Symbolic models for braking systems

Nikolaos Kekatos

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Symbolic models for braking systems

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Nikolaos Kekatos

February 10, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was partly carried out at Volvo GTT - Advanced Technology & Research. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Symbolic models for braking systems

by

Nikolaos Kekatos

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: February 10, 2015

Supervisor(s):

dr.ir. Manuel Mazo Jr.

dr.ir. Ubaldo Tiberi

Reader(s):

prof.dr.ir. Hans Hellendoorn

dr.ir. Meng Wang

Abstract

A constantly increasing number of driving assistance and safety systems is being incorporated into road vehicles. Yet, the safe integration of new software modules to existing control functionalities presents many challenges in terms of ensuring infallible performance. In the automotive industry, simulation and testing remain the widely adopted paradigm for the establishment of any behavioral property of control software systems. Under this setup, the verification efforts cannot provide a high degree of confidence that control software will behave correctly whenever deployed.

In the case of automated vehicles, this challenge is even greater, insofar no human intervention and corrective actions may be possible. The goal of this thesis is to address the braking control problem of a fully automated vehicle. The principal control objective is to minimize the braking distance of the vehicle while maintaining its steerability. The braking systems normally constitute the most important active safety system, with the industrial practice relying on wheel deceleration heuristics. As such, they may raise severe performance concern. These issues can be alleviated by formulating the braking control objectives as a wheel slip regulation problem.

The necessity to provide guarantees of correctness and performance has motivated the use of formal methods, which may allow the elimination of design flaws and software bugs, and reduce the consumption of the design budget in validation and verification efforts. Two different approaches have been used. The first one is formal synthesis (correct-by-design paradigm), which combines control synthesis and verification. The control objectives are formalized in Linear Temporal Logic formulae, while the differential equations are transformed into equivalent finite-state models with the use of approximate simulation relations. The second approach involves traditional control design followed by the verification process.

Different braking scenarios including the uncertainties in tire friction models and physical parameters have been considered. The findings show that the verification of automotive control software and the applicability of the correct-by design approach, albeit its relative limitations, may offer a very promising alternative tool available to the academic and industrial society for further development. Other contributions of the thesis are the generation of a wheel slip reference signal under varied road conditions and a hybrid scheme for online estimation of the optimal wheel slip.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my supervisor dr. ir. Manuel Mazo for his support throughout my thesis project. I am thankful for his scientific guidance, his constructive criticism and his friendly advice. But above all, I am grateful to him for sharing his illuminating views and his solid way of thinking on quite a number of issues.

Also, I would like to thank dr. ir. Ubaldo Tiberi for the opportunity to conduct part of my thesis in Volvo Trucks in Sweden. Our discussions on requirement management and V-model methodology have been invaluable and his industrial perspective on control software development is greatly appreciated.

Additionally, I would like to thank prof. dr. ir. Hans Hellendoorn and dr. ir. Meng Wang for for taking part in the in the defense committee, despite the difficulties imposed by their very tight business schedule.

Delft, University of Technology Nikolaos Kekatos
February 10, 2015

Chapter 1

Introduction

1-1 Introduction to the subject

According to the National Highway Traffic Safety Administration (NHTSA) vehicle automation may be classified into five distinct levels [1].

- **Level 0 - No automation:** The driver completely controls the vehicle at all times.
- **Level 1 - Driving Assistance:** The driver permanently controls either longitudinal (acceleration/deceleration) or lateral (steering) dynamics. The other control task can be automated to a certain extent by the assistance system.
- **Level 2 - Partial Automation:** The vehicle takes over longitudinal and lateral control (acceleration/deceleration and steering). The driver should constantly monitor the system and be ready to take over control at any time.
- **Level 3 - High Automation:** The vehicle takes over longitudinal and lateral control. The driver does not have to permanently monitor the system. Under certain conditions, the vehicle may notify the driver to take control by offering to them sufficient transition time.
- **Level 4 - Full Automation:** The vehicle performs all safety-critical functions during the entire course trip, with the driver not expected to control the vehicle at any time. In this case, the vehicle is able to undertake full driving all functions from start to stop, including parking.

This purpose of this thesis project is to tackle the issue of braking control in the context of Dutch Automatic Vehicle Initiative (DAVI). The DAVI project has been initiated by Delft University of Technology (TU Delft) and is presently developed in collaboration with research and industrial partners [2]. Part of this thesis was completed in Volvo Advanced Technology & Research Group, which provides the appropriate real world environment for specifying the

benefits of formal methods (correct-by-design & verification) in process and product quality, vis-a-vis the standard traditional industrial practices (VV model, simulation & testing) by using a comparative analytical methodology.

The DAVI project has set the ambitious goal to develop safe and reliable automated ground vehicles in Netherlands that fulfill the requirements foreseen in the level 4 NTHSA classification [3]. Automated vehicles may be very beneficial not only for the drivers and but also for the society, as a whole. The traffic accidents and congestion may be largely reduced, while the travel time could become smaller with positive repercussions in terms of savings in energy efficiency costs. However, there exist several difficulties that restrict implementation on public roads. Among the issues that should be addressed there are human factors (user acceptance), legal aspects, traffic infrastructure as well as technological challenges. Relating to the latter, it is worth mentioning that the absence of human driver interference requires the existence of multiple sensors and actuators, connected by control strategies. Unless these technologies are accurately and reliably synthesized, the successful implementation of such demanding projects would be possible delayed raising a lot of concerns to all involved parties. That makes necessary to design and develop control systems that would always meet their objectives and operate correctly under all possible conditions.

1-2 Motivation

One of the most influential factors in technological advances and engineering innovations has been the continuous improvement of digital processor technology. The analog implementations have been replaced by digital implementations of control algorithms, offering adaptability, simplicity and low costs [4]. As a result, connections between physical systems and computing elements have been intensified in many industries, such as automotive, energy, chemical and transportation. Systems with strong interconnection between physical and computational elements are known as cyber-physical systems (CPS) [5]. The DAVI project is by design a cyber-physical system, as far as a close interaction between the vehicle, a physical system and information processing based devices (computers, fpga, controllers) is present. CPS offer clear benefits, but they are heterogeneous and difficult to analyze. Faced with the increasing complexity of these systems, it is vital to use analysis tools to better understand their behavior and ensure their safety.

Traditional system design has used extensive testing to verify behavior. The big advantage of a simulation is that it might produce a counter-example, i.e. a trajectory that hits a set of unsafe states. In this case, one can show that a system is unsafe. However, one cannot prove that the system is safe if no counter-example is produced, since there exist infinitely many possible trajectories due to uncertain initial states, inputs, and parameters [6]. Dijkstra and others have demonstrated that tests can only show the situations where a system will not fail, but cannot say anything about the behavior of the system outside of the testing scenarios [10].

The most popular approach is the use of formal methods. Formal methods are techniques used to model complex systems as mathematical entities. By building a mathematically rigorous model of a complex system, it is possible to verify the system properties in a more thorough fashion than empirical testing [9]. It is important to note that formal verification

does not replace the need for testing. Formal verification cannot fix bad assumptions in the design, but it can help identify errors in reasoning which would otherwise be left unverified. In several cases, engineers have reported finding flaws in systems once they reviewed their designs formally [11].

In the past, formal methods were solely used on software and hardware applications. However, the introduction of "Hybrid Systems" has provided a framework that extends to control systems. Hybrid systems are systems that are influenced and characterized by models with both discrete and continuous components, ranging from switched linear systems to full-scale hybrid automata. A number of results have emerged in this area with a classic control-theoretic flavor, including optimal control, stability, system identification, estimation, and well-posedness of solutions [7]. Hybrid systems form the appropriate mathematical model for embedded control systems, combining the traditional state-machine based models for discrete control with classical differential-equations based models for continuously evolving physical activities [8].

There are two main lines of research in formal methods for control systems. The first approach is the correct-by-design control synthesis, which enables a drastic reduction of validation iterations by automating the design process. This paradigm is grounded on mathematical and technical results that transform differential equations, describing a physical system, into an equivalent finite-state machine. Controller design problems are solved by using efficient synthesis algorithms operating over the equivalent finite-state machine models. The resulting controllers are finite-state, thus they guarantee the enforcement of the control specifications on the original physical system, and can be readily transformed into bug-free code for any desired digital platform. In particular, it is possible to automatically synthesize controllers enforcing discrete specifications (languages, finite-state machines, temporal logics) on continuous systems.

The second approach includes separate control design and formal verification. First, the control synthesis part is completed and then the verification process is initiated. Powerful verification methods already exist to prove the safe operation of complex discrete systems, known as model checking [12]. Starting from discrete automata, verification methods have been extended to timed automata which contain clocks, where each clock represents a continuous variable [13]. Verification methods have also been extended to hybrid automata which allow discrete dynamics with general continuous dynamics to be combined. Verification methods developed in the hybrid systems community have also become popular tools for the analysis of purely continuous dynamics. Additionally, a large variety of verification techniques have been suggested for stochastic systems with continuous and hybrid dynamics.

There are three main approaches for formal verification [8].

- Symbolic Reachability Analysis
- Deductive Verification
- Abstraction

More information about these approaches is available on the 2nd chapter. It is recalled that this thesis elaborates on both correct-by-design synthesis and formal verification.

1-3 Objectives

Braking control systems have been extensively applied in road vehicles. Anti-lock braking systems (ABS), emergency braking control, cornering brake control and engine braking are popular applications [17]. In particular, the technology of ABS control systems is considered to be rather mature. However, the control methods for practical ABS system are mostly based on deceleration thresholds and heuristics. The methods are reliable, but they require much experimentation for implementation. Typically, the braking performance and the feeling of the system is degraded [18]. However, all these systems consider the existence of a driver and run in parallel with driver actions. For the case of automated road vehicles, the braking actions should be prompted by a high level controller and should be active continuously.

The closed loop control system is shown as below.

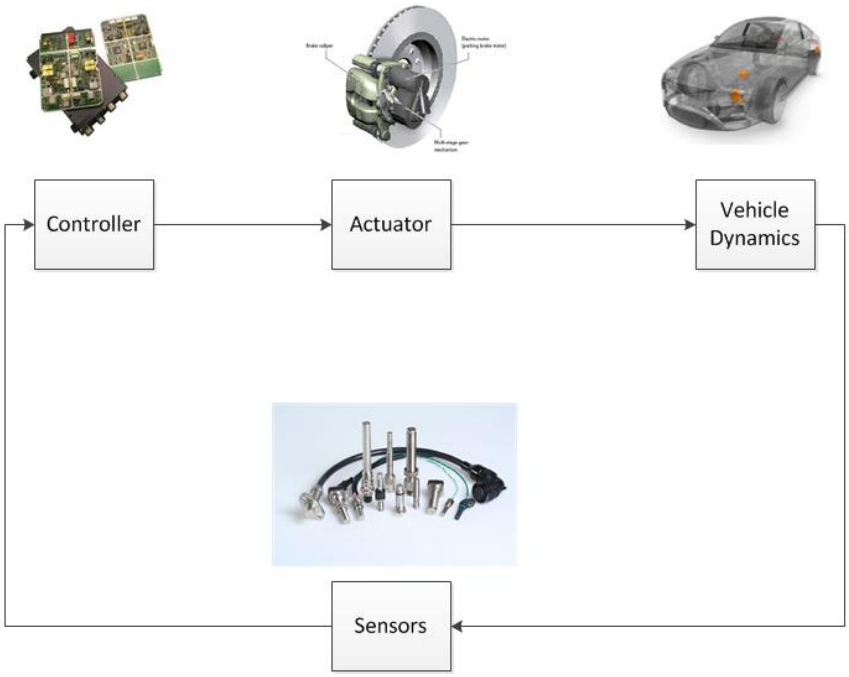

Figure 1-1: Braking system

The objectives of the braking control system are similar to commercial Anti-lock Braking Systems and are threefold.

1. **Reduction of stopping distances**
2. **Stability Improvement**
3. **Steerability during braking**

In this context, the aim is to minimize the braking distance of an autonomous automotive vehicle, while maintaining its driveability. There are limitations that should be considered. Firstly, the adhesion conditions vary over time, and the braking control system should be able

to function properly under different types of grip levels (i.e. dry, wet, snow). Secondly, the vehicle longitudinal speed, a key variable in the model, is difficult to measure experimentally, which requires the implementation of observers to estimate this variable [19]. Traditional braking actuators, based on hydraulic systems, also pose an additional problem, since they have a considerable pure delay and have discrete dynamics.

However, the design can be facilitated by recent technological advances in actuators which have led to both electro-hydraulic and electro-mechanical braking systems. This technology enables a continuous modulation of the braking torque, thereby allowing us to formulate active braking control as a classical regulation problem.

The braking distance is a function of the vehicle mass, the initial velocity, the road conditions and the friction force. Maximization of the friction force leads to minimization of the braking distance, if all other factors remain constant.

A more technical description of the problem statement is presented on the 2nd chapter.

1-4 Outline

The second chapter is devoted to the mathematical background and preliminary theory. Chapter 3 refers to dynamic modeling and control objectives in detail. In the fourth chapter, the correct-by-design approach is developed. In the fifth chapter, traditional control synthesis and verification processes are presented. The thesis ends up with conclusions and recommendations for future work.

Chapter 2

Preliminaries

This chapter is devoted to the technical preliminaries of the thesis work. First, the formal methods are introduced. Special attention is given to reachability analysis. Then, the correct-by-design framework is presented. Afterwards, the software tools for synthesis and verification are demonstrated. Next, the modeling of the braking systems is conducted. Finally, the problem statement is described in detail.

2-1 Formal Methods

A detailed overview of available techniques is given in [8]. The categorization presented below is motivated by [43]. This taxonomy has been selected due to the nature of this thesis work, which combines ideas from control theory and computer science.

2-1-1 Computer Science Oriented Approaches

In computer science, there are two main approaches: algorithmic and deductive.

The algorithmic approach could be seen as a brute-force simulation. The state space is exhaustively explored in order to check whether the desired properties of the system are satisfied. Model checking is the most established method that falls into this category [12]. A key requirement of algorithmic approaches is to describe the system and its requirements in a precise mathematical language. Based on this description, it is feasible to derive all the behaviors of the system. Then, it can be automatically checked whether the possible behaviors intersect with invalid behaviors. In case, the intersection is indeed an empty space, the model checker terminates with a positive answer. This method has been successfully used for software verification, but it suffers from state explosion and is limited to finite systems [44]. SPIN and SMV are two of the most popular model checkers [45].

The deductive approach relies on axioms and proof rules to prove the correctness of a system. The proofs are typically based on inductive invariants: If property ϕ is valid at the initial

stage of the system and all legal successors of every ϕ-state are ϕ states, then the property always holds. This approach is not limited to finite state systems. However, it demands a skilled human interaction. Examples of theorem provers are the Prototype Verication System [47] for finite state systems and KeYmaera for continuous, infinite state systems [46].

2-1-2 Control Oriented Approaches

Parallel to studies in computer science, control scientists have developed methodologies for verifying that a control system of the form $\dot{x}(t) = f(x(t), u(t))$ or $x[k+1] = f(x[k], u[k])$ stays within a certain safe set. The dual of this safety problem is the reachability problem that concerns that concerns the e proof of existence of a trajectory that starts from an initial set and reaches another given (goal) set. The two main approaches to solve these problems are reachability analysis and Lyapunov-type methods [43]. Direct reachability techniques seek to compute either a set of all states that can be reached by trajectories starting from a certain initial set, or a set of all initial states from which trajectories to a certain set of final states can be computed. The former case is called forward reachability and the latter is called backward reachability. A historical perspective can be found in [48]. More details about reachability analysis on continuous and hybrid systems are given in the next section.

Lyapunov-type methods do not require explicit computation of reachable sets and have the ability to handle non-linearity, uncertainty and constraints. A Lyapunov function satisfying certain algebraic conditions has to be found. This function, together with the corresponding algebraic conditions, provide a confirmation that all trajectories of the system starting from a given initial set remain within the safe set [43]. Examples of such function are the barrier certificates and density functions [50]. A barrier certificate and a density function can be automatically constructed using sum of squares techniques in conjunction with semidefinite programming. It is necessary that provided that the vector field is polynomial and the sets are semialgebraic (i.e., can be described by polynomial equalities and inequalities) [49]. Such construction relies on using the generalized S-procedure [51] and sum of square relaxations to translate the set containment constraints to a sum of squares optimization problem. This technique can be extended to certain non-polynomial systems by using suitable transformation [53].

A graphical explanation of aforementioned verification methods is illustrated below [6].

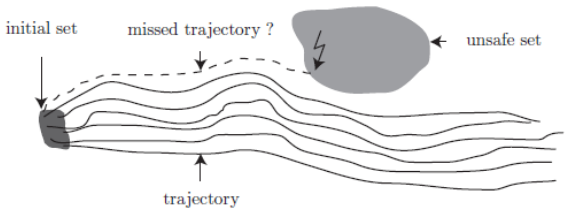

Figure 2-1: Searching for counter-examples by simulation [6].

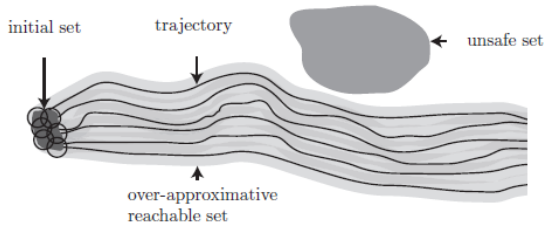

Figure 2-2: Verification by Simulation [6]

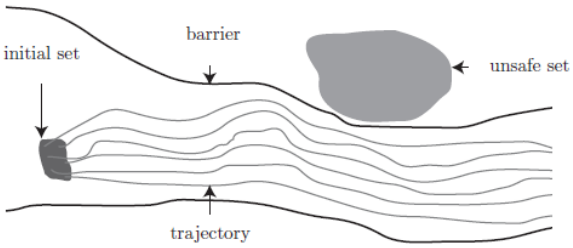

Figure 2-3: Verification using barrier certificates [6]

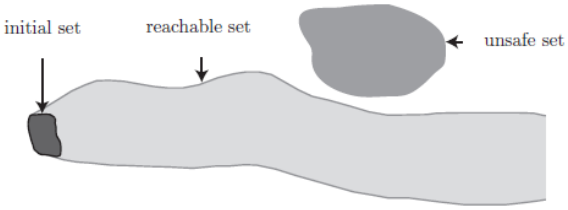

Figure 2-4: Verification using reachable sets [6]

2-1-3 Reachability Analysis

Concept & Theory

This section summarizes the most important contributions to the problem of computing reachable states of continuous and hybrid systems.

For continuous systems, the problem can be formulated as follows:

Let's consider a continuous dynamical system with input over some state space X defined via a differential equation of the form:

$$\dot{x} = f(x, u) \tag{2-1}$$

where u ranges over some pre-specified set of admissible input signals. Given a set $X_0 \subset X$, the problem is how to compute all the states visited by trajectories of the system starting from any $x_0 \in X_0$. The goal is to verify whether a system behaves correctly in the presence of all admissible disturbances, where "correctly" can mean the avoidance of a "bad" subset of the state space or, potentially, more complex temporal properties.

An intuitive explanation of reachability computation can be given in terms of numerical simulation. As stated in the previous chapter, each individual simulation consists of one initial condition and one input stimulus (random, periodic, step, etc.), and produces the trajectory using numerical integration. Then, one should observe whether this trajectory behaves correctly. Ideally we would like to repeat this procedure with all possible disturbances whose number is huge and typically even non countable[55].

Reachability computation achieves the same effect as exhaustive simulation by performing the simulation in a "breadth-first" manner: rather than running each individual simulation to completion and then starting a new one, one computes at each time step all the states reachable by all possible inputs. This set-based simulation is, of course, more costly than the simulation of individual trajectories but provides more confidence in the correctness of the system [43]. Unlike other methods, reachability computation covers also the transient behavior of the system in question, and not only its stationary behavior [55].

The exact reachable set of a continuous or hybrid system can only be obtained for small class of systems, such as timed automata, multirate automata ($\dot{x} = k, k \in \mathbb{R}^n$), rectangular automata, linear hybrid automata and a very special class of linear systems [56, 58]. The above system representations for exact reachable set computations can be used to verify general hybrid systems with linear or nonlinear continuous dynamics. This can be achieved by a conservative abstraction, i.e. all possible behaviors of the complex system are represented by simpler dynamics [6]. Abstractions for general hybrid automata to linear hybrid automata have been developed in [60].

Since the exact reachable set can be computed in special cases, the set has to be computed in an over-approximative way. Ideas from computational geometry have been well established for linear systems.

Several geometric representations for linear continuous systems have been investigated: Polytopes [61], griddy polyhedra [62], ellipsoids [63], oriented rectangular hulls [64], zonotopes [59], or support functions [65]. For linear systems with uncertain input, zonotopes and support functions have outperformed existing methods, allowing the verification of systems with more than 100 continuous state variables [55]. The algorithms for linear continuous systems can be applied to general nonlinear systems or hybrid systems. This is achieved by conservative linearization, i.e. by considering the linearization error as an additional uncertain input of the linearized system [55]. Abstraction to linear systems and multi-affine systems using a fixed partition has been investigated in [66]. This method is also called hybridization since one obtains a hybrid system where each linearization region is subject to a different continuous dynamics. The disadvantage of the hybridization method is the limited scalability [6]. One reason for this is the exponential growth of state space regions with the system dimension n. Another reason is the high computational effort for transitions between state space regions, which can be dropped when using on-the-fly partitioning with overlapping state-space regions [67]. The explicit computation of over-approximated reachable sets has been performed for polynomial nonlinear systems using Bezier control nets and the Bernstein expansion in [68].

2-1-4 Usage & Computational Issues

Besides safety verification, there are other possible applications for reachability analysis [6]:

- Performance assessment of control strategies: It can be checked if the system trajectories stay in a region around a reference trajectory, or reach a goal region around a setpoint.
- Scheduling: Reachability analysis can verify if the optimal schedule of a system (typically a production system) is ensured under all conditions, or if e.g. the supervisory controller might run into a recovery mode.
- Controller synthesis: The safety verification capabilities of reachability analysis can be used to find parameter sets of controllers that satisfy safety constraints.
- Deadlocks: Reachability analysis can determine whether a system might get stuck in a certain region of the continuous state space or an operation mode of a hybrid system.
- Set based observers: Instead of estimating the state of a system with stochastic methods (e.g. Kalman filtering), one can develop set based observers which return the set of possible states. Therefore, the set of successor states has to be computed via reachability analysis.

An issue that arises in complex, hybrid systems is the non-uniqueness of solutions. Especially, hybrid systems are challenging from the point of view of simulation. These problems affect reachability analysis and are mainly related to the following issues [14].

- Existence: simulation algorithms may run into trouble if the simulated model has no solutions. Incorporating tests for existence in the simulation packages can alleviate this problem. More challenging is the case of Zeno executions. In this case, unless special care is taken, the simulation may grind to a halt, or produce spurious results.
- Uniqueness: Non-determinism introduces further complications from the point of view of simulation. Here the simulation algorithm may be called upon to decide between different alternatives. When a choice between continuous evolution and discrete transition is possible, a common approach is to take transitions the moment they are enabled.
- Discontinuity: Lack of continuity of the solution with respect to initial conditions, an inherent characteristic of hybrid systems, can also lead to problems, both theoretical and practical. The most common problem is event detection (guard crossing).
- Composability: When simulating large scale systems, one would like to be able to build up the simulation by composing different components. It may also be desirable to add components to the simulation on-line, eliminate components, or redefine the interactions between components. Object oriented modeling languages have been developed to address these needs.

Some simple examples are shown below [15].

1. Existence problem:
$$\dot{x} = f(x) = \begin{cases} -1 & x \geqslant 0 \\ 1 & x < 0 \end{cases}$$
There is no solution to this differential equation that starts with $x(0) = 0$.

2. Uniqueness problem:
$$\dot{x} = f(x) = \sqrt{|x|} \text{ with } x(0) = 0$$
This function has two solutions, $x(t) = 0$ and $x(t) = \frac{t^2}{4}$

3. Finite time escape problem:
$$\dot{x} = x^2 \text{ with } x(0) = 0$$
The solution will be infinite after some time $x(t) = \frac{1}{T-t} \quad t \in [0, T]$.

2-2 Correct-by-design Synthesis

2-2-1 Concept

Nowadays, the most common paradigm for embedded control systems design is based on an iterative scheme.

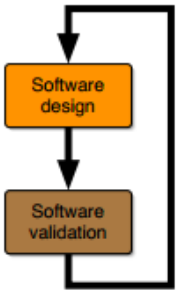

Figure 2-5: Existing paradigm [21]

However, this iterative scheme has several drawbacks:

- Validation by extensive simulation and testing increases our confidence in the software but fails to provide adequate guarantees of correct operation and performance
- Formal verification is currently limited to finite state systems and thus cannot be used to verify properties depending on continuous components
- Extensive validation is time consuming thus increasing the cost and time-to-market of embedded software.

Some of these disadvantages can be mitigated by adopting a correct-by-design approach to the development of embedded control software [21]. The key idea in this approach is to regard the synthesis of software as a control problem to be solved along with the synthesis of control algorithms. The most important part of this approach is the definition of a symbolic model that approximates the continuous control system. Given that these symbolic models are of the same nature as the models describing software, they provide a unified framework to study controller synthesis problems, which are characterized by the interaction of the physical layer with software and hardware [100].

The synthesis of embedded control software is a three-phase approach.

1. Abstraction
2. Discrete controller synthesis
3. Controller refinement

Graphically,

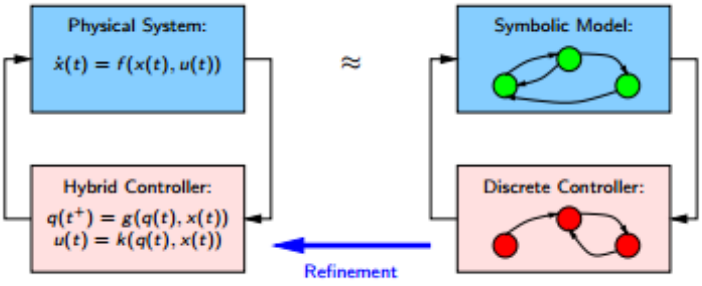

Figure 2-6: Correct-by-design paradigm [22]

2-2-2 System Definition

Although there are many mathematical models that describe a dynamical phenomenon, our particular interest is in models that can capture states that belong to finite, infinite sets, and combinations of them. A finite-state system is a system that can be described by finitely many states, such as a digital circuit [25]. While infinite state systems are described by difference or differential equations with solutions that evolve in infinite sets, i.e. $\mathbb{R}^n$ [26]. The class of hybrid systems should also be captured [27, 28]. The definitions that are used thereafter in this section are adopted by [25].

Definition 2.1 (Systems). A system S is a sixtuple $(X, X_0, U, \rightarrow, Y, H)$ consisting of:

- a set of states X;
- a set of initial states $X \subseteq X_0$;
- a set of inputs U;

- a transition relation $\rightarrow \subseteq X \times U \times X$;
- a set of outputs Y;
- an output map $H : X \rightarrow Y$.

States in X are regarded as internal to the system whereas outputs are externally visible. A system is called finite-state if X is a finite set. A system that is not finite-state is called infinite-state. The evolution of a system is captured by the transition relation. A transition $(x, u, x') \in \rightarrow$ is denoted by $x \xrightarrow{u} x'$. For such a transition, state x' is called a u-successor, or simply successor, of state x. Similarly, x is called a u-predecessor, or predecessor, of state x'. Note that, since $\rightarrow \subseteq X \times U \times X$ a relation, for any state and any input $u \in U$ there may be: no u-successors, one u-successor, or many u-successors. The set of u-successors of a state x is denoted by $\text{Post}_u(x)$. Since $\text{Post}_u(x)$ may be empty, we denote by $U(x)$ the set of inputs $u \in U$ for which $\text{Post}_u(x)$ is nonempty. Inputs in U can represent choices to be made by a controller, choices to be made by the environment, or they can simply describe the passage of time.

A system is called blocking if there is a state $x \in X$ from which no further transitions are possible, i.e., x has no u-successors for any $u \in U$. This can also be expressed as $U(x) = 0$. A system is called non-blocking if the set of successors of every $x \in X$ is nonempty. An equivalent characterization is $U(x) \neq 0$ for every $x \in X$.

A system is called deterministic if for any state $x \in X$ and any input $u \in U$, (x, u, x') and (x, u, x'') imply $x' = x''$. Therefore, a system is deterministic if given any state $x \in X$ and any input $u \in U$, there exists at most one u-successor (there may be none). A system is output deterministic if: $H|_{X_0}$ is injective; and for any state $x \in X$ and any inputs $u, u' \in U, (x, u, x')$ and (x, u, x'') with $H(x') = H(x'')$ imply $x' = x''$. For output deterministic systems, different successors of a state always have different outputs. A system is called non-deterministic if it is not deterministic. Hence for a non-deterministic system it is possible for a state to have two (or possibly more) distinct u-successors.

If we want to explicitly refer to the possible sequences of states and outputs that a system can generate, we make use of the so called system behaviors.

Definition 2.2 (Internal behavior). For a system S and given any state $x \in X$, a finite internal behavior generated from x is a finite sequence of transitions:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \dots x_{n-1} \xrightarrow{u_n-1} x_n$$

such that $x_0 = x$ and $x_i \xrightarrow{u_i} x_{i+1}$ for all $0 \leqslant i \leqslant n$. A finite internal behavior generated from x is initialized if $x \in X_0$.

An infinite behavior generated from x is an infinite sequence of transitions:

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \dots$$

such that $x_0 = x$ and $x_i \xrightarrow{u_i} x_{i+1}$ for all $i \in N$. An infinite internal behavior generated from x is initialized if $x \in X_0$.

The sequence of outputs that are caused internally are called external behaviors and are defined as follows.

Definition 2.3 (External behavior). For a system S and given any state $x \in X$, every finite internal behavior

$$x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} x_2 \xrightarrow{u_2} \dots x_{n-1} \xrightarrow{u_{n-1}} x_n$$

defines a finite external behavior through the map H:

$$y_0 \to y_1 \to y_2 \to \dots y_{n-1} \to y_n$$

with $H(x_i) = y_i$ for all $0 \leqslant i \leqslant n$. Similarly, every infinite internal behavior defines an infinite external behavior:

$$y_0 \to y_1 \to y_2 \to \dots$$

with $H(x_i) = y_i$ for all $i \in N$. The external behavior is initialized if the corresponding internal behavior is initialized.

The set of external behaviors that are defined by internal behaviors generated from state x is denoted by $\mathcal{B}_x(S)$ and is called the external behavior from state x. It should be noted that for output deterministic systems any finite external behavior y determines uniquely the corresponding internal behavior.

Also, infinite behaviors describe the non-terminating interaction of a system with other systems and the environment. They are, thus, adequate to model the operation of reactive systems, such as embedded controllers, that must operate without interruption for arbitrarily long periods of time.

2-2-3 Temporal Logic

Temporal logic is a logical formalism that extends propositional and predicate logic by modal operators that allow describing infinite behavior of systems. Temporal logic is well suited for reactive systems, where the correctness of operation depends on the timeliness of the behavior in addition to input/output computation. In temporal logic, the underlying nature of time can be either linear or branching. These different views of time in addition to other factors gave rise to different flavors of temporal logic that are collectively referred to as temporal logics. Note that time here is referred to in the abstract sense, in which time is discrete and advances as the system evolves from one state to the next. Linear Temporal Logic (LTL) is a subset of the powerful logic called Computational Tree Logic (CTL). It was first introduced by Pnueli [30], and it is a logic for specifying temporal properties for reactive and concurrent systems. Intuitively, an LTL formula specifies a property that should apply to all behaviors starting from the initial set [24].

In LTL, formulas are composed of temporal operators and logical operators. The alphabet of LTL is defined as follows [12]:

- Atomic proposition symbols, such as p, q, r
- Logical connectives: $\vee$ (or), $\neg$ (not)
- Temporal connectives: X or $\bigcirc$, U or $\mathcal{U}$

The syntax of LTL is defined below [30]. LTL formulas over the set of atomic propositions AP is generated by the following grammar:

$$\phi ::= true|\alpha|\phi_1 \vee \phi_2|\phi_1 \wedge \phi_2|\Diamond\phi|\bigcirc\phi|\phi_1\mathcal{U}\phi_2$$

where $\alpha \in AP$

As for the precedence, the unary operators bind stronger than the binary ones and the *until* operator takes precedence over the propositional logic operators ($\vee$, $\wedge$, $\neg$). The parentheses have the highest precedence, while $\neg$ and $\bigcirc$ (next) bind equally strong. Intuitively, $\Diamond\phi$ ensures that ϕ will finally be true in future, while $\Box\phi$ ensures that ϕ will always be true from now and forever in future.

By combining the above operators new temporal modalities can be obtained, such as $\Box\Diamond$ for "infinitely often" or $\Diamond\Box$ for "eventually always". LTL is especially suited for expressing and verifying important properties of symbolic controllers, such as safety and reachability [29]. In a transition system, such as a symbolic controller, a state is called reachable if there is a computation path from a defined initial state leading to this state. Reachability is one of the most important properties of transition systems in connection with safety properties. Suppose that u is a formula which expresses an undesirable property of a transition system. States satisfying u are usually called unsafe or bad. Naturally, one would like to know whether the system is safe [23]. Reachability of a state satisfying u can be expressed as the existence of a path satisfying $\Diamond u$. Then the safety of the system can be expressed as non-reachability of a state satisfying u, i.e. $\Box\neg u$. The transition system is safe if this property is held on all computation paths [31].

MITL formulas extend the LTL(Linear Temporal Logic) to incorporate more complex system behaviors. These include the addition of a notion of metric or distance/depth to the existent discrete time concepts. Metric Temporal Logic (MTL) and Metric Interval Temporal Logic (MITL) are extensions designed to handle dense time logics. In essence, MITL can be defined as a linear temporal logic that is characterized by timed state sequences [39].

2-2-4 System Relations

A very important relation between two systems can be illustrated by system relationships. The following definitions are also taken from [25].

Behavioral inclusion is important for abstracting a system from another one or for verifying desired behavior containment. However, for infinite-state systems it is difficult, from a technical point of view, to work directly with behaviors. On the contrary, simulation relationships are stronger than their behavioral counterparts and are defined below.

Definition 2.4 (Simulation Relation). Consider systems S_a and S_b with $Y_a = Y_b$. A relation $R \subseteq X_a \times X_b$ is a simulation relation from S_a to S_b if the following three conditions are satisfied:

- for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
- for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
- for every $(x_a, x_b) \in R$ we have that:
  $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b satisfying $(x'_a, x'_b) \in R$.

Intuitively, a simulation relation $R \subseteq X_a \times X_b$ captures which states of S_a are simulated by which states of S_b. Therefore, if $(x_a, x_b) \in R$, then state x_a of system S_a is simulated by state x_b of system S_b. There is a clear order between the two systems as system S_b simulates system S_a.

Definition 2.5 (Bisimulation Relation). Given two systems S_a and S_b with $Y_a = Y_b$, we consider that S_a is bisimilar to S_b, denoted by $S_a \cong_S S_b$, if there exists a relation R satisfying:

1. R is a simulation relation S_a to S_b;
2. R^{-1} is a simulation relation S_b to S_a.

Based on a simulation relation Q, we can construct a quotient system $S_{/Q}$ which simulates S by construction. This can be seen by taking π_Q as the required simulation relation. The quotient system $S_{/Q}$ is also called a symbolic model of S since each state $x_{/Q} \in X_{/Q}$ can be regarded as a symbol representing all the states $\pi_Q^{-1}(x_{/Q})$ of the original system.

For problems of control, a different kind of similarity relationship is needed. Simulation relations require the matching of transitions while in problems of control we require the existence of inputs enforcing desired transitions. We thus need a similarity relationship that captures the effect that different choices of inputs have on transitions.

Definition 2.6 (Alternating simulation relation). Consider systems S_a and S_b with $Y_a = Y_b$. A relation $R \subseteq X_a \times X_b$ is an alternating simulation relation from S_a to S_b if the following three conditions are satisfied:

- for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
- for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
- for every $(x_a, x_b) \in R$ and for every $u_a \in U_a$ there exists $u_b \in U_b(x_b)$ such that for every $x'_b \in \mathrm{Post}_{u_b}(x_b)$ there exists $x'_a \in \mathrm{Post}_{u_a}(x_a)$ satisfying $(x'_a, x'_b) \in R$.

Let's continue with an example [25]. In the next figure two systems are considered.

It can be readily seen that the relation

$$R = \{(x_{a0}, x_{b0}), (x_{a1}, x_{b1}), (x_{a1}, x_{b2})\}$$

is a simulation relation from S_a to S_b, but not an alternating simulation. While, the relation

$$R' = \{(x_{a0}, x_{b0}), (x_{a1}, x_{b1}), (x_{a2}, x_{b2}), (x_{a1}, x_{b3})\}$$

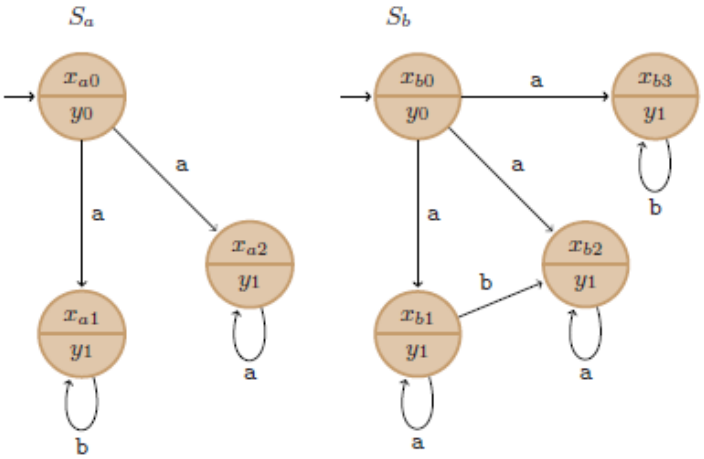

Figure 2-7: Example of similarity relationships [25]

is an alternating simulation relation but not a simulation relation from S_a to S_b. Although alternating simulation is substantially different from simulation, these two notions coincide in the very special case of deterministic systems.

At this point, it should be mentioned that the algorithmic construction of similarity relations may create computational problems. However, the existence of simulation or bisimulation relations between two finite-state systems can be studied through the fixed-points of certain operators. The perspective offered by fixed-points of operators is advantageous on two counts. Operators are a convenient mathematical abstraction allowing us to study correctness and termination of algorithms without being distracted by the implementation details. The second advantage is the possibility of implementing symbolically the algorithms defined by fixed-points. This means that explicit enumeration of the states is avoided by manipulating instead succinct representations for sets of states.

The fixed point operator closely mirrors the definition of simulation relation and is based on the Tarksi lemma and lattice theory [42, 40].

Intuitively, the fixed point operator finds the maximal simulation relation from S_a to S_b based on an iterative procedure. First it considers all the possible state combinations, then it excludes all the combinations with different outputs and finally it iterates until the transition requirements are satisfied [41].

Taking as an example the two systems of fig. 2-7, the fixed-point algorithm is shown below.

2-2-5 Control Synthesis

This section is based on [25].

Whenever a system S_a fails to conform to its specification S_b, in the sense that $S_a \not\preceq S_b$, the goal is to find out if there exists a controller such that $S_c \times_{\mathcal{I}} S_a \preceq S_b$ or even $S_c \times_{\mathcal{I}} S_a \cong S_b$. The controller synthesis problem can be solved by computing fixed-points of suitable defined operators. In particular, we will focus on fixed-point solutions specialized for safety and reachability control problems that arise in applications.

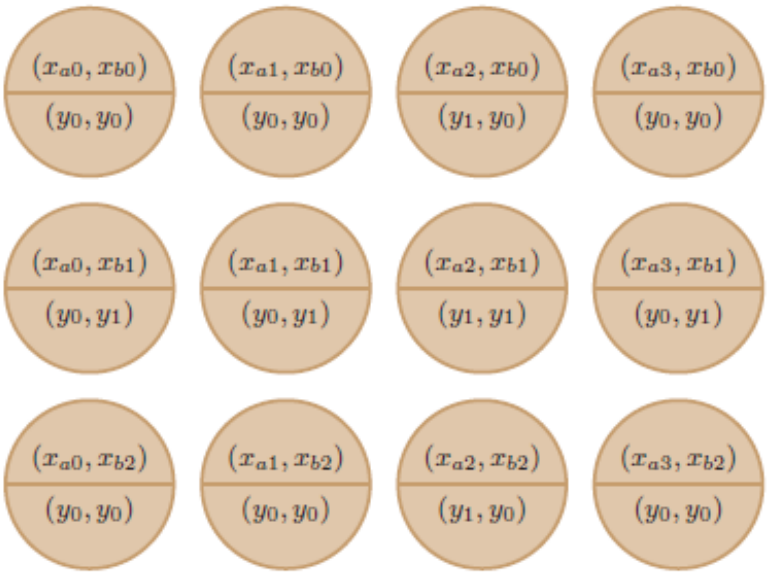

Figure 2-8: States in the set $X_a \times X_b$ over which the operator F acts [25].

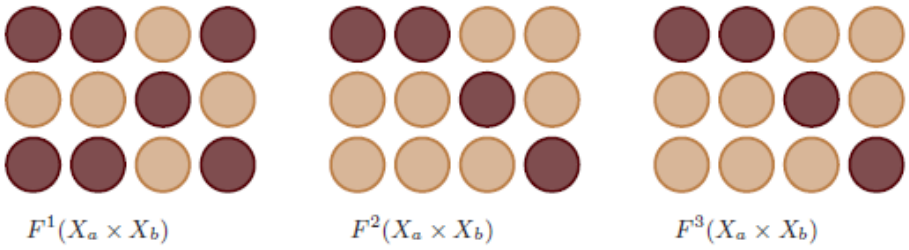

Figure 2-9: Computation of simulation relation from S_a to S_b. The image of the operator F is represented by the dark-colored states [25].

The notion of controller can be formalized in different ways. A controller could be considered as a mechanism that determines which input should be fed into the system being controlled on observed states. However, there exists a limitation that there could be more than one input that lead to a correct or desirable behavior. In this respect, the notion of controllers should be revised to a mechanism that determines which inputs can be fed to the controlled system based on a sequence of observed outputs.

Definition 2.7 (Feedback composition). A system S_c is said to be feedback composable with a system S_a if there exists an alternating simulation relation from S_c to S_a. When S_c is feedback composable with S_a, the feedback composition of S_c and S_a , with interconnection relation $\mathcal{F} = R^e$, is given by $S_c \times_{\mathcal{F}} S_a$.

We can thus interpret an internal behavior of $S_c \times_{\mathcal{F}} S_a$ as the result of a feedback process during which the controller offers a set of inputs, measures the states of S_a, updates its own state, offers again a new set of inputs based on the updated state, and so on.

For the example of next figure [25], we want to eliminate all the internal behaviors of the form $x_{a1} \xrightarrow[\alpha]{a}$ or of the form $x_{a0} \xrightarrow[\alpha]{b}$. This objective is achieved by using the controller S_c represented in the second part of the figure. The required alternating simulation relation is given by:

$$\{(x_{c0}, x_{a0}), (x_{c1}, x_{a1}), (x_{c2}, x_{a2})\}$$

. The composed system is also depicted in the figure.

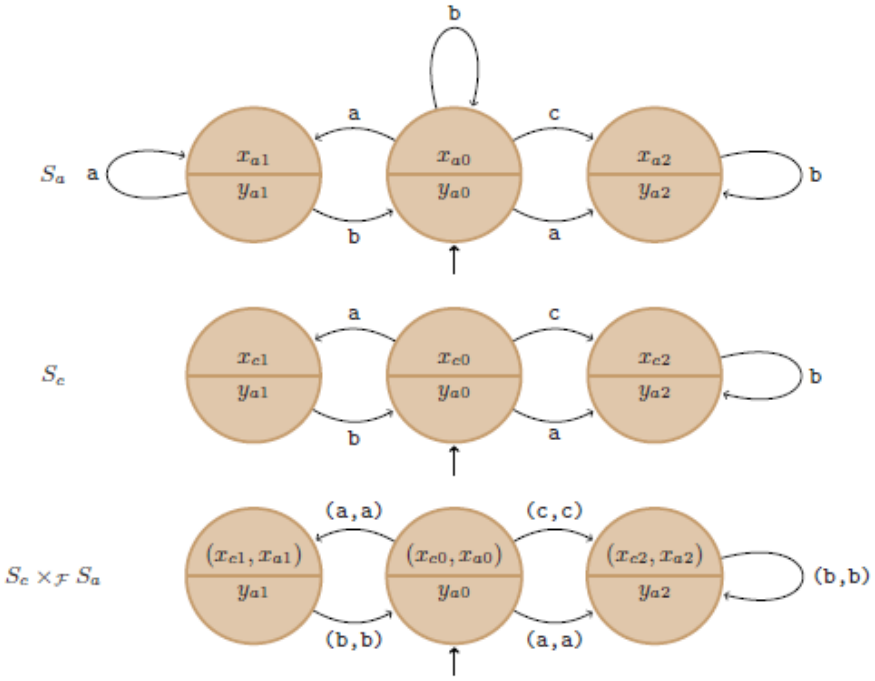

Figure 2-10: From top to bottom: system S_a, controller S_c and feedback composed system $S_c \times_{\mathcal{F}} S_a$ [25].

Continuing with controller synthesis for safety specifications, a controller can be synthesized by a given fixed point of a mathematical operator. In the next figure, a finite-state system is shown, with the objective to avoid the transition to the state x_{a2} (dark colored). The least restrictive controller is found after five iterations.

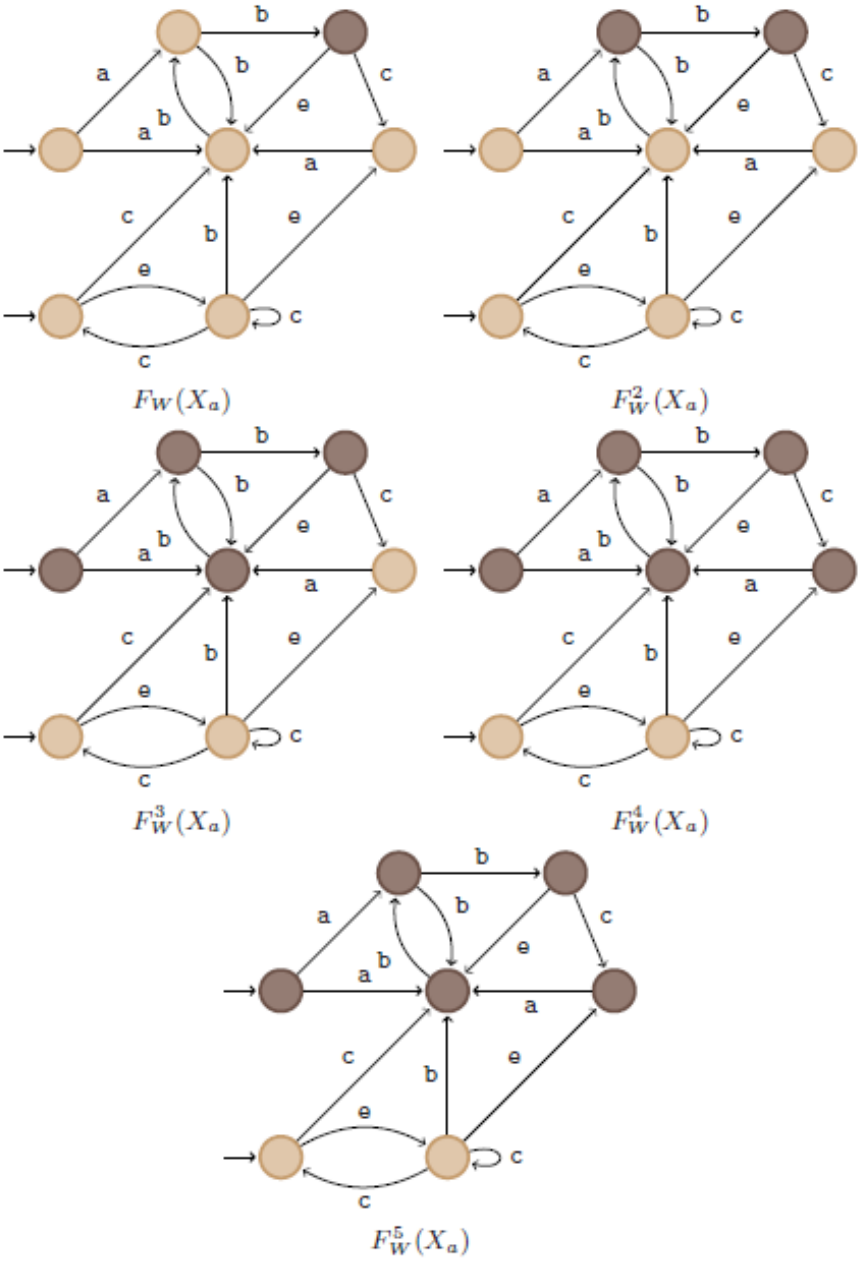

Figure 2-11: Controller solving safety game by maximal fixed-point [25]

While the objective of safety games is to keep the behaviors of the composed system within a safe set, reachability games require a certain set W of outputs to be reached. Now by using the minimal fixed point of a suitably selected operator, the control problem can be solved.

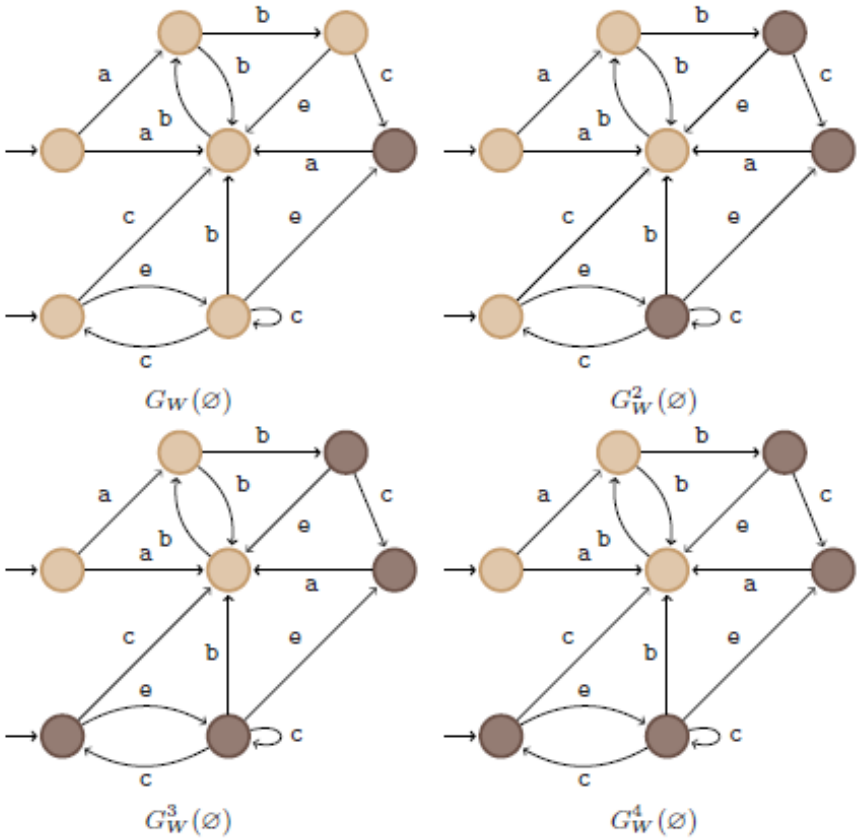

Figure 2-12: Controller solving reachability game by minimal fixed-point [25]

2-3 Tools for Synthesis and Verification of Hybrid Systems

In this section, the main tools that are used in the thesis are described. More information about the existing tools is available on the Appendix.

2-3-1 PESSOA

Pessoa is a tool for the synthesis of correct-by-design embedded control software [32]. Pessoa relies on approximate abstractions of control systems to reduce the synthesis of control software to the synthesis of reactive controllers for finite-state models. Pessoa offers three main functionalities:

- the construction of finite symbolic models of linear control systems,
- the synthesis of symbolic controllers for simple specifications,
- simulation of the closed-loop behavior in Simulink.

Nonlinear and switched dynamics can also be used in Pessoa, albeit not natively [34]. All the transition relations $\rightarrow$ of the abstractions generated by Pessoa, and the sets used in the

specifications, are stored as Reduced Order Binary Decision Diagrams (ROBDD) through their corresponding characteristic functions. The usage of ROBDDs enables the efficient computation of reachable sets and pre-images of sets, both fundamental operations in the design of controllers. Pessoa supports the synthesis of controllers enforcing four kinds of specifications defined using a target set $Z \subseteq X$ and a constraint set $W \subseteq X$:

1. **Stay**: trajectories start in the target set Z and remain in Z. This specification corresponds to the Linear Temporal Logic (LTL) formula $\Box\phi_Z$ where ϕ_Z is the predicate defining the set Z,
2. **Reach**: trajectories enter the target set Z in finite time. This specification corresponds to the LTL formula $\Diamond\phi_Z$,
3. **Reach and Stay**: trajectories enter the target set Z in finite time and remain within Z thereafter. This specification corresponds to the LTL formula $\Diamond\Box\phi_Z$,
4. **Reach and Stay while Stay**: trajectories enter the target set Z in finite time and remain within Z thereafter while always remaining within the constraint set W. This specification corresponds to the LTL formula $\Diamond\Box\phi_Z \wedge \Box\phi_Z$ where ϕ_W is the predicate defining the set W.

The controllers for the above specifications are memory-less controllers that can be synthesized through fixed point computations. Furthermore, the finite state nature of the synthesized controllers permits a direct compilation into code. Closed-loop simulation in Simulink is possible. For this purpose, Pessoa comes with a Simulink block implementing a refinement of any synthesized controller. The controllers synthesized in Pessoa are, in general, non-deterministic. The Simulink block resolves this non-determinism in a consistent fashion thus providing repeatable simulations. In order to increase the simulation speed, the Simulink block selects, among all the inputs available for the current state, the input with the shortest description in the ROBDD encoding the controller. Moreover, the input is chosen in a lazy manner, i.e., the input is only changed when the previously used input cannot be used again [33].

2-3-2 Multi-Parametric Toolbox

The Multi-Parametric Toolbox (MPT) is a software tool for Matlab that aims at solving parametric optimization problems that arise in constrained optimal control [36, 35]. In particular, its primal objective is to provide computationally efficient means for design and application of explicit model predictive control (MPC). MPT is also one of the tools that combines computational geometry with control routines. MPT has integrated the ellipsoidal toolbox, SEDUMI toolbox [38] as well as YALMIP [37] which provides a high level language for modeling and formulating optimization problems. The content of MPT can be divided into four modules:

- modeling of dynamical systems,
- MPC-based control synthesis,
- closed-loop analysis,

- deployment of MPC controllers to hardware.

Each part represents one stage in design and implementation of explicit MPC. The modeling module of MPT allows the description of discrete-time systems with either linear or hybrid dynamics. The latter can be directly imported from the HYSDEL environment. The control module allows to formulate and solve constrained optimal control problems for both linear and hybrid systems. The analysis module provides methods for investigation of closed-loop behavior and performance. Moreover, it features methods to reduce complexity of explicit MPC feedback loops. Most importantly, MPT is useful for the verification of safety[1] and liveness[2] properties of hybrid systems, while it facilitates reachability and invariant set computations. The deployment part allows to export control routines to C language, which can be subsequently downloaded to a target hardware implementation platform [36].

2-3-3 Breach

Breach is a toolbox for verification and parameter synthesis of hybrid systems. It provides a coherent set of simulation-based techniques aiming at the analysis of deterministic models of hybrid dynamical systems. The primary feature of Breach is to facilitate the computation and the property investigation of large sets of trajectories. It relies on an efficient numerical solver of ordinary differential equations that can also provide information about sensitivity with respect to parameters variation. The latter is used to perform approximate reachability analysis and parameter synthesis. A key feature is the robust monitoring of metric interval temporal logic (MITL) formulas. Simulation-based techniques replace extremal points and sets, by finitely many transitions. Especially, for nonlinear systems whose dynamics don't preserve convexity or a clean mathematical model [54].

[1] A safety property states that something bad never happens.

[2] A liveness property states that something good should eventually happen. "Any specification can be expressed as the conjunction of a safety property and a liveness property."

2-4 Braking Systems

2-4-1 Tire Forces

Let's start with a short description of a vehicle tire. The tire is the single element that most affects the dynamic behavior and performance of a road vehicle. With the exception of aerodynamic forces, the tire is the source of all the forces acting on the vehicle. Forces are generated at the contact patch [17]. An illustrative figure of the tire-road interaction is shown below.

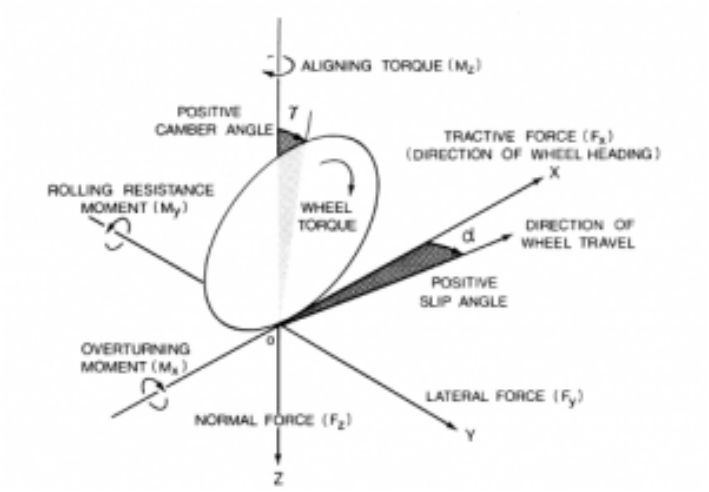

Figure 2-13: Tire axis system [16]

There are:

- 3 forces (longitudinal, lateral, vertical[3])
- 3 moments (overturn, roll, yaw)
- 3 angles (roll, pitch and yaw)

In terms of vehicle dynamics we are mainly interested in the longitudinal and lateral forces. Braking control is also self-consistent and meaningful when the analysis is restricted to longitudinal dynamics. The analysis is also focused on straight line braking and does not extend to cornering.

In the literature, three vehicle models are mainly used [16].

- Single Corner Model (Quarter Car Model)
- Single Track Model (Double Corner Model)
- Full Car Model

[3]Vertical forces are also be called as normal forces.

For the design and testing of braking control algorithms, the simple but effective single-corner model is typically used [18]. The individual control of each wheel is also vital in the case of autonomous driving. Autonomous vehicles are safety critical systems and the use of braking controllers for front or rear axles is not sufficient for performance guarantees.

2-4-2 Braking Model

The single corner model is sufficient to describe the main phenomena affecting the wheel dynamics.

The model is illustrated in the next figure.

Figure 2-14: Single Corner Model [16]

The dynamical equations of motion are given by:

$$J\dot{\omega} = rF_x - T_b \tag{2-2a}$$
$$m\dot{v} = -F_x \tag{2-2b}$$

where

- ω: angular speed of the wheel [rad/s]
- v: longitudinal speed of the vehicle body [m/s]
- T_b: braking torque (control/input variable)
- F_x: longitudinal road-tire contact force [N]
- F_z: vertical road-tire contact force [N]
- J, m, r: inertia of the wheel, single corner mass and wheel radius

The effect of aerodynamic drag force is not modeled and the braking maneuver is supposed to happen on a road with no grade.

Modeling the friction force F_x is the main difficulty in the model [2a]-[2b]. The tire forces are nonlinear functions and depend on a large number of features of the road, tire, and suspensions. Most often, they can be described as:

$$F_x = \mu_x(\lambda, \alpha, \gamma, F_z) \approx \mu_x(\lambda, \alpha, \gamma)F_z \tag{2-3a}$$
$$F_y = \mu_y(\lambda, \alpha, \gamma, F_z) \approx \mu_y(\lambda, \alpha, \gamma)F_z \tag{2-3b}$$

where

- γ is the camber angle, i.e. the tire inclination with respect to the vertical direction
- F_z is the vertical force at the tire-road contact point. In quasi-static conditions, it can be simply described as $F_z = mg$, where the mass m corresponds to the mass of each wheel, based on the load distribution of the vehicle. However, in a braking scenario F_z can significantly change due to dynamic load transfer.
- α is the side-slip angle, which is the angle between the tire longitudinal axis and the speed vector of the contact point.
- μ_x and μ_y are the so called longitudinal and lateral friction coefficients. For the case of linear relationship between $F_{x,y}$ with F_z, it holds that

  $$\mu_x(\lambda, \alpha, \gamma) = \frac{F_x}{F_z} \tag{2-4a}$$
  $$\mu_y(\lambda, \alpha, \gamma) = \frac{F_y}{F_z} \tag{2-4b}$$

  In fact, this assumption does not hold for very large values of the vertical load, where the relationship between the forces and the vertical load shows a saturation. Considering only straight line braking allows us to assume negligible load transfer and zero side-slip angle. As a result,

  $$F_x = \mu_x(\lambda)F_z \tag{2-5}$$

- λ is the longitudinal slip, which is the normalized difference between the vehicle velocity and the equivalent wheel linear velocity, in case of zero tire side-slip angle is defined as

  $$\lambda = \frac{v - \omega r}{\max\{v, \omega r\}}$$

  where v is the axle velocity, r is the tire rolling radius and ω is the wheel angular velocity.

  Since we are focused on braking maneuvers, the vehicle speed will be larger than the wheel speed. So,

  $$\lambda = \frac{v - \omega r}{v} \tag{2-6}$$

  Given that $v \geqslant \omega r$, $\lambda \in [0, 1]$. In addition, $\lambda = 0$ results in pure rolling and $\lambda = 1$ to a locked wheel.

Substituting [2.5] - [2.6] into set of equations [2.2], the equivalent system is

$$J\dot{\omega} = rF_z\mu\left(\frac{v-\omega r}{v}\right) - T_b \tag{2-7}$$

$$m\dot{v} = -F_z\mu\left(\frac{v-\omega r}{v}\right) \tag{2-8}$$

As a next step, the friction coefficient μ should be modeled. The most popular approaches are based on two types of models: empirical and first principle (analytical) [71]. Empirical models are based on curve fitting and real time experiments [70]. Well known tire friction models are the Magic Tyre Formula of Pacejka and the Burckhardt model [69]. First principle models are based on approximate modeling of the physical mechanisms that generate the force. One of the most renowned models is the Dahl model [72]. The most accurate one is the LuGre model which incorporates the Stribeck effect [70]. If the friction model captures the transient behavior of the tire-road contact forces under time varying velocity conditions, it is considered to be a dynamic model. Otherwise, it is considered to be a static one. Based on tire road interaction, the friction models are divided into lumped and distributed. On the one hand, lump models assume a point tire-road friction contact. On the other hand, distributed models assume the existence of a contact patch between the tire and the ground with an associated normal pressure distribution [71].

In the framework of this thesis, the static Burckhardt model is selected, since it offers accuracy and has a relatively simple structure. The friction coefficient is represented by the following relation.

$$\mu(\lambda) = c_1(1 - e^{-c_2\lambda}) - c_3\lambda \tag{2-9}$$

where c_1, c_2, c_3 are model parameters, which include information on the road adhesion conditions and usually vary in time. Typical values of these parameters are given in the next table [16].

Table 2-1: Road condition vs Friction characteristic parameters

Road Condition	c_1	c_2	c_3
Dry Ashpalt	1.28	23.99	0.52
Wet Asphalt	0.86	33.82	0.35
Cobblestone	1.37	6.46	0.67
Snow	0.19	94.13	0.066

2-4-3 Actuation and Sensing

Commercial vehicles make use of friction brakes. They are divided into disc and drum brakes [19].

A disc brake is composed of 3 elements:

- disc / rotor (rotates with the wheel)

- brake pads (generate friction with disc)
- brake caliper (apply the force to press brake pads against the disc)

Disc brakes generate larger braking torque, more easily controllable, better heat dissipation. On the other hand disc brakes, form old fashioned solutions.

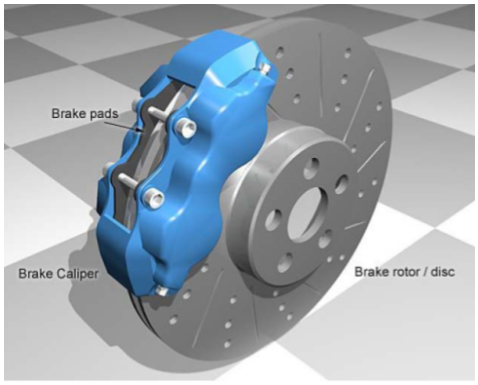

Figure 2-15: Disc Brake
source: www.carbibles.com/brake_bible.html

The braking torque that is generated by friction at the brake disk is given by [17]:

$$T_b = r_d c_f A p_b \tag{2-10}$$

- r_d is the brake disk radius
- c_f is the brake pad friction coefficient
- A is the brake pad contact area
- p_b is the piston pressure

The braking hydraulic system has proven itself to be reliable and efficient. Nevertheless braking control systems require the possibility of automatically control the braking pressure [17]. Control is achieved in three main ways:

- Hydraulic Actuated Brakes (HAB)
- Electro Hydraulic Actuated Brakes (EHB)
- Electro Mechanical Brakes (EMB)

For the context of this thesis, electro-mechanical brakes (brake-by-wire) are considered. Electro-mechanical brakes replace traditional hydraulic and mechanical components, such as pumps, hoses, fluids, belts, master cylinders, mechanical links, with electronic sensors and actuators [19].

Graphically,

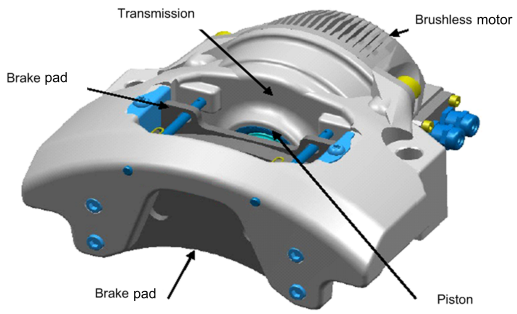

Figure 2-16: Components of an EMB system
source: www.carbibles.com/brake_bible.html

Brake by wire technology has several advantages in terms of response, torque distribution; it frees up space, weight and does not require brake oil. The most important advantage though is relate to continuous force modulation. That means that the braking pressure can be directly controlled, in comparison with traditional hydraulic actuating brakes where the rate of change of braking pressure can be controlled [16]. Traditional hydraulic brakes correspond to $\frac{dp_b}{dt} = u$, while with EMB it is possible to say $p_b = u$. In order to simplify the the control design, the actuator is assumed to have a negligible response time. That is a fair assumption in case that actuator bandwidth is sufficiently large or in-wheel electric motors are used [73].

Given the linear relationship between braking torque and pressure as well as the aforementioned characteristics of brake-by-wire, the control input will be considered to be

$$u = T_b \tag{2-11}$$

As for sensing, we will consider that a wheel velocity sensor and a body velocity sensor are available. An example of wheel velocity sensor could be an encoder, while for body velocity measurement an optical encoder is considered. In practical settings, optical encoders are not used in production, since they are very expensive and fragile. As a result, vehicle linear velocity is not directly measured but is reconstructed with linear and nonlinear observers [**?**].

2-4-4 Notes

To sum up, the single-corner model relies on the following simplifications [16, 74]:

- The four wheels are treated as dynamically decoupled, so the dynamic load transfer and pitch motion are ignored.
- The suspension dynamics are neglected.
- The dependence of friction forces from the vertical load is modeled as a proportionality relation. This is the case only for quasi-static conditions.
- The wheel radius is assumed to be constant. However, during braking there is a dynamic change in the wheel radius, which is a function of the instantaneous vertical load.

- Straight-line braking is considered, so the dependence on friction forces on γ and α_t is neglected.

Despite these simplifications, the single-corner model is the most widely used for braking control systems design. It provides a sufficiently rich description of the braking dynamics. It should be taken into account that despite its simpler structure, the control design is a challenging task. That is related to the the inherent nonlinearities as well as to the fact that the road conditions and the values of vertical load are unknown and may change rapidly.

2-5 Problem Statement - Control Objectives

In this section, the problem statement will be formulated as control objectives. Let's recall our objectives:

1. Reduction of stopping distances
2. Stability Improvement
3. Steerability during braking

In this context, the goal is to minimize the braking distance of a fully automated vehicle, while maintaining its driveability. As it can be seen from the braking dynamics,

$$\dot{v} = -\frac{F_x}{m} = -\frac{F_z}{m}\mu(\lambda)$$

That means that the vehicle deceleration is related to the wheel slip and the road conditions. As such, for every road condition there exists a slip value that maximizes the deceleration. So, the purpose of minimizing braking distance can seen as a regulation problem of the wheel slip around its optimal value. The wheel slip control has a straight forward graphical interpretation (figure 2-17).

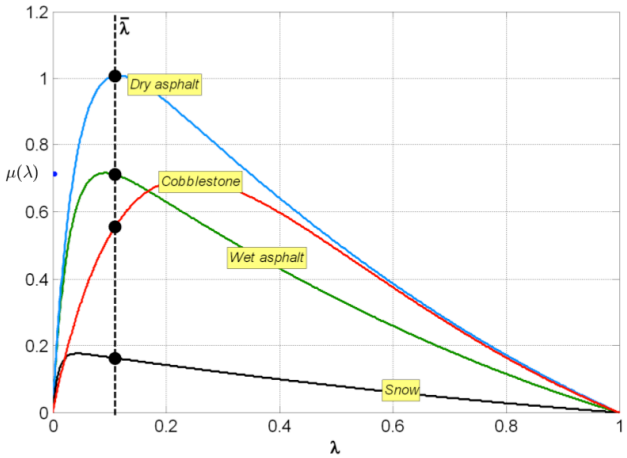

Figure 2-17: Graphical interpretation of wheel slip control

It should be noted that for every value of the set-point λ, the regulation scheme guarantees the uniqueness of the equilibrium point. Also, under the condition that $0.1 \leqslant \lambda \leqslant 0.2$ the friction coefficient is close to its optimal value for every road condition. That issue is crucial since it allows the use of a fixed structure controller, with no need for online identification and detection of the road conditions.

As for the steerability of the vehicle, it is directly related to the lateral forces that are exerted on the wheel during braking [69]. On the condition that the wheel of the vehicle is locked up and doesn't rotate ($\omega = 0$), it is impossible to avoid uncontrolled skidding. The lateral friction coefficient of a locked up wheel is zero, resulting in zero lateral force ($F_y = \mu_y * F_z$) and loss of controlled steering. A graphical interpretation of lateral friction coefficient and wheel slip, based on empirical Pacejka model, is shown in the next figure.

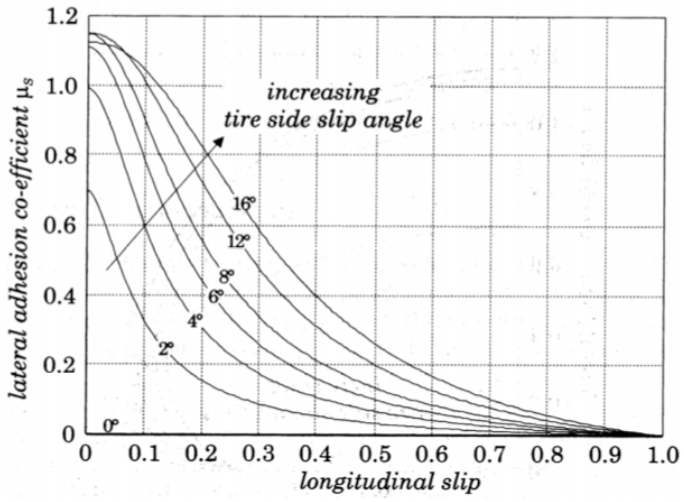

Figure 2-18: Slip vs lateral friction coefficient [16]

As for stability, with a simple fixed structure controller (i.e. P controller), the asymptotically stability of the closed-loop is guaranteed for every value of $\bar{\lambda}$ and for every road condition. That can be seen by considering the linearized model around the equilibrium $(\bar{\lambda}, \bar{v})$ [16].

The open loop transfer function is:

$$G_\lambda = \frac{\left[\frac{r}{\bar{v}J}\right]}{s + \left[\frac{\mu_1(\bar{\lambda})F_z}{m\bar{v}}\left((1-\bar{\lambda}) + \frac{mr^2}{J}\right)\right]} \tag{2-12}$$

Then, by simply closing the loop and considering a pole with negative real part, the stability condition becomes:

$$K > -\frac{\mu_1(\bar{\lambda})F_z}{m\bar{v}}\left((1-\bar{\lambda}) + \frac{mr^2}{J}\right)$$

To conclude with, wheel slip control offers a choice λ set point that yields good results for any surface, avoids wheel lock and guarantees stability.

Chapter 3

Modeling

In this chapter, different modeling representations of braking systems are presented. The aim is to arrive at a simplified model, which captures the main phenomena and provides a more manageable mathematical basis.

3-1 Slip Dynamics

Let's recall the dynamic model of a braking system. As shown in the previous chapter, the following equations hold.

$$\begin{aligned} J\dot{\omega} &= rF_z\mu(\lambda) - T_b \\ m\dot{v} &= -F_z\mu(\lambda) \\ \lambda &= \frac{v-\omega r}{v} \\ \mu(\lambda) &= c_1(1-e^{-c_2\lambda}) - c_3\lambda \end{aligned} \tag{3-1}$$

This model is based on wheel dynamics, and its state variables are v, ω. The parameters r, F_x, r, m, J describe physical quantities of the vehicle, while c_1, c_2, c_3 relate to the road conditions.

Considering the control objectives that were described in previous section, it is necessary to reformulate the aforementioned model into a different form. Since, we want to regulate the slip, it is plausible to replace the state variable ω with the state variable λ.

Then,

$$\dot{\lambda} = \frac{d}{dt}\left(\frac{v-\omega r}{v}\right) = \frac{(\dot{v}-\dot{\omega}r)v - (v-\omega r)\dot{v}}{v^2} = -\frac{r}{v}\dot{\omega} + \frac{r\omega}{v^2}$$

So, the equivalent expression becomes:

$$\dot{\lambda} = -\frac{1}{v}\left(\frac{1-\lambda}{m} + \frac{r^2}{J}\right) F_z \mu(\lambda) + \frac{r}{vJ} T_b \tag{3-2a}$$

$$\dot{v} = -\frac{F_z}{m}\mu(\lambda) \tag{3-2b}$$

$$\mu(\lambda) = c_1(1 - e^{-c_2\lambda}) - c_3\lambda \tag{3-2c}$$

Given that $v \leqslant \omega r$, $\lambda \in [0, 1]$. In addition, $\lambda = 0$ results in pure rolling and $\lambda = 1$ to a locked wheel.

Following the previous analysis on actuator and sensor technology, the measurable outputs are v and ω, and the input of the system is the braking torque. $u = T_b$.

3-2 Analysis

In this section, different slip-based dynamical models are designed, and compared against original wheel-based dynamical model.The benchmark conditions for the analysis will be wet asphalt road. The physical parameters of the vehicle are shown in the next table.

Table 3-1: Vehicle parameters for braking maneuvers

Paramaters	Values
c_1	1.28
c_2	0.86
c_3	1.37
m	225
J	1.0
r	0.28
g	9.8
F_z	mg

The simulations have been executed under the assumption of constant braking torque $T_b = 550$Nm, $v_0 = 100$ km/h and $\lambda_0 = 0.01$.

Thereafter, a comparison of wheel dynamics and slip dynamics is made. It can be readily seen that the models are equivalent, even though the termination conditions are different.

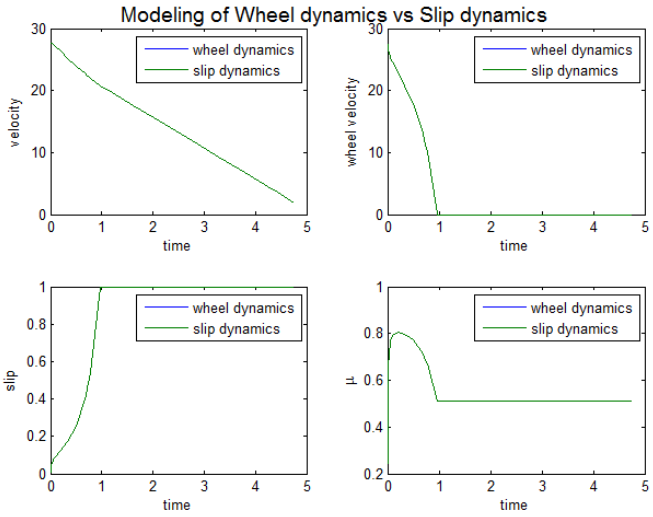

Figure 3-1: Simulation of slip dynamics vs wheel dynamics

For slip dynamics, $\lambda \in [0,1], v \leqslant v_{off}$ while for wheel dynamics $\omega \leqslant 0, v \leqslant v_{off}$. The variable v_{off} describes the minimum velocity value for which the braking control is active.

Given that $1-\lambda \ll m$ and $\left(\frac{1-\lambda}{m} + \frac{r^2}{J}\right) \approx \frac{r^2}{J}$, it could be plausible to neglect the first term. Then,

$$\dot{\lambda} = -\frac{1}{v}\frac{r^2}{J}F_z\mu(\lambda) + \frac{r}{vJ}T_b \tag{3-3}$$

$$\dot{v} = -\frac{F_z}{m}\mu(\lambda) \tag{3-4}$$

As for the simpler slip model, it is tested against the original one.

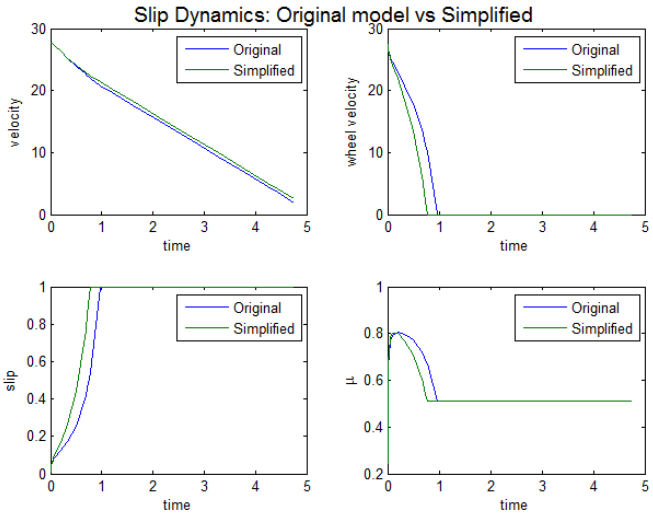

Figure 3-2: Simulation of slip dynamics vs simpler slip dynamics

A deviation can be seen, but the difference is small and the resulting dynamical behavior is slightly altered. The modified model, with friction is shown below.

$$\dot{\lambda} = -\frac{1}{v}\frac{r^2}{J}F_z(c_1(1-e^{-c_2\lambda}) - c_3\lambda) + \frac{r}{vJ}T_b \tag{3-5}$$

$$\dot{v} = -\frac{F_z}{m}(c_1(1-e^{-c_2\lambda}) - c_3\lambda) \tag{3-6}$$

The model is highly nonlinear, so it is difficult to implement analytical techniques. In this respect, the nonlinear friction curve μ is approximated by piecewise linear functions. The original Burckhardt friction curve is shown in the next figure for different surfaces.

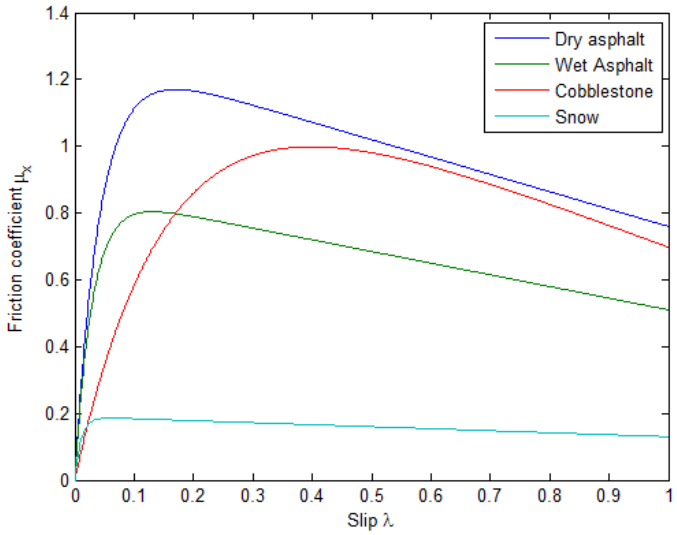

Figure 3-3: Friction curves for different surfaces - Burckhardt model

The focus will be on wet asphalt, but the same methodology can be applied to other cases. In particular, it is proposed to approximate the nonlinear curve μ by Piecewise Linear Function (PLF) divided into N sections:

$$\mu(\lambda) = \mu_q(\lambda) = \begin{cases} \mu_1(\lambda) & if \;\; q = 1 \\ \dots & \\ \mu_N(\lambda) & if \;\; q = N \end{cases}$$

The linear function μ_q are obtained by linearizing the μ curve around an operation point λ_q^* based on the first terms of the Taylor series:

$$\mu_q(\lambda) = \alpha_q\lambda + \beta_q \tag{3-7}$$

where α_q and β_q represent the slop and offset for each of the PLF sections. A graphical example of the approximation of a typical friction curve through a PLF divided into 2 sections ($N = 2$).

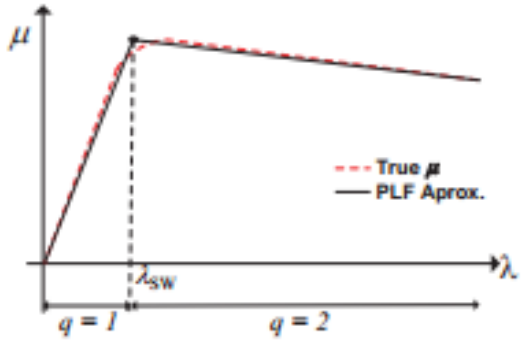

Figure 3-4: Approximation of the friction coefficient by Piecewise Linear Functions [73]

The first section, $q = 1$, is characterized by a linear increase of μ with λ, and has positive slope and zero offset ($\alpha_1 > 0, \beta_1 = 0$). When the slip exceeds a given threshold, λ_{SW}, the second section becomes active, $q = 2$, which is characterized by a descending slope and positive offset ($\alpha_2 < 0, \beta_2 > 0$). The switching between sections is defined by the parameter λ_{SW}, corresponding to the slip where the peak friction occurs, and can be defined as:

$$q = \sigma(\lambda) = \begin{cases} 1 & if \ \ \lambda \leqslant \lambda_{SW} \\ 2 & if \ \ \lambda > \lambda_{SW} \end{cases}$$

The single corner model is defined as:

$$\dot{x} = f_q(x, u) \tag{3-8}$$

$$= \begin{bmatrix} -\frac{1}{v}\frac{r^2}{J}F_z\left(\alpha_q x + \beta_q - \frac{u}{rF_z}\right) \\ -\frac{F_z}{m}(\alpha_q x + \beta_q) \end{bmatrix} \tag{3-9}$$

where $x = [\lambda \ \ v]'$ represents the continuous state, $q \in \{1, 2\}$ is the discrete state governed by the switching function σ and $u = T_b$ is the control input (braking torque) which is restricted to the actuation range.

At this point, it should be mentioned that the PWL functions shall be selected in a way that the real friction values are larger or equal to values of the approximate model. In this way, the piecewise model would not lead to larger friction values than the real ones, and the simulated braking distance would always be larger than the real one.

Selecting a smooth approximation, leads to $\alpha_1 = 7.9578$, $\beta_1 = 0$, $\alpha_2 = -0.3175$ and $\beta_2 = 0.8275$.

Comparing this approximation against the original friction curve, the following results are found.

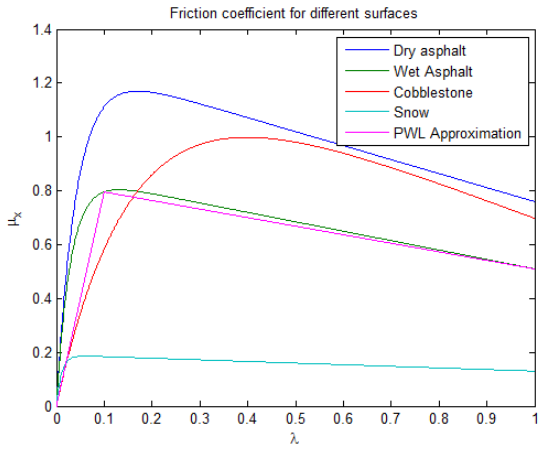

Figure 3-5: Longitudinal slip vs friction co-efficient (2 PWL regions)

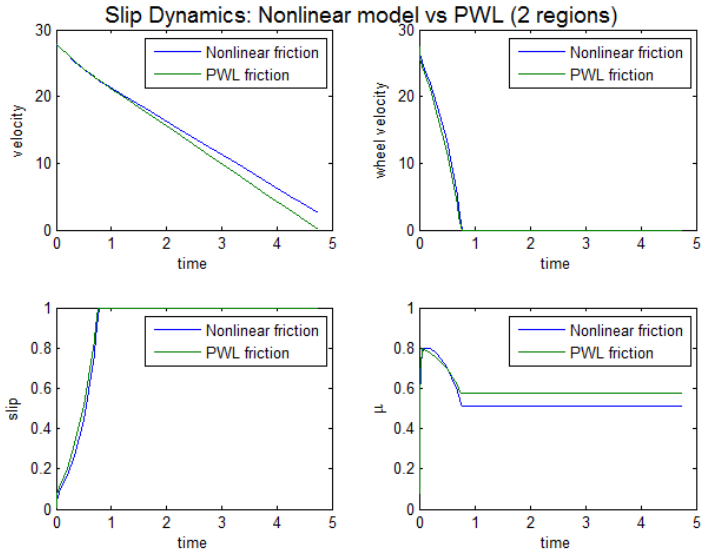

Figure 3-6: Nonlinear Friction vs PWL Friction

Analyzing the friction curves of Figure 3-5, there usually exists a zone near the friction peak where the μ derivative, is approximately zero. Based on this observation, we can increase the number of PWL regions to 3.

By selecting $\lambda_{min} = 0.085$ and $\lambda_{max} = 0.26$ and

$$\mu_1 = 9.06\lambda$$
$$\mu_2 = 0.77$$
$$\mu_3 = -0.35\lambda + 0.8625$$

we get the following approximation. It shall be observed that the approximate function is smaller that the original in terms of numerical values.

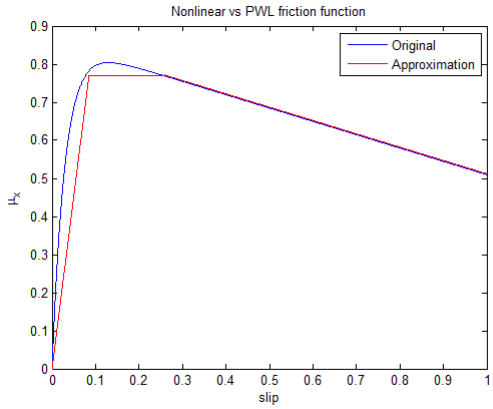

Figure 3-7: Longitudinal slip vs longitudinal friction co-efficient (3 PWL regions)

The simulation of the new model, yields to the following results.

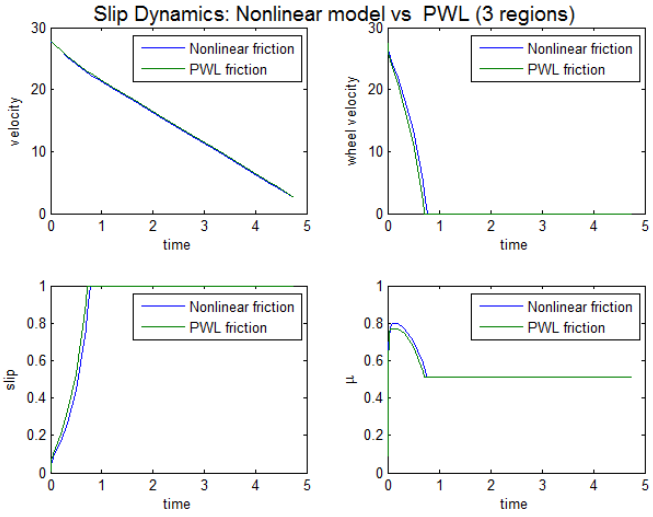

Figure 3-8: Nonlinear Friction vs PWL Friction

Using three regions yields more accurate results. However, it still introduces an error. The resulting hybrid model is a switched one with nonlinear dynamics in each discrete state. That fact creates limitations in the correct-by-design synthesis process, which is conducted in next chapter. The major problem is that the existence of a nonlinear map (nonlinear dynamics) does not guarantee convexity and the reachability analysis cannot be done directly.

In this vein, the next step is to decompose the state space into linearization domains and in each domain compute an affine function. The state space is partitioned into boxes and the nonlinear function is linearized using first order Taylor series.

$$f(\lambda, v) \approx f(\lambda_{op}, v_{op}) + \frac{df}{d\lambda}|\lambda_{op}, v_{op}(\lambda - \lambda_{op}) + \frac{df}{dv}|\lambda_{op}, v_{op}(v - v_{op})$$

Without change of coordinates, we arrive at piecewise affine system description:

$$\begin{bmatrix} \dot{x} = A_q x + B_q u + E_q \\ y = C_q x \\ q = f(x) \end{bmatrix} \tag{3-10}$$

For each linearization $q \in 1, 2, ..., N$, $\mu_q = \alpha\lambda + \beta$

$$A_q = \begin{bmatrix} 0 & -\frac{\alpha F_z}{m} \\ \frac{(\alpha\lambda_e q+\beta) r^2 F_z}{J v_e q^2} & -\frac{\alpha F_z r^2}{v_e q J} \end{bmatrix}$$

$$E_q = \begin{bmatrix} -\frac{\beta F_z}{m} \\ -(\alpha\lambda_{eq} - 2\beta)\frac{r^2 F_z}{J v_e q} \end{bmatrix}$$

For linearizations where $0 \leqslant \lambda < 0.085$, $\alpha = 9.06$, $\beta = 0$.

For linearizations where $0.085 \leqslant \lambda < 0.26$, $\alpha = 0$, $\beta = 0.77$.

For linearizations where $0.26 \leqslant \lambda \leqslant 1$, $\alpha = -0.35$, $\beta = 0.8625$.

Graphically, the state space partition is shown below. The slip $\lambda \in [0, 1]$ and $v \in [1, 65]$.

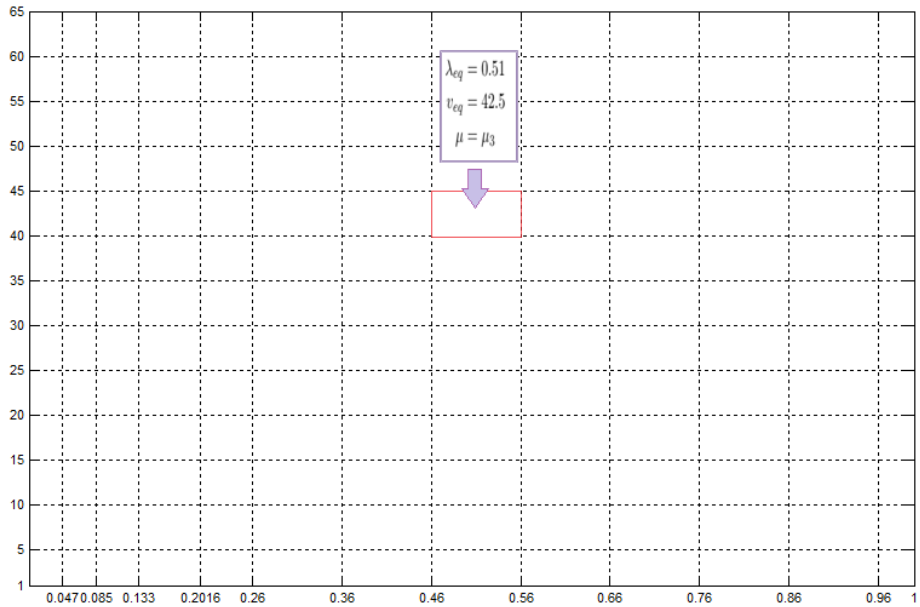

Figure 3-9: Partition of state space

The resulting partitions are 169 and are simulated with ODE45. Particularly, the scenario of a braking maneuver that starts after 1 second with a constant torque is shown below. For PWA model the braking control is deactivated when the velocity reaches 1m/s.

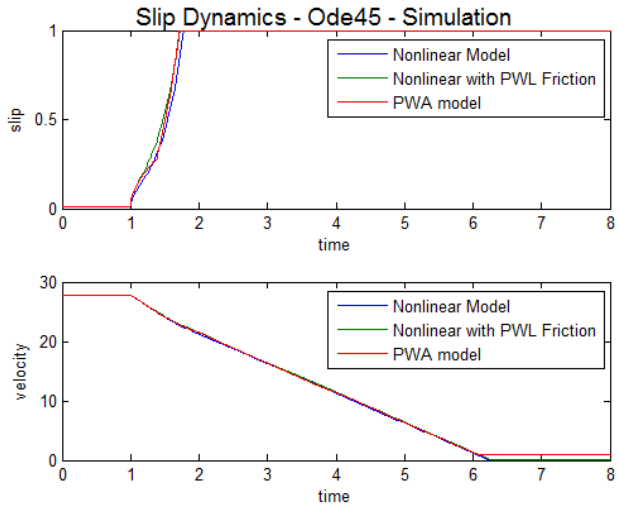

Figure 3-10: Partition of state space

Typical sampling time for braking control systems ranges from 1 to 20 ms. For our case, the continuous system is discretized with $T_s = 10$ ms. Next figure shows a comparison of continuous and discrete PWA models.

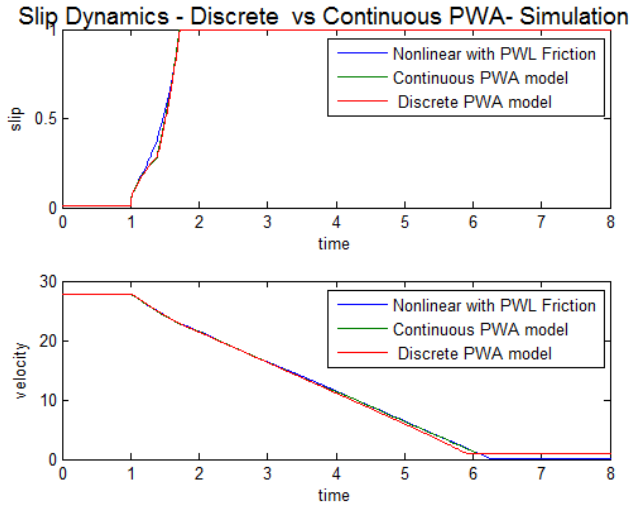

Figure 3-11: Partition of state space

Chapter 4

Correct-by-Design Control Synthesis

4-1 Formal Specifications

In this section we formalize the braking control requirements using Linear Temporal Logic (LTL). The basic building blocks of an LTL specification are the so called atomic propositions. The set of atomic propositions represents the quantities necessary to express the desired behavior. The specifications considered in this chapter can be expressed using the atomic propositions: the propositional logic conjunction $\wedge$ and negation $\neg$, and the temporal operators always $\square$ and eventually $\lozenge$. Disjunction $\vee$ and implication $\Rightarrow$ can be constructed from conjunction and negation.

There are four kinds of specification that are sufficient to synthesize braking control specifications. These specifications are defined using a target set $Z \subseteq X$ and a constraint set $W \subseteq X$:

1. **Stay**: trajectories start in the target set Z and remain in Z. This specification corresponds to the Linear Temporal Logic (LTL) formula $\square\phi_Z$ where ϕ_Z is the predicate defining the set Z,

2. **Reach**: trajectories enter the target set Z in finite time. This specification corresponds to the LTL formula $\lozenge\phi_Z$,

3. **Reach and Stay**: trajectories enter the target set Z in finite time and remain within Z thereafter. This specification corresponds to the LTL formula $\lozenge\square\phi_Z$,

4. **Reach and Stay while Stay**: trajectories enter the target set Z in finite time and remain within Z thereafter while always remaining within the constraint set W. This specification corresponds to the LTL formula $\lozenge\square\phi_Z \wedge \square\phi_W$ where ϕ_W is the predicate defining the set W.

The states of the braking model are bounded in practice. In particular, given v_0 as the initial vehicle velocity, it holds that

$$\lambda \in [0, 1]$$
$$v \in [0, v_0]$$

These requirements should be always met, a fact that necessitates the incorporation of stay LTL formula. These limits constitute the constraints set W. Considering now the braking objective, we want to regulate the slip around a desired value λ^*. Because it is not possible to stay at the exact value, we introduce some limits $d\lambda$. That means that the desired values are described as

$$\lambda \in [\lambda^* - d\lambda, \lambda^* + d\lambda]$$
$$v \in [v^* - eps, v^* + eps]$$

These values form the target set Z.

As a result, there are two prospective objectives: $\Diamond\Box\phi_Z \wedge \Box\phi_W$ and $\Diamond\Box\phi_Z$. However, correct-by-design control software could only be designed for the latter formula. So the purpose is to meet:

$$\Diamond\Box\phi_{\mathbb{Z}}$$

There are also constraints in the control input. The relation $T_b > 0$ forms a necessary technical requirement, since the wheels during braking cannot be accelerated. In practice, the actuator input is always saturated. So, on the one hand, the input should be large enough so that the braking system reaches the desired slip under any road conditions. On the other hand, the input should not exceed certain limits in order to avoid wheel lock.

In order to find $T_{b,min}$ and $T_{b,max}$ it is crucial to approximate the nonlinear vehicle dynamics by equation (3-10). For linearizations where $\lambda \leqslant \lambda_{min}(0.085)$ we would like to increase or maintain λ, by setting $\dot{\lambda} \geqslant 0$. For $\lambda \geqslant \lambda_{min}$ we would like to achieve better steerability and braking performance, by setting $\dot{\lambda} \leqslant 0$. For the aforementioned assumptions we get that

$$\alpha_{min} r F_z \leqslant T_b \leqslant \alpha_{max} r F_z \tag{4-1}$$

By substituting the worst-case value of α, the control inputs should be $0 \leqslant Tb \leqslant 1.2rF_z$. These values stabilize the system for every initial condition and ensure that the wheels will not lock.

Typical sampling time for braking control systems ranges from 1 to 20 ms. For the case under discussion, the continuous system is discretized with $T_s = 10$ ms. Next figure shows a comparison of continuous and discrete PWA models.

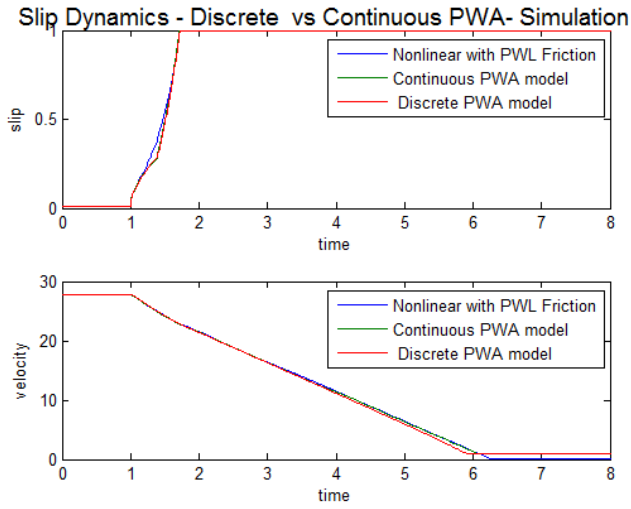

Figure 4-1: Simulation of discrete vs continuous braking dynamics

However, in the next section we display that this sampling time is very large for successful control synthesis.

4-2 Abstraction

In this section, we describe how we compute a controller that enforces the desired behavior $\Diamond\Box\phi_Z$ on the system using PESSOA. The computation is based on a discrete abstraction of the system. The abstraction is computed by a discretization of the state space, input space and time. The result is a finite state transition system $\Sigma = (Q, U, \delta)$, where Q is the set of states, U is the set of inputs and $\delta : Q \times U \rightarrow Q$ is the transition relation. Note that Q forms a grid on the state space. Therefore the sets Z, W can easily be mapped to Q by computing the grid points that fall inside the regions.

In case of an incrementally stable system, the selection of the quantization parameters of the symbolic model can be seen as:

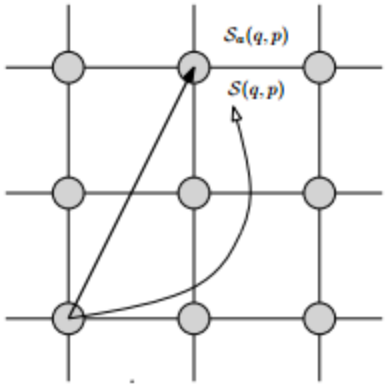

Figure 4-2: Quantization of Symbolic model

A more descriptive figure [100] for the effect of space quantization:

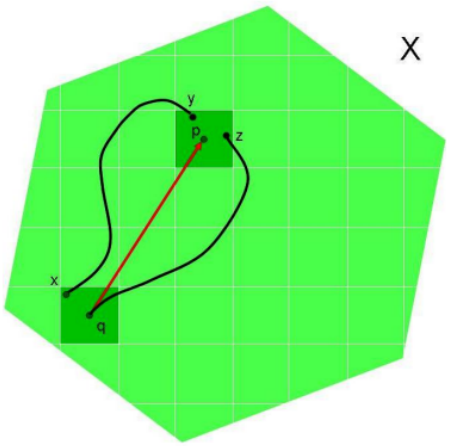

Figure 4-3: Quantization of Symbolic model [100]

However, the braking model is not incrementally stable and special attention should be given to the selection of quantizations parameters. The system is non deterministic, since for any state $x \in X$ and any input u, there exist multiple successors. That means that the approximate simulation relations should be replaced by alternating simulation relations. Graphically,

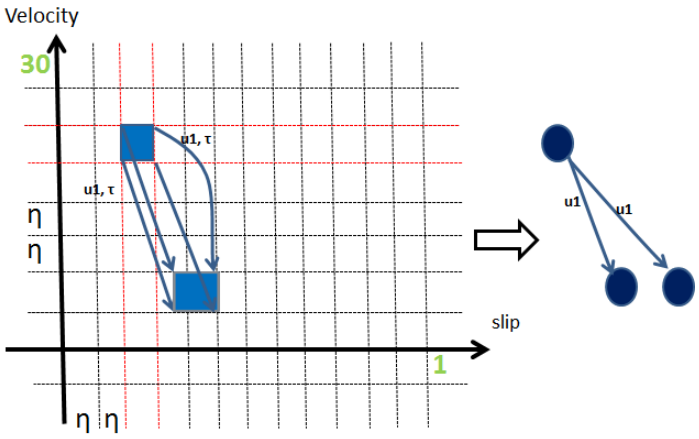

Figure 4-4: Transition relations for braking system

Based on stability concepts and theory from [99], we have made a selection of sampling time $\tau = 0.001$, state space quantization $\eta = 0.01$ and input quantization $\mu = T_{b,max}$. With these parameters we can abstract the continuous dynamics to a finite state system, with accuracy $\epsilon = \frac{\eta}{2}$. The input takes only two values 0, $T_{b,max}$, which form constant curves of duration $\tau \in \mathbb{R}^n$. We restricted our attention to a compact subset of X, so that $v \in [5, 35]$.

Sampling times larger than 0.001 seconds, were not sufficient for meeting the specifications. That was the case, because the control action could not be updated fast enough.

The transition relations in the resulting finite state machine are found through set-based reachability. Considering, the uniform gridding of the state space, polytope representation is the most suitable option. The reachability analysis has been computed with MPT toolbox.

Graphically,

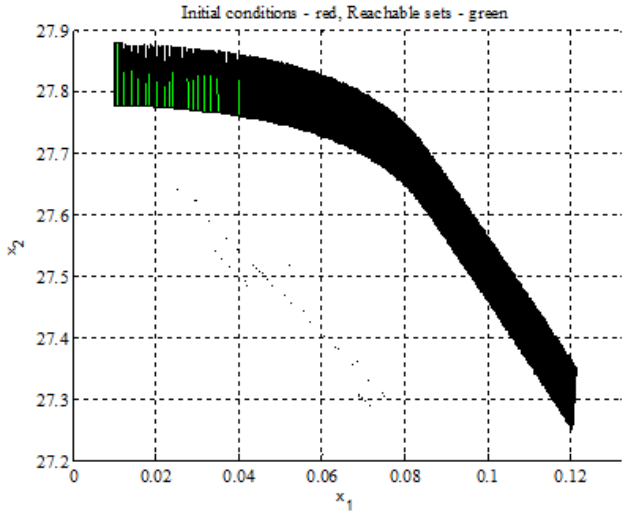

Figure 4-5: Computation of reachable sets

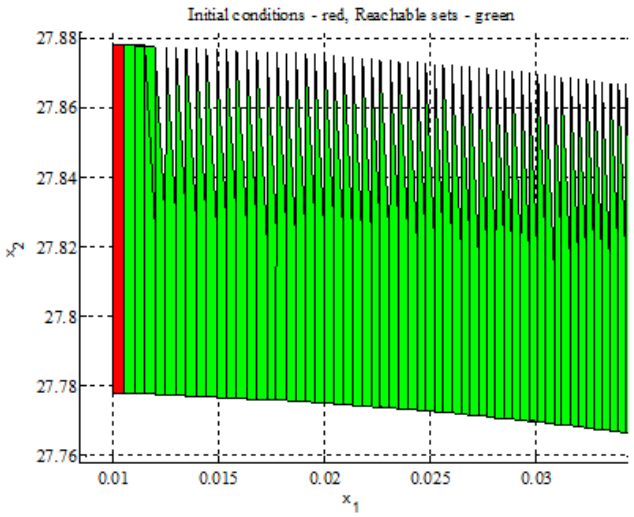

Figure 4-6: Computation of reachable sets

The reachability analysis is conducted on the resulting hybrid model and the sets are over-approximated to match the symbolic states. In essence, every symbolic state (infinite continuous states) represents a polytope (box) in the original state space. The next step is to find the image of this polytope for the two available inputs.

Finally, the resulting end points are over-approximated and are mapped to the symbolic states. Graphically,

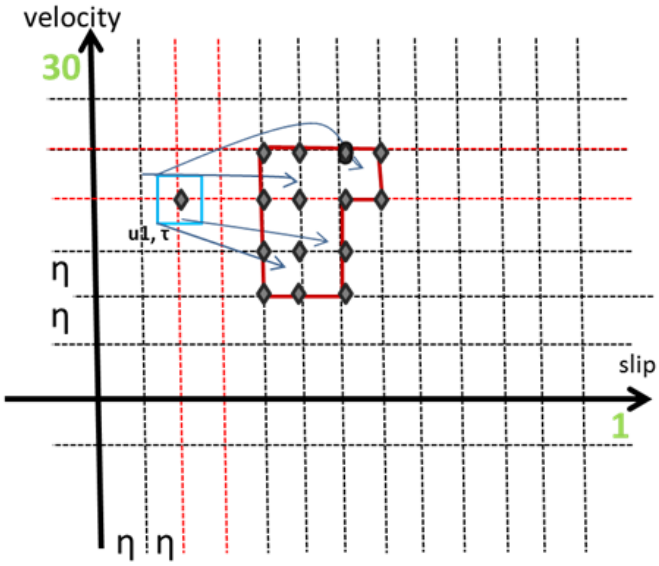

Figure 4-7: Overapproximation of reachable sets

The resulting abstraction has been constructed after 59,382 seconds and consists of 600,000 states.

4-3 Braking Control Software Synthesis

Given the abstraction and the desired LTL formula, the controller is synthesized automatically by PESSOA and is defined as a large look-up table. This look-up table is stored on the hard disk in terms of a binary decision diagram (BDD). Based on the current mode and state of the system, the BDD selects the valid inputs so that the closed-loop system satisfies the specification. It should be noted that the control input is chosen in a lazy manner; the input is only changed, when the previously used input cannot be used again.

It is straightforward to implement the controller on an embedded device, since it reduces basically to query the BDD at each sampling time [101]. The time to query the BDD is at least two magnitudes smaller than the sampling time and therefore does not constitute a problem.

For a braking maneuver from $15\frac{m}{s}$ to $10\frac{m}{s}$, with $\lambda_0 = 0.1$ and $\lambda^* = 0.13$, we get the following simulation results:

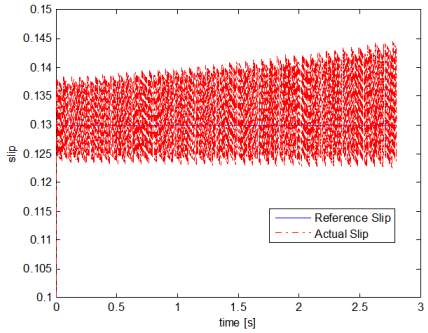

Figure 4-8: Reference vs slip

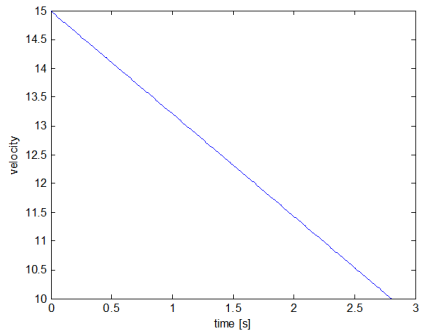

Figure 4-9: Velocity evolution

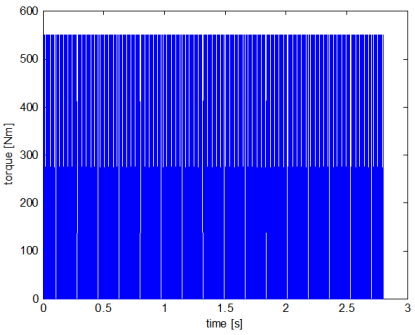

Figure 4-10: Input evolution

The specifications are met, while the effect of decreasing speed is highlighted. Decreasing the value of time quantization parameter leads to better results.

4-4 Conclusions

In the above chapter of the thesis, we used the correct-by-design approach for control synthesis. By casting the braking control as a wheel slip regulation problem and expressing the specifications in LTL formula, we designed control software that is provably correct by construction. We acknowledged the problematic behavior of the braking system for low speeds. That issue was resolved, by deactivating the controller for velocity values less than 5m/s. Additionally, it was clear that use of a lazy controller does not provide optimal results, since there are oscillations around the desired slip value. Finally, the importance of time quantization was observed.

We have made some simplifying assumptions with regard to the dynamic modeling and braking technology. The longitudinal vehicle velocity is not easily measured in practice and the slip reference should be generated by means of a high-level controller. In addition, we have not considered the behavior of the system in the inter-sampling time. That should be done bloating the convex hull of reachable sets, expressed as zonotopes. However, the selection of a small quantization time partly alleviates this problem.

Chapter 5

Control Synthesis and Verification

This chapter will cover the control synthesis and the parametric verification part. For the control synthesis, the objective is the wheel slip regulation. Different control approaches are used, such as hybrid, sliding-mode and PID. As for the verification part, the nonlinear dynamics render the use of "verification by simulation" as the most applicable method.

5-1 Control Synthesis

In this section, the control designs are analyzed. In order to evaluate the performance of the controllers, the reference signal (slip) is considered to be known. In practical settings, the reference signal is typically generated by a high level controller.

5-1-1 Bang-Bang Controller

Bang-bang control, also known as on-off control, is one of the simplest strategies. It refers to a feedback controller which changes abruptly between two states. In order to select the these two control actions, it is necessary to reiterate the control objectives and the nonlinear physical model. The model is:

$$\dot{\lambda} = -\frac{1}{v}\frac{r^2}{J}F_z(c_1(1-e^{-c_2\lambda}) - c_3\lambda) + \frac{r}{vJ}T_b \tag{5-1}$$

$$\dot{v} = -\frac{F_z}{m}(c_1(1-e^{-c_2\lambda}) - c_3\lambda) \tag{5-2}$$

The objective is to design a controller which decelerates the vehicle as fast as possible and maintains its steerability. For any road surface, the maximum deceleration is achieved at an optimal slip value, λ_{opt}. This slip value should be the reference signal and the vehicle should be able to track it. For the case of wet asphalt, the Burckhardt model yields $\lambda_{opt} \approx 0.13$.

At this slip value, the wheel is far away from being locked, and the steerability of the car is maintained.

In essence, for values where $\lambda \leqslant 0.13$, we would like to increase or maintain λ, so $\dot{\lambda} \geqslant 0$. While for values $\lambda \geqslant 0.13$, we would like to reduce λ, to get better steerability and braking performance, so $\dot{\lambda} < 0$. Now we can compute the control input space in relation to the state space.
For $\lambda \leqslant 0.13$, $v > 0$ and $\dot{\lambda} \geqslant 0$. Recalling the analysis of Chapter 4, the minimum and maximum torque values were selected to be $T_{b.min} = 0$Nm and $T_{b.max} = 550$Nm. The control law is

$$u(t) = T_b = \begin{cases} T_{b.max} & \text{if } \lambda \leqslant 0.13 \\ T_{b.min} & \text{if } \lambda > 0.13 \end{cases}$$

Considering a braking maneuver that starts after 1 second, the controller performance is shown below.

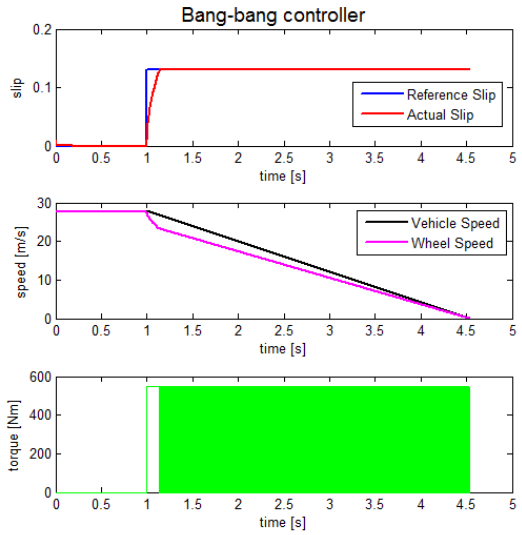

Figure 5-1: Bang - Bang Controller

The simulations have been done in MATLAB and the braking distance has been calculated through numerical integration. The first two graphs illustrate the state evolution (velocity, slip), while the third figure indicates the braking torque. For the above selected scenario, the braking distance is 76.98m.

5-1-2 Hysteretic

A different implementation of bang-bang control, known as hysteretic control is tested. The control logic is shown below:

$$u(t) = T_b = \begin{cases} T_{b.max} & \text{if } \lambda \leqslant \lambda_{min} \\ T_{b.min} & \text{if } \lambda \geqslant \lambda_{max} \\ u(t^-) & \text{otherwise} \end{cases}$$

The controller principle of operation is as follows: whenever the continuous state reaches or crosses the boundary of the set F, defined by $\partial F = \{\lambda_{min}, \lambda_{max}\}$, a fixed actuator value is selected, $u \in \{T_{b.min}, T_{b.max}\}$. If λ belongs to the interior of the set F, the interior of the set F, then the previous actuator value is maintained. The resulting performance for $\lambda_{min} = 1.1$ and $\lambda_{max} = 1.5$ is shown below. The braking distance is 77.02m.

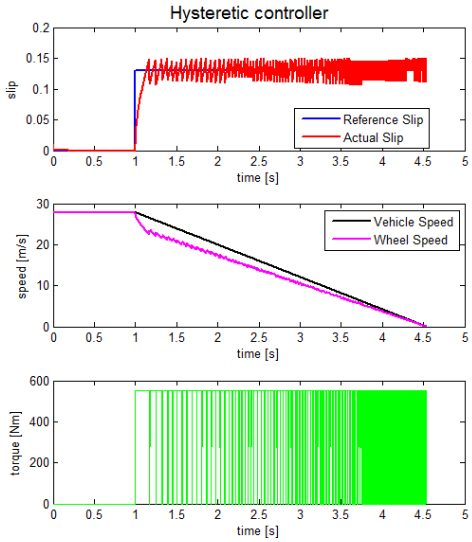

Figure 5-2: Hysteretic Controller

The resulting performance for $\lambda_{min} = 1.2$ and $\lambda_{max} = 1.4$ is:

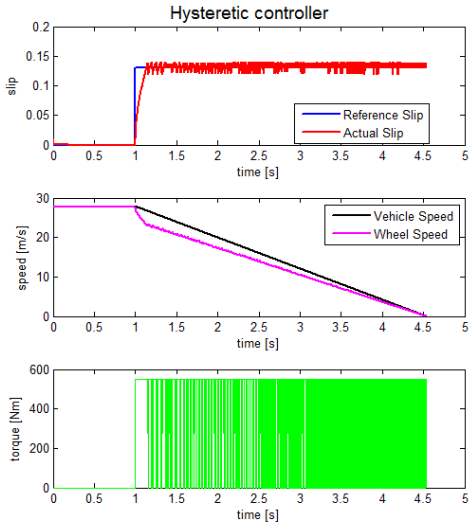

Figure 5-3: Hysteretic Controller

The braking distance is 76.99m, smaller than before but at the expense of robustness.

5-1-3 State feedback

Based on our analysis on previous sections, it is plausible that the use of a continuous state feedback controller cannot achieve the desired performance on the condition that the friction can vary arbitrarily.

Considering the state feedback, $u = -Kx$,we first tried a simple linear version with $K = [K_1; K_2]$. However, the results verified that the reference could not be tracked and the braking distance was very large.

In this end, integral action is added in order to achieve tracking. For $K = [15000; -5]$ and integrator gain $H = 3.5 \cdot 10^6$, we have:

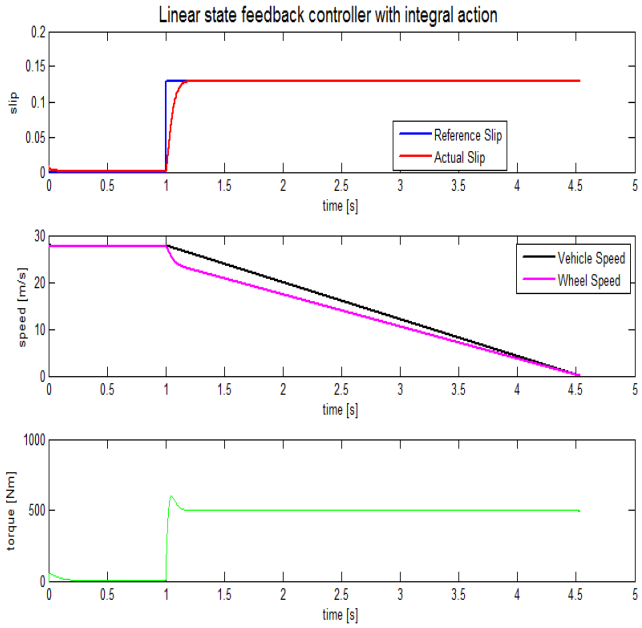

Figure 5-4: State Feedback Controller

The braking distance is 76.83m.

5-1-4 PID

PID controllers are widely used in industrial applications and have been the most common choice for braking applications. The PID design is based on linearized model with $\bar{v}$ at 30m/s and 2m/s at $\bar{\lambda} = 0.13$. The actuator constraints are considered, since $T_b \leqslant T_{b.max}$. At low velocity the system becomes unstable, since dynamics are faster and controller is not fast enough to handle this problem. Proportional action should be high in order to stabilize the possibly unstable open loop model and the design should handle the worst case scenario.

After manual tuning of the PID gains, we select $K_p = 5 \cdot 10^9$, $K_d = -5 \cdot 10^4$, $K_i = 10$, $N = 0$.

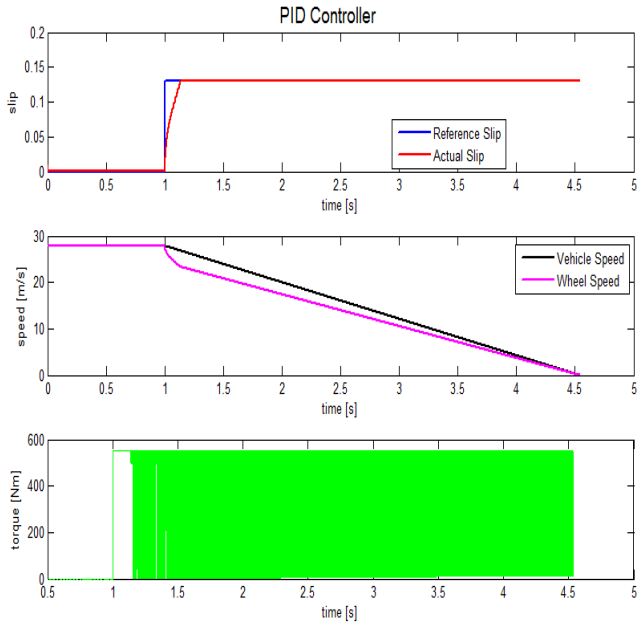

Figure 5-5: PID Controller

The braking distance is 77.05m.

The performance can be further improved by choosing $K_p = 1.5724 \cdot 10^6$, $K_d = -51.4434$, $K_i = 1.9891 \cdot 10^8$, $N = 3.0565 \cdot 10^4$.

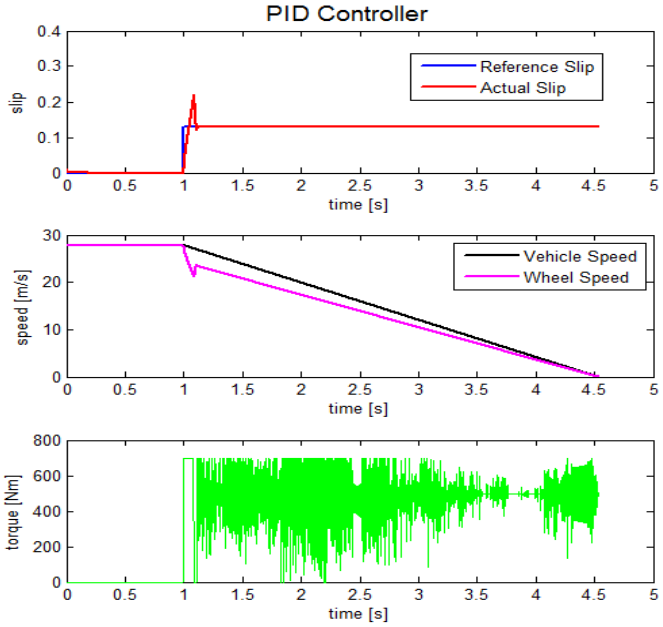

Figure 5-6: PID Controller

The braking distance is 76.86m.

5-1-5 Sliding mode

Another possibility is to design a sliding mode where the sliding surface is $s = (\frac{d}{dt} + K)\int_0^t e d\tau$ with $e = \lambda - \lambda_{des}$. So, $\dot{s} = \dot{e} + Ke$. Thus,

$$\dot{s} = -\frac{r^2 F_z \mu(\lambda)}{vJ} + \frac{1}{v}\frac{r}{J}T_b + Ke$$

The objective is to stay on the surface $\dot{s} = 0$. Solving for T_b, we get $\hat{T}_b = rF_z\mu(\lambda) - \frac{vJ}{r}Ke$. The control input is a function of the friction, which in practice cannot be efficiently measured.

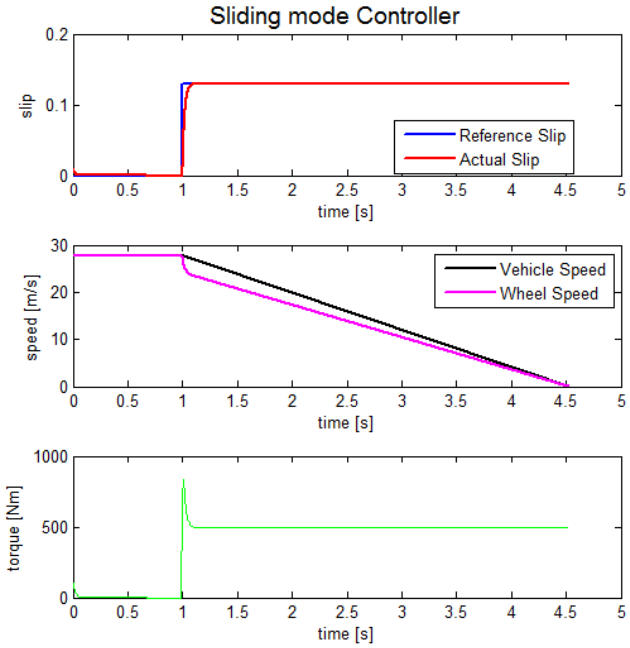

Figure 5-7: Sliding mode Controller

The braking distance is 76.66m.

5-1-6 Hybrid

In this section, a hybrid controller is designed. The advantages of sliding mode control and the difficulty of friction measurements, has motivated the use of a modified sliding mode controller.

The control logic is:

$$u(t) = T_b = \begin{cases} 10rF_z\lambda - \frac{vJ}{r}Ke & \text{if } \lambda \geqslant \lambda_{ref} - 0.2 \\ (-\frac{1}{4}\lambda + \frac{3}{4} - 0.2)rF_z - \frac{vJ}{r}Ke & \text{if } \lambda \leqslant \lambda_{ref} + 0.2 \\ -(K_p + K_i\frac{d}{dt})e & \text{otherwise} \end{cases}$$

In this way, we get vector fields that point towards $\lambda = 0.13$ and in directions of smaller velocities v. To avoid chattering, the PI Controller stabilizes the dynamics around the desired slip. It can further handle smaller variations in the friction coefficient.

Selecting $K = 100$, $K_p = 1.6 \cdot 10^6$, $K_i = 2 \cdot 10^8$.

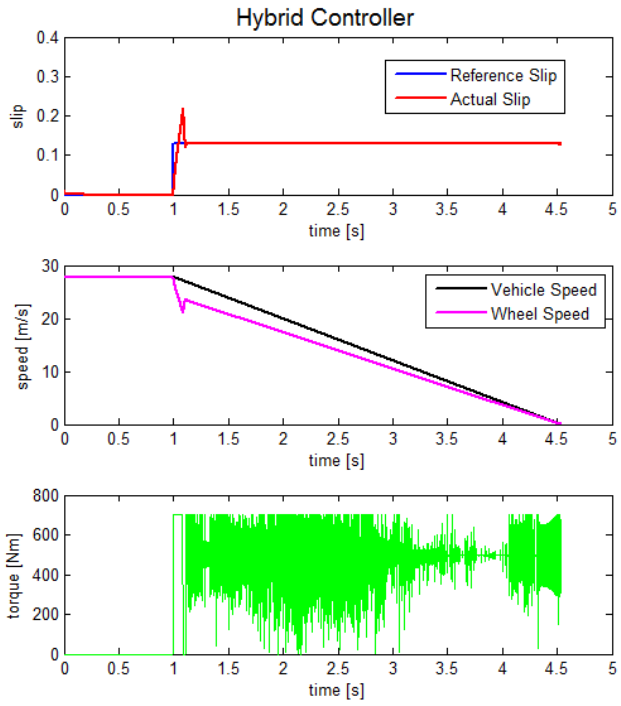

Figure 5-8: Hybrid Controller

The braking distance is 76.86m.

5-1-7 Partial Feedback Linearization

Feedback linearization is a common approach used in controlling nonlinear systems. The approach is fundamentally based on a transformation of the nonlinear system into an equivalent linear system through a change of variables and a suitable control input. For our system, the goal is to control one parameter, the slip. Feedback linearization is not directly applicable to the original dynamical model. To this end, partial feedback linearization is applied. In essence, we are interested in cancelling the nonlinear dynamics of the slip equation.

Recalling the dynamics:

$$\dot{\lambda} = -\frac{1}{v}\frac{r^2}{J}F_z\mu(\lambda) + \frac{r}{vJ}T_b$$

It is desirable to select:

$$T_b = rF_z\mu(\lambda) + \frac{vJ}{r}z \tag{5-3}$$

where z is the new control input. On the condition that the friction can be accurately measured, this controller cancels the nonlinearities and achieves good performance. In particular, by $z = -K(\lambda - \lambda_{ref})$ the tracking is feasible.

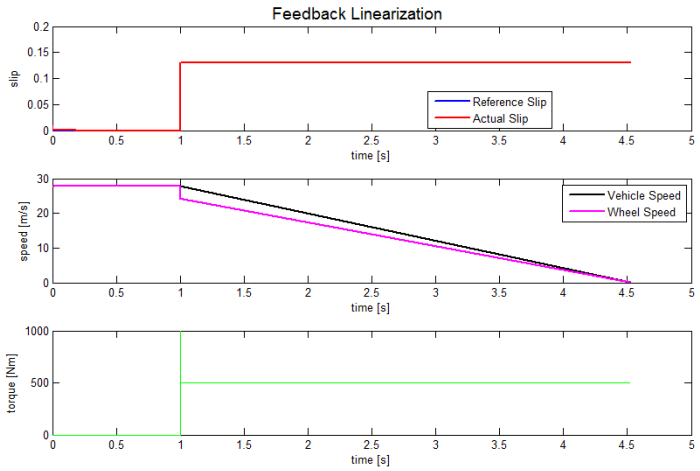

Figure 5-9: Partial Feedback Linearization

5-1-8 Comparison

At this section, the overall performance of the designed controllers is compared and evaluated.

Table 5-1: Controller Performance

Control approach	Braking distance
On-off	76.98
Hysteretic	77.02
State Feedback	76.83
PID	76.86
Sliding mode	76.76
Hybrid	76.86
Feedback Linearization	76.68

It is apparent that for the above selected scenario and the accompanied assumptions the feedback linearization is the best option. However, a real time application requires testing many different scenarios, initial conditions and considering uncertainties, sampling. That means that simulations do not offer a sufficient tool for determining the suitability of a control design.

5-2 Parametric Verification

Model-based analysis and design techniques for complex systems with uncertainty rely mostly on extensive simulation. Efficient techniques and tools exist for verification of hybrid systems with linear continuous dynamics but no tool can be readily scalable for hybrid non-linear dynamics [95]. To handle this issue, the concept of verification by simulation can be fruitful. Simulation-based techniques replace extreme points and sets, by finitely many transitions. The Breach toolbox [54] has been used for following analysis.

5-2-1 Uncertainties in initial conditions

As a first step, we would like to check the performance of our design under different initial conditions. The range of state variables is shown in the next table.

Table 5-2: Range of continuous states

Variable	Minimum	Maximum
λ	0	1
v	0	40

Essentially, Breach computes the reachable set for a specified set of trajectories. By refining the values of initial conditions, Breach calculates new reachable sets. In case we select, only one nominal value for initial conditions, Breach calculates only one trajectory. There are two ways to select the parameter values, either by grid refining or by Halton algorithm. For example, we may select 1, 4, 2 x 3, 70 x 70 parameter sets for grid sampling.

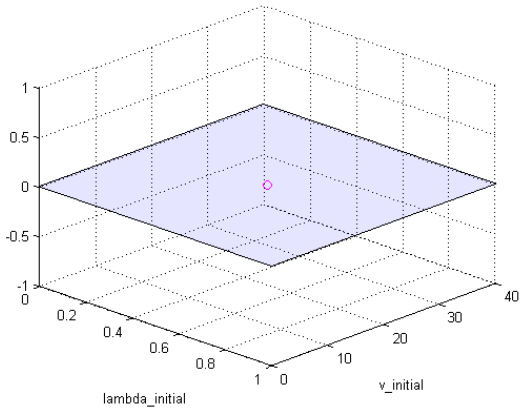

Figure 5-10: Parameter Sets

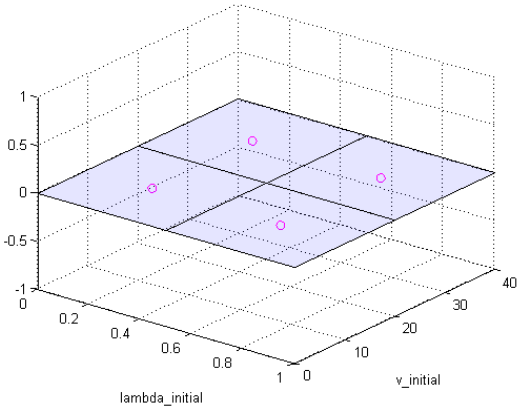

Figure 5-11: Parameter Sets

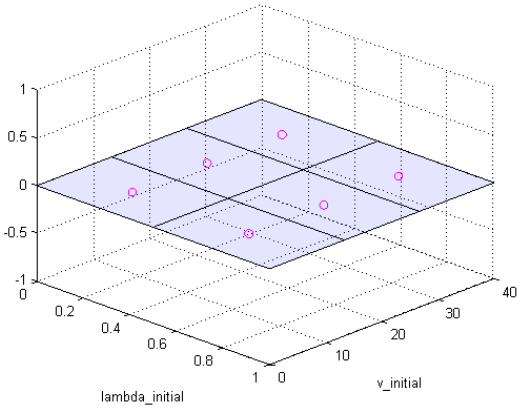

Figure 5-12: Parameter Sets

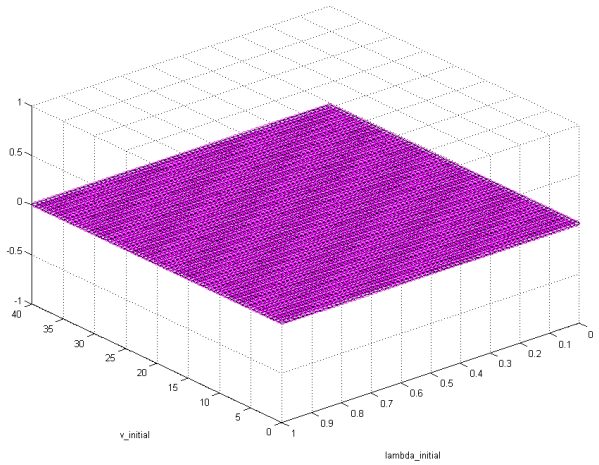

Figure 5-13: Parameter Sets

The underlying idea is to simulate the trajectory for the nominal value of the parameter sets. That means that Figure 5-10 would produce one trajectory, while Figure 5-12 would simulate six different trajectories. Of course, the accuracy is increased in conjunction with the increase in the number of parameters.

When the number of uncertain parameters is large, grid sampling becomes impractical since it generates a number of new parameter values which is exponential in the dimensionality of the uncertain set. In this situation, Breach proposes a method to sample uniformly using quasi-random numbers. Quasi-random numbers provide more uniform repartition of a given number of points inside a cube than pseudo-random numbers.

For instance, if we select 100 and 2500 parameter values, we simulate our system for the following parameter sets.

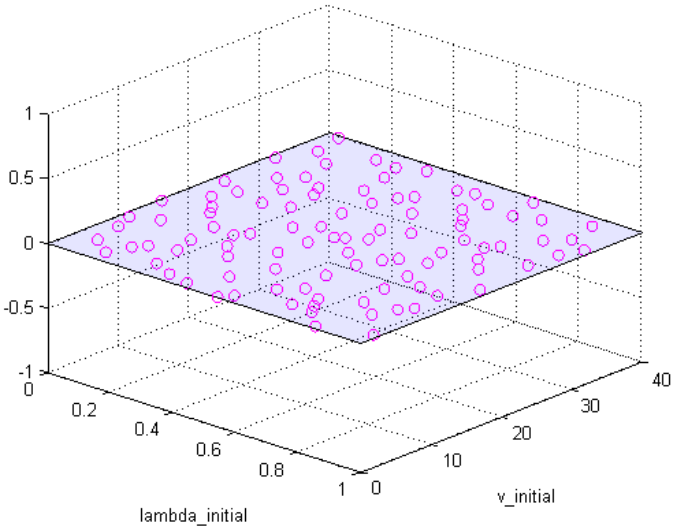

Figure 5-14: Parameter Sets

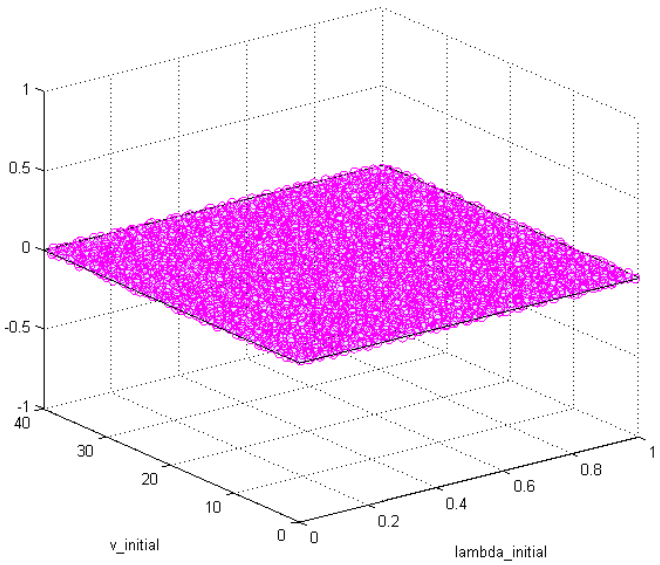

Figure 5-15: Parameter Sets

Lets start with the bang-bang controller that we designed. In case, we limit the initial velocities between 25 and 40 m/s and consider a uniform grid 10 x 10 for the initial conditions, we get the following results.

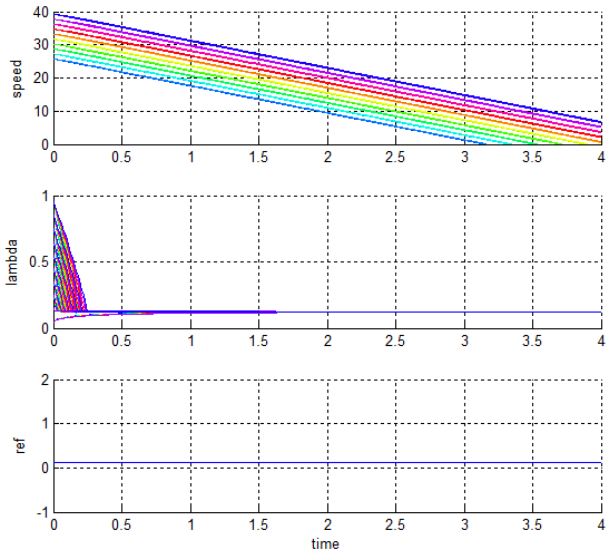

Figure 5-16: State evolution

However, with Halton refinement, we can observe more outward behaviors, with the same number of potential initial conditions.

In realistic applications, the initial value of the slip would be rather small. For $0 \leqslant \lambda \leqslant 0.3$,

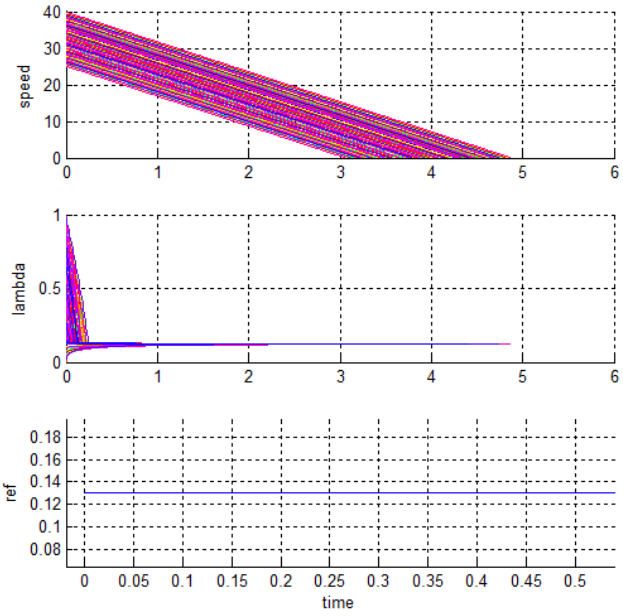

Figure 5-17: State evolution

with Halton refinement method and 100 parameter sets, we get:

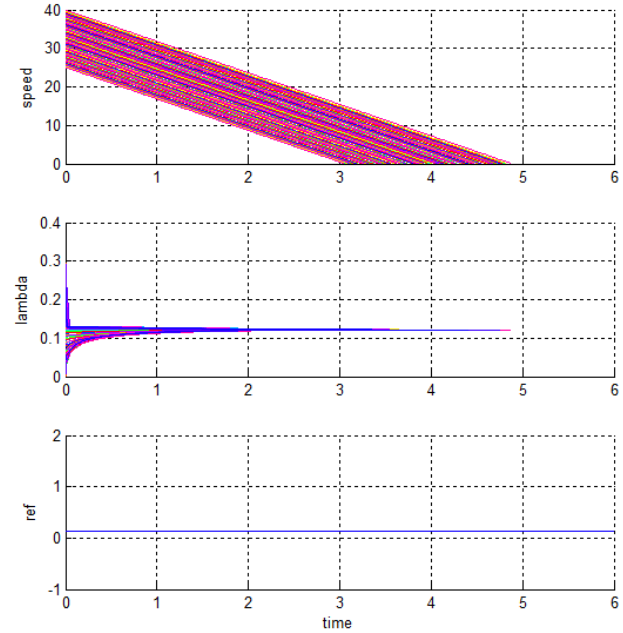

Figure 5-18: State evolution

With Halton refinement and 250 parameter sets, we have more accurate results.

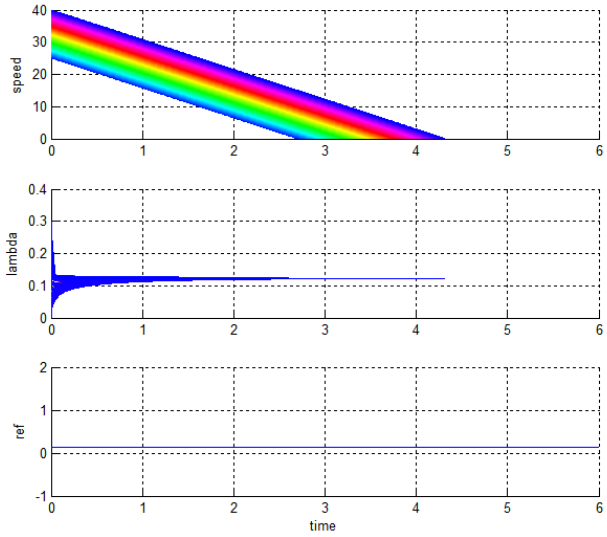

Figure 5-19: State evolution

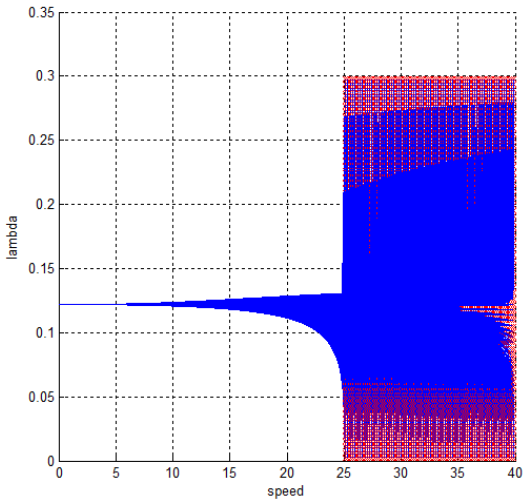

Figure 5-20: Phase Plane

The performance of the nonlinear controller (partial feedback linearization) is depicted in figures 5-21, 5-22. It can be seen that the nonlinear controller out-performs bang-bang controller for the above selected scenario.

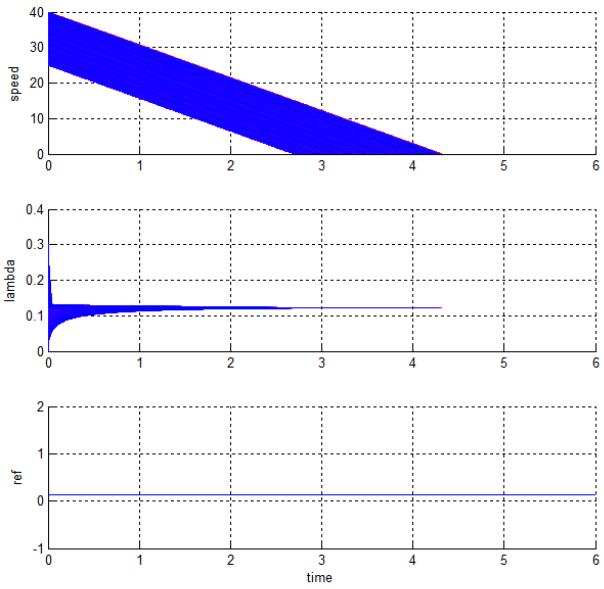

Figure 5-21: State evolution

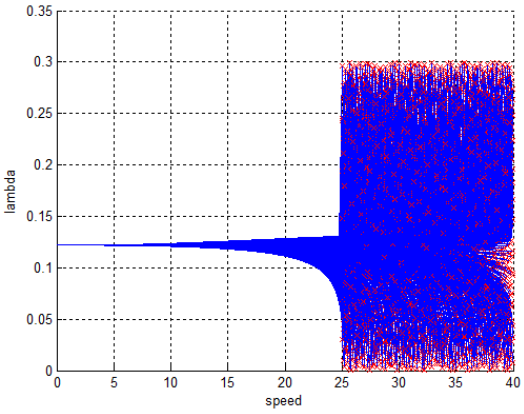

Figure 5-22: Phase Plane

5-2-2 Uncertainties in parameters of physical system

In real time applications, uncertainties in the physical system are very common. For the braking scenario, the mass constitutes a highly uncertain parameter, since it describes the load distribution changes. In this vein, the mass may not be 225 kg, but may vary $200 \leqslant m \leqslant 250$.

By considering 10 x 8 x 20 (λ, v, m) uniformly distributed values, such as

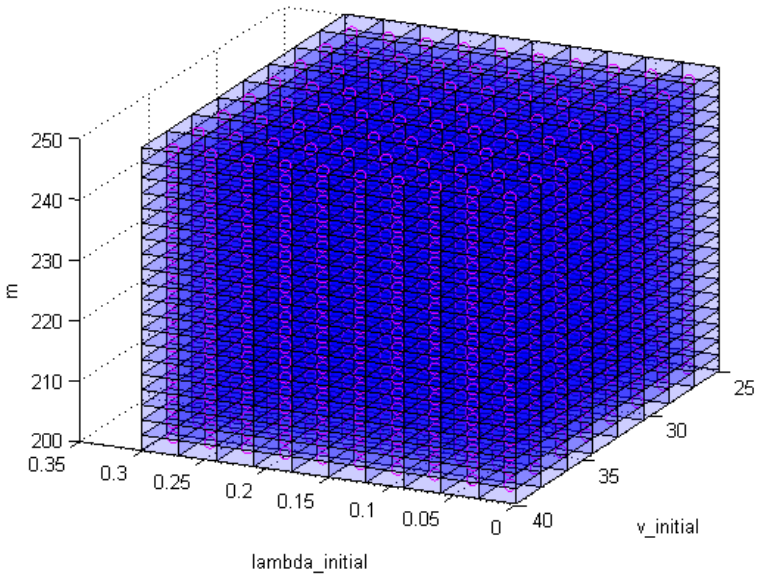

Figure 5-23: Parameter Partition

We have the following results for nonlinear control:

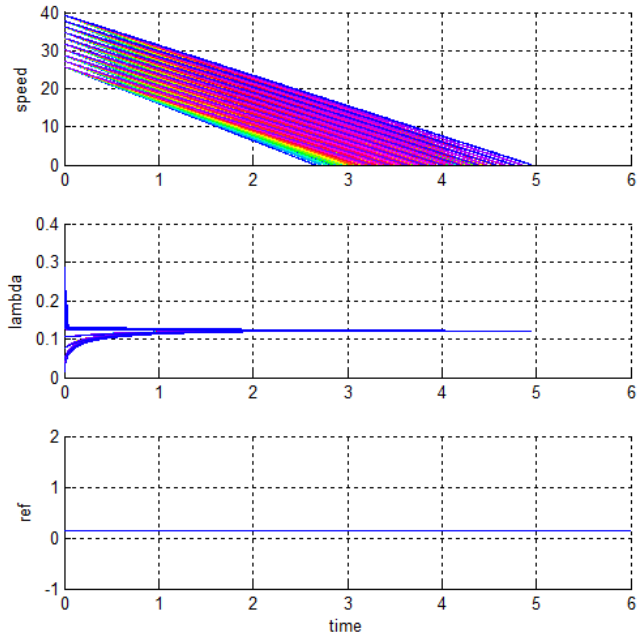

Figure 5-24: State Evolution

For bang-bang control:

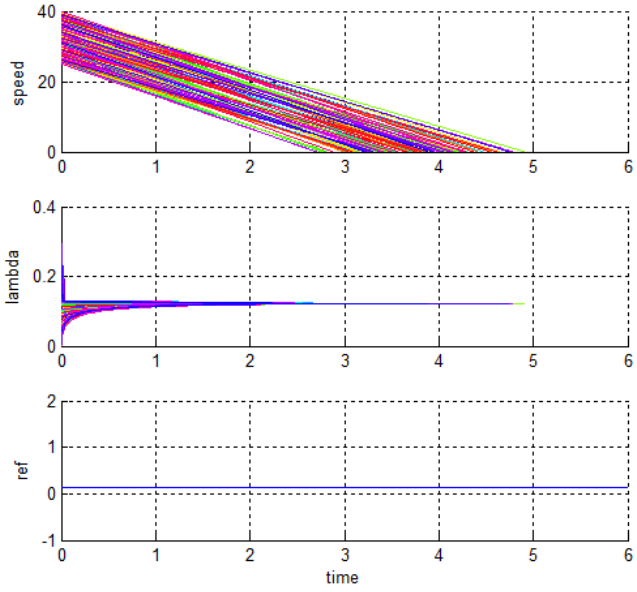

Figure 5-25: State Evolution

For the PID controller, we arrive at:

5-2-3 Requirements in Mixed Interval Temporal Logic

In this section, we are going to test some Temporal Logic Formulas. Breach toolbox has the capability to check a dynamical model, against pre-specified requirements. It performs approximate reachability analysis, relying on systematic (barbaric) simulation. That means that the results are not formally verified. However, the influence of a parameter variation on a simulation trace is evaluated with sensitivity analysis [95].

The specifications are validated in continuous time and can be described in Metric Interval Temporal Logic (MITL) or Signal Temporal Logic (SNL). In the context of this section, the braking requirements can be specified in MITL, as described below. The analysis focuses on the hybrid controller.

φ_1 : "ev($\lambda[t] < \lambda_{max}$)"

φ_2 : "alw($\lambda[t] < \lambda_{max}$)"

φ_3 : "$\text{ev}_{[0,inf]}$ $((\lambda[t] > \lambda_{min})$ and $(\lambda[t] < \lambda_{max}))$"

φ_4 : "alw((ev $(\lambda[t] > \lambda_{min}))$and $(\text{ev}(\lambda[t] < \lambda_{max})))$"

As a first step, we want to verify that the dynamics preserve the operational modes and select $\lambda_{min} = 0$ and $\lambda_{max} = 1$. No counterexamples exist and the formulas are satisfied. In particular,

In essence, the formulas are satisfied as long as the red satisfaction line is not zero. Now, let's change the limit values to $\lambda_{min} = 0.08$ and $\lambda_{max} = 0.2$. The reference signal is 0.13.

The formula φ_1 is satisfied, as it can be seen at the next figure.

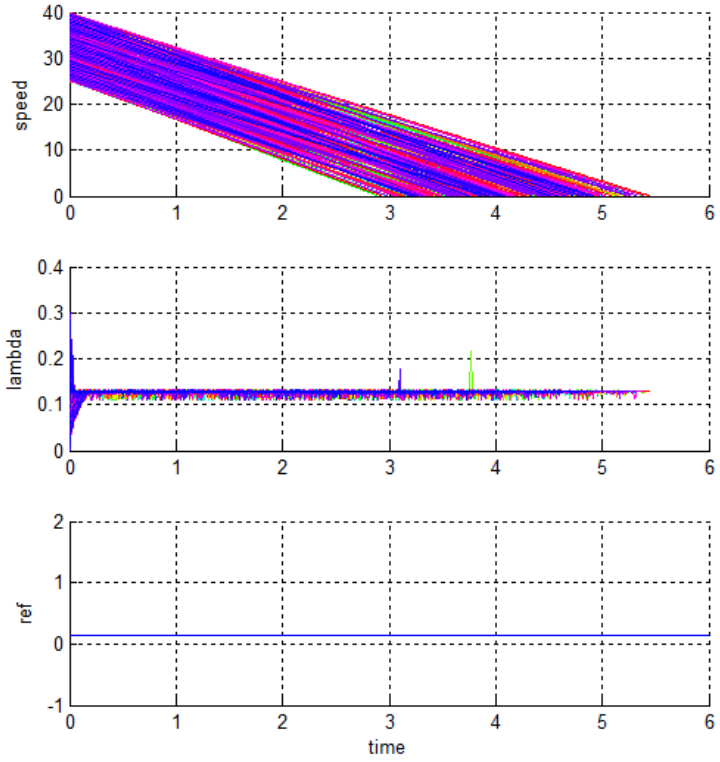

Figure 5-26: State Evolution

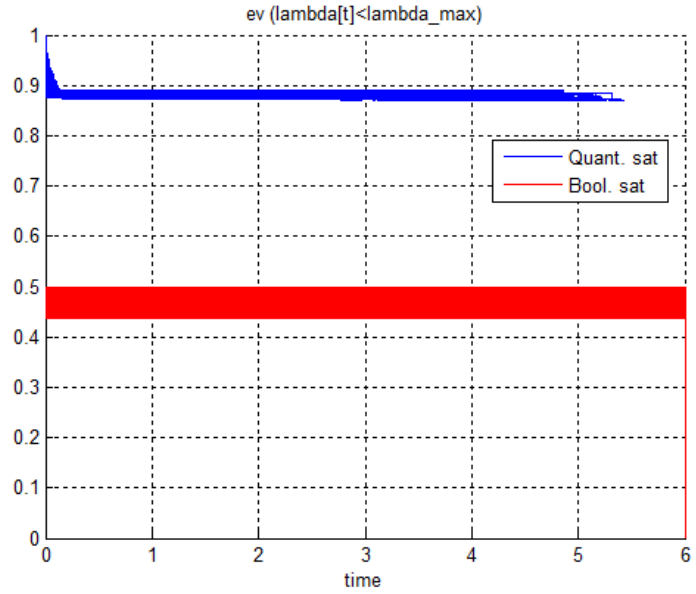

Figure 5-27: Formula satisfaction

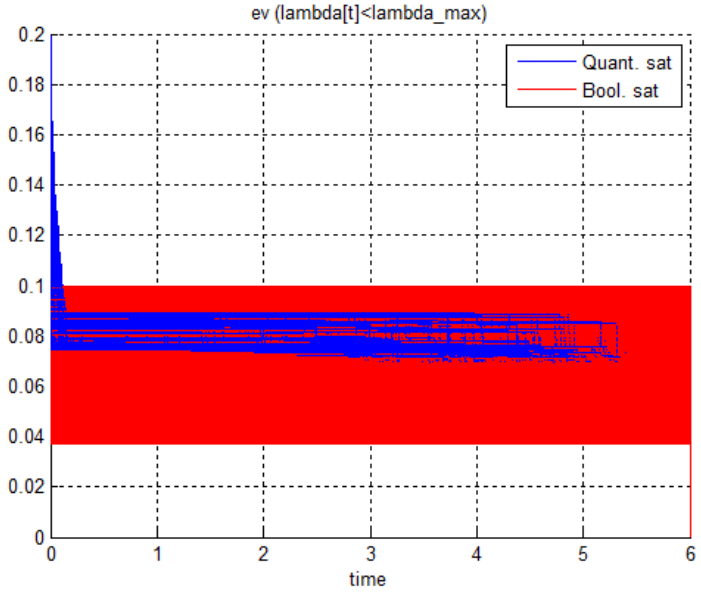

Figure 5-28: Formula Satisfaction

The formula φ_4 which is a composition of several individual formulas is also satisfied. That means that the hybrid controller maintains the wheel slip, within the desired values.

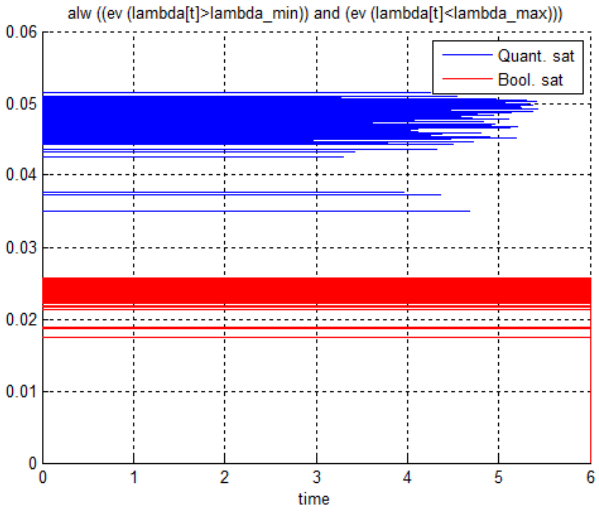

Figure 5-29: Formula Satisfaction

5-2-4 Parametric synthesis - Uncertainties in tire friction

In this section, we would like to evaluate the impact of uncertainties in the tire friction model and develop a strategy for defining the reference slip. A crucial issue in braking control is

friction measurements. Let's come back to the equation of the Burckhardt friction model:

$$\mu(\lambda) = c_1(1 - e^{-c_2\lambda}) - c_3\lambda$$

The uncertainty range can be defined by manipulation of parameters c_1, c_2, c_3. Graphically, the uncertainty interval can be seen as:

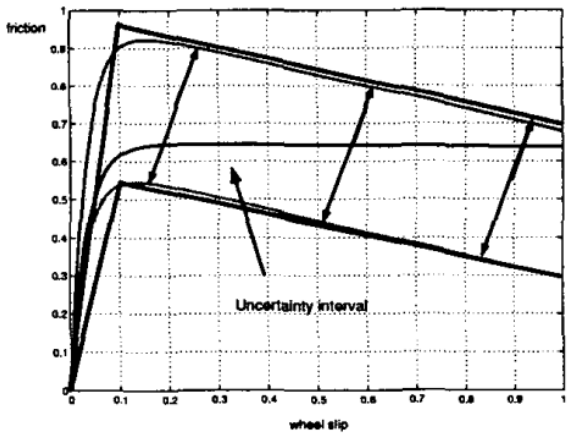

Figure 5-30: Approximation of friction coefficient by uncertain Piecewise Linear Functions [96]

In this vein, the nominal values should be replaced by appropriate ranges.

Table 5-3: Friction characteristic parameters

Parameter	Nominal	Minimum	Maximum
c_1	0.86	0.66	1.06
c_2	33.82	24	44
c_3	0.35	0.3	0.7

In addition, the uncertain parameters in that scenario should be increased to five (c_1, c_2, c_3, v_0 and λ_0). The computed trajectories of the bang-bang controller are shown below.

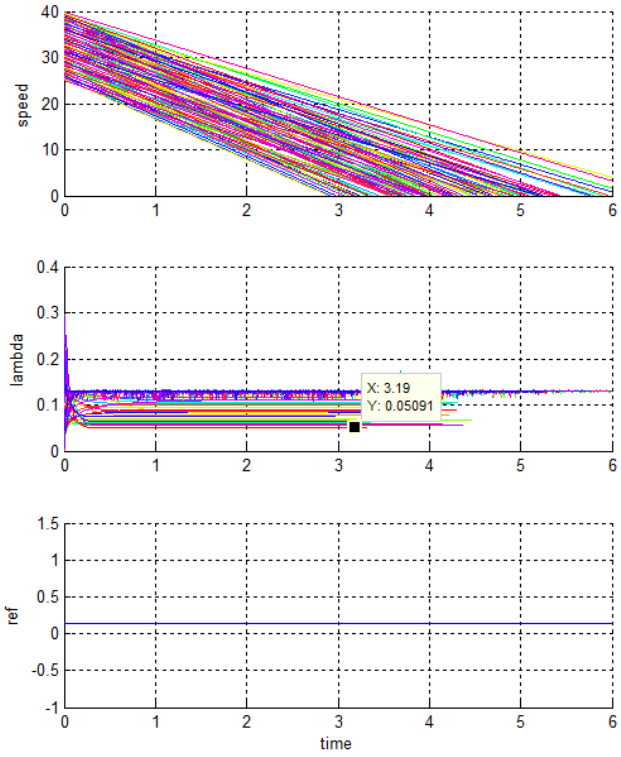

Figure 5-31: Simulation Results

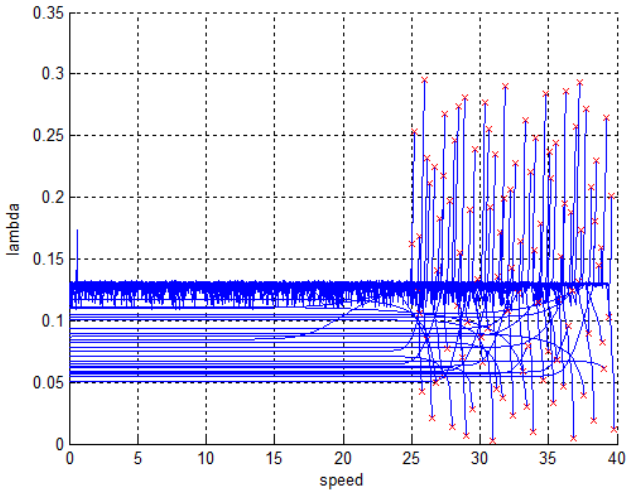

Figure 5-32: Simulation Results

Looking back on the problem statement in Chapter 2, it is known that that the actual slip is close to its optimal value for every surface, when $\lambda \in [0.08, 0.2]$. That means that we want to specify the range of the reference signal that guarantees these limits. On the condition that the reference signal is between $\lambda_{ref} \in [0.11, 0.18]$, the specifications cannot be satisfied. The falsification process of Breach returns:

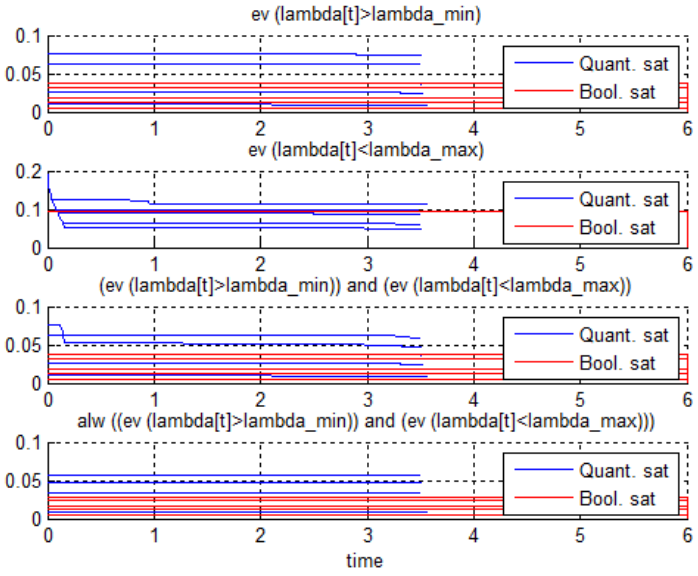

Figure 5-33: Formula Satisfaction

That means that the reference signal should be more restricted. In particular, by selecting the reference signal in the range $[0.12, 0.15]$, the specifications are satisfied and the absolute value of robustness is 0.041905.

5-3 Formal Verification

The parametric verification is a bridge between simulation and formal verification. The main limitation though is that the whole state space is not explored. The analysis is based on trajectories, not on sets. So, the resulting trajectories are selected in a way which allows us to verify whether the synthesized controller complies with the specifications. In the case the controller is falsified, it is known that the controller should be redesigned. On the scenario that the properties are satisfied, we lack formal guarantees. There may be some cases that are not tested.

A good way to resolve this guarantee issue is the property of incremental stability. Incremental stability is a property of dynamical and control systems that has been distinguished from traditional stability concepts. It deals with the stability and convergence of trajectories with respect to each other, rather than stability of an equilibrium point [22].

In essence, incremental stability is the asymptotic forgetfulness of past history. The notion that we are interested in is the one of incremental global asymptotic stability (δ -GAS). In case that the closed loop system is switched, then we need incremental global uniform asymptotic stability (δ - GUAS). If we want to consider quantization errors, the stronger incremental global input-to-state stability is needed. The main idea is that accumulation of errors due to successive quantizations is contained by incremental stability.

Graphically,

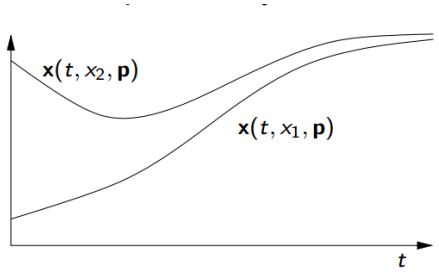

Figure 5-34: Concept of Incremental Stability

In order to test, incremental stability properties, the original dynamical model should be transformed to an equivalent polynomial one.

We begin by testing the nominal model for the dry asphalt condition. Let's recall the friction curve.

$$\mu(\lambda) = c_1(1 - e^{-c_2\lambda}) - c_3\lambda \tag{5-4}$$

It can be cast as:

$$\mu(\lambda) = \frac{d_1\lambda + d_2}{\lambda^3 + d_3 * \lambda^2 + d_4 * \lambda + d_5} \tag{5-5}$$

By minimizing the mean square error, we get the coefficients:

- $d_1 = 1.396 \cdot 10^5$
- $d_2 = -209.7$
- $d_3 = 1.324 \cdot 10^5$
- $d_4 = 1.316 \cdot 10^5$
- $d_5 = 3146$

Graphically,

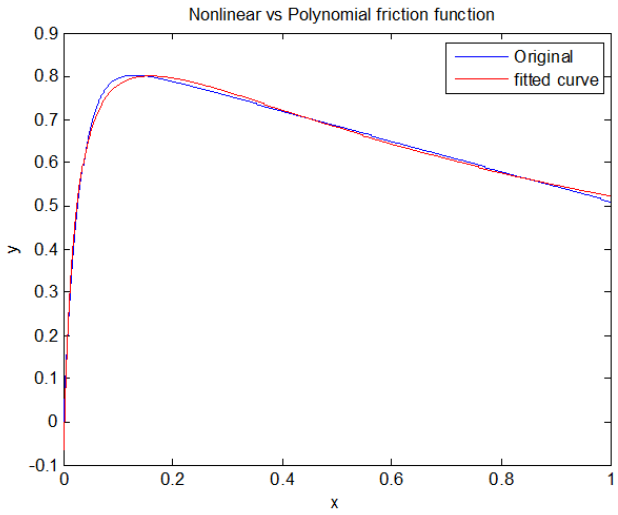

Figure 5-35: Polynomial vs Original Nonlinear Tire Friction

In order to avoid the introduction of extra switches, we have tested a continuous feedback controller. However, the closed-loop model cannot guarantee the existence of an appropriate polynomial Lyapunov function. The problem relates to the "explosion" of the system at low velocities.

The implementation was based on semi-definite programming and done with SOSTOOLS. In the end, and there could not be found any Lyapunov function (4th order) to meet the necessary constraints

That means that the state feedback controller should be redesigned and different alternatives should be sought.

5-4 Reference Generation

In traditional hybrid algorithms based on wheel acceleration, the acceleration is used both for detection and control. First, the acceleration is employed for detecting whether the tire has passed the maximum of its characteristics. This is made feasible by indirectly observing

the tire force when the brake torque remains constant. Second, the acceleration needs to be controlled to drive the tire to a limit cycle, by varying the brake torque [97]. Those two utilizations are conflicting with each other. This often requires complicated logics with many states and the alternation between phases where the brake torque is changing quickly or kept constant; this can make the tuning more difficult and affect the performance. Thanks to direct tire force measurement, the above limitations can be alleviated, the detection of the friction peak can be made more precise and the acceleration can permanently be controlled to provide tight cycles [98]. At the time being, SKF has working prototypes for tire friction sensors.

The idea of force-based control is shown in the next figure.

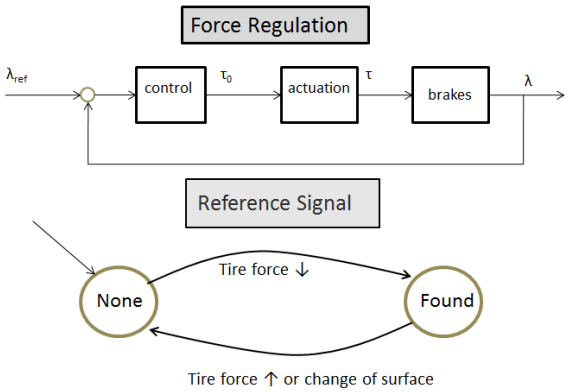

Figure 5-36: Force-based control scheme

In the beginning of the braking maneuver, the vehicle cannot recognize the road surface and does not know the optimal slip value. However, it can measure the friction force. The friction force is increasing up to a maximum point and then it starts decreasing. The maximum value of the friction force corresponds to the optimal slip value. So, by observing the time history of the brake force, we can find when the measured force reaches its maximum value and when it starts to drop. Graphically,

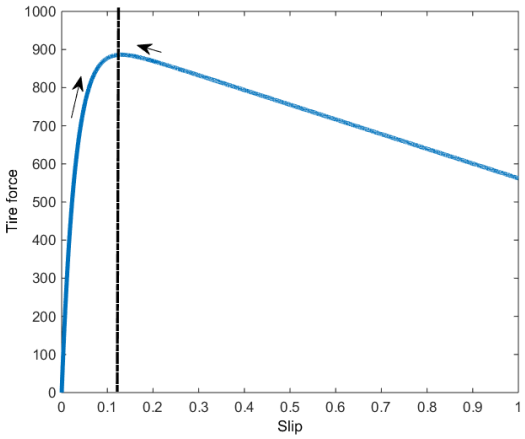

Figure 5-37: Force-based control scheme

In other words, the vehicle should apply a sufficiently large torque T_{ini} to increase the slip and the tire force. The moment the measured force starts to drop, after a period of increase, we identify the optimal slip value. Then, we may use the controllers designed in the previous section. For the scenario of the bang-bang controller, we arrive at:

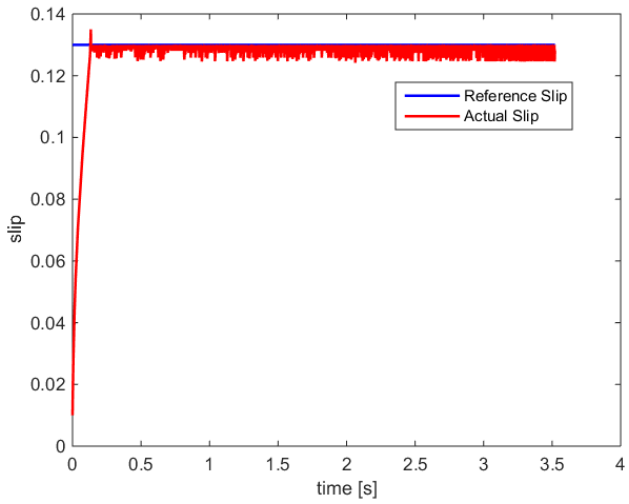

Figure 5-38: State evolution: Force-based control

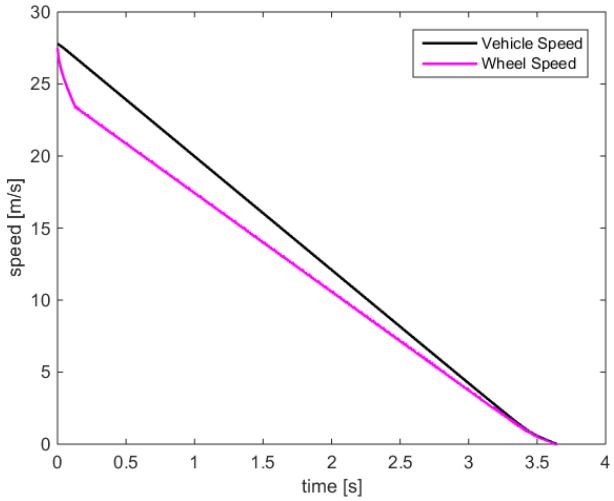

Figure 5-39: State evolution: Force-based control

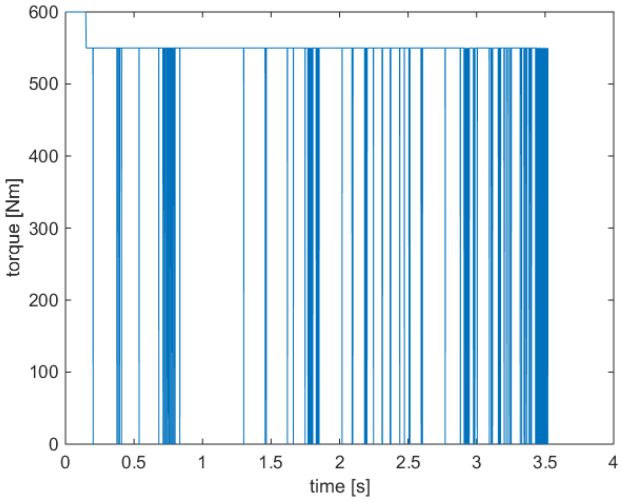

Figure 5-40: Input evolution: Force-based control

On the contrary for the nonlinear controller, the objectives cannot be reached.

5-5 Conclusions

In this chapter, several control objectives were presented and verified. It was demonstrated that the braking control system, under consideration, does not have to be deactivated at low speeds. Further to this, we are sufficiently able to guarantee for all road surfaces other than snow it is possible to select a reference slip in the region 0.12 to 0.15. This leads to superior braking performance and the actual slip remains in the range 0.08 to 0.2. Uncertainties in the road conditions and the vehicle mass have been taken into account.

Chapter 6

Conclusions and Future Work

In the context of this thesis, several braking controllers were designed and their performance was evaluated by means of formal verification. In this chapter, the relevant conclusions are presented and recommendations for further research are made.

6-1 Conclusions

This thesis project was motivated by the advent of automated road vehicles as well as the need to replace the paradigm of simulation and testing by formal methods for synthesis and verification problems. The purpose was the synthesis of braking control algorithms for a fully automated vehicle. The control specifications were to minimize the braking distance, while maintaining the steerability of the vehicle. This objective was formulated as a wheel slip regulation problem, by suitable selection of sensor and actuator technology.

For the formal synthesis part, it was necessary to transform the highly nonlinear model into a more manageable mathematical model. By partitioning the state space domain into boxes and linearizing around the centers, we arrived at an equivalent piecewise affine model. Then, this hybrid model was abstracted to a finite-state machine, by selecting proper quantization parameters for input, state and time. The abstraction process was completed, by determining the transition relations, between the finite states, through reachability analysis. It was shown that the quantization errors did not accumulate, while the original and the simplified model were equivalent up to an error term e. The control objectives were transformed into equivalent Linear Temporal Logic (LTL) "Reach and Stay" formula, and a lazy controller was constructed. The PESSOA toolbox automated the control software synthesis, with the use of alternating approximate simulation relationships. This analysis exhibited the importance of quantization parameters. In particular, for time quantization values larger than 0.001s, the specifications could not be achieved. Moreover, we demonstrated that the control system should be deactivated at low velocities.

As for the formal verification part, the original nonlinear model was used. Initially, several controllers were designed and compared against the simulated braking distance. For the nominal, deterministic scenario it was exhibited that the sliding-mode and the nonlinear controller outperformed the other controllers. However, on the case of uncertainties in the initial conditions, the bang-bang controller performed better than the sliding mode controller. Furthermore, uncertainties in the friction coefficient deteriorated the performance of the nonlinear controller (partial feedback linearization). Following a falsification process in Breach, requirements in the form of Metrics Interval Temporal Logic were tested and evaluated. Further to the above conclusions, we demonstrated the existence of a reference slip range that results to almost optimal results for every road condition. In particular, it is desired to restrict the reference signal in the range of 0.12 to 0.15, since the wheel slip will stay in the range of 0.08 to 0.02. Finally, a hybrid scheme for reference slip generation and slip regulation was designed. This approach would require the use of load-bearing sensors, but no knowledge of the road surface and the friction dynamics would be necessary.

6-2 Future Work

This section is devoted to recommendations for future work.

1. **Co-design of automotive systems:** In commercial vehicles, braking systems are connected with many other assistance systems. For example, Electronic Stability Program (ESP) and Traction Control Systems (TCS) are directly related to braking systems, sharing hardware and software functionalities. However, the design of these systems is made separately, without considering the needs, guarantees and assumptions of the rest. As such, the seamless integration of braking systems in a new vehicle may pose sincere challenges. A good starting point would be to combine the braking control with the design of active suspensions. Another alternative would be the co-design of regenerative, anti-lock and engine braking control systems. Based on this thesis work, the emergency braking problem could also be reformulated and re-evaluated. Finally, the potential of brake by steering on trucks could be assessed.

2. **Sensor limitations:** This thesis work assumed the accurate measurement of the slip variable. However, wheel slip is related to the longitudinal velocity of the vehicle; in practice, its estimation is difficult and not always reliable, especially at low speeds. There are approaches that reconstruct the wheel slip (or vehicle velocity) from indirect measurements. They are mainly divided into industrial heuristic methods and academic model-based approaches. An interesting research path would be to incorporate linear observers in the braking model and synthesize automated controllers with the correct-by-design paradigm. Another way could be the introduction of Extended Kalman Filters in the traditional control problem and perform reachability analysis to evaluate the performance of the closed-loop system.

3. **Stochastic methods:** Commercial vehicles extensively use braking systems; anti-lock braking systems, emergency braking, or cornering brake control are being incorporated in most new vehicle productions. Also, braking control poses many safety-critical issues, while potential failures could have detrimental effects. In this respect, deterministic

modeling may be too coarse. For example, the question "Is it possible for a fatal accident to happen due to braking system failures?" may be interesting, but the answer (which is most likely "yes") does not convey nearly as much information as the answers to the questions "What is the probability that a fatal accident happens?" and "How can the probability of a fatal accident be reduced?". The need for finer, probabilistic analysis of uncertain systems has led to the study of stochastic hybrid systems that allow random failures. Randomness may appear though noise, environmental uncertainties, software or hardware failures.

4. **Online tire-friction identification:** The scheme of slip reference generation, which was presented in this thesis, offers a fresh approach in online tire-friction identification. By taking advantage of SKF load-bearing sensors, this scheme could be applicable to all road conditions. A good industrial opportunity could be to introduce load sensors in electro-mechanical setups (brake-by-wire), since load sensors have only been used for actuators with discrete dynamics (hydraulic) so far.

5. **Tool development:** The increasing interest of academic society and large companies in formal methods has demonstrated the need for versatile and scalable software tools. Despite the big number of theoretical contributions, there is a limited number of tools that could be practical, tractable and user-friendly. The purpose could be the development of tools that focus more on generality and less on performance.

Appendix A

Tools for Hybrid Systems

A-1 Tools for Control Synthesis

A-1-1 TuLIP

TuLiP is a Python-based software toolbox for the synthesis of embedded control software that is provably correct with respect to an expressive subset of linear temporal logic (LTL) specifications [75]. TuLiP combines routines for

- finite state abstraction of control systems,
- digital design synthesis from LTL specifications,
- receding horizon planning.

The underlying digital design synthesis routine treats the environment as adversary. Hence, the resulting controller is guaranteed to be correct for any admissible environment profile. TuLiP applies the receding horizon framework, allowing the synthesis problem to be broken into a set of smaller problems, and consequently alleviating the computational complexity of the synthesis procedure, while preserving the correctness guarantee. In comparison with PESSOA, it directly handles all LTL specifications. However, it only admits general affine dynamics with bounded disturbances. Real time applications are presented in [76].

A-1-2 LTLMoP

The LTLMoP (Linear Temporal Logic MissiOn Planning) toolkit is a collection of Python applications for designing, testing, and implementing hybrid controllers generated automatically from task specifications written in Structured English or Temporal Logic.

LTLMoP provides a complete development environment, encapsulating each step of the controller generation and implementation process-from parsing of specifications to continuous

robot motion control and thereby helps to bridge the gap mentioned above. At the same time, LTLMoP is designed to be modular, so that research can be performed on any single component (e.g. semantic parsing or controller synthesis) in isolation, while still benefiting from the integrated system. Furthermore, by treating the robot under control as an abstract interface, LTLMoP allows seamless transition from computer simulation to physical experiment, with the same task specification. The software is written in Python and Java, and is thus cross-platform. There also user-friendly GUIs available for most tasks. On the downside, LTLMoP only considers fully actuated systems acting in the Euclidean plane [77].

A-1-3 CoSyMa

CoSyMA (COntroller SYnthesis using Multi-scale Abstractions)is a tool implementing symbolic approaches based on multi-scale abstractions to synthesize controllers for incrementally stable switched systems. CoSyMA accepts as input a switched system defined by differential equations indexed by a set of modes, time and space sampling parameters used to set an approximation of the continuous state-space, and a safety or a time-bounded reachability specification. If it exists, it computes a controller satisfying the specification. However, it can't handle both specifications at the same time. The tool is implemented using OCaml. The benchmarks provide evidence that the use of multi-scale abstractions leads to a substantial reduction of synthesis time and size of the obtained controller, while maintaining coverage of the state space [78].

A-1-4 Con-Pas/Con-Pas2

ConPAS is a computational tool for automatic synthesis of feedback control strategies for a piecewise affine (PWA) system from specifications given as Linear Temporal Logic (LTL) formulas. ConPAS consists of two main steps: First, by defining appropriate partitions for its state and input spaces, it constructs a finite abstraction of the PWA system in the form of a control transition system. Second, by leveraging ideas and techniques from Buchi games and qualitative probabilistic LTL model checking, it generates a control strategy for the finite abstraction. The tool conPAS handles only specifications that can be expressed as deterministic Buchi automata, while its extension conPAS2 can handle arbitrary LTL formulas through a translation to deterministic Rabin automata. While provably correct and robust to small perturbations in both state measurements and applied controls, both procedure are conservative and expensive [79].

A-1-5 LTLCon

LTLCon is a software toolbox for embedded control of systems from LTL formulas over linear predicates. It initially constructs a finite transition system that serves as an abstract model of the physical system. Then, a strategy, based on the given properties, is synthesized and is represented by finite state automata. LTLCon can handle affine systems and arbitrary specifications [80].

A-2 Tools for Verification

A-2-1 Hybrid Toolbox

The Hybrid Toolbox is a MATLAB/Simulink toolbox for modeling, simulating, and verifying hybrid dynamical systems, for designing and simulating model predictive controllers for hybrid systems subject to constraints, and for generating linear and hybrid MPC control laws in piecewise affine form that can be directly embedded as C-code in real-time applications [81].

The main features of the toolbox are:

- Hybrid model design and simulation
  - Mixed logical dynamical(MLD) systems are handled as MATLAB objects, obtained from HYSDEL models.
  - MLD objects can be automatically converted into piecewise affine (PWA) objects.
  - MLD and PWA objects can be simulated in MATLAB or in Simulink.
  - Safety properties can be verified through reachability analysis.
- Control design
  - Model predictive controllers based on on-line optimization (MILP / MIQP) can be designed for hybrid systems.
  - Quadratic and infinity norms are supported.
- Explicit control design
  - Piecewise affine controllers can be designed via offline optimization (multi-parametric programming) for linear systems with and without constraints and for hybrid systems.
  - MPC controllers designed for linear constrained systems with the new Model Predictive Control Toolbox for MATLAB can be also converted into piecewise affine form via multi-parametric quadratic programming.
- C-code generation

A-2-2 HyEQ

The Hybrid Equations (HyEQ) Toolbox is implemented on Matlab/Simulink for the simulation of hybrid dynamical systems. This toolbox is capable of computing approximations of trajectories to hybrid systems given in terms of differential and difference equations with constraints. The toolbox is suitable for the simulation of hybrid systems with different type of trajectories, including those that are Zeno and that have multiple jumps at the same instant. It is also capable of simulating hybrid systems without inputs, with inputs, as well as interconnections of hybrid systems [82].

A-2-3 MATISSE

MATISSE (Metrics for Approximate TransItion Systems Simulation and Equivalence) is a MATLAB toolbox for safety verification and reachable set computation of large dimensional, constrained linear systems. The main functionalities of MATISSE are:

- Given a constrained linear system, the toolbox computes a lower dimensional approximation of the system, and provides error bounds for the precision of the approximation.
- The computation of a bisimulation function between a constrained linear system and its projection.
- The computation of the reachable sets of a constrained linear system using zonotope methods.

Matisse is based on the framework of abstracting linear systems using approximate bisimulation relations. Approximate bisimulation relations aim in capturing the most significant characteristics of a system dynamics and neglect less important ones. The degree of approximation is given by the precision of the approximate bisimulation. This precision notably provides a bound of the distance between the trajectories of a system and of its abstraction. MATISSE has been used for robotic applications in [83].

A-2-4 HYSTAR

The HYSTAR is a Matlab toolbox for computational analysis and hierarchical controller synthesis of piece-wise linear hybrid dynamical systems. The analysis and design are based on computation of predecessor operator and backward reachability analysis. The specifications that HYSTAR can handle are static ones that do not change as time progresses and dynamic ones that include sequencing of events and eventual execution of actions are considered. Control design is implemented using finite automata and linear programming techniques [84].

A-2-5 SimHPN

SimHPN is a MATLAB embedded package for hybrid Petri nets. It offers a collection of tools devoted to simulation, analysis and synthesis of dynamical systems modeled by hybrid Petri nets. The package supports several server semantics for the firing of both, discrete and continuous, types of transitions. Besides providing different simulation options, SimHPN offers the possibility of computing steady state throughput bounds for continuous nets. For such a class of nets, optimal control and observability algorithms are also implemented [85].

A-2-6 VeriSiMPL

This toolbox is used to generate finite abstractions of autonomous and non-autonomous Max-Plus-Linear (MPL). Abstractions are characterized as finite-state Labeled Transition Systems (LTS). The LTS finite abstractions are shown to either simulate or to bisimulate the original MPL model. LTS models are verified against specifications expressed as formulae in Linear Temporal Logic (LTL). The toolbox intends to leverage the SPIN model checker [87].

A-2-7 Checkmate

CheckMate is a MATLAB-based tool for modeling, simulating, and verifying properties of hybrid dynamic systems. The specifications are expressed as properties of the feasible trajectories. Accordingly, the verification process determines if the given specifications are true for all trajectories starting from a polyhedral set of initial continuous states [57].

A-2-8 PHAVer

PHAVer is a tool for verifying safety properties of linear hybrid automata. It provides infinite precision arithmetic in a robust implementation, on-the-fly over-approximation of affine dynamics, and supports compositional reasoning. To manage the complexity of the underlying polyhedral computation, the authors propose heuristics to conservatively over-approximate polyhedra, by limiting the number of bits and constraints. In comparison with CheckMate, PHAVer is able to analyze dynamical systems at higher accuracy and obtain reachable sets for cases where CheckMate does not [60].

A-2-9 ProHVer

ProHVer is a tool to handle systems which feature both discrete and continuous behavior, and also involve randomness. ProHVer is capable of computing the unbounded reachability probability for a very general class of probabilistic hybrid automata. It relies on PHAVer and transforms a probabilistic hybrid automaton description to a non-probabilistic hybrid automaton description. Then it generates an abstract transition system, forming an over-approximation of the non-probabilistic hybrid automaton [91].

A-2-10 SpaceEX

The SpaceEx tool platform is designed to facilitate the implementation of algorithms related to reachability and safety verification. While these methods are based on different representations (polyhedra, zonotopes) and are tailored to different dynamics (piecewise constant, affine, multi-affine, nonlinear), they have several things in common: The model is a composition of hybrid automata (including extensions such as hierarchy and templates). Basic components of analysis algorithms are post- and pre-operators in various flavors. The reachable states are explored using symbolic states. They require basic infrastructure such as parsing input and visualizing states. The SpacEx framework provides the common components and developers can substitute components as well as easily add new functionality. It extensively makes use of polymorphism to enable the development of heterogeneous analysis methods, such as using different set representations in different parts of the state space or at different levels of refinement, or combining symbolic computations with simulation. In general, this framework allows all set operators (union, intersection, etc.) to return a set of a different type. The development of the framework was spawned by recent progress in finding efficient data structures and algorithms for reachability computation [92].

A-2-11 KeYmaera

KeYmaera is a hybrid verification tool for hybrid systems that combines deductive, real algebraic, and computer algebraic prover technologies. It is an automated and interactive theorem prover for a natural specification and verification logic for hybrid systems. KeYmaera supports differential dynamic logic, which is a first-order dynamic logic for hybrid programs, a program notation for hybrid systems. KeYmaera also supports hybrid systems with nonlinear discrete jumps, nonlinear differential equations, differential-algebraic equations, differential inequalities, and systems with non-deterministic discrete or continuous input. For automation, KeYmaera implements a free-variable sequent calculus and automatic proof strategies that decompose the hybrid system specification symbolically. This compositional verification principle helps scaling up verification, because KeYmaera verifies a big system by verifying properties of subsystems. To overcome the complexity of real arithmetic, the authors have integrated real quantifier elimination following an iterative background closure strategy [46].

A-2-12 S-TaLiRo

TaLiRo (TemporAl LogIc RObustness) is a tool for the verification of hybrid systems with specifications expressed in Temporal Logic. S-TaLiRo (Systems TaLiRo) is a software toolbox for Matlab for the temporal logic robustness analysis of discrete time signals that take values in metric spaces. S-TaLiRo is based on the principle of the Robustness Guided Model Checking (RGMC). In essence, it searches for trajectories of minimal robustness and is useful in the analysis of complex systems. It utilizes stochastic optimization methods such as Monte-Carlo, Ant-Colony optimization, Genetics Algorithms and Cross Entropy [86].

A-2-13 HybridSAL

To become practical for assurance, automated formal methods must be made more scalable, automatic, and cost-effective. Such an increase in scope, scale, automation, and utility can be derived from an emphasis on a systematic separation of concerns during verification. SAL (Symbolic Analysis Laboratory) attempts to address these issues. It is a framework for combining different tools to calculate properties of concurrent systems. The heart of SAL is a language, developed for specifying concurrent systems in a compositional way. It is supported by a tool suite that includes symbolic (BDD-based) and bounded (SAT-based) model checkers, an experimental "Witness" model checker, and an "infinite" bounded model checker based on SMT solving[1]. HybridSal offers a language extension to SAL for specifying Hybrid Systems and a tool that performs hybrid abstraction to automatically generate discrete SAL specifications that can be model checked by other SAL tools [94].

A-2-14 Ariadne

Ariadne is a C++ package for set-based analysis of dynamical and control systems, including reachability analysis, robust simulation and safety verification. The package can handle systems with reset, flow and guard predicates given by nonlinear functions, making it a general-purpose reachability tool [93].

[1] Auxiliary tools include a simulator, deadlock checker and an automated test generator.

A-2-15 d/dt

The tool d/dt is used for reachability analysis of continuous and hybrid systems with linear differential inclusions. It offers solution for hybrid automata which have linear continuous dynamics and uncertain bounded inputs, while all invariants and transition guards are defined by convex polyhedra. This tool offer solutions to reachability problems, safety verification and safety switching control synthesis. It is considered a refinement of Checkmate and the flowpipe method is used for over-approximation of the reachable sets [88].

A-2-16 HSolver

HSolver is a software package for the formal verification of safety properties of continuous time hybrid systems over unbounded time. It allows hybrid systems with non-linear ordinary differential equations, and non-linear jumps. Even though it is based on fast machine-precision floating point arithmetic, it uses sound rounding, and hence the correctness of its results cannot be hampered by round-off errors. HSolver can be used to verify safe hybrid systems, and compute abstractions of the input system. So, even for input systems that are unsafe, or for which exhaustive formal verification is too difficult, it will compute abstractions that can be used by other tools.

The method used by HSolver is interval constraint propagation based abstraction refinement. This method incrementally refines an abstraction of the input systems. Special care is taken to reflect as much information as possible into the abstraction without increasing its size. However, it is optimized for special classes of hybrid systems and it does not provide support for finding counter-examples for unsafe input systems [89].

A-2-17 HYSDEL

HYSDEL (Hybrid System DEscription Language) allows modeling a class of hybrid systems described by interconnection of linear dynamic systems, automata, if-then-else and propsostional logic rules. Once a hybrid system is modeled in a human-readable fashion, HYSDEL transforms it to the mixed-logical dynamical (MLD) form which can be immediately used for optimization, to solve optimal control, safety verification, or estimation and fault detection problems [90].

Bibliography

[1] "U.S. Department of Transportation Releases Policy on Automated Vehicle Development". National Highway Traffic Safety Administration, 2013.

[2] Davi official website, http://davi.connekt.nl/.

[3] R. Hoogendoorn, B. van Arem, R. Happee, M. M. Espinoza, and D. Kotiadis, "Towards safe and efficient driving through vehicle automation: The dutch automated vehicle initiative"

[4] K. Ogata, Discrete-time Control Systems.

[5] R. Rajkumar, "Cyber-physical systems: the next computing revolution", 2010.

[6] M. Althoff, Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars.

[7] M. Egerstedt, E. Fazzoli, G. Pappas,Special Section on Symbolic Methods for Complex Control Systems.

[8] R. Alur, Formal Verification of Hybrid Systems

[9] M. Collins, Formal Methods.

[10] Bowen, Stavridou. Safety Critical Systems, Formal Methods And Standards.

[11] R. Kling. Systems Safety, Normal Accidents And Social Vulnerability".

[12] E. M. Clarke, O. Grumberg, and D. A. Peled. Model Checking. MIT Press, 2000.

[13] R. Alur and D. L. Dill. A theory of timed automata.

[14] J. Lygeros, S. Sastry, and C. Tomlin. The Art of Hybrid Systems.

[15] Hespanha, Hybrid and Embedded Systems, http://www.ece.ucsb.edu/~hespanha/ece229/Lectures/Lecture3.pdf.

[16] S. Savaresi and M. Tanelli. Active Braking Control Systems Design for Vehicles.

[17] Automotive Handbook, 8th edition. Bosch, Wiley.

[18] R. Bosch, Automotive Electrics and Automotive Electronics. Wiley, 5th Edition, 2007

[19] T. Gillespie, Fundamentals of Vehicle Dynamics.

[20] G. Pola, P. Tabuada. Symbolic models for linear control systems with disturbances.

[21] P. Tabuada. On the synthesis of correct-by-design embedded control software. Available at: https://chess.eecs.berkeley.edu/pubs/577/CorrectByDesign.pdf

[22] A. Girard. Symbolic Control of Incrementally Stable Systems. Available at: http://www-ljk.imag.fr/membres/Antoine.Girard/Talks/LCCC-Girard.pdf

[23] Hybrid Modelling and Reachability on Autonomous RC-Cars. Thesis, Lygeros.

[24] A. Morgenstern. Symbolic Controller Synthesis for LTL Specifications.

[25] P. Tabuada. Verification and Control of Hybrid Systems. Springer, 2009.

[26] Reachability Analysis. T, Koo. Course in Embedded Systems.

[27] J. Lygeros, S. Sastry, and C. Tomlin. Hybrid Systems: Foundations, advanced topics and applications.

[28] J. Lygeros, et al. Dynamical Properties of Hybrid Automata.

[29] A. Girard. Controller synthesis for safety and reachability via approximate bisimulation.

[30] A. Pnueli, The temporal logic of programs.

[31] Y. Gao, The Reachability Problem for Uncertain Hybrid Systems Revisited: A Viability Theory Perspective.

[32] Homepage of Pessoa. https://sites.google.com/a/cyphylab.ee.ucla.edu/pessoa/

[33] M.Mazo, A. Davitian, and P. Tabuada. PESSOA: towards the automatic synthesis of correct-by-design control software. HSSC, 2010.

[34] M.Mazo, A. Davitian, and P. Tabuada. PESSOA: A tool for embedded control software synthesis.

[35] M. Kvasnica, P. Grieder, and M. Baotic. Multi-Parametric Toolbox (MPT), 2004.

[36] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. Multi-Parametric Toolbox 3.0.

[37] J. Lofberg. YALMIP : A toolbox for modeling and optimization in MATLAB.

[38] J. F. Sturm. Using SEDUMI 1.02, a MATLAB toolbox for optimization over symmetric cones.

[39] J. Ouaknine and J. Worrell, Some Recent Results in Metric Temporal Logic.

[40] A. Tarski. A lattice theoretical fixed point and its applications.

[41] Tarski lemma.

[42] J. Nation. Lattice Theory.

[43] T. Wongpiromsarn, Formal methods for design and verification of embedded control systems: Application to an autonomous vehicle.

[44] C. Baier and J. Katoen. Principles of Model Checking (Representation and Mind Series).

[45] K. McMillan. Symbolic Model Checking.

[46] Keymaera: A hybrid theorem prover for hybrid systems.

[47] S. Owre et al., PVS System Guide.

[48] I. M. Mitchell. Comparing forward and backward reachability as tools for safety analysis.

[49] S. Prajna, A. Papachristodoulou, and P. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver.

[50] S. Prajna and A. Rantzer. Primal-dual tests for safety and reachability.

[51] U. Topcu, A. Packard, and P. Seiler. Local stability analysis using simulations and sum-of-squares programming.

[52] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. Dullerud. Stormed hybrid systems.

[53] A. Papachristodoulou and S. Prajna. Analysis of non-polynomial systems using the sum of squares decomposition.

[54] Alexandre Donze. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems.

[55] C. Le Guernic. Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics.

[56] O. Maler and G. Batt. Approximating continuous systems by timed automata.

[57] Checkmate: Hybrid system verification toolbox for MATLAB.

[58] E. Asarin, et al. Recent progress in continuous and hybrid reachability analysis.

[59] Antoine Girard, Reachability of uncertain linear systems using zonotopes.

[60] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech.

[61] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification.

[62] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems.

[63] A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis.

[64] O. Stursberg and B. Krogh. Efficient representation and computation of reachable sets for hybrid systems.

[65] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions.

[66] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems.

[67] T. Dang, C. Le Guernic, and O. Maler. Computing reachable states for nonlinear biological models.

[68] T. Dang. Approximate reachability computation for polynomial systems.

[69] Hans Pacejka, Tyre and Vehicle Dynamics.

[70] C. Canudas, P. Tsiotras Dynamic Tire Friction Models for Vehicle Traction Control.

[71] E. Velenis, C. Canudas, P. Tsiotras, M. Sorine. Dynamic Tire Friction Models for for Combined Longitudinal and Lateral Vehicle Motion.

[72] D. Chou. Dahl Friction Modeling.

[73] R. Castro, et al.,Hybrid Modeling, Control and Estimation in ABS Applications based on In-Wheel Electric Motors. L. Bascetta, G. Magnani, and P. Rocco. Velocity Estimation: Assessing the Performance of NonModel-Based Techniques.

[74] H. Raza, Z. Xu, P. Ioannou, B. Yang. Modeling and Control Design for a Computer Controlled Brake System.

[75] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. Murray.

[76] R. Murray, Specification, Design, and Verification of Networked Control Systems.

[77] LTLMop toolkit. Available at http://ltlmop.github.io/

[78] Sebti Mouelhi. CoSyMA: A Tool for Controller Synthesis Using Multi-scale Abstractions.

[79] B. Yordanov and C. Belta. conPAS2. http://hyness.bu.edu/conPAS2.html

[80] M. Kloetzer and C. Belta. A Fully Automated Framework for Control of Linear Systems from LTL Specifications.

[81] A. Bemporad. Hybrid Toolbox. Electronically available at: http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/

[82] R. Sanfelice, D. Copp, P. Nanez. Hybrid Equations (HyEQ) Toolbox. A Toolbox for Simulation of Hybrid Systems in Matlab/Simulink.

[83] A. Girard, A. Julius, and G. Pappas. MATISSE: Metrics for Approximate Transition Systems Simulation and Equivalence.

[84] H. Lin, X. Koutsoukos, P. Antsaklis. HYSTAR: A ToolBox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems.

[85] J. Julvez, C.Mahulea, C-R Vazquez. SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets.

[86] TALIRO (TemporAl LogIc RObustness) tools. Available at: https://sites.google.com/a/asu.edu/s-taliro/home

[87] VeriSiMPL: Verification via biSimulations of Max-Plus Linear models.

[88] Homepage of d/dt. Available at: http://www-verimag.imag.fr/~tdang/Tool-ddt/ddt.html

[89] Hsolver homepage. http://hsolver.sourceforge.net/

[90] HYSDEL (Hybrid System DEscription Language) toolbox.

[91] ProHVer. Electronically available at: http://depend.cs.uni-sb.de/tools/prohver/

[92] SpaceEx toolbox. Available at http://spaceex.imag.fr/

[93] Ariadne: An open tool for hybrid system analysis.

[94] Hybridsal. Electronically available at: http://sal.csl.sri.com/hybridsal/.

[95] A. Donze. Breach Toolbox: Instrumenting Simulation and Signal Temporal Logics for the Analysis of Hybrid Systems.

[96] M. Schinkel and K. Hunt, Anti-lock braking control using a sliding mode like approach.

[97] R. Castro, F. Todeschini, R. Araujo, S. Savaresi, M. Corno and D. Freitas. Adaptive-robust friction compensation in a hybrid brake-by-wire actuator.

[98] M. Corno, et al., Hybrid ABS Control Using Force Measurement.

[99] M. Zamani, G. Pola, M. Mazo, P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions.

[100] G. Pola. Discrete abstractions of continuous control systems.

[101] P. Nilsson, et al.,Preliminary Results on Correct-by-Construction Control Software Synthesis for Adaptive Cruise Control.

Glossary

List of Acronyms

ADAS	Advanced Driver Assistance Systems
ABS	Anti-lock Braking Systems
CPS	Cyber-Physical Systems
NHTSA	National Highway Traffic Safety Administration
ESC	Electronic Stability Control
DAVI	Dutch Automated Vehicle Initiative
LTL	Linear Temporal Logic
PID	Proportional-Integral-Derivative
MITL	Metrics Interval Temporal Logic
EMB	Electro Mechanical Brakes
HAB	Hydraulic Actuated Brakes
EHB	Electro Hydraulic Actuated Brakes
MPT	Multi-Parametric Toolbox
MPC	Model Predictive Control
ROBDD	Reduced Order Binary Decision Diagrams
PWA	Piecewise Affine