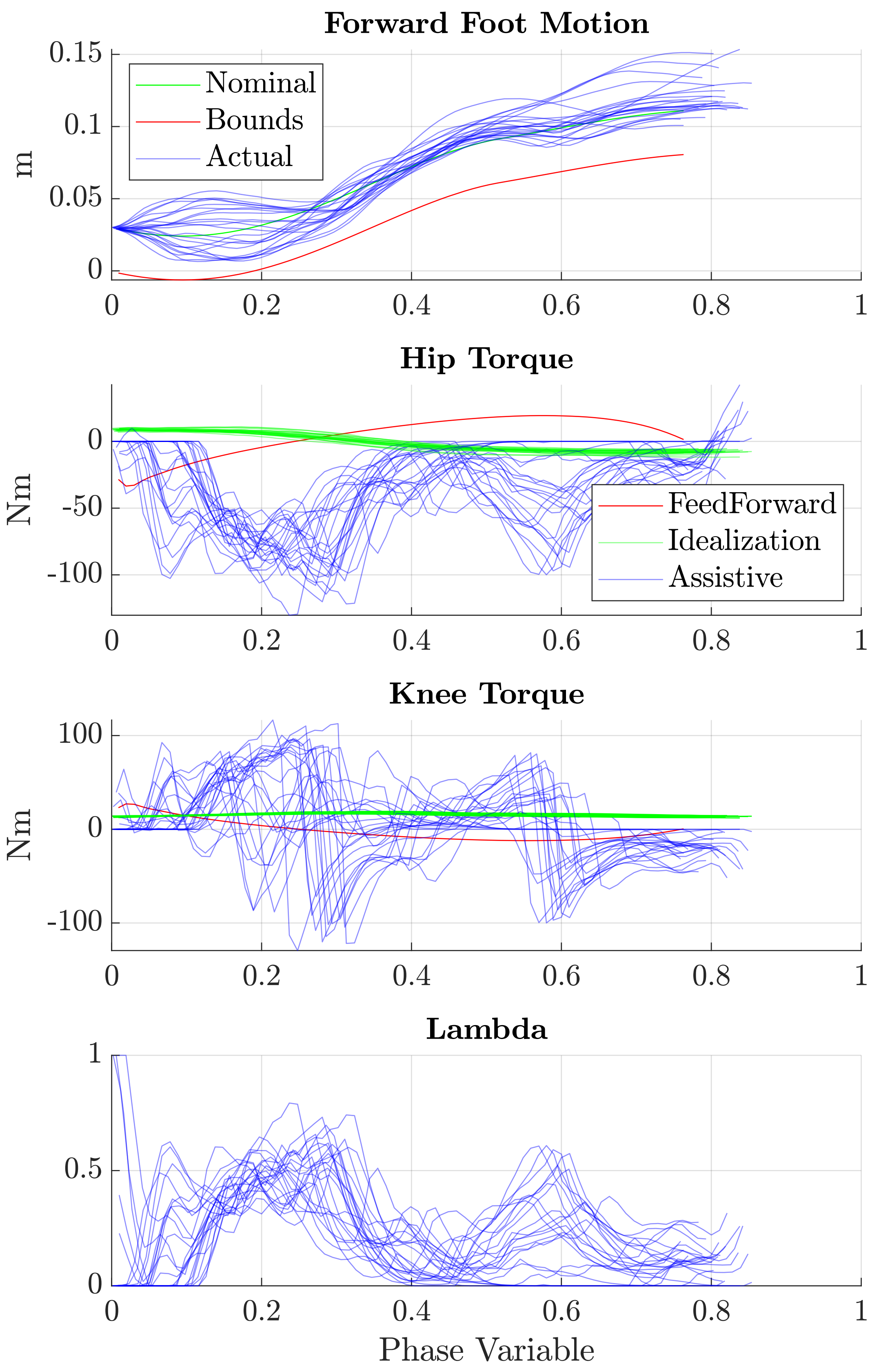


Figure 3.24: Foot Level Assistance for an able-bodied subject being passive.

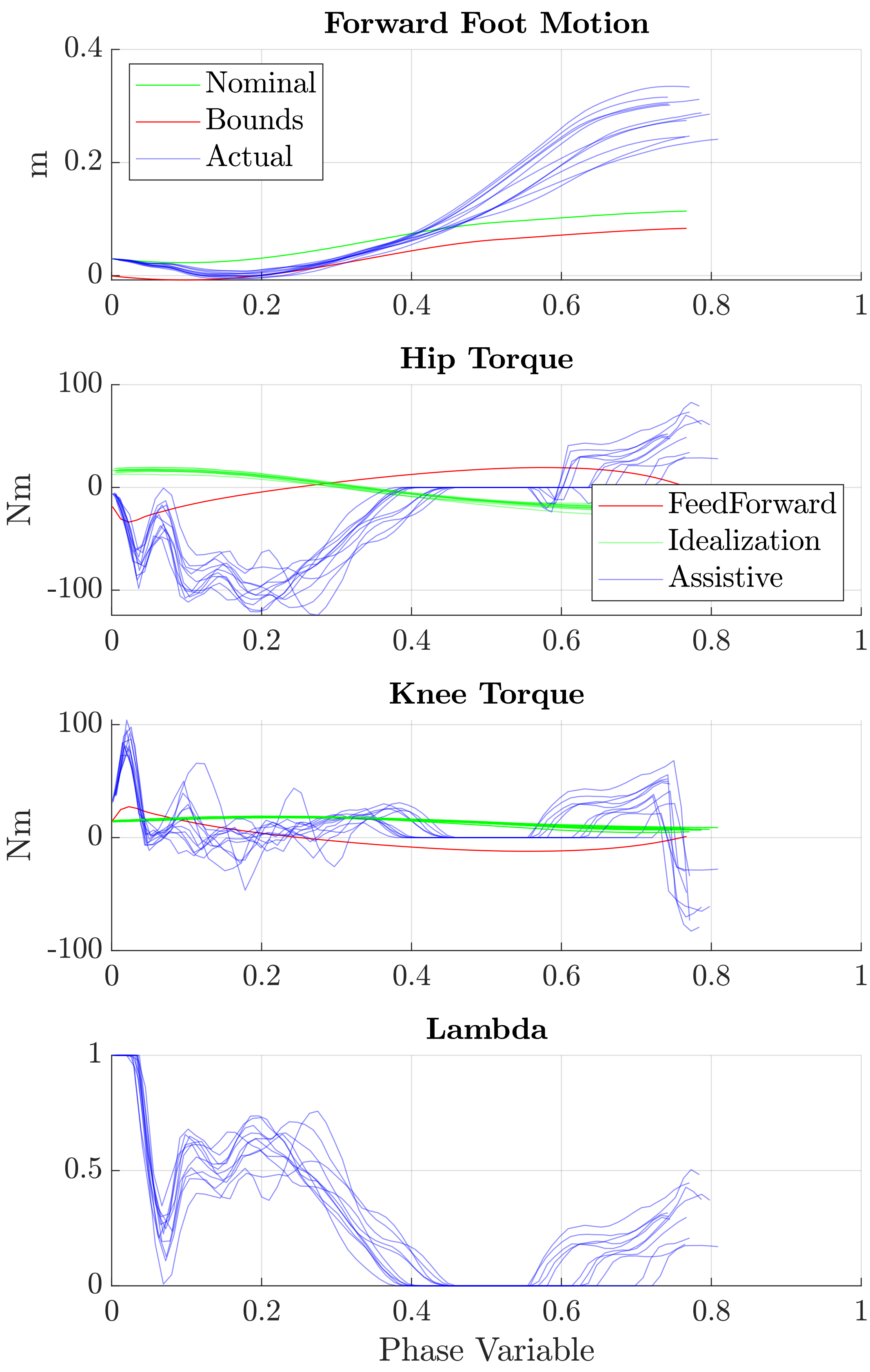
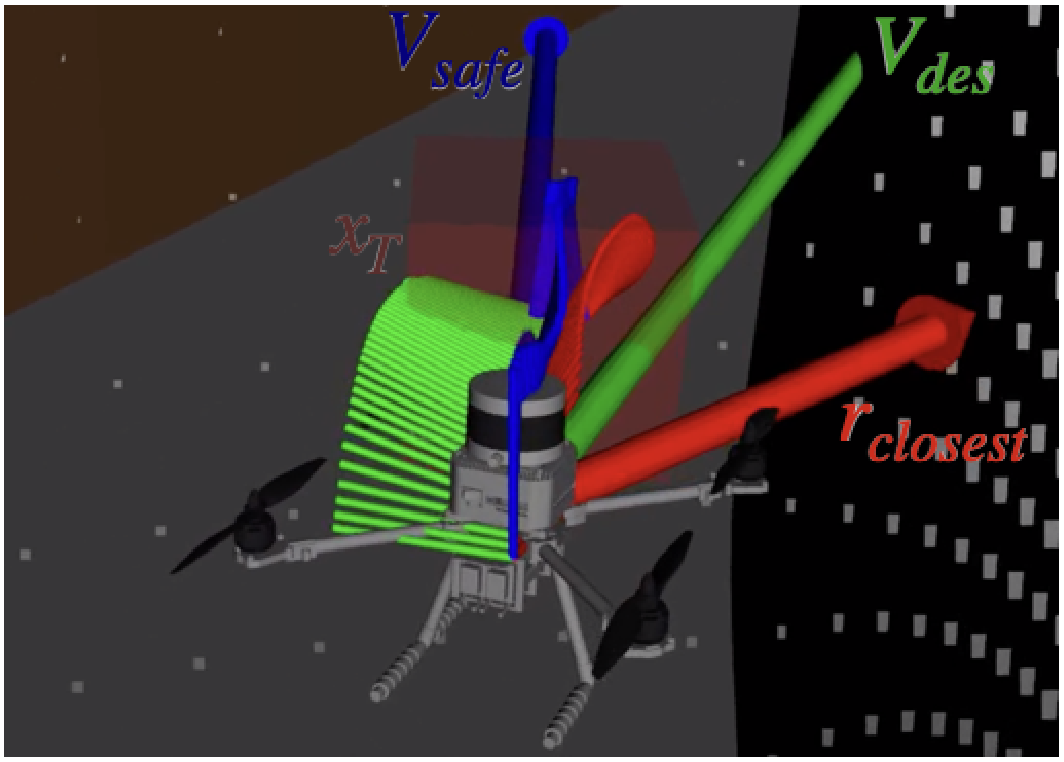


Figure 3.25: Foot Level Assistance for an able-bodied subject actively trying to take long steps.



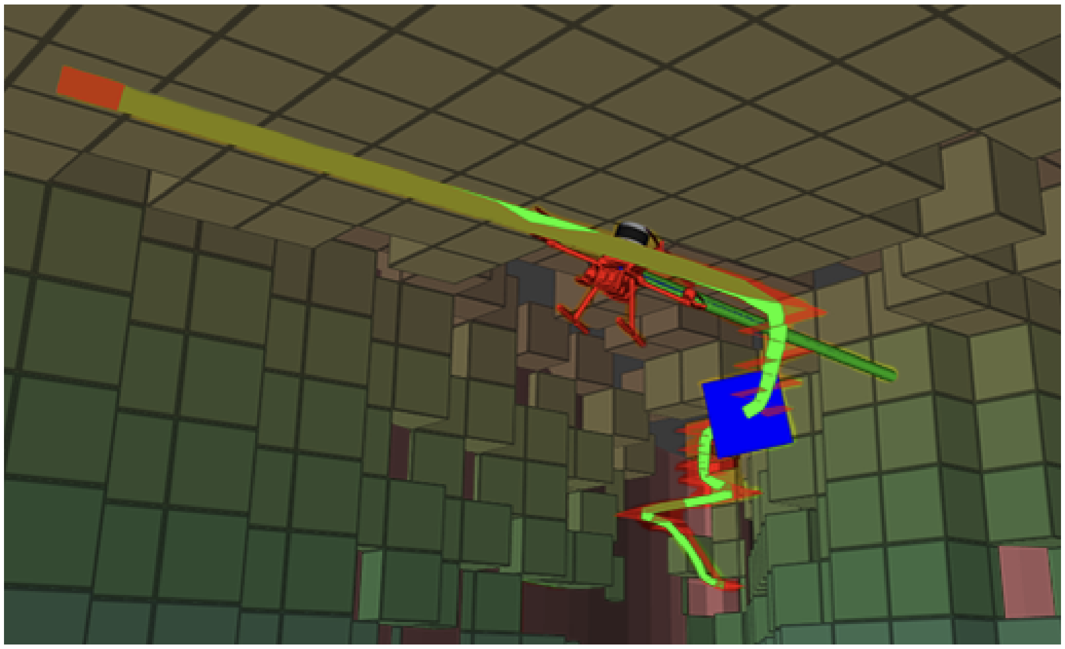


Figure 3.26: Simulation environment. The top shows the desired and filtered velocity commands based on the closest point in the point cloud. The bottom shows the drone navigating through the cave.

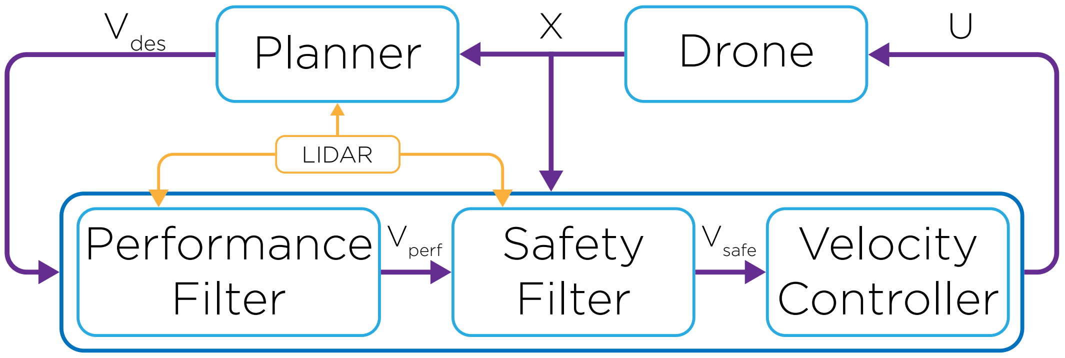


Figure 3.27: Safety filtering control structure for UAV exploration.

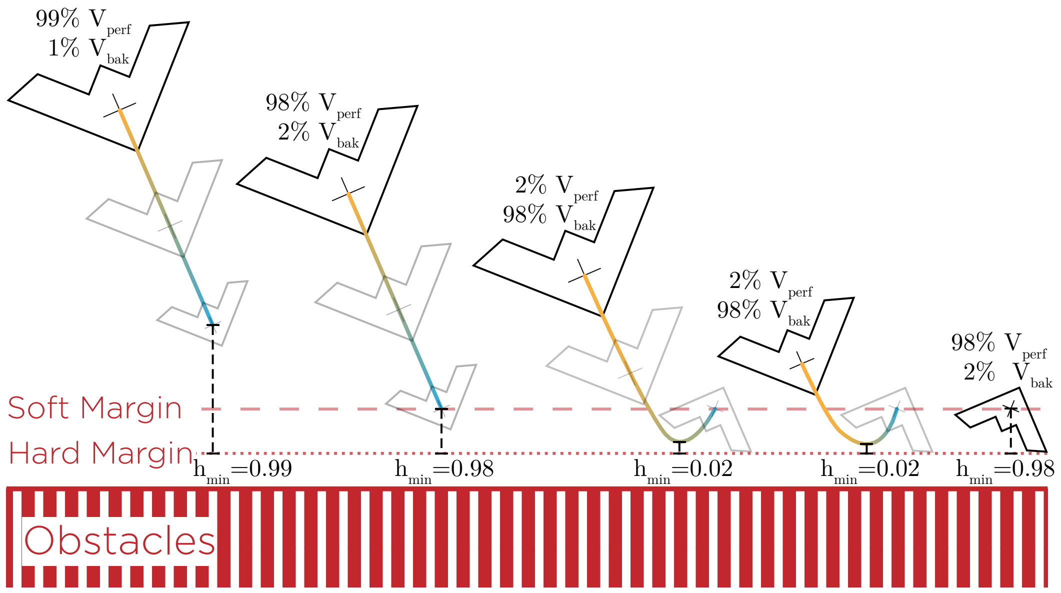


Figure 3.28: Illustration of the safety filter. From left to right is a depiction of the evolution of the drone as it gets close to the obstacles. In yellow to blue are the backup trajectories. In grey are the hypothetical positions of the drone if it were to follow the backup trajectory. The size of the drone icon corresponds to its velocity.

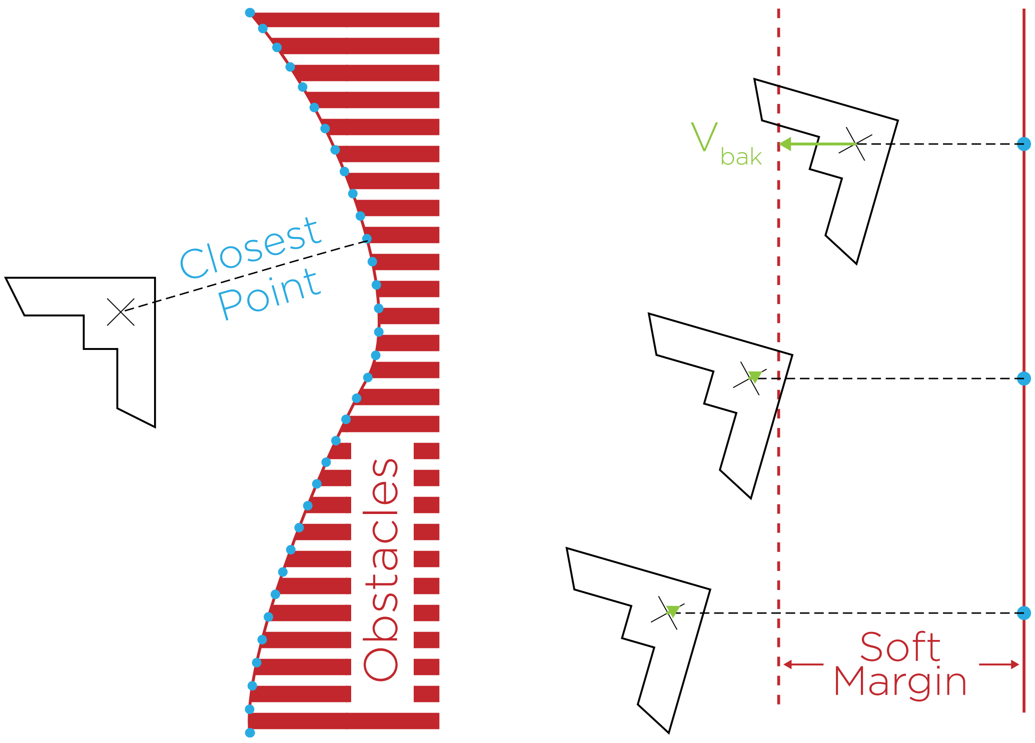


Figure 3.29: Illustration of the UAV backup law.

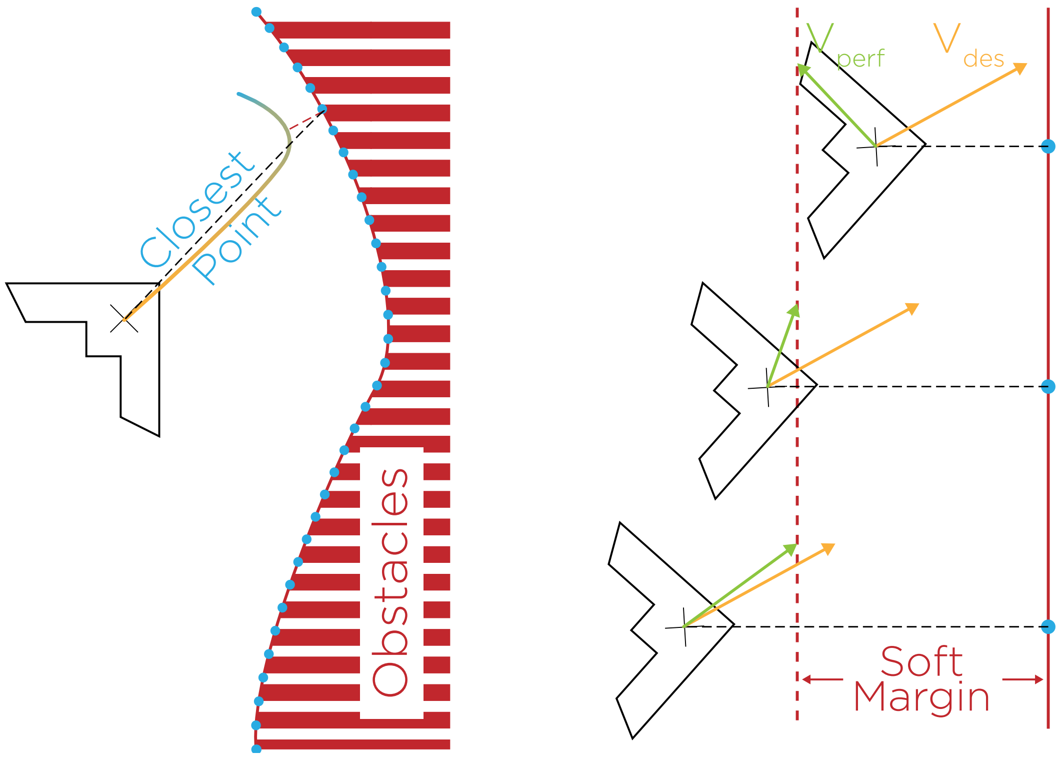


Figure 3.30: Illustration of the UAV performance filter.

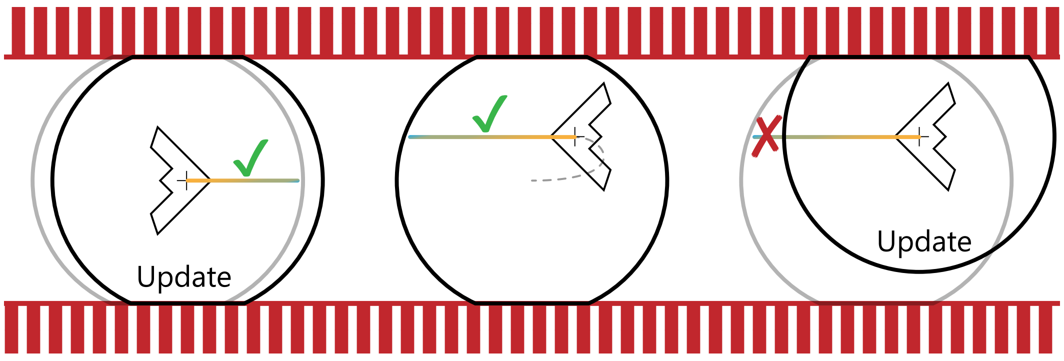


Figure 3.31: Illustration of the UAV recursive feasibility issue.

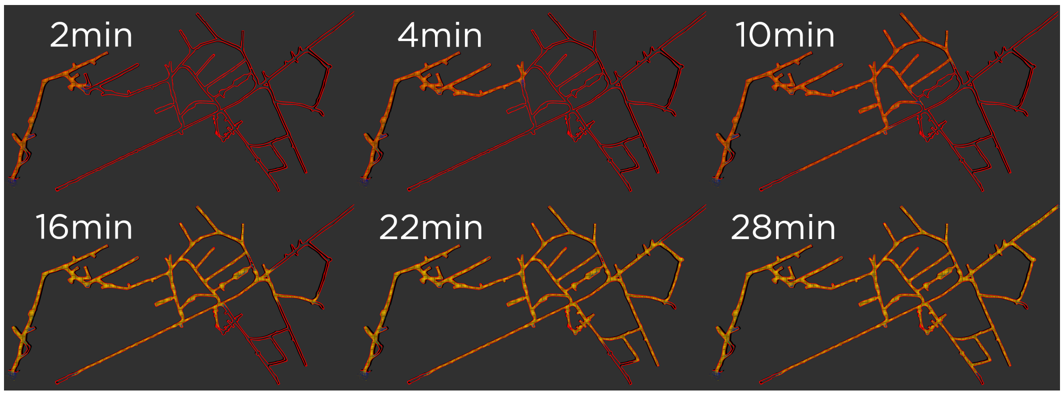


Figure 3.32: Pictures of the cave (in red) and the octomap (in yellow) being built throughout the 28 minutes it takes for the drone to completely explore the cave.

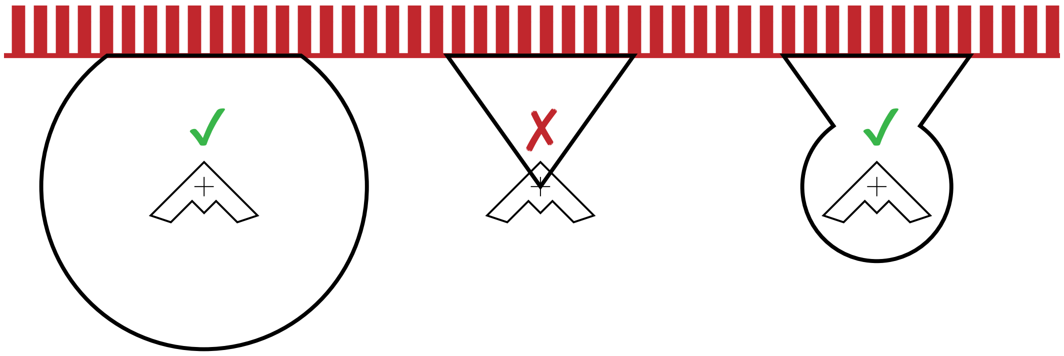


Figure 3.33: Illustration of the issue when using a mapping sensor with restricted field of view. An assumption about the safety of the vehicle has to be made at the initialization of the mapping in the form of a *bubble* of free space around the vehicle. The black line around the UAV represent the boundary between free and either unknown or occupied space

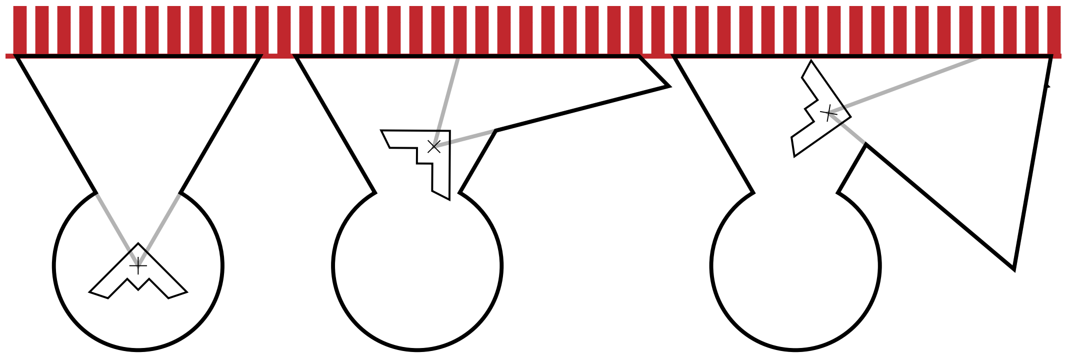


Figure 3.34: Illustration of the mapping process. The black line around the UAV represent the boundary between free and either unknown or occupied space.

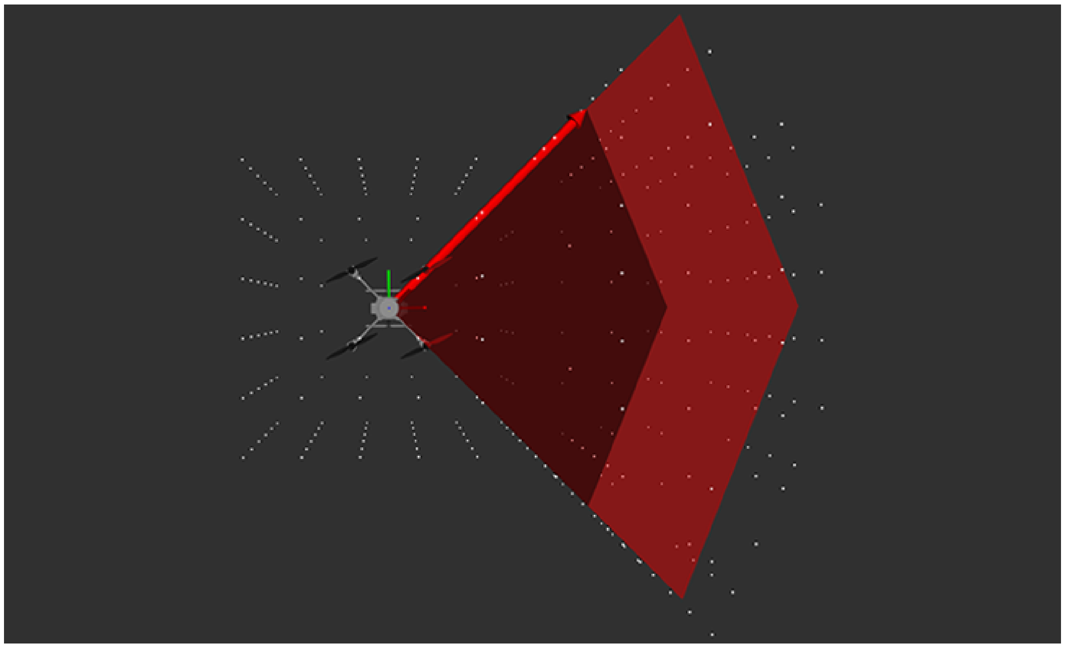
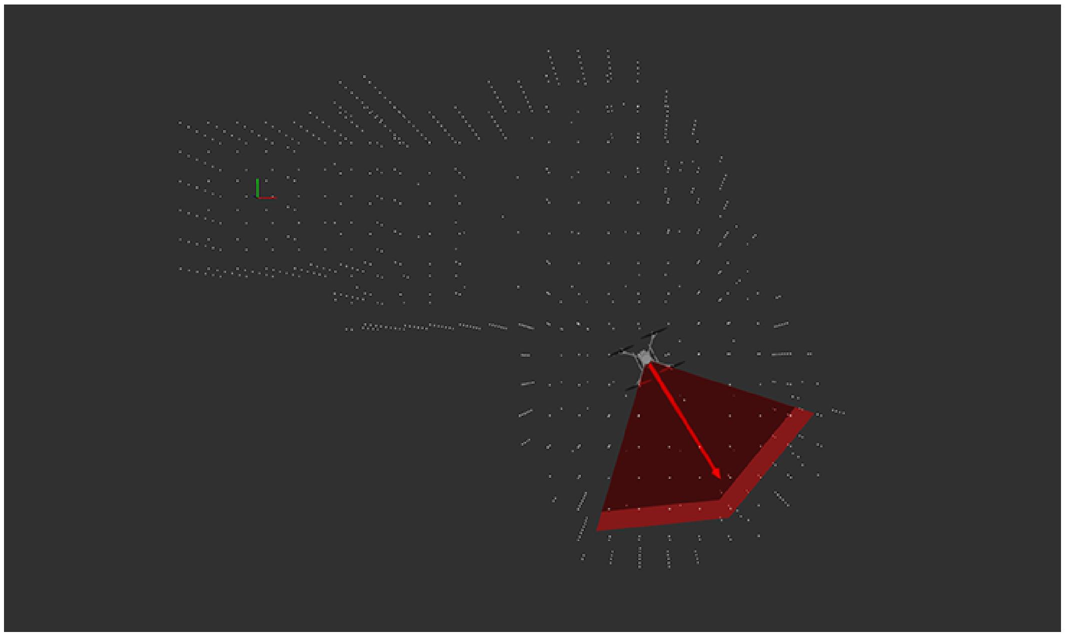



Figure 3.35: Image of the mapping process in the simulation environment. The free space boundary is represented by a point cloud.

Chapter 4

CONCLUSIONS

4.1 Summary

In this work, we presented a practical approach to run-time assurance to help bridge the gap between theoretical claims and the reality of cyber-physical systems implementation. We began by making a clear theoretical distinction between systems and a models, and outlined how the two need to be related for guarantees to transfer from the latter to the former. We then introduced set-invariance conditions accounting for the complexity and intricacy of cyber-physical system implementation. We then showed how these conditions can be rigorously enforced in a systematic and minimally invasive way through a convex optimization based Safety Filter. To guarantee the feasibility of such safety filter, a new algorithm was presented to compute appropriate control invariant sets. Finally, the effectiveness of this proposed framework was demonstrated experimentally on a two-wheeled inverted pendulum. First, the aptitude of the framework to handle system's dynamics uncertainty was illustrated by varying the mass of the vehicle and observing that safety is conserved. Then, the aptitude of the framework to provide guarantees that account for controller implementation's constraints was illustrated by varying the frequency of the control loop and observing that safety is maintained.

In the second part of this work, a Scalable Safety Critical Control Framework is presented. This framework makes it possible to enforce safety for high dimensional nonlinear systems in a minimally invasive way. The trade-off between computational complexity and conservativeness is analysed and approaches with varying levels of scalability are proposed. The idea of composing backup controllers through functional approximation of optimal policies is explored as a potential method of combining the advantages of scalability seen in simpler controllers with the permissiveness characterizing optimal controllers. Finally, the effectiveness of the framework is illustrated with multiple relevant applications. In particular, we show how this framework makes it easy to address safety in both time varying and uncertain environments. We also show how it enables fast and safe exploration of unknown environments. Finally, we showcase the effectiveness of the framework on hardware through the safe control of a two-wheeled inverted pendulum (Segway), and with the assistive control of a lower body exoskeleton.

4.2 Improvements in Regulation Kernel Computations

Despite achieving the goal of computing regulation kernels, the algorithms as presented here suffers from two majors drawbacks.

Firstly, the proposed formulation restricts itself to convex sets. However, the fundamental constraint that actually needs to be enforced is convexity of the regulation map. In the case of the polygonal parameterization we chose, it is equivalent to the convexity of the set itself. Therefore, if one can find a parameterization where convexity of the regulation map and the set are decoupled, it would make it possible to find potentially larger regulation kernels.

Secondly, for a given number of vertices N_v, the number of facets and faces increases very quickly with the dimensions of the system considered. However, the proposed algorithm has the benefit of being highly parallelizable, as most of the optimization time is spent evaluating the constraints for every faces, none of which are coupled.

This scalability issue can also be tackled along with the conservatism issue by performing mesh refinement steps between the resolutions of the smooth non-linear problems (2.94). Indeed, some conservatism is introduced by evaluating each sub-tangentiality constraint over an entire face. The larger the face, the more conservatism is introduced. Therefore, by modulating the size of the faces such that they are small in regions where the dynamics varies a lot, and large where it does not vary much, conservatism can be reduced. Furthermore, a metric of this dynamics variation is already accessible as the width of the computed bounds on (2.72).

Finally, the constraint bound update laws (2.96) and Algo. (1) are fairly simplistic. The robustness in terms of convergence of this two algorithms could most likely be improved by formulating an optimization problem and using an evolutionary algorithm to solve for the $\varepsilon^*_{i,j,k}$ that maximize the $\min\limits_{i,j,k}(STC_{i,j,k})$ such that $-\epsilon \leq STC^*_{i,j,k} \leq 0$.

4.3 Beyond Interval Arithmetic

Aside from the possible improvements to the regulation kernel algorithm, one of the main cause of conservatism in the proposed approach is the interval arithmetic based over-approximation of the different sub-regulation maps. Finding other methods or formulations to generate polytopic over-approximations of the sub-regulation maps that are computationally efficient is a real challenge that, to our knowledge, has yet to be undertaken.

Furthermore, the various safety filter proposed in this work has a tendency to generate filtered input with high frequency oscillatory in certain cased. A formulation allowing to explicitly trade off smoothness of the filter for performance (with a modified cost for example) would be a useful addition to the collection of formulations already proposed.

4.4 More Practical Implicit Safety Filtering

As discussed in the first part of this work, and touched upon in the work of Sec. 3.6.4, the extension of the implicit filtering framework to handle dynamics and sensing uncertainty is key to providing meaningful guarantees for real world safety critical applications.

Another challenge with safety critical control frameworks in general is the issue of representing the environment in such a way that provides meaningful guarantees when used in such framework. In Sec. 3.6.4, a discrete point-cloud representation is used that does not actually provide guarantees of avoidance with obstacles. Indeed, with this representation, only the parts of the obstacles represented by the points of the point cloud will be avoided, which can be an issue if the point cloud is not dense enough. Furthermore, in this same work, a map of the environment that is generated from perfect estimation of the vehicle position is used, which is obviously not realistic. Developing strategies for obstacle mapping that are both dense and truly safe, i.e. guarantee that no obstacles are within the free space while accounting for all sources of uncertainties and that are not too computationally expensive is a real challenge that deserves attention.

4.5 Quantifying Model Validity

From a more fundamental point of view, the guarantees provided by this framework are contingent on the correctness of the model used. As discussed, validity of models can only be verified probabilistically. Therefore, it would be interesting to quantify the probability of safety from the probability of model correctness in a less crude way than the obvious lower bound we provided.

Following a similar train of thoughts, being able to quantify model and controller validity for a given cyber-physical hardware would be very valuable in practice. Finally, it would be interesting to characterise system representativity and get a grasp on how to choose a *good* system and model for a given physical system.

BIBLIOGRAPHY

[1] Juan-Pablo Afman, Laurent Ciarletta, Eric Feron, John Franklin, Thomas Gurriet, and Eric N Johnson. Towards a new paradigm of uav safety. *arXiv preprint arXiv:1803.09026*, 2018.

[2] Ayush Agrawal, Omar Harib, Ayonga Hereid, Sylvain Finet, Matthieu Masselin, Laurent Praly, Aaron D. Ames, Koushil Sreenath, and Jessy W. Grizzle. First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons. *IEEE Access*, 5:9919–9934, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2690407.

[3] Aaron D Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, 2014.

[4] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, 2014. ISBN 978-1-4673-6090-6. doi: 10.1109/CDC.2014.7040372.

[5] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017. URL http://ames.caltech.edu/ames2017cbf.pdf.

[6] Khairul Anam and Adel Ali Al-Jumaily. Active exoskeleton control systems: State of the art. *Procedia Engineering*, 41:988–994, 2012.

[7] David Angeli and Eduardo D Sontag. Monotone control systems. *IEEE Transactions on Automatic Control*, 48(10):1684–1698, 2003.

[8] Jean-Pierre Aubin. *Viability theory*. Springer Science, 2009.

[9] Jean-Pierre Aubin and Hélène Frankowska. *Set-valued analysis*. Springer Science & Business Media, 2009.

[10] Stanley Bak, Deepti K Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo, and Lui Sha. The system-level simplex architecture for improved real-time embedded system safety. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 99–107. IEEE, 2009.

[11] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[12] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53, 2011.

[13] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[14] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*. Springer, 2008.

[15] Franco Blanchini and Mario Sznaier. A Convex Optimization Approach to Synthesizing Bounded Complexity ℓ^∞ Filters. *IEEE Trans. Autom. Control*, 57(1):216–221, 2012. doi: 10.1109/TAC.2011.2162893.

[16] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27): 68–73, 2015.

[17] Guy André Boy. From automation to tangible interactive objects. *Annual Reviews in Control*, 38(1):1–11, 2014.

[18] Lance L Cai, Andy J Fong, Chad K Otoshi, Yongqiang Liang, Joel W Burdick, Roland R Roy, and V Reggie Edgerton. Implications of assist-as-needed robotic step training after a complete spinal cord injury on intrinsic strategies of motor learning. *Journal of Neuroscience*, 26(41): 10564–10568, 2006.

[19] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[20] Jun Chai and Ricardo G. Sanfelice. On robust forward invariance of sets for hybrid dynamical systems. *Proc. ACC*, pages 1199–1204, 2017. doi: 10.23919/ACC.2017.7963116.

[21] Charles Dabadie, Shahab Kaynama, and Claire J Tomlin. A practical reachability-based collision avoidance algorithm for sampled-data systems: Application to ground robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4161–4168. IEEE, 2014.

[22] Luiz Henrique De Figueiredo and Jorge Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[23] Rudolfs Drillis, Renato Contini, et al. *Body segment parameters*. New York University, School of Engineering and Science Research Division, NY, 1966.

[24] Xiaocong Fan. *Real-Time Embedded Systems: Design Principles and Engineering Practices*. Newnes, 2015.

[25] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9):867–875, 1999.

[26] Aleksej Fedorovič Filippov. *Differential equations with discontinuous righthand sides: control systems*, volume 18. Springer Science & Business Media, 2013.

[27] Olivier Gay, David Coeurjolly, and Nathan Hurst. Libaffa-c++ affine arithmetic library for gnu/linux, 2006.

[28] Jeremy H Gillula, Shahab Kaynama, and Claire J Tomlin. Sampling-based approximation of the viability kernel for high-dimensional linear sampled-data systems. In *Proceedings of the 17th international conference on Hybrid systems*, pages 173–182. ACM, 2014.

[29] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[30] T. Gurriet, M. Tucker, A. Duburcq, G. Boeris, and A. D. Ames. Towards variable assistance for lower body exoskeletons. *IEEE Robotics and Automation Letters*, 5(1):266–273, Jan 2020. ISSN 2377-3774. doi: 10.1109/LRA.2019.2955946.

[31] Thomas Gurriet. Experimental results for the segway:. youtu.be/7wZJksFMAvk, 2016.

[32] Thomas Gurriet. Simulation of the robotic arm:. vimeo.com/320906655, 2019.

[33] Thomas Gurriet. Experimental results for the exoskeleton:. youtu.be/UJC5j4BFxyo, 2019.

[34] Thomas Gurriet. Simulation of the uav:. youtu.be/MdOKY-ykCSw, 2019.

[35] Thomas Gurriet and Laurent Ciarletta. Towards a generic and modular geofencing strategy for civilian uavs. In *International Conference on Unmanned Aircraft Systems*, pages 540–549. IEEE, 2016.

[36] Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, Ayonga Hereid, Omar Harib, Matthieu Masselin, Jessy Grizzle, and Aaron D Ames. Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2804–2811. IEEE, 2018.

[37] Thomas Gurriet, Mark Mote, Aaron D Ames, and Eric Feron. An online approach to active set invariance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3592–3599. IEEE, 2018.

[38] Thomas Gurriet, Andrew Singletary, Jake Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. Towards a framework for realizable safety critical control through active set invariance. In *Proceedings of the 9th International Conference on Cyber-Physical Systems*. IEEE, 2018.

[39] Omar Harib, Ayonga Hereid, Ayush Agrawal, Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, M Eva Mungai, Mattieu Masselin, Aaron D Ames, et al. Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking. *IEEE Control Systems Magazine*, 38(6):61–87, 2018.

[40] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1447–1454. IEEE, 2016.

[41] Ian A Hiskens and MA Pai. Trajectory sensitivity analysis of hybrid systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(2):204–220, 2000.

[42] Nhut Ho, Garrett G Sadler, Lauren C Hoffmann, Kevin Zemlicka, Joseph Lyons, William Fergueson, Casey Richardson, Artemio Cacanindin, Samantha Cals, and Mark Wilkins. A longitudinal field study of auto-gcas acceptance and trust: First-year results and implications. *Journal of Cognitive Engineering and Decision Making*, 11(3):239–251, 2017.

[43] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[44] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 807–814. IEEE, 2008.

[45] IFR. International federation of robotics, 2018. URL https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf.

[46] Ali Jadbabaie, Jie Yu, and John Hauser. Unconstrained receding-horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[47] Zachary Jarvis-Wloszek, Ryan Feeley, Weehong Tan, Kunpeng Sun, and Andrew Packard. Control applications of sum of squares programming. In *Positive Polynomials in Control*. Springer, 2005.

[48] René Jimenez-Fabian and Olivier Verlinden. Review of control algorithms for robotic ankle systems in lower-limb orthoses, prostheses, and exoskeletons. *Medical engineering & physics*, 34(4):397–408, 2012.

[49] Volker Kaibel and Marc E Pfetsch. Computing the face lattice of a polytope from its vertex-facet incidences. *Computational Geometry*, 23(3): 281–290, 2002.

[50] Wisama Khalil. Dynamic modeling of robots using recursive newton-euler techniques. In *ICINCO2010*, 2010.

[51] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505. IEEE, 1985.

[52] Edward A Lee. Cyber physical systems: Design challenges. In *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on*, pages 363–369. IEEE, 2008.

[53] John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–29. Springer, 2003.

[54] Nancy G Leveson. System safety engineering: Back to the future. *Massachusetts Institute of Technology*, 2002.

[55] Yucong Lin and Srikanth Saripalli. Sampling-based path planning for uav collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3179–3192, 2017.

[56] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017.

[57] Brett T Lopez and Jonathan P How. Aggressive 3-d collision avoidance for high-speed navigation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5759–5765. IEEE, 2017.

[58] Lui Sha. Using simplicity to control complexity. *IEEE Software*, 18(4): 20–28, July 2001. ISSN 1937-4194. doi: 10.1109/MS.2001.936213.

[59] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[60] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control and verification of high-dimensional systems with dsos and sdsos programming. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, page 394. IEEE, 2014.

[61] Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. Tira: Toolbox for interval reachability analysis. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 224–229, 2019.

[62] Ian Mitchell. A summary of recent progress on efficient parametric approximations of viability and discriminating kernels. In *SNR@ CAV*, pages 23–31, 2015.

[63] Ian M Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35(2):300–329, 2008.

[64] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 2005.

[65] Ian M Mitchell, Shahab Kaynama, Mo Chen, and Meeko Oishi. Safety preserving control synthesis for sampled data systems. *Nonlinear Analysis: Hybrid Systems*, 10:63–82, 2013.

[66] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*, volume 110. Siam, 2009.

[67] César Munoz, Anthony Narkawicz, and James Chamberlain. A tcas-ii resolution advisory detection algorithm. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4622, 2013.

[68] Quan Nguyen and Koushil Sreenath. Optimal robust safety-critical control for dynamic robotics. *International Journal of Robotics Research (IJRR)*, in review, 2016.

[69] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[70] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th design automation conference*, pages 731–736. ACM, 2010.

[71] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 3, pages 1936–1941. IEEE, 2002.

[72] Muhammad Zakiyullah Romdlony and Bayu Jayawardhana. On the new notion of input-to-state safety. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 6403–6409. IEEE, 2016.

[73] Hanan Samet. Neighbor finding in images represented by octrees. *Computer Vision, Graphics and Image Processing*, 46(3):367–386, 1989. ISSN 0734189X. doi: 10.1016/0734-189X(89)90038-8.

[74] Tom Schouwenaars. *Safe trajectory planning of autonomous vehicles*. PhD thesis, Massachusetts Institute of Technology, 2005.

[75] Hans Seywald and Renjith R Kumar. Desensitized optimal trajectories. *Advances in the Astronautical Sciences*, 93(1):103–116, 1996.

[76] Jeff S. Shamma and Kuang Yang Tu. Set-valued observers and optimal disturbance rejection. *IEEE Trans. Autom. Control*, 44(2):253–264, 1999. doi: 10.1109/9.746252.

[77] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames. Online active safety for robotic manipulators. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 173–178, Nov 2019. doi: 10.1109/IROS40897.2019.8968231.

[78] A. Singletary, T. Gurriet, P. Nilsson, and A. D. Ames. Enabling rapid aerial exploration of unknown environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[79] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*, November 2017.

[80] Weehong Tan and Andrew Packard. Searching for control lyapunov functions using sums of squares programming. In *42nd Annual Allerton Conference on Communications, Control and Computing*, 2004.

[81] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.1 edition, 2020. URL https://doc.cgal.org/5.0.1/Manual/packages.html.

[82] Michael R Tucker, Jeremy Olivier, Anna Pagel, Hannes Bleuler, Mohamed Bouri, Olivier Lambercy, José del R Millán, Robert Riener, Heike Vallery, and Roger Gassert. Control strategies for active lower extremity prosthetics and orthotics: a review. *Journal of neuroengineering and rehabilitation*, 12(1):1, 2015.

[83] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[84] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Safety Barrier Certificates for Collisions-Free Multirobot Systems. *IEEE Trans. Robotics*, 33 (3):661–674, 2017. doi: 10.1109/TRO.2017.2659727.

[85] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[86] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion.* CRC press, 2018.

[87] David A Winter. *Biomechanics and motor control of human movement.* John Wiley & Sons, 2009.

[88] Xiangru Xu, Jessy W Grizzle, Paulo Tabuada, and Aaron D Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 2017.