Abstraction of In-Vehicle Event-Triggered Networked Control Systems for Scheduling

Christiaan Hop

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Abstraction of In-Vehicle Event-Triggered Networked Control Systems for Scheduling

Master of Science Thesis

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

Christiaan Hop

July 11, 2017

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

Abstract

The number of electronic control systems applied in vehicles has increased dramatically over the years. This trend will only continue with the introduction of novel technologies such as advanced driver assistance systems (ADAS). To save cable weight and costs in-vehicle control systems often use shared communication networks. With the increasing use of in-vehicle control systems the scheduling of these networks becomes a serious challenge.

In this master thesis a new event-triggered approach for the scheduling of networked control systems (NCSs) using timed game automata (TGA) is considered. This new approach yields some potential benefits over existing communication protocols and has already been implemented successfully on two-dimensional linear time-invariant (LTI) systems. To obtain the control system TGA for this scheduling approach the NCSs have to be abstracted. The control system abstraction method that is used has to be modified in order to be able to handle higher dimensional systems as well, and that is exactly the focus of this master thesis. The abstraction method is extended and applied to three-dimensional and four-dimensional in-vehicle LTI control systems. The outcomes are checked using simulations which show that the applied abstraction method works.

Table of Contents

List of Figures

List of Tables

Preface

This report is a part of my Master of Science graduation thesis at the Technische Universiteit Delft. After taking some automotive related elective courses during my Systems and Control MSc program I became interested in control techniques applied in the automotive industry.

Over the years the number of electronic control modules in the average vehicle has drastically increased. The increasing interest in autonomous vehicles will only result in more control systems being developed for use in vehicles. This increasing use of in-vehicle control systems yields many possibilities, but also comes with serious challenges. Therefore I thought a project related to control applications in the automotive field would be more than interesting. After consultation with Manuel Mazo, my supervisor for this project, I came to the subject of scheduling in-vehicle NCSs. I think this subject is extremely relevant regarding the previously mentioned increasing use of in-vehicle control systems.

I would like to thank Manuel Mazo for providing me with a relevant and interesting project topic and guiding me during my work on this project and the writing of this report.

Chapter 1

Introduction

Over the last decades the number of electronic control systems applied in the average car has drastically increased. Some common examples are powertrain control, active cruise control (ACC), anti-lock braking system (ABS) and electronic stability program (ESP). These in-vehicle control systems generally consist of sensors, actuators and electronic control units (ECUs). Modern cars can contain anywhere from 50 to 140 ECUs. The cabling necessary for that many ECUs is one of the most heavy and costly components in a car [4]. For this reason point-to-point linking of all the components of in-vehicle control systems is not feasible. Instead, bus-based networks are used in cars in order to share sensor information and send commands from the ECUs to actuators. The use of such communication networks reduces the car weight and production costs.

Various communication buses are being applied in vehicles. Two examples are controller area network (CAN) [5],[6] and Flexray [7]. While existing buses have proven to be effective, the increasing number of control systems used in vehicles poses a problem. This is especially the case with the introduction of new complex technologies such as advanced driver assistance systems (ADAS) and x-by-wire technologies, which results in more ECUs being connected to the communication buses. The increasing bandwidth requirements that come with these novel in-vehicle control systems make that the existing communication networks will not satisfy the needs for those to be applied in future cars. For autonomous vehicles, which nowadays receive an increasing interest and require even more complex control systems, this issue will only become more prevalent.

The communication buses currently applied in vehicles use different scheduling approaches and channel access schemes. A channel access scheme determines which control system is granted access to the communication network at what time in order to send and receive information. Combined with a scheduler it takes care of the real-time scheduling of the networked control systems (NCSs). A new approach to scheduling a shared communication network is described in [1]. This approach could potentially be applied to in-vehicle NCSs and yield

benefits over scheduling protocols that are currently used.

In this report the scheduling approach presented in [1] is outlined and implemented on various in-vehicle control systems. The focus lies on the abstraction method applied to the NCSs to obtain the control system timed game automata (TGA). From these TGA a network of timed game automata (NTGA) could be constructed over which a scheduling policy can be synthesized. Some practical issues when implementing the control system abstraction method (especially for three-dimensional or higher dimensional systems) are discussed, along with solutions to solve these issues. The abstraction method is then applied to various in-vehicle linear time-invariant (LTI) control systems. Finally some simulations for these systems are performed, which show that the applied abstraction method works.

Chapter 2

Scheduling using Timed Game Automata

2-1 Introduction

A new approach to scheduling networked control systems (NCSs) is presented in [1]. The idea is to construct a timed game automaton (TGA) for each control loop and for the shared communication network itself. A TGA is a timed safety automaton (TSA) [8] where the action set consists of controllable and uncontrollable actions. From the obtained set of TGA a parallel composition is taken, resulting in a network of timed game automata (NTGA) [9]. For the constructed NTGA a scheduling strategy is synthesized where the goal is to avoid that multiple control loops try to access the communication network at the same time, i.e. to avoid conflicts. In order to reach this goal, after each transmission on the network the scheduler decides the policy for the next update for each control loop. This policy could either be to base the next update time for a control loop on its event-triggered mechanism or to force an earlier update at a predefined time. Since all control loops are supervised by the scheduler conflicts can be prevented from occurring rather than just being handled when they occur, as is done in carrier sense multiple access (CSMA) protocols such as CAN [5],[6].

2-2 Preliminaries

Before outlining the approach presented in [1], some definitions from this work are introduced.

Definition 1 (Timed Automaton [1], [10])**.** *A timed automaton* TA *is a sextuple* $(L, \ell_0, \mathsf{Act}, C, E, \mathsf{Inv})$ *where:*

- L *is a set of finitely many locations (or nodes);*

- $\ell_0 \in L$ *is an initial location;*
- Act *is a set of finitely many actions;*
- C *is a set of finitely many real-valued clocks;*
- $E \subseteq L \times \mathcal{B}(C) \times \mathsf{Act} \times 2^C \times L$ *is a set of edges;*
- $\mathsf{Inv} : L \to \mathcal{B}(C)$ *assigns invariants to locations.*

Location invariants are restricted to constants that are downwards closed, in the from: $c \leq n$ *or* $c < n$ *where* c *is a clock and* $n \in \mathbb{N}_0$.

A timed automaton (TA) is a directed graph consisting of a number of nodes and edges. The logical clocks are modeled by real-valued variables, which are simply called clocks. The automation behavior is restricted by clock constraints. If an edge is enabled (i.e. if the guard condition of that edge is satisfied by the clock) an edge transition can be taken. When this happens a clock can be reset to zero. From now on TSA [8] which use local invariant conditions will be considered. For simplicity they will be referred to as TA. When $(\ell, g, a, \mathbf{r}, \ell') \in E$, this can be written as $\ell \xrightarrow{g,a,\mathbf{r}} \ell'$ whereas $\ell \to \ell'$ denotes he existence of an edge from ℓ to ℓ' with arbitrary labels.

The set of actions consists of input actions, output actions and internal actions which are denoted by $a?$, $a!$ and $*$ respectively. Input and output actions are used for hand-shake synchronizing communication between TA.

Definition 2 (Operational Semantics [1])**.** *The semantics of a timed automaton is a transition system (also known as a timed transition system) in which states are pairs of location* ℓ *and clock assignment* u*, and transitions are defined by the rules:*

- *Delay transition:* $(\ell, u) \xrightarrow[TS]{d} (\ell, u + d)$ *if* $u \vDash \mathsf{Inv}(\ell)$ *and* $(u + d) \vDash \mathsf{Inv}(\ell)$ *for a non-negative real number* $d \in \mathbb{R}_{\geq 0}$*;*
- *Discrete transition:* $(\ell, u) \xrightarrow[TS]{a} (\ell', u')$ *if* $\ell \xrightarrow{g,a,\mathbf{r}} \ell'$, $u \vDash g$, $u' = u[\mathbf{r}]$ *and* $u' \vDash \mathsf{Inv}(\ell')$.

A run of a timed automaton is a sequence of alternating delay and discrete transitions in the transition system.

The set of runs of timed automaton TA starting from initial state (ℓ_0, u_0) is denoted by Runs(TA). The last state of a finite run ρ is denoted by $last(\rho)$.

Definition 3 (Timed Game Automaton [1]). *A timed game automaton* TGA *is a septuple* $(L, \ell_0, \mathsf{Act}_c, \mathsf{Act}_u, C, E, \mathsf{Inv})$ *where:*

- $(L, \ell_0, \mathsf{Act}_c \cup \mathsf{Act}_u, C, E, \mathsf{Inv})$ *is a timed automaton;*
- Act_c *is a set of controllable actions;*
- Act_u *is a set of uncontrollable actions;*
- $\mathsf{Act}_c \cap \mathsf{Act}_u = \emptyset$.

Definition 4 (Parallel Composition [1]). *Let* TGA^i *be a timed automaton for* $i \in \{1, \ldots, n\}$. *The parallel composition of* $\mathsf{TGA}^i, \ldots, \mathsf{TGA}^n$ *denoted by* $\mathsf{TGA}^i \mid \cdots \mid \mathsf{TGA}^n$ *is a timed game automaton* $\mathsf{TGA} = (L, \ell_0, \mathsf{Act}_c, \mathsf{Act}_u, C, E, \mathsf{Inv})$ *where:*

- $L = L^1 \times \cdots \times L^n$;
- $\ell_0 = \ell_0^1 \times \cdots \times \ell_0^n$;
- $\mathsf{Act}_c = \{*\} \cup \bigcup_{i=1}^n \{a \in \mathsf{Act}_c^i \mid a \text{ is an internal action}\}$;
- $\mathsf{Act}_u = \{\circledast\} \cup \bigcup_{i=1}^n \{a \in \mathsf{Act}_u^i \mid a \text{ is an internal action}\}$;
- $C = C^1 \cup \cdots \cup C^n$;
- *E is defined according to the following two rules:*
  - *a TA makes a move on its own via its internal action: the edge is controllable iff the internal action is controllable;*
  - *two TA move simultaneously via a synchronizing action: the edge is controllable iff both input and output actions are controllable (i.e. the environment has priority over the controller);*
- $\mathsf{Inv}((\ell_1, \ldots, \ell_n)) = \mathsf{Inv}^1(\ell_1) \wedge \cdots \wedge \mathsf{Inv}^n(\ell_n)$

The parallel composition of TGA is called an NTGA . The goal now is to find such a strategy [9] f that a predefined set of bad states $\mathcal{A}$ is avoided at all times. A strategy is a function that

dictates what the controller should do during a run. The strategy can instruct the controller to take a certain controllable action or to not take any action. The latter is referred to as a delay (denoted by λ) and is actually considered a controllable action as well. A strategy is called winning from a state if all maximal runs [11] in the outcome originated from that state are winning. A winning state is a state from which there exists a winning strategy . UPPAAL-Tiga [11] is an example of a software tool which uses on-the-fly algorithms to solve safety control problems.

Definition 5 (Strategy [1],[11]). *Let* $\mathsf{TGA} = (L, \ell_0, \mathsf{Act}_c, \mathsf{Act}_u, C, E, \mathsf{Inv})$ *be a timed automaton. We define* $\mathsf{TA} = (L, \ell_0, \mathsf{Act}_c \cup \mathsf{Act}_u, C, E, \mathsf{Inv})$ *as the timed automaton derived from the timed game automaton. A strategy* f *over* TGA *is a partial function from* Runs(TA) *to* $\mathsf{Act}_c \cup \{\lambda\}$ *s.t. for every finite run* ρ*, if* $f(\rho) \in \mathsf{Act}_c$ *then* $last(\rho) \xrightarrow[TS]{f(\rho)} (\ell', u')$ *for some* (ℓ', u').

2-3 Event-Triggered LTI Control Systems

In [1], the scope is limited to linear time-invariant (LTI) control systems with state-feedback control. The execution of this state-feedback control is event-triggered , as proposed in [12]. The control loops are described as:

$$\dot{\xi}(t) = A\xi(t) + Bv(t), \qquad \xi(t) \in \mathbb{R}^n, \qquad v(t) \in \mathbb{R}^m \tag{2-1}$$

where:

$$v(t) = K\xi(t_k), \qquad t \in [t_k, t_{k+1}) \tag{2-2}$$

A sample-and-hold method is applied to the state-feedback law $v(t)$, meaning that it is kept constant during a time interval $[t_k, t_{k+1})$. It is assumed that $v(t)$ results in a globally asymptotically stable closed-loop system. The update times of the feedback law are based on a sampling triggering law. To introduce this law the measurement error $e(t)$ is defined as:

$$e(t) = \xi(t_k) - \xi(t), \qquad t \in [t_k, t_{k+1}), \qquad k \in \mathbb{N}_0 \tag{2-3}$$

where k is an integer. The feedback law update time is now based on the sampling triggering law :

$$t_{k+1} = min\{t \mid t > t_k \text{ and } |e(t)|^2 \geq \alpha|\xi(t)|^2\}, \qquad \alpha \in \mathbb{R}^+ \tag{2-4}$$

where α is called the triggering coefficient. In [12] it is shown that there exists a lower bound on the time between consecutive control law updates, i.e. that there is no Zeno behavior [13] occurring. An estimate of this lower bound is computed. Some possible advantages of event-triggered control sampling for simple systems are discussed in [14]. The time between consecutive updates for a sampled state x is called the inter-sample time . It is formally defined as:

$$\tau^{\alpha}(x) = min\{t \mid |e(t)|^2 \geq \alpha|\xi(t)|^2 \text{ and } \xi(0) = x\} \tag{2-5}$$

2-4 Communication Network TGA

First a TGA for the communication network that is shared by the control loops in the network is constructed. This TGA (see Figure 2-1) is called TGA^{net} and has three locations: $Idle$, $InUse$ and Bad. When the location $Idle$ is active, the network is available for access. This is the initial location. If the network is available and one of the control loops wants to transmit, the active location changes to $InUse$. When this happens the clock variable c of TGA^{net} is reset to zero. The occupancy time Δ is defined as the time taken by a transmission of a control loop on the network before it becomes available again (and the location of TGA^{net} changes back to $Idle$). This time is assumed to be valid for each control loop. The last location, Bad, represents the case where a conflict occurs, i.e. a situation where a control loop tries to access the network while it is already occupied (and the active location is $InUse$). This location is a so called absorbing location, meaning that it cannot be left once it has been entered. TGA^{net} is formally defined as:

Definition 6 ([1]). *Let Δ represent the maximum channel occupancy time, a timed game automaton associated with the communication network is given by*
$\mathsf{TGA}^{net} = (L^{net}, \ell_0^{net}, \mathsf{Act^{net}}_c, \mathsf{Act}_u^{net}, C^{net}, E^{net}, \mathsf{Inv}^{net})$

- $L^{net} = \{Idle, InUse, Bad\}$;
- $\ell_0^{net} = Idle$;
- $\mathsf{Act}_c^{net} = \{*\}$;
- $\mathsf{Act}_u^{net} = \{up?\}$;
- $C^{net} = \{c\}$;
- $E^{net} = \{(Idle, true, up?, \{c\}, InUse)$
  $(InUse, c = \Delta, *, \emptyset, Idle), (InUse, true, up?, \emptyset, Bad),$
  $(Bad, true, up?, \emptyset, Bad)\}$;
- $\mathsf{Inv}^{net}(InUse) = \{c \mid 0 \leq c \leq \Delta\}$,
  $\mathsf{Inv}^{net}(Idle) = \{c \mid c \geq 0\}$,
  $\mathsf{Inv}^{net}(Bad) = \{c \mid c \geq 0\}$.

The guard true represents a condition that is always satisfied, for example $c \geq 0$.

2-5 Control System TGA

For each control system a TA is constructed. This is done using a three-step procedure described in [15] and is inspired by [16]. First, the state space of the LTI control system is

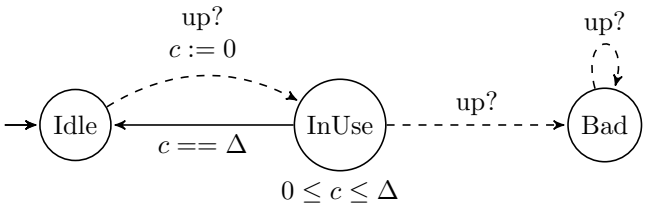

Figure 2-1: Graphical representation of a TA modeling the communication network. Controllable and uncontrollable edges are represented by solid and dashed arrows respectively (adapted from [1]).

partitioned into a finite number of regions by applying the state space abstraction technique proposed in [17]. Subsequently, for each region a time interval that contains all the inter-sample times of the states in that region is calculated using the convex embedding approach from [18]. Finally a reachability analysis [19] is applied to find all the possible transitions between state space regions. From this three-step procedure a finite-state quotient system is obtained. It is shown that this system is semantically equivalent to a TA. The constructed TA captures the sampling behavior of the event-triggered LTI control system and is described by the following definitions:

Definition 7 (Flow Pipe [1])**.** *The set of reachable states or the flow pipe at the time interval* $[t_1, t_2]$ *from a set of initial states* X_0 *is denoted by:*

$$\mathcal{X}_{[t_1,t_2]}(X_0) = \bigcup_{t\in[t_1,t_2]} \{\xi(t) \mid \xi(0) \in X_0\}$$

Definition 8 ([1],[16])**.** *A timed automaton abstracting the triggering timing behavior of system (2-1)-(2-2) with triggering coefficient* α *is given by* $\mathsf{TA}^\alpha = (L^\alpha, \ell_0^\alpha, \mathsf{Act}^\alpha, C^\alpha, E^\alpha, \mathsf{Inv}^\alpha)$ *where:*

- $L^\alpha = \{\mathcal{R}_1^\alpha, \ldots, \mathcal{R}_q^\alpha\}$;
- $\ell_0^\alpha = \mathcal{R}_s^\alpha$ *such that* $\xi(0) \in \mathcal{R}_s^\alpha$;
- $\mathsf{Act}^\alpha = \{*\}$;
- $C^\alpha = \{c\}$;
- $(\mathcal{R}_s^\alpha, \underline{\tau}_s^\alpha \leq c \leq \bar{\tau}_s^\alpha, *, \{c\}, \mathcal{R}_t^\alpha) \in E^\alpha$ *if* $\mathcal{X}_{[\underline{\tau}_s^\alpha, \bar{\tau}_s^\alpha]}(\mathcal{R}_s^\alpha) \cap \mathcal{R}_t^\alpha \neq \emptyset$;
- $\mathsf{Inv}^\alpha(\mathcal{R}_s^\alpha) = \{c \mid 0 \leq c \leq \bar{\tau}_s^\alpha\}$ *for all* $s \in \{1, \ldots, q\}$.

Each location corresponds to one of the regions in which the state space is partitioned. These regions are cones pointed at the origin. The conic shape of the regions is convenient since the

inter-sample time for states that lie on a line through the origin (the origin itself excluded) is the same [17]. The clock variable c represents the time that has elapsed since the last update of the control system. For each region a lower and upper bound ($\underline{\tau}_s^\alpha$ and $\bar{\tau}_s^\alpha$ respectively) of the inter-sample time for the states in that region are calculated by applying the convex embedding approach from [18]. For the inter-sample times of the states x in the TA it holds that [16]:

$$\forall s \in \{1, \dots, q\}, \forall x \in \mathcal{R}_s^\alpha : \tau_\alpha(x) \in [\underline{\tau}_s^\alpha, \bar{\tau}_s^\alpha]$$

The outgoing edges for a region $\mathcal{R}_s^\alpha$ are enabled when the clock variable c lies in the time interval $[\underline{\tau}_s^\alpha, \bar{\tau}_s^\alpha]$. It is guaranteed that a triggering event occurs before $\bar{\tau}_s^\alpha$ and therefore the TA can only remain in a location for a maximum of that time [1]. An example of a TA modeling an event-triggered control loop is shown in Figure 2-2. Now for the control loops timed game automata TGA^{clim} are constructed:

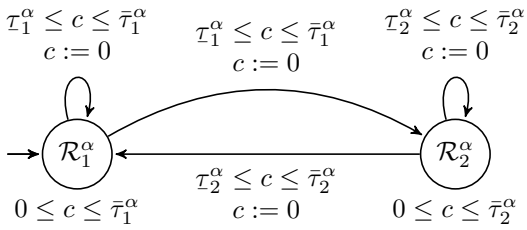

Figure 2-2: Graphical representation of a TA modeling an event-triggered control loop. The initial location is $\mathcal{R}_1^\alpha$ (adapted from [1]).

Definition 9 ([1])**.** *Consider a set of timed automata* $\mathsf{TA}^{\alpha_j} = (L^{\alpha_j}, \ell_0^{\alpha_j}, \mathsf{Act}^{\alpha_j}, C^{\alpha_j}, E^{\alpha_j}, \mathsf{Inv}^{\alpha_j})$ *generated from an event-triggered control loop with triggering coefficients* $\alpha_j \in (0, \bar{\alpha})$ *for* $j \in \{1, \dots, p\}$ *and assume that* $\mathcal{R}_s^{\alpha_1} = \cdots = \mathcal{R}_s^{\alpha_p}$ *for all* $s \in \{1, \dots, q\}$.
Consider also a set of earlier update time parameters $\{\underline{d}_1, \bar{d}_1, \dots, \underline{d}_q \bar{d}_q\}$*, such that*

$$\forall s \in \{1, \dots, q\}, \exists j \in \{1, \dots, p\} : \bar{d}_s \le \underline{\tau}_s^{\alpha_j}.$$

Then, the timed game automata with options for earlier update, choice of triggering coefficients and limiting the consecutive earlier updates are given by $\mathsf{TGA}^{\mathsf{clim}} = (L^{clim}, \ell_0^{clim}, \mathsf{Act}_c^{clim}, \mathsf{Act}_u^{clim}, C^{clim}, E^{clim}, \mathsf{I}$
where:

- $L^{clim} = \bigcup_{j=1}^{p} L^{\alpha_j} \cup \bigcup_{s=1}^{q} \{\mathcal{R}_s, Ear_s\}$;
- $\ell_0^{clim} = \mathcal{R}_s$ *such that* $\xi(0) \in \mathcal{R}_s^{\alpha_1}$
- $\mathsf{Act}_c^{clim} = \{up!\} \cup \mathsf{Act}^{\alpha_1} \cup \bigcup_{j=1}^{p} \{a_j\}$;
- $\mathsf{Act}_u^{clim} = \emptyset$;
- $C^{clim} = C^{\alpha_1}$;

- $E^{cl} = \bigcup_{s=1}^{q} \bigcup_{t \in \mathcal{E}_s} \{Ear_s, c = 0, up!, \emptyset, \mathcal{R}_t\} \cup \bigcup_{s=1}^{q} \bigcup_{j=1}^{p} \{(\mathcal{R}_s, c = 0, a_j, \emptyset, \mathcal{R}_s^{\alpha_j}),$
  $(\mathcal{R}_s^{\alpha_j}, (\underline{d}_s \leq c \leq \bar{d}_s) \wedge (\mathsf{earNum} < \mathsf{earMax}), *, \{c\}, \wedge(\mathsf{earNum} := \mathsf{earNum} + 1), Ear_s)\} \cup$
  $\bigcup_{s=1}^{q} \bigcup_{j=1}^{p} \bigcup_{\{t|(\mathcal{R}_s \to \mathcal{R}_t) \in E^{\alpha_j}\}} \{(\mathcal{R}_s^{\alpha^j}, \underline{\tau}_s^{\alpha_j} \leq c \leq \bar{\tau}_s^{\alpha_j}, up!, \{c\}, \wedge(\mathsf{earNum} := 0), \mathcal{R}_t)\}$;
- $\mathsf{Inv}^{clim}(\mathcal{R}_s^{\alpha_j}) = \{c \mid c \leq \bar{\tau}_s^{\alpha_j}\}$,
  $\mathsf{Inv}^{clim}(\mathcal{R}_s) = \{c \mid c = 0\}$.

For these TA the scheduler can choose the trigger coefficients for the event-triggered control loop, but also force an early update before the control loop is actually triggered by an event. Each location $\mathcal{R}_s^{\alpha^j}$ corresponds to a certain triggering coefficient α_j. When the scheduler has not picked a triggering coefficient yet the active location is $\mathcal{R}_s$. The controllable action a_j represents the scheduler choosing a triggering coefficient and changes the active location from $\mathcal{R}_s$ to $\mathcal{R}_s^{\alpha^j}$. Now the event-triggering mechanism with the chosen triggering coefficient can be used for the next controller update, but the scheduler can also force an early update.
If the event-triggering mechanism is used, the controller update will occur at a time in the interval $[\underline{\tau}_s^{\alpha}, \bar{\tau}_s^{\alpha}]$. When this happens the uncontrollable action $up!$ changes the location to $\mathcal{R}_t$, i.e. the edge from $\mathcal{R}_s^{\alpha^j}$ to $\mathcal{R}_s$ is taken. If an earlier update is forced by the controller, the edge from $\mathcal{R}_s^{\alpha^j}$ to the location Ear_s is taken by the controllable action $*$. Then, from this location the edge to a new location $\mathcal{R}_t$ is taken immediately by the uncontrollable action $up!$. The integer earMax defines the maximum number of consecutive earlier updates allowed to be forced by the scheduler. The number of consecutive earlier updates is counted using the integer earNum. These integers are used to limit the number of consecutive early updates. If the counter earNum reaches the maximal value earMax the next controller update is forced to be based on the event-triggering mechanism. If an event-triggered controller update occurs the counter earNum is reset to zero. In [1] it is shown that the switching between different triggering coefficients or triggering earlier controller updates does not hinder stability.

2-6 Constructing the NTGA

By taking the parallel composition of the TGA representing the communication network and the control loops, the following NTGA is obtained:

$$\mathsf{TGA}^{NCSs} = \mathsf{TGA}^{net} \mid \mathsf{TGA}^{clim1} \mid \cdots \mid \mathsf{TGA}^{climN} \tag{2-6}$$

where N is the number of networked event-triggered control loops. The state of this set of NCSs is described by:

$$(\ell_{net}, \ell_1, \ldots, \ell_N, u_1, \ldots, u_N, \mathsf{earNum}) \tag{2-7}$$

where the bad states $\mathcal{A}$ are:

$$\{(\ell_{net}, \ell_1, \ldots, \ell_N, u_1, \ldots, u_N, \mathsf{earNum}) \mid \ell_{net} = Bad\} \tag{2-8}$$

The goal is to apply a strategy f that avoids these bad states. After a transmission on the communication network, the strategy (see Definition 5) first chooses a triggering coefficient

and then decides which control loop in is updated next. It also chooses whether the next update is based on the event-triggering mechanism or forced at an earlier time. When a controller update occurs the new sampled state (and the conic region it lies in) is determined by the environment. An illustration of a strategy is given in [1]. This work also contains a case study where the outlined approach is applied to two event-triggered NCSs sharing a communication network. UPPAAL-Tiga [11] is used to create the model and generate a strategy.

2-7 Comparing the New Approach to Existing Protocols

Compared to existing CSMA based protocols such as controller area network (CAN) [5],[6], which is widely applied in the automotive industry, the scheduling approach presented in [1] yields a crucial difference. The way in which conflicts (i.e. situations where a control loops tries to access the communication network while it is already occupied by another) are handled is different. In CSMA protocols network access is granted based on priorities of the control loops. If a control loop tries to transmit while the communication network is occupied, the message priority of the control loops competing for network access determines whether the ongoing transmission is completed first or access is granted to the second competing control loop.

In the new approach it is tried to find a strategy in such a way that conflicts are prevented from occurring. The supervisory scheduler has an overview of all the NCSs and the communication network itself, which is not the case in existing protocols where scheduling and network access allocation are handled in separate layers. Situations where multiple control loops have to compete for network access are avoided by allowing the scheduler to pick different triggering coefficients and force earlier updates. As mentioned before an illustration of this approach is given in a case study presented in [1]. Simply put, the new approach prevents conflicts from occurring rather than let them occur and then handle them.

Chapter 3

State Space Partitioning

As mentioned in section 2-5, the first step in constructing a control system abstraction is to partition the state space into a finite number of regions. To obtain this partitioning the state space abstraction technique proposed in [17] is applied. This isotropic covering technique results in the state space being partitioned into conic regions pointing at the origin. The choice for conic regions is convenient since states that lie on a line through the origin have the same inter-sample time $\tau(x)$ [17]:

$$\tau(x) = \tau(\lambda x),\ \forall \lambda \neq 0 \tag{3-1}$$

A cone pointing at the origin is a union of infinitely many of such rays.

3-1 Two-dimensional systems

Consider a two-dimensional state space as shown in Figure 3-1. The goal is to partition the state space into a finite number of conic regions $\mathcal{R}_s$ defined by:

$$\mathcal{R}_s = \{x \in \mathbb{R}^2 | x^T Q_s x \geq 0\} \tag{3-2}$$

If the condition in Equation 3-2 holds for a certain state x then it will hold for the state $-x$ as well. Because of this symmetry it suffices to calculate the matrices Q_s for only half of the state space (where $\theta \in [0, \pi]$). The same matrices then apply to the second half of the state space.

In Figure 3-1 the number of subdivisions for half of the state space m is chosen as four. The total number of regions for the entire state space is then eight. To distinguish between regions that have the same matrix Q_s defining it (e.g. $\mathcal{R}_1$ and $\mathcal{R}_5$ in Figure 3-1) the sign of the state values has to be considered. The region matrices Q_s can be calculated using geometric arguments, as explained in section 3-3.

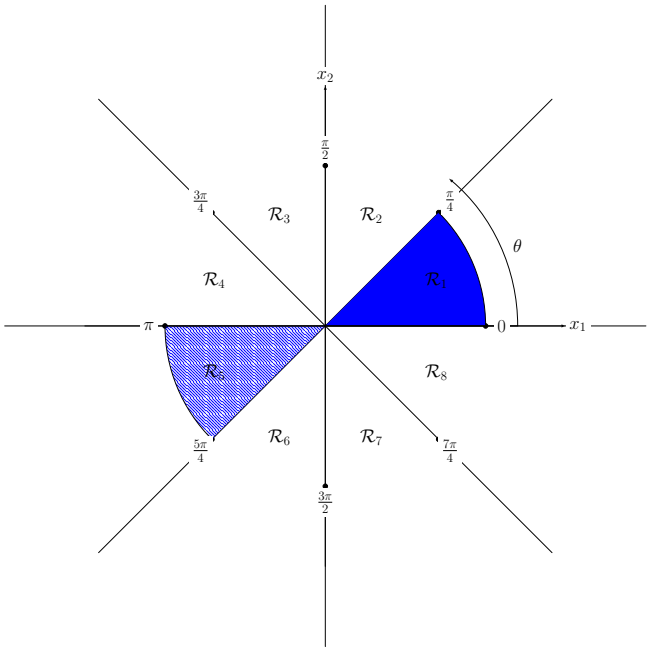

Figure 3-1: Partitioning of a two-dimensional state space with $m = 4$. Note that the blue arcs are only highlighting the regions $\mathcal{R}_1$ and $\mathcal{R}_5$ which have the same matrix Q_s defining them, but are not bounding these regions.

3-2 n-Dimensional systems

For n-dimensional systems (with $n \geq 3$) conic regions pointing at the origin can be defined as:

$$\mathcal{R}_s = \{x \in \mathbb{R}^n | E_s x \geq 0\} \tag{3-3}$$

Again the region matrices $E_s \in \mathbb{R}^{p \times n}$ (with $p \leq 2n-2$) can be calculated using geometric arguments. For higher dimensional systems these calculations become very complicated and moreover can lead to numerical issues when implemented into MATLAB [20]. MATLAB code that calculates the matrices E_s is available [21], but only works for three-dimensional systems.

Instead of defining a region $\mathcal{R}_s$ by one matrix E_s (Equation 3-3) it can be defined by looking at its projections onto the $(n-1)$ two-dimensional $(x_i, x_{(i+1)})$-planes and their corresponding 2×2 matrix $Q_s^{i,(i+1)}$ (with $s \in \{1, 2, \ldots, m\}$). In this way each region is defined by a particular combination of $(n-1)$ matrices:

$$\mathcal{R}_s = \{x \in^n | x_{1,2}^T Q_s^{1,2} x_{1,2} \geq 0 \wedge x_{2,3}^T Q_s^{2,3} x_{2,3} \geq 0 \wedge \ldots \wedge x_{(n-1),n}^T Q_s^{(n-1),n} x_{(n-1),n} \geq 0\} \tag{3-4}$$

where $x_{i,(i+1)}$ refers to $[x_i \ x_{i+1}]^T$, i.e. the i-th and $(i+1)$-th index of the n-dimensional state. Now each region $\mathcal{R}_s$ corresponds to a combination of $(n-1)$ 2×2 matrices $Q_s^{i,(i+1)}$. An

illustration of this region representation is shown in Figure 3-2, where a three-dimensional conic region is projected onto the (x_1, x_2)-plane and the (x_2, x_3)-plane. The projection onto the (x_1, x_2)-plane will correspond to a certain matrix $Q_i^{1,2}$ whereas the projection onto the (x_2, x_3)-plane will correspond to a certain matrix $Q_j^{2,3}$ (with $i, j \in \{1, 2, \ldots, m\}$). This particular combination of 2×2 matrices $Q_i^{1,2}$ and $Q_j^{2,3}$ defines the three-dimensional conic region.

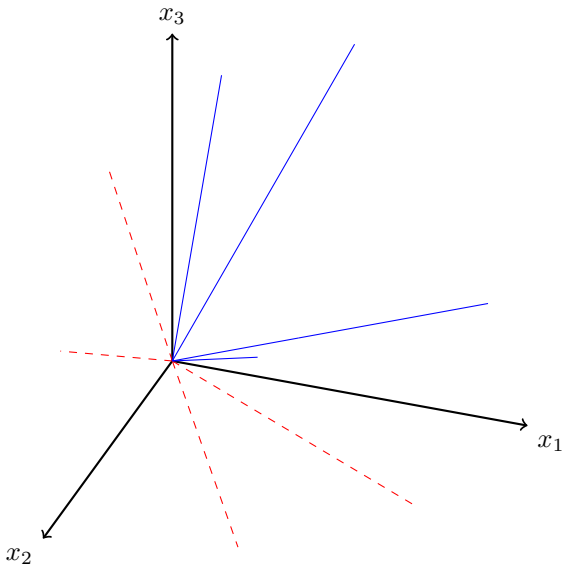

Figure 3-2: Example of a three-dimensional conic region (blue) and its projections onto the (x_1, x_2)-plane and the (x_2, x_3)-plane (red).

The calculation of the 2×2 matrices $Q_s^{i,(i+1)}$ is simple compared to the calculation of the $p \times n$ matrices E_s and far less prone to numerical issues when implemented in MATLAB. Therefore defining the regions as in Equation 3-4 is a useful alternative to the definition in Equation 3-3.

3-3 Region Matrix Calculations

Consider the region $\mathcal{R}_1$ in Figure 3-1. This region is characterized by the angles $\theta_{min} = 0$ rad and $\theta_{min} = \frac{\pi}{4}$. At both these angles a line through the origin can be drawn, as shown in Figure 3-3 and Figure 3-4. Both these lines divide the state space into two half-spaces. They can be defined by their normal vector as $a_i^T x = 0$ where $x = [x_1 \ x_2]^T$. From both these lines the normal vector a_i pointing to the inside of the region $\mathcal{R}_1$ is drawn. The half-spaces to which these normal vectors point (coloured blue in Figure 3-3 and Figure 3-4) are defined by $a_i^T x \geq 0$. The normal vectors are calculated as:

$$\begin{aligned} a_1^T &= [-\sin(\theta_{min}) \ \ \cos(\theta_{min})] \\ a_2^T &= [\sin(\theta_{max}) \ \ -\cos(\theta_{max})] \end{aligned} \tag{3-5}$$

The intersection of the two half-spaces is exactly the region $\mathcal{R}_1$. Since for states that lie within the half-spaces it holds that:

$$a_1^T x \geq 0 \tag{3-6}$$

and:

$$a_2^T x \geq 0 \tag{3-7}$$

for the states that lie within the intersection of the half-spaces (which is $\mathcal{R}_1$) it holds that:

$$x^T (a_1 a_2^T) x \geq 0 \tag{3-8}$$

Instead of defining the matrices Q_s (corresponding to the regions $\mathcal{R}_s$) as $Q_s = (a_1 a_2^T)$ they are defined as:

$$Q_s = (a_1 a_2^T + a_2 a_1^T) \tag{3-9}$$

In this way the matrices Q_s are made symmetric, which is numerically advantageous since the linear matrix inequality (LMI) solvers that are used to calculate the sample time bounds for the regions $\mathcal{R}_s$ can handle symmetric matrices more efficiently compared to general matrices.

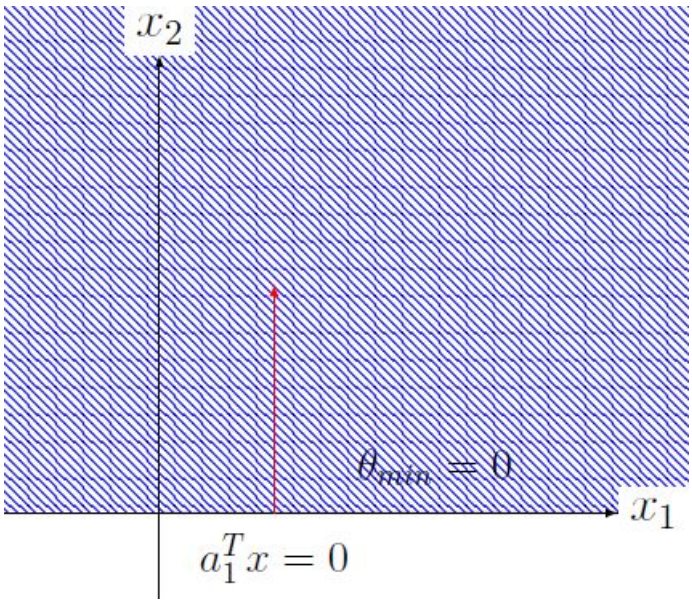

Figure 3-3: The line through the origin corresponding to θ_{min} and its normal vector a_1 (red). The half-space to which the normal vector points is coloured in blue.

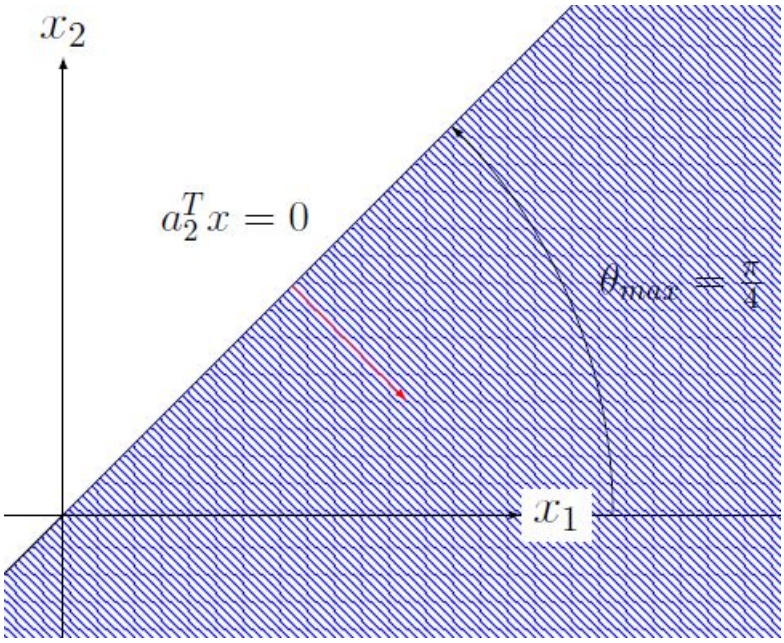

Figure 3-4: The line through the origin corresponding to θ_{max} and its normal vector a_2 (red). The half-space to which the normal vector points is coloured in blue.

3-4 Calculating Region Vertices

The region polyhedra in which the state space is partitioned can be defined by their vertices. Calculating these vertices is useful for the reachability analysis. For a two-dimensional system (Figure 3-1), where each region is defined by one angular coordinate, the vertices can be calculated as:

$$\begin{aligned} x_1 &= cos(\theta) \\ x_2 &= sin(\theta) \end{aligned} \tag{3-10}$$

for both the minimum and maximum angle (θ^{min} and θ^{max}) for that region, resulting in two different vertices.

For an n-dimensional systems (with $n \geq 3$), each region is defined by $(n-1)$ angular coordinates. With a minimum and maximum value for each angular coordinate, an n-dimensional region is defined by $2^{(n-1)}$ vertices. The coordinates for each vertex V can be calculated as:

$$
\begin{cases}
x_1 = \cos(\theta_1) \ \textit{if} \ |\theta_2| \neq \dfrac{\pi}{2} \\
x_2 = \sin(\theta_1) \ \textit{if} \ |\theta_2| \neq \dfrac{\pi}{2} \\
x_2 = x_1 = 0 \ \textit{if} \ |\theta_2| = \dfrac{\pi}{2} \\
x_3 = |x_2| \tan(\theta_2) \ \textit{if} \ |\theta_2| \neq \dfrac{\pi}{2} \\
x_3 = 1 \ \textit{if} \ \theta_2 = \dfrac{\pi}{2} \\
x_3 = -1 \ \textit{if} \ \theta_2 = -\dfrac{\pi}{2} \\
\quad \vdots \\
x_{(i+1)} = |x_i| \tan(\theta_i) \ \textit{if} \ |\theta_i| \neq \dfrac{\pi}{2} \\
x_{(i+1)} = 1, \ x_i = x_{(i-1)} = \cdots = x_1 = 0 \ \textit{if} \ \theta_i = \dfrac{\pi}{2} \\
x_{(i+1)} = -1, \ x_i = x_{(i-1)} = \cdots = x_1 = 0 \ \textit{if} \ \theta_i = -\dfrac{\pi}{2} \\
\quad \vdots \\
x_n = |x_{(n-1)}| \tan(\theta_{(n-1)}) \ \textit{if} \ |\theta_{(n-1)}| \neq \dfrac{\pi}{2} \\
x_n = 1, \ x_{(n-1)} = x_{(n-2)} = \cdots = x_1 = 0 \ \textit{if} \ \theta_{(n-1)} = \dfrac{\pi}{2} \\
x_n = -1, \ x_{(n-1)} = x_{(n-2)} = \cdots = x_1 = 0 \ \textit{if} \ \theta_{(n-1)} = -\dfrac{\pi}{2}
\end{cases}
\tag{3-11}
$$

where $\theta_1 \in [0, \pi]$ and $\theta_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ for $i \in \{2, \ldots, (n-1)\}$. Note that by considering all angular coordinates over an interval of length π only half of the state space is considered. With each of the $(n-1)$ angular coordinates (lying within an interval of length π) divided into m subintervals, this half of the state space will contain $m^{(n-1)}$ conic regions. Due to the aforementioned symmetry the regions in the first half of the state space can be mapped to the second half of the state space by taking $V = -V$ for each vertex V. The entire state space then is divided into $2 \times m^{(n-1)}$ conic regions. Calculating the vertices in this way is consistent with the region representation as given in Equation 3-4.

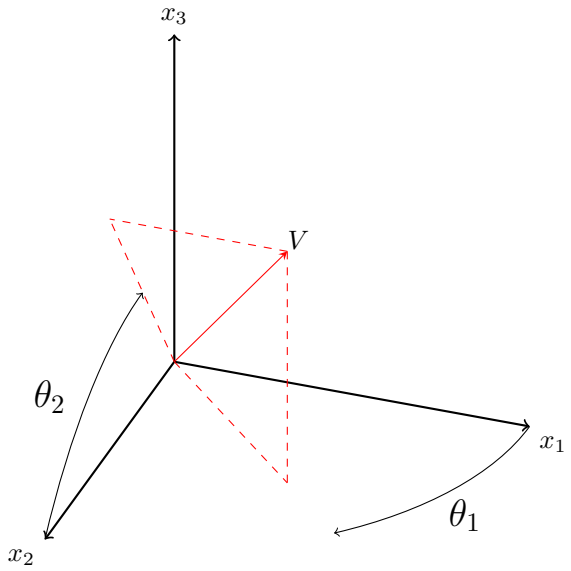

Figure 3-5: Vertex V in a three-dimensional state space and its projections onto the (x_1, x_2)-plane and the (x_2, x_3)-plane, and their corresponding angular coordinates.

Chapter 4

Inter-Sample Time Bound Calculations

After partitioning the state space into conic regions, for each region $\mathcal{R}_s$ an upper bound $\bar{\tau}_s$ and lower bound $\underline{\tau}_s$ on the sampling time for the states within that region have to be calculated. The calculation of this time interval is done using the convex embedding approach described in [18], making use of the S-procedure [22, Appendix B.2, p. 655].

4-1 Time Bound Calculations using Convex Embedding

Recall from section 2-3 that the linear time-invariant (LTI) control systems with event-triggered state feedback that are considered could be described as:

$$\dot{\xi}(t) = A\xi(t) + Bv(t), \qquad \xi(t) \in \mathbb{R}^n, \qquad v(t) \in \mathbb{R}^m \tag{4-1}$$

with state-feedback law:

$$v(t) = K\xi(t_k), \qquad t \in [t_k, t_{k+1}) \tag{4-2}$$

The sampling triggering law is:

$$t_{k+1} = min\{t \mid t > t_k \text{ and } |e(t)|^2 \geq \alpha|\xi(t)|^2\}, \qquad \alpha \in (0, \bar{\sigma}) \subset \mathbb{R}^+ \tag{4-3}$$

where the measurement error $e(t)$ is:

$$e(t) = \xi(t_k) - \xi(t), \qquad t \in [t_k, t_{k+1}), \qquad k \in \mathbb{N}_0 \tag{4-4}$$

where k is an integer. Denote a sampled state at sampling instant t_k by $\xi(t_k) = x$. This sampled state x has an inter-sample time $\tau(x) = t_{k+1} - t_k$. Within the interval $[t_k, t_{k+1}]$ the evolution of the state and the measurement error are described by:

$$\xi_x(t_k + \sigma) = \Lambda(\sigma)x \tag{4-5}$$

$$e_x(t_k + \sigma) = [I - \Lambda(\sigma)]x \tag{4-6}$$

with $\sigma \in [0, t_{k+1} - t_k]$ and:

$$\Lambda(\sigma) = [I + \int_0^\sigma e^{Ar} dr(A + BK)] \tag{4-7}$$

Now by combining Equation 4-3, 4-5 and 4-6 the inter-sample time $\tau(x)$ can be expressed as:

$$\tau(x) = min\{\sigma > 0 \mid x^T \Phi(\sigma)x = 0\} \tag{4-8}$$

where:

$$\Phi(x) = [I - \Lambda^T(\sigma)][I - \Lambda(\sigma)] - \alpha\Lambda^T(\sigma)\Lambda(\sigma) \tag{4-9}$$

To find a lower bound $\underline{\tau}_s$ on the inter-sample time, a finite set of matrices $\underline{\Phi}_{\kappa,s}$ is constructed such that:

$$(x^T \underline{\Phi}_{\kappa,s} x \leq 0, \forall \kappa \in \mathcal{K}_s) \implies (x^T \Phi(\sigma)x \leq 0, \forall \sigma \in [0, \underline{\tau}_s]). \tag{4-10}$$

Similarly, to find an upper bound $\bar{\tau}_s$ on the inter-sample time, a finite set of matrices $\bar{\Phi}_{\kappa,s}$ is constructed such that:

$$(x^T \bar{\Phi}_{\kappa,s} x \geq 0, \forall \kappa \in \mathcal{K}_s) \implies (x^T \Phi(\sigma)x \geq 0, \forall \sigma \in [\bar{\tau}_s, \bar{\sigma}]) \tag{4-11}$$

where $\bar{\sigma} > 0$ is a time instant that is larger than the inter-sample time (Equation 4-8) for any state in the entire state space, which yields:

$$x^T \Phi(\bar{\sigma})x \geq 0, \forall x \in \mathbb{R}^n. \tag{4-12}$$

Finding $\bar{\sigma}$ is done by simply increasing it until the resulting inter-sample time bounds $\underline{\tau}_s$ and $\bar{\tau}_s$ satisfy $\underline{\tau}_s, \bar{\tau}_s < \bar{\sigma}$.

How exactly the matrices $\underline{\Phi}_{\kappa,s}$ are constructed and how they can be used to find a lower bound $\underline{\tau}_s$ on the inter-sample time is described in Lemma 1.

Lemma 1 ([16]). *Consider a time limit $\underline{\tau}_s \in (0, \bar{\sigma}]$. If*

$$x^T \underline{\Phi}_{(i,j),s} x \leq 0,$$

$$\forall (i,j) \in \mathcal{K}_s = (\{0, \ldots, N_{conv}\} \times \{0, \ldots, \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor\}),$$

then

$$x^T\Phi(\sigma)x \leq 0\,, \forall \sigma \in [0, \underline{\tau}_s],$$

with Φ *as in (4-8) and*

$$\underline{\Phi}_{(i,j),s} := \underline{\Phi'}_{(i,j),s} + \underline{\nu} I, \tag{4-13}$$

$$\underline{\Phi'}_{(i,j),s} := \begin{cases} \sum_{k=0}^{i} L_{k,j}(\frac{\bar{\sigma}}{l})^k & if\ j < \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor, \\ \sum_{k=0}^{i} L_{k,j}(\underline{\tau}_s - \frac{j\bar{\sigma}}{l})^k & j = \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor, \end{cases} \tag{4-14}$$

$$\begin{cases} L_{0,j} := & I - \Pi_{1,j} - \Pi_{1,j}^T + (1-\alpha)\Pi_{1,j}^T\Pi_{1,j}, \\ L_{1,j} := & [(1-\alpha)\Pi_{1,j}^T - I]\Pi_{2,j} \\ & +\Pi_{2,j}^T[(1-\alpha)\Pi_{1,j} - I], \\ L_{k\geq 2,j} := & [(1-\alpha)\Pi_{1,j}^T - I]\frac{A^{k-1}}{k!}\Pi_{2,j} \\ & +\Pi_{2,j}^T\frac{(A^{k-1})^T}{k!}[(1-\alpha)\Pi_{1,j} - I] \\ & +(1-\alpha)\Pi_{2,j}^T(\sum_{i=1}^{k-1}\frac{(A^{i-1})^T}{i!}\frac{A^{k-i-1}}{(k-i)!})\Pi_{2,j}, \end{cases} \tag{4-15}$$

$$\begin{cases} \Pi_{1,j} := I + M_j(A + BK), \\ \Pi_{2,j} := N_j(A + BK), \end{cases} \tag{4-16}$$

$$M_j := \textstyle\int_0^{j\frac{\bar{\sigma}}{l}} e^{As} ds, \quad N_j := AM_j + I, \tag{4-17}$$

and

$$\underline{\nu} \geq \max_{\sigma' \in [0, \frac{\bar{\sigma}}{l}],\ r \in \{0,\ldots,l-1\}} \lambda_{\max}(\Phi(\sigma' + r\tfrac{\bar{\sigma}}{l}) - \tilde{\Phi}_{N_{conv},r}(\sigma')), \tag{4-18}$$

where

$$\tilde{\Phi}_{N_{conv},r}(\sigma') := \textstyle\sum_{k=0}^{N_{conv}} L_{k,r}\sigma'^k. \tag{4-19}$$

Proof. See Appendix A-1. □

To obtain a less conservative result the following theorem is used:

Theorem 1 (Regional Lower Bound Approximation [16])**.** *Consider the inter-sampling time set* $\{\underline{\tau}_1, \ldots, \underline{\tau}_q\}$ *and matrices* $\underline{\Phi}_{\kappa,s}$ *satisfying* $\forall s \in \{1, \ldots, q\}$, $\forall\, \kappa = (i,j) \in \mathcal{K}_s$, $0 < \underline{\tau}_s \leq \bar{\sigma}$, $\underline{\Phi}_{\kappa,s} \preceq 0$. *If there exist scalars* $\underline{\varepsilon}_{\kappa,s} \geq 0$ *(for* $n = 2$*) or symmetric matrices* $\underline{U}_{\kappa,s}$ *with nonnegative entries (for* $n \geq 3$*) such that the linear matrix inequalities (LMIs)*

$$\underline{\Phi}_{\kappa,s} + \underline{\varepsilon}_{\kappa,s} Q_s \preceq 0 \quad if \quad n = 2 \tag{4-20}$$

$$\underline{\Phi}_{\kappa,s} + E_s^T \underline{U}_{\kappa,s} E_s \preceq 0 \quad if \quad n \geq 3 \tag{4-21}$$

hold, then $\forall x \in \mathcal{R}_s$ *determined by (3-2) or (3-3) the inter-sample time (4-3) of the control system (5-1-5-2) is regionally bounded from below by* $\underline{\tau}_s$.

Proof. The result is a direct consequence of applying a lossless (if $n = 2$) or a lossy (if $n \geq 3$) version of the S-procedure: if the stated conditions hold then, for every $x \in \mathbb{R}^n$, $x^T Q_s x \geq 0$ (or $x^T E_s^T \underline{\mathrm{U}}_{\kappa,s} E_s x \geq 0$) implies that $x^T \underline{\Phi}_{\kappa,s} x \leq 0$. From (3-2) (or (3-3)) we have that $x^T Q_s x \geq 0$ (or $E_s x \geq 0$) iff $x \in \mathcal{R}_s$. Thus one can conclude that for every $x \in \mathcal{R}_s$, $x^T \underline{\Phi}_{\kappa,s} x \leq 0$ and by Lemma 1 the result follows. □

Following the same principles the upper bound $\bar{\tau}_s$ on the inter-sample time for each region can be calculated. The procedure for these calculations is described in Lemma 2 and Theorem 2.

Lemma 2. *Consider a time limit $\bar{\tau}_s \in (\underline{\tau}_s, \bar{\sigma}]$. If*

$$x^T \bar{\Phi}_{(i,j),s} x \geq 0,$$

$$\forall (i,j) \in \mathcal{K}_s = (\{0, \ldots, N_{conv}\} \times \{\lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \ldots, l-1\}),$$

then

$$x^T \Phi(\sigma) x \geq 0, \, \forall \sigma \in [\bar{\tau}_s, \bar{\sigma}],$$

with Φ as in (4-8) and

$$\bar{\Phi}_{(i,j),s} := \bar{\Phi}'_{(i,j),s} + \bar{\nu} I,$$

$$\bar{\Phi}'_{(i,j),s} := \begin{cases} \sum_{k=0}^{i} L_{k,j}((j+1)\frac{\bar{\sigma}}{l} - \bar{\tau}_s)^k \; if \; j = \lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \\ \sum_{k=0}^{i} L_{k,j}(\frac{\bar{\sigma}}{l})^k \quad if \; j > \lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \end{cases} \tag{4-22}$$

$$\bar{\nu} \leq \max_{\sigma' \in [0, \frac{\bar{\sigma}}{l}], \, r \in \{0, \ldots, l-1\}} \lambda_{\min}(\Phi(\sigma' + r\frac{\bar{\sigma}}{l}) - \tilde{\Phi}_{N_{conv}, r}(\sigma')), \tag{4-23}$$

and $L_{k,j}$ given by (4-15).

Proof. See Appendix A-2. □

Theorem 2 (Regional Upper Bound Approximation [16])**.** *Consider the inter-sampling time set $\{\bar{\tau}_1, \ldots, \bar{\tau}_q\}$ and matrices $\bar{\Phi}_{\kappa,s}$ satisfying $\forall s \in \{1, \ldots, q\}$, $\forall \kappa = (i,j) \in \mathcal{K}_s = (\{0, \ldots, N_{conv}\} \times \{\lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \ldots, l-1\})$, $\underline{\tau}_s < \bar{\tau}_s \leq \bar{\sigma}$, $\bar{\Phi}_{\kappa,s} \succeq 0$. If there exist scalars $\bar{\varepsilon}_{\kappa,s} \geq 0$ (for $n = 2$) or symmetric matrices $\bar{U}_{\kappa,s}$ with nonnegative entries (for $n \geq 3$) such that the LMIs*

$$\bar{\Phi}_{\kappa,s} - \bar{\varepsilon}_{\kappa,s} Q_s \succeq 0 \quad if \quad n = 2 \tag{4-24}$$

$$\bar{\Phi}_{\kappa,s} - E_s^T \bar{U}_{\kappa,s} E_s \succeq 0 \quad if \quad n \geq 3 \tag{4-25}$$

hold, then $\forall x \in \mathcal{R}_s$ determined by (3-2) or (3-3) the inter-sample time (4-3) of the control system (5-1-5-2) is regionally bounded from above by $\bar{\tau}_s$.

Proof. Analogous to the proof of Theorem 1. □

4-2 Time Bound Calculations for Alternative Region Representation

When the alternative region representation introduced in section 3-2 for an n-dimensional system (with $n \leq 3$) some adjustments to Theorem 1 and Theorem 2 are necessary. Recall with this representation the regions are defined as:

$$\mathcal{R}_s = \{x \in^n | x_{1,2}^T Q_s^{1,2} x_{1,2} \geq 0 \wedge x_{2,3}^T Q_s^{2,3} x_{2,3} \geq 0 \wedge \ldots \wedge x_{(n-1),n}^T Q_s^{(n-1),n} x_{(n-1),n} \geq 0\} \tag{4-26}$$

where $x_{i,(i+1)}$ refers to $[x_i \; x_{i+1}]^T$, i.e. the i-th and $(i+1)$-th index of the n-dimensional state. instead of:

$$\mathcal{R}_s = \{x \in \mathbb{R}^n | E_s x \geq 0\} \tag{4-27}$$

Looking at Theorem 1, instead of Equation 4-21 the LMIs that have to be considered are:

$$\underline{\Phi}_{\kappa,s} + \underline{\varepsilon}^{1,2}_{\kappa,s}\tilde{Q}^{1,2}_s + \underline{\varepsilon}^{2,3}_{\kappa,s}\tilde{Q}^{2,3}_s + \cdots + \underline{\varepsilon}^{(n-1),n}_{\kappa,s}\tilde{Q}^{(n-1),n}_s \preceq 0 \tag{4-28}$$

where $\underline{\varepsilon}^{i,(i+1)}_{\kappa,s}$ are nonnegative scalars. Similarly, instead of Equation 4-25, the LMIs that have to be considered to find the upper bounds on the inter-sample times are:

$$\bar{\Phi}_{\kappa,s} - \bar{\varepsilon}^{1,2}_{\kappa,s}\tilde{Q}^{1,2}_s - \bar{\varepsilon}^{2,3}_{\kappa,s}\tilde{Q}^{2,3}_s - \cdots - \bar{\varepsilon}^{(n-1),n}_{\kappa,s}\tilde{Q}^{(n-1),n}_s \succeq 0 \tag{4-29}$$

where $\bar{\varepsilon}^{i,(i+1)}_{\kappa,s}$ are nonnegative scalars. By applying the S-procedure to these LMIs and using Lemma 1 and Lemma 2 (analogous to the proof of Theorem 1) it can be proven that this approach indeed gives the lower and upper inter-sample time bounds.

Note that the matrices $Q^{i,(i+1)}_s$ in Equation 4-26 are 2×2 matrices whereas the matrices $\tilde{Q}^{i,(i+1)}_s$ in Equation 4-28 and Equation 4-29 are $n \times n$ matrices (since $\underline{\Phi}_{\kappa,s}$ and $\bar{\Phi}_{\kappa,s}$ are $n \times n$ too). The matrices $\tilde{Q}^{i,(i+1)}_s$ are $n \times n$ matrices with all zero entries except for the (i,i)-th, $(i,i+1)$-th, $(i+1,i)$-th and $(i+1,i+1)$-th entries. These entries are equal to the entries $(1,1)$, $(1,2)$, $(2,1)$ and $(2,2)$ of the corresponding matrices $Q^{i,(i+1)}_s$. For example, a region $\mathcal{R}_s$ in a three-dimensional state space with:

$$Q^{1,2}_s = \begin{bmatrix} q^{1,2}_{11} & q^{1,2}_{12} \\ q^{1,2}_{21} & q^{2,2}_{12} \end{bmatrix}$$

and:

$$Q^{2,3}_s = \begin{bmatrix} q^{2,3}_{11} & q^{2,3}_{12} \\ q^{2,3}_{21} & q^{2,3}_{12} \end{bmatrix}$$

would yield:

$$\tilde{Q}^{1,2}_s = \begin{bmatrix} q^{1,2}_{11} & q^{1,2}_{12} & 0 \\ q^{1,2}_{21} & q^{1,2}_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and:

$$\tilde{Q}^{2,3}_s = \begin{bmatrix} 0 & 0 & 0 \\ 0 & q^{2,3}_{11} & q^{2,3}_{12} \\ 0 & q^{2,3}_{21} & q^{2,3}_{22} \end{bmatrix}$$

Solving the LMIs to obtain the inter-sample time bounds is done using the YALMIP toolbox [23] in MATLAB [20].

Chapter 5

Reachability Analysis

5-1 The Multi-Parametric Toolbox

The final step in constructing a control system timed automaton (TA) is a reachability analysis that gives all the possible transitions between state space regions. To perform this reachability analysis the Multi-Parametric Toolbox (MPT) [24] in MATLAB [20] is used. This toolbox contains a function that computes reachable sets for discrete linear time-invariant (LTI) systems. The domain of the states for which the reachable set is computed can be defined by a polyhedron. The same can be done for the domain of the inputs. Furthermore the number of steps N, that combined with the sampling time t_s defines the time $N \cdot t_s$ for which the reachable set is computed, can be specified.

5-2 Implementing the Reachability Analysis in the Multi-Parametric Toolbox

The goal is to obtain all possible transitions between regions as defined in chapter 3. Therefore a reachability analysis for each of these regions is done, where the domain of the states is the conic region $\mathcal{R}_s$. To construct these polyhedra in MATLAB their vertices are calculated as in Equation 3-11. Since the upper and lower bound ($\bar{\tau}_s$ and $\underline{\tau}_s$ respectively) on the inter-sample time for each region are known the reachability analysis is done for a time interval $[\underline{\tau}_s, \bar{\tau}_s]$. This is done as explained below.

Recall from chapter 4 that the considered continuous-time control systems are of the form:

$$\dot{\xi}(t) = A\xi(t) + Bv(t), \qquad \xi(t) \in \mathbb{R}^n, \qquad v(t) \in \mathbb{R}^m \tag{5-1}$$

with state-feedback law:

$$v(t) = K\xi(t_k), \qquad t \in [t_k, t_{k+1}) \tag{5-2}$$

For the reachability analysis, that will be performed on the discretized system, first a general sampling time t_s is defined as:

$$t_s = \underline{\tau}_s/h, \qquad h \in \mathbb{N} \tag{5-3}$$

The reachability analysis is then done for the steps:

$$N = \{N_{min}, N_{min+1}, \ldots, N_{max}\} \tag{5-4}$$

where:

$$\begin{aligned} N_{min} &= \left\lfloor \frac{\underline{\tau}_s}{t_s} \right\rfloor = h \\ N_{max} &= \left\lceil \frac{\bar{\tau}_s}{t_s} \right\rceil \end{aligned} \tag{5-5}$$

Now at each the step N the system is discretized with sampling time $t_s^N = N \cdot t_s$. In this way at each step N new discrete-time system matrices A_d and B_d are obtained from the continuous-time system with system matrices A and B. In the MPT reachability analysis function the closed-loop system is then defined as:

$$\begin{aligned} A_{cl} &= A_d + B_d K \\ B_{cl} &= 0 \end{aligned} \tag{5-6}$$

From these matrices the next discrete-state can be calculated:

$$x^+ = A_{cl} x_0 \tag{5-7}$$

As this is done for all the steps N (or actually: times $t_s^N = N \cdot t_s$) where the domain for x_0 is restricted to the conic region $\mathcal{R}_s$ this will give the reachable set for the interval $[\underline{\tau}_s, \bar{\tau}_s]$ (denoted by $\mathcal{R}^s_{[\underline{\tau}_s, \bar{\tau}_s]}$) starting from the initial state x_0. The conic regions $\mathcal{R}_i$ that intersect with this set then belong to the possible transition regions of the region $\mathcal{R}_s$. Since $B_{cl} = 0$ the input does not have to be restricted in the reachability analysis steps.

The implementation of the reachability analysis as described in this chapter is successful for three-dimensional systems. For higher dimensional systems the MPT has trouble to compute the reachable sets for the conic regions $\mathcal{R}_s$. The exact reason for this is unclear.

Chapter 6

Case Studies

In this chapter some case studies on in-vehicle control systems are presented. Three linear time-invariant (LTI) systems are considered of which two are third order and one is fourth order. For each system the abstraction is performed for different values of m, the number of subdivisions for the angular coordinates in the state space partition (see chapter 3). Inter-sample time bound calculations are done using both region representations presented in chapter 3, so that the outcomes of both methods can be compared. Finally for each system a simulation is done.

6-1 Intelligent Headway Controller

6-1-1 System Description

The first control system is an intelligent vehicle headway controller [25]. The aim of this controller is to maintain a constant headway time with respect to a vehicle driving in front of the host vehicle, i.e. the vehicle that is controlled. In [25] the following state space model is derived:

$$\begin{bmatrix} \dot{E}_r \\ \dot{E}_v \\ \dot{a} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1.43 & -2.149 \end{bmatrix} \begin{bmatrix} E_r \\ E_v \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.01077 \end{bmatrix} u \tag{6-1}$$

where:

$$E_r = R_h - R \tag{6-2}$$

and:

$$E_v = V - V_p \tag{6-3}$$

Note that time indications (t) for the states are left out for simplicity, as is done for all control systems presented in this chapter. A description of all the variables in this model is given in Table 6-1. Furthermore the following optimal state feedback regulator is obtained:

$$u = -Kx = -\begin{bmatrix} 40.0125 & 55.7833 & 24.4587 \end{bmatrix} x \tag{6-4}$$

Table 6-1: Intelligent headway controller variables

Variable	Description
E_r	Distance error $[m]$
E_v	Velocity error $[\frac{m}{s}]$
R_h	Desired headway $[m]$
R	Headway $[m]$
V	Host vehicle velocity $[\frac{m}{s}]$
V_p	Preceding vehicle velocity $[\frac{m}{s}]$
a	Host vehicle acceleration $[\frac{m}{s^2}]$

6-1-2 Abstraction Results

The abstraction for the intelligent headway control system is done using both the E_s matrix method (Equation 3-3) and the Q_s matrix method (Equation 3-4). The outcome of both methods can be compared by looking at the maximum and average differences between the upper and lower inter-sample time bound for the regions:

$$\epsilon_\tau^{max} = \max\{\bar{\tau}_s - \underline{\tau}_s \mid s \in \{1, 2, \ldots, q\}\} \tag{6-5}$$

$$\epsilon_\tau^{avg} = \frac{1}{q}\sum_{s=1}^{q}(\bar{\tau}_s - \underline{\tau}_s) \tag{6-6}$$

where $q = 2 \times m^{(n-1)}$ is the number of conic regions in the state space partitioning. The tighter the inter-sample time bounds are (and thus the smaller ϵ_τ^{max} and ϵ_τ^{avg} are) the more precise the abstraction is. In Table 6-2 the values for the abstraction parameters (except m, which is varied) are given. For a detailed explanation of the abstraction procedure and these parameters see Appendix B.

In Table 6-3 an overview of ϵ_τ^{max} and ϵ_τ^{avg} obtained from both abstraction methods and for different values of m (the number of subdivisions for the angular coordinates) is given. The computation time [1] of the abstraction (excluding the reachability analysis, which requires the Q_s matrix method to be used) is given as well. In Figure 6-1 and Figure 6-2 the regional inter-sample time bounds for these abstractions are shown in descending order. Note that in these plots only the inter-sample time bounds for regions in the first half of the state space are shown since the inter-sample time bounds for regions $\mathcal{R}_s$ in the second half of the state space

[1] Computation time for abstractions performed on an Intel® Core™ i7-4710MQ processor.

Table 6-2: Abstraction and simulation parameters for the intelligent headway controller

Parameter	Value
N_{conv}	5
$\bar{\sigma}$	1 s
l	100
α	0.05
τ_{min}	0 s
$\Delta\tau_1$	0.001 s
$\Delta\tau_2$	0.001 s
$\Delta\sigma$	0.001 s
$\Delta\tau_3$	0.001 s
σ^{max}	2 s
l^*	1000
$\Delta\tau_4$	0.001 s
x_0	$[3 \quad -2 \quad 5]^T$
t_{end}	20 s
t_s	0.001 s

are identical due to the symmetry mentioned in chapter 3. Looking at Table 6-3 it stands out that the Q_s matrix method gives smaller values for ϵ_τ^{avg} and thus yields tighter inter-sample time bounds than the E_s matrix method. Also the Q_s matrix method takes less computation time. The fact that the Q_s matrix method yields tighter inter-sample time bounds can also be seen in Figure 6-1 and Figure 6-2, where the upper (lower) bounds lie below (above) those obtained from the E_s matrix method for most indices.

Table 6-3: Comparison of E_s and Q_s abstraction method results for the intelligent headway controller

Abstraction method	ϵ_τ^{max}	ϵ_τ^{avg}	Computation time
E_s, $m = 4$	1.7710 s	1.1133 s	30 min
Q_s, $m = 4$	1.3490 s	0.7209 s	19 min
E_s, $m = 10$	1.2470 s	0.4029 s	6 hrs
Q_s, $m = 10$	1.2990 s	0.2975 s	3 hrs

Table 6-4 shows an overview of ϵ_τ^{max} and ϵ_τ^{avg} obtained from abstractions using the Q_s matrix method for more values of m. Again the computation time of the abstraction is given. For larger m the computation time logically increases (since the inter-sample time bound calculations have to be performed for more conic regions) but the obtained inter-sample time bounds become tighter, i.e. the abstraction is more precise.

Figure 6-3 shows the result of a simulation done for the intelligent headway controller. During this simulation the inter-sample times and sampled states are captured. For each sampled state it can be determined in which conic region $\mathcal{R}_s$ this state lies (using Equation 3-4) and thus what the corresponding upper and lower bounds $\bar{\tau_s}$ and $\underline{\tau_s}$ on the inter-sample time (i.e. the time until the next triggering) are. The inter-sample times for triggerings that occurred during the simulation are plotted in Figure 6-4 along with the inter-sample time bounds for the region $\mathcal{R}_s$ of the corresponding sampled state. All the inter-sample times (the

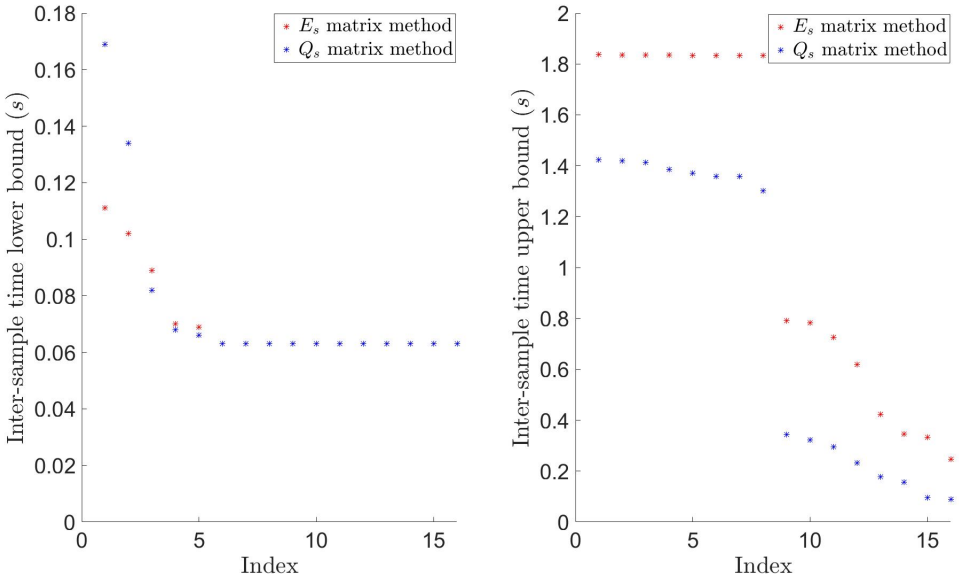

Figure 6-1: Regional inter-sample time lower bounds $\underline{\tau}_s$ (left) and upper bounds $\bar{\tau}_s$ (right) obtained from both the E_s and Q_s abstraction method for the intelligent headway controller with $m = 4$ in descending order.

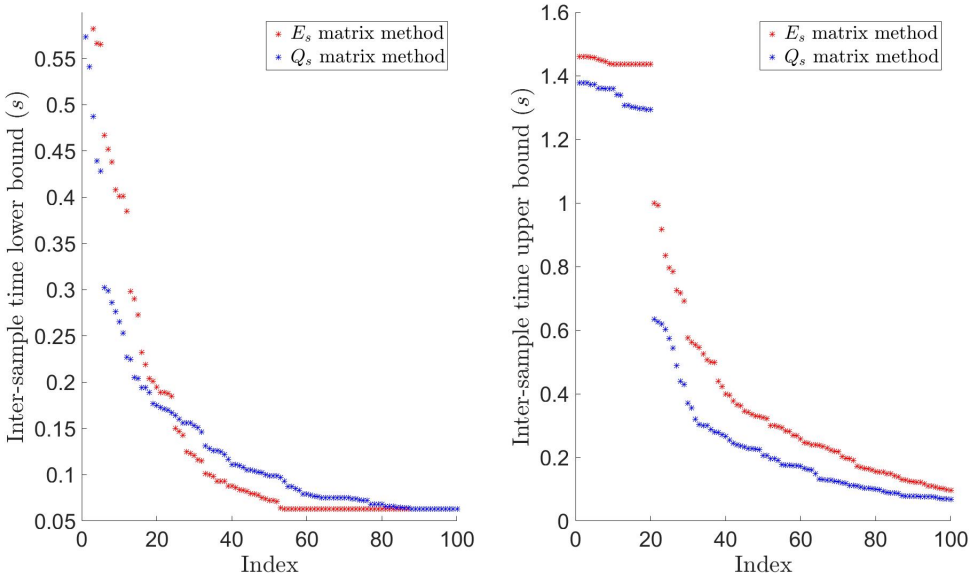

Figure 6-2: Regional inter-sample time lower bounds $\underline{\tau}_s$ (left) and upper bounds $\bar{\tau}_s$ (right) obtained from both the E_s and Q_s abstraction method for the intelligent headway controller with $m = 10$ in descending order.

Table 6-4: Comparison of Q_s abstraction method results for different values of m for the intelligent headway controller

Abstraction method	ϵ_τ^{max}	ϵ_τ^{avg}	**Computation time**
Q_s, $m = 4$	1.3490 s	0.7209 s	19 min
Q_s, $m = 6$	1.3240 s	0.4841 s	1 hr
Q_s, $m = 8$	1.3070 s	0.3675 s	1 hr 50 min
Q_s, $m = 10$	1.2990 s	0.2975 s	3 hrs

times between triggerings) fall within the calculated bounds. Figure 6-5 shows the possible transitions obtained from the reachability analysis. None of the transitions observed during the simulation violates this transition relation. The abstraction applied in this simulation was done using the Q_s matrix method with $m = 10$.

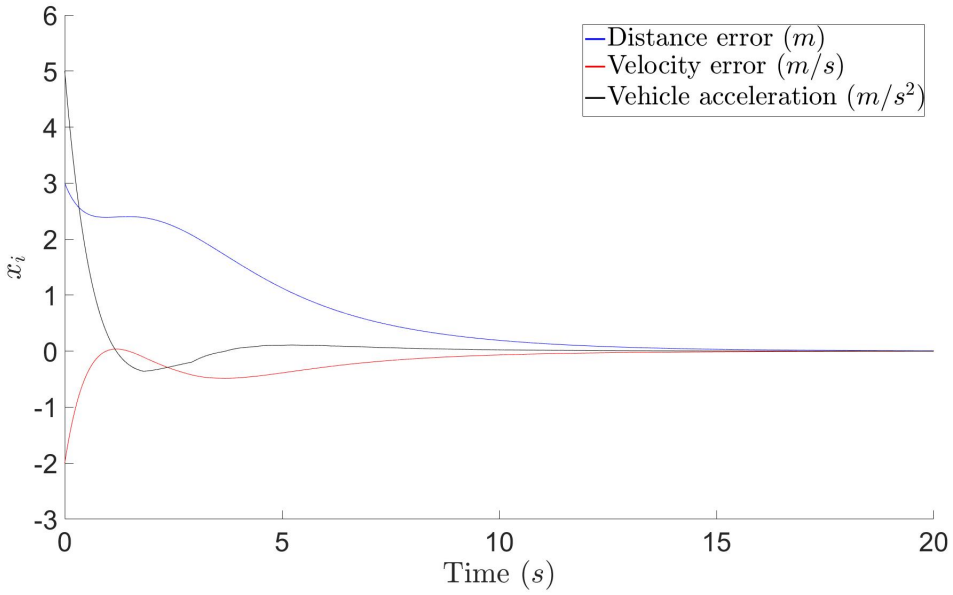

Figure 6-3: State evolution during the intelligent headway controller simulation.

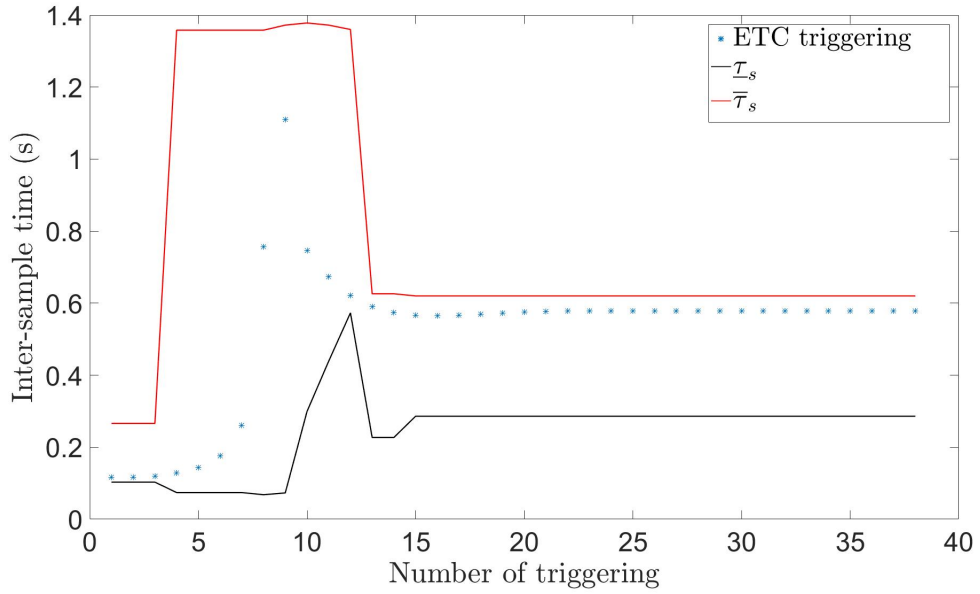

Figure 6-4: Intelligent headway controller simulation triggering sequence.

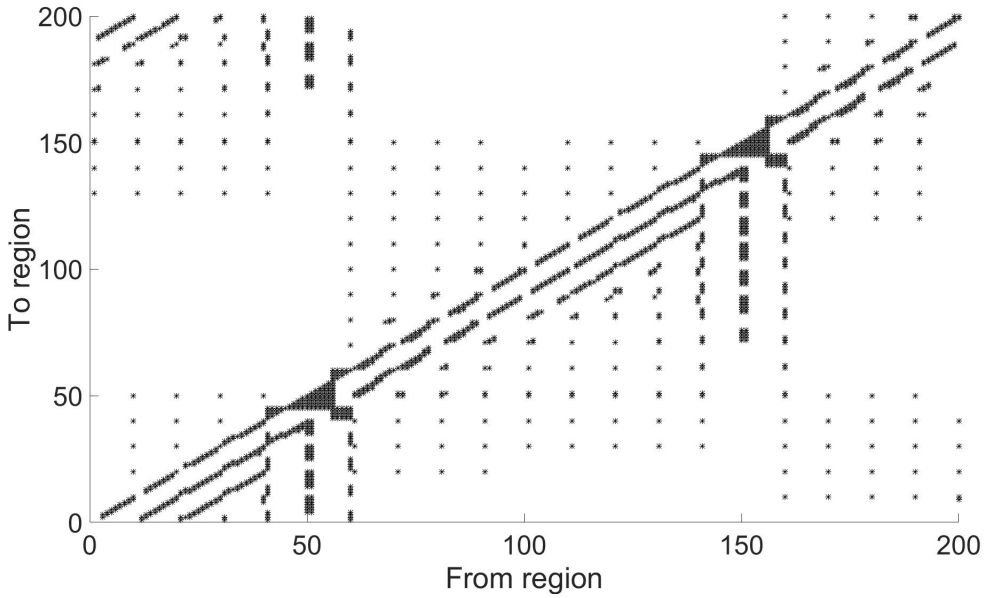

Figure 6-5: Possible region transitions for the intelligent headway controller abstraction obtained using the Q_s matrix method with $m = 10$.

6-2 Active Steering

6-2-1 System Description

The second in-vehicle control system that is considered is an active steering system. An example of such a system is found in [2]. In this study the left and right wheels are lumped together, resulting in a half car model shown in Figure 6-6. Descriptions of the parameters and variables in this model are given in Table 6-5 and Table 6-6

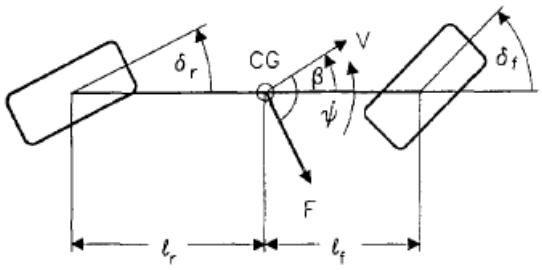

Figure 6-6: Active steering half car model (adapted from [2]).

No rear wheel steering will be used, so the rear steering angle δ_r has a constant value of 0. In [2] the following state space model is derived:

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \dot{\delta}_f \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \delta_f \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{6-7}$$

where:

$$\begin{aligned} a_{11} &= -\frac{C_f + C_r}{mv} \\ a_{12} &= -1 - \frac{C_f l_f - C_r l_r}{mv^2} \\ a_{21} &= -\frac{C_f l_f - C_r l_r}{I_{gz}} \\ a_{22} &= -\frac{C_f l_f^2 + C_r l_r^2}{I_{gz} v} \\ b_1 &= \frac{C_f}{mv} \\ b_2 &= \frac{C_f l_f}{I_{gz}} \end{aligned} \tag{6-8}$$

The moment of inertia with respect to a vertical axis I_{gz} is calculated as $I_{gz} = \frac{m}{l_f l_r}$. In this model a dry road surface condition is assumed. A wet road surface condition changes the cornering stiffness and vehicle mass influences. The vehicle mass is assumed to be 1840 kg. Assuming that the cars weight distribution is such that the center of gravity is in the middle of the front and rear axle, ℓ_f and ℓ_f are both assumed to be 2 m. A typical value for C_f and C_r is 1000 $\frac{N}{deg}$ [26] or 57300 $\frac{N}{rad}$.

Table 6-5: Active steering model parameters

Parameter	Description	Value
ℓ_f	Distance of front axle from center of gravity (CG)	2 [m]
ℓ_r	Distance of rear axle from CG	2 [m]
C_f	Front wheel cornering stiffness	57300 $[\frac{N}{rad}]$
C_r	Rear wheel cornering stiffness	57300 $[\frac{N}{rad}]$
m	Vehicle mass	1840 [kg]

Table 6-6: Active steering variables

Variable	Description
v	Velocity $[\frac{m}{s}]$
β	Sideslip angle at CG $[rad]$
$\dot{\psi}(\approx r)$	Yaw rate with respect to an inertial coordinate system $[\frac{rad}{s}]$
F	Lateral force at CG $[N]$
δ_f	Front steering angle $[rad]$

To achieve tracking of a (piecewise) constant reference yaw rate r_{ref} the system output is defined as:

$$y = Cx = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} x = r \tag{6-9}$$

The error with respect to the reference yaw rate is then:

$$\Delta y = y - y_d = r - r_{ref} \tag{6-10}$$

For asymptotic tracking it must hold that the steady-state values for x and u satisfy:

$$\begin{aligned} \lim_{t\to\infty} x &= x^* \\ \lim_{t\to\infty} u &= u^* \\ \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x^* \\ u^* \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} y_d \end{aligned} \tag{6-11}$$

Assuming the composed matrix in Equation 6-11 is nonsingular (which is the case for the chosen parameters) the steady-state values for x and u can be calculated as:

$$\begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} y_d \tag{6-12}$$

Now define the shifted states, input and output as:

$$\begin{aligned} \Delta x &= x - x^* \\ \Delta u &= u - u^* \\ \Delta y &= y - y_d \end{aligned} \tag{6-13}$$

The system can then be defined as:

$$\begin{aligned} \Delta\dot{x} &= A\Delta x + B\Delta u \\ \Delta y &= C\Delta x \end{aligned} \tag{6-14}$$

A linear-quadratic regulator (LQR) is applied to the system:

$$\Delta u = -K\Delta x \tag{6-15}$$

The matrix K is obtained using the LQR function in MATLAB [20], where the second state $r - r^* = r - r_{ref}$ is given a high weight [27].

6-2-2 Abstraction Results

Again the control system abstraction is done using both the E_s and the Q_s matrix method, with different values for m. The parameters for the abstraction are given in Table 6-7. A comparison of the obtained ϵ_τ^{max} and ϵ_τ^{avg} is given in Table 6-8. Figure 6-7 and Figure 6-8 show the inter-sample time bounds for these abstractions. Again it can be concluded that the Q_s matrix method results in tighter bounds and takes less computation time compared to the E_s matrix method.

Table 6-7: Abstraction and simulation parameters for the active steering control system

Parameter	Value
N_{conv}	5
$\bar{\sigma}$	0.001 s
l	100
α	0.05
τ_{min}	0 s
$\Delta\tau_1$	$1 \cdot 10^{-5}$ s
$\Delta\tau_2$	$1 \cdot 10^{-5}$ s
$\Delta\sigma$	$1 \cdot 10^{-5}$ s
$\Delta\tau_3$	$5 \cdot 10^{-5}$ s
σ^{max}	0.1 s
l^*	1000
$\Delta\tau_4$	$5 \cdot 10^{-5}$ s
x_0	$[0.25 \quad 0.17 \quad 0.5]^T$
t_{end}	2 s
t_s	$1 \cdot 10^{-4}$ s

Table 6-9 shows the abstraction outcomes obtained by applying the Q_s matrix method with different values for m. Again with increasing m the calculated inter-sample time bounds become tighter as could be expected. Also the computation time increases with m.

Figure 6-9 shows the state evolution during the simulation done for the active steering control system. Figure 6-10 shows the triggering sequence during this simulation. All the inter-sample

Table 6-8: Comparison of E_s and Q_s abstraction method results for the active steering control system

Abstraction method	ϵ_τ^{max}	ϵ_τ^{avg}	Computation time
E_s, $m = 4$	0.0695 s	0.0363 s	18 min
Q_s, $m = 4$	0.0468 s	0.0233 s	14 min
E_s, $m = 10$	0.0497 s	0.0117 s	45 min
Q_s, $m = 10$	0.0447 s	0.0099 s	35 min

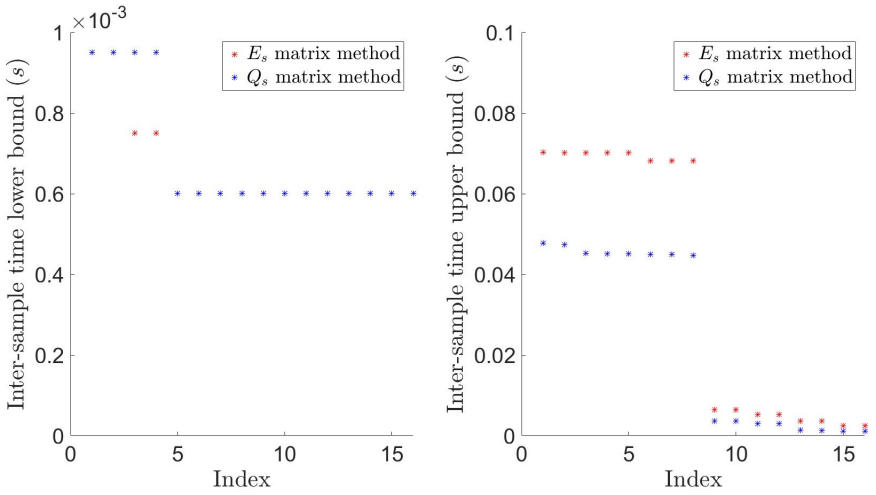

Figure 6-7: Regional inter-sample time lower bounds $\underline{\tau_s}$ (left) and upper bounds $\bar{\tau}_s$ (right) obtained from both the E_s and Q_s abstraction method for the active steering control system with $m = 4$ in descending order.

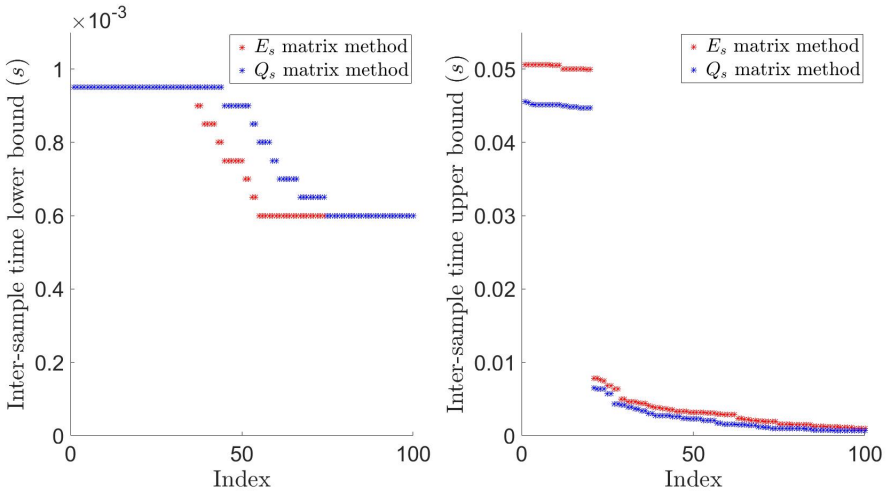

Figure 6-8: Regional inter-sample time lower bounds $\underline{\tau_s}$ (left) and upper bounds $\bar{\tau}_s$ (right) obtained from both the E_s and Q_s abstraction method for the active steering control system with $m = 10$ in descending order.

Table 6-9: Comparison of Q_s abstraction method results for different values of m for the active steering control system

Abstraction method	ϵ_τ^{max}	ϵ_τ^{avg}	Computation time
Q_s, $m = 4$	0.0468 s	0.0233 s	14 min
Q_s, $m = 6$	0.0454 s	0.0158 s	24 min
Q_s, $m = 8$	0.0449 s	0.0121 s	34 min
Q_s, $m = 10$	0.0447 s	0.0099 s	35 min

times fall within the corresponding calculated bounds. The transition relation of the system is shown in Figure 6-11. No transitions other than the ones in this transition relation occurred during the simulation.

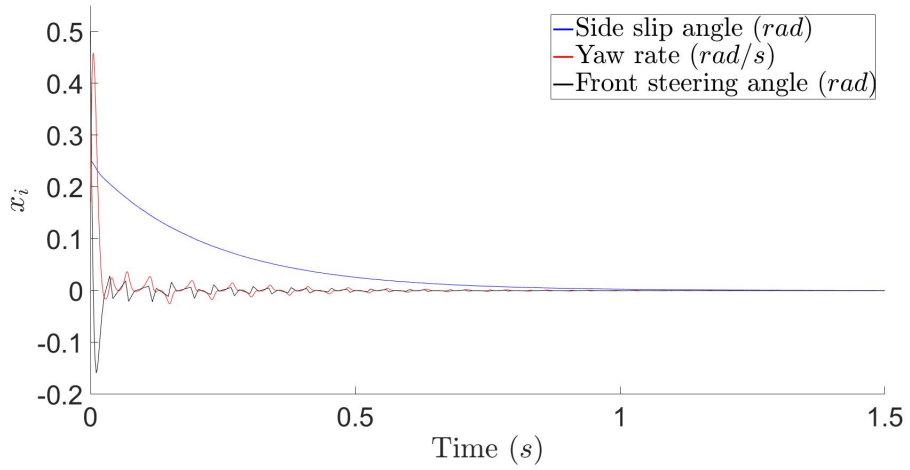

Figure 6-9: State evolution during the active steering control system simulation.

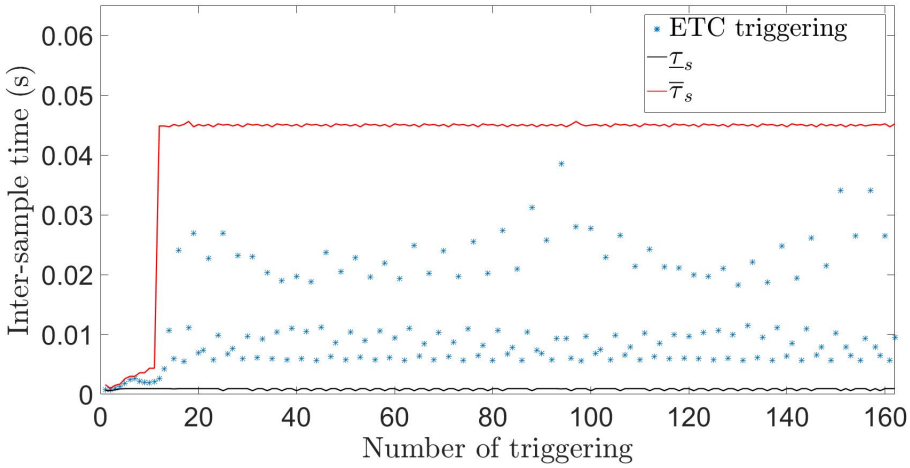

Figure 6-10: Active steering control system simulation triggering sequence.

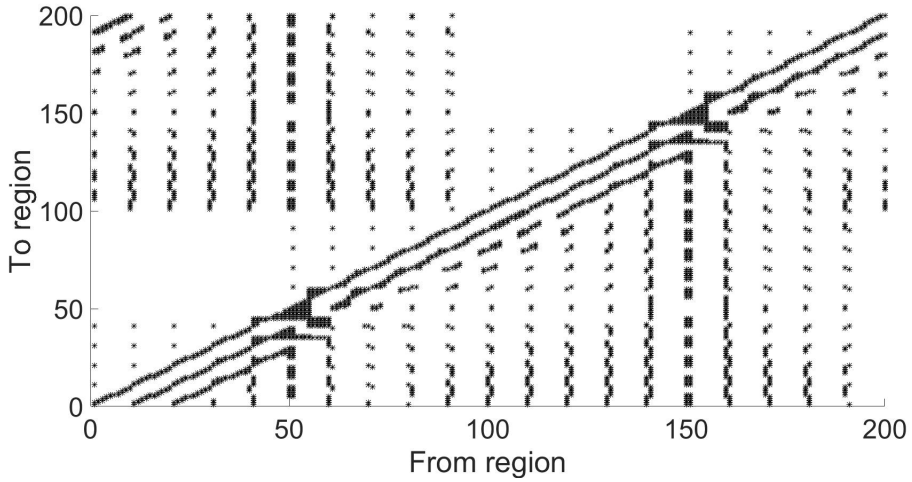

Figure 6-11: Possible region transitions for the active steering control system abstraction obtained using the Q_s matrix method with $m = 10$.

6-3 Active Suspension

6-3-1 System Description

The last control system that is considered is an active suspension. Active suspension improves road handling and passenger comfort [3]. A quarter car model in the form of a double mass-spring-damper system (Figure 6-12) is considered.

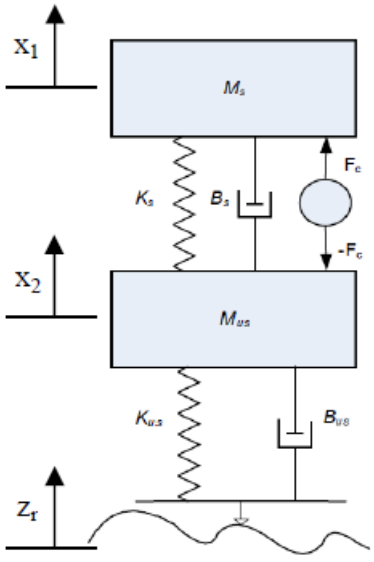

Figure 6-12: Active suspension on a quarter car model (adapted from [3]).

The parameters in this model are defined in Table 6-10. Their values are taken from [28].

Table 6-10: Active suspension model parameters

Parameter	Description	Value
M_s	Vehicle body mas (or sprung mass)	460 [kg]
M_{us}	Tire mass (or unsprung mass)	50 [kg]
k_s	Suspension spring constant	29500 [N/m]
b_s	Suspension damping coefficient	290 [Ns/m]
k_{us}	Tire spring constant	215000 [N/m]
b_{us}	Tire damping coefficient	1500 [Ns/m]

The equations of motions for the quarter car model are given by[2]:

$$M_s\ddot{x}_1 = F_c - b_s\Big(\dot{x}_1 - \dot{x}_2\Big) - k_s\Big(x_1 - x_2\Big) \tag{6-16}$$

$$\begin{aligned} M_{us}\ddot{x}_2 = -F_c + b_s\Big(\dot{x}_1 - \dot{x}_2\Big) + k_s\Big(x_1 - x_2\Big) \\ -b_{us}\Big(\dot{x}_2 - \dot{z}_r\Big) - k_{us}\Big(x_2 - z_r\Big) \end{aligned} \tag{6-17}$$

By choosing a state vector x, control input u and disturbance w as:

$$\begin{aligned} x &= \begin{bmatrix} x_1 - x_2 & \dot{x}_1 & x_2 - z_r & \dot{x}_2 \end{bmatrix}^T \\ u &= F_c \\ w &= \dot{z}_r \end{aligned} \tag{6-18}$$

The state space description of the system is given by:

$$\dot{x} = Ax + Bu + Gw \tag{6-19}$$

where:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 & -1 \\ -\frac{k_s}{M_s} & -\frac{b_s}{M_s} & 0 & \frac{b_s}{M_s} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{M_s} & \frac{b_s}{M_s} & -\frac{k_{us}}{M_s} & -\frac{b_s}{M_s} \end{bmatrix} \\ B &= \begin{bmatrix} 0 & \frac{1}{M_s} & 0 & -\frac{1}{M_{us}} \end{bmatrix}^T \\ G &= \begin{bmatrix} 0 & 0 & -1 & \frac{b_{us}}{M_{us}} \end{bmatrix}^T \end{aligned} \tag{6-20}$$

An LQR is applied to the system:

$$u = F_c = -Kx \tag{6-21}$$

[2]The derivations of the equations of motion in [3] and [28] contain some errors, which are corrected here. The controller design is redone as well.

where the matrix K is again obtained using the LQR function in MATLAB [27]. For this system the state $x_1 - x_2$ is given a high weight. This state is related to road handling quality [3]. Note that the presence of a disturbance term (in this case Gw) is not considered in [1]. A disturbance however could be represented by picking an initial condition other than the equilibrium of the system, as is done in this case study.

6-3-2 Abstraction Results

Since the MATLAB script [21] that generates the matrices E_s does not work for systems with $n > 3$ the E_s matrix abstraction method cannot be applied to this system. The abstraction is only done using the Q_s matrix method with different values for m. The abstraction parameters are given in Table 6-11. An overview of the obtained ϵ_τ^{max} and ϵ_τ^{avg} for the different abstractions is given in Table 6-12.

Table 6-11: Abstraction and simulation parameters for the active suspension control system

Parameter	**Value**
N_{conv}	5
$\bar{\sigma}$	0.001 s
l	100
α	0.05
τ_{min}	0 s
$\Delta\tau_1$	$1 \cdot 10^{-5}$ s
$\Delta\tau_2$	$1 \cdot 10^{-5}$ s
$\Delta\sigma$	$1 \cdot 10^{-5}$ s
$\Delta\tau_3$	$5 \cdot 10^{-5}$ s
σ^{max}	0.25 s
l^*	1000
$\Delta\tau_4$	$1 \cdot 10^{-4}$ s
x_0	$[0.1 \quad 2 \quad -0.05 \quad -5]^T$
t_{end}	5 s
t_s	$5 \cdot 10^{-4}$ s

Table 6-12: Comparison of Q_s abstraction method results for different values of m for the active suspension control system

Abstraction method	$\boldsymbol{\epsilon_\tau^{max}}$	$\boldsymbol{\epsilon_\tau^{avg}}$	**Computation time**
Q_s, $m = 4$	0.2190 s	0.0376 s	49 min
Q_s, $m = 6$	0.2140 s	0.0175 s	1 hr 31 min
Q_s, $m = 8$	0.2104 s	0.0100 s	3 hrs 28 min
Q_s, $m = 10$	0.2071 s	0.0067 s	6 hrs 39 min

In Figure 6-13 the state evolution during the simulation done for the active suspension control system is plotted. The triggering sequence during this simulation is shown Figure 6-14. Again all the inter-sample times fall within the corresponding calculated bounds. Since the reachability analysis is not working for four-dimensional (and higher dimensional) systems

the region transitions are not shown here.

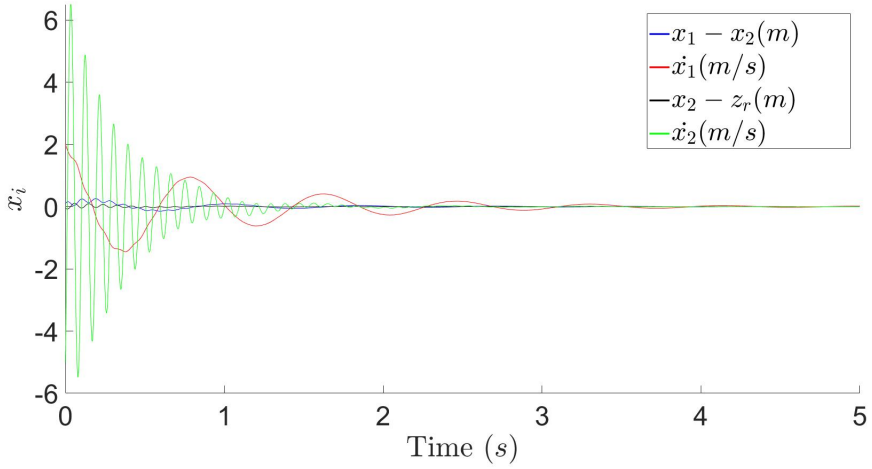

Figure 6-13: State evolution during the active suspension control system simulation.

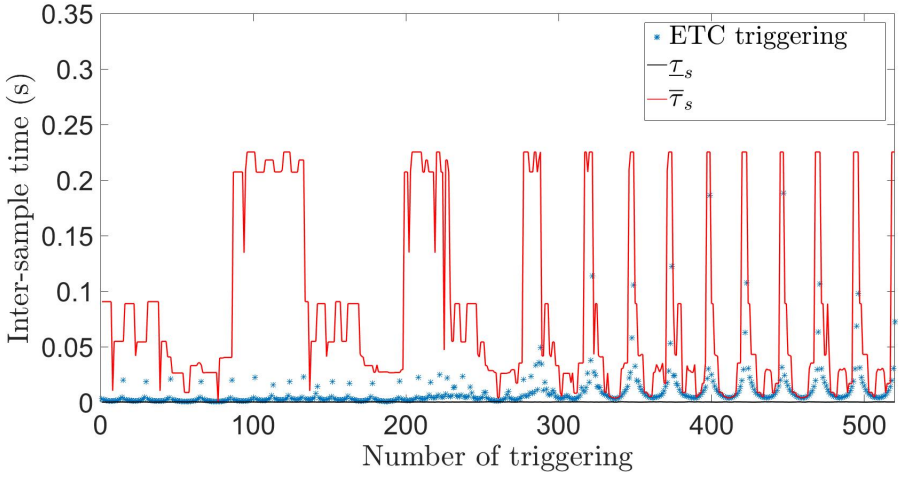

Figure 6-14: Active suspension control system simulation triggering sequence.

Chapter 7

Conclusion

The abstraction method as laid out in chapter 2 is implemented in MATLAB [20], [27] and applied to event-triggered linear time-invariant (LTI) control systems. The abstraction partitions the state space into conic regions $\mathcal{R}_s$ pointing at the origin. For each of these conic regions an upper bound $\bar{\tau}_s$ and lower bound $\underline{\tau}_s$ on the inter-sample time are derived. Finally a reachability analysis is implemented to determine the possible transitions between regions. Altogether this defines the control system timed game automaton (TGA). By taking the parallel composition of multiple control system TGA and a communication network TGA a network of timed game automata (NTGA) could be obtained over which a scheduling policy can be synthesized.

From the case studies in chapter 6 it can be concluded that the inter-sample time bound computations done using the Q_s matrix method (based on the region representation as given in Equation 3-4) give good results. When implemented in MATLAB this method yields less conservative inter-sample time bounds and takes less computation time than the E_s matrix method (based on the region representation as given in Equation 3-3). Moreover the Q_s matrix method is able to handle four-dimensional and higher-dimensional systems.

The reachability analysis [27] (performed using the Multi-Parametric Toolbox (MPT) in MATLAB) is only working for two-dimensional and three-dimensional systems. When applied to higher-dimensional systems the MPT cannot perform the computation of the reachable sets for the conic regions $\mathcal{R}_s$ (in which the state space is partitioned) and gives an error. The reason for this issue is unclear.

Chapter 8

Further Work

In order to be able to obtain complete abstractions for n-dimensional systems (with $n > 3$) the issue that occurs in the reachability analysis has to be solved. Since the exact issue is unclear it is hard to propose an approach to do so. Perhaps an alternative way to define the region polyhedra in the MATLAB code [27] could solve the problem. Another option is to program the reachability analysis manually. This latter approach has been tried based on the method presented in [19] but was unsuccessful. The difficulty here lies in making an appropriate over-approximation of the reachable set.

To complete the scheduling approach proposed in [1] the control system timed game automata (TGA) have to be defined based on the abstraction results. Taking the parallel composition of these TGA and the communication network gives the network of timed game automata (NTGA) over which a scheduling could be synthesized. This can be done by using dedicated software such as UPPAAL-Tiga [11]. After this the performance of the networked control systems (NCSs) could be evaluated by performing simulations in a vehicle simulation environment such as Dynacar [29].

To decrease the computation time (which increases significantly with the system order n) of the abstraction a bisection method could be applied in the inter-sample time bound calculations. Since the abstraction is performed offline its computation time does not influence the real-time performance of the NCSs but still the reduction in computation time could be worthwhile, especially for higher dimensional systems.

Appendix A

Appendix A

A-1 Proof to Lemma 1

Proof of Lemma 1: We provide a constructive proof in four steps.

First: divide the time interval $[0, \bar{\sigma}]$ into l subintervals. The aim of this step is to make the preparations to compute a precise estimation of $\Phi(\cdot)$ by building l small convex embeddings around it instead of building an overly conservative single embedding. For every time instant $\sigma \in [0, \underline{\tau}_s]$, there exists $j \in \{0, \ldots, \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor\}$ such that $j\frac{\bar{\sigma}}{l} \leq \sigma \leq (j+1)\frac{\bar{\sigma}}{l}$. Define $\sigma' = \sigma - j\frac{\bar{\sigma}}{l}$ ($\sigma' \in [0, \chi]$, with $\chi = \frac{\bar{\sigma}}{l}$ for $j < \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor$ and $\chi = \underline{\tau}_s - j\frac{\bar{\sigma}}{l}$ otherwise).

Second: compute a polynomial approximation of $\Phi(\cdot)$ over each subinterval. Employ the relation $\int_0^{a+b} e^{Ar}dr = \int_0^a e^{Ar}dr + \int_0^b e^{Ar}dr(A\int_0^a e^{Ar}dr + I)$ to simplify $\Lambda(\sigma)$ given in (4-7) as:

$$\Lambda(\sigma) = I + M_j(A + BK) + \int_0^{\sigma'} e^{Ar}dr N_j(A + BK) \tag{A-1}$$

with M_j and N_j as in (4-17). Then, by defining two new matrices $\Pi_{1,j}$ and $\Pi_{2,j}$ as in (4-16), equation (A-1) can be rewritten as:

$$\Lambda(\sigma) = \Pi_{1,j} + \int_0^{\sigma'} e^{Ar}dr\Pi_{2,j}. \tag{A-2}$$

Replace $\int_0^{\sigma'} e^{Ar}dr$ by its $N_{conv} - th$ order Taylor series expansion to approximate Φ given by (4-9), i.e.:

$$\int_0^{\sigma'} e^{Ar}dr \simeq \sum_{i=1}^{N_{conv}} \frac{A^{i-1}}{i!}\sigma'^i. \tag{A-3}$$

Remember that $N_{conv} + 1$ is the number of vertices we consider for polytopic embedding according to time. The Taylor series expansion of Φ is given by $\sum_{k=0}^{\infty} L_{k,j}\sigma'^k$ with $L_{k,j}$ defined in (4-15).

Third: bound the error introduced by the N_{conv}-th order approximation of Φ with a constant term. One can derive the N_{conv}-th order approximation of Φ on the time interval $[j\frac{\bar{\sigma}}{l}, (j+1)\frac{\bar{\sigma}}{l}]$ using $\tilde{\Phi}_{N_{conv},j}(\sigma')$ given in (4-19). Denote by $R_{N_{conv},j}(\sigma') = \Phi(\sigma) - \tilde{\Phi}_{N_{conv},j}(\sigma')$. We seek to compute a constant scalar $\underline{\nu}$ independent of σ' such that $R_{N_{conv},j}(\sigma') \preceq \underline{\nu} I$, to establish $x^T\Phi(\sigma)x \leq 0$ from $x^T(\tilde{\Phi}_{N_{conv},j}(\sigma') + \underline{\nu} I)x \leq 0$. Since $R_{N_{conv},j}$ is symmetric $R_{N_{conv},j}(\sigma') \preceq \lambda_{\max}(\sigma')I$, where $\lambda_{\max}(\sigma')$ is the maximal eigenvalue of $R_{N_{conv},j}(\sigma')$, and thus $\underline{\nu}$ is provided by (4-18).

Fourth: build a convex polytope that contains the matrix exponential function $\tilde{\Phi}_{N_{conv},j} + \underline{\nu} I : [0, \chi] \to \mathcal{M}_n(\mathbb{R})$, using the method proposed in [18]. From [18] we know that if $x^T\underline{\Phi}_{(i,j),s}x \leq 0,\ \forall (i,j) \in \mathcal{K}_s = (\{0, \ldots, N_{conv}\} \times \{0, \ldots, \lfloor \frac{\underline{\tau}_s l}{\bar{\sigma}} \rfloor\})$, with $\underline{\Phi}_{(i,j),s} = \sum_{k=0}^{i} L_{k,j}\chi^k + \underline{\nu} I$, the following holds $x^T(\tilde{\Phi}_{N_{conv},j}(\sigma') + \underline{\nu} I)x \leq 0$ and as a result $x^T\Phi(\sigma)x \leq 0,\ \forall \sigma \in [0, \underline{\tau}_s]$. □

A-2 Proof to Lemma 2

Proof of Lemma 2: The proof of Lemma 2 is analogous to the proof of Lemma 1 with the following changes. In the first step now: for every $\sigma \in [\bar{\tau}_s, \bar{\sigma}]$, $\exists j \in \{\lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \ldots, l-1\}$ such that $j\frac{\bar{\sigma}}{l} \leq \sigma \leq (j+1)\frac{\bar{\sigma}}{l}$, we define $\sigma' = \sigma - j\frac{\bar{\sigma}}{l}$ ($\sigma' \in [0, \chi]$, with $\chi = (j+1)\frac{\bar{\sigma}}{l} - \bar{\tau}_s$ for $j = \lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor$ and $\chi = \frac{\bar{\sigma}}{l}$ for $j > \lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor$). In the third step, now we seek to compute instead a constant scalar $\bar{\nu}$ independent of σ' such that $R_{N_{conv},j} \succeq \bar{\nu} I$, to establish $x^T\Phi(\sigma)x \geq 0$ from $x^T(\tilde{\Phi}_{N_{conv},j}(\sigma') + \bar{\nu} I)x \geq 0$. Since $R_{N_{conv},j}$ is symmetric, it follows $R_{N_{conv},j}(\sigma') \succeq \lambda_{\min}(\sigma')I$, where $\lambda_{\min}(\sigma')$ is the minimal eigenvalue of $R_{N_{conv},j}(\sigma')$. This constant can be computed now from (4-23). In the fourth step, applying again the results from [18], one has that $x^T\bar{\Phi}_{(i,j),s}x \geq 0,\ \forall (i,j) \in \mathcal{K}_s = (\{0, \ldots, N_{conv}\} \times \{\lfloor \frac{\bar{\tau}_s l}{\bar{\sigma}} \rfloor, \ldots, l-1\})$, with $\bar{\Phi}_{(i,j),s} = \sum_{k=0}^{i} L_{k,j}\chi^k + \bar{\nu} I$, implies $x^T(\tilde{\Phi}_{N_{conv},j}(\sigma') + \bar{\nu} I)x \geq 0$ and consequently $x^T\Phi(\sigma)x \geq 0,\ \forall \sigma \in [\bar{\tau}_s, \bar{\sigma}]$. □

Appendix B

Appendix B

B-1 Abstraction Procedure in MATLAB

The entire control system abstraction is implemented in MATLAB [20]. MATLAB scripts that performed the abstraction for two-dimensional systems was already available [21]. For this project this existing code was modified and extended to make it work for higher dimensional systems as well.

The abstraction procedure consists of several steps which are laid out below. The most convenient is to run these steps in sequence from a main running script, were the control system matrices and abstraction parameters (see Table B-1) are defined.

Step 1: Line search on $\underline{\tau}'$

In this step, which is performed by running the script 'step1_etc.m' a line search on a lower bound on the inter-sample time bound for the entire state space is done. This is done by finding a finite set of matrices $\underline{\Phi}_{\kappa,s}$ (see section 4-1) such that:

$$\underline{\Phi}_{\kappa,s} \preceq 0 \tag{B-1}$$

In this step $\underline{\nu}$ (Equation 4-18) is not taken into account yet.

Step 2: Deriving $\underline{\nu}$ and new line search on $\underline{\tau}'$

In step 2 $\underline{\nu}$ (Equation 4-18) is derived, after which the line search on $\underline{\tau}'$ is redone, now with taking $\underline{\nu}$ into account. The script 'step2_Nu_etc.m' performs the calculations for this step.

Step 3: Calculating a lower bound on the inter-sample time for each region

In step 3 the calculations to find a lower bound on the inter-sample time per region are performed. This is done using the script 'step3_nD_Q_etc_lowerbounds.m' (if the region representation from Equation 3-4 is used) or 'step3_nD_E_etc_lowerbounds.m' (if the region representation from Equation 3-3 is used).

Step 4: Calculating an upper bound on the inter-sample time for each region

In this step the calculations to find an upper bound on the inter-sample time per region are performed. The calculations for this step are performed by the script 'step4_nD_Q_etc_upperbounds.m' (if the region representation from Equation 3-4 is used) or 'step4_nD_E_etc_upperbounds.m' (if the region representation from Equation 3-3 is used).

Step 5: Reachability analysis

In this final step (performed by the script 'step5_nD_Reachability') of the system abstraction the reachability analysis as described in chapter 5 using the Multi-Parametric Toolbox (MPT) is done. Since the calculations of the region vertices in this step are consistent with the region representation from Equation 3-4, this reachablility analysis only works if the region representation from this equation is used.

B-2 Simulation

A simulation for an example system can be done by running the script 'fig_Sim_Q_etc' (if the region representation from Equation 3-4 is used) or 'fig_Sim_E_etc' (if the region representation from Equation 3-3 is used). The initial condition, time step size and end time for the simulation can be defined in the main running script. During the simulation the state evolution, measurement error and inter-sample times are saved. At the end it can be evaluated if the inter-sample times lie within the calculated bounds. Also it can be checked if all the region transitions that occur during the simulation were predicted to be possible by the reachability analysis.

B-3 Abstraction and Simulation Parameters

For the abstraction and simulation some parameters have to be defined. As mentioned above the most convenient is to define these in a main script in MATLAB, along with the control system matrices. An overview of all the abstraction and simulation parameters is given in Table B-1.

Table B-1: Abstraction and simulation parameters

Parameter	Description
N_{conv}	Order of Taylor series approximation of Φ (see section 4-1)
$\bar{\sigma}$	Upper limit for the line search in step 1,2 and 3
l	Number of subdivisions in the interval $[0, \bar{\sigma}]$
m	Number of subdivisions for the angular coordinates (within an interval of length π)
α	Triggering coefficient (see Equation 2-4)
τ_{min}	Starting time for line search in step 1
$\Delta\tau_1$	Time step size for the line search in step 1
$\Delta\tau_2$	Time step size for the line search in step 2
$\Delta\sigma$	Increment on σ' in step 2 (see Equation 4-18)
$\Delta\tau_3$	Time step size for the line search in step 3
σ^{max}	Upper limit for the line search in step 4
l^*	Number of subdivisions in the interval $[0, \sigma^{max}]$
$\Delta\tau_4$	Time step size for the line search in step 4
x_0	Initial condition for the simulation
t_{end}	Simulation end time
t_s	Simulation time step size

Bibliography

[1] D. Adzkiya and M. Mazo Jr, "Scheduling of event-triggered networked control systems using timed game automata," *arXiv preprint arXiv:1610.03729*, 2016.

[2] J. Ackermann, "Robust car steering by yaw rate control," in *Proc. 29th Conf. on Decision and Control*, pp. 2033 – 2034, December 1990.

[3] S. Kamal and K. G. Namboothiri, "Disturbance rejection and feedback control algorithm for active suspension plant," in *Proc. Int. Conf. on Control, Commun. and Computing (ICCC)*, pp. 253–258, December 2013.

[4] K. Pretz, "Fewer wires, lighter cars." http://theinstitute.ieee.org/resources/standards/fewer-wires-lighter-cars, Apr. 2013. [Online].

[5] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems* (D. Hristu-Varsakelis and W. S. Levine, eds.), ch. Vehicle Applications of Controller Area Network, pp. 741–765, Boston, Massachusetts, USA: Birkhäuser, 2005.

[6] S. Corrigan, "Introduction to the controller area network (CAN)," tech. rep., Texas Instruments, Dallas, Texas, August 2002.

[7] R. Makowitz and C. Temple, "Flexray – a communications network for automotive control systems," in *"Proc. IEEE Int. Workshop Factory Commun. Sys."*, (Torino, Italy), pp. 207–212, June 2006.

[8] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," *Information and Computation*, vol. 111, no. 2, pp. 193–244, 1994.

[9] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *E.W. Mayr and C. Puech (Eds), Proc. STACS'95, LNCS 900*, pp. 229–242, Springer, 1995.

[10] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[11] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *IN CONCUR 05, LNCS 3653*, pp. 66–80, Springer, 2005.

[12] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Trans. Autom. Control*, vol. 52, pp. 1680–1685, September 2007.

[13] A. D. Ames, P. Tabuada, and S. Sastry, "On the stability of zeno equilibria.," in *HSCC* (J. P. Hespanha and A. Tiwari, eds.), vol. 3927 of *Lecture Notes in Computer Science*, pp. 34–48, Springer, 2006.

[14] K. Astrom and B. Bernhardsson, "Comparison of riemann and lebesgue sampling for first order stochastic systems," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2, pp. 2011 – 2016 vol.2, dec. 2002.

[15] A. S. Kolarijani, D. Adzkiya, and M. Mazo Jr, "Symbolic abstractions for the scheduling of event-triggered control systems," in *Proc. 54th IEEE Conf. Decision and Control*, (Osaka, Japan), December 2015.

[16] A. S. Kolarijani and M. Mazo Jr, "A formal traffic characterization of lti event-triggered control systems," *arXiv preprint arXiv:1503.05816*, 2015.

[17] C. Fiter, L. Hetel, W. Perruquetti, and J.-P. Richard, "A state dependent sampling for linear state feedback," *Automatica*, vol. 48, no. 8, pp. 1860–1867, 2012.

[18] L. Hetel, J. Daafouz, and C. Iung, "Lmi control design for a class of exponential uncertain systems with application to network controlled switched systems," in *2007 American Control Conference*, pp. 1401–1406, IEEE, 2007.

[19] A. Chutinan and B. H. Krogh, "Computing polyhedral approximations to flow pipes for dynamic systems," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 2, pp. 2089–2094, IEEE, 1998.

[20] "MATLAB - MathWorks." https://nl.mathworks.com/products/matlab/.

[21] A. S. Kolarijani and D. Adzkiya, "ETC MATLAB code."

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[23] "Yalmip." https://yalmip.github.io/. [Online].

[24] "Multi-parametric toolbox 3." http://people.ee.ethz.ch/~mpt/3/. [Online].

[25] L. Bin, W. Rongben, and C. Jiangwei, "A new optimal controller for intelligent vehicle headway distance," in *IEEE Intelligent Vehicle Symposium*, June 2002.

[26] N. D. Smith, "Understanding parameters influencing tire modeling," tech. rep., Colorado State University, 2004. Formula SAE Platform.

[27] C. Hop, "ETC MATLAB code (updated)."

[28] M. M. Bello, A. A. Shafie, and R. Khan, "Active vehicle suspension control using full state-feedback controller," *Advanced Materials Research*, vol. 1115, pp. 440–445, 2015.

[29] "Dynacar: advanced research full electric vehicle." http://www.dynacar.es/en/home.php.

Glossary

List of Acronyms

ADAS	advanced driver assistance systems
ECU	electronic control unit
NCS	networked control system
CAN	controller area network
CSMA	carrier sense multiple access
LTI	linear time-invariant
TA	timed automaton
TSA	timed safety automaton
TGA	timed game automaton
NTGA	network of timed game automata
LQR	linear-quadratic regulator
CG	center of gravity
LMI	linear matrix inequality
MPT	Multi-Parametric Toolbox

List of Symbols

α	Triggering coefficient
$\bar{\tau}_s$	Inter-sample time upper bound

$\tau(x)$	Inter-sample time
$\underline{\tau}_s$	Inter-sample time lower bound
ϵ_τ^{avg}	Abstraction average upper and lower inter-sample time bound difference
ϵ_τ^{max}	Abstraction maximum upper and lower inter-sample time bound difference
θ_i	Angular coordinate
$\xi(t)$	LTI system state
$\mathcal{R}^s_{[\underline{\tau}_s, \bar{\tau}_s]}$	Reachable set
$\mathcal{R}_s$	Conic region
$e(t)$	Measurement error
E_s	N-dimensional conic region matrix
m	Number of angular coordinate subdivisions
Q_s	Two-dimensional conic region matrix
x	LTI system sampled state

Index