Control of High-Dimensional Systems with Applications in Transportation

Stanley Smith

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-175
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-175.html

August 9, 2021

Control of High-Dimensional Systems with Applications in Transportation

by

Stanley W. Smith

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Murat Arcak, Chair
Professor Francesco Borrelli
Professor Sanjit A. Seshia

Summer 2021

Control of High-Dimensional Systems with Applications in Transportation

Abstract

Control of High-Dimensional Systems with Applications in Transportation

by

Stanley W. Smith

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Murat Arcak, Chair

Important control applications in the fields of robotics and transportation frequently involve high-dimensional systems; for example, robots with many degrees of freedom, or groups of connected automated vehicles (CAVs). In safety-critical applications, the system of interest must also meet a complex set of requirements formalized in a control specification. Formal methods for control synthesis are useful in such applications since they partially automate the controller design process, while also providing guarantees that the control specification will be satisfied. However, when the system dimension becomes too large, such techniques cannot be directly applied due to their computational complexity.

This dissertation focuses on addressing this challenge using hierarchical and distributed control techniques, with an emphasis on applications in transportation. To enable control synthesis for high-dimensional systems, in Chapter 2 we propose techniques for constructing approximate abstractions of a class of interconnected control systems, to be used in a hierarchical control framework. On the application side, in Chapters 3 and 4 we explore the potential for CAVs to improve the efficiency and safety of traffic flows at intersections. For improving efficiency, we discuss how traffic throughput at intersections can be increased by forming vehicle platoons. In particular, this is accomplished using a distributed model-predictive control approach, which is enabled by vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. For improving safety, we show how safe behaviors can be accurately characterized for certain challenging driving maneuvers, such as unprotected left turns.

Contents

Contents **i**

List of Figures **iii**

List of Tables **vi**

1 Introduction **1**
1.1 Control Synthesis for Large-Scale Systems 1
1.2 Applications in Transportation 2
1.3 Further Background and Summary of Contributions 3

2 Approximate Abstractions of Control Systems **11**
2.1 Introduction 11
2.2 Control Systems 12
2.3 Abstraction Synthesis for Linear Systems 15
2.4 Compositionality 18
2.5 Aggregation 22
2.6 Example 30
2.7 Proofs of Main Results 33

3 Vehicle Platooning **40**
3.1 Introduction 40
3.2 Platoon Model and Management 42
3.3 MPC Formulation 50
3.4 Safety Constraints and MPC Solution 54
3.5 Simulation Results 59
3.6 Experimental Results 62

4 Safety in Real Driving Scenarios **67**
4.1 Introduction 67
4.2 Monotonicity Concepts 68
4.3 Vehicle-Following Scenario 69
4.4 Unprotected Left Turn Scenario 76

5 Conclusion and Future Directions **81**

Bibliography **83**

List of Figures

1.1 As part of an experimental research effort, we have developed a scaled-car test platform for conducting various experiments. We integrated our test vehicle's hardware based on the f1tenth platform (see https://f1tenth.org/about.html). The test platform was developed in collaboration with Sirej Dua, Milad Noori, Canberk Hurel, and Nicholas Liu. 4
1.2 The three possible traffic movements (through, left turn, and right turn) are shown for one of the lanes approaching the intersection. While there are a total of twelve possible traffic movements at the intersection, only a fraction of these movements are allowable simultaneously. This means the intersection's capacity is reduced by a factor g_i/T as in (1.4). 6
1.3 A photograph taken after the crash that occurred between an Uber automated Volvo and a Honda CR-V in Tempe, Arizona. 9
1.4 Conflict zone (shown in orange) between a through movement and a left turn movement at an intersection. To ensure safety, only one vehicle can occupy the conflict zone at a time. 10

2.1 An interconnection of N control subsystems $\Sigma_1, \ldots, \Sigma_N$. 19
2.2 An equitable partition of a circle graph with $L = 5$ nodes into three groups (left). Note that the partition into two groups (right) is *not* equitable. 24
2.3 Simulation results for the temperature regulation example. We require the temperature in each area of the building to reach its corresponding target temperature range (indicated by the dashed lines) within 20 minutes after the signal is triggered. The signal is triggered at the 20 minute mark - the aggregate system (left) reaches the temperature target within 20 minutes, and the concrete system (right) closely follows the reference. 33

3.1 Test vehicles at the Hyundai-KIA Motors California Proving Grounds, California City, CA. 41
3.2 Depiction of the states for a platoon of size $N = 3$ and public lead vehicle approaching an upcoming traffic light. 43

3.3 Flow of V2V messages for a platoon of size $N = 4$, where the blue node represents the leader vehicle and the grey node represents the rear vehicle. Figure 3a shows the transmission of velocity forecasts and 3b shows the transmission of GPS coordinates from the rear vehicle (used by the leader to determine if the platoon can make it through the intersection, see Section 3.2). Figure 3c shows how we share radar measurements when we use the second method for estimating $s_i(t)$ as in (3.10). 47
3.4 A diagram of the transitions in our finite state machine, shown here for the leader vehicle for simplicity. 49
3.5 Experimental data collected in Arcadia, CA during the platoon formation process. The vehicles begin at a low speed and unequal spacing. At around the 2s mark, the platoon leader proposes a 'plan' which is accepted by the following vehicles, and the plan status signal (plotted above) switches from 0 to 1. This engages all platooning controllers simultaneously, and the vehicles quickly converge to the desired speed and distance. 50
3.6 In 3.6a and 3.6b we plot the terminal sets (3.28) and (3.30) for the front vehicle and upcoming intersection, respectively. For computing the sets, we use $a_{\text{min,brake}} = 3.2$ m/s^2 and $a_{\text{max,brake}} = 5.0912$ m/s^2. 56
3.7 View from the middle platooning vehicle as it approaches an intersection during our demonstration in Arcadia, CA. In Figure 3.7a there is a slow-moving truck attempting to turn right ahead of the leader vehicle. Since the truck takes priority over the intersection at this point, the platoon is forced to slow down. In Figure 3.7b the truck completes the right turn and priority shifts to the intersection. . . 57
3.8 Simulation results for an urban traffic scenario with a non-platooning lead vehicle and multiple signalized intersections. In the top plot, we show the position of all simulated vehicles (including the public vehicle which is not platooning), as well as the position of each intersection which has either a yellow or red phase. In the bottom three plots, we show the inter-vehicle distances (including the distance from the leader to the public vehicle), velocities, and torque commands for the platooning vehicles. 60
3.9 Depiction of the on-board hardware setup for the test vehicles. The local CAN bus (in red) connects the computational devices (Matrix embedded PC and dSPACE MicroAutoBox) to the Cohda OBU for DSRC communication. The HCU (CAN gateway) provides an interface between the local CAN bus and the production systems of the test vehicle. Using the local CAN bus and the gateway functionality of the HCU, we can send commands and access measurements to and from the production systems without needing access to proprietary vehicle data. . . . 63

3.10 Experimental results from the Hyundai-KIA Motors California Proving Grounds with the test vehicles shown in Figure 3.1. Here, we had the platoon track a reference trajectory which was generated via our simulation tool. The position, inter-vehicle distance, velocity, and MPC torque command for each vehicle are shown in each subplot, respectively. The desired distance between vehicles was 6 meters. 64
3.11 Overhead view of the platoon crossing an intersection in Arcadia, CA. 65

4.1 The state space is divided into three areas: the area corresponding to set S (bottom left cell), the area corresponding to unsafe impacts (top left cell), and the area where no impact has occurred (right cells). 73
4.2 Boundary of safe set $Z \subset X$ for the strict (top surface) and relaxed (bottom surface) vehicle-following specification. The safe sets lay below the depicted boundaries. 75
4.3 Depiction of the states for the ego (blue) and oncoming (yellow) vehicle in the unprotected left turn scenario. 76
4.4 Safe set boundaries for the unprotected left turn scenario. In Figure 4.4a all states below the surface are in Z^{wait}. Conversely, in Figure 4.4b all states above the surface are in Z^{go}. 78
4.5 Simulation results for the unprotected left turn scenario. The input bounds are indicated with dotted red lines. We note two vehicles never occupy the intersection (bounded by the dotted purple lines) simultaneously. 79

List of Tables

2.1 Partitioning of the 30 rooms into 3 groups. 32

3.1 Model Parameters 44
3.2 MPC Parameters 52
3.3 Improved Throughput 62
3.4 Baseline Throughput 62

Acknowledgments

First of all, I want to thank Murat Arcak for being a great advisor and mentor during my time at UC Berkeley. I appreciate Murat's encouragement to take on challenging projects that ended up being very rewarding in the end, and the trust he placed in my work throughout my PhD. I also thank Murat for the freedom he gave me to explore my intellectual interests, which enabled me to discover the research areas I am most passionate about.

Thanks to Alan Brown, who piqued my initial interest in control engineering during my first internship at HELLA Electronics. Thanks as well to Necmiye Ozay, Petter Nilsson, and Jessy Grizzle for guiding me as I started doing research at the University of Michigan and while I was applying to graduate programs.

Thanks to Majid Zamani for your collaboration on the hierarchical control work, and also to Galaxy Yin for helping extend this work to incorporate sum-of-squares programming methods. Thanks as well to Adnane Saoud for your collaboration on the work involving real driving scenarios, and for your helpful advice. Thanks also to Matthew Wright for your advice and interesting discussions which led me to some of the research presented in this thesis. Thanks to Sanjit Seshia for having me as a GSI in EECS 149/249A (a great experience!), and for being on my qualifying exam and dissertation committees.

To the many people I worked with on the vehicle platooning project - especially Yeojun Kim, Jacopo Guanetti, and Bruce Wootton - thanks for your collaboration and for the truly memorable experiences working on the project. Thanks as well to Alex Kurzhanskiy, Roberto Horowitz, Ching-Yao Chan, and Pravin Varaiya for providing guidance throughout which helped us make progress. Of course, special thanks to Francesco Borrelli for his leadership on the project, and also for being on my qualifying exam and dissertation committees. Finally, thanks to Ruolin Li, Roya Firoozi, Mikhail Burov, Galaxy Yin, and Emmanuel Sin for helping conduct experiments with the test vehicles.

Thanks to everyone in the Arcak lab for pleasant chats over lunch and for creating a welcoming and fun atmosphere in our group: John Maidens, Marcella Gomez, Eric Kim, Mindy Perkins, Galaxy Yin, Emmanuel Sin, Pierre-Jean Meyer, Mikhail Burov, Kate Schweidel, Alex Devonport, and Adnane Saoud.

Thanks to the Berkeley gang: Gautam Gunjala, Alan Dong, Alain Anton, Alex Reinking, Chandan Singh, Kieran Peleaux, Phong Nguyen, and David Ren for many fun times which really enriched my PhD experience. Thanks as well to Kristina Monakhova, Kevin Laeufer, and Michael Dennis for your friendship and for getting me more into hiking and camping. Finally, thanks to the many friends from Michigan I've kept in touch with during my PhD, especially Mike Choe, Kevin Zywicki, Trevor Brust, Sameer Desai, and Mostafa Shuva. It has been great spending time with you all whenever I am visiting home.

Of course, thanks so much to my family for their love and encouragement as I pursued my PhD. Thanks to my parents Anthony and Michele and my siblings Madeline and Nathan for inspiring me to always do my best, and for always being there for me. Finally, thanks as well to all of my extended family for your support during my graduate studies.

Finally, I would like to acknowledge the generous support I received from the National Science Foundation, the Department of Transportation, the Air Force Office of Scientific Research, and the Office of Naval Research, which made the research in this thesis possible. Lastly, I am also grateful for the support I received from the Department of Defense through the National Defense Science and Engineering Graduate Fellowship program.

Chapter 1

Introduction

This dissertation focuses on the task of designing controllers for large-scale systems. In particular, we are mainly interested in formal methods for *control synthesis*, where the controller is generated using a partially-automated design procedure, and the control specification is typically described using a formal language such as *linear temporal logic* (LTL) [45] or *signal temporal logic* (STL) [35]. An important benefit of such formal specification languages is their ability to rigorously describe the requirements that are imposed on the system of interest. For example, a reach-avoid specification, which requires a vehicle to reach a target area while avoiding several obstacles along the way, is common in robotics and transportation, and can be easily represented as an LTL / STL specification. While formal synthesis procedures are desirable for these reasons, a well-known limitation of these methods is that they usually cannot be applied directly to large-scale systems due to the complexity of the required computations. Because of this, recent research has focused on developing different methods for addressing this issue (see, for example, [38]).

1.1 Control Synthesis for Large-Scale Systems

There are a number of approaches for making control synthesis computationally tractable when dealing with large-scale systems. In hierarchical control, for example, the goal is to instead use an *abstraction* in the control synthesis procedure, where the abstraction is a simplified representation of the original control system with a low state dimension, thereby making it amenable to formal synthesis techniques. We consider two primary types of abstraction in this dissertation: *continuous abstraction* and *discrete abstraction*, which are relevant in Chapters 2 and 4, respectively. In continuous abstraction (see, for example, [1]), we represent the system of interest with a continuous-state control system, where the state, inputs, and outputs are continuous variables and, hence, each of their respective domains is infinite. Similarly, in discrete abstraction (see [66]), we represent the system of interest with a discrete-state (and usually discrete-time) control system, where the state, inputs, and outputs are discrete variables, each belonging to a respective finite domain.

Several other methods employ a divide-and-conquer approach for control synthesis, where the goal is to break down the large-scale system into smaller subcomponents, each of which is more manageable in size. This allows one to do control synthesis for each subsystem separately, where the system of interest is modelled as an interconnection / composition of these subsystems. In Chapter 2, for example, we consider a general class of interconnected control systems, where the inputs of each subsystem depend (linearly) on the outputs of the other subsystems. Techniques for cases where the subsystems affect each other in other ways are also available - for example, [60] considers an application where the dynamics of each subsystem depend (possibly nonlinearly) on the dynamics of the other subsystems.

1.2 Applications in Transportation

Throughout the dissertation, we will focus on applications in the area of transportation, where hierarchical and compositional control design procedures can be very useful. For example, an interesting application of compositional control synthesis can be found in transportation: in advanced driver-assistance systems (ADAS), the longitudinal and lateral dynamics of the vehicle are typically handled by separate controllers, each of which also has a separate control specification. The longitudinal controller can, for example, ensure that a safe distance to the lead vehicle (when present) is maintained at all times, and the lateral controller can ensure that the ego vehicle does not stray out of its desired lane. Since the longitudinal and lateral vehicle dynamics are interdependent, however, the overall vehicle dynamics must be treated as a composition of the longitudinal and lateral subsystems, and the effect of each subsystem on the other must be taken into account during the control synthesis process [60]. Another important transportation application of interest is *vehicle-platooning* (see, for example, [37]), where a group of vehicles coordinate their motion in order to move together in a tight formation. When the platoon has a large number of vehicles in it, the system state dimension becomes large and we are faced again with the same problem discussed previously. Therefore, in this application it is much more desirable to use a distributed control approach, where each vehicle has its own separate longitudinal controller, and the group of vehicles coordinate their motion using vehicle-to-vehicle (V2V) communication.

We are also interested in traffic flows at intersections and, more specifically, how we can improve the ways that traffic moves through intersections. Indeed, intersections are complex environments, and navigating them safely is a challenging task even for a human driver. Furthermore, intersections form bottlenecks in the flow of traffic, as we will discuss in Section 1.3, and therefore they significantly decrease the efficiency of traffic. To address these issues, we will explore the potential for connected automated vehicles (CAVs) to improve the efficiency and safety of traffic flows at intersections in Chapters 3 and 4.

1.3 Further Background and Summary of Contributions

This dissertation is divided into three main parts: Chapter 2 focuses on using approximate abstractions for hierarchical control, Chapter 3 focuses on vehicle platooning, and Chapter 4 focuses on ensuring safety in real driving scenarios. The remainder of the introduction provides further background and motivation for the research presented in this dissertation, as well as a summary of any preliminary papers that led to the work contained in each chapter.

Chapter 2: Approximate Abstractions of Control Systems

To provide further intuition behind continuous abstraction, we begin by walking through an example application in *vehicle motion planning*. The goal in motion planning is to have a vehicle navigate through its environment and eventually reach a desired destination. To do so safely, the vehicle may also have to avoid several (potentially mobile) obstacles in its environment at all times. At a high level, the inputs to the motion planning algorithm are a desired destination and the safety constraints imposed on the vehicle, and the outputs are the final sequence of acceleration and steering angle commands to be sent to the vehicle's actuators. Indeed, this problem falls neatly into a hierarchical control framework. The control objective for the vehicle can be represented as a reach-avoid specification using LTL or STL, and the motion planning problem itself can be naturally decomposed into separate *planning* and *control* tasks. In particular, the planner system processes the system inputs in order to produce a safe pathway for the vehicle, and the control system then converts the resulting pathway into the final system outputs. Indeed, these systems interact with each other in a hierarchical fashion: the planner system takes into account all of the high-level requirements imposed on the vehicle, and the control system then handles the low-level control of the vehicle actuators. This framework is especially useful since we can use a simplified vehicle model at the planning stage, as the path-planning algorithms used by the planner system are often computationally intensive. Therefore, the model used by the planner system can be viewed as an abstraction of the true vehicle dynamics.

It turns out that this framework is also the standard approach used to do motion planning for autonomous vehicles (AVs). Indeed, two important elements of the AV software stack are the *planner* and *control* systems [18], which interact in the same hierarchical fashion discussed above. In particular, the planner system analyzes information from the vehicle's onboard sensors (and V2X communication module, if present) and then uses this information to compute a safe pathway through the surrounding environment. In doing so, the planner system must, for example, make use of the predicted trajectories of nearby agents - such as other vehicles, bicyclists, and pedestrians - in order to always maintain a safe distance to these agents (this is part of the motivation for V2V communication, since it allows nearby vehicles to broadcast their planned trajectories). The control system then commands the

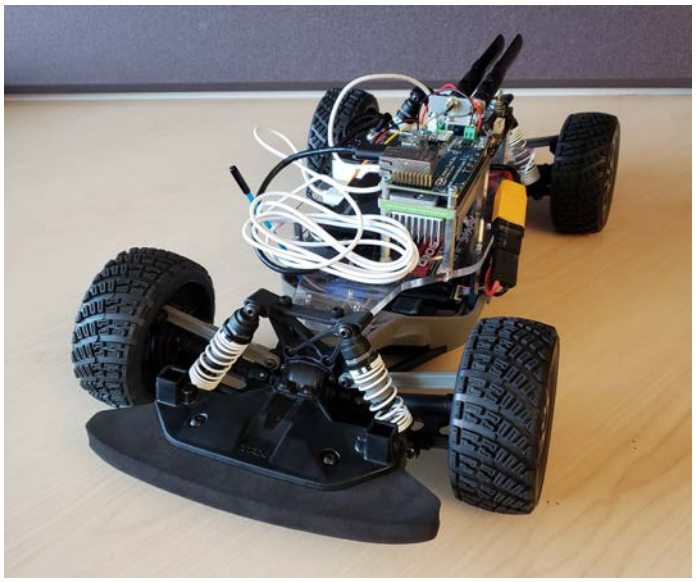


Figure 1.1. As part of an experimental research effort, we have developed a scaled-car test platform for conducting various experiments. We integrated our test vehicle's hardware based on the f1tenth platform (see https://f1tenth.org/about.html). The test platform was developed in collaboration with Sirej Dua, Milad Noori, Canberk Hurel, and Nicholas Liu.

vehicle steering angle and wheel torque such that the AV accurately follows the planned pathway with small tracking error. So long as this tracking error is small enough, the AV will meet the given reach-avoid specification.

To further elucidate the hierarchical control framework, we now discuss some models that could be used by the planner and control systems in our motion planning example. Recently, we have also developed a scaled car (see Figure 1.1) to be used as a test platform for our approach. For the planner system we can use a *Dubin's vehicle* model:

$$\dot{\hat{p}}_x(t) = \hat{v}(t)\cos(\hat{\theta}(t)), \tag{1.1a}$$

$$\dot{\hat{p}}_y(t) = \hat{v}(t)\sin(\hat{\theta}(t)), \tag{1.1b}$$

$$\dot{\hat{\theta}}(t) = \hat{u}(t), \tag{1.1c}$$

where the states $(\hat{p}_x(t),\ \hat{p}_y(t))$ and $\hat{\theta}(t)$ are the vehicle's position and heading, and the control inputs $\hat{v}(t)$ and $\hat{u}(t)$ are the vehicle's velocity and turning rate. Indeed, the Dubin's vehicle model captures the essence of the vehicle's motion in a simplified three-dimensional model. With its low state dimension, it can be used to compute pathways through the vehicle's environment with limited computational resources, making it a good choice of model for the planner system. We note, however, that (1.1) does not contain the low-level control inputs to be implemented on the vehicle's actuators, such as the acceleration and steering angle commands. Furthermore, (1.1) does not take into consideration other elements that impact the vehicle's motion, such as the size and dimensions of the vehicle itself. Due to this, for

the control system we can use a more realistic *kinematic bicycle* model:

$$\dot{p}_x(t) = v(t)\cos(\theta(t) + \beta), \tag{1.2a}$$
$$\dot{p}_y(t) = v(t)\sin(\theta(t) + \beta), \tag{1.2b}$$
$$\dot{\theta}(t) = \frac{v(t)}{l_r}\sin(\beta), \tag{1.2c}$$
$$\dot{v}(t) = a(t), \tag{1.2d}$$
$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta_f)\right) \tag{1.2e}$$

where the control inputs $a(t)$ and $\delta_f(t)$ are the vehicle's acceleration and steering angle, and here, the vehicle's velocity $v(t)$ is a state rather than a control input, as it was in (1.1). The remaining states, however, are the same as before. Furthermore, l_f and l_r are the distances from the vehicle's center of mass to its front and rear axles, respectively, and β is the angle that the vehicle's direction of motion makes with its longitudinal axis (see, for example, [29] for an illustrative figure and an experimental evaluation for this particular model).

Given the aforementioned models, the goal for the control system is to accurately track the reference trajectory provided by the planner system. In our motion planning example, this means the control system must have the vehicle accurately follow the planned pathway through the environment. Let $\hat{x}(t)$ and $\hat{u}(t)$ be the state and input for the abstraction (corresponding to (1.1) in this example), and $x(t)$ and $u(t)$ be the state and input for the high-fidelity model (corresponding to (1.2) in this example). Then, in the general case, the control system can accomplish its task by applying a feedback control law $u(t) = k(x(t), \hat{x}(t), \hat{u}(t))$ which attempts to drive the system state $x(t)$ to a manifold

$$x(t) = \pi(\hat{x}(t), \hat{u}(t)) \tag{1.3}$$

where, in this example, the manifold we are interested in is given by $[p_x(t);\ p_y(t);\ \theta(t);\ v(t)] = [\hat{p}_x(t);\ \hat{p}_y(t);\ \hat{\theta}(t);\ \hat{v}(t)]$. In [59], we focused on a few special cases of this problem and provided a method for obtaining a bound on the error $e(t) := x(t) - \pi(\hat{x}(t), \hat{u}(t))$ using a *simulation function* $V(x(t), \hat{x}(t))$, where the bound depends on a residual term $r(\hat{u}(t), \hat{x}(t))$. The main benefit of our approach from [59] is that the abstraction is not required to satisfy certain restrictive geometric conditions - hence, we refer to abstractions of this type as *approximate abstractions*. In Chapter 2, we extend this approach to a class of interconnected control systems and then connect our results to an application in aggregation [11].

Chapter 3: Vehicle Platooning

In Chapter 3 we focus on improving traffic throughput at intersections by forming platoons of vehicles. As discussed in [32], we can define the *capacity* of an intersection as

$$\text{Capacity} = \sum_i s_i \cdot \frac{g_i}{T} \tag{1.4}$$

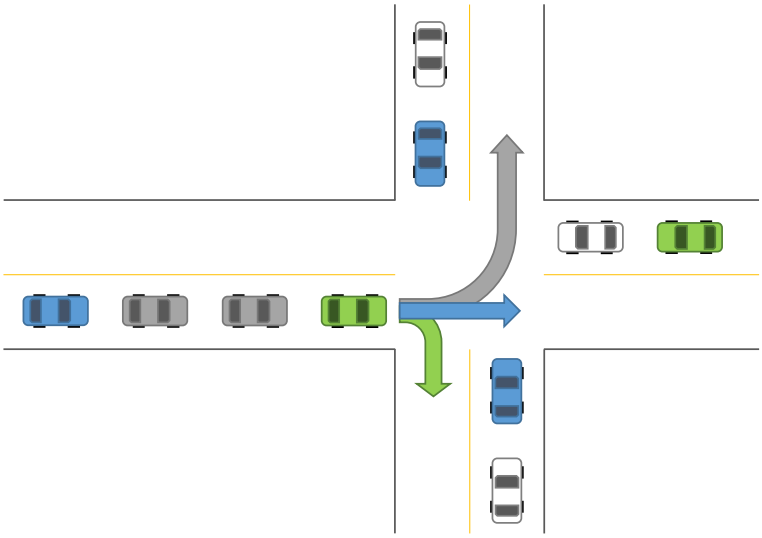


Figure 1.2. The three possible traffic movements (through, left turn, and right turn) are shown for one of the lanes approaching the intersection. While there are a total of twelve possible traffic movements at the intersection, only a fraction of these movements are allowable simultaneously. This means the intersection's capacity is reduced by a factor g_i/T as in (1.4).

where T is the traffic light cycle time, and for movement i, s_i is the saturation flow rate in vehicles per hour (vph) and g_i is the green time. Here, the saturation flow rate s_i is the throughput at the intersection (in vehicles per hour) that is achieved if an infinitely long string of vehicles were to travel through the intersection along one of its movements i, completely uninterrupted by changes in the traffic light signal. This scenario describes the steady state behavior of the system, and therefore s_i is a steady state flow rate. Since the traffic light signal cycles periodically between the red, yellow, and green phases, however, there is only a finite window of opportunity for the vehicles entering the intersection to achieve this saturation flow rate, meaning the intersection capacity is reduced by a factor of g_i/T as shown in (1.4). Furthermore, when the light cycles from red to green, there is also a short period of transient behavior when the vehicles are accelerating from a complete stop to their nominal velocity, meaning saturation flow is not achieved immediately. Although this latter effect has no impact on the capacity of the intersection as defined in (1.4), it further reduces the intersection efficiency in practice.

In order to improve the capacity of the intersection, [32] proposes increasing the saturation flow rate s_i in (1.4) by forming vehicle platoons at the intersection. Indeed, their simulation results show that doing so could potentially double traffic throughput in urban roadways. One benefit of this approach is that it does not require modifying the timing of the traffic light phases, but only the behavior of the vehicles themselves moving through the intersection. To further improve the efficiency of the intersection in practice, it is also important to consider the transient behavior mentioned earlier. That is, in the ideal case, a long platoon of vehicles

stopped at a red light will quickly and safely approach steady state behavior (where the saturation flow rate s_i is achieved) when the traffic light cycles from red to green, thereby minimizing the negative impact of the transient behavior on the intersection efficiency. To do so, the vehicles in the platoon need to be able to accurately maintain constant inter-vehicle distances while accelerating and decelerating, which is a challenging engineering task. More precisely, the platoon control system needs to ensure *string stability* of the group of vehicles (for more details on string stability, we refer the interested reader to [65]). The idea is to ensure that any disturbances that begin at the front of the platoon - for example, changes in velocity or acceleration of the platoon leader - are not amplified as they move toward the back of the platoon, as each subsequent vehicle in the group reacts to the disturbance. String stability has been shown to depend, in particular, on the flow of V2V messages within the platoon [23]. In our work, we ensure string stability by using an information flow topology which enables each follower vehicle in the platoon to do distance tracking (that is, maintaining a constant distance to a target object) of the leader vehicle, meaning that when the platoon leader accelerates, each follower vehicle in the platoon accelerates simultaneously in response. This is a significant improvement over human drivers, who typically respond to the vehicle directly in front of them with a nontrivial reaction time. For the reasons discussed earlier, this can cause instabilities in the flow of traffic and decrease the efficiency of intersections. Indeed, vehicle platooning offers a significant improvement over the way most humans drive.

In practice, forming platoons of vehicles at intersections is a challenging engineering problem, and not unlike many engineering problems, there are a few tradeoffs present. One tradeoff is between passenger comfort and traffic throughput. Consider when the light cycles from red to green, for example - the platoon could accelerate more quickly from a stop in order to pass through the intersection sooner and increase throughput, but this will come at the cost of passenger comfort. Another important tradeoff is between safety and traffic throughput, which was studied using simulation in our preliminary work [61]. Intuitively, by pushing the vehicles in the platoon closer together, we can achieve higher levels of traffic throughput at the intersection; however, doing so can introduce some risk to the follower vehicles in the platoon. Our results in [61] show that, in order to achieve a reasonable improvement in traffic throughput at the intersection, the vehicles in the platoon must essentially "trust" each other and drive as closely together as possible. More specifically, each platooning vehicle must broadcast a custom V2V message with a prediction on its own future velocity trajectory, and the other platooning vehicles must assume the prediction is accurate and use it in their respective MPC problems. Building off this work in [61], in Chapter 3 we provide an overview of a recent experimental project, in collaboration with the MPC lab at UC Berkeley, where we implemented a vehicle platooning system on Hyundai Ioniq test vehicles and evaluated it in an urban traffic setting.

Chapter 4: Safety in Real Driving Scenarios

While in Chapter 3 we focused on improving the efficiency of traffic flows by forming vehicle platoons at intersections, in Chapter 4 we turn our focus toward improving the *safety* of traffic flows. Specifically, we focus on efficiently designing vehicle controllers which can guarantee safety in two real driving scenarios, one of which typically occurs at an intersection. The first scenario we consider is a *vehicle-following scenario*. In this scenario, the longitudinal dynamics of the ego vehicle and a lead vehicle ahead of it are considered, and the goal is to design a controller for the ego vehicle such that a safety specification on the relative distance and velocity of the two vehicles is met. In particular, we consider a safety specification from [31] which relaxes the constraints on the ego vehicle, allowing the two vehicles to drive more closely together. This is crucial in vehicle platooning, for example, where the goal for the ego vehicle is to maintain a short distance to the lead vehicle at all times. The second scenario we consider is an *unprotected left turn scenario*. In this scenario, the ego vehicle is trying to make a left turn (typically at an intersection) in a situation where it does not have the right of way. Hence, the ego vehicle must yield to any oncoming vehicles that are passing through the intersection while it is attempting to execute its turn.

The second driving scenario we consider is particularly challenging due to the fact that intersections are complex and dangerous environments. Indeed, navigating through an intersection safely requires a driver to analyze and respond to the nuanced behavior of nearby vehicles, bicyclists, and pedestrians. For example, while executing an unprotected left turn, an AV must rely on its predictions of the velocity trajectories of any oncoming vehicles in order to consistently maintain a safe distance to those vehicles. In fact, Cruise, a self-driving startup owned by General Motors, has said that making an unprotected left turn at an intersection is one of the most difficult maneuvers a self-driving vehicle can perform [71]. Due to this, Cruise has done extensive testing in San Francisco focused on improving how well their AVs execute unprotected left turns, as well as other difficult driving maneuvers. On the other hand, other startups developing self-driving technology have faced setbacks while conducting tests at public intersections; in 2017, an Uber AV was involved in a collision at an intersection in Tempe, Arizona - see Figure 1.3 for a photo [46]. The crash occurred between the Uber automated Volvo, which was driving straight through the intersection, and a human-driven Honda CR-V, which was attempting to make a left turn. Unfortunately, heavy traffic occluded the Honda CR-V so that neither the Uber AV's sensors nor its human operator could detect the oncoming vehicle in time to avoid the collision. Since the Uber AV had a yellow light with enough time to make it through the intersection, it simply maintained its speed of 38 miles per hour immediately prior to the crash. No one was injured in the accident, and since the Uber AV technically had the right of way, it was not legally at fault for the collision. Nevertheless, this incident highlights the importance of improving the safety of intersections, and has since become a useful case study in the academic community [22]. In particular, the inability of the Uber AV's onboard sensors to detect the Honda CR-V early on were an interesting contributing factor to the accident. To address this limitation and avoid such an accident in the future, the authors of [22] suggest

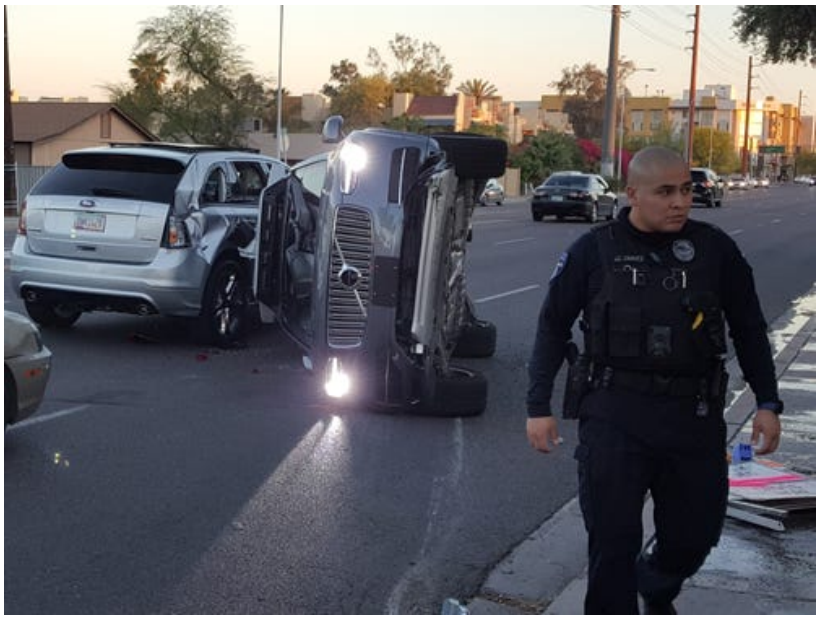


Figure 1.3. A photograph taken after the crash that occurred between an Uber automated Volvo and a Honda CR-V in Tempe, Arizona.

utilizing vehicle-to-infrastructure communication. For example, a SPaT message broadcast by the intersection could notify the approaching Uber AV that the light is about to turn yellow, encouraging it to slow down. In Chapter 4, however, we are most interested in the *conflict zone* analysis discussed in [22]. The main idea is to identify areas in the intersection where two driving movements overlap with one another, and then ensure that two vehicles never occupy the same conflict zone simultaneously. As an example, Figure 1.4 shows the conflict zone for a through movement and unprotected left turn movement at an intersection - we will return to this example in Chapter 4.

In order to design a safety controller for the ego vehicle in each driving scenario, we apply recently developed symbolic control techniques from [28] and [52]. The salient features of our approach are: 1) we use discrete abstractions of the system dynamics, and 2) we exploit monotonicity properties of the system dynamics to reduce computational complexity. Before diving into the details in Chapter 4, here we provide some intuition behind our approach for the unprotected left turn scenario. The system dynamics are monotone since we assume neither vehicle will reverse. Furthermore, there are two strategies for the ego vehicle to safely execute its turn - it can either wait and let the oncoming vehicle pass through the intersection (or, more specifically, the conflict zone) before beginning its turn, or it can go first and complete its turn before the oncoming vehicle enters the intersection. The formal control specification corresponding to each of these strategies is *directed* (see [28] for a definition). Indeed, if the ego vehicle wants to turn first, it is always better off if 1) the ego vehicle is farther along in its turn, 2) its velocity is larger, and 3) the oncoming vehicle is farther away. On the other hand, if the ego vehicle wants to let the oncoming vehicle go

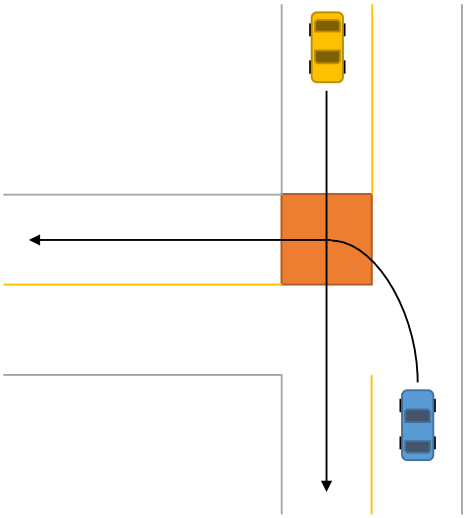


Figure 1.4. Conflict zone (shown in orange) between a through movement and a left turn movement at an intersection. To ensure safety, only one vehicle can occupy the conflict zone at a time.

first, the converse is true. This means we can employ an efficient algorithm from [52] to identify the set of safe states in this scenario and design a safety controller. The main idea is that we only need to test states on the boundary of a region to determine if the entire region is safe, due to the monotonicity properties of the system and specification. This saves us significant computational effort, as we will demonstrate.

Chapter 2

Approximate Abstractions of Control Systems

2.1 Introduction

The synthesis of controllers for dynamical systems enforcing complex logic properties, e.g. those expressed as linear or signal temporal logic (LTL/STL) formulas [8, 16], is hampered by computational challenges. One way of tackling the design complexity is by employing abstractions, which are simpler representations of original systems with the property that controllers designed for them to enforce desired properties can be refined to the ones for the concrete systems. The errors suffered in this controller synthesis detour can be quantified a priori. The abstraction is called finite if its set of states is finite, and infinite otherwise. In this chapter, we only deal with infinite abstractions.

Abstractions of non-stochastic dynamical systems has a long history. Examples of such results include constructive procedures for the construction of infinite abstractions of linear control systems using exact simulation relations [53]. In contrast to the exact notions, the results in [1] provide an approach for the construction of infinite abstractions of linear control systems using approximate simulation relations based on *simulation functions*. The construction schemes proposed in [53, 1] are monolithic in the sense that infinite abstractions are constructed from the complete system model. Compositional construction of approximate abstractions for the interconnection of two subsystems is studied in [19] using small gain type conditions. This result was extended in [50] to networks of systems, again with small gain type reasoning. The recent result in [74] employs broader dissipativity methods for constructing approximate abstractions for networks.

The infinite abstractions discussed here are also related to the rich theory of model order reduction, which seeks abstractions with reduced state-space dimensions [5, 26]. However, the model mismatch in [5, 26] is established with respect to $\mathcal{H}_2/\mathcal{H}_\infty$ norms whereas we use notions of simulation functions to derive $\mathcal{L}_\infty$ error bounds, which are crucial to reason about complex logic properties, e.g., LTL or STL formulas [8, 16].

The aforementioned results on the construction of exact or approximate infinite abstractions, [53, 1, 50, 74], require restrictive geometric conditions which, in some cases, are satisfied only when the state dimensions of the abstraction and the original system are the same (i.e., no order reduction).

In this work, we address this shortcoming as follows. We first show that, when constructing an abstraction monolithically, one can relax the geometric conditions appearing in [53, 1, 50, 74]. We quantify the effect of this relaxation via a nonnegative function which can be bounded in a formal synthesis of the abstract controller. To translate this bound into one on the error between the concrete system and its abstraction, we modify the definition of simulation functions from [1] to that of *practical simulation functions*, which include the nonnegative function in the upper bound on their derivative.

Next, we show that when constructing an abstraction in a compositional manner, one can also relax a restrictive condition on the interconnection topology from [50, 74]. We show that this relaxation greatly expands the domain of applicability of model order reduction via *aggregation*, where one creates an abstraction by partitioning agents into aggregate areas. In addition, our construction utilizes a modified version of storage functions from [74], which we refer to as *practical storage functions*. This notion allows us to accommodate heterogeneity in the agent models in aggregation.

The flexibility of our approach greatly broadens the applicability of infinite abstractions, including their usage in formal control synthesis procedures. Indeed, it was previously difficult and at times intractable to find an infinite abstraction satisfying the aforementioned geometric conditions. Thus, our method overcomes a significant limitation of abstraction-based controller design by allowing one to instead use an approximate abstraction which need not satisfy such conditions. The additional error introduced by this approach can then be quantified with our newly introduced notion of a practical simulation function.

The chapter is organized as follows. In Section 2.2, we introduce the class of control systems and corresponding abstractions studied in the chapter. We show in Section 2.3 how one can construct an abstraction in a monolithic manner for the class of linear systems. The discussion in Section 2.3 is based on the preliminary work in [59]; however, the content after Section 2.3 is entirely new. In Section 2.4, we consider a class of interconnected control systems, and present a result on the compositional construction of an abstraction for such systems. In Section 2.5, we show how our theory can aid in the procedure of aggregation, and include an example in building temperature regulation in Section 2.6. All proofs are given in the Appendix.

2.2 Control Systems

Notation.

We denote the set of real numbers as $\mathbb{R}$, and write the set of positive and nonnegative real numbers as $\mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$, respectively. For $a, b \in \mathbb{R}$ with $a \leq b$, we denote with (a, b) the

open interval from a to b. The n-dimensional Euclidean space is denoted with $\mathbb{R}^n$. We use $\mathbf{1}_n$ and $\mathbf{0}_n$ to denote the n-dimensional vector with all entries equal to 1 and 0, respectively. The vector space of matrices with n rows and m columns is represented by $\mathbb{R}^{n\times m}$. We use I_n to denote the identity matrix with n rows and columns. The concatenation of vectors $x_i \in \mathbb{R}^{n_i}$ for $i = 1, \ldots, N$ is given by $[x_1; x_2; \ldots; x_N] \in \mathbb{R}^n$, where $n = \sum_{i=1}^N n_i$. Similarly, the block-diagonal concatenation of matrices $P_i \in \mathbb{R}^{m_i \times n_i}$ for $i = 1, \ldots, N$ is written as $\mathrm{diag}(P_1, \ldots, P_N) \in \mathbb{R}^{m\times n}$, where m and n are defined in the same way. The null space of a matrix $P \in \mathbb{R}^{m\times n}$ is given by $\mathcal{N}(P) := \{x \in \mathbb{R}^n : Px = \mathbf{0}_m\}$. Furthermore, $\|P\|_F$ and $\mathbf{tr}(P)$ refer to the Frobenius norm and trace of P, respectively. The map $\|\cdot\| : \mathbb{R}^{n\times m} \to \mathbb{R}_{\geq 0}$ refers to the Euclidean norm when the argument is a vector, and the matrix norm induced by the Euclidean norm when the argument is a matrix. For a symmetric matrix $P \in \mathbb{R}^{n\times n}$, we use $\lambda_{\min}(P)$ and $\lambda_{\max}(P)$ to denote the minimum and maximum eigenvalue of P, respectively. We denote the Kronecker product of matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ as $A \otimes B \in \mathbb{R}^{mp\times nq}$.

A continuous function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$; furthermore, $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to class $\mathcal{K}_\infty$ if $\alpha \in \mathcal{K}$ and $\alpha(s) \to \infty$ as $s \to \infty$. A continuous function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to belong to class $\mathcal{KL}$ if, for each fixed t, the map $\beta(r, t)$ belongs to class $\mathcal{K}$ with respect to r and, for each fixed nonzero r, the map $\beta(r, t)$ is decreasing with respect to t and $\beta(r, t) \to 0$ as $t \to \infty$. Lastly, for a measurable function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, we use $\|f\|_\infty$ to indicate $\sup_{t\geq 0} \|f(t)\|$.

Control systems and their abstractions.

We first define the class of control systems studied in this chapter:

Definition 2.2.1. *A control system Σ is a tuple $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, f, \mathbb{R}^q, h)$, where $\mathbb{R}^n$, $\mathbb{R}^m$, and $\mathbb{R}^q$ are the state, input, and output spaces, respectively. The evolution of the state and output trajectories are governed by*

$$\Sigma : \begin{cases} \dot{\xi}(t) = f(\xi(t), \upsilon(t)), \\ \zeta(t) = h(\xi(t)), \end{cases}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is locally Lipschitz, and we refer to $h : \mathbb{R}^n \to \mathbb{R}^q$ as the output map.

We denote by $\xi_{x\upsilon}(t)$ the state reached at time t under the input $\upsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ from the initial condition $x = \xi_{x\upsilon}(0)$; the state $\xi_{x\upsilon}(t)$ is uniquely determined due to the assumptions on f [64]. We also denote by $\zeta_{x\upsilon}(t)$ the corresponding output value of $\xi_{x\upsilon}(t)$, i.e. $\zeta_{x\upsilon}(t) = h(\xi_{x\upsilon}(t))$.

When the dimension of the state space is large, one can avoid the computational burden of a direct controller synthesis for Σ by introducing an abstraction $\hat{\Sigma}$, potentially with a smaller state-space dimension $\hat{n}$. Typically, the abstraction $\hat{\Sigma}$ is related to the concrete system Σ via a *simulation function* [1], which enables one to bound the error between the outputs of the two systems. We now define a modified version of simulation functions, which we refer to as *practical* simulation functions:

Definition 2.2.2. *Consider a control system* $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, f, \mathbb{R}^q, h)$ *with corresponding abstraction* $\hat{\Sigma} = (\mathbb{R}^{\hat{n}}, \mathbb{R}^{\hat{m}}, \hat{f}, \mathbb{R}^q, \hat{h})$. *Let* $V : \mathbb{R}^n \times \mathbb{R}^{\hat{n}} \to \mathbb{R}_{\geq 0}$ *be a continuously differentiable function and* $v : \mathbb{R}^n \times \mathbb{R}^{\hat{n}} \times \mathbb{R}^{\hat{m}} \to \mathbb{R}^m$ *a locally Lipschitz function. We say that* V *is a practical simulation function from* $\hat{\Sigma}$ *to* Σ *with an associated interface* v *if there exist* $\nu, \eta \in \mathcal{K}_\infty$, $\rho \in \mathcal{K} \cup \{0\}$, *and* $\Delta : \mathbb{R}^{\hat{n}} \to \mathbb{R}_{\geq 0}$ *such that for all* x, $\hat{x}$, *and* $\hat{u}$ *we have*

$$\nu(\|h(x) - \hat{h}(\hat{x})\|) \leq V(x, \hat{x}) \tag{2.1}$$

and

$$\frac{\partial V(x,\hat{x})}{\partial x} f(x, v(x, \hat{x}, \hat{u})) + \frac{\partial V(x,\hat{x})}{\partial \hat{x}} \hat{f}(\hat{x}, \hat{u}) \leq -\eta(V(x, \hat{x})) + \rho(\|\hat{u}\|) + \Delta(\hat{x}). \tag{2.2}$$

Here, we modified the definition of simulation functions to include a nonnegative term $\Delta(\hat{x})$ in the upper bound of their derivatives. Thus, when $\Delta(\hat{x}) = 0$ we refer to $V(x, \hat{x})$ as a simulation function. We note that the associated interface $v(x, \hat{x}, \hat{u})$ helps to achieve (2.2) and, in particular, can be used to reduce the term $\Delta(\hat{x})$ as much as possible. The usefulness of $\Delta(\hat{x})$ will become apparent in Section 2.3, where we show that its addition allows one to relax the geometric conditions typically required in the construction of infinite abstractions. To further motivate the addition of the term $\Delta(\hat{x})$, we provide an example of a system and abstraction which admit a practical simulation function as in Definition 2.2:

Example 1. *Consider the control system*

$$\Sigma : \begin{cases} \begin{pmatrix} \dot{\xi}_1(t) \\ \dot{\xi}_2(t) \end{pmatrix} = \begin{pmatrix} -1.5\xi_1^3(t) + \upsilon(t) \\ -\xi_2^3(t) + \upsilon(t) \end{pmatrix}, \\ \zeta(t) = (\xi_1(t) + \xi_2(t))/2, \end{cases}$$

with $\xi_1(t), \xi_2(t), \zeta(t) \in \mathbb{R}$, *and where* $\xi_1(t)$ *and* $\xi_2(t)$ *are aggregated into a single state variable* $\hat{\xi}(t) \in \mathbb{R}$ *governed by*

$$\hat{\Sigma} : \begin{cases} \dot{\hat{\xi}}(t) = -1.5\hat{\xi}^3(t) + \hat{\upsilon}(t), \\ \hat{\zeta}(t) = \hat{\xi}(t), \end{cases}$$

with $\hat{\zeta}(t) \in \mathbb{R}$. *Then, by defining the associated interface*

$$v(x, \hat{x}, \hat{u}) = \hat{u},$$

we have that $V(x, \hat{x}) := (1/2)(x - \hat{x}\mathbf{1}_2)^T(x - \hat{x}\mathbf{1}_2)$ *is a practical simulation function from* $\hat{\Sigma}$ *to* Σ *since one can verify that*

$$((x_1 + x_2)/2 - \hat{x})^2 \leq V(x, \hat{x})$$

and

$$\begin{aligned}\dot{V}(x,\hat{x}) &= (x-\hat{x}\mathbf{1}_2)^T(\dot{x}-\dot{\hat{x}}\mathbf{1}_2)\\ &= \begin{bmatrix}x_1-\hat{x}\\ x_2-\hat{x}\end{bmatrix}^T\begin{bmatrix}-1.5x_1^3+\hat{u}-(-1.5\hat{x}^3+\hat{u})\\ -x_2^3+\hat{u}-(-1.5\hat{x}^3+\hat{u})\end{bmatrix}\\ &\leq -\frac{1}{8}V^2(x,\hat{x})+\frac{3}{8}\hat{x}^4\end{aligned}$$

hold. Thus, we have that (2.1) *and* (2.2) *from Definition 2.2 are satisfied with* $\nu(s) := s^2,\ \eta(s) := (1/8)s^2,\ \rho(s) = 0,$ *and* $\Delta(\hat{x}) := (3/8)\hat{x}^4$.

The next theorem shows the usefulness of a practical simulation function by providing a bound on the error between the output behaviors of control systems to those of their abstractions.

Theorem 1. *Consider a system* $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, f, \mathbb{R}^q, h)$ *with corresponding abstraction* $\hat{\Sigma} = (\mathbb{R}^{\hat{n}}, \mathbb{R}^{\hat{m}}, \hat{f}, \mathbb{R}^q, \hat{h})$, *and let* V *be a practical simulation function from* $\hat{\Sigma}$ *to* Σ. *Then, there exists a class* $\mathcal{KL}$ *function* β *and class* $\mathcal{K}$ *functions* γ_1, γ_2 *such that for any measurable* $\hat{\upsilon} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\hat{m}}$ *and* $x \in \mathbb{R}^n$, $\hat{x} \in \mathbb{R}^{\hat{n}}$, *there exists a measurable* $\upsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ *via the associated interface* υ *such that the following bound holds for all* $t \in \mathbb{R}_{\geq 0}$*:*

$$\|\zeta_{x\upsilon}(t) - \hat{\zeta}_{\hat{x}\hat{\upsilon}}(t)\| \leq \beta(V(x,\hat{x}),t) + \gamma_1(\|\hat{\upsilon}\|_\infty) + \gamma_2(\|\Delta(\hat{\xi}_{\hat{x}\hat{\upsilon}})\|_\infty).$$

The proof of Theorem 1 is similar to the one of Theorem 3.5 in [74] and is omitted here due to lack of space.

2.3 Abstraction Synthesis for Linear Systems

To demonstrate the relaxation of geometric constraints, here we adapt our approach to linear control systems

$$\Sigma : \begin{cases}\dot{\xi}(t) = A\xi(t) + B\upsilon(t),\\ \zeta(t) = C\xi(t),\end{cases} \tag{2.3}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{q\times n}$, and the pair (A, B) is stabilizable. Our goal is to represent (2.3) with an abstract control system

$$\hat{\Sigma} : \begin{cases}\dot{\hat{\xi}}(t) = \hat{A}\hat{\xi}(t) + \hat{B}\hat{\upsilon}(t),\\ \hat{\zeta}(t) = \hat{C}\hat{\xi}(t),\end{cases} \tag{2.4}$$

where $\hat{A} \in \mathbb{R}^{\hat{n}\times\hat{n}}$, $\hat{B} \in \mathbb{R}^{\hat{n}\times\hat{m}}$, and $\hat{C} \in \mathbb{R}^{q\times\hat{n}}$. It has been shown in [1, Theorem 2] that if one can find matrices $P \in \mathbb{R}^{n\times\hat{n}}$ and $Q \in \mathbb{R}^{m\times\hat{n}}$ such that $\hat{C} = CP$, and the condition

$$AP = P\hat{A} - BQ \tag{2.5}$$

holds, then there exists a practical simulation function from $\hat{\Sigma}$ to Σ with an associated interface given by

$$v(x, \hat{x}, \hat{u}) = K(x - P\hat{x}) + Q\hat{x} + R\hat{u} \tag{2.6}$$

where the matrix $K \in \mathbb{R}^{m \times n}$ in (2.6) is a feedback gain to be designed and $R \in \mathbb{R}^{m \times \hat{m}}$ is selected to minimize $\|BR - P\hat{B}\|$. As alluded to previously, the requirement (2.5) can be restrictive in general. Indeed, the following lemma, quoted from [1, Lemma 2], provides the geometric conditions on P such that (2.5) is satisfiable:

Lemma 1. *For given matrices A, P, and B, there exist matrices $\hat{A}$ and Q satisfying* (2.5) *if and only if*

$$\mathrm{Im}(AP) \subseteq \mathrm{Im}(P) + \mathrm{Im}(B).$$

To address the restriction implicit in (2.5), we propose a relaxation by allowing a nonzero residual term given by

$$D := AP - P\hat{A} + BQ.$$

The effect of a nonzero matrix D is seen by examining the dynamics of the error $e(t) := \xi(t) - P\hat{\xi}(t)$, which become

$$\dot{e}(t) = (A + BK)e(t) + D\hat{\xi}(t) + (BR - P\hat{B})\hat{v}(t) \tag{2.7}$$

where

$$D\hat{\xi}(t) + (BR - P\hat{B})\hat{v}(t) \tag{2.8}$$

is treated as a disturbance. Thus, by relaxing (2.5), we have introduced a new term depending on $\hat{\xi}$ into the disturbance (2.8), which previously only depended on $\hat{v}$.

We next design the feedback gain K to mitigate the effect of this disturbance. To this end we rewrite (2.7) as

$$\dot{e}(t) = (A + BK)e(t) + Wd(t) \tag{2.9}$$

where we have defined

$$W := \begin{bmatrix} I & BR - P\hat{B} \end{bmatrix}, \quad d := \begin{bmatrix} D\hat{\xi} \\ \hat{v} \end{bmatrix}, \tag{2.10}$$

where I is the identity matrix of appropriate size. The magnitude of d can be bounded by placing constraints on $D\hat{\xi}$ and $\hat{v}$, to be respected for all $t \geq 0$. This can be done by introducing an appropriate STL specification for $\hat{\Sigma}$ which constrains $D\hat{\xi}$ and $\hat{v}$, and then synthesizing a control law $\hat{v}$ such that the resulting trajectories of $\hat{\Sigma}$ satisfy said specification - known as a formal synthesis procedure. In this chapter, we apply a formal synthesis procedure utilizing *model predictive control* (MPC) [47]; MPC is well known for being able to handle such constraints. Note that we do not need to constrain $\hat{\xi}$ itself to be small, but rather the value of $D\hat{\xi}$. For example, in a motion coordination application in [59], $D\hat{\xi}$ yields relative

positions and the constraints do not unreasonably restrict the absolute positions contained in the vector $\hat{\xi}$.

We remark that using MPC to design $\hat{v}$ requires discretization of the dynamics (2.4). This is important to note, in particular, since this implies the constraints on d in (2.10) will only hold at each sampling instant. Thus, we must establish a growth bound on each component of d in order to characterize its inter-sample behavior. For $\hat{\xi}$, one can impose constraints such that $\hat{A}\hat{\xi}$ and $\hat{B}\hat{v}$ are bounded, and then subsequently bound $\dot{\hat{\xi}}$ from (2.4). Furthermore, since $\hat{v}$ is a zero-order hold signal its derivative between samples is zero. Combining these facts to provide a bound on d, we ensure the quality of the abstraction $\hat{\Sigma}$.

After designing $\hat{v}$, our goal becomes to design K to minimize the $\mathcal{L}_\infty$ gain from d to error e. Since (2.9) is linear, an estimate for this gain is obtained by finding a bound $\overline{e} := \|e\|_\infty$ when $\overline{d} := \|d\|_\infty \leq 1$. We pursue this by numerically searching for $U = U^T > 0$ such that the ellipsoid $\mathcal{E} = \{e : e^T U e \leq 1\}$ is invariant. This results in $\overline{e} = 1/\sqrt{\lambda_{\min}(U)}$, since this is the radius of the smallest ball enclosing $\mathcal{E}$. The following optimization problem combines the search for U with a simultaneous search for a K that minimizes $\overline{e}$. Its derivation is similar to Section 6.1.3 of [10] and is omitted here due to lack of space.

Optimization Problem 1:

$$\begin{aligned} \text{minimize} \quad & \beta \quad \text{over } Z := U^{-1},\ Y := KZ, \\ \text{subject to} \quad & Z \leq \beta I, && (2.11) \\ & X(Z, Y, \alpha) \leq 0, && (2.12) \end{aligned}$$

where

$$X(Z, Y, \alpha) := \begin{bmatrix} AZ + ZA^T + Y^T B^T + BY + \alpha Z & W \\ W^T & -\alpha I \end{bmatrix},$$

which is an LMI in Z and Y if the scalar $\alpha > 0$ is fixed. In particular, by minimizing β and imposing (2.11), we are effectively maximizing $\lambda_{\min}(U)$. Here, this is equivalent to minimizing the error bound since $\overline{e} = 1/\sqrt{\lambda_{\min}(U)}$. The next theorem states that a solution to Optimization Problem 1 yields a practical simulation function from $\hat{\Sigma}$ to Σ.

Theorem 2. *Suppose that U and K are a solution to Optimization Problem 1, and $\hat{C}$ in (2.4) satisfies $\hat{C} = CP$. Then $V(x, \hat{x}) := (x - P\hat{x})^T U (x - P\hat{x})$ is a practical simulation function from $\hat{\Sigma}$ to Σ with an associated interface $v(x, \hat{x}, \hat{u})$ as in (2.6).*

As mentioned in Theorem 1, the practical simulation function $V(x, \hat{x})$ bounds the error between the outputs of Σ and $\hat{\Sigma}$. This allows us to translate guarantees on $\hat{\Sigma}$ to weakened guarantees on Σ. For example, if one designs a controller enforcing a set $\hat{\Omega}$ to be invariant for $\hat{\Sigma}$, then the refined controller makes $\Omega^{\overline{e}}$ invariant for Σ, where in this case $\Omega^{\overline{e}} := \{e + P\hat{x} : \|e\|_\infty \leq \overline{e},\ \hat{x} \in \hat{\Omega}\}$. The question then becomes how to obtain a small bound on e so that the desired behavior is realized on Σ. A rigorous procedure for doing so is not the main focus

of this chapter, but is explored in [72]. Here, we simply focus on improving the error bound via the two steps outlined in this section: first, by designing $\hat{v}$ to restrict d, and second, by using the interface $v(x, \hat{x}, \hat{u})$ to reduce the gain from d to e. Our procedure is oriented towards control synthesis, as our goal is to move from designing an abstract controller towards designing a concrete one. In verification, where one wants to verify behavior correctness via abstraction, these steps cannot be applied in the reverse direction to reduce error, which could result in poor abstraction quality. Thus, we remark that our approach cannot be extended to verification in a straightforward way.

2.4 Compositionality

Interconnected control systems

In this section we propose an approach to construct an abstraction and corresponding practical simulation function for a class of interconnected control systems. In particular, we show how to do so by composing the abstractions of the subsystems. We start by defining the class of subsystems that we consider:

Definition 2.4.1. *A control subsystem Σ is a tuple $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, \mathbb{R}^p, f, \mathbb{R}^{q_1}, \mathbb{R}^{q_2}, h_1, h_2)$, where $\mathbb{R}^n$, $\mathbb{R}^m$, $\mathbb{R}^p$, $\mathbb{R}^{q_1}$, and $\mathbb{R}^{q_2}$ are the state, external input, internal input, external output, and internal output spaces, respectively. The evolution of the state and output trajectories are governed by the equations*

$$\Sigma : \begin{cases} \dot{\xi}(t) = f(\xi(t), \upsilon(t), \omega(t)), \\ \zeta_1(t) = h_1(\xi(t)), \\ \zeta_2(t) = h_2(\xi(t)), \end{cases}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ and $h_2 : \mathbb{R}^n \to \mathbb{R}^{q_2}$ are locally Lipschitz. We refer to $h_1 : \mathbb{R}^n \to \mathbb{R}^{q_1}$ and $h_2 : \mathbb{R}^n \to \mathbb{R}^{q_2}$ as the external and internal output maps, respectively.

Similar to a practical simulation function, a *storage function* [74] can be used to relate a control subsystem Σ to its abstraction $\hat{\Sigma}$ by describing a dissipativity property of the error dynamics.

Definition 2.4.2. *Consider a control system $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, \mathbb{R}^p, f, \mathbb{R}^{q_1}, \mathbb{R}^{q_2}, h_1, h_2)$ and corresponding abstraction $\hat{\Sigma} = (\mathbb{R}^{\hat{n}}, \mathbb{R}^{\hat{m}}, \mathbb{R}^{\hat{p}}, \hat{f}, \mathbb{R}^{q_1}, \mathbb{R}^{\hat{q}_2}, \hat{h}_1, \hat{h}_2)$. Let $V : \mathbb{R}^n \times \mathbb{R}^{\hat{n}} \to \mathbb{R}_{\geq 0}$ be a continuously differentiable function and $v : \mathbb{R}^n \times \mathbb{R}^{\hat{n}} \times \mathbb{R}^{\hat{m}} \to \mathbb{R}^m$ a locally Lipschitz function. We say that V is a practical storage function from $\hat{\Sigma}$ to Σ if there exist $\nu, \eta \in \mathcal{K}_\infty$, $\rho \in \mathcal{K} \cup \{0\}$, a function $\Delta : \mathbb{R}^{\hat{n}} \to \mathbb{R}_{\geq 0}$, matrices W, $\hat{W}$, H of appropriate dimensions, and matrix $X = X^T$ of appropriate dimension with conformal block partitions X^{11}, X^{12}, X^{21}, and X^{22}, such that for any $x \in \mathbb{R}^n$, $\hat{x} \in \mathbb{R}^{\hat{n}}$, $\hat{u} \in \mathbb{R}^{\hat{m}}$, $\hat{w} \in \mathbb{R}^{\hat{p}}$, and $w \in \mathbb{R}^p$ we have*

$$\nu(\|h_1(x) - \hat{h}_1(\hat{x})\|) \leq V(x, \hat{x})$$

and

$$\frac{\partial V(x,\hat{x})}{\partial x} f(x, v(x,\hat{x},\hat{u}), w) + \frac{\partial V(x,\hat{x})}{\partial \hat{x}} \hat{f}(\hat{x},\hat{u},\hat{w})$$
$$\leq -\eta(V(x,\hat{x})) + \rho(\|\hat{u}\|) + \Delta(\hat{x}) + \begin{bmatrix} Ww - \hat{W}\hat{w} \\ h_2(x) - H\hat{h}_2(\hat{x}) \end{bmatrix}^T \begin{bmatrix} X^{11} & X^{12} \\ X^{21} & X^{22} \end{bmatrix} \begin{bmatrix} Ww - \hat{W}\hat{w} \\ h_2(x) - H\hat{h}_2(\hat{x}) \end{bmatrix}.$$

Here, we relaxed the definition of storage functions given in [74] to *practical* storage functions by allowing the upper bound on their derivative to include a nonnegative function $\Delta(\hat{x})$. The term $v(x,\hat{x},\hat{u})$ acts as the associated interface in Definition 2.4.2 by providing the concrete control input u. We note that the purpose of matrix H is to allow comparison between $h_2(x)$ and $\hat{h}_2(\hat{x})$, which can have different output dimensions. Similarly, matrices W and $\hat{W}$ allow comparison between w and $\hat{w}$. The choice of matrices X^{11}, X^{12}, X^{21}, and X^{22} specify the type of dissipativity property being described [7].

Next, we define the class of interconnected control systems that we consider in this chapter:

Definition 2.4.3. *Consider N control subsystems $\Sigma_i = (\mathbb{R}^{n_i}, \mathbb{R}^{m_i}, \mathbb{R}^{p_i}, f_i, \mathbb{R}^{q_{1i}}, \mathbb{R}^{q_{2i}}, h_{1i}, h_{2i})$, $i = 1, \ldots, N$, and a static matrix M of appropriate dimension describing the coupling of these subsystems. The interconnected control system $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, f, \mathbb{R}^q, h)$, denoted as $\mathcal{I}(\Sigma_1, \ldots, \Sigma_N)$, is given by $n = \sum_{i=1}^N n_i$, $m = \sum_{i=1}^N m_i$, $q = \sum_{i=1}^N q_{1i}$, and*

$$f(x,u) := [f_1(x_1,u_1,w_1); \ldots; f_N(x_N,u_N,w_N)],$$
$$h(x) := [h_{11}(x_1); \ldots; h_{1N}(x_N)],$$

where $u = [u_1; \ldots; u_N] \in \mathbb{R}^n$, $x = [x_1; \ldots; x_N] \in \mathbb{R}^m$, and with the internal variables constrained by

$$[w_1; \ldots; w_N] = M[h_{21}(x_1); \ldots; h_{2N}(x_N)]. \tag{2.13}$$

A depiction of an interconnected control system $\mathcal{I}(\Sigma_1, \ldots, \Sigma_N)$ is given in Figure 2.1.

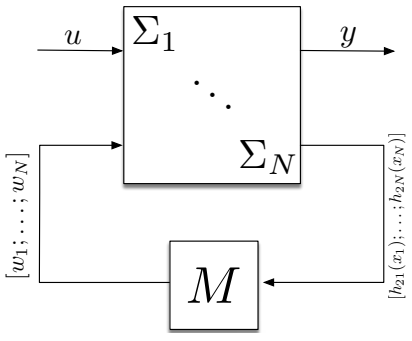


Figure 2.1. An interconnection of N control subsystems $\Sigma_1, \ldots, \Sigma_N$.

Compositionality result

We now provide a theorem containing our main result on the compositional construction of an abstraction and corresponding practical simulation function. In Definition 2.4.2 we included a nonnegative term $\Delta(\hat{x})$, allowing one to construct abstractions at the subsystem level by utilizing a relaxation similar to what was done in Section 2.3. Our next result is to show that a similar relaxation can also be made at the level of the interconnected control system. We first review a theorem from [74] that constructs simulation functions from storage functions associated to subsystems; we then present a modified version with relaxed conditions.

Theorem 3. *[74, Theorem 4.2] Consider the interconnected control system $\mathcal{I}(\Sigma_1, \ldots, \Sigma_N)$ induced by N control subsystems Σ_i and the coupling matrix M. Suppose each subsystem Σ_i admits an abstraction $\hat{\Sigma}_i$ and corresponding storage function V_i, each with the associated functions and matrices ν_i, η_i, ρ_i, v_i, H_i, W_i, $\hat{W}_i$, X_i, X_i^{11}, X_i^{12}, X_i^{21}, and X_i^{22} appearing in Definition 2.4.2 (by dropping term $\Delta(\hat{x})$). If there exist scalars $\mu_i > 0$, $i = 1, \ldots, N$, and matrix $\hat{M}$ of appropriate dimension such that the following matrix (in)equality constraints*

$$\begin{bmatrix} WM \\ I_{\tilde{q}} \end{bmatrix}^T X(\mu_1 X_1, \ldots, \mu_N X_N) \begin{bmatrix} WM \\ I_{\tilde{q}} \end{bmatrix} \leq 0, \tag{2.14}$$

$$WMH = \hat{W}\hat{M}, \tag{2.15}$$

are satisfied, where $\tilde{q} = \sum_{i=1}^{N} q_{2i}$ and

$$W := diag(W_1, \ldots, W_N),\ \hat{W} := diag(\hat{W}_1, \ldots, \hat{W}_N),\ H := diag(H_1, \ldots, H_N), \tag{2.16}$$

$$X(\mu_1 X_1, \ldots, \mu_N X_N) := \begin{bmatrix} \mu_1 X_1^{11} & & & \mu_1 X_1^{12} & & \\ & \ddots & & & \ddots & \\ & & \mu_N X_N^{11} & & & \mu_N X_N^{12} \\ \mu_1 X_1^{21} & & & \mu_1 X_1^{22} & & \\ & \ddots & & & \ddots & \\ & & \mu_N X_N^{21} & & & \mu_N X_N^{22} \end{bmatrix}, \tag{2.17}$$

then

$$V(x, \hat{x}) := \sum_{i=1}^{N} \mu_i V_i(x_i, \hat{x}_i) \tag{2.18}$$

is a simulation function from the interconnected control system $\hat{\Sigma} = \mathcal{I}(\hat{\Sigma}_1, \ldots, \hat{\Sigma}_N)$, with the coupling matrix $\hat{M}$, to Σ.

The following theorem relaxes (2.15) in Theorem 3 as follows:

Theorem 4. *Suppose, instead of (2.15), one can only find a matrix $\hat{M}$ yielding a residual*

$$Y := WMH - \hat{W}\hat{M} \tag{2.19}$$

which is nonzero, and all other hypotheses of Theorem 3 hold with each V_i being a practical storage function as in Definition 2.4.2. Then (2.18) *is a practical simulation function from $\hat{\Sigma}$ to Σ if there exist $\mu_i > 0$ and matrix $Z = Z^T \geq 0$ of appropriate dimensions such that the following matrix inequality constraint holds*

$$Q(Z, \mu_1, \ldots, \mu_N) := \begin{bmatrix} Y & WM \\ 0 & I_{\tilde{q}} \end{bmatrix}^T X(\mu_1 X_1, \ldots, \mu_N X_N) \begin{bmatrix} Y & WM \\ 0 & I_{\tilde{q}} \end{bmatrix} - \begin{bmatrix} Z & 0 \\ 0 & 0 \end{bmatrix} \leq 0. \quad (2.20)$$

In particular, the function $\Delta(\hat{x})$ in Definition 2.2.2 is given by

$$\Delta(\hat{x}) := \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix}^T Z \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix} + \sum_{i=1}^{N} \mu_i \Delta_i(\hat{x}_i). \quad (2.21)$$

Theorem 4 dropped the constraint (2.15) from Theorem 3, resulting in a residual term (2.19). The effect of this relaxation is then quantified via the term $\Delta(\hat{x})$, which is parameterized by the matrix Z and scalars μ_i in (2.21). Therefore, Theorem 4 is beneficial when no matrix $\hat{M}$ satisfying (2.15) exists. For such a scenario, we provide two optimization problems that can be solved in sequence to minimize the resulting $\Delta(\hat{x})$. First, with matrices W, M, H, and $\hat{W}$ fixed, we select the matrix $\hat{M}$ to minimize the residual (2.19) as measured by the Frobenius norm:

Optimization Problem 2:

$$\text{minimize } \|WMH - \hat{W}\hat{M}\|_F \quad \text{over } \hat{M}.$$

With $\hat{M}$ thus selected, our next goal is to find a minimal $\Delta(\hat{x})$ as defined in (2.21). We first introduce a diagonal scaling matrix S that induces the functions $\tilde{h}_{2i}$, $i = 1, \ldots, N$, as follows

$$\begin{bmatrix} \tilde{h}_{21}(\hat{x}_1) \\ \vdots \\ \tilde{h}_{2N}(\hat{x}_N) \end{bmatrix} := \underbrace{\begin{bmatrix} s_1 I_{\hat{q}_{21}} & & \\ & \ddots & \\ & & s_N I_{\hat{q}_{2N}} \end{bmatrix}}_{:=S} \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix}.$$

In particular, the scalars $s_i > 0$ are to be chosen so the outputs of the functions $\tilde{h}_{2i}(\hat{x}_i)$, $i = 1, \ldots, N$, are comparable in order of magnitude. Next, we define the scalars $r_i > 0$, $i = 1, \ldots, N$, which scale the functions $\Delta_i(\hat{x}_i)$ in the same way. Then, we propose finding a minimal $\Delta(\hat{x})$ by solving the following optimization problem.

Optimization Problem 3:

$$\begin{aligned} \text{minimize} \quad & \mathbf{tr}(S^{-T} Z S^{-1}) + \sum_{i=1}^{N} \frac{\mu_i}{r_i} \quad \text{over } Z \geq 0,\ \mu_i \geq 1,\ i = 1, \ldots, N, \\ \text{subject to} \quad & Q(Z, \mu_1, \ldots, \mu_N) \leq 0. \end{aligned} \quad (2.22)$$

In particular, here the objective function represents our goal of minimizing $\Delta(\hat{x})$ in (2.21), thus minimizing the error bound obtained via Theorem 1. Here, we constrain $\mu_i \geq 1$ so that the decision variables μ_i and Z do not become too small and, as a result, poorly scaled. We note that Optimization Problems 2 and 3 are both conic, and thus can be solved with a conic optimization tool such as MOSEK [6].

2.5 Aggregation

A common approach to model order reduction in large scale systems is aggregation, which combines physical variables into a small number of groups and studies the interaction among these groups. Examples include power systems, where geographical areas in which generators swing in synchrony are aggregated into *equivalent machines* [11], and multicellular ensembles, where groups of cells exhibiting homogeneous behavior are represented with lumped biochemical reaction models [17].

In this section we study a network of agents and first review an *equitable partition* criterion for aggregation when the agents have identical models. We next relax the identical model assumption and the equitability criterion by using the results of the previous sections. We formulate an optimization problem that penalizes the violation of the equitability condition when partitioning the agents into aggregate groups and, finally, study a special class of systems that encompasses the temperature control example in the next section.

Equitable partition criterion for aggregation

Consider L agents with identical dynamical models:

$$\dot{\xi}^\ell(t) = g(\xi^\ell(t), \upsilon^\ell(t), \omega^\ell(t)) \tag{2.23}$$

$$\zeta_1^\ell(t) = \varsigma(\xi^\ell(t)) \tag{2.24}$$

$$\zeta_2^\ell(t) = \sigma(\xi^\ell(t)) \qquad \ell = 1, 2, \cdots, L, \tag{2.25}$$

$\xi^\ell(t) \in \mathbb{R}^n$, $\upsilon^\ell(t) \in \mathbb{R}^m$, $\omega^\ell(t) \in \mathbb{R}^p$, $\zeta_1^\ell(t) \in \mathbb{R}^q$, $\zeta_2^\ell(t) \in \mathbb{R}^p$, for any $t \geq 0$, interconnected according to the relation

$$\begin{bmatrix} \omega^1(t) \\ \vdots \\ \omega^L(t) \end{bmatrix} = (\tilde{M} \otimes I_p) \begin{bmatrix} \zeta_2^1(t) \\ \vdots \\ \zeta_2^L(t) \end{bmatrix}, \quad \tilde{M} \in \mathbb{R}^{L \times L}. \tag{2.26}$$

We partition the agents $\{1, \ldots, L\}$ into $N \leq L$ groups and describe the assignment of the agents to the groups with the $L \times N$ partition matrix

$$P_{\ell,i} = \begin{cases} 1 & \text{if } \ell \in \text{ group } i \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

In particular, each agent is assigned to exactly one group, and each group must have at least one agent assigned to it. We then aggregate the agents comprising each group into a single agent model that describes homogeneous behavior within the group. Thus, the abstraction for group i is

$$\dot{\hat{\xi}}_i(t) = g(\hat{\xi}_i(t), \hat{\upsilon}_i(t), \hat{\omega}_i(t)) \tag{2.28}$$

$$\hat{\zeta}_{1i}(t) = \mathbf{1}_{L_i} \otimes \varsigma(\hat{\xi}_i(t)) \tag{2.29}$$

$$\hat{\zeta}_{2i}(t) = \sigma(\hat{\xi}_i(t)) \qquad i = 1, 2, \cdots, N, \tag{2.30}$$

where L_i is the number of agents in group i, $\hat{\xi}_i(t) \in \mathbb{R}^n$, $\hat{\upsilon}_i(t) \in \mathbb{R}^m$, $\hat{\omega}_i(t) \in \mathbb{R}^p$, $\hat{\zeta}_{1i}(t) \in \mathbb{R}^{qL_i}$, $\hat{\zeta}_{2i}(t) \in \mathbb{R}^p$, for any $t \geq 0$, and the interconnection relation is

$$\begin{bmatrix} \hat{\omega}_1(t) \\ \vdots \\ \hat{\omega}_N(t) \end{bmatrix} = (\bar{M} \otimes I_p) \begin{bmatrix} \hat{\zeta}_{21}(t) \\ \vdots \\ \hat{\zeta}_{2N}(t) \end{bmatrix}, \tag{2.31}$$

where $\bar{M} \in \mathbb{R}^{N \times N}$ is to be selected.

For the groups to exhibit perfectly homogeneous behavior, the trajectories must converge to and remain on the subspace where $\xi^\ell = \hat{\xi}_i$ for each ℓ in group i, $i = 1, \ldots, N$. The invariance of this subspace is ensured if $\upsilon^\ell = \hat{\upsilon}_i$ and $\omega^\ell = \hat{\omega}_i$ on the subspace, because $\xi^\ell(0) = \hat{\xi}_i(0)$, $\upsilon^\ell = \hat{\upsilon}_i$ and $\omega^\ell = \hat{\omega}_i$ imply $\dot{\xi}^\ell = \dot{\hat{\xi}}_i$ by (2.23) and (2.28). The internal inputs ω^ℓ, however, are not independent variables and the condition that $\omega^\ell = \hat{\omega}_i$ for having $\xi^\ell = \hat{\xi}_i$ for each ℓ in group i must be further examined. To do so, first note from (2.25) and (2.30) that $\xi^\ell = \hat{\xi}_i$ implies $\zeta_2^\ell = \hat{\zeta}_{2i}$, which means

$$\begin{bmatrix} \zeta_2^1(t) \\ \vdots \\ \zeta_2^L(t) \end{bmatrix} = (P \otimes I_p) \begin{bmatrix} \hat{\zeta}_{21}(t) \\ \vdots \\ \hat{\zeta}_{2N}(t) \end{bmatrix}$$

and, from (2.26),

$$\begin{bmatrix} \omega^1(t) \\ \vdots \\ \omega^L(t) \end{bmatrix} = (\tilde{M}P \otimes I_p) \begin{bmatrix} \hat{\zeta}_{21}(t) \\ \vdots \\ \hat{\zeta}_{2N}(t) \end{bmatrix}. \tag{2.32}$$

The desired condition is $\omega^\ell(t) = \hat{\omega}_i(t)$ for each ℓ in group i, that is

$$\begin{bmatrix} \omega^1(t) \\ \vdots \\ \omega^L(t) \end{bmatrix} = (P \otimes I_p) \begin{bmatrix} \hat{\omega}_1(t) \\ \vdots \\ \hat{\omega}_N(t) \end{bmatrix},$$

which is consistent with (2.32) if and only if $\bar{M}$ in (2.31) satisfies

$$\tilde{M}P = P\bar{M}. \tag{2.33}$$

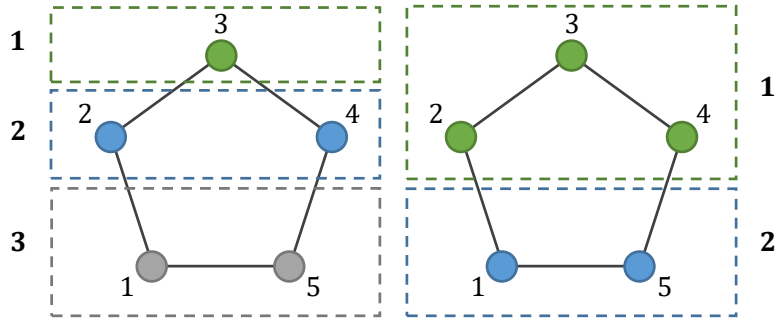


Figure 2.2. An equitable partition of a circle graph with $L = 5$ nodes into three groups (left). Note that the partition into two groups (right) is *not* equitable.

Thus, the invariance of the subspace $\xi^\ell = \hat{\xi}_i$ for each ℓ in group i hinges upon the property (2.33), formalized in the following definition:

Definition 2.5.1. *Given L agents with interconnection matrix $\tilde{M} \in \mathbb{R}^{L\times L}$, a partition into N groups is said to be equitable if the partition matrix P in (2.27) satisfies (2.33) for some $\bar{M} \in \mathbb{R}^{N\times N}$.*

To provide intuition behind equitability, suppose $\tilde{M}$ corresponds to the Laplacian matrix of an unweighted, undirected graph, where each node represents an agent and edges are drawn between agents which are connected to one another. In this case, a partition of the graph is equitable if each node in group k has exactly $c_{k\ell}$ neighbors in group ℓ, regardless of which node in class k we select [21]. Here, the constant $c_{k\ell}$ depends on k and ℓ. As an illustration, an equitable partition of a five-node circle graph is displayed in Figure 2.2 (left), where group 1 consists of node 3, group 2 consists of nodes $\{2, 4\}$, and group 3 consists of nodes $\{1, 5\}$. Each node in group 2 is connected to $c_{21} = 1$ node in group 1 and $c_{23} = 1$ node in group 3. On the other hand, note that the partition displayed in Figure 2.2 (right) into groups $\{2, 3, 4\}$ and $\{1, 5\}$ is *not* equitable. Although we discussed unweighted graphs for simplicity, the extension to weighted graphs is straightforward by considering the sum of the edge weights connected to a particular node instead of the number of neighbors.

Relaxing the identical agent and equitable partition assumptions

The assumptions that the agent dynamics be identical and that an equitable partition exist for their interconnection can be restrictive in practice. The control specifications may further limit the choice of partition, since the states of agents in the same group are lumped together in the abstraction and the specifications cannot distinguish between them.

Here we relax both assumptions using the results of Section 2.4. First we replace the agent dynamics (2.23) with

$$\dot{\xi}^\ell(t) = g^\ell(\xi^\ell(t), v^\ell(t), \omega^\ell(t)), \tag{2.34}$$

where $g^\ell : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$, $\ell = 1, \dots, L$, allow for deviations from the nominal model g used in the abstraction (2.28). We note that it is also possible to define separate nominal dynamics for each group in order to minimize said deviations further. For simplicity, here we use the same nominal model for each group. In preparation for constructing a simulation function, we assume that there exist practical storage functions from the agents to the nominal model with identical supply rates as the following:

Assumption 1. *There exist a locally Lipschitz function $\tilde{v}^\ell : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$, a continuously differentiable function $\tilde{V}^\ell : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, $\tilde{\nu}^\ell, \tilde{\eta}^\ell \in \mathcal{K}_\infty$, $\tilde{\rho}^\ell \in \mathcal{K} \cup \{0\}$, $\tilde{\Delta}^\ell : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, and a matrix $\tilde{X} = \tilde{X}^T \in \mathbb{R}^{2p \times 2p}$ such that for all $x \in \mathbb{R}^n$, $\hat{x} \in \mathbb{R}^n$, $\hat{u} \in \mathbb{R}^m$, $w \in \mathbb{R}^p$, $\hat{w} \in \mathbb{R}^p$,*

$$\tilde{\nu}^\ell(\|\varsigma(x) - \varsigma(\hat{x})\|) \leq \tilde{V}^\ell(x, \hat{x}), \tag{2.35}$$

$$\frac{\partial \tilde{V}^\ell(x,\hat{x})}{\partial x} g^\ell(x, \tilde{v}^\ell(x, \hat{x}, \hat{u}), w) + \frac{\partial \tilde{V}^\ell(x,\hat{x})}{\partial \hat{x}} g(\hat{x}, \hat{u}, \hat{w}) \tag{2.36}$$

$$\leq -\tilde{\eta}^\ell(\tilde{V}^\ell(x, \hat{x})) + \tilde{\rho}^\ell(\|\hat{u}\|) + \tilde{\Delta}^\ell(\hat{x}) + \begin{bmatrix} w - \hat{w} \\ \sigma(x) - \sigma(\hat{x}) \end{bmatrix}^T \tilde{X} \begin{bmatrix} w - \hat{w} \\ \sigma(x) - \sigma(\hat{x}) \end{bmatrix}. \tag{2.37}$$

In the next subsection we show a class of systems that satisfy Assumption 1. One will see, in particular, that the term $\tilde{\Delta}^\ell(\hat{x})$ in (2.37) is critical for absorbing the mismatch between g^ℓ and g, which is due to the heterogeneity of the agent models. In the example in Section 6, we show how the interface function can be used to help to shrink $\tilde{\Delta}^\ell(\hat{x})$ and satisfy Assumption 1 with a tight upper bound in (2.36).

We let each group $i = 1, \dots, N$ in the partition define a subsystem, and derive a composite storage function and dissipation inequality from Assumption 1. Let $L_i \geq 1$ denote the number of agents in group i, $L_1 + \cdots + L_N = L$, and define the state vector $x_i \in \mathbb{R}^{L_i n}$ by concatenating the state vectors x^ℓ of the agents assigned to group i. Defining $u_i \in \mathbb{R}^{L_i m}$, $w_i \in \mathbb{R}^{L_i p}$, $y_{2i} \in \mathbb{R}^{L_i q}$ and $y_{2i} \in \mathbb{R}^{L_i p}$, we write the model for subsystem i as

$$\dot{\xi}_i(t) = f_i(\xi_i(t), \upsilon_i(t), \omega_i(t)) \tag{2.38}$$

$$\zeta_{1i}(t) = h_{1i}(\xi_i(t)) \tag{2.39}$$

$$\zeta_{2i}(t) = h_{2i}(\xi_i(t)) \tag{2.40}$$

where $f_i(\xi_i(t), \upsilon_i(t), \omega_i(t))$, $h_{1i}(\xi_i(t))$ and $h_{2i}(\xi_i(t))$ are obtained by concatenating the terms $g^\ell(\xi^\ell(t), v^\ell(t), \omega^\ell(t))$, $\varsigma(\xi^\ell(t))$ and $\sigma(\xi^\ell(t))$, respectively, over each ℓ in group i.

We assume, without loss of generality, that the agents are indexed such that the first L_1 constitute group 1, the next L_2 group 2, and so on. It then follows from (2.26) that

$$\begin{bmatrix} \omega_1(t) \\ \vdots \\ \omega_N(t) \end{bmatrix} = (\tilde{M} \otimes I_p) \begin{bmatrix} \zeta_{21}(t) \\ \vdots \\ \zeta_{2N}(t) \end{bmatrix}, \tag{2.41}$$

since the respective vectors in (2.26) and (2.41) are identical. Without this assumption an appropriate permutation can be applied to the matrix $\tilde{M}$ and the subsequent results do not change.

Using Assumption 1 we let each agent ℓ in group i apply the feedback $u^\ell = \tilde{v}^\ell(x^\ell, \hat{x}_i, \hat{u}_i)$, and define the practical storage function for subsystem i to be

$$V_i(x_i, \hat{x}_i) = \sum_{\ell \in \text{group } i} \tilde{V}^\ell(x^\ell, \hat{x}_i). \tag{2.42}$$

Then, we obtain the dissipativity property:

$$\begin{aligned}
&\frac{\partial V_i(x_i,\hat{x}_i)}{\partial x_i} f_i(x_i,u_i,w_i) + \frac{\partial V_i(x_i,\hat{x}_i)}{\partial \hat{x}_i} g(\hat{x}_i,\hat{u}_i,\hat{w}_i) \\
&= \sum_{\ell \in \text{group } i} \left\{ \frac{\partial \tilde{V}^\ell(x^\ell,\hat{x}_i)}{\partial x^\ell} g^\ell(x^\ell, \tilde{v}^\ell(x^\ell,\hat{x}_i,\hat{u}_i), w^\ell) + \frac{\partial \tilde{V}^\ell(x^\ell,\hat{x}_i)}{\partial \hat{x}_i} g(\hat{x}_i,\hat{u}_i,\hat{w}_i) \right\} \\
&\leq \sum_{\ell \in \text{group } i} \left\{ -\tilde{\eta}^\ell(\tilde{V}^\ell(x^\ell,\hat{x}_i)) + \tilde{\rho}^\ell(\|\hat{u}_i\|) + \tilde{\Delta}^\ell(\hat{x}_i) + \begin{bmatrix} w^\ell - \hat{w}_i \\ \sigma(x^\ell) - \sigma(\hat{x}_i) \end{bmatrix}^T \tilde{X} \begin{bmatrix} w^\ell - \hat{w}_i \\ \sigma(x^\ell) - \sigma(\hat{x}_i) \end{bmatrix} \right\} \\
&\leq -\eta_i(V_i(x_i,\hat{x}_i)) + \rho_i(\|\hat{u}_i\|) + \Delta_i(\hat{x}_i) + \begin{bmatrix} w_i - (\mathbf{1}_{L_i} \otimes I_p)\hat{w}_i \\ y_{2i} - (\mathbf{1}_{L_i} \otimes I_p)\hat{y}_{2i} \end{bmatrix}^T X_i \begin{bmatrix} w_i - (\mathbf{1}_{L_i} \otimes I_p)\hat{w}_i \\ y_{2i} - (\mathbf{1}_{L_i} \otimes I_p)\hat{y}_{2i} \end{bmatrix}
\end{aligned}$$

where, for $s \in \mathbb{R}_{\geq 0}$ and $y \in \mathbb{R}^n$, we define

$$\eta_i(s) := \min_{z \in \mathbb{R}^L_{\geq 0}} \sum_{\ell \in \text{group } i} \tilde{\eta}^\ell(z_\ell) \text{ s.t.} \sum_{\ell \in \text{group } i} z_\ell = s, \quad \rho_i(s) := \sum_{\ell \in \text{group } i} \tilde{\rho}^\ell(s), \quad \Delta_i(y) := \sum_{\ell \in \text{group } i} \tilde{\Delta}^\ell(y), \tag{2.43}$$

$$X_i := \begin{bmatrix} I_{L_i} \otimes \tilde{X}^{11} & I_{L_i} \otimes \tilde{X}^{12} \\ I_{L_i} \otimes \tilde{X}^{21} & I_{L_i} \otimes \tilde{X}^{22} \end{bmatrix}, \tag{2.44}$$

and where $\tilde{X}^{11}$, $\tilde{X}^{12}$, $\tilde{X}^{21}$, $\tilde{X}^{22}$ denote $p \times p$ matrices obtained by partitioning $\tilde{X} \in \mathbb{R}^{2p \times 2p}$ conformally. Defining, in addition,

$$W_i := I_{L_i p}, \ \hat{W}_i = H_i := \mathbf{1}_{L_i} \otimes I_p \tag{2.45}$$

and

$$\nu_i(s) := \min_{z \in \mathbb{R}^L_{\geq 0}} \sum_{\ell \in \text{group } i} \tilde{\nu}^\ell(z_\ell) \quad \text{s.t.} \sum_{\ell \in \text{group } i} z_\ell = s, \tag{2.46}$$

we summarize the conclusion in the following proposition:

Proposition 1. *Suppose the agents $\ell = 1, \ldots, L$ satisfy Assumption 1, and each subsystem $i = 1, \ldots, N$ is defined as in (2.38)-(2.40), with the abstraction (2.28)-(2.30) obtained by aggregating L_i agents. Then V_i in* (2.42) *is a practical storage function as in Definition 2.4.2, with (2.43)-(2.46), $\hat{h}_{1i}(\hat{x}_i) = \mathbf{1}_{L_i} \otimes \varsigma(\hat{x}_i)$, and $\hat{h}_{2i}(\hat{x}_i) = \sigma(\hat{x}_i)$.*

We next examine the conditions of Theorem 3 and Theorem 4. From (2.45) and (2.16) we have:

$$W = I_{Lp}, \text{ and } \hat{W} = H = P \otimes I_p, \tag{2.47}$$

where

$$P = \text{diag}(\mathbf{1}_{L_1}, \cdots, \mathbf{1}_{L_N}) \tag{2.48}$$

Since we assumed that the agents are indexed such that the first L_1 constitute group 1, the next L_2 group 2, and so on, the definition of P in (2.48) is consistent with the partition matrix defined in (2.27). If the subsystem abstractions are interconnected as in (2.31), then $\hat{M} = \bar{M} \otimes I_p$ and, thus, condition (2.15) of Theorem 3 is identical to the equitability criterion (2.33). This means that we can relax the equitability condition with Theorem 4. The first residual term in (2.21) is then due to the relaxation of equitability, and the second term is due to model variations of non-identical agents, absorbed into $\tilde{\Delta}^\ell$ in Assumption 1 and combined into Δ_i in (2.43).

An optimization problem for near-equitability

We note that relaxing the equitability condition (2.33) results in a residual term given by

$$\bar{Y} := \tilde{M}P - P\bar{M}. \tag{2.49}$$

Our goal now becomes choosing a partition of the agents - equivalently, a partition matrix P and coupling matrix $\bar{M}$ - such that (2.49) is minimized. We propose approaching this task in two steps. First, we allow for some agents to be assigned to groups by hand. Since aggregated agents share the same specification, this allows one to assign agents to separate groups if they require separate specifications. Conversely, one can also assign agents to the same group if it is desirable for them to abide by the same specification. In the second step, the remaining agents are to be assigned to groups automatically via an optimization problem to be defined next. The pre-assigned agents induce an $L \times N$ matrix $\bar{P}$ as follows

$$\bar{P}_{\ell,i} = \begin{cases} 1 & \text{if } \ell \text{ is pre-assigned to group } i \\ 0 & \text{otherwise} \end{cases} \tag{2.50}$$

as well as a diagonal matrix

$$T = \text{diag}(t_1, \cdots, t_N) \tag{2.51}$$

where t_i is the number of agents pre-assigned to group i. We note that if an agent ℓ is not pre-assigned to any group, then the corresponding row ℓ of $\bar{P}$ will contain only zeros.

To partition the remaining agents, we solve a mixed-integer program. We model $\bar{M}$ as a continuous decision variable and, noting (2.27), model P as a binary decision variable. The objective function of our problem is the Frobenius norm of the residual term $\bar{Y}$, the minimization of which yields an equitable partition when one exists, and a near-equitable partition otherwise.

We also note it is possible to enforce (2.49) using *linear* constraints. Since $\tilde{M}$ is fixed, the term $\tilde{M}P$ is linear - the problematic term is $P\bar{M}$, as it is the product of two decision variables. Linearity is achieved with a reformulation, implemented as the command "binmodel" [33] in the toolbox YALMIP [34]. To see the idea for the scalar case, consider the product of a binary variable $p \in \{0, 1\}$ and a continuous variable $m \in \mathbb{R}$. Suppose that m has lower bound $\underline{m} \in \mathbb{R}$ and upper bound $\overline{m} \in \mathbb{R}$. Then, the product $p \cdot m$ can be replaced with a continuous auxiliary variable $y \in \mathbb{R}$ by including the following linear constraints

$$\underline{m}p \leq y \leq \overline{m}p, \quad \underline{m}(1-p) \leq m - y \leq \overline{m}(1-p).$$

This procedure can be applied in a similar fashion to (2.49). Thus, the following optimization problem can be cast as a mixed-integer quadratic program with linear constraints:

Optimization Problem 4:

$$\begin{aligned} &\text{minimize } \|\bar{Y}\|_F \quad \text{over } P,\ \bar{M} \\ &\text{such that } P \text{ is binary}, && (2.52)\\ &\qquad P\mathbf{1}_N = \mathbf{1}_L, && (2.53)\\ &\qquad \mathbf{1}_L^T P \geq \mathbf{1}_N, && (2.54)\\ &\qquad \bar{P}^T P = T, && (2.55)\\ &\qquad \bar{Y} = \tilde{M}P - P\bar{M}, && (2.56) \end{aligned}$$

where (2.53) ensures each node is assigned to exactly one class, (2.54) requires that each class has at least one node assigned to it, and (2.55) assures that the pre-assignments represented by $\bar{P}$ and T, as defined in (2.50) and (2.51), are respected. We note that Optimization Problem 4 is a mixed integer quadratic program and therefore can be solved with an optimization tool such as Gurobi [24].

Note that Optimization Problem 4 minimizes the same residual as Optimization Problem 2, since Y in (2.19) is equal to $\bar{Y} \otimes I_p$. However, here we have the additional flexibility of adjusting P, whereas the equivalent matrices $\hat{W}$ and H in Optimization Problem 2 are fixed. Furthermore, since $\bar{M}$ is selected to minimize the Frobenius norm, the special structure of the matrix P implies that $\bar{Y}$ has the following property:

Lemma 2. *The matrix $\bar{Y}$ obtained by solving Optimization Problem 4 satisfies $\bar{Y}^T\mathbf{1}_L = 0$.*

We will refer back to this fact after we state Theorem 5, at which point it will become relevant.

A special class of agent models

We now study a class of agent models of the form (2.24), (2.25), (2.34) with

$$g^\ell(x, u, w) = \alpha_\ell(x) + \beta_\ell(x)u + Bw,\ \varsigma(x) = x,\ \sigma(x) = Cx, \tag{2.57}$$

where $\alpha_\ell : \mathbb{R}^n \to \mathbb{R}^n$ and $\beta_\ell : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are allowed to vary by agent ℓ and are replaced with nominal ones $\alpha : \mathbb{R}^n \to \mathbb{R}^n$ and $\beta : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, respectively, in the abstraction (2.28)-(2.30):

$$g(\hat{x}, \hat{u}, \hat{w}) = \alpha(\hat{x}) + \beta(\hat{x})\hat{u} + B\hat{w}. \tag{2.58}$$

We note that α_ℓ in (2.57) is assumed to be continuously differentiable. The following proposition gives sufficient conditions under which Assumption 1 holds for (2.57) and (2.58) above:

Proposition 2. *If there exists $\tilde{v}^\ell : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$, $\tilde{\rho}^\ell \in \mathcal{K} \cup \{0\}$, constants λ_ℓ, ϑ_ℓ, and $n \times n$ matrix $Q_\ell = Q_\ell^T > 0$ such that, for all $x \in \mathbb{R}^n$, $\hat{x} \in \mathbb{R}^n$,*

$$Q_\ell \left(\frac{\partial \alpha_\ell(x)}{\partial x}\right) + \left(\frac{\partial \alpha_\ell(x)}{\partial x}\right)^T Q_\ell \leq 2\lambda_\ell I_n \tag{2.59}$$

$$(x - \hat{x})^T Q_\ell \left(\beta_\ell(x)\tilde{v}^\ell(x, \hat{x}, \hat{u}) - \beta(\hat{x})\hat{u}\right) \leq \vartheta_\ell \|x - \hat{x}\|^2 + \tilde{\rho}^\ell(\|\hat{u}\|) \tag{2.60}$$

$$\lambda_\ell + \vartheta_\ell < 0 \tag{2.61}$$

$$Q_\ell B = C^T, \tag{2.62}$$

then Assumption 1 holds with

$$\tilde{V}(x, \hat{x}) = \frac{1}{2}(x - \hat{x})^T Q_\ell (x - \hat{x}), \quad \tilde{\eta}^\ell(s) = \frac{2\varepsilon_\ell}{\lambda_{\max}(Q_\ell)} s,$$
$$\tilde{\Delta}^\ell(\hat{x}) = \frac{1}{4(|\lambda_\ell + \vartheta_\ell| - \varepsilon_\ell)} \|Q_\ell(\alpha_\ell(\hat{x}) - \alpha(\hat{x}))\|^2, \quad \tilde{X} = \frac{1}{2}\begin{bmatrix} 0 & I_p \\ I_p & 0 \end{bmatrix} \tag{2.63}$$

for any choice of $\varepsilon_\ell \in (0, |\lambda_\ell + \vartheta_\ell|)$.

Note that the conditions (2.59) - (2.62) imply that the system in (2.57) is incrementally stabilizable. We also note, in particular, that the term $\tilde{\Delta}^\ell(\hat{x})$ is due to the deviation of $\alpha_\ell(\hat{x})$ from $\alpha(\hat{x})$. Under the hypotheses of Proposition 2 it follows from Proposition 1 that the subsystems and their abstractions satisfy the dissipativity property in Definition 2.4.2 with

$$X_i = \frac{1}{2}\begin{bmatrix} 0 & I_{L_i p} \\ I_{L_i p} & 0 \end{bmatrix}$$

and, if we use identical weights $\mu_i = 1$, $i = 1, \ldots, N$, then the matrix X in Theorem 3 is

$$X = \frac{1}{2}\begin{bmatrix} 0 & I_{Lp} \\ I_{Lp} & 0 \end{bmatrix}.$$

Since $W = I_{Lp}$ by (2.47), condition (2.14) of Theorem 3 is

$$\begin{bmatrix} WM \\ I \end{bmatrix}^T X \begin{bmatrix} WM \\ I \end{bmatrix} = \frac{1}{2}(\tilde{M} + \tilde{M}^T) \otimes I_p \leq 0.$$

Theorem 5. *Suppose the agents $\ell = 1, \ldots, L$ are described by (2.24) - (2.26), (2.34), with the special form (2.57) and interconnection matrix*

$$\tilde{M} + \tilde{M}^T \leq 0, \tag{2.64}$$

and let the hypothesis of Proposition 2 hold. If the partition of the agents is equitable, then V in (2.18) is a practical simulation function from $\hat{\Sigma}$ to Σ with V_i as in (2.42) and $\mu_i = 1$, $i = 1, \ldots, N$. If the equitability condition (2.33) is relaxed so $\bar{Y}$ in (2.49) is nonzero, then V is a practical simulation function if there exists a matrix $Z = Z^T \geq 0$ satisfying (2.20) with $Y = \bar{Y} \otimes I_p$ and $\mu_i = 1$ for $i = 1, \ldots, N$. Furthermore, $\mathcal{N}(\tilde{M} + \tilde{M}^T) \subseteq \mathcal{N}(\bar{Y}^T)$ is a necessary and sufficient condition for such a Z to exist.

The matrix Z in Theorem 5 can be found by solving Optimization Problem 3, where we append the constraint $\mu_i = 1$ for $i = 1, \ldots, N$. Furthermore, when $\bar{M}$ and $\bar{Y}$ are obtained via Optimization Problem 4, the null space condition of Theorem 5 holds automatically if $\mathcal{N}(\tilde{M} + \tilde{M}^T)$ is spanned by $\mathbf{1}_L$, since $\bar{Y}$ satisfies $\bar{Y}^T \mathbf{1}_L = 0$ from Lemma 2. More generally, we also note if the stronger condition

$$\tilde{M} + \tilde{M}^T < 0$$

on the interconnection matrix holds, then the null space condition is satisfied since $\mathcal{N}(\tilde{M} + \tilde{M}^T) = \{\mathbf{0}_L\}$.

2.6 Example

Room Temperature Model

We now consider a temperature control application adapted from [20]. Our goal is to control the temperature of L rooms connected in a circle. We model the dynamics of the temperature $\xi_\ell(t) \in \mathbb{R}$ in room $\ell \in \{1, \ldots, L\}$ as

$$\begin{aligned} \dot{\xi}^\ell(t) &= a_\ell(T_e - \xi^\ell(t)) + b_\ell(T_h - \xi^\ell(t))\upsilon^\ell(t) + \gamma\omega^\ell(t), \\ \omega^\ell(t) &= \xi^{\ell+1}(t) + \xi^{\ell-1}(t) - 2\xi^\ell(t), \end{aligned} \tag{2.65}$$

where $a_\ell, b_\ell, \gamma \in \mathbb{R}_{>0}$ are conduction coefficients (where the former two may depend on room index), T_e and T_h are the temperatures of the external environment and room heater, respectively, and υ^ℓ is a control input. Furthermore, we let $\xi^0 = \xi^L$ and $\xi^1 = \xi^{L+1}$ so that the indices in (2.65) are valid for rooms $\ell = 1$ and $\ell = L$. Note that this model can be represented as in (2.57) with $\alpha_\ell(s) = a_\ell(T_e - s)$, $\beta_\ell(s) = b_\ell(T_h - s)$, $B = \gamma$, and $C = 1$.

Furthermore, the coupling matrix is given by:

$$\tilde{M} = \begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & \ddots & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}. \tag{2.66}$$

Aggregate Model

For the aggregate model, we partition the rooms into $N \leq L$ distinct areas via Optimization Problem 4. The aggregate temperature $\hat{\xi}_i(t)$ in area $i \in \{1, \ldots, N\}$ is governed by

$$\dot{\hat{\xi}}_i(t) = a(T_e - \hat{\xi}_i(t)) + b(T_h - \hat{\xi}_i(t))\hat{v}_i(t) + \gamma\hat{\omega}_i(t)$$

where, in this case, the coupling $\hat{w}_i$ depends on the particular $\bar{M}$ we obtain by solving Optimization Problem 4. The conduction coefficients a and b in the nominal model are obtained by averaging over the conduction coefficients a_ℓ and b_ℓ for the individual rooms, so that $a := \frac{1}{L}\sum_{\ell=1}^{L} a_\ell$ and $b := \frac{1}{L}\sum_{\ell=1}^{L} b_\ell$. In this case, conditions (2.59), (2.60), and (2.62) hold for the function

$$\tilde{v}^\ell(x, \hat{x}, \hat{u}) = \frac{1}{b_\ell(T_h - x)}\left[b(T_h - \hat{x})\hat{u} - k_\ell(x - \hat{x})\right] \tag{2.67}$$

where $k_\ell \in \mathbb{R}_{\geq 0}$, $\tilde{\rho}^\ell(\|\hat{u}\|) = 0$, $\lambda_\ell = -a_\ell/\gamma$, $\vartheta_\ell = -k_\ell$, and $Q_\ell = 1/\gamma$. Furthermore, condition (2.61) is satisfied if the gain k_ℓ is chosen such that $k_\ell > -a_\ell/\gamma$. Therefore, the result of Theorem 5 is applicable to this example, since $\tilde{M} = \tilde{M}^T \leq 0$. We also note that division by zero in (2.67) can be avoided by imposing constraints on $\hat{x}$ in a formal synthesis procedure - indeed, by combining this with a bound on the error between x and $\hat{x}$, we can conclude that x will never reach the heater temperature T_h. A similar line of reasoning ensures the inputs of the aggregated systems will not diverge arbitrarily far from each other due to varying state errors. Indeed, when the error is zero for all aggregated systems in a group, i.e. $x = \hat{x}$, (2.67) reduces to $\tilde{v}^\ell = (b/b_\ell) \cdot \hat{u}$. Thus, taking into account the difference between each b_ℓ and b, one can again use the bound on the error between x and $\hat{x}$ to bound the deviation of $\tilde{v}^\ell$ from $\hat{u}$.

Temperature Regulation

We consider the task of regulating the temperature in a network of $L = 30$ rooms connected in a circle. The coupling matrix $\tilde{M} \in \mathbb{R}^{30\times 30}$ is as shown in (2.66). We assume rooms 1-6, 11-18, and 21-27 are pre-assigned to 3 separate groups; the remaining rooms are assumed to

Table 2.1: Partitioning of the 30 rooms into 3 groups.

	Group 1	Group 2	Group 3
Pre-assignments	1-6	11-18	21-27
Final partition	1-6	7-20	21-30

be flexible with regard to temperature level, and are assigned to groups automatically via Optimization Problem 4. The pre-assignments and final partition are shown in Table 2.1. The aggregate coupling matrix between the groups, obtained simultaneously with the final partition via Optimization Problem 4, is given by

$$\bar{M} = \begin{bmatrix} -1/3 & 1/6 & 1/6 \\ 1/14 & -1/7 & 1/14 \\ 1/10 & 1/10 & -1/5 \end{bmatrix}.$$

One notes that this partition is *not* equitable - indeed, with the pre-assignments shown in Table 2.1, an equitable partition cannot be achieved. This is not problematic, however, since Theorem 5 relaxes the requirement of equitability of our partition, as long as we can find a matrix $Z \geq 0$ satisfying (2.20), where $Y = \bar{Y} \otimes I_p$ and $\mu_i = 1$, $i = 1, 2, 3$. Lemma 2 and Theorem 5 guarantee this is possible, however, since $\mathcal{N}(\tilde{M} + \tilde{M}^T)$ is spanned by $\mathbf{1}_L$ in this case, as $\tilde{M}$ is a Laplacian matrix. Thus, we solve Optimization Problem 3, with the additional constraint $\mu_i = 1$, $i = 1, 2, 3$ as mentioned, and obtain

$$Z = \begin{bmatrix} 2.0016 & -1.0490 & -0.9526 \\ -1.0490 & 1.9897 & -0.9407 \\ -0.9526 & -0.9407 & 1.8933 \end{bmatrix}.$$

Since we also relaxed the assumption of identical agents, the conduction coefficients a_ℓ and b_ℓ in our concrete model are permitted to vary between rooms. For each room, we select a_ℓ from a normal distribution with mean 0.005 and standard deviation 0.0015, and select b_ℓ from a normal distribution with mean 0.035 and standard deviation 0.0075. Furthermore, since Theorem 1 allows us to aggregate subsystems with non-equal initial conditions, we select the initial temperature for each concrete room from a normal distribution with mean 18 and standard deviation 0.15. We then set

$$\hat{\xi}_i(0) = \frac{1}{L_i} \left(\sum_{\ell \in \text{group } i} \xi^\ell(0) \right) \tag{2.68}$$

that is, the initial temperature of each aggregate room is equal to the average temperature of the aggregated rooms in its group. To demonstrate the robustness of our approach, we chose the standard deviation for the parameters and the initial conditions to be sufficiently

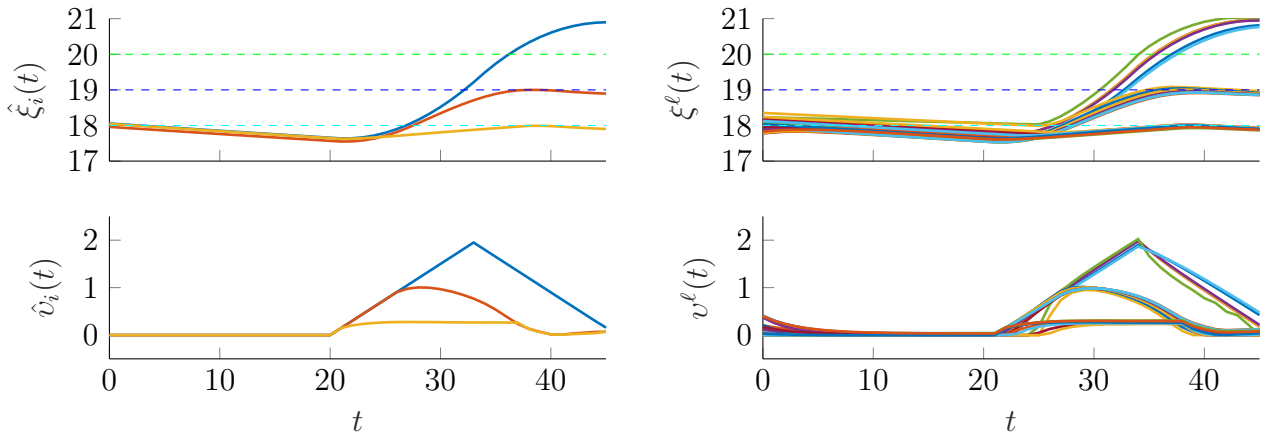


Figure 2.3. Simulation results for the temperature regulation example. We require the temperature in each area of the building to reach its corresponding target temperature range (indicated by the dashed lines) within 20 minutes after the signal is triggered. The signal is triggered at the 20 minute mark - the aggregate system (left) reaches the temperature target within 20 minutes, and the concrete system (right) closely follows the reference.

large so that room temperatures within each group deviate visibly from each other during simulation (as seen in Figure 2.3).

We require the room temperature in the three areas of the building to increase to three separate temperature ranges in response to a signal which indicates, for example, that the building is currently occupied and must be adjusted to a more comfortable temperature. This specification can be represented via, for example, a signal temporal logic (STL) formula [16, 35]. Due to lack of space, we omit the details of the STL formula and refer the reader to [59], which includes two similar examples. Although STL formulae are typically evaluated with respect to continuous time signals (see [35], which considers dense-time real-valued signals), here we use the MPC approach from [47] which defines a semantics for STL over discrete time signals. Since the approach in [47] utilizes mixed-integer programming, the computational burden of control synthesis of $\hat{v}$ is reduced significantly by using an aggregate model. The aggregate input is refined to a concrete input via the interface function (2.67) with $k_\ell = 2.5$ for $\ell = 1, \ldots, L$. Simulation results are shown in Figure 2.3.

2.7 Proofs of Main Results

Proof of Theorem 2

Let $\epsilon = x - P\hat{x}$ and note that we have the following bounds

$$\|h(x) - \hat{h}(\hat{x})\|^2 \ = \epsilon^T C^T C \epsilon \leq \lambda_{\max}(C^T C)\|\epsilon\|^2, \quad \lambda_{\min}(U)\|\epsilon\|^2 \leq \epsilon^T U \epsilon = V(x, \hat{x}),$$

for all x and $\hat{x}$, since $\hat{C} = CP$. Thus, (2.1) holds with $\nu(s) = s^2\lambda_{\min}(U)/\lambda_{\max}(C^TC)$, where $\nu \in \mathcal{K}_\infty$ since U is positive definite.

Next, we apply the congruency transformation $\mathrm{diag}(U, I)$ to (2.12), yielding the equivalent condition

$$\begin{bmatrix} A_K^TU + UA_K + \alpha U & UW \\ W^TU & -\alpha I \end{bmatrix} \leq 0.$$

where we have defined $A_K \triangleq A + BK$. Thus, for all x, $\hat{x}$, and $\hat{u}$ (determining ϵ and d), we have

$$\begin{bmatrix} \epsilon \\ d \end{bmatrix}^T \begin{bmatrix} A_K^TU + UA_K + \alpha U & UW \\ W^TU & -\alpha I \end{bmatrix} \begin{bmatrix} \epsilon \\ d \end{bmatrix} \leq 0$$

so that

$$\begin{aligned} \nabla V\,(x,\hat{x})^T \begin{bmatrix} Ax + BK(x - P\hat{x}) + BQ\hat{x} + BR\hat{u} \\ \hat{A}\hat{x} + \hat{B}\hat{u} \end{bmatrix} &= \epsilon^T \left[A_K^TU + UA_K\right]\epsilon + d^TW^TU\epsilon + \epsilon^TUWd \\ &\leq -\alpha\epsilon^TU\epsilon + \alpha d^Td \\ &= -\alpha V(x,\hat{x}) + \alpha\|\hat{u}\|^2 + \alpha\|D\hat{x}\|^2 \end{aligned}$$

which verifies that (2.2) holds with $\eta(s) = \alpha s$, $\rho(s) = \alpha s^2$, and $\Delta(\hat{x}) = \alpha\|D\hat{x}\|^2$.

Proof of Theorem 4

Without modifications due to our relaxation, we can construct a $\mathcal{K}_\infty$ function ν satisfying (2.1) as in the proof of Theorem 4.2 given in [74]. Thus, we omit this portion of the proof and focus on showing that (2.2) holds. We define the following error between the concrete and aggregate systems

$$\begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} := \begin{bmatrix} h_{21}(x_1) - H_1\hat{h}_{21}(\hat{x}_1) \\ \vdots \\ h_{2N}(x_N) - H_N\hat{h}_{2N}(\hat{x}_N) \end{bmatrix}.$$

Then, from (2.13) and (2.19), it follows that

$$W \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} - \hat{W} \begin{bmatrix} \hat{w}_1 \\ \vdots \\ \hat{w}_N \end{bmatrix} = WM \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} + Y \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix}. \tag{2.69}$$

Now, using the relation (2.69), we obtain

$$\begin{bmatrix} W\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} - \hat{W}\begin{bmatrix} \hat{w}_1 \\ \vdots \\ \hat{w}_N \end{bmatrix} \\ h_{21}(x_1) - H_1\hat{h}_{21}(\hat{x}_1) \\ \vdots \\ h_{2N}(x_N) - H_N\hat{h}_{2N}(\hat{x}_N) \end{bmatrix}^T X(\mu_1 X_1, \ldots, \mu_N X_N) \begin{bmatrix} W\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} - \hat{W}\begin{bmatrix} \hat{w}_1 \\ \vdots \\ \hat{w}_N \end{bmatrix} \\ h_{21}(x_1) - H_1\hat{h}_{21}(\hat{x}_1) \\ \vdots \\ h_{2N}(x_N) - H_N\hat{h}_{2N}(\hat{x}_N) \end{bmatrix}$$

$$= \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \\ e_1 \\ \vdots \\ e_N \end{bmatrix}^T \begin{bmatrix} Y & WM \\ 0 & I_{\tilde{q}} \end{bmatrix}^T X \begin{bmatrix} Y & WM \\ 0 & I_{\tilde{q}} \end{bmatrix} \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \\ e_1 \\ \vdots \\ e_N \end{bmatrix} \leq \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix}^T Z \begin{bmatrix} \hat{h}_{21}(\hat{x}_1) \\ \vdots \\ \hat{h}_{2N}(\hat{x}_N) \end{bmatrix}$$

where the inequality follows from the fact that Z and $\mu_1, \ldots, \mu_N$ satisfy (2.20). Using this bound, the proof of Theorem 4.2 given in [74] can be easily modified to show that (2.2) holds for an appropriate choice of $\eta \in \mathcal{K}_\infty$, $\rho \in \mathcal{K} \cup \{0\}$, and with $\Delta(\hat{x})$ as defined in (2.21). Therefore, we conclude that V in (2.18) is a practical simulation function from $\hat{\Sigma}$ to Σ.

Proof of Lemma 2

We note that P has the form $P = \text{diag}(\mathbf{1}_{L_1}, \cdots, \mathbf{1}_{L_N})$ upon a permutation. Therefore,

$$P\bar{M} = \begin{bmatrix} \bar{m}_{11}\mathbf{1}_{L_1} & \ldots & \bar{m}_{1N}\mathbf{1}_{L_1} \\ \bar{m}_{21}\mathbf{1}_{L_2} & \ldots & \bar{m}_{2N}\mathbf{1}_{L_2} \\ \vdots & & \vdots \\ \bar{m}_{N1}\mathbf{1}_{L_N} & \ldots & \bar{m}_{NN}\mathbf{1}_{L_N} \end{bmatrix}$$

where the $\bar{m}_{ij} \in \mathbb{R}$ denote entries of $\bar{M}$. Let

$$\begin{bmatrix} v_{11} & \ldots & v_{1N} \\ v_{21} & \ldots & v_{2N} \\ \vdots & & \vdots \\ v_{N1} & & v_{NN} \end{bmatrix} := \tilde{M}P, \quad v_{ij} \in \mathbb{R}^{L_i}.$$

Then, we see that

$$\tilde{M}P - P\bar{M} = \begin{bmatrix} v_{11} - \bar{m}_{11}\mathbf{1}_{L_1} & \ldots & v_{1N} - \bar{m}_{1N}\mathbf{1}_{L_1} \\ \vdots & & \vdots \\ v_{N1} - \bar{m}_{N1}\mathbf{1}_{L_N} & \ldots & v_{NN} - \bar{m}_{NN}\mathbf{1}_{L_N} \end{bmatrix}$$

and

$$\|\tilde{M}P - P\bar{M}\|_F = \left\| \begin{bmatrix} v_{11} - \bar{m}_{11}\mathbf{1}_{L_1} \\ \vdots \\ v_{N1} - \bar{m}_{N1}\mathbf{1}_{L_N} \\ \vdots \\ v_{1N} - \bar{m}_{1N}\mathbf{1}_{L_1} \\ \vdots \\ v_{NN} - \bar{m}_{NN}\mathbf{1}_{L_N} \end{bmatrix} \right\|.$$

Minimization of the latter Euclidean norm over $\bar{M}$ can be decomposed into the independent problems

$$\min_{\bar{m}_{ij}} \|v_{ij} - \bar{m}_{ij}\mathbf{1}_{L_i}\|, \quad i,j = 1,\dots,N.$$

Since $\|v_{ij} - \bar{m}_{ij}\mathbf{1}_{L_i}\|^2 = (v_{ij} - \bar{m}_{ij}\mathbf{1}_{L_i})^T(v_{ij} - \bar{m}_{ij}\mathbf{1}_{L_i}) = v_{ij}^T v_{ij} - 2\bar{m}_{ij}\mathbf{1}_{L_i}^T v_{ij} + \bar{m}_{ij}^2 L_i$, the minimizer is $\bar{m}_{ij}^* = (1/L_i)\mathbf{1}_{L_i}^T v_{ij}$.

We now verify the claim of Lemma 2; we have

$$\mathbf{1}_L^T P = \mathbf{1}_L^T \begin{bmatrix} \mathbf{1}_{L_1} & & \\ & \ddots & \\ & & \mathbf{1}_{L_N} \end{bmatrix} = \begin{bmatrix} L_1 & \dots & L_N \end{bmatrix}$$

thus,

$$\mathbf{1}_L^T P\bar{M} = \begin{bmatrix} L_1 & \dots & L_N \end{bmatrix} \bar{M} = \begin{bmatrix} \sum_{i=1}^N \bar{m}_{i1}L_i & \dots & \sum_{i=1}^N \bar{m}_{iN}L_i \end{bmatrix}.$$

Since the optimal values for $\bar{m}_{ij}$ give

$$\sum_{i=1}^N \bar{m}_{ij}^* L_i = \sum_{i=1}^N \mathbf{1}_{L_i}^T v_{ij} = \mathbf{1}_L^T \begin{bmatrix} v_{1j} \\ \vdots \\ v_{Nj} \end{bmatrix},$$

we get

$$\mathbf{1}_L^T P\bar{M} = \mathbf{1}_L^T \begin{bmatrix} v_{11} & \dots & v_{1N} \\ \vdots & & \vdots \\ v_{N1} & & v_{NN} \end{bmatrix} = \mathbf{1}_L^T \tilde{M}P$$

and therefore $\mathbf{1}_L^T(P\bar{M} - \tilde{M}P) = \mathbf{1}_L^T \bar{Y} = 0$.

Proof of Proposition 2

If we let $\tilde{V}^\ell = \frac{1}{2}(x-\hat{x})^T Q_\ell (x-\hat{x})$, then (2.35) holds with $\varsigma(x) = x$, $\tilde{\nu}^\ell(s) = \frac{1}{2}\lambda_{\min}(Q_\ell)s^2$, and (2.36) becomes

$$\begin{aligned} &(x-\hat{x})^T Q_\ell(\alpha_\ell(x) - \alpha(\hat{x})) + (x-\hat{x})^T Q_\ell\left(\beta_\ell(x)\tilde{v}^\ell(x,\hat{x},\hat{u}) - \beta(\hat{x})\hat{u}\right) + (x-\hat{x})^T Q_\ell B(w-\hat{w}) \\ &\le (x-\hat{x})^T Q_\ell(\alpha_\ell(x) - \alpha(\hat{x})) + \vartheta_\ell\|x-\hat{x}\|^2 + \tilde{\rho}^\ell(\|\hat{u}\|) + (\sigma(x) - \sigma(\hat{x}))^T(w-\hat{w}), \end{aligned} \tag{2.70}$$

where the inequality follows from (2.60) and (2.62), combined with $\sigma(x) = Cx$ from (2.57). We rewrite the first term on the right hand side of (2.70) as

$$(x-\hat{x})^T Q_\ell(\alpha_\ell(x)-\alpha(\hat{x})) = (x-\hat{x})^T Q_\ell(\alpha_\ell(x)-\alpha_\ell(\hat{x})) + (x-\hat{x})^T Q_\ell(\alpha_\ell(\hat{x})-\alpha(\hat{x})). \quad (2.71)$$

It follows from (2.59) that

$$(x-\hat{x})^T Q_\ell(\alpha_\ell(x)-\alpha_\ell(\hat{x})) \leq \lambda_\ell \|x-\hat{x}\|^2. \quad (2.72)$$

To see this, define the function $\Omega(t) = \alpha_\ell(\hat{x} + t(x-\hat{x}))$ and note

$$(x-\hat{x})^T Q_\ell \int_0^1 \Omega'(t)dt \quad (2.73)$$

is equal to the left hand side of (2.72) by the fundamental theorem of calculus. From the chain rule, (2.73) equals

$$(x-\hat{x})^T Q_\ell \int_0^1 J(\hat{x} + t(x-\hat{x}))dt(x-\hat{x}) \quad (2.74)$$

where J is the Jacobian of α_ℓ. Rewriting (2.74) as

$$\frac{1}{2}(x-\hat{x})^T \left(\int_0^1 \left(Q_\ell J + J^T Q_\ell \right) dt \right) (x-\hat{x}),$$

we see from (2.59) that the integrand is bounded above by $2\lambda_\ell I_n$, which confirms (2.72). Next, we note that

$$(x-\hat{x})^T Q_\ell(\alpha_\ell(\hat{x})-\alpha(\hat{x})) \leq \kappa\|x-\hat{x}\|^2 + \frac{1}{4\kappa}\|Q_\ell(\alpha_\ell(\hat{x})-\alpha(\hat{x}))\|^2 \quad (2.75)$$

for any choice of $\kappa > 0$, which follows from Young's inequality [73]. Then, from (2.71), (2.72) and (2.75), an upper bound on (2.70) is

$$(\lambda_\ell + \vartheta_\ell + \kappa)\|x-\hat{x}\|^2 + \frac{1}{4\kappa}\|Q_\ell(\alpha_\ell(\hat{x})-\alpha(\hat{x}))\|^2 + \tilde{\rho}^\ell(\|\hat{u}\|) + (\sigma(x)-\sigma(\hat{x}))^T(w-\hat{w}). \quad (2.76)$$

We select $\kappa = |\lambda_\ell + \vartheta_\ell| - \varepsilon_\ell$, which is positive since $\varepsilon_\ell \in (0, |\lambda_\ell + \vartheta_\ell|)$, and note that (2.76) becomes

$$-\varepsilon_\ell\|x-\hat{x}\|^2 + \frac{1}{4(|\lambda_\ell + \vartheta_\ell| - \varepsilon_\ell)}\|Q_\ell(\alpha_\ell(\hat{x})-\alpha(\hat{x}))\|^2 + \tilde{\rho}^\ell(\|\hat{u}\|) + (\sigma(x)-\sigma(\hat{x}))^T(w-\hat{w}). \quad (2.77)$$

Substituting the inequality $\varepsilon_\ell\|x-\hat{x}\|^2 \geq \frac{2\varepsilon_\ell}{\lambda_{\max}(Q_\ell)}\tilde{V}^\ell = \tilde{\eta}^\ell(\tilde{V}^\ell)$ in (2.77), we obtain (2.37) with the terms defined in (2.63).

Proof of Theorem 5

We have shown the equitability criterion (2.33) is identical to condition (2.15) of Theorem 3; also, that if we select $\mu_i = 1$, $i = 1, \ldots, N$, then (2.64) implies condition (2.14) of Theorem 3 holds. Thus, if we use an equitable partition for aggregation and (2.64) holds, then both conditions of Theorem (3) also hold so that (2.18) is indeed a simulation function from $\hat{\Sigma}$ to Σ, with $V_i(x_i, \hat{x}_i)$ as in (2.42), and where $\mu_i = 1$, $i = 1, \ldots, N$. It follows that relaxing the equitability condition as in (2.49) is identical to the relaxation (2.19) given in Theorem 4. Thus, in this case one must choose a matrix $Z = Z^T \geq 0$ satisfying (2.20), with $Y = \hat{Y} \otimes I_p$ and $\mu_i = 1$ for $i = 1, \ldots, N$.

To show that $\mathcal{N}(\tilde{M} + \tilde{M}^T) \subseteq \mathcal{N}(\bar{Y}^T)$ is a necessary and sufficient condition for such a Z to exist, we prove the following fact. Let $B \in \mathbb{R}^{m \times n}$ be an arbitrary matrix and $C \in \mathbb{R}^{n \times n}$ be such that $C = C^T \leq 0$. Then, there exists a matrix $A \in \mathbb{R}^{n \times n}$ such that $A = A^T \geq 0$ and

$$\begin{bmatrix} -A & B \\ B^T & C \end{bmatrix} \leq 0 \tag{2.78}$$

if and only if $\mathcal{N}(C) \subseteq \mathcal{N}(B)$. To see the necessity, suppose there exists a vector $y \in \mathbb{R}^n$ such that $y \in \mathcal{N}(C)$ but $y \notin \mathcal{N}(B)$. Then, for any $x \in \mathbb{R}^m$, we have

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} -A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = -x^T A x + 2x^T B y. \tag{2.79}$$

Let $x = \theta B y$, where $\theta \in \mathbb{R}_{>0}$. Then, a lower bound for (2.79) is

$$(2\theta - \theta^2 \lambda_{\max}(A)) \|By\|^2$$

which is positive for any choice of $\theta \in (0, 2/\lambda_{\max}(A))$. Thus, for any $A = A^T \geq 0$, condition (2.78) does not hold. For the sufficiency, suppose $\mathcal{N}(C) \subseteq \mathcal{N}(B)$, and let $\phi > 0$ be the smallest nonzero eigenvalue of $-C$ (if $-C$ has no nonzero eigenvalues, then C is the zero matrix and the proof follows trivially). We select $A = -(1/\phi) B B^T$, and note that

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} -(1/\phi) B B^T & B \\ B^T & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = -(1/\phi) x^T B B^T x + 2x^T B y + y^T C y. \tag{2.80}$$

Next, we decompose y as $y = y_1 + y_2$, where $y_1 \in \mathcal{N}(C)$ and $y_1^T y_2 = 0$. We note that, by assumption, (2.80) becomes

$$\begin{aligned} -(1/\phi) x^T B B^T x + 2x^T B y_2 + y_2^T C y_2 &\leq -(1/\phi) x^T B B^T x + 2x^T B y_2 - \phi \|y_2\|^2 \\ &= -(1/\phi) \|z\|^2 + 2z^T y_2 - \phi \|y_2\|^2 \end{aligned} \tag{2.81}$$

where the second step follows since $y_2 \notin \mathcal{N}(C)$, and the third step results from the definition $z := B^T x$. Finally, using Young's inequality [73] as

$$z^T y_2 \leq \frac{1}{2\phi} \|z\|^2 + \frac{\phi}{2} \|y_2\|^2$$

one can see that (2.81) is bounded above by zero.

We can then recover the null space condition of Theorem 5 as follows. Using the notation in (2.20), we note that

$$Q(Z,1,\ldots,1) = \begin{bmatrix} -Z & \frac{1}{2}\bar{Y}^T \otimes I_p \\ \frac{1}{2}\bar{Y} \otimes I_p & \frac{1}{2}(\tilde{M}+\tilde{M}^T) \otimes I_p \end{bmatrix}$$

which can be mapped to the matrix in (2.78) by taking $A = Z$, $B = \frac{1}{2}\bar{Y}^T \otimes I_p$ and $C = \frac{1}{2}(\tilde{M}+\tilde{M}^T)\otimes I_p$. Thus, a necessary and sufficient condition for the existence of a $Z = Z^T \geq 0$ such that $Q(Z,1,\ldots,1) \leq 0$ is $\mathcal{N}(\frac{1}{2}(\tilde{M}+\tilde{M}^T)\otimes I_p) \subseteq \mathcal{N}(\frac{1}{2}\bar{Y}^T \otimes I_p)$, which is equivalent to $\mathcal{N}(\tilde{M}+\tilde{M}^T) \subseteq \mathcal{N}(\bar{Y}^T)$.

Chapter 3

Vehicle Platooning

3.1 Introduction

Vehicle connectivity and autonomy are important areas of research, both of which have made a notable impact on the automotive industry [23]. For example, advanced driver assist systems (ADAS) which automate the longitudinal and lateral motion of the vehicle, such as the Tesla Autopilot and Cadillac Super Cruise systems, are being offered as an option in an increasing number of production vehicles. Furthermore, V2V communication technology is now included as a standard feature in Cadillac CTS sedans [69].

The advent of connected automated vehicles has also paved the way towards significant improvements in transportation broadly [68], including increased safety (by allowing, for example, the detection of vehicles occluded from sight) and reduced reliance on traffic lights [48]. V2V communication allows for nearby vehicles to coordinate their motion accurately and to form *vehicle platoons*: strings of vehicles driving at the same speed and at short distance. There are two primary benefits of vehicle platooning: an improvement in traffic efficiency due to increased roadway capacity, and an increase in fuel efficiency due to reduced aerodynamic drag forces acting on the platooning vehicles, especially for heavy-duty vehicles such as semi-trucks. Regarding the first point, there is demonstrated potential for platooning to increase the capacity of both highways and urban roadways. For example, a microscopic simulation study in [57] predicts that increasing the penetration of vehicles capable of cooperative adaptive cruise control (CACC) will result in an increase in highway capacity, since it enables the driver to select smaller time headways. In [32] the authors predict that the throughput of urban roadways could potentially be doubled by forming platoons of vehicles, particularly by increasing the capacity of intersections, which they confirm with a subsequent simulation study. For the second point, experiments presented in [9] confirm that small spacings between two heavy-duty trucks results in reduced fuel consumption.

Previous demonstrations have showcased the technical feasibility of vehicle platooning. For example, vehicle platooning was demonstrated in 1994 and 1997 by the California PATH team on the I-18 highway in San Diego, CA [56]. Other experimental evaluations conducted

Figure 3.1. Test vehicles at the Hyundai-KIA Motors California Proving Grounds, California City, CA.

on highways include [3], where the authors develop a platooning system architecture for heavy-duty vehicles. The system is evaluated in terms of controller tracking performance and fuel consumption over varying levels of road grade. In [42, 37] the authors present the design of a CACC system and tested it on a fleet of test vehicles. A primary controller performance metric in these works is *string stability* [65], meaning that the preceding vehicles are able to attenuate disturbances in traffic downstream (for example, changes in velocity). In 2011 the first Grand Cooperative Driving Challenge was held in the Netherlands [44], with the goal of accelerating the deployment of cooperative driving technologies. The competition focused on CACC and included both an urban and highway driving challenge [40]. For the urban driving challenge one criterion used to judge the participating teams was throughput improvement at the traffic light. This scenario is similar to the one we considered in our previous work [61], where we focused on the trade-off between traffic throughput gains and safety.

In addition to maintaining a platoon formation, the related tasks of forming, merging, and splitting platoons require structured coordination between vehicles, i.e. interaction protocols, which can be achieved in principle with V2V communication. For example, in [27] state machines are provided which describe the sequence of events, coordinated via V2V communication, that must occur during merge, split, and change lane maneuvers. Furthermore, low level control laws for the leader vehicle to execute these maneuvers have been developed [31]. In [58] an extended message set is proposed for the purpose of enabling connected vehicles to coordinate more complex maneuvers in merging, intersection, and emergency vehicle scenarios for a follow-up Grand Cooperative Driving Challenge which was held in 2016 [43]. Other works studying communication include [15], where the authors present a strategy for maintaining string stability in a vehicle platoon while using significantly fewer communication resources.

Unlike the aforementioned studies, in this work we focus on advancing vehicle platoon-

ing to a public urban environment where increased intersection throughput can result in significant improvements in overall traffic efficiency. Enabling platooning in an urban environment involves addressing various challenges, such as forming and disbanding platoons in moving traffic, decision-making (e.g. whether or not to proceed through an upcoming intersection), and ensuring safety when a lead vehicle is present. These challenges are especially important on a public roadway, where the future behavior of vehicles ahead of the platoon and the phase of upcoming traffic lights are uncertain. We present a design for the urban platooning system, and then analyze performance by estimating throughput using data obtained from simulations and experiments conducted on a closed track. We also introduce a state machine for managing the participating platooning vehicles, and propose strategies for the platoon to ensure safety when it encounters an intersection and / or a leading vehicle, utilizing predictions of their future behavior.

The closest comparable effort that we are aware of is the MAVEN project, which has laid out the various technologies that are needed to develop and deploy urban platooning, and reported on test results with two automated vehicles [54], where technologies such as a green light optimal speed advisory system and a collective perception message were utilized. The MAVEN project has similar goals of increasing traffic efficiency and safety by managing CAVs at signalised intersections. Unlike [54], however, our focus in this chapter is on improving throughput primarily by maintaining short (constant) distances between the vehicles as the platoon accelerates from rest to a nominal speed. In particular, we achieve such accurate tracking by transmitting velocity forecasts between platooning vehicles and using them as disturbance previews in our MPC problems. In contrast, in [54] the platooning vehicles do not create such a tight formation.

The remainder of the chapter is organized as follows. We outline our design for the urban vehicle platooning system in Sections 3.2 - 3.4, including a platoon model and management system, MPC formulation, and strategy for the leader to ensure safety. In Section 3.5 we present results from our simulation tool, and analyze the performance of the platooning system by estimating the potential gains in intersection throughput. Next, in Section 3.6 we discuss the experimental setup and present results from conducting tests on a closed track and on public roadways in Arcadia, CA, including our estimates of throughput. We note that parts of Sections 3.2 - 3.4 are adapted from our previous work [61], but the remaining content in the chapter is completely new and advances platooning to an urban setting.

3.2 Platoon Model and Management

In this section we introduce the model of the platoon and various systems that enable management of its behavior (beyond the control algorithms themselves), including state estimation via on-board sensors, V2X communication, and a finite-state machine (FSM) system which ensures that the platoon acts in a coordinated manner, that is, vehicles start moving as a single platoon at the same time and break the platoon at the same time as needed. In particular, we discuss how vehicle-to-vehicle communication enables the follower

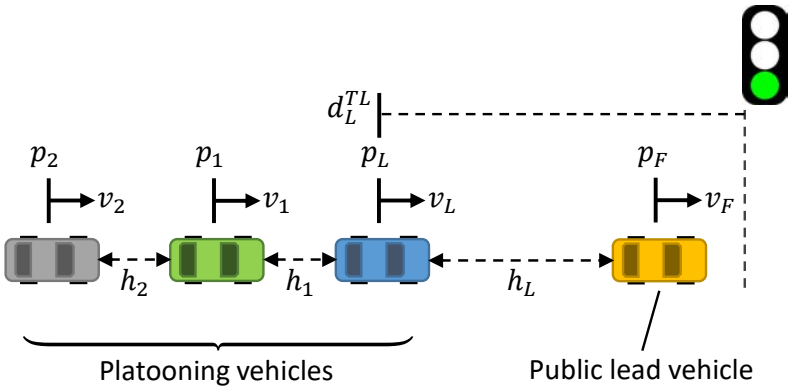


Figure 3.2. Depiction of the states for a platoon of size $N = 3$ and public lead vehicle approaching an upcoming traffic light.

vehicles to do accurate distance tracking of the leader, and how vehicle-to-infrastructure communication enables the leader to decide whether or not to proceed through an upcoming intersection.

Vehicle Models

The longitudinal dynamics of the leader vehicle [25] are modelled as

$$\dot{p}_L(t) = v_L(t), \tag{3.1a}$$
$$\dot{h}_L(t) = v_F(t) - v_L(t), \tag{3.1b}$$
$$\dot{d}_L^{TL}(t) = -v_L(t), \tag{3.1c}$$
$$\dot{v}_L(t) = \frac{1}{M}\left(\frac{T_L^a(t) - T_L^b(t)}{R_w} - F_f(t)\right), \tag{3.1d}$$
$$\dot{T}_L^a(t) = \frac{1}{\tau}\left(T_L^{a,ref}(t) - T_L^a(t)\right), \tag{3.1e}$$

where the states are as follows: $p_L(t)$ is the position, $h_L(t)$ is the distance to the public vehicle ahead (specifically, the distance from the front bumper of the leader vehicle to the rear bumper of the front vehicle), $d_L^{TL}(t)$ is the distance to the nearest upcoming intersection stop bar, $v_L(t)$ is the ego vehicle velocity, and $T_L^a(t) \in \mathbb{R}_{\geq 0}$ is the accelerating wheel torque. The inputs $T_L^{a,ref}(t) \in \mathbb{R}_{\geq 0}$ and $T_L^b(t) \in \mathbb{R}_{\geq 0}$ are the accelerating wheel torque command and the braking wheel torque. Lastly, $v_F(t)$ is the velocity of the public vehicle ahead, henceforth referred to as the front vehicle. The parameters M, R_w, and τ are the vehicle mass, wheel radius, and actuation time constant for acceleration, respectively. We note that (3.1e) models actuation delay while the vehicle is accelerating, which has been empirically estimated by collecting wheel torque measurements from the test vehicle. During these experiments we observed no delay while braking, and therefore the model does not include actuation delay

Table 3.1: Model Parameters

M	vehicle mass	kg	2044
R_w	wheel radius	m	0.3074
β	frictional force modelling parameter	-	339.1329
γ	(same as above)	-	0.77
τ	accelerating torque actuation time constant	s	0.7868
Δt	sampling time	s	0.1

while braking. Lastly, $F_L^f(t)$ is a longitudinal force acting on the leader vehicle, given by

$$F_L^f(t) = Mg\left((\sin(\theta) + r\cos(\theta)\right) + \frac{1}{2}\rho A c_x v_L(t)^2 \tag{3.2}$$

where g is the gravitational constant, θ is road grade, A is the area of the vehicle, r is a rolling coefficient of the vehicle, ρ is air density, and c_x is an air drag coefficient. We assume road grade is negligible, and thus $\theta = 0$ for $t \geq 0$. For simplicity, we represent (3.2) as

$$F_L^f(t) = \beta + \gamma v_L(t)^2 \tag{3.3}$$

where the parameters β, $\gamma \in \mathbb{R}_{\geq 0}$ were identified by collecting driving data at a testing area near UC Berkeley, and then fitting predictions from (3.3) to the data (see Table 3.1). We write the leader vehicle dynamics (3.1) concisely as

$$\dot{x}_L(t) = f_L(x_L(t), u_L(t), w_L(t)) \tag{3.4}$$

where $x_L(t) := [p_L(t);\ h_L(t);\ d_L^{TL}(t);\ v_L(t);\ T_L^a(t)]$, $u_L(t) := [T_L^{a,ref}(t);\ T_L^b(t)]$, and $w_L(t) := v_F(t)$. Note that the velocity of the front vehicle $v_F(t)$ appears as a disturbance here. Since we cannot accurately predict the behavior of non-platooning vehicles, we make the conservative assumption that the front vehicle will decelerate from its current speed until coming to a stop. This assumed trajectory of the front vehicle is used for planning, to be discussed further in Section 3.3.

We model the longitudinal dynamics of each of the $N-1$ follower vehicles in the platoon as

$$\dot{p}_i(t) = v_i(t), \tag{3.5a}$$
$$\dot{h}_i(t) = v_{i-1}(t) - v_i(t), \tag{3.5b}$$
$$\dot{s}_i(t) = v_L(t) - v_i(t), \tag{3.5c}$$
$$\dot{v}_i(t) = \frac{1}{M}\left(\frac{T_i^a(t) - T_i^b(t)}{R_w} - F_f(t)\right), \tag{3.5d}$$
$$\dot{T}_i^a(t) = \frac{1}{\tau}\left(T_i^{a,ref}(t) - T_i^a(t)\right), \quad i = 1, \ldots, N, \tag{3.5e}$$

where $s_i(t)$, used for distance tracking relative to the leader vehicle, is defined as follows:

$$s_i(t) = \sum_{k=1}^{i} h_k(t). \tag{3.6}$$

We refer to $s_i(t)$ as the distance from follower i to the leader (note that (3.6) implies $s_1(t) = h_1(t)$). Furthermore, we let $v_0(t) = v_L(t)$ so that (3.5c) is valid for follower $i = 1$. We write (3.5) compactly as

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t), w_i(t)), \quad i = 1, \ldots, N-1, \tag{3.7}$$

where $x_i(t) := [p_i(t);\ h_i(t);\ d_i^{TL}(t);\ v_i(t);\ T_i^a(t)]$, $u_i(t) := [T_i^{a,ref}(t);\ T_i^b(t)]$, and $w_i(t) := [v_L(t);\ v_{i-1}(t)]$. We note that the velocity of the leader and front vehicle $v_L(t)$ and $v_{i-1}(t)$ appear as disturbances here - since these are both platooning vehicles in this case, we can receive a forecast of their future behavior via V2V communication. In Section 3.2 we discuss the information transmitted between platooning vehicles which includes a velocity forecast, to be used as a disturbance preview in our MPC formulation.

For planning, our goal is to obtain linear, discrete time models from (3.4) and (3.7). We use the procedure outlined in [61] for doing so: we first linearize the leader and follower vehicle dynamics about the nominal velocities v_L^0 and v_i^0, respectively, and then discretize the resulting linear models each with time step $\Delta t = 0.1$s, resulting in

$$\begin{aligned} x_L(k+1) &= A_L x_L(k) + B_L u_L(k) + E_L w_L(k), \\ x_i(k+1) &= A_i x_i(k) + B_i u_i(k) + E_i w_i(k), \end{aligned} \tag{3.8}$$

where the matrices $A_L \in \mathbb{R}^{5\times5}$ and $B_L \in \mathbb{R}^{5\times2}$ are functions of the velocity v_L^0, and $A_i \in \mathbb{R}^{6\times6}$ and $B_i \in \mathbb{R}^{6\times2}$ are functions of the velocity v_i^0. At each time step, the current ego vehicle velocity is substituted into these expressions to obtain the appropriate dynamics matrices to be used for MPC.

State Estimation

To localize the leader and follower vehicle positions $p_L(t)$ and $p_i(t)$ we use a differential GPS measurement which has lane-level accuracy. Furthermore, with GPS and information received from nearby traffic lights we can also estimate the distances $d_L^{TL}(t)$ and $d_i^{TL}(t)$ from each vehicle to the nearest upcoming traffic light. The forward-looking radar on each vehicle measures the headways $h_L(t)$ and $h_i(t)$, and standard on-board sensors provide the current velocity estimates $v_L(t)$ and $v_i(t)$, as well as estimates of the accelerating wheel torques $T_L^a(t)$ and $T_i^a(t)$.

An important sensing challenge for each follower i is to estimate the distance to the leader as defined in (3.6). We have tested two methods for doing so: 1) estimating $s_i(t)$ using GPS, and 2) estimating $s_i(t)$ directly using the radar measurements $h_i(t)$, which can

be transmitted via V2V communication. For the first method, we use GPS to measure the distance $d_i^L(t)$ from the center of vehicle i to the center of the leader vehicle and use the estimate

$$\hat{s}_i(t) = \hat{d}_i^L(t) - i \cdot L_{\text{veh}} \tag{3.9}$$

where $\hat{d}_i^L(t)$ is an estimate of $d_i^L(t)$ from GPS. The main drawback to this approach is GPS measurement noise - we observed up to 3 meters of error when estimating $s_i(t)$ using GPS. Because of this, we also used a Kalman filter, where the idea is to use the current velocity of the leader (received via V2V communication) and the ego vehicle velocity to improve our estimate of $s_i(t)$. For the second method, we use the estimate

$$\hat{s}_i(t) = \sum_{k=1}^{i} \hat{h}_k(t), \tag{3.10}$$

where $\hat{h}_k(t)$ is an estimate of $h_k(t)$ from radar. Since measurements from the forward-looking radar are generally very reliable, we observed smaller measurement errors using the second method. The main drawback to the second approach, however, is that it will require more vehicles in the platoon to communicate with one another (discussed further in the next section). For the experiments discussed in Section 3.6 we used GPS to estimate $s_i(t)$, and for the experiments in Section 3.6 we used radar measurements to estimate $s_i(t)$.

Vehicle-to-vehicle communication

We assume each platooning vehicle is capable of V2V communication. An important piece of information transmitted within the platoon is a forecast of the future velocity trajectory for each vehicle, given by

$$\begin{aligned} v_L^{\text{forecast}} &= [v_L(t|t);\ v_L(t+1|t);\ \ldots;\ v_L(t+N_p|t)], \\ v_i^{\text{forecast}} &= [v_i(t|t);\ v_i(t+1|t);\ \ldots;\ v_i(t+N_p|t)], \end{aligned} \tag{3.11}$$

for the leader vehicle and follower vehicle i, respectively. Here, $v_L(k|t)$ is the planned velocity of the leader vehicle at time step k, obtained by solving an MPC problem at the current time step t (the notation is the same for the follower vehicles), and N_p is the MPC horizon in time steps. Each follower vehicle receives a velocity forecast from the front vehicle and the leader vehicle, corresponding to the flow of information depicted in Figure 3a. The front vehicle forecast is used to ensure safety, and the leader vehicle forecast is used to do distance tracking of the leader.

In addition to the velocity forecast, each experimental vehicle transmits its radar measurement, current GPS coordinates, and plan status signal. A secondary reason for transmitting GPS coordinates, beyond estimating $s_i(t)$, is so that the leader vehicle can estimate the distance $d_L^{N-1}(t)$ from itself to the rear platooning vehicle. The transmission of GPS coordinates from follower $N-1$ to the leader is shown in Figure 3b. This lets the leader check whether

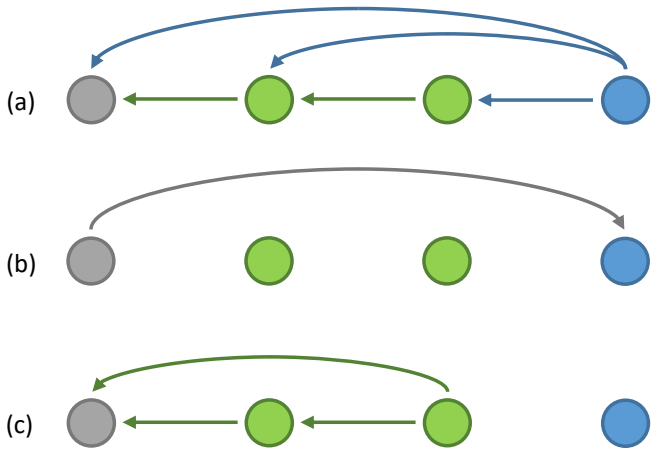


Figure 3.3. Flow of V2V messages for a platoon of size $N = 4$, where the blue node represents the leader vehicle and the grey node represents the rear vehicle. Figure 3a shows the transmission of velocity forecasts and 3b shows the transmission of GPS coordinates from the rear vehicle (used by the leader to determine if the platoon can make it through the intersection, see Section 3.2). Figure 3c shows how we share radar measurements when we use the second method for estimating $s_i(t)$ as in (3.10).

the entire platoon has enough time to pass through an upcoming intersection, as discussed in the next section. As mentioned in the previous section, for some of our experiments we used radar measurements, transmitted via V2V communication, to estimate $s_i(t)$. In Figure 3c we depict the flow of information in this case, for $N = 4$. We note that each vehicle, upon receiving an incoming message, checks the ID of the vehicle that transmitted it (indicating the vehicle's position in the platoon, e.g. leader vehicle, rear vehicle, etc.) to determine which information fields to extract, if any.

Vehicle-to-infrastructure communication

In addition to V2V messages, we assume the platooning vehicles also receive SPaT (signal, phase, and timing) messages from nearby traffic lights via V2I communication. In this way, each vehicle obtains the following prediction on the nearest upcoming traffic light state:

$$\hat{x}_{TL}(t) = [p_{\text{up}}(t);\ c_r(t)] \tag{3.12}$$

where $p_{\text{up}}(t) \in \{\text{red, yellow, green}\}$ is the current phase of the nearest upcoming traffic light and $c_r(t) \in \mathbb{R}_{\geq 0}$ is a prediction on the time remaining in the current phase. We note that it is necessary to predict $c_r(t)$ here since in our experiments the traffic signals are actuated.

In the remainder of this section, we discuss how the leader decides whether or not the platoon should stop at an upcoming traffic light. This decision is handled by the leader only - the follower vehicles simply track the leader, and therefore we do not allow platoon

separation. Suppose the platoon is approaching a traffic light during its green phase, with $c_r(t)$ seconds remaining in the phase. In this scenario, the leader checks if the following condition holds

$$c_r(t) \cdot v_L(t) \geq d_L^{N-1}(t) + d_L^{TL}(t) + L_{int} \tag{3.13}$$

to determine whether a stop is necessary (specifically, if (3.13) is false the platoon should stop), where L_{int} is the intersection length. Condition (3.13) provides a quick and simple way to check whether the rear platooning vehicle, travelling at the current leader velocity $v_L(t)$, will pass through the intersection during the green phase. We use $v_L(t)$ in (3.13) since the leader effectively sets the speed for all platooning vehicles behind it, and also to avoid having to transmit $v_{N-1}(t)$ to the leader. We note that when N is large, $d_L^{N-1}(t)$ is large and thus (3.13) is easily violated. This means the platoon may begin braking during a green light, which can be unexpected for nearby drivers. To avoid this, for large N allowing platoon separation may become necessary.

At low velocity (3.13) is not easily satisfied and will be overly restrictive, for example if the light just turned green and the platoon is stopped. For this reason, if $v_L(t) \leq v_{\text{low}}$ the leader simply checks if the following condition holds

$$c_r(t) \geq t_{\min} \tag{3.14}$$

where the threshold $t_{\min}$ is a tuning parameter. If so, it is considered safe to proceed. By checking (3.13) and (3.14) to determine whether to stop, we try to ensure the platoon will not be crossing the intersection when the phase becomes yellow. However, since the traffic signal is actuated and can change randomly due to uncertain traffic conditions, we cannot formally guarantee that this will never occur.

Suppose the leader determines it should stop while the phase is green, or that the phase is yellow, in which case the leader should stop if it can do so safely. Then, we also check if the leader is capable of stopping before the intersection stop line with a margin of $d_{\min}$, that is

$$\frac{v_L(t)^2}{2a_{\text{min,brake}}} \leq d_L^{TL}(t) - d_{\min} \tag{3.15}$$

where $-a_{\text{min,brake}} \in \mathbb{R}_{<0}$ is an upper bound on (3.1d) while the maximum braking force is applied. If (3.15) does not hold, then it is deemed safer for the leader to proceed through the intersection (in this scenario, for large N a platoon separation may also be necessary). For a red phase, however, we require the platoon to stop in any case.

Finite state machine (FSM)

We have designed a FSM (see Figure 3.4) which acts as a mechanism for safely forming and maintaining a platoon. There are four primary states in our FSM: 'Ready', 'Plan Proposed', 'Plan Active', and 'Plan Cancel'. Each platooning vehicle is initialized in the 'Ready' state and communicates its state at all times. The platoon formation process is initiated when the leader moves to the 'Plan Proposed' state by proposing to the follower vehicles the 'plan',

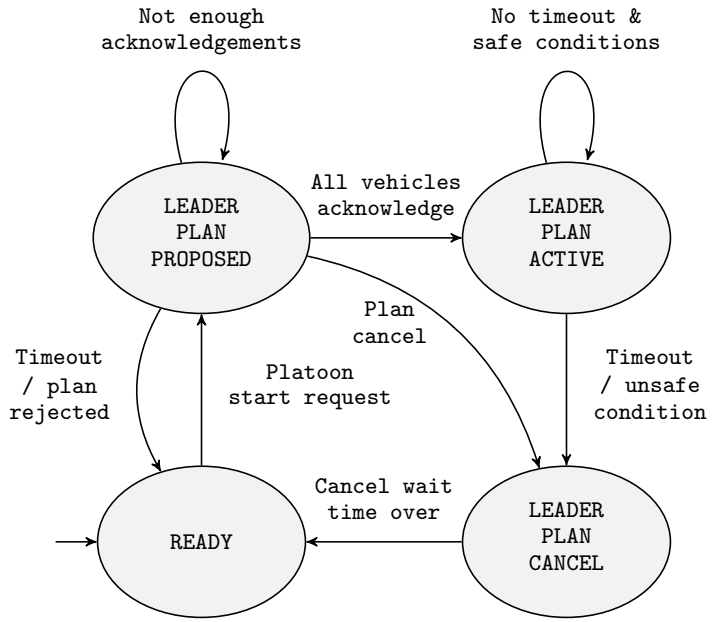


Figure 3.4. A diagram of the transitions in our finite state machine, shown here for the leader vehicle for simplicity.

including a plan ID, ordering of the vehicles in the platoon, desired gap / speed, etc. Note that the ordering of vehicles in the platoon refers to the list of vehicle IDs ordered from the leader to the last follower. As soon as the 'plan' is received by the followers, the states of the followers transition to the 'Plan Proposed' state. In the 'Plan Proposed' state, each vehicle acknowledges that the 'plan' is valid by checking the on-board sensor data and communicated GPS data. For example, each vehicle can confirm that the driver agrees to join the platoon and that the proposed 'Plan' is safe to follow. We also note that the leader can manually cancel the plan while in the 'Plan Proposed' state, forcing a transition to the 'Plan Cancel' state.

When the leader receives an acknowledgement from every vehicle in the 'Plan', it moves to the 'Plan Active' state while also informing the followers so that all vehicles move to the 'Plan Active' state together. To ensure safety, while in the 'Plan Active' state every vehicle in the platoon continuously monitors the surrounding conditions to decide if the 'Plan' must stop. In our experiments, the conditions that cancel the plan include: 1) incorrect ordering of the vehicles, 2) message timeout, 3) any driver taps the gas / brake pedal, 4) front vehicle out of range (radar measurement too high), and 5) velocity upper / lower bound violated. Here, message timeout refers to when a particular message has not been received for a period of time longer than a specified threshold. When one of these conditions is detected by one vehicle, it informs the other vehicles in the platoon and they move together to the 'Plan Cancel' state. After some threshold time, each vehicle transitions from the 'Plan Cancel' state to the 'Ready' state and the platoon can be restarted as needed.

In Figure 3.5 we display some data collected while forming a platoon during testing in

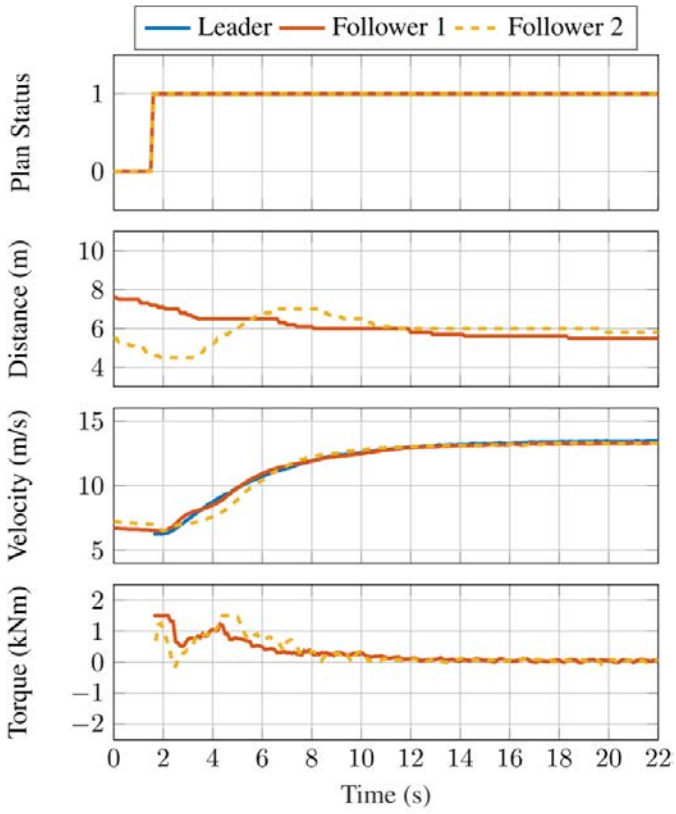


Figure 3.5. Experimental data collected in Arcadia, CA during the platoon formation process. The vehicles begin at a low speed and unequal spacing. At around the 2s mark, the platoon leader proposes a 'plan' which is accepted by the following vehicles, and the plan status signal (plotted above) switches from 0 to 1. This engages all platooning controllers simultaneously, and the vehicles quickly converge to the desired speed and distance.

Arcadia, CA (see Section 3.6). The procedure for forming a platoon was to manually drive the test vehicles to get them close together and moving at similar speeds, at which point the leader vehicle would propose a 'plan' via the state machine and engage the platooning controllers simultaneously. This enabled platoon formation even while the vehicles are moving.

3.3 MPC Formulation

In this section we present our MPC problem formulation for the platoon. The leader vehicle has a separate MPC problem which allows it to react to changing traffic conditions and set the desired velocity for the following vehicles. For example, if a stop at an intersection is necessary, the leader computes a velocity trajectory in order to stop safely and comfortably at the intersection stop bar. Furthermore, the leader maintains a safe following distance

when a vehicle is present ahead of it. The follower vehicles simply do distance tracking relative to the leader, as we do not allow platoon separation.

Leader vehicle MPC

The goal for the leader is to track a desired velocity when it is safe to do so. When necessary, it must yield to a slower-moving front vehicle or stop at the intersection stop bar. The MPC problem for the leader is

$$\min_{u_i(\cdot|t)} \quad J_L = \sum_{k=t}^{t+N_p+1} (v_L(k|t) - v_L^{des})^2 \tag{3.16a}$$

$$+ \sum_{k=t}^{t+N_p} u_L(k|t)^T R u_L(k|t) \tag{3.16b}$$

$$+ \alpha \sum_{k=t}^{t+N_p-1} \|u_L(k+1|t) - u_L(k|t)\|^2 \tag{3.16c}$$

$$\text{s.t.} \quad x_L(k+1|t) = \tag{3.16d}$$

$$A_L x_L(k|t) + B_L u_L(k|t) + E_L \hat{w}_L(k),$$

$$v_{\min} \leq v_L(k|t) \leq v_{\max}, \tag{3.16e}$$

$$d_{\min} + t_h v_L(k|t) \leq d_L^*(k|t), \tag{3.16f}$$

$$0 \leq T_L^a(k|t) \leq T_{\max}^a, \tag{3.16g}$$

$$0 \leq T_L^{a,ref}(k|t) \leq T_{\max}^a, \tag{3.16h}$$

$$0 \leq T_L^b(k|t) \leq T_{\max}^b, \tag{3.16i}$$

$$x_L(t|t) = \hat{x}_L(t), \tag{3.16j}$$

$$\forall k = t, \ldots, t+N_p,$$

$$\begin{bmatrix} d_L^*(t+N_p|t) \\ v_L(t+N_p|t) \end{bmatrix} \in C(\hat{x}_L(t), \hat{v}_F(t), \hat{v}_F(t+N_p)), \tag{3.16k}$$

where N_p is the MPC horizon in time steps, and $x_L(k|t)$ and $u_L(k|t)$ are the planned state and input of the leader vehicle at time step k, computed at time step t, respectively (the notation for the other states is the same). Furthermore, $d_L^*(k|t)$ is the distance from the leader vehicle to either the front vehicle or the upcoming intersection stop bar - whichever is a higher priority obstacle (the method for determining this is outlined in Section 3.4). Lastly, $\hat{x}_L(t)$, $\hat{v}_F(t)$ are estimates of the leader vehicle and front vehicle state, based on measurements from the on-board sensors, and $\hat{v}_F(t+N_p)$ is an estimate of the front vehicle velocity at the end of the MPC planning horizon. Indeed, since $\hat{w}_L(k) := \hat{v}_F(k)$ appears as a disturbance in (3.16d), we must predict the future velocity trajectory of the front vehicle. To ensure safety, we assume worst-case behavior, i.e. the front vehicle will decelerate from

Table 3.2: MPC Parameters

d^{des}	desired distance	m	6
$d_{\min}$	minimum distance (front vehicle)	m	6
$d_{\min}$	minimum distance (stop bar)	m	5
t_h	time headway	s	1.6
v_L^{des}	desired velocity (leader)	m/s	15
$v_{\min}$	minimum velocity	m/s	0
$v_{\max}$	maximum velocity	m/s	20
$T^a_{\max}$	maximum accelerating torque	Nm	1500
$T^b_{\max}$	maximum braking torque	Nm	2000
N_p	MPC horizon	-	20

its current speed at the rate $a_{\text{max,brake}} \in \mathbb{R}_{>0}$ until coming to a complete stop as follows

$$\hat{w}_L(k) := \hat{v}_F(k) = \begin{cases} \tilde{v}^0, & k = t, \\ \max(0,\ \hat{v}_F(k-1) - k \cdot a_{\text{max,brake}} \cdot \Delta t), & k = t+1, \dots, t+N_p, \end{cases} \tag{3.17}$$

where $\tilde{v}^0$ is an under-approximation of the front vehicle's current velocity v^0, to be discussed further in Section 3.4. Here, $-a_{\text{max,brake}} \in \mathbb{R}_{<0}$ is a lower bound for (3.1d) and (3.5d) while the maximum braking force is applied.

The leader vehicle cost function J_L penalizes deviations from the desired velocity v_L^{des} (3.16a), nonzero control inputs (3.16b), and nonzero control input rates (3.16c), effectively penalizing vehicle jerk. The scalar $\alpha \in \mathbb{R}_{>0}$ and matrix $R \in \mathbb{R}^{2\times 2}$ are design parameters which allow one to tune controller performance. Increasing α, for example, smooths the acceleration and deceleration profiles of the vehicle, but reduces the controller's agility. Furthermore, we set

$$R = \begin{bmatrix} R_a & R_0 \\ R_0 & R_b \end{bmatrix} \tag{3.18}$$

where the diagonal entries $R_a,\ R_b \in \mathbb{R}$ can be increased to encourage the controller to use smaller actuation torques $T^L_{a,ref}(t)$ and $T^L_a(t)$, respectively, and the off-diagonal entries $R_0 \in \mathbb{R}$ are made sufficiently large in order to prevent the accelerating and braking control inputs from being active simultaneously.

The leader MPC problem is subject to the following constraints: vehicle dynamics (3.16d), lower and upper bounds on velocity (3.16e), distance constraint (3.16f), torque and reference torque constraints (3.16g) - (3.16i), and initial condition (3.16j). The terminal constraint (3.16k) ensures the leader maintains a safe distance to any obstacle ahead (namely, a front vehicle or intersection requiring a stop), and will be discussed further in Section 3.4. The parameters $d_{\min}$ and t_h are tuned to increase passenger comfort. For example, if t_h is too small it may feel as if the vehicle is braking late when approaching slow-moving traffic or

a stop bar, and if t_h is too large the vehicle will brake harshly in response to cut-in vehicles. The values of all MPC parameters are given in Table 3.2.

At each time step, the leader vehicle solves its MPC problem and obtains an optimal control input sequence and velocity trajectory:

$$u_L(t|t),\ u_L(t+1|t),\ \ldots,\ u_L(t+N_p|t), \tag{3.19}$$
$$v_L(t|t),\ v_L(t+1|t),\ \ldots,\ v_L(t+N_p+1|t). \tag{3.20}$$

The first control input $u_L(t|t)$ of the sequence (3.19) is then implemented on the vehicle, and the MPC problem is solved again at the next time step. Furthermore, the computed velocity trajectory in (3.20) is sent to the other platooning vehicles at each time step via V2V communication as a velocity forecast, as discussed in Section 3.2.

Follower vehicle MPC

The goal of each follower vehicle is to maintain a desired distance s_i^{des} to the leader vehicle, while also maintaining a minimum safety distance d_{min} to the front vehicle at all times. The MPC problem to be solved is defined as follows

$$\min_{u_i(\cdot|t)} \quad J_i = \sum_{k=t}^{t+N_p+1} (s_i(k|t) - s_i^{des})^2 \tag{3.21a}$$
$$+ \sum_{k=t}^{t+N_p} u_i(k|t)^T R u_i(k|t) \tag{3.21b}$$
$$+ \alpha \sum_{k=t}^{t+N_p-1} \|u_i(k+1|t) - u_i(k|t)\|^2 \tag{3.21c}$$
$$\text{s.t.} \quad x_i(k+1|t) = \tag{3.21d}$$
$$A_i x_i(k|t) + B_i u_i(k|t) + E_i \hat{w}_i(k),$$
$$v_{\min} \leq v_i(k|t) \leq v_{\max}, \tag{3.21e}$$
$$d_{\min} \leq h_i(k|t), \tag{3.21f}$$
$$0 \leq T_i^a(k|t) \leq T_{\max}^a, \tag{3.21g}$$
$$0 \leq T_i^{a,ref}(k|t) \leq T_{\max}^a, \tag{3.21h}$$
$$0 \leq T_i^b(k|t) \leq T_{\max}^b, \tag{3.21i}$$
$$x_i(t|t) = \hat{x}_i(t), \tag{3.21j}$$
$$\forall k = t, \ldots, t+N_p,$$
$$\begin{bmatrix} h_i(t+N_p|t) \\ v_i(t+N_p|t) \end{bmatrix} \in C_F(\hat{v}_{i-1}(t+N_p)), \tag{3.21k}$$

where the notation used is the same as in (3.16). The follower vehicle objective function J_i penalizes deviations from the desired distance to the leader vehicle, given by

$$s_i^{\text{des}} := d^{\text{des}} \cdot i, \tag{3.22}$$

where d^{des} is a design parameter. Furthermore, we also include penalties on input (3.21b) and jerk (3.21c). Similar to the leader, these penalties have to be adjusted carefully to balance performance and passenger comfort. Furthermore, we note that constraints (3.21e) and (3.21k) are imposed with respect to the front (platooning) vehicle only, since safety tasks regarding an upcoming intersection are handled by the platoon leader (the terminal constraint (3.21k) will be discussed further in the next section).

Similar to the leader, at each time step the follower vehicle solves its MPC problem and obtains an optimal control input sequence and velocity trajectory. We apply the first control input of the sequence, and the computed velocity trajectory is broadcast to the platoon via V2V communication. Hence, since velocity forecasts (3.11) are received by all follower vehicles via V2V communication, we use the following disturbance preview for MPC:

$$\hat{w}_i(k) := [\hat{v}_L(k);\ \hat{v}_{i-1}(k)] = [v_L(k|t);\ v_{i-1}(k|t)], \quad k = t, \dots, t + N_p, \tag{3.23}$$

where the planned velocity trajectories $v_L(k|t)$ and $v_{i-1}(k|t)$ were computed by the leader and front vehicle when they solved their respective MPC problems.

Remark 1. *Since we use the full velocity forecast as a disturbance preview in* (3.23)*, a natural question that arises is whether or not these predictions are reliable. To address this question, in [61] we defined the trust horizon F, which allows us to adjust how much of the velocity forecasts are used. For a trust horizon of F, time steps t through $t + F$ of all velocity forecasts are used. After time step $t + F$ the front vehicle is assumed to decelerate at the maximum rate until coming to a stop, and therefore the terminal constraint* (3.21k) *is imposed at time step $t + F$. This is in contrast to the approach in this chapter, where we assume the front (platooning) vehicle will fully realize the trajectory in its velocity forecast as in* (3.23)*, corresponding to $F = N_p$. Doing so introduces some risk to the follower vehicles; however, this is necessary to achieve a reasonable increase in traffic throughput with vehicle platooning, as shown in our previous work [61].*

3.4 Safety Constraints and MPC Solution

We now discuss how we formally ensure safety in an urban traffic setting. First, in Section 3.4 we describe the set of safe states for a vehicle in relation to the two primary obstacles it can encounter in an urban setting: another vehicle ahead of it, and an upcoming intersection. Furthermore, we show that at each time instant the vehicle needs to consider only one of these obstacles, which we refer to as the *priority obstacle*, thereby simplifying the task of ensuring safety. Next, in Section 3.4 we discuss how we use the safe sets from Section 3.4 in our MPC problems, as well as how we efficiently solve the MPC problems at runtime.

Safe States and Priority Obstacle

Consider an ego vehicle (representing either a platoon leader or follower here), a front vehicle ahead of it, and an upcoming intersection. Throughout the section, we let $a(t)$ and $a_F(t)$ be the accelerations of the ego and front vehicles, respectively, so that the vehicle dynamics become

$$\begin{aligned} \dot{h}(t) &= v_F(t) - v(t), \\ \dot{d}^{TL}(t) &= -v(t), \\ \dot{v}_F(t) &= a_F(t), \\ \dot{v}(t) &= a(t), \end{aligned} \tag{3.24}$$

where $h(t)$ is the headway of the ego vehicle, $d^{TL}(t)$ is the distance from the ego vehicle to the upcoming traffic light stop bar, and $v_F(t)$ and $v(t)$ are the velocities of the front and ego vehicles, respectively. Since we observed no actuation delay while braking during experimentation, it is sufficient to use (3.24) in place of (3.1) for the analysis here.

We first assume that only a front vehicle is present, and define safety for the ego vehicle with respect to the front vehicle as

$$h(t) \geq d_{\min}, \quad t \geq 0. \tag{3.25}$$

To enforce (3.25), the ego vehicle must ensure it can maintain a minimum safety distance $d_{\min}$ if the front vehicle applies the maximum braking force until coming to a stop. We formalize this requirement in the following Proposition:

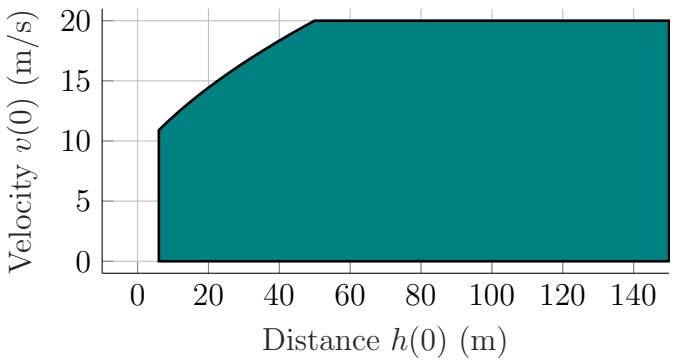
Proposition 3. *Consider the vehicle dynamics given in* (3.24)*. Let* $a_{min,brake}$, $a_{max,brake} \in \mathbb{R}_{>0}$, *and* $a_{min,brake} \leq a_{max,brake}$. *Suppose the accelerations* $a_F(t)$ *and* $a(t)$ *satisfy*

$$a_F(t) = \begin{cases} -a_{max,brake}, & t \in [0, t_F^s], \\ 0, & t > t_F^s, \end{cases} \tag{3.26}$$

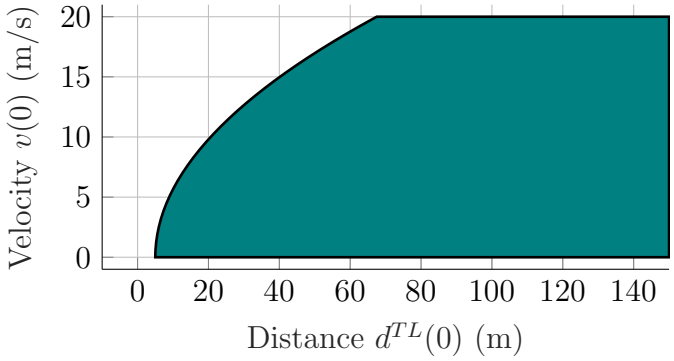
$$a(t) = \begin{cases} -a_{min,brake}, & t \in [0, t^s], \\ 0, & t > t^s, \end{cases} \tag{3.27}$$

where $t_F^s := v_F(0)/a_{max,brake}$ *and* $t^s := v_L(0)/a_{min,brake}$ *are the first time instants in seconds such that* $v_F(t_F^s) = 0$ *and* $v(t^s) = 0$, *respectively. Then,* (3.25) *will hold if* $[h(0); v(0)] \in \mathcal{C}_F(v_F(0))$, *where*

$$\mathcal{C}_F(v_F(0)) := \left\{ \begin{bmatrix} h(0) \\ v(0) \end{bmatrix} : \begin{array}{l} h(0) \geq \dfrac{v(0)^2}{2a_{min,brake}} - \dfrac{v_F(0)^2}{2a_{max,brake}} + d_{\min}, \\ h(0) \geq d_{\min}, \quad v(0) \geq 0 \end{array} \right\} \tag{3.28}$$



(a) $\mathcal{C}_F(v_F(0))$ for $v_F(0) = 14$m/s and $d_{\min} = 6$m.



(b) $\mathcal{C}_{TL}$ for $d_{\min} = 5$m.

Figure 3.6. In 3.6a and 3.6b we plot the terminal sets (3.28) and (3.30) for the front vehicle and upcoming intersection, respectively. For computing the sets, we use $a_{\text{min,brake}} = 3.2$ m/s^2 and $a_{\text{max,brake}} = 5.0912$ m/s^2.

for $v_F(0) \in \mathbb{R}_{\geq 0}$. For a proof we refer to [67], Lemma 1 (see also [31, 55]). We note that in addition to $v_F(0)$, the set $\mathcal{C}_F(v_F(0))$ also depends on $a_{\text{min,brake}}$, $a_{\text{max,brake}}$, $d_{\min} \in \mathbb{R}_{>0}$. A plot of $\mathcal{C}_F$ is given in Figure 3.6a.

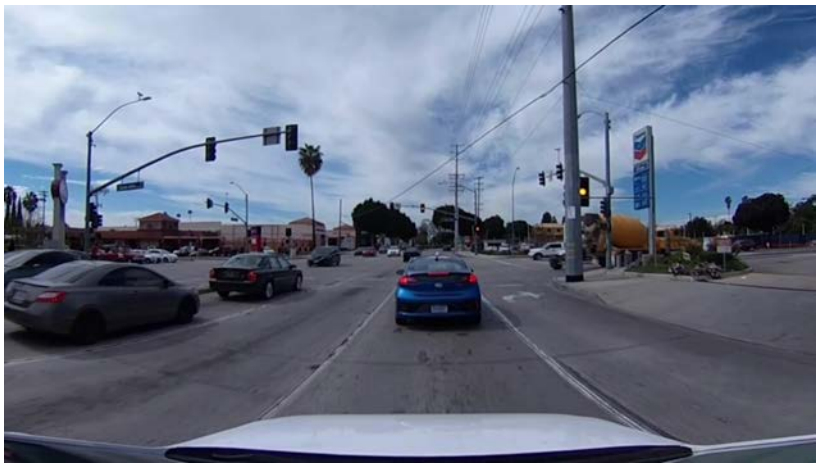
Next, we suppose that only an upcoming intersection requiring a stop is present. In this case, the ego vehicle must ensure it can make a complete stop and leave a distance of $d_{\min}$ to the intersection stop bar. Formally, we require that if the ego vehicle decelerates until coming to a stop as in (3.27), then the following will hold

$$d^{TL}(t) \geq d_{\min}, \quad t \geq 0. \tag{3.29}$$

We note that when the light cycles to green, this constraints is relaxed and the platoon is allowed to proceed. Similar to Proposition 5, we can show that (3.29) holds if the ego vehicle decelerates as in (3.27) and $[d^{TL}(0);\ v(0)] \in \mathcal{C}_{TL}$, where

$$\mathcal{C}_{TL} := \left\{ \begin{bmatrix} d^{TL}(0) \\ v(0) \end{bmatrix} :\ d^{TL}(0) \geq \frac{v(0)^2}{2a_{\text{min,brake}}} + d_{\min}, \quad v(0) \ \geq 0 \right\}. \tag{3.30}$$

(a) Truck has priority.



(b) Intersection has priority.

Figure 3.7. View from the middle platooning vehicle as it approaches an intersection during our demonstration in Arcadia, CA. In Figure 3.7a there is a slow-moving truck attempting to turn right ahead of the leader vehicle. Since the truck takes priority over the intersection at this point, the platoon is forced to slow down. In Figure 3.7b the truck completes the right turn and priority shifts to the intersection.

A plot of $\mathcal{C}_{TL}$ is given in Figure 3.6b.

Now, we suppose that both a front vehicle and an upcoming intersection requiring a stop are present simultaneously. In this scenario, we require that if the front and ego vehicle (representing the platoon leader here) decelerate until coming to a stop as in (3.26) and (3.27), then *both* (3.25) and (3.29) will hold. To determine which obstacle is prioritized, the ego vehicle can check if the following condition holds:

$$h(0) + \frac{v_F(0)^2}{2a_{\max,\text{brake}}} \leq d^{TL}(0). \tag{3.31}$$

If (3.31) holds then the front vehicle is capable of stopping in front of the intersection stop line, and therefore must be prioritized. If (3.31) does not hold then the upcoming intersection is prioritized (see Figure 3.7 for an illustration). We summarize this idea in the following Proposition, which follows directly from the definitions of $\mathcal{C}_F$ and $\mathcal{C}_{TL}$.

Proposition 4. *Assume $h(0) \geq d_{\min}$. If (3.31) does not hold, then $[d^{TL}(0);\ v(0)] \in \mathcal{C}_{TL}$ implies $[h(0);\ v(0)] \in \mathcal{C}_F(v_F(0))$. Otherwise, if (3.31) holds, then $[h(0);\ v(0)] \in \mathcal{C}_F(v_F(0))$ implies $[d^{TL}(0);\ v(0)] \in \mathcal{C}_{TL}$.*

Based on Proposition 4, we conclude that for the leader vehicle MPC problem discussed in the previous section, it is sufficient to impose a terminal constraint with respect to only the priority obstacle. This is beneficial for efficiently solving the MPC problems at runtime, as discussed further in the next section.

Remark 2. *In the above discussion we assumed $d_{\min}$ is the same for both the front vehicle and the intersection, whereas in our experiments we used slightly different values of $d_{\min}$ for each. Although this is beneficial for passenger comfort, there is one drawback to this adjustment: in corner cases where priority between the two obstacles can easily switch, we may only satisfy (3.25) and (3.29) for the minimum of these two values, i.e. for $d_{\min} := \min\{d_{\min,F},\ d_{\min,TL}\}$, where $d_{\min,F}$ and $d_{\min,TL}$ are the unique minimum distance values used for the front vehicle and intersection, respectively. We ensured, however, that this minimum safety margin is still sufficient for testing purposes. Furthermore, in normal traffic conditions the priority between obstacles is clear (usually, the front vehicle is clearly stopping at the intersection, or clearly passing through it).*

Remark 3. *If an upcoming intersection is not present (or does not require a stop), then the front vehicle is prioritized if one is present. This allows, for example, the platoon to pass through a green light if it is safe to do so. Similarly, if only a front vehicle is present then it is prioritized. If neither obstacle is present, then no obstacle-related constraints are imposed on the leader.*

Terminal Constraints and MPC Solution

We now connect the discussion in the previous section to terminal constraints (3.16k) and (3.21k). For the follower vehicles, the primary safety task is to maintain a minimum distance to the front (platooning) vehicle. Therefore, the terminal constraint (3.21k) is imposed with respect to the front vehicle only. For the leader vehicle, the primary safety tasks are to stop at an upcoming intersection when necessary, and to maintain a minimum distance to the front (non-platooning) vehicle. Based on the discussion in Section 3.4, this is accomplished by imposing the terminal constraint (3.16k) with respect to the priority obstacle. To this end, we define

$$d_L^*(t+k|t) := \begin{cases} h_L(t+k|t), & \text{if } \hat{x}_L(t) \text{ and } \hat{v}_F(t) \text{ satisfy (3.31)}, \\ d_L^{TL}(t+k|t), & \text{otherwise}, \end{cases} \tag{3.32}$$

as the planned distance from the leader to the priority obstacle at time step k, computed at time step t, and

$$\mathcal{C}(\hat{x}_L(t), \hat{v}_F(t), \hat{v}_F(t+N_p)) := \begin{cases} \mathcal{C}_F(\hat{v}_F(t+N_p)), & \hat{x}_L(t) \text{ and } \hat{v}_F(t) \text{ satisfy (3.31)}, \\ \mathcal{C}_{TL}, & \text{otherwise}, \end{cases} \tag{3.33}$$

as the terminal set with respect to the priority obstacle. We note that the priority obstacle will be the same throughout the MPC planning horizon, since (3.31) checks whether the front vehicle can stop before the intersection stop bar if it decelerates at the rate $a_{\max,\text{brake}}$, which is its assumed behavior in the leader MPC problem in (3.17).

To solve the leader and follower vehicle MPC problems at runtime we use the tool CVXGEN [36], which allows one to generate C code for solving a custom quadratic program (QP) reliably and efficiently. Since CVXGEN can only be used for moderately-sized QPs, it is beneficial to impose terminal constraint (3.16k) with respect to only the priority obstacle, as imposing a terminal constraint with respect to both obstacles would create additional (redundant) constraints. Furthermore, since our MPC problems must be represented as QPs with linear constraints, the sets $\mathcal{C}_F$ and $\mathcal{C}_{TL}$ discussed in the previous section cannot be directly encoded into our MPC problems. Instead, we use a procedure from [30] to compute polyhedral constraint sets to be used in place of $\mathcal{C}_F$ and $\mathcal{C}_{TL}$. In particular, we compute a collection of sets $\mathcal{C}_F(v_F(0))$ to be used for $v_F(0) \in [v_{\min}, v_{\max}]$. This collection of sets is computed offline, and the proper set is selected during runtime to be used for MPC (for more details, we refer the reader to [61]).

Since it is important to avoid infeasibility of the MPC problems during experimentation, all constraints in each problem (except for the vehicle dynamics constraints) are converted to soft constraints. This means that for a hard constraint such as $Gx \leq h$, where $x \in \mathbb{R}^n$, $G \in \mathbb{R}^{m \times n}$, and $h \in \mathbb{R}^m$, we instead add the term $\lambda \mathbb{1}^T (Gx - h)_+$ to the objective function, where $\lambda \in \mathbb{R}_{>0}$, $\mathbb{1} \in \mathbb{R}^m$ is the vector of all 1's, and y_+ for $y \in \mathbb{R}^m$ indicates that we are thresholding each element of y so that $y_+ \in \mathbb{R}^m_{\geq 0}$ (see [12]).

3.5 Simulation Results

We now present results from our simulation tool developed in MATLAB, which enabled us to validate the platooning software prior to conducting real-world experiments. In particular, the tool is useful to confirm that the platoon preserves safety even when it encounters traffic lights and other non-platooning vehicles, using the approach in Sections 3.2 and 3.4. Furthermore, we are able to estimate the potential gains in traffic throughput at intersections, using a metric from [61].

Urban Stop and Go Scenario

We use our tool to simulate the vehicle platoon travelling along an arterial roadway with moderate traffic. In particular, our goal here is to imitate the conditions we will encounter

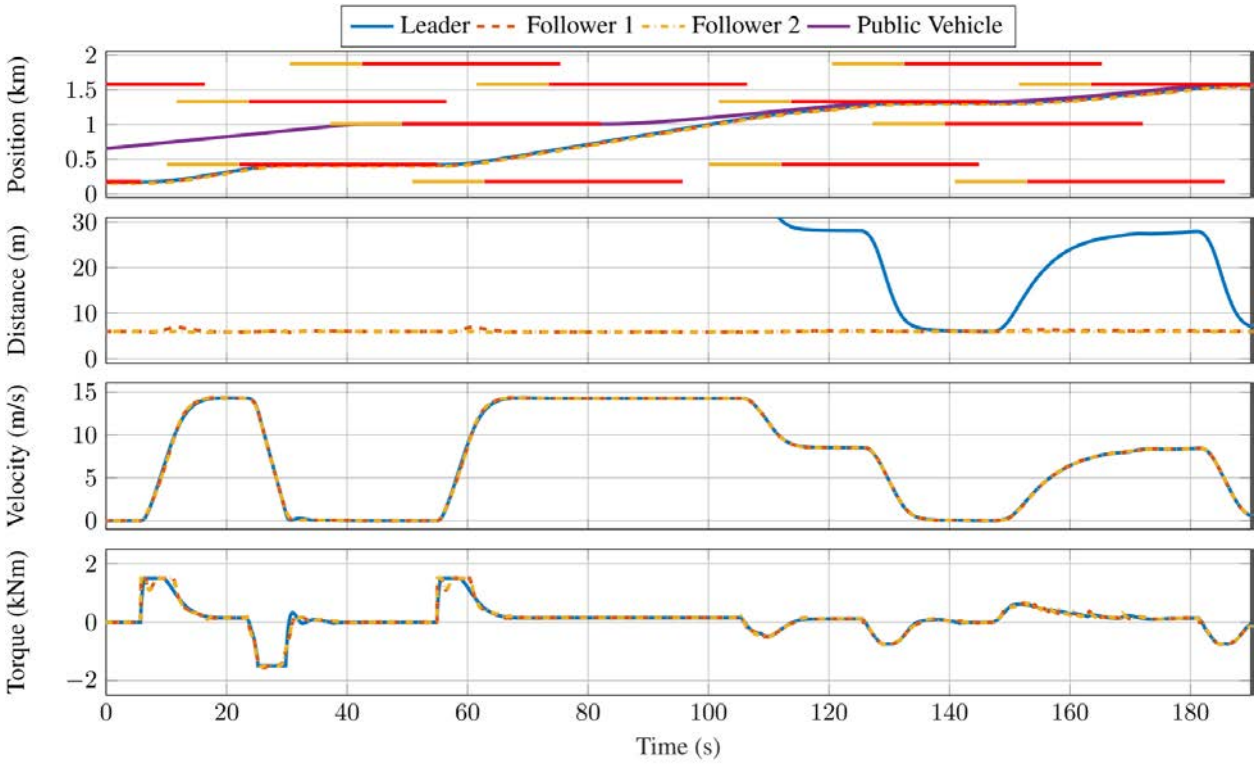


Figure 3.8. Simulation results for an urban traffic scenario with a non-platooning lead vehicle and multiple signalized intersections. In the top plot, we show the position of all simulated vehicles (including the public vehicle which is not platooning), as well as the position of each intersection which has either a yellow or red phase. In the bottom three plots, we show the inter-vehicle distances (including the distance from the leader to the public vehicle), velocities, and torque commands for the platooning vehicles.

during our field experiments in Arcadia, CA (see Section 3.6). To simulate public vehicles in traffic, we create velocity trajectories in simulation and then replay them so that simulations are repeatable. Taking into account the positions / velocities of the platoon leader and a public vehicle ahead of it, we can send radar signals as an input to the leader controller and observe how the platoon responds. Furthermore, we can also create signalized intersections with the following attributes: position (m), V2I communication range (m), cycle offset (s), red / yellow / green time (s), and cycle length (s). We placed intersections along the simulated arterial road so that the distances between traffic lights are similar to the Arcadia corridor discussed in Section 3.6. All the individual intersections are composed to create a traffic network object which can be queried to determine the nearest upcoming traffic light relative to the platoon leader. As the platoon leader approaches the intersection, we send V2I messages from that traffic light as an input to the leader controller and simulate the platoon response.

The simulation results are shown in Figure 3.8. In particular, we note that the horizontal

yellow and red lines in the top plot represent intersections which have a yellow and red phase at that time instant, respectively. Furthermore, the purple line represents the position of a public vehicle which is not platooning. In the beginning of the simulation, the platoon encounters red lights at the first few intersections, stopping at each. Near the end of the simulation the platoon approaches a (non-platooning) public vehicle which is travelling much more slowly, and the platoon is forced to reduce its speed for the remainder of the simulation. We note that near the end of the simulation, the public vehicle comes to a complete stop at an intersection and as a result the platoon leader also stops, leaving a distance of 6m as desired. As mentioned previously, one of the primary goals of the simulation tool is to verify that the platoon responds appropriately when it encounters other non-platooning vehicles and signalized intersections. Observing the simulation results, we can see that the platoon stops at each intersection when necessary, and that the distance from the leader to the public vehicle stays above 6m at all times as desired. Lastly, we remark that for simulation we did not use the same controller parameters that we did for experimentation, where the parameters were mainly selected to improve passenger comfort.

Estimating Throughput

We now analyze the performance of the vehicle platooning system by estimating intersection throughput. To do so, we recall a performance metric defined in [61]. At time $t = 0$ let the platoon be stopped at the (current) intersection stop bar with no vehicles ahead

$$\begin{aligned}[p_L(0);\ v_L(0)] &= [-d_{\min};\ 0],\\ [p_i(0);\ v_i(0)] &= [-d_{\min} - (L_{\text{veh}} + d^{\text{des}}) \cdot i;\ 0], \quad i = 1, \ldots, N-1,\end{aligned}$$

where L_{veh} is the vehicle length (assumed to be uniformly 4.5 meters for all vehicles), and the intersection stop bar is assumed to be positioned at 0 meters. Suppose at time $t = 0$ the traffic light cycles from red to green, and the platoon immediately starts moving through the intersection. Let $\ell \in \mathbb{R}_{>0}$ be the length of the intersection in meters, and define t_L and t_{N-1} to be the smallest time instants in seconds such that $p_L(t_L) \geq \ell$ and $p_{N-1}(t_{N-1}) \geq \ell$, respectively. We then estimate intersection throughput in vehicles per hour as

$$\text{throughput (vph)} \approx 3600 \cdot \frac{N-1}{t_{N-1} - t_L}. \tag{3.34}$$

Thus, performance is maximized when the platoon 1) accelerates to a high velocity while crossing the intersection, and 2) accurately maintains the desired inter-vehicle gaps while accelerating. We note that for the estimate (3.34) to be accurate, we must consider the length of each vehicle, as opposed to treating each as a point mass.

Throughput analysis of simulation results (as well as the test-track experiments discussed in Section 3.6) is shown in Tables 3.3 and 3.4, where all estimates are obtained via (3.34). In particular, throughput is estimated at the 1st, 2nd, and 4th intersection, located at approximately 0.18 km, 0.43 km, and 1.33 km in the simulation, respectively. In Table 3.3

Table 3.3: Improved Throughput

Simulation Intersection 1	4,336.4 vph
Simulation Intersection 2	4,336.4 vph
Simulation Intersection 4	2,477.8 vph
Test Track Intersection (Virtual)	4,463.4 vph

Table 3.4: Baseline Throughput

Simulation Intersection 1	2149.8 vph
Simulation Intersection 2	2156.9 vph
Simulation Intersection 4	1710.5 vph
Test Track Intersection (Virtual)	2730.7 vph

we show improved levels of throughput achieved using our vehicle platooning system, which are estimated from the simulation run shown in Figure 3.8. In Table 3.4 we show baseline levels of throughput, which are estimated by running the same simulation with the *trust horizon* (discussed in Remark 1) set to $F = 0$. We note that throughput is much lower at the 4th intersection, due to the presence of a slower-moving public vehicle ahead of the platoon. Indeed, in situations like this, the benefit of vehicle platooning in terms of traffic throughput may not be fully realized. We note also that our predictions here are in line with predictions from previous works which utilized simulation. For example, in [32] the authors predict that vehicle platooning could enable a saturation flow rate of 4800 vph per intersection movement.

3.6 Experimental Results

In this section we present the experimental results and evaluate the performance of our platooning controller via the throughput metric from Section 3.5. We discuss the experimental setup in Section 3.6, and in Section 3.6 we present results from preliminary tests on a closed track at the Hyundai-KIA Motors California Proving Grounds in California City, CA. Next, we give an overview of a final platooning demonstration on public roadways in Arcadia, CA in Section 3.6. Links to drone videos of each series of tests are also provided.

Test Vehicles

We use the three test vehicles shown in Figure 3.1, each of which is equipped with a production forward-looking radar and camera that estimate the front vehicle distance, velocity,

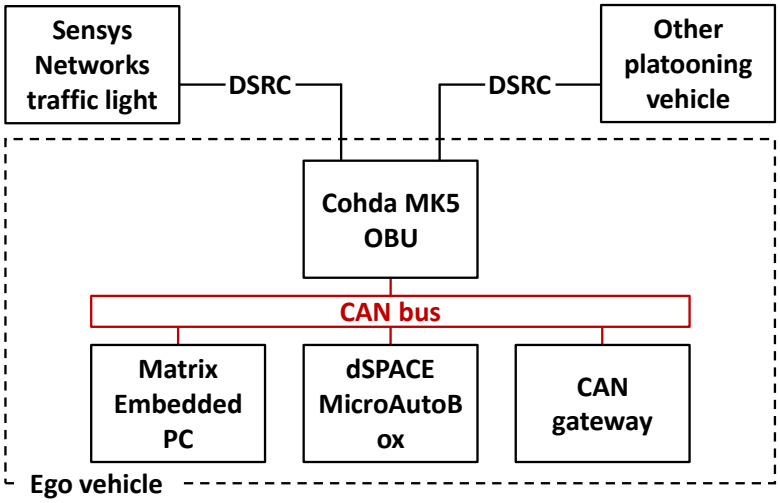


Figure 3.9. Depiction of the on-board hardware setup for the test vehicles. The local CAN bus (in red) connects the computational devices (Matrix embedded PC and dSPACE MicroAutoBox) to the Cohda OBU for DSRC communication. The HCU (CAN gateway) provides an interface between the local CAN bus and the production systems of the test vehicle. Using the local CAN bus and the gateway functionality of the HCU, we can send commands and access measurements to and from the production systems without needing access to proprietary vehicle data.

and acceleration. To enable V2V and V2I communication, we use a Cohda Wireless MK5 V2X on-board unit (OBU), which also has an integrated GPS. The Cohda OBU allows the vehicles to exchange BSMs and custom V2V messages, which include a velocity forecast and other information. This transmitted information allows the third vehicle in the platoon, for instance, to estimate its current distance to the leader vehicle. The Cohda also allows each vehicle to communicate with any nearby traffic lights which are instrumented to broadcast SPaT and MAP messages. Lastly, the controller for each vehicle is implemented on a dSpace MicroAutoBox, and a Matrix embedded PC exchanges information between the Cohda, MicroAutoBox, and the ego vehicle controller area network (CAN bus). The Matrix also runs a state machine which manages the role of each vehicle in the platoon, and is discussed further in Section 3.2. A diagram of the hardware setup is shown in Figure 3.9.

An important hardware consideration for platooning is that of communication latencies. In [61] we discussed how including a time stamp in transmitted messages enables each vehicle to account for V2V communication delays. The idea is to use the time stamp to estimate the delay d in time-steps (with sampling time $\Delta t = 0.1$s), and then to shift the velocity forecast used for MPC by d steps, where we assume the transmitting vehicle will maintain a constant velocity beyond its planned trajectory. For the experimental work presented in this chapter, however, we assume there are no communication delays between vehicles, which is done for two reasons. The first reason is that we have observed that communication latencies are typically small enough to be ignored for our application. The second reason is

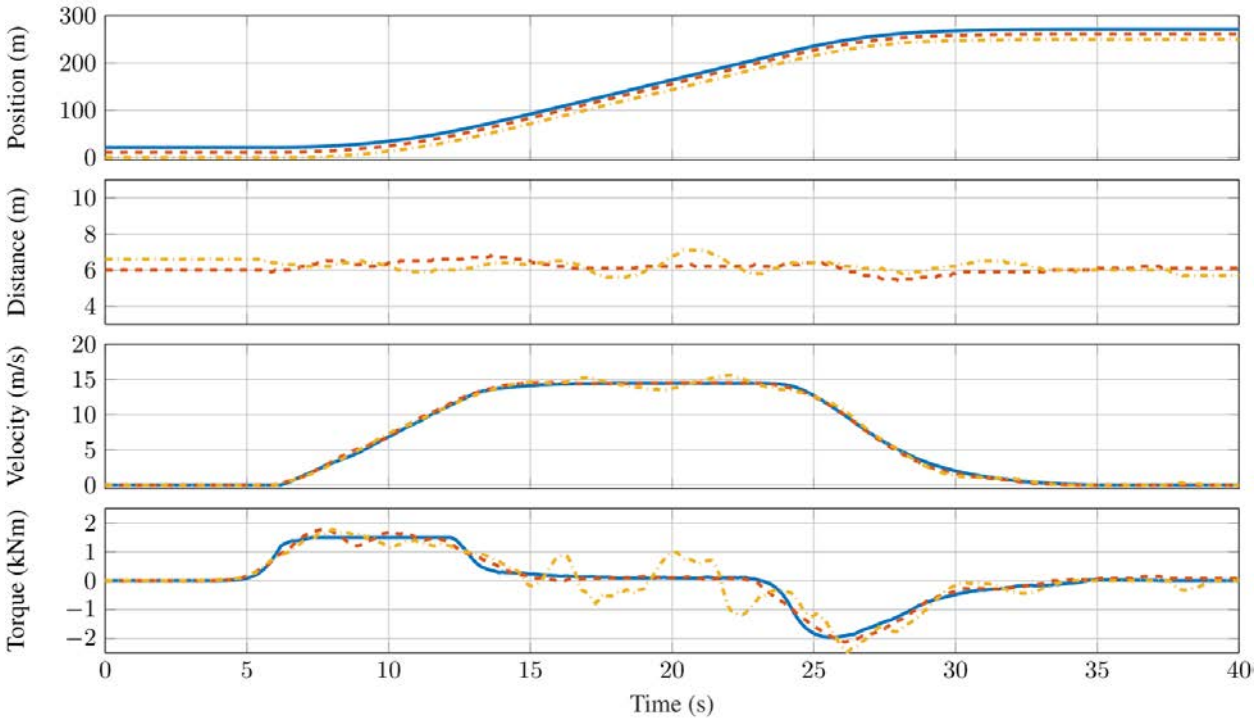


Figure 3.10. Experimental results from the Hyundai-KIA Motors California Proving Grounds with the test vehicles shown in Figure 3.1. Here, we had the platoon track a reference trajectory which was generated via our simulation tool. The position, inter-vehicle distance, velocity, and MPC torque command for each vehicle are shown in each subplot, respectively. The desired distance between vehicles was 6 meters.

that estimating d accurately is challenging in practice. Since the clocks on the test vehicle computers are not synchronized, one must estimate the clock skew between vehicles, which could potentially be time-varying, in order to accurately estimate delays.

Closed track experiments

Preliminary vehicle platooning experiments were conducted on a closed test track at the Hyundai-KIA Motors California Proving Grounds in California City, CA (see Figure 3.1). For all of the tests the leader vehicle does velocity tracking of a predetermined velocity trajectory (meaning v_L^{des} in (3.16a) becomes time-dependent), and the follower vehicles do distance tracking relative to the leader vehicle. The predetermined velocity trajectories used for tracking were either from real velocity data collected during previous experiments, or artificial velocity data generated by our simulation tool. In Figure 3.10 we show experimental results from a test using artificial velocity data which has a step function-like trajectory. For these experiments we used a larger admissible range of the wheel torque for the follower vehicles, as seen in the bottom plot of Figure 3.10. In particular, we note that the torque

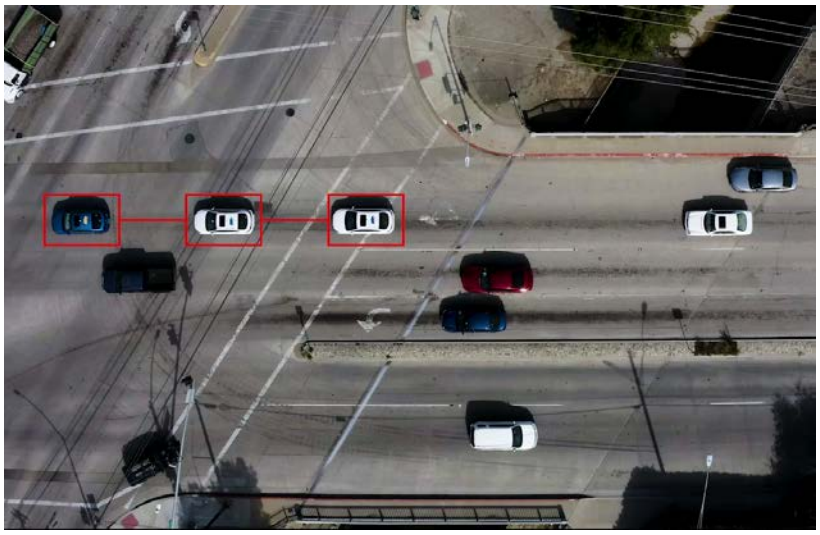


Figure 3.11. Overhead view of the platoon crossing an intersection in Arcadia, CA.

plotted is the *desired* torque, i.e. the output of the MPC algorithm, as opposed to the *measured* torque (estimated by the vehicle). However, the inter-vehicle distances and vehicle velocities are both from on-board measurements (the position data is then obtained offline by integrating the velocity data). We can see that as the platoon accelerates and decelerates, the followers accurately track the desired distance of 6m to the front vehicle - all tracking errors stay below about 1m throughout the experiment. We note, however, that there is slightly larger tracking error (as well as larger variation of the wheel torque command) for the second follower in this experiment. We can mainly attribute this to state estimation error since GPS was used to estimate the distance $s_i(t)$ for all experiments at the California Proving Grounds, as discussed in Section 3.2.

A video of the testing is available online at `https://youtu.be/U-O9iUZElR8`, which includes several test runs with varying levels of the *trust horizon* F (discussed in Remark 1). We note that in test runs with a small trust horizon, for example $F = 10$ (half of the velocity forecast is trusted) or $F = 0$ (none of the velocity forecast is trusted, meaning the vehicles effectively do not use V2V communication), large gaps appear between the platooning vehicles while they are accelerating. This behavior is expected, since using the full velocity forecast relaxes the constraints on following distance so that the follower vehicles can get closer to the vehicle ahead. In the test run shown in Figure 3.10 we used $F = 15$, demonstrating that we are able to get accurate tracking performance when using a large portion of the velocity forecast (elsewhere in the chapter we use $F = N_p = 20$). Similar to Section 3.5, we estimate throughput for the test run shown in Figure 3.10 by treating the platoon as if it begins stopped at an intersection - our estimate is shown in Table 3.3. Furthermore, in Table 3.4 we show a baseline level of throughput computed using data from a test run with $F = 0$. As expected, significantly higher throughput is achieved by utilizing the velocity forecast.

Public Road Demonstration

To demonstrate vehicle platooning in an urban environment with a moderate level of traffic, we conducted further experiments in Arcadia, CA. Our testing area is a 2.45 km long stretch of roadway on Live Oak Ave between S Santa Anita Ave and Peck Rd, and has eight consecutive intersections which are instrumented to send out SPaT and MAP messages for our vehicle platoon to receive. All tests in Arcadia were completed with a 3-vehicle platoon using the same MPC parameters as shown in Table 3.1, with the exception that $v_L^{\text{des}} = 14$ m/s was used here. Footage of our testing is available online: `https://youtu.be/xPYR_xP3FuY`. It captures a few instances where the platoon stops at the stop bar for a red light with no vehicles queued ahead of it. When the light turns green the platoon reacts immediately and moves through the intersection more quickly and compactly than the human-driven vehicles near it, further demonstrating the potential for throughput improvement (see Figure 3.11).

Chapter 4

Safety in Real Driving Scenarios

4.1 Introduction

A commonly studied traffic situation is the vehicle-following scenario. For this scenario, one typically designs a controller for the ego vehicle with the goal of meeting a particular specification, e.g. ensuring a safety constraint on the distance between the ego and lead vehicle is maintained at all times [39]. In some cases, it is beneficial to relax this safety specification slightly - for example, in *vehicle platooning*, where the goal is to have a group of vehicles drive closely together in a tight formation (see [61] and [62]). To this end, in [31] the authors allow a soft impact with bounded relative velocity to occur in a worst-case driving scenario. By doing so, the time required to safely execute a platoon join maneuver, for example, is reduced.

Another common maneuver that a driver must execute is an unprotected left turn. Recent works have studied this scenario due to its complexity; see e.g. [41]. In [70] the authors consider an analogous scenario of highway merging for connected vehicles, where the ego vehicle can merge either ahead of or behind the other vehicle. In each case, the state space is separated into conflict, nonconflict, and uncertain regions, where the boundaries of these regions are dependent on the acceleration capabilities of each vehicle. Similarly, in [13] the authors compute a *capture set*, i.e. the set of states that lead to conflict regardless of input choice. In particular, this computation can be done efficiently if the system has an order preserving property. The authors propose a control map ensuring the capture set is avoided, and the approach is demonstrated on an example where two connected vehicles approach an intersection, and also on an autonomous roundabout scenario in [14].

In this chapter, we apply symbolic control techniques from [28] and [52] to both driving scenarios mentioned above. Symbolic control techniques have multiple advantages - for example, they can handle complex specifications, and can be applied directly to nonlinear systems. In contrast, [31], [70], and [14] ignore nonlinearities in the vehicle dynamics, and [13] uses feedback linearization. Similarly, in [2] the authors only consider nonlinear vehicle dynamics on a restricted input space, otherwise using a linear approximation.

By exploiting the monotonicity of the system dynamics, we reduce the computational complexity of the controller synthesis and implementation [52]. We show how monotonicity makes it possible to deal with complex scenarios, such as the vehicle following scenario with safe impact and left turn scenario, while handling model uncertainty.

The contribution of this work is two-fold. First, we showcase the flexibility of symbolic control techniques by applying them in two realistic driving situations: a vehicle-following scenario in Section 4.3, and an unprotected left turn scenario in Section 4.4, each of which is of independent interest. Second, to deal with the specification in each scenario we construct a non-standard abstraction, in which we introduce new special states to transform the specifications into *lower closed safety specifications* [52]. We also introduce a new construction of the transition relation which ensures monotonicity of the abstraction, where a new partial order has been defined to deal with the special states.

4.2 Monotonicity Concepts

In this section, we overview the monotonicity and symbolic control concepts we will use throughout the chapter.

Partial orders

A partially ordered set $\mathcal{L}$ has an associated binary relation $\leq_{\mathcal{L}}$ where for all $l_1, l_2, l_3 \in \mathcal{L}$, the binary relation satisfies: (i) $l_1 \leq_{\mathcal{L}} l_1$, (ii) if $l_1 \leq_{\mathcal{L}} l_2$ and $l_2 \leq_{\mathcal{L}} l_1$ then $l_1 =_{\mathcal{L}} l_2$ and, (iii) if $l_1 \leq_{\mathcal{L}} l_2$ and $l_2 \leq_{\mathcal{L}} l_3$ then $l_1 \leq_{\mathcal{L}} l_3$. Given a partially ordered set $\mathcal{L}$, for $a \in \mathcal{L}$ the lower closure of the element $a \in \mathcal{L}$ is denoted $\downarrow a$ and defined as $\downarrow a := \{x \in \mathcal{L} : x \leq_{\mathcal{L}} a\}$. The lower closure of a set $A \subseteq \mathcal{L}$ is $\downarrow A := \bigcup_{a \in A} \downarrow a$. A subset $A \subseteq \mathcal{L}$ is said to be lower-closed if $\downarrow A = A$.

Monotone Transition Systems

Below we recall the notion of a transition system [66] and define *monotone* transition systems that preserve a partial order on input and state spaces.

Definition 4.2.1. *A transition system is a tuple $T = (X, X_0, U, \Delta)$, where X is a set of states, $X_0 \subseteq X$ is a set of initial states, U is a set of inputs and $\Delta : X \times U \to X$ is a deterministic transition relation.*

Definition 4.2.2. *A transition system $T = (X, X_0, U, \Delta)$ is said to be input-state monotone if X and U are equipped with partial orders $\leq_X$, $\leq_U$, respectively, and for all $x_1, x_2 \in X$, for all $u_1, u_2 \in U$, with $x_1 \leq_X x_2$ and $u_1 \leq_U u_2$, it follows that $\Delta(x_1, u_1) \leq_X \Delta(x_2, u_2)$.*

Controller Synthesis for Safety Specifications

Maximal safety controller

Given a transition system $T = (X, X_0, U, \Delta)$, a controller for T is a set-valued map $\mathcal{C} : X \rightrightarrows U$ and its domain is defined as $dom(\mathcal{C}) = \{x \in X : \mathcal{C}(x) \neq \emptyset\}$. A safety controller is then defined as:

Definition 4.2.3. *A safety controller $\mathcal{C}$ for the transition system $T = (X, X_0, U, \Delta)$ and the safe set $X^S \subseteq X$ satisfies:*

- $dom(\mathcal{C}) \subseteq X^S$;
- $\forall x \in dom(\mathcal{C})$ *and* $\forall u \in \mathcal{C}(x)$, $\Delta(x, u) \subseteq dom(\mathcal{C})$.

A suitable solution to the safety problem is a controller that enables as many actions as possible. This controller $\mathcal{C}^*$ is said to be a maximal safety controller, in the sense that for any other safety controller and for all $x \in X$, we have $\mathcal{C}(x) \subseteq \mathcal{C}^*(x)$.

Lazy controller synthesis for safety specifications

Consider an input-state monotone transition system $T = (X, X_0, U, \Delta)$ and a safety specification $X^S \subseteq X$. The safety specification X^S is said to be lower closed (respectively, upper closed) if X^S is a lower closed (respectively, upper closed) subset of X. Classical approaches use a fixed-point algorithm [66] for general safety specifications. For upper and lower safety specifications, efficient symbolic abstractions and lazy synthesis approaches have been proposed recently in [28] and [52]. These approaches allow us to compute the maximal safety controller while reducing the computational cost required for the synthesis and implementation of the maximal safety controller. Indeed, in classical approaches [66], one first constructs the entire abstraction for the original system and then uses the pre-computed abstraction to synthesize the controller. In lazy approaches, however, the abstraction and controller synthesis are done in parallel, making it possible to compute only a fragment of the abstraction that is essential for the controller synthesis.

4.3 Vehicle-Following Scenario

In this section, we consider a vehicle-following scenario. We first introduce the vehicle dynamics model that we use and present the control objective. We then use the monotonicity properties of the model to construct a symbolic abstraction and to synthesize a controller.

Monotone Vehicle Dynamics

The vehicle-following model is:

$$\begin{aligned} \dot{h} &= v_L - v, \\ \dot{v} &= f(u, v, \theta), \\ \dot{v}_L &= f(u_L, v_L, \theta_L), \end{aligned} \tag{4.1}$$

where $h \in \mathbb{R}$ is the headway between the vehicles, $v,\ u \in \mathbb{R}$ are the velocity and wheel torque for the ego vehicle, $v_L,\ u_L \in \mathbb{R}$ are the velocity and wheel torque for the lead vehicle, and $\theta,\ \theta_L \in \mathbb{R}^5$ contain modelling parameters. The individual vehicle dynamics evolve according to

$$f(u, v, \theta) := \begin{cases} g(u, v, \theta), & v > 0, \\ \max\left\{g(u, v, \theta),\ 0\right\}, & v = v_{\min}, \\ \min\left\{g(u, v, \theta),\ 0\right\}, & v = v_{\max}, \end{cases} \tag{4.2}$$

where

$$g(u, v, \theta) = \frac{1}{M}\left(\frac{u}{R_w} - F_f\right) \text{ and } F_f = \alpha + \beta v + \gamma v^2 \tag{4.3}$$

give the vehicle's acceleration and frictional force acting on it. We note the vehicle dynamics model ensures both vehicles never exceed their velocity bounds - that is $v(t),\ v_L(t) \in [v_{\min}, v_{\max}]$ for $t \geq 0$. Furthermore, (4.1) - (4.3) contain the following modelling parameters: $M > 0$ is the vehicle mass, $R_w > 0$ is the wheel radius, and $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are friction coefficients. We collect all modelling parameters in $\theta := [M;\ R_w;\ \alpha;\ \beta;\ \gamma]$ for the ego vehicle and, similarly, in θ_L for the lead vehicle (which may have different modelling parameters). For each vehicle, the values of the modelling parameters are unknown and are only assumed to lie within a bounded interval of values, where the interval bounds are known. For example, we assume $\gamma, \gamma_L \in [\gamma_{\min}, \gamma_{\max}]$, where $\gamma_{\min} > 0$ and $\gamma_{\max} > 0$ are known.

Next, we define the state of (4.1) as $x(t) := [h(t);\ v(t);\ v_L(t)]$, the input $u(t)$, and the disturbance $w(t) := u_L(t)$, each of which are assumed to lie within a corresponding constraint set at all times

$$\begin{aligned} X &:= \left\{x : v_{min} \leq v \text{ and } v_{L,min} \leq v_L \leq v_{L,max}\right\}, \\ U &:= \left\{u : u_{min} \leq u \leq u_{max}\right\}, \\ W &:= \left\{w : w_{min} \leq u_L \leq w_{max}\right\}. \end{aligned} \tag{4.4}$$

The solution of the vehicle model (4.1) at time $t > 0$, from an initial condition $x_0 \in X$, under a control input $u : [0, t] \to U$, a disturbance input $w : [0, t] \to W$ and a vector of unknown parameters $[\theta; \theta_L]$ is denoted $\Phi(t; x_0, u, w, [\theta; \theta_L])$. Hence, under the same conditions, the reachable set over the time interval $[0, t]$ reads $\Phi([0, t]; x_0, u, w, [\theta; \theta_L])$.

Finally, we equip the state, input and disturbance spaces of the model in (4.1) with the partial orders

$$\begin{aligned}(x_1 \leq_X x_2) &\iff \big[(h_1 \geq h_2) \wedge (v_1 \leq v_2) \wedge (v_{L,1} \geq v_{L,2})\big] \\ (u_1 \leq_U u_2) &\iff (u_1 \leq u_2), \qquad (4.5) \\ (w_1 \leq_W w_2) &\iff (u_{L,1} \geq u_{L,2}) \qquad (4.6)\end{aligned}$$

where $\leq$ is the usual partial order on $\mathbb{R}$. With the partial order defined above, it is easy to verify that the dynamics in (4.1) are monotone [4]. This property states that for $x_1 \leq_X x_2$, $u_1 \leq_U u_2$, and $w_1 \leq_W w_2$, we have for $t \geq 0$:

$$\Phi(t; x_1, u_1, w_1, [\theta; \theta_L]) \leq \Phi(t; x_2, u_2, w_2, [\theta; \theta_L]). \qquad (4.7)$$

Control Objective

We now discuss the control objective we want the ego vehicle to satisfy. Typically, one would require

$$x(t) \in X \cap \mathcal{H},\ \mathcal{H} := \big\{x : h_{min} < h \text{ and } v \leq v_{max}\big\}, \qquad (4.8)$$

to hold for $t \geq 0$. From the definition of the set of constraints X in (4.4), the condition $x(t) \in X$, for all $t \geq 0$ is already satisfied. The objective here is to synthesize a controller for the ego vehicle ensuring that $x(t) \in \mathcal{H}$, for all $t \geq 0$, which, as discussed in Section 4.2, is a lower closed safety specification with respect to the partial order (4.5). In words, (4.8) means the ego vehicle must ensure it never collides with the lead vehicle. Moreover, the ego vehicle velocity must be bounded by the maximum velocity v_{max}, while assuming the lead vehicle velocity is also bounded by v_{max}.

Next, we define the set of states for which a *soft impact* has occurred [31]:

$$S := \big\{x : h \leq h_{min} \text{ and } v - v_L \leq v_{\text{allow}}\big\}. \qquad (4.9)$$

For our modified safety specification, we allow a soft impact to occur in a worst-case driving scenario, but never an unsafe impact - that is, one that violates (4.9). This is beneficial since it relaxes the restrictive constraint (4.8) on the ego vehicle, allowing it to follow the lead vehicle more closely, for example. We now formally state the control objective considered in this section:

Problem 1. *Given the model of the vehicle-following scenario in (4.1), synthesize a sampled-data controller $\mathcal{C} : X \rightrightarrows U$ such that either* (4.8) *holds or the following holds:*

$$\exists t_0 \geq 0 \text{ s.t. } x(t_0) \in S \text{ and } x(t) \in X \cap \mathcal{H} \text{ for } t \in [0, t_0).$$

The control objective described above is in the same spirit of a reach-avoid specification, in the sense that the system state must either remain in the set $X \cap \mathcal{H}$ for all time (avoiding an unsafe impact), or eventually reach the set S. We emphasize that the set S will only be reached in a worst-case scenario - for example, if the ego vehicle fails to satisfy (4.8) because the lead vehicle applied harsh brakes.

Synthesis using the symbolic approach

In this section, we design a control law $\mathcal{C} : X \rightrightarrows U$ which is a solution to Problem 1 using the symbolic control approach [66] that relies on the use of symbolic models, which are discrete abstractions of continuous dynamics.

Symbolic abstraction

An abstraction Σ^a for the vehicle model in (4.1) is a transition system $\Sigma^a := (X^a, X_0^a, U^a, \Delta^a)$, where X^a, X_0^a and U^a are finite (symbolic) sets of states and control inputs respectively, while $\Delta : X^a \times U^a \to X^a$ is a transition relation. For constructing the symbolic sets and in view of the control objective defined in Problem 1, the set X of constraints on the state-space defined in (4.4) is decomposed into three regions: an impact-free region, represented by the set $\mathcal{H}$ in (4.8), a region of soft impact, represented by the set S in (4.9), and a remaining unsafe region given by $X \setminus (\mathcal{H} \cup S)$. Each of these regions is represented in symbolic form as follows:

- We discretize the impact-free region $\mathcal{H}$ into $N \geq 1$ half-open intervals $q_i = (\underline{q}_i; \overline{q}_i]$ using a finite partition. Since $\mathcal{H}$ is unbounded, we follow the approach in [51, Section V-B-3] which uses bounded and unbounded intervals to construct the partition. The states of these regions are represented by the green states in Figure 4.1.
- We use a unique state q_{sink} to model the safe-impact region S, represented by the blue state in Figure 4.1.
- We use a unique state q_{unsafe} to represent the unsafe region $X \setminus (\mathcal{H} \cup S)$; see the red state in Figure 4.1.

The symbolic set X^a consists then of $N+2$ states $X^a := \{q_i : i = 1, \ldots, N\} \cup \{q_{sink}, q_{unsafe}\}$. The set of initial conditions corresponds to $X_0^a = \{q_i : i = 1, \ldots, N\}$. Moreover, we discretize the set of inputs U into $M \geq 2$ values, with the discrete input set given by $U^a := \{u_j : j = 1, \ldots, M\}$.

Assuming the controller to be designed is implemented by a microprocessor with a sampling time $\tau > 0$, the transition relation $\Delta : X^a \times U^a \rightrightarrows X^a$ can be defined as follows. For any $q, q' \in X^a, u \in U^a$, $q' = \Delta(q, u)$ if and only if one of the following scenarios holds:

(i) For $q, q' \in X_0^a$ and $u \in U^a$, $q' = \Delta(q, u)$ if and only if $\Phi([0, \tau]; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \subseteq \mathcal{H}$ and $\Phi(\tau; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \in q'$;

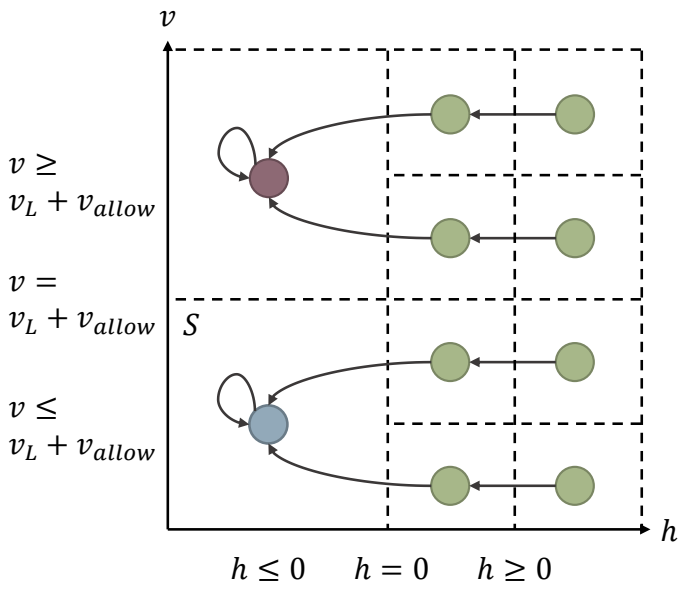


Figure 4.1. The state space is divided into three areas: the area corresponding to set S (bottom left cell), the area corresponding to unsafe impacts (top left cell), and the area where no impact has occurred (right cells).

(ii) For $q \in X_0^a \cup \{q_{sink}\}$ and $u \in U^a$, $q_{sink} = \Delta(q, u)$ if and only if $q = q_{sink}$ or there exists $s \in [0, \tau]$ such that $\Phi(s; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \in S$ and $\Phi([0, \tau]; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \subseteq \mathcal{H} \cup S$;

(iii) For $q \in X_0^a \cup \{q_{unsafe}\}$ and $u \in U^a$, $q_{unsafe} = \Delta(q, u)$ if and only if $q = q_{unsafe}$ or $\Phi([0, \tau]; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \cap (X \setminus (\mathcal{H} \cup S)) \neq \emptyset$.

In each scenario, for each transition the vector of unknown parameters $[\theta_{\max}; \theta_{L,\min}]$ are selected to maximize (minimize) the acceleration of the ego (lead) vehicle during the sampling period, depending on the control input applied. For example, intuitively we want to underestimate how much air drag will help the ego vehicle avoid a collision, and overestimate how much it will help the lead vehicle cause one. This represents the worst-case values for the modelling parameters in (4.1). Furthermore, $w_{\min}$ is the maximum braking torque for the lead vehicle. For the construction of the transition relation, the first scenario is used to represent the impact-free case where the trajectory of the vehicles remains in the set $X \cap \mathcal{H}$. The second scenario represents the case of soft impact. Moreover, in this second scenario we added a self-loop to the sink state q_{sink} to transform the reach-avoid specification in Problem 1 to a safety problem. Finally, the last scenario is used to represent the fact that the trajectory of the vehicle is unsafe, in the sense that an unsafe impact violating (4.9) occurs.

Remark 4. *In view of Problem 1, a transition to q_{sink} should be created from $q \in X^a$ and $u \in U^a$ if and only if $q = q_{sink}$ or there exists $s \in [0, \tau]$ such that $\Phi(s; \overline{q}, u, w_{\min}, [\theta_{\max}; \theta_{L,\min}]) \in S$*

and also $\Phi([0,s];\overline{q},u,w_{\min},[\theta_{\max};\theta_{L,\min}]) \subseteq \mathcal{H} \cup S$. *The latter condition is replaced in (ii) by* $\Phi([0,\tau];\overline{q},u,w_{\min},[\theta_{\max};\theta_{L,\min}]) \subseteq \mathcal{H} \cup S$ *in order to preserve the monotonicity property of the transition system, at the cost of a small additional conservatism.*

Abstract control objective

Using such construction of the symbolic abstraction Σ^a, the concrete control objective in Problem 1 can be transformed to the following abstract control objective:

Problem 2. *Given the abstraction* Σ^a *of the vehicle-following model in (4.1), synthesize the maximal discrete safety controller* $\mathcal{D} : X^a \rightrightarrows U^a$ *keeping the trajectories of the transition system* Σ^a *in the set* $X_0^a \cup \{q_{sink}\}$.

To synthesize the controller $\mathcal{D}$, we rely on the use of the monotonicity concepts introduced in Section 4.2. We first have the following result, characterizing the structural properties of the abstraction Σ^a and the considered specification.

Proposition 5. *The transition system* $\Sigma^a := (X^a, X_0^a, U^a, \Delta^a)$ *defined above is an input-state monotone transition system and the safety specification* $X_0^a \cap \{q_{sink}\}$ *is lower closed.*

Proof. We start by defining the partial order for the discrete state and input spaces. We define a partial order $\leq_{X^a}$ over the set of discrete states X^a as follows: for $q_1, q_2 \in X_0^a$, $q_1 \leq_{X^a} q_2$ if and only if $\overline{q}_1 \leq_X \overline{q}_2$. For the special states q_{unsafe} and q_{sink} we have the following: for all $q \in X_0^a$, $q \leq_{X^a} q_{sink} \leq_{X^a} q_{unsafe}$. Moreover, since $U^a \subseteq U$, the partial order $\leq_{U^a}$ on the discrete input space is inherited from $\leq_U$. The fact that the set $X_0^a \cap q_{sink}$ is lower closed follows immediately from the definition of the partial order $\leq_{X^a}$.

Let us show the monotonicity of the transition system Σ^a. Consider $q_1, q_2 \in X^a$, $u_1, u_2 \in U^a$ with $q_1 \leq_{X^a} q_2$ and $u_1 \leq_{U^a} u_2$. We will show that $\Delta(q_1,u_1) \leq \Delta(q_2,u_2)$. From the definition of the monotonicity property in (4.7), we have that $\Phi(\tau;\overline{q}_1,u_1,w_{\min},[\theta_{\max};\theta_{L,\min}]) \leq \Phi(\tau;\overline{q}_2,u_2,w_{\min},[\theta_{\max};\theta_{L,\min}])$. To complete the proof, we distinguish three cases:

- $\Delta(q_1,u_1) \in X_0^a$: In this case, we have two options. If $\Delta(q_2,u_2) \in X_0^a$, then we get directly from (4.7) that $\Delta(q_1,u_1) \leq \Delta(q_2,u_2)$. Otherwise, we have that $\Delta(q_2,u_2) = q_{sink}$ or $\Delta(q_2,u_2) = q_{unsafe}$, which implies from the construction of the partial order $\leq_{X^a}$ above that $\Delta(q_1,u_1) \leq \Delta(q_2,u_2)$.

- $\Delta(q_1,u_1) = q_{sink}$: In this case, we have from (4.7) that $\Delta(q_2,u_2) = q_{sink}$ or $\Delta(q_2,u_2) = q_{unsafe}$, which implies from the construction of the partial order $\leq_{X^a}$ above that $\Delta(q_1,u_1) \leq \Delta(q_2,u_2)$.

- $\Delta(q_1,u_1) = q_{unsafe}$: In this case we have either $q_1 \in X_0^a$ or $q_1 = q_{unsafe}$. If $q_1 \in X_0^a$, we have from the construction of the transition relation Δ and using (4.7) that $\Delta(q_2,u_2) = q_{unsafe}$, which implies that $\Delta(q_1,u_1) \leq \Delta(q_2,u_2)$. Otherwise, if $q_1 = q_{unsafe}$ then $q_2 = q_{unsafe}$ and $\Delta(q_1,u_1) = q_{unsafe} \leq \Delta(q_2,u_2) = q_{unsafe}$.

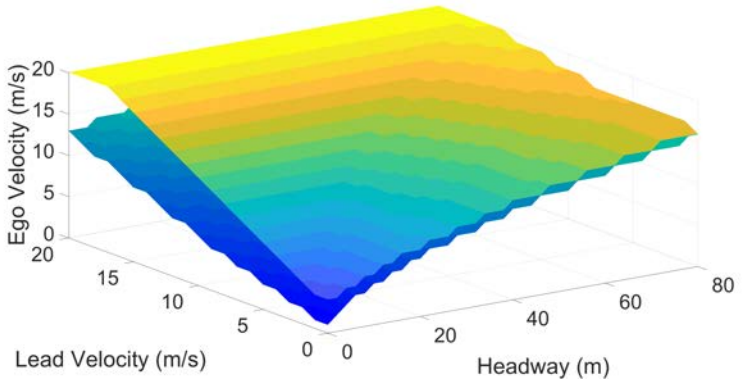


Figure 4.2. Boundary of safe set $Z \subset X$ for the strict (top surface) and relaxed (bottom surface) vehicle-following specification. The safe sets lay below the depicted boundaries.

□

We now have all the ingredients to provide a solution to Problem 1. First, using the lazy controller synthesis approach for input-state upper monotone transition systems and directed safety specification (see Section 4.2) we can construct the maximal abstract safety controller $\mathcal{D} : X^a \rightrightarrows U^a$ for the transition system Σ^a and lower closed safety specification $X_0^a \cup \{q_{sink}\}$, which is indeed a solution to Problem 2. Second, using the construction of the abstraction Σ^a, one can show, similarly to [28], that the abstraction Σ^a is related to the original system in (4.1) by an upper alternating simulation relation[1]. This relation is useful for controller refinement for our lower closed safety specification $X_0^a \cup \{q_{sink}\}$. Based on this relationship, we can refine the abstract controller $\mathcal{D} : X^a \rightrightarrows U^a$ into a concrete controller $\mathcal{C} : X \rightrightarrows U$, providing a solution to Problem 1. In this case, the concrete controller $\mathcal{C}$ can be defined for $x \in X$ as follows: $\mathcal{C}(x) = \mathcal{D}(Q(x))$, where Q is the *quantizer* associated to the abstraction Σ^a and relating the continuous state-space X to the discrete state-space X^a as follows: $Q : X \to X^a$, with $Q(x) = q$ if and only if $x \in q$.

Using the lazy controller synthesis approach, we compute a safe set (that is, the set $Z = dom(\mathcal{C}) \subset X$ where we can enforce the given specification) with respect to both the strict specification (4.8) and the relaxed specification given in Problem 1. The numerical values of the vehicle parameters and the control objective are as follows: $M \in [2000\text{kg}, 2250\text{kg}]$, $R_w \in [0.30\text{m}, 0.35\text{m}]$, $\alpha \in [300, 350]$, $\beta \in [0.10, 0.25]$, $\gamma \in [0.30, 0.65]$, $v_{min} = v_{L,min} = 0\text{m/s}$, $v_{max} = v_{L,max} = 20\text{m/s}$, $u_{min} = -2500\text{Nm}$, $u_{max} = 1200\text{Nm}$, $w_{min} = -1800\text{Nm}$, $w_{max} = 1200\text{Nm}$, $h_{min} = 0\text{m}$, and $v_{allow} = 3\text{m/s}$. Furthermore, we discretized the state and input using the following resolutions: $h_{\text{res}} = 2\text{m}$, $v_{\text{res}} = v_{L,\text{res}} = 1\text{m/s}$, and $T_{\text{res}} = 100\text{Nm}$. As expected, relaxing the safety specification expands the safe set. This allows the vehicles to

[1]While traditional alternating simulation relations [66] impose output equivalence, the upper alternating simulation relation relaxes that condition to an ordering relation. In our case, the upper alternating simulation relation between the abstraction Σ^a and the original system in (4.1) is defined for $(x, q) \in X \times X^a$, with $q = (\underline{q}; \overline{q}]$, as $(x, q) \in \mathcal{R}$ if and only if $x \leq \overline{q}$.

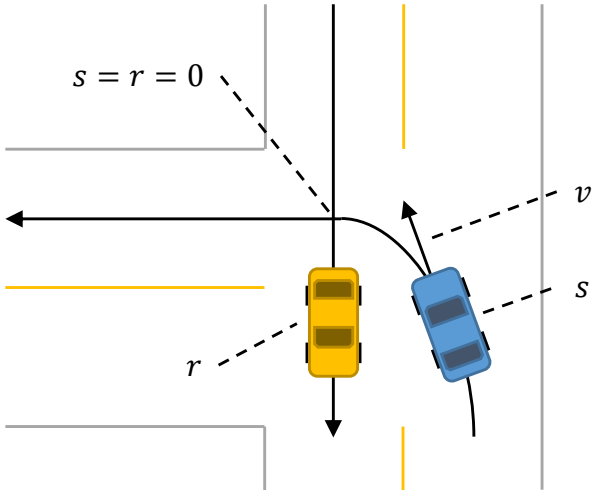


Figure 4.3. Depiction of the states for the ego (blue) and oncoming (yellow) vehicle in the unprotected left turn scenario.

drive more closely together and improve traffic efficiency - for example, in vehicle platooning [62].

4.4 Unprotected Left Turn Scenario

In this section, we compute a safety controller for an unprotected left turn scenario using the approach established in Section 4.3. Indeed, the vehicle dynamics in this scenario are monotone, and collision avoidance only requires the ego vehicle to adjust its velocity along its current path [2].

Monotone Vehicle Dynamics and Control Objective

We model the vehicle dynamics in the unprotected left turn scenario as follows

$$\begin{aligned} \dot{s} &= v, \\ \dot{v} &= f(u, v, \theta), \\ \dot{r} &= v_0, \end{aligned} \tag{4.10}$$

where $s,\ v \in \mathbb{R}$ are the position and velocity of the ego vehicle along its (curved) path, and $r \in \mathbb{R}$ is the position of the oncoming vehicle along its path. The positions s and r increase in the direction of travel, and at the point $s = r = 0$ the vehicle paths cross. Furthermore, $u \in \mathbb{R}$ is the ego vehicle wheel torque, and the ego vehicle dynamics evolve as in (4.1), where $\theta \in \mathbb{R}^5$ includes the modelling parameters from the previous example. All of the modelling parameters in θ are again unknown and only assumed to lie within

bounded intervals. Similarly, the value of $v_0 > 0$ is also uncertain here, and we only assume $v_0 \in [v_{0,\min}, v_{0,\max}]$, where $v_{0,\min} > 0$ and $v_{0,\max} > 0$ are known.

To address the possibility of a collision between the ego and oncoming vehicles, we define a *conflict zone* [22] around this crossing point, and require the two vehicles to never occupy the conflict zone simultaneously. Formally, we define the following set of conflicting states

$$C := \big\{x : |s| \leq \ell \text{ and } |r| \leq \ell\big\}, \tag{4.11}$$

where $\ell > 0$ is an adjustable parameter. To avoid the unsafe set (4.11) at all times, the ego vehicle can either go first and complete its turn before the oncoming vehicle enters the intersection, or wait for the the oncoming vehicle to pass through the intersection first, and then start its turn. For each case, we define a respective goal set

$$G^{\text{wait}} := \big\{x : r > \ell\big\}, \quad G^{\text{go}} := \big\{x : s > \ell\big\}, \tag{4.12}$$

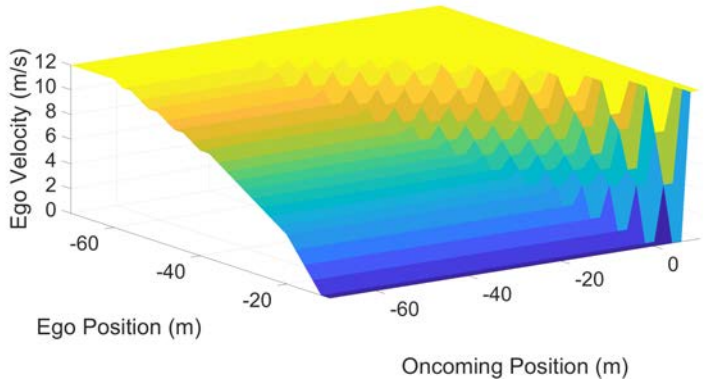
which represents the opposite side of the intersection for each vehicle. Next, we define the following constraint sets for the state and input. For $i \in \{wait,\ go\}$, we have

$$\begin{aligned} X^i &:= \left\{ x : \begin{array}{l} s^i_{\min} \leq s \leq s^i_{\max},\ v^i_{\min} \leq v \leq v^i_{\max}, \\ r^i_{\min} \leq r \leq r^i_{\max} \end{array} \right\}, \\ U &:= \big\{u :\ u_{\min} \leq u \leq u_{\max}\big\}, \end{aligned} \tag{4.13}$$

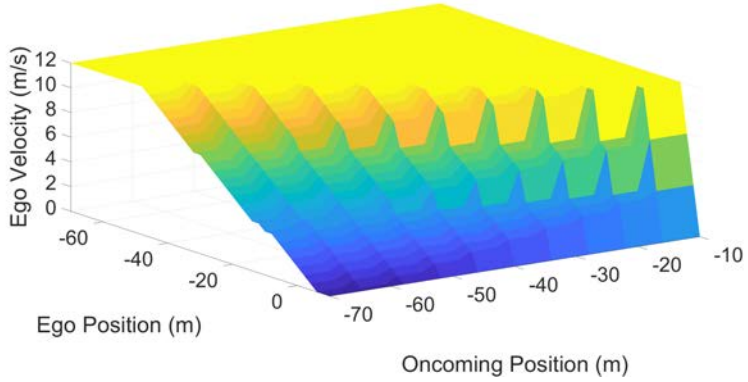
where the bounds on each of the state variables depend on the ego vehicle's strategy for executing the turn. For example, the set X^{go} will exclude states where the oncoming vehicle occupies the intersection, since we want the ego vehicle to go first in this case. With (4.11) - (4.13), we state our control objective.

Problem 3. *Our control objective is to ensure the conflict set is avoided at all times, that is $x(t) \notin C$ for $t \geq 0$, and a goal set is eventually reached, that is $\exists t_0$ s.t. $x(t) \in G^i$ for $t > t_0$ and $x(t) \in X^i$ for $t \in [0, t_0]$, where $i \in \{wait,\ go\}$ depending on the ego vehicle's strategy for executing the turn.*

We again wish to accurately characterize the set of states $Z^{\text{wait}} \subset X^{\text{wait}}$ and $Z^{\text{go}} \subset X^{\text{go}}$ from which it is possible for the ego vehicle to safely execute its left turn, by either waiting for the oncoming vehicle or going first, respectively. Since the system dynamics are monotone, and since we are again considering a (directed) reach-avoid type specification, we are able to compute safe sets Z^{wait} and Z^{go} using the same symbolic control approach outlined in Section 4.3. The numerical values are as follows: $l = 10$m, $v^{\text{wait}}_{min} = v^{\text{go}}_{min} = 0$m/s, $v^{\text{wait}}_{max} = v^{\text{go}}_{max} = 12$m/s $s^{\text{wait}}_{min} = s^{\text{go}}_{min} = -70$m, $s^{\text{wait}}_{max} = -10$m, $s^{\text{go}}_{max} = 10$m, $r^{\text{wait}}_{max} = 10$m, $r^{\text{go}}_{max} = -10$m, $u_{min} = -2500$Nm and $u_{max} = 1200$Nm. Furthermore, $v_0 \in [8\text{m/s}, 12\text{m/s}]$, and we use the same uncertainty bounds on θ from the previous example. For each scenario, we discretized the state and input using the following resolutions: $s_{\text{res}} = 2$m, $v_{\text{res}} = 0.5$m/s, $r_{\text{res}} = 2$m, and $u_{\text{res}} = 100$Nm. The resulting safe sets are shown in Figure 4.4. Furthermore, to demonstrate



(a) Safe set $Z^{\text{wait}} \subset X^{\text{wait}}$.



(b) Safe set $Z^{\text{go}} \subset X^{\text{go}}$.

Figure 4.4. Safe set boundaries for the unprotected left turn scenario. In Figure 4.4a all states below the surface are in Z^{wait}. Conversely, in Figure 4.4b all states above the surface are in Z^{go}.

the computational advantages of our approach, for this example we have also computed these sets using a standard fixed-point algorithm. Indeed, computing safe sets Z^{wait} and Z^{go} took 47.86s and 202.92s using the lazy fixed-point algorithm, whereas the same computations took 3540.29s and 9991.78s using the standard fixed-point algorithm. We also note that the controller synthesized using the lazy approach can be stored more efficiently, since it only needs to specify upper and lower safety bounds on the control input u for each state. This is in contrast to the controller synthesized using the standard approach, which lists the set of safe control inputs for each state. As a result, controllers $\mathcal{C}^{\text{wait}}$ and $\mathcal{C}^{\text{go}}$ synthesized using the lazy approach can each be stored with 254.2KB of memory, but require 8744.2KB and 4706.4KB of memory, respectively, when synthesized using the standard approach.

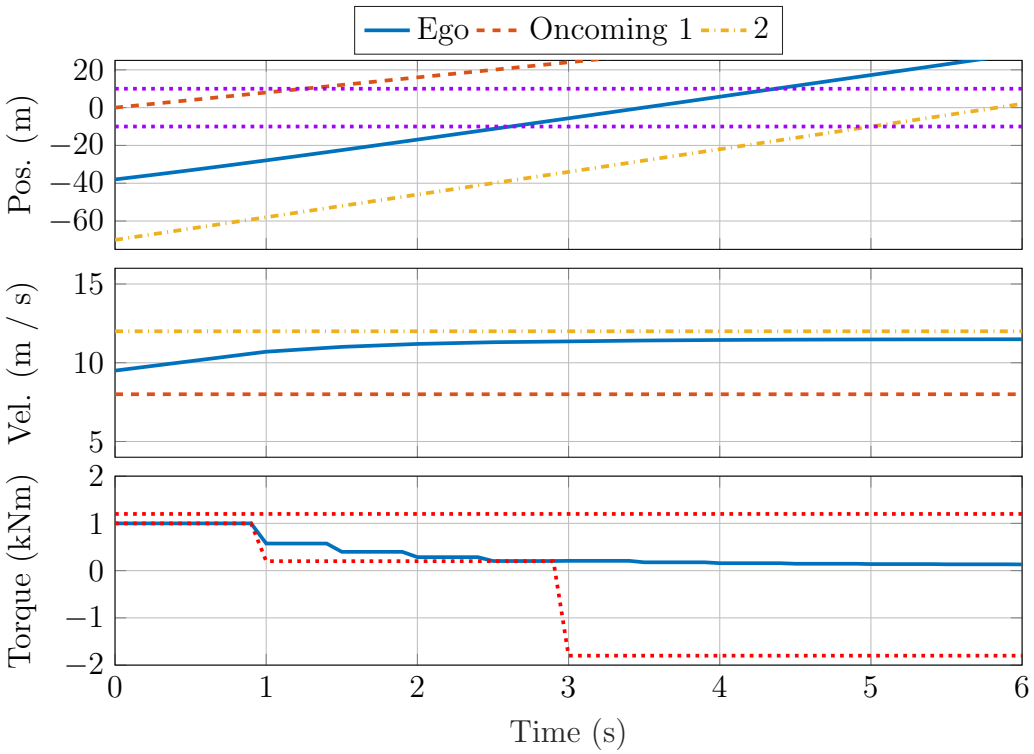


Figure 4.5. Simulation results for the unprotected left turn scenario. The input bounds are indicated with dotted red lines. We note two vehicles never occupy the intersection (bounded by the dotted purple lines) simultaneously.

Two Oncoming Vehicles

We now apply safe sets Z^{wait} and Z^{go} in an unprotected left turn scenario with two oncoming vehicles. Our goal is to design a controller for the ego vehicle such that it safely cuts in-between the two oncoming vehicles to execute its turn, i.e, a controller that keeps the state in $Z^{\text{wait}} \cap Z^{\text{go}}$ at all times. The standard approach to resolve this problem relies on the use of the classical fixed-point algorithm [66], which consists of exploring all the states in $Z^{\text{wait}} \cap Z^{\text{go}}$ and all the inputs $u \in U$. Since we represent the 'wait' and 'go' strategies for executing the turn as upper and lower-closed safety specifications, we can do this by performing an incremental synthesis procedure for the intersection of an upper and lower-closed safety specification in two steps:

1. We synthesize the controllers $\mathcal{C}_{Z^{\text{wait}}}$ and $\mathcal{C}_{Z^{\text{go}}}$ for the lower and upper closed safety specifications Z^{wait} and Z^{go}, respectively.

2. We synthesize the maximal safety controller for the transition system T and safety specification $dom(\mathcal{C}_{Z^{\text{wait}}}) \cap dom(\mathcal{C}_{Z^{\text{go}}})$, where for each state $x \in dom(\mathcal{C}_{Z^{\text{wait}}}) \cap dom(\mathcal{C}_{Z^{\text{go}}})$, we explore only the inputs $u \in \mathcal{C}_{Z^{\text{wait}}}(x) \cap \mathcal{C}_{Z^{\text{go}}}(x)$.

Figure 4.5 shows simulation results with this controller. Since the velocity of each oncoming vehicle is uncertain, we simulate the worst-case scenario where the first and second oncoming vehicles travel at velocities $v_{0,\min} = 8\text{m/s}$ and $v_{0,\max} = 12\text{m/s}$, respectively. At each time step we obtain a feasible range of inputs via the synthesized controller. As long as a control input in this range is selected, the ego vehicle will not conflict with either oncoming vehicle. A simple model-predictive controller is used to choose the optimal control input in this feasible range, with the objective of maintaining a velocity of 11.5m/s. The input bounds and optimal input are both plotted in the bottom of Figure 4.5.

Chapter 5

Conclusion and Future Directions

In this dissertation we explored various methods for designing controllers for large-scale systems, with an emphasis on applications to self-driving vehicles. In particular, we were mainly interested in designing controllers via formal synthesis procedures, such that the system of interest meets a given specification. In order to make the control synthesis problem tractable for large-scale systems, in Chapter 2 we showed how to represent large-scale systems with approximate abstractions that have increased flexibility, to be used in a hierarchical control framework. Next, in Chapter 3 we showed how distributed control techniques can be applied to coordinate CAVs to form platoons of vehicles. Finally, in Chapter 4 we applied formal synthesis techniques to design vehicle controllers for more complex driving maneuvers. The results in Chapters 3 and 4 can help improve the safety and efficiency of traffic flows at intersections, which often become congested with traffic such that they are less safe.

In order for autonomous vehicles to be deployed and operated successfully on a large scale, there are a few additional problems in the area of controls that must be overcome. We now discuss a few interesting research directions related to these challenges which build off of the material presented in this dissertation.

Hierarchical Control for Embedded Systems

The hierarchical control framework discussed in Chapter 2 is a standard approach used in self-driving vehicles. Therefore, it is important to test the control algorithms developed using this framework on real embedded platforms. The main goal is to investigate how well the theoretical error bounds, such as the ones derived in Chapter 2, hold up on a real test vehicle with unmodelled dynamics, limited onboard computational power, and imperfect sensors and actuators. If the theoretical error bounds are violated, additional modifications to the controller may be necessary in order to improve performance. In particular, it will also be interesting to also test feedback laws which are designed using sum-of-squares methods (see related work [63]).

A primary motivation for assembling the scaled car test platform mentioned in Section 1.3 is to evaluate how well it can track a desired reference trajectory in practice, us-

ing the framework established in Chapter 2. We have already begun implementing some of the software that can be used to control the vehicle in this manner using ROS (see https://www.ros.org/about-ros/). Currently, our planner system generates common maneuvers (such as a left or right turn) for the vehicle to track, and the control system then uses NLopt (see https://nlopt.readthedocs.io/en/latest/) to solve a nonlinear MPC problem, allowing the vehicle to accurately follow the planned reference trajectory. Preliminary code is available at https://github.com/swsmth4776/f1tenth.

Automated Tuning of Embedded Controllers

The control system in an autonomous vehicle often utilizes techniques such as model predictive control or feedback control which require manual tuning in a potentially time-consuming process. Indeed, in the project described in Chapter 3 of this dissertation, multiple trial runs on a closed test track were necessary in order to converge on acceptable values for the tuning parameters in our MPC problem. To reduce development time, it will be interesting to see how a learning-based approach could potentially expedite this procedure. Furthermore, since different classes of vehicles have different performance characteristics and therefore need separate tuning values, learning-based performance tuning may also help to accelerate the deployment of autonomous vehicles more broadly. Indeed, many companies are now looking at developing self-driving semi trucks, for example, which certainly handle much differently than smaller vehicle classes such as sedans or crossovers. As these companies expand their operations to include more and more classes of vehicles, the need for automated tuning of embedded controllers will become increasingly important. There is already some promising research along this front; for example, in [49] the authors present a learning model predictive control framework in which a controller automatically improves its own performance by leveraging collected data.

Bibliography

[1] A. Girard and G. J. Pappas. "Hierarchical control system design using approximate simulation". In: *Automatica* 45.2 (2009), pp. 566–571.

[2] H. Ahn and D. Del Vecchio. "Safety verification and control for collision avoidance at road intersections". In: *IEEE Transactions on Automatic Control* 63.3 (2017), pp. 630–642.

[3] A. Alam et al. "Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency". In: *IEEE Control Systems Magazine* 35.6 (2015), pp. 34–56.

[4] D. Angeli and E. D. Sontag. "Monotone control systems". In: *IEEE Transactions on automatic control* 48.10 (2003), pp. 1684–1698.

[5] A. C. Antoulas. *Approximation of large-scale dynamical systems*. Vol. 6. SIAM, 2005.

[6] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.* 2017. URL: `http://docs.mosek.com/8.1/toolbox/index.html`.

[7] M. Arcak, C. Meissen, and A. Packard. *Networks of dissipative systems: compositional certification of stability, performance, and safety*. Springer, 2016.

[8] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[9] C. Bonnet and H. Fritz. *Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing*. Tech. rep. SAE Technical Paper, 2000.

[10] S. Boyd et al. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[11] J. H. Chow, ed. *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems*. Berlin Heidelberg: Springer-Verlag, 1982.

[12] *CVXGEN: Code Generation for Convex Optimization - Handling Infeasibility*. `https://cvxgen.com/docs/infeasibility.html`. Accessed: 2020-04-13.

[13] D. Del Vecchio, M. Malisoff, and R. Verma. "A separation principle for a class of hybrid automata on a partial order". In: *2009 American Control Conference*. IEEE. 2009, pp. 3638–3643.

[14] V. Desaraju et al. "Partial order techniques for vehicle collision avoidance: Application to an autonomous roundabout test-bed". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 82–87.

[15] V. S. Dolk, J. Ploeg, and W. P. M. H. Heemels. "Event-triggered control for string-stable vehicle platooning". In: *IEEE Transactions on Intelligent Transportation Systems* 18.12 (2017), pp. 3486–3500.

[16] A. Donzé. "On signal temporal logic". In: *International Conference on Runtime Verification*. Springer. 2013, pp. 382–383.

[17] A. S. R. Ferreira and M. Arcak. "A graph partitioning approach to predicting patterns in lateral inhibition systems". In: *SIAM Journal on Applied Dynamical Systems* 12.4 (2013), pp. 2012–2031.

[18] *General Motors 2018 Self-Driving Safety Report*. `https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf`. Accessed: 2021-04-20.

[19] A. Girard. "A composition theorem for bisimulation functions". In: *arXiv preprint arXiv:1304.5153* (2013).

[20] A. Girard, G. Gössler, and S. Mouelhi. "Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models". In: *IEEE Transactions on Automatic Control* 61.6 (2016), pp. 1537–1549.

[21] C. Godsil and G. F. Royle. *Algebraic graph theory*. Vol. 207. Springer Science & Business Media, 2013.

[22] O. Grembek et al. "Making intersections safer with I2V communication". In: *Transportation Research Part C: Emerging Technologies* 102 (2019), pp. 396–410.

[23] J. Guanetti, Y. Kim, and F. Borrelli. "Control of connected and automated vehicles: State of the art and future challenges". In: *Annual Reviews in Control* 45 (2018), pp. 18–40.

[24] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018. URL: `http://www.gurobi.com`.

[25] L. Guzzella, A. Sciarretta, et al. *Vehicle propulsion systems*. Vol. 1. Springer, 2007.

[26] H. Sandberg and R. M. Murray. "Model reduction of interconnected linear systems". In: *Optimal Control Applications and Methods* 30.3 (2009), pp. 225–245.

[27] A. Hsu et al. *Design of platoon maneuver protocols for IVHS*. Tech. rep. 1991.

[28] E. S. Kim, M. Arcak, and S. A. Seshia. "Symbolic control design for monotone systems with directed specifications". In: *Automatica* 83 (2017), pp. 10–19. DOI: `https://doi.org/10.1016/j.automatica.2017.04.060`.

[29] J. Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2015, pp. 1094–1099.

[30] S. Lefèvre, A. Carvalho, and F. Borrelli. "A learning-based framework for velocity control in autonomous driving". In: *IEEE Transactions on Automation Science and Engineering* 13.1 (2015), pp. 32–42.

[31] P. Li, L. Alvarez, and R. Horowitz. "AHS safe control laws for platoon leaders". In: *IEEE Transactions on Control Systems Technology* 5.6 (1997), pp. 614–628.

[32] J. Lioris et al. "Platoons of connected vehicles can double throughput in urban roads". In: *Transportation Research Part C: Emerging Technologies* 77 (2017), pp. 292–305.

[33] J. Lofberg. *binmodel*. https://yalmip.github.io/command/binmodel/. 2016 (accessed December 2017).

[34] J. Lofberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*. IEEE. 2004, pp. 284–289.

[35] O. Maler and D. Nickovic. "Monitoring temporal properties of continuous signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[36] J. Mattingley and S. Boyd. "CVXGEN: A code generator for embedded convex optimization". In: *Optimization and Engineering* 13.1 (2012), pp. 1–27.

[37] V. Milanés et al. "Cooperative adaptive cruise control in real traffic situations". In: *IEEE Transactions on Intelligent Transportation Systems* 15.1 (2013), pp. 296–305.

[38] P. Nilsson. "Correct-by-construction control synthesis for high-dimensional systems". PhD thesis. 2017.

[39] P. Nilsson et al. "Correct-by-construction adaptive cruise control: Two approaches". In: *IEEE Transactions on Control Systems Technology* 24.4 (2015), pp. 1294–1307.

[40] E. van Nunen et al. "Cooperative competition for future mobility". In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 1018–1025.

[41] S. Oh et al. "Safe Decision and Control of Connected Automated Vehicles for an Unprotected Left Turn". In: *ASME 2020 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection. 2020.

[42] J. Ploeg et al. "Design and experimental evaluation of cooperative adaptive cruise control". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2011, pp. 260–265.

[43] J. Ploeg et al. "Guest editorial introduction to the special issue on the 2016 grand cooperative driving challenge". In: *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2018), pp. 1208–1212.

[44] J. Ploeg et al. "Introduction to the special issue on the 2011 grand cooperative driving challenge". In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 989–993.

[45] A. Pnueli. "The temporal logic of programs". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE. 1977, pp. 46–57.

[46] *Police: The self-driving Uber in the Arizona crash was hit crossing an intersection on yellow*. `https://www.businessinsider.com/uber-self-driving-car-accident-arizona-police-report-2017-3`. Accessed: 2021-05-16.

[47] V. Raman et al. "Model Predictive Control for Signal Temporal Logic Specification". In: *CoRR* abs/1703.09563 (2017). http://arxiv.org/abs/1703.09563.

[48] *Red light, green light - no light: Tomorrow's communicative cars could take turns at intersections*. `https://doi.org/10.1109/MSPEC.2018.8482420`. Accessed: 2018-10-04.

[49] Ugo Rosolia and Francesco Borrelli. "Learning model predictive control for iterative tasks: A computationally efficient approach for linear system". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 3142–3147.

[50] M. Rungger and M. Zamani. "Compositional Construction of Approximate Abstractions of Interconnected Control Systems". In: *IEEE Transactions on Control of Network Systems* 5.1 (2016), pp. 116–127.

[51] A. Saoud, A. Girard, and L. Fribourg. "Contract-based Design of Symbolic Controllers for Safety in Distributed Multiperiodic Sampled-Data Systems". In: *IEEE Transactions on Automatic Control* (2020). DOI: `10.1109/TAC.2020.2992446`.

[52] A. Saoud, E. Ivanova, and A. Girard. "Efficient synthesis for monotone transition systems and directed safety specifications". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 6255–6260.

[53] A. van der Schaft. "Equivalence of dynamical systems by bisimulation". In: *IEEE Transactions on Automatic Control* 49.12 (2004), pp. 2160–2172.

[54] J. Schindler et al. "MAVEN Deliverable 6.4: Integration Final Report". In: (2020).

[55] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "On a formal model of safe and scalable self-driving cars". In: *arXiv preprint arXiv:1708.06374* (2017).

[56] S. E. Shladover. "PATH at 20 - History and major milestones". In: *IEEE Transactions on intelligent transportation systems* 8.4 (2007), pp. 584–592.

[57] S. E. Shladover, D. Su, and X. Y. Lu. "Impacts of cooperative adaptive cruise control on freeway traffic flow". In: *Transportation Research Record* 2324.1 (2012), pp. 63–70.

[58] J. van de Sluis et al. "D3. 2 Proposal for extended message set for supervised automated driving". In: *Brussels, Belgium, Tech. Rep* 4092490 (2015).

[59] S. W. Smith, M. Arcak, and M. Zamani. "Hierarchical control via an approximate aggregate manifold". In: *American Control Conference, 2018*. IEEE. 2018, pp. 2378–2383.

[60] S. W. Smith, P. Nilsson, and N. Ozay. "Interdependence quantification for compositional control synthesis with an application in vehicle safety systems". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 5700–5707.

[61] S. W. Smith et al. "Balancing Safety and Traffic Throughput in Cooperative Vehicle Platooning". In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 2197–2202.

[62] S. W. Smith et al. "Improving Urban Traffic Throughput With Vehicle Platooning: Theory and Experiments". In: *IEEE Access* 8 (2020), pp. 141208–141223.

[63] Stanley W Smith, He Yin, and Murat Arcak. "Continuous abstraction of nonlinear systems using sum-of-squares programming". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 8093–8098.

[64] E. D. Sontag. *Mathematical control theory*. 2nd. Vol. 6. New York: Springer-Verlag, 1998.

[65] D. V. A. H. G. Swaroop. "String stability of interconnected systems: An application to platooning in automated highway systems". PhD thesis. 1997.

[66] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[67] V. Turri. "Fuel-efficient and safe heavy-duty vehicle platooning through look-ahead control". PhD thesis. KTH Royal Institute of Technology, 2015.

[68] E. Uhlemann. "Platooning: connected vehicles for safety and efficiency [Connected Vehicles]". In: *IEEE Vehicular Technology Magazine* 11.3 (2016), pp. 13–18.

[69] *V2V Safety Technology Now Standard on Cadillac CTS Sedans*. `https://media.cadillac.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2017/mar/0309-v2v.html`. Accessed: 2020-04-01.

[70] H. M. Wang et al. "Conflict Analysis for Cooperative Merging Using V2X Communication". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 1538–1543.

[71] *Watch Cruise's self-driving cars perform 1,400 unprotected left turns in 24 hours*. `https://www.theverge.com/2019/5/23/18637358/cruise-gm-self-driving-unprotected-left-turn`. Accessed: 2021-05-16.

[72] H. Yin et al. "Optimization Based Planner Tracker Design for Safety Guarantees". In: *arXiv preprint arXiv:1910.00782* (2019).

[73] W. H. Young. "On classes of summable functions and their Fourier series". In: *Proc. R. Soc. Lond. A* 87.594 (1912), pp. 225–229.

[74] M. Zamani and M. Arcak. "Compositional abstraction for networks of control systems: A dissipativity approach". In: *IEEE Transactions on Control of Network Systems* 5.3 (2018), pp. 1003–1015.