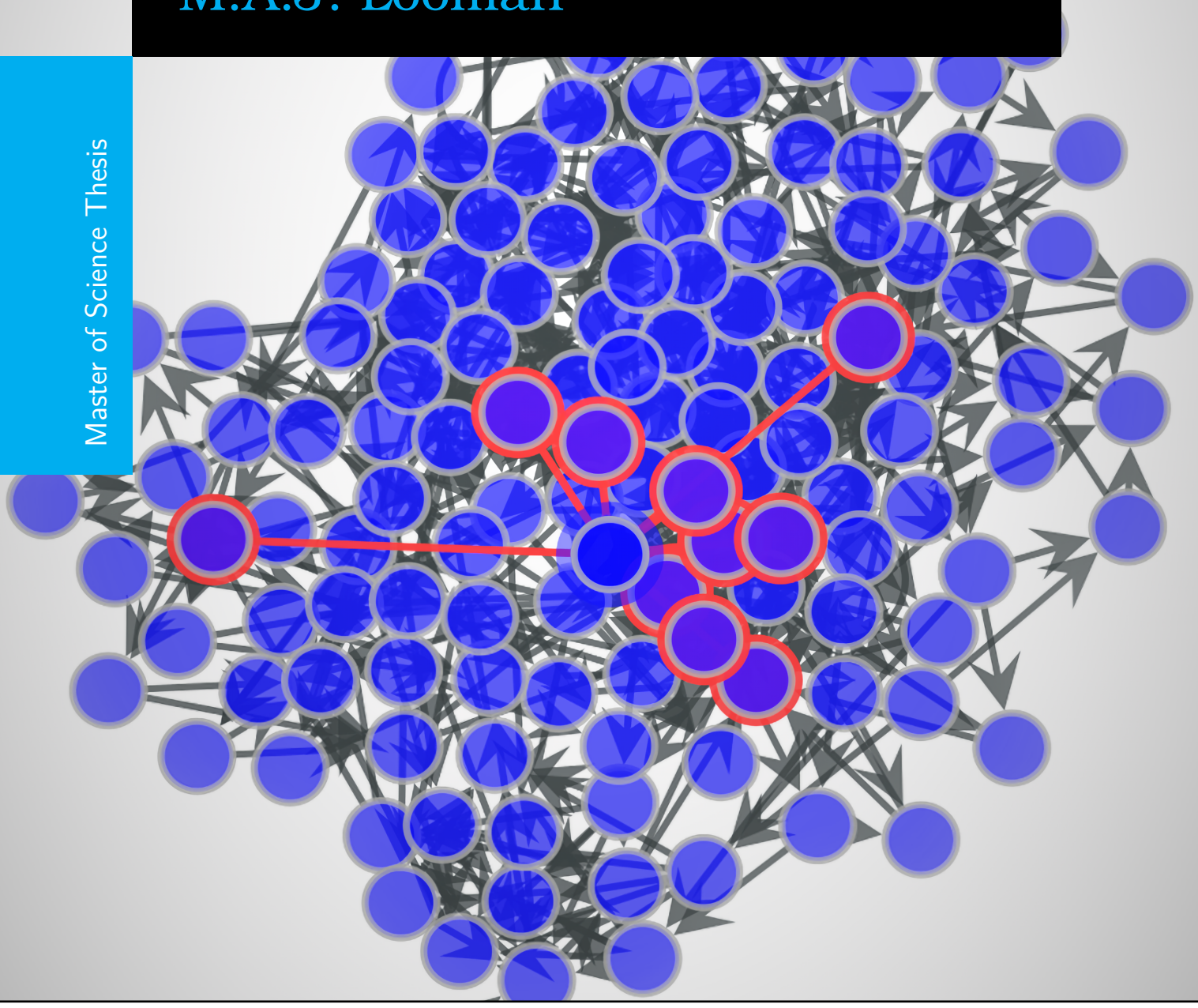
Scheduling of Periodic Event-Triggered Control to balance Control Performance and Average Inter-Sample Times

M.A.J. Looman

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Scheduling of Periodic Event-Triggered Control to balance Control Performance and Average Inter-Sample Times

Master of Science Thesis

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

M.A.J. Looman

August 16, 2023

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

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

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

Scheduling of Periodic Event-Triggered Control to balance Control Performance and Average Inter-Sample Times

by

M.A.J. Looman

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: August 16, 2023

Supervisor(s):

dr.ir. Manuel Mazo Jr.

Reader(s):

Dr.ir. K. Batselier

Abstract

Traditionally, Event-Triggered Control (ETC) methods are sample-and-hold control schemes that implement a triggering condition in order to reduce the number of control updates. Given a decay rate of the Lyapunov function, they focus on minimizing the (average) Inter-Sample Time (IST). In this thesis, we focused on the scheduling of Periodic Event-Triggered Control (PETC) controllers. By dynamically switching between triggering conditions, we are maximizing the average rate of decay of the Common Lyapunov Function (CLF) given a minimum Average Inter Sample Time (AIST) $\hat{\tau}$ or burst condition.

Given the physical system S, we construct a switched system $S_{\mathcal{Q}}$ which captures all possible scheduling behaviors. The l-complete abstraction of the switched system $S_{\mathcal{Q}}$ is constructed by solving a conjunction of quadratic equations and is denoted by $S_{\mathcal{Q}/Q_l}$, with equivalence relation Q_l. By setting a minimum AIST $\hat{\tau}$ or burst condition, a set of states in $S_{\mathcal{Q}/Q_l}$ is marked and a safety game is played to construct the maximal permissive controller S_{mpc}.

On the safe behaviors inside S_{mpc}, the guaranteed minimum control performance $\hat{\sigma}$ is maximized for the infinite horizon problem, i.e. by maximizing the minimum weighted time average of the primitive cycles in S_{mpc}. First, several energy games are played to estimate the maximum value of $\hat{\sigma}$. Thereafter, a mean-payoff game is played to generate the strategy securing this maximum control performance $\hat{\sigma}$, which is used to construct the controller S_{ihc}.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor dr.ir. Manuel Mazo Jr. for his assistance and guidance during the time I was a part of the research group. Many times, I thought I had a grasp of the subject, but the challenging discussions would broaden my understanding.

I would like to thank everyone that was a part of the research group for the insights and questions during the weekly meetings. Specifically dr. G. Gleizer for his expertise in the subject and for always being willing to help if I had questions.

Lastly, I would like to thank my friends and family as well for supporting me during my thesis. The master thesis is a period that tests your expectations with reality while progress can be limited at times. The past period was difficult sometimes, but you would give me the motivation to continue.

Delft, University of Technology M.A.J. Looman
August 16, 2023

"A mathematical problem should be difficult in order to entice us, yet not completely inaccessible, lest it mock at our efforts. It should be to us a guide post on the mazy paths to hidden truths, and ultimately a reminder of our pleasure in the successful solution." — *David Hilbert*

Introduction

Control systems are nowadays present in a variety of applications. From an old example from the 19th century as how to keep course on a ship [1], to evolved control processes that can lift a boat out of the water using hydrofoils [2]. This was all due to the invention of the computer, which made it possible to run more complex algorithms and advanced control schemes. However, using a computer to determine the control action also has some disadvantages. Due to the computations, there will always be a delay between the measurement and control action.

Here the types of control methods divergences into different fields. One of these fields uses triggering events to determine if the control action needs to be revised [3, 4, 5, 6], i.e. ETC. The benefit of ETC is that the controller does not need to alter the control action every time instant, only if certain control bounds are exceeded (i.e. triggering condition is triggered). This can reduce the number of calculations of the controller and the amount of communication between the controller and the actuators. Which can be very beneficial when operating on Networked Control System (NCS) where the bandwidth is limited.

Existing ETC methods focus on reducing the number of communications as much as possible while still stabilizing the system. This can be done in different manners and focuses on the goal of reducing communications [7, 8]. One can set a custom desired guaranteed control performance. However, setting this too high can result in an unwanted amount of communications for some states in the state-space. If one neglects the control performance, this will result in very slow guaranteed convergence of the systems.

In recent years, the knowledge about ETC and PETC methods has increased. The insight with the traffic models and Strongest (Asynchronous) l-Complete Approximations (SAlCA) has opened new possibilities to analyze and construct triggering conditions for these methods and the introduction of Self-Triggered Control (STC) [9, 10, 11, 12].

In this thesis project we will seek to strike a balance between communication reduction and control performance of PETC methods. For this we will limit the research on Linear Time-Invariant (LTI) systems with a predetermined static feedback gain. The PETC will be suited with multiple quadratic triggering conditions with known guaranteed decay rates of the CLF. Our goal will be to design a scheduler which can switch between a fixed set of quadratic triggering conditions for the PETC system in order to achieve the desired balance.

0-1 Notation

We denote by $\mathbb{N}$ the set of natural numbers, a small extension being $\mathbb{N}_0$ which also includes 0. The reals are denoted by $\mathbb{R}$. Since each of these sets is ordered, an inequality in the subscript denoted a set restriction, e.g. $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$. The n-Cartesian product of a set A is given by A^n, e.g. $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$. A shorthand for the time derivative of a function f will be $\dot{f}$. Regarding a vector x, the transpose is denoted by $x^\top$, the (2-)norm by $||x||$. A matrix is always written with a capital letter, where I being the identity and $\mathbf{0}$ the zero matrix, each of appropriate size regarding the problem. We write $A \succ 0$ ($A \succeq 0$) if A is positive (semi-) definite, for systems this notation will denote different equivalence classes. Basic mathematical symbols apply, such as: implication $\implies$, for all $\forall$, there exists $\exists$, element in $\in$, set inclusion $\subseteq$, such that $'\,:'$ or $'\,|'$, except $\setminus$, sum $\sum$, infinity ∞, set union $\cup$ and intersection $\cap$. Sets can be defined using set-builder notation $\{\text{elements} \mid \text{constraints}\}$, e.g. $\{n \in \mathbb{N} \mid n < 9\}$. The size of a set A is defined by the amount of elements $|A|$ and the complement by $\overline{A}$. The empty set is given by $\varnothing$. Further notations will be given in the relevant sections.

Chapter 1

Theoretical preliminaries

In this chapter, the preliminary knowledge to this thesis will be discussed. This will form a solid knowledge base and will introduce the necessary notations for the further chapters.

First we will take a look at Event-Triggered Control (ETC) and some more advanced concepts. In order to fully grasp these concepts, we need to describe a system as a finite automata. Therefore, we will also give some information on (in)finite state machines and hybrid systems. There is a vast amount of knowledge on these concepts, but we will discuss only what is necessary for this thesis.

This chapter is also the place where certain algorithms will be discussed in detail. The choice on why these specific algorithms are chosen, will be motivated in the designated section once all the details of the problem are clear. Lastly, the Pareto frontier will be introduced in order to talk about the multi-objective optimization nature of the problem.

1-1 Event-Triggered Control

Classic control methods continuously (or periodically for discrete control methods) recalculate the 'ideal' control output. Continuous control methods are only possible with dedicated hardware, either mechanically or electrically. Which comes with high cost since every controller needs an unique design. On the other hand, periodic control methods have had a rise in popularity ever since computer chips became more powerful and affordable. However, periodic control often requires a quick sampling period since this is the only way to guarantee a fast response to perturbations. When having a dedicated channel for the control system, this is not a problem. It becomes problematic when the sensor and control data is send over same network as other vital components of the system. Let alone when multiple control systems operate on the same network as in Networked Control System (NCS), the network can be overloaded resulting in packet losses and unexpected behavior.

Instead of evaluating the output of the controller continuously or every time step, ETC evaluates the control action only in certain events. In 1999 Åström, Bernhardsson and Åarzén

first discussed the benefits of event based controllers [4, 3]. This can reduce the unnecessary high number of computations and data send over the network. For continuous time controllers this can be done by introducing an additional hardware. For the interested reader into different ETC methods, I refer the reader to my literature review. Here I touched on different topics such as Continuous Event Triggered Control (CETC), Self-Triggered Control (STC), early triggering and nonlinear ETC. For this thesis, I will limit the information to only the essentials.

1-1-1 System introduction

We start of with the classical Linear Time-Invariant (LTI) control scheme given in Eq. (1-1). Moreover, S is a general LTI system defined by the matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{k\times n}$ and $D \in \mathbb{R}^{k\times m}$. The dynamics $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t)$ describe how the state of the system $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n\times 1}$ progresses over time $t \in \mathbb{R}_{\geq 0}$ given an input signal $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^{m\times 1}$. In this project, we assume full state knowledge at the current time. If this is not the case, but the system is observable, one could design an observer to obtain the state $\mathbf{x}(t)$ from the output $y(t)$ over time. Therefore, let us set dimensions $k = n$, the matrices $C = I$ equal to the identity and $D = \mathbf{0}$ to the zero matrix of appropriate sizes.

$$S = \begin{cases} \frac{d}{dt}\mathbf{x}(t) & = A\mathbf{x}(t) + Bu(t) \\ y(t) & = C\mathbf{x}(t) + Du(t) \end{cases} \tag{1-1}$$

The LTI system S can either have a single equilibrium point at $\mathbf{x}(t) = \mathbf{0}$ or infinite equilibria if the kernel of A is of higher dimension. Considering the latter, the matrix A will be singular. Therefore, we will focus on the former and require A to be non-singular. In order to freely control the system, independently of the initial state, we also require the pair (A, B) to be controllable, i.e. the controllability matrix $[B\ AB\ A^2B\ \dots\ A^{n-1}B]$ is of full rank.

For the system S, one has designed a static stabilizing feedback gain $K \in \mathbb{R}^{m\times n}$ using their favorite method, e.g. pole-placement. Moreover, the input $u(t) = K\mathbf{x}(t)$ renders the closed loop dynamics stable, i.e. $(A + BK)$ is Hurwitz.

1-1-2 Event-triggered control scheme

Notice that the control input $u(t)$ is continuously depending on the state $x(t)$. As mentioned before, when communicating over a network with limited bandwidth, it becomes beneficial to limit the amount of updates of the control input. Here, ETC methods come into play. In a zero-order-hold fashion, the input values stay constant for a certain period of time, until the state of the system drifted too far and there is a too big mismatch in current input and ideal input. Let us define this process more precisely.

The input u of system S will only be updated at the triggering times, defined by the increasing time sequence $\{t_i\}_{i\in\mathbb{N}_0}$, starting at $t_0 = 0$. The triggering times will be determined by a quadratic homogeneous triggering condition $\xi^\top(t)\mathcal{Q}\xi(t) > 0$. Moreover, details about the triggering matrix $\mathcal{Q} \in \mathbb{R}^{2n\times 2n}$ will be given later on in Section 1-1-5 and ξ is defined as

$\xi(t) = [\mathbf{x}^\top(t),\ \hat{\mathbf{x}}^\top(t)]^\top$. Furthermore, let us define $\hat{\mathbf{x}}(t)$ as the last state known to the controller, specified in Eq. (1-2). This is a snapshot of the state $\mathbf{x}$, which stays constant until the next triggering instance.

$$\text{for } t \in [t_i, t_{i+1}) \hat{\mathbf{x}}(t) = \begin{cases} \mathbf{x}(t_i) & \text{if } \xi^\top(t)\mathcal{Q}\xi(t) \leq 0 \\ \mathbf{x}(t) & \text{if } \xi^\top(t)\mathcal{Q}\xi(t) > 0 \end{cases} \tag{1-2}$$

We can use $\hat{\mathbf{x}}$ to define our zero-order-hold input signal: $u(t) = K\hat{\mathbf{x}}(t)$. The resulting system dynamics of S are given below in Eq. (1-3). The key to designing a ETC method is choosing the triggering sequence in a smart way. As we will see, one could choose $\mathcal{Q}$ in a way that guarantees stability of the closed loop system, while guaranteeing a certain decay rate of the Lyapunov function. However, as seen in the literature review, some methods go beyond simply choosing a triggering matrix $\mathcal{Q}$ and implement more complex triggering conditions.

$$\begin{aligned} \frac{d}{dt}\mathbf{x}(t) &= A\mathbf{x}(t) + Bu(t) \\ u(t) &= K\hat{\mathbf{x}}(t) = K\mathbf{x}(t_i) \qquad \text{for } t \in [t_i,\ t_{i+1}) \end{aligned} \tag{1-3}$$

The time between two triggering instances t_i and t_{i+1} is called the Inter-Sample Time (IST) $\tau_i = t_{i+1} - t_i$. The IST's, or more precisely the Average Inter Sample Time (AIST)'s, are an important measure on how aggressive a triggering condition is. Likewise to the triggering times, one can define the IST sequence $\{\tau_i\}_{i\in\mathbb{N}_0}$, capturing all inter-sample times between triggering instances.

1-1-3 Periodic event-triggered control

Instead of continuously checking the triggering condition $\xi^\top(t)\mathcal{Q}\xi(t) > 0$. Regarding Periodic Event-Triggered Control (PETC) methods, this triggering condition is only checked periodically. Likewise to discretized controllers, the sampling frequency needs to be relatively high due to needing a decent reacting time to disturbances. These methods also have the practical benefit that the triggering condition can be checked using a computer (which can not operate in continuous time).

The sampling frequency will be defined as a constant $\Delta \in \mathbb{R}_{\geq 0}$. Since the trigger condition is only checked periodically and $t_0 = 0$, all triggering times $\{t_i\}_{i\in\mathbb{N}_0}$ are a multiple of Δ, i.e. for all $i \in \mathbb{N}$ there exists a $m \in \mathbb{N}$ such that $t_i = m\Delta$. Furthermore, one can extend this reasoning to the IST's. Meaning, each IST τ_i is a multiple of Δ, i.e. for all $i \in \mathbb{N}$ there exists a $k \in \mathbb{N}$ such that $\tau_i = k\Delta$. Since Δ is kept constant throughout the simulations, one could define the IST's by the discrete inter-sample time sequence $\{k_i\}_{i\in\mathbb{N}_0}$. Recall, each $k_i = \frac{\tau_i}{\Delta}$ is an integer.

1-1-4 Lyapunov theory

In order to define stability criteria, let us introduce some Lyapunov theory, or more precisely, the second method of Aleksandr Lyapunov [13]. A Lyapunov function can be seen as some

measure of how far the state of a system is from the origin, and the guarantee that as time progresses, the system will get closer to the origin. Therefore, the mere existence of a Lyapunov function guarantees stability. The precise definition of a Lyapunov function is given in Definition (1.1).

Definition 1.1 (Lyapunov Function)**.** Given a system describing the state dynamics $\frac{d}{dt}\mathbf{x}(t)$, the function
$V : \mathbb{R}^n \to \mathbb{R}$ is called a Lyapunov Function if the following conditions are met:

- $V(\mathbf{x}) > 0$ for $\mathbf{x} \neq \mathbf{0}$
- $V(\mathbf{x}) = 0$ for $\mathbf{x} = \mathbf{0}$
- $\frac{d}{dt}V(\mathbf{x}) \leq 0$ for $\mathbf{x} \neq \mathbf{0}$

Remark. We may write $\dot{V}(\mathbf{x})$ for $\frac{d}{dt}V(\mathbf{x}(t)) = \frac{dV(\mathbf{x})}{d\mathbf{x}}\frac{d\mathbf{x}(t)}{dt}$. Furthermore, the function V is only a Lyapunov function in regards to a given system.

There are different ideas on how to construct the Lyapunov function. For some physical systems, a Lyapunov function can be constructed via the idea of total amount of energy in the system in for example [14]. As long as the total amount of energy always decreases, the system will converge to a minimum. Another interpretation of the Lyapunov function is the sense of a cost function or a measure of performance [6]. Below, different theorems are given concerning stability by Lyapunov's second method. For thorough modern proofs, I refer to [15].

Theorem 1.1 (Lyapunov Stability)**.** *Given a system $\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t))$, if there exists a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ with regards to the system, then the system is stable.*

Theorem 1.2 (Lyapunov Asymptotic Stability)**.** *Given a system $\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t))$, if there exists a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ with regards to the system and for $\mathbf{x} \neq \mathbf{0}$ it holds that $\frac{d}{dt}V(\mathbf{x}(t)) < 0$, then the system is asymptotically stable.*

There are many more notions of stability, e.g. global (asymptotic) stability or exponential. In general, finding if a Lyapunov function exists is a hard problem. Luckily, due to many research in asymptotically stable LTI autonomous system $\dot{x} = Ax$, one can find a quadratic Lyapunov function with ease as shown in Eq. (1.3). Since the closed loop dynamics of our system S with continuous inputs are such an autonomous system $\dot{\mathbf{x}}(t) = (A + BK)\mathbf{x}(t)$, one can use the Lyapunov equation to construct a Lyapunov function for the system.

Theorem 1.3 (Lyapunov Equation)**.** *Given Q a positive definite matrix and an LTI system $\dot{x} = Ax$. The system is global asymptotically stable if and only if there exists a unique positive definite P such that $A^\top P + PA + Q = 0$ such that $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$ is a Lyapunov function for the system.*

Notice that the eigenvalues of P can indicate the maximum and minimum decay rate of the Lyapunov function. However, in the context of ETC, the minimum rate can lay lower due to only updating the input at triggering instances.

1-1-5 Triggering conditions

In the literature review, we investigated quadratic, nonlinear, state error, input error, and Lyapunov methods to construct triggering conditions for the system S. Here we will limit us to two examples of quadratic Lyapunov triggering conditions which each guarantee a variable decay rate of the Lyapunov function. This to illustrate the existence of of multiple Lyapunov methods that allow to set variable decay rates. However, since only the monotone Lyapunov triggering condition is used for simulations, the explanation of the Dynamic Triggering Mechanism (DTM) will kept short and the interested reader is referred to either Girard [16] or the literature review.

Monotone Lyapunov triggering condition The first method is the monotone Lyapunov triggering condition, which is the more conservative option of the two. Due to the PETC approach, we will tolerate a slower decay of the Lyapunov function then is usually the case (for contiuous control, the Lyapunov decay is a combination of the eigenvalues of P). The key insight here is that we limit the derivative of the Lyapunov function $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$ and each time our slower rate is about to be surpassed, a trigger instance is initiated. Below a simple derivation of the derivative of V is given. This notion can be used to construct the PETC triggering matrix $\mathcal{Q}_{ML}$ of the triggering condition $\xi^\top(t)\mathcal{Q}_{ML}\xi(t) > 0$.

$$\dot{V}(\mathbf{x}) = (A\mathbf{x} + BK\mathbf{x})^\top P\mathbf{x} + \mathbf{x}^\top P(A\mathbf{x} + BK\mathbf{x}) = -\mathbf{x}^\top Q\mathbf{x}$$

In order to tolerate a slower Lyapunov decrease, introduce $\sigma \in [0, 1)$. Recall that the decrease of $V(\mathbf{x}(t))$ has to be smaller or equal to zero in order for V to be a Lyapunov function. As long as the old input from time t_k is hold, a softer restriction can be applied shown in Eq. (1-4). Furthermore, a trigger condition can be constructed when the softer restriction becomes close to being violated, i.e. when $\dot{V}(\mathbf{x}(t_k)) = -\sigma\mathbf{x}^\top Q\mathbf{x}$.

$$\frac{d}{dt}V(\mathbf{x}(t_k)) \leq -\sigma\mathbf{x}^\top Q\mathbf{x} \tag{1-4}$$

Moreover, for a given σ, as long as the Lyapunov function is decreasing, the current input from the controller can be hold. Otherwise, a new input is calculated by the controller using the stabilizing static feedback gain K, which implies that the Lyapunov decrease will be equal to $-\mathbf{x}^\top Q\mathbf{x}$ again.

The problem that remains is how to calculate the Lyapunov decrease at every point in time if a sample-and-hold strategy such as ETC is applied. Define $e_{\mathbf{x}}(t) = \mathbf{x}(t_k) - \mathbf{x}(t)$ for $t \in [t_k, t_{k+1})$ as the difference between the current state and the last state the input is updated. For $t \in [t_k, t_{k+1})$, the derivative of $\mathbf{x}$ is expressed in Eq. (1-5). Using this state derivative, in Eq. (1-6) the derivative of the Lyapunov function is expressed.

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= A\mathbf{x} + BK\mathbf{x}(t_k) \\ &= A\mathbf{x}(t) + BK\mathbf{x}(t) + BKe_{\mathbf{x}}(t) \end{aligned} \tag{1-5}$$

$$\begin{aligned} \frac{d}{dt}V(\mathbf{x}(t)) &= \frac{dV(\mathbf{x})}{d\mathbf{x}}\frac{d\mathbf{x}(t)}{dt} \\ &= \frac{dV(\mathbf{x})}{d\mathbf{x}}(A\mathbf{x}(t) + BK\mathbf{x}(t) + BKe_{\mathbf{x}}(t)) \\ &= -\mathbf{x}^\top(t)Q\mathbf{x}(t) + 2\mathbf{x}^\top PBKe_{\mathbf{x}}(t) \end{aligned} \tag{1-6}$$

Introducing the extended state for this system $z(t) = [\mathbf{x}^\top(t)\ e_{\mathbf{x}}^\top(t)]^\top$, Eq. (1-6) can be combined with Eq. (1-4). Resulting in the condition shown in Eq. (1-7). Notice if the condition is strictly smaller then zero, the Lyapunov decrease is negative. Before this condition is violated, the control output should be updated. Therefore, the triggering condition becomes when 1-7 becomes an equality, the control output should be updated. [6, 17, 18, 19]

$$z^\top(t)\begin{bmatrix}(\sigma-1)Q & PBK\\ (PBK)^\top & \mathbf{0}\end{bmatrix}z(t) \leq 0 \tag{1-7}$$

Translating this to the state $z(t)$ to $\xi(t) = [\hat{\mathbf{x}}^\top(t)\ \mathbf{x}(t)]^\top$ results in our desired triggering condition $\xi^\top(t)\mathcal{Q}_{ML}\xi(t) > 0$, where $\mathcal{Q}_{ML}$ is given by Eq. (1-8). This method guarantees an exponential decay rate of $\kappa\sigma$ of the Lyapunov function V. Here κ is the maximum decay rate of the Lyapunov function of the given system. How one can determine κ is given in appendix A-1. The proof from a different perspective is given in appendix A-2. Here, the rate of decay is derived more clearly and is shown that for any initial condition $\mathbf{x}(0)$, the Lyapunov function is guaranteed to decay: $V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{-\kappa\sigma t}$.

$$\mathcal{Q}_{ML} = \begin{bmatrix}(\sigma-1)Q - 2PBK & PBK\\ (PBK)^\top & \mathbf{0}\end{bmatrix} \tag{1-8}$$

Dynamic triggering mechanism

Instead of obligating a monotone Lyapunov decrease, one of the relaxed Lyapunov conditions can be applied to derive a non-monotone Lyapunov triggering condition. The monotone Lyapunov condition requires the derivative of V to be always below a limit, this is not the case for the relaxed Lyapunov condition. The key insight by Girard [16]: V is allowed to increase, but as long as we stay below the exponential function, exponential decay is preserved. Therefore, this method allows for gibber AIST. The rate of decrease of the exponential function may vary, but gives a guaranteed decay rate for some point in time.

The system is complemented with an additional dynamic of η, which can be seen as a filtered value of $\sigma\alpha(\|x\|) - \gamma(\|e\|)$. This quantity should be non-negative on average to guarantee stability. Therefore, we can require η to be always non-negative, and thereby enforce stability. For the linear case, the dynamics of η are given by:

$$\dot{\eta} = -\lambda\eta + (1-\sigma)x^\top Qx - 2x^\top PBKe, \eta(0) = \eta_0$$

Here, $\sigma \in (0,1)$ varies the guaranteed exponential decay of the Lyapunov function, $\lambda > 0$ is a design parameter, but in general set to the maximum possible decay using σ, namely $\lambda = \kappa\sigma$. Furthermore, η_0 is the initial condition of η, the starting point of the dynamics. In our simulation $\eta_0 = 0$ is set to zero. The DTM is given below for how to determine the triggering times.

$$t_{i+1} = \inf\left\{\begin{array}{l} t \in \mathbb{R} \mid t > t_i \wedge \\ (1-\sigma)x(t)^\top Qx(t) - 2x(t)^\top PBKe\,(t^-) \leq 0 \end{array}\right\}$$

Notice an extra design parameter θ. As θ approaches infinity, the performance of the DTM approaches the static performance of continuously using the feedback gain K. Furthermore, for values of θ below $\frac{1}{2\|A+BK\|-\lambda}$, no gain in AIST is obtained. One is free to choose it in a range between the two, but for our simulations, we set it to the minimum functional value $\theta = \frac{1}{2\|A+BK\|-\lambda}$.

For stability requirements, a Lyapunov candidate V is extended with the dynamics of η, resulting in $W(x,\eta) = V(x) + \eta > V(x)$. Girard guarantees the decay of W as shown below, and since $W(x,\eta) > V(x)$, guarantees asymptotically stability.

$\frac{d}{dt}W(x(t),\eta(t)) = \sigma x(t)^\top Q x(t) - \lambda\eta(t)$

Using this method, Girard even gives an lower bound on the IST, for which we will refer the reader to [16]. In Figure 1-1, we made a simulation showing a possible behavior ot the DTM and visualizing the concept.

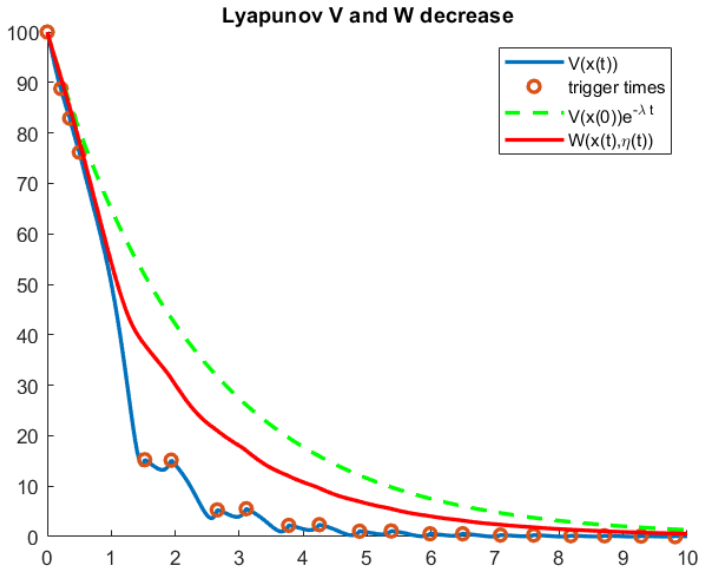


Figure 1-1: Simulation of DTM for PETC strategy on the same system introduced in Chapter 5. Using a decay of $\sigma = 0.9$ and initial condition $x(0) = [10\ 0]^\top$.

1-2 Graph Theory

Graph theory is a broad topic in which much research is done. For this thesis, we incorporate some ideas from graph theory and will link some concepts to definitions we will make later on. Since we only need a limited amount of ideas, the concepts below are reduced to only what is useful for this thesis.

Let us first of introduce what a graph is below in Definition (1.2). A visual representation of a graph $G = (E, V)$ can be constructed by making a sphere for every node in V and connect the spheres with lines according to the transitions in E. For a directed graph, the transitions are given by arrows.

Definition 1.2 ((Directed) Graph)**.** Let $G = (E, V)$ be a (directed) graph, defined by a set of vertices V and a set of edges $E \subseteq V \times V$. In a directed graph, a element $(v_0, v_1) \in E$ describes only an edge from v_0 to v_1, instead of also an edge from v_1 to v_0.

Remark. Vertices and edges will also be referred to as nodes and transitions respectively.

When there are only a finite amount of vertices in the graph $|V| \in \mathbb{N}$, the graph G is called a finite graph. Notice that in a finite graph, there can only exists a finite amount of edges since $E \subseteq V \times V$. Graphs can be extended to weighted graphs as shown in Definition (1.3) by introducing a weight mapping between the edges and weights.

Definition 1.3 (Weighted Graph)**.** Let $G = (E, V, w)$ be a weighted graph, where (E, V) is a graph and $w : E \to \mathbb{R}$ is a weight function which maps a weight to every transition

Later on, further requirements can be set on the weight function. For example, some algorithms may require non-negative weights or only integer values for weights. An important notion will be the average weight along several edges, or more formally, along a path. A path is defined by a sequences of connected vertices by edges as shown in Definition (1.4).

Definition 1.4 (Path)**.** Let $G = (E, V)$ be a graph. A path in G is defined by a sequence of vertices $v_0 v_1 v_2 \dots$ such that from every vertex in the sequence there exists an edge to the next vertex, i.e. for all i it holds $(v_i, v_{i+1}) \in E$

Remark. A path can be finite or infinite depending if there are a finite or infinite amount of vertices in the path. Moreover, sometimes paths will be visualized as a sequence of edges, instead of vertices.

The existence of a path between vertices is a useful tool to describe other properties of a graph G. Let us also briefly introduce the reachability of a vertex. Given a graph $G = (V, E)$ and a starting vertex $v \in V$, define $Reach(G, v)$ as the set of all vertices $v' \in V$ such that there exists a path from v to v'. This concept will explored further in 1-9-2. The following definitions all use the existence, or the lack, of a possible path between vertices. For example, a graph G is strongly connected if there exists always a path between every pair of vertices as shown in Definition (1.5). This property of a graph, while being demanding, will be a necessary condition for some algorithms later on.

Definition 1.5 (Strongly Connected Graph)**.** A graph $G = (E, V)$ is strongly connected if and only if from every vertex $v \in V$ there exists a path to every other vertex $v' \in V$

Another very useful concept will be cycles, which are a special kind of paths as shown in Definition (1.6). Since a cycle starts and ends in the same vertex, as the name suggests, the path is circular. A cycle, unlike a path, has not necessarily a starting node. Given a cycle $c = v_0 v_1 v_2 \dots v_m$, one can shift the vertices such that $c' = v_1 v_2 \dots v_m v_{m+1}$ describes the same cycle $c = c'$.

Definition 1.6 (Cycle)**.** A cycle c in a graph $G = (E, V)$ is given by a finite path $c = v_0 v_1 v_2 \dots v_m$ for some $m \in \mathbb{N}$ such that $v_0 = v_m$

Remark. Cycles are normally presented by the minimal path, meaning for a cycle $c = v_0 v_1 v_2 \dots v_m$, v_m is the first occurrence when $v_0 = v_m$, i.e. for all $i \in \{1, 2, \dots, m = 1\}$ it holds $v_i \neq v_0$. As well as, just as with paths, cycles will sometimes also be described by a sequence of edges instead of vertices.

Once there exists at least two cycles c_1, c_2 in a graph G which share a common vertex, then one can make an infinite amount of new cycles c' by using different combinations of c_1, c_2. In order to regain some structure, we differentiate between so called primitive cycles as shown in Definition (1.7), which are the smallest cycle representation we can consider since all other cycles can be constructed from primitive cycles.

Definition 1.7 (Primitive Cycle)**.** A primitive cycle $c = v_0v_1v_2 \dots v_m$ in a graph $G = (E, V)$ is a cycle which does not consists of other cycles, i.e. every vertex along the path of c is unique, except v_m.

We will denote the set of all primitive cycles of G by C_G. If there does not exists an outgoing edge from a vertex, a path can end up in a dead-lock. If there exists such a vertex in the graph, the graph is said to be blocking as shown in Definition (1.8). If no such vertex exists, the graph is said to be non-blocking. Often, a non-blocking graph is required in order to make sure one never gets stuck.

Definition 1.8 (Blocking)**.** A graph $G = (E, V)$ is said to be blocking if there exists a $v \in V$ for which there does not exist an outgoing edge in E. When in v, the system encounters a so called dead-lock.

1-2-1 Infinite paths and cycles

There is a big connection between infinite paths and cycles in a finite graph. In order to sketch this connection, we introduce Theorem (1.4) which describes the existence of a cycle in non-blocking graph. Using this theorem, we can conclude small trivial statements around infinite paths and cycles.

Theorem 1.4 (Cycle Existence)**.** *Given a finite graph $G = (E, V)$. If for every vertex $v \in V$, there exists a $v' \in V$ such that $(v, v') \in E$, then there exists at least one cycle in the graph G, i.e. $C_G \neq \varnothing$.*

Proof. Given a finite graph $G = (E, V)$. Start in a vertex $v \in V$ and create a path. The path can always be extended since every vertex had an outgoing edge, and thus G is non-blocking. Since G is finite, there are only a finite amount of vertices. When the path reaches a length of $|V| + 1$, at least one vertex v' has to be in the path twice. Therefore, within the path creating a cycle from v' □

Notice, Theorem (1.4) can say something about the existence of an infinite path. Given a finite graph $G = (E, V)$. There exists an infinite path in G if and only if there exists a cycle in G. Moreover, if G is non-blocking, one can create an infinite path starting from every vertex in V. Which describes the connection between infinite paths and cycles.

By the definition of a (primitive) cycle, if in a path a vertex $v \in V$ appears more then once, there exists a cycle including this vertex v. Therefore, given a finite graph $G = (E, V)$ and an infinite path $v_0v_1 \dots$. Vertices not present in any primitive cycle are at most visited once in an infinite path.

This last conclusion will be useful when interested in the average edge weight of an infinite path on a finite graph. Since vertices not present in primitive cycles are visited at most once,

they do not contribute to the average edge weight of an infinite path, since they can occur only a finite amount of times in a finite graph. Therefore, when investigating the average edge weight of infinite paths in a finite graph, one can limit the focus on the average edge weights of the separate primitive cycles. Recall every cycle is made up of a combination of primitive cycles.

1-2-2 Transition systems

In later sections we will talk about Transition System (TS) which will have a clear connection with graphs as defined earlier. Moreover, even when defining hybrid systems in Section 1-3, the internal dynamics of a system S can be interpreted as progressing through a graph. Since graph theory is used by many algorithms. It is essential to see the relation between the concepts we will introduce, since we will switch between them.

A transition system is, like a graph, defined by a set of states and a set of transitions as shown in Definition (1.9). In computer science, transition systems are more used to describe different states and connections in a system, while graphs are more used to describe a process or logical problem. Like graphs, we can add labels and weights to a TS to describe more complex problems. In Definition (1.10), the formal definition of a Labeled Transition System (LTS) is given.

Definition 1.9 (Transition System (TS))**.** A Transition System (TS) T is a tuple (X,E), where:

- X the set of states
- $E \subset X \times X$ the set of possible transitions

Definition 1.10 (Labeled Transition System (LTS))**.** A Labeled Transition System (LTS) T is a tuple (X,E, U), where:

- X the set of states
- U the set of labels (or inputs)
- $E \subset X \times U \times X$ the set of possible transitions

1-3 Hybrid Systems

A hybrid system is a combination of a continuous time and discrete time system. The system can endure flow like behavior from the continuous time dynamics and jump like behavior from the discrete time dynamics. In this section, different descriptions of hybrid systems and automata will be given. For the thesis, we only included subjects which we need later on. For a more elaborate description of different hybrid systems, hybrid time domain, the basic assumptions and stability conditions for hybrid systems, we will refer to the literature studies.

1-3-1 Jump-Flow Systems

The first type of hybrid system we will discuss is the jump-flow system. In a jump-flow system, the system can either flow according to continuous time dynamics given by a function $f : \mathbb{R}^n \to \mathbb{R}^n$ if the state $\mathbf{x} \in \mathbb{R}^n$ is inside the flow set $\mathcal{C} \subseteq \mathbb{R}$, or the system can jump according to discrete time dynamics given by the function $g : \mathbb{R}^n \to \mathbb{R}^n$ if the state $\mathbf{x}$ is inside the jump set $\mathcal{D} \subseteq \mathbb{R}$ [20, 15]. The corresponding description is given in Eq. (1-9).

$$\mathcal{H} = \begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) & \text{if } \mathbf{x}(t) \in \mathcal{C} \\ \mathbf{x}(t^+) = g(\mathbf{x}(t)) & \text{if } \mathbf{x}(t) \in \mathcal{D} \end{cases} \tag{1-9}$$

Notice an important question arises when the state $\mathbf{x} \in \mathcal{C} \cap \mathcal{D} \neq \varnothing$ is in both the flow and jump set. According to the dynamics of the system given in Eq. (1-9), both the flow and the jump dynamics could occur. Therefore, a solution to the jump-flow system is in general not unique and the system is said to be non-deterministic. Accepting the non-uniqueness of the solution is the start of a more general approach given in section 1-3-2.

1-3-2 Set Theory Approach

Instead of jump-flow systems from Eq. (1-9), hybrid systems can more generally be described by a set theory approach [15, 21]. Define the state as function of time $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. The system is allowed to flow if $\mathbf{x}(t) \in \mathcal{C}$ and may flow in different directions, specified by $\mathcal{F}(\mathbf{x}(t))$. Similarly, the system is allowed to jump if $\mathbf{x}(t) \in \mathcal{D}$ and may jump in different directions, specified by $\mathcal{G}(\mathbf{x}(t))$. A hybrid system $\mathcal{H}$ will then be given by Eq. (1-10) below.

$$\mathcal{H} = \begin{cases} \dot{\mathbf{x}}(t) \in \mathcal{F}(\mathbf{x}(t)) & \text{for } \mathbf{x}(t) \in \mathcal{C} \\ \mathbf{x}(t^+) \in \mathcal{G}(\mathbf{x}(t)) & \text{for } \mathbf{x}(t) \in \mathcal{D} \end{cases} \tag{1-10}$$

The set-value maps $\mathcal{F}$ and $\mathcal{G}$ are defined as follows: $\mathcal{F} : \mathcal{C} \to \mathcal{P}(\mathbb{R}^n)$ and $\mathcal{G} : \mathcal{D} \to \mathcal{P}(\mathbb{R}^n)$, where $\mathcal{P}(\mathcal{A})$ denotes the power set of the set $\mathcal{A}$, describing all possible sets one can make using $\mathcal{A}$.

This definition of a hybrid system deliberately makes it possible to introduce non-determinism. Both by making it possible to overlap the flow and jump sets: $\mathcal{C} \cap \mathcal{D} \neq \varnothing$, and including the possibility that for a given state $\mathbf{x}$, the sets $\mathcal{F}(\mathbf{x})$ and $\mathcal{G}(\mathbf{x})$ can both have more then a single element.

1-3-3 Switched systems

Next to jump-flow systems, switched (or switching) systems are also a class of hybrid systems. They are a powerful tool that can describe saturated inputs (e.g. physical maximum or minimum limitations on actuators) as well as changing system dynamics [20, 22]. Where jump-flow systems are very broad and can be intertwined a lot, switched systems consist of a set of system dynamics, e.g. a set of matrices $\{A_1, A_2, \dots, A_m\}$ that describe a linear dynamic shown in Eq. (1-11) for some switching rule $\alpha \in \{1, 2, \dots, m\}$.

$$\dot{\mathbf{x}}(t) = A_\alpha \mathbf{x}(t) \tag{1-11}$$

At first sight, one could suggest that evaluating every system separately on stability would be sufficient such that the switched system is stable. However, as shown in [22], this is not the case. Even if all separate systems are stable, the switched system may, for some switching patterns, be unstable. They even give an example of the inverse problem and show that with a switching pattern the switched system can be stable, even when all separate systems are unstable.

To ensure stability of the switched system, one could again go over the previous definition of stability of hybrid systems. However, due to the neat framework of switched systems, stability can be guaranteed by the existence of a Common Lyapunov Function (CLF). If a function V is a Lyapunov function for all subsystems, then V is called a CLF. Recall from Section 1-1 the definition of a Lyapunov function from Definition (1.1) and the stability conditions that followed.

Remark. I like to think of a Lyapunov function as if it is some measure of how close the system is to the origin, i.e. the equilibrium point. A CLF is nothing more then saying all subsystems must converge to the origin in regards to the same measure.

Notice, in general it is very hard to find a Lyapunov function for a given system. Let alone to find a CLF for a given switched system. However, for linear systems one can solve a small optimization problem to construct a quadratic Lyapunov function. In the next section we will show that it is also possible to construct a CLF for switched systems when dealing with linear subsystems with quadratic triggering conditions.

1-3-4 Constructing a common Lyapunov function

In our case, given the system, we will construct a quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$. This V can then be used to construct different triggering conditions, all guaranteeing a certain exponential decay of V. Therefore, V will be a CLF. Down below, more a extensive motivation is given and the resulting decay rate of the CLF V is computed.

By switching between triggering conditions, each guaranteeing a different decay rate of the Lyapunov function, a different triggering sequence can be obtained. Preferably, we want a triggering condition that guarantees an optimal decay rate ($\sigma = 1$). However at some points in time, this could result in a lot of triggering times within a small amount of time, which may be undesirable. Therefore, it could be beneficial to switch from triggering condition. The switching times are a subset of the sampling times and given by $\{\hat{t}_i^s\}_{i\in\mathbb{N}_0} \subseteq \{\hat{t}_i\}_{i\in\mathbb{N}_0}$. Moreover, the only time we can switch from triggering condition is at the sampling times, because the switching will be depended on the state of the system $\mathbf{x}(t)$. However, the only known state following from the current triggering condition is $\hat{\mathbf{x}}(t)$ and is updated only at the sampling times. Remark, we set the initial value of the switching times sequence to 0 for simplicity reasons $\hat{t}_0^s = 0$.

The Lyapunov function is still bounded by the same decay rate, for the given decay rate in the corresponding time interval $e^{-\sigma\kappa t}$. Only the value of σ changes depending on the triggering condition. In order to track which σ value is used, define σ_k as the value used between $[\hat{t}_k,\ \hat{t}_{k+1})$. Recall that $\tau_i = \hat{t}_{i+1} - \hat{t}_i$ denotes the IST, now define $\tau_i^s = \hat{t}_{i+1}^s - \hat{t}_i^s$ as the inter switching times. The concept of switching between triggering conditions is best illustrated by a small example:

- Before the first time we switch triggering conditions $t \in [\hat{t}_0^s,\ \hat{t}_1^s)$, the original decay rate still applies $V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{-\sigma_0 \kappa t}$
- When switched once, from that point onward we endure a new decay rate σ_1 until the next switching time is reached $t \in [\hat{t}_1^s,\ \hat{t}_2^s)$. Notice that the starting point form the decay function is $V(\mathbf{x}(0))e^{-\sigma_0 \kappa(\hat{t}_1^s - \hat{t}_0^s)}$ instead of just $V(\mathbf{x}(0))$. Therefore, for $t \in [\hat{t}_1^s,\ \hat{t}_2^s)$ it holds that $V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{-\sigma_0 \kappa(\hat{t}_1^s - \hat{t}_0^s)}e^{-\sigma_1 \kappa(t - \hat{t}_1^s)} = V(\mathbf{x}(0))e^{-\sigma_0 \kappa \tau_0^s}e^{-\sigma_1 \kappa(t - \hat{t}_1^s)}$
- More generally, for $N \in \mathbb{N}$ and $t \in [\hat{t}_N^s, \hat{t}_{N+1}^s)$, the Lyapunov function $V(\mathbf{x}(t))$ is bound with $V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{\sum_{k=0}^{N-1} -\sigma_k \kappa \tau_k^s}e^{-\sigma_N \kappa(t - \hat{t}_N^s)}$

Remark, since the function V is at all times bounded by an exponential decreasing function, the function V is exponentially decreasing to 0 and therefore still a Lyapunov function.

Switching between triggering conditions can be beneficial since the quadratic condition can now depend on the state. In some states it could be possible to apply a desired triggering condition (e.g. $\sigma = 1$) and achieve a long IST, while in other states this would result in the need to trigger constantly. Instead of applying a conservative triggering condition to all states, this could only be done for the states needing it. Remark that the exact triggering condition already varied per state, the restriction being that they were related to each other in quadratic manner. By switching between triggering condition, this restriction becomes non-continuous in the sense we can jump to a different quadratic relation.

1-3-5 Automaton

Another approach to model a hybrid system is with the use of automata. An automaton is a way to model the state and output of a system. Originating form studying computer devices in an abstract manner, the first principles of transition systems were introduced by George H. Mealy with the Mealy machine [23] and by Edward F. Moore with the Moore machine [24]. Nowadays, they are widely used in Computer Science and play an fundamental part in some software and hardware solutions [25, p. 2]. Moreover, next to the Mealy and Moore machines, different variations are made in order to model different behavior. A formal definition of an automaton A is given in Definition (1.11).

Definition 1.11 (Automaton)**.** An automaton A is given by the tuple $A = (\mathcal{Q},\ \Sigma,\ \delta,\ \mathcal{Q}_0,\ \mathcal{F})$, further specified by:

- $\mathcal{Q}$ a countable set of states
- Σ, the input alphabet, a countable set of input symbols
- $\delta : \Sigma \times \mathcal{Q} \to \mathcal{Q}$ the transition function
- $\mathcal{Q}_0 \subseteq \mathcal{Q}$ the set of possible starting states
- $\mathcal{F} \subseteq \mathcal{Q}$ the set of accepted states

Different types of automata are often visually represented with a transition diagram to describe the possible states and transitions. An example of such a transition diagram is given in Figure 1-2. Here, the states are given by all the nodes of the system $\mathcal{Q} = \{q_0,\ q_1,\ q_2\}$, the input alphabet is given by $\Sigma = \{0,\ 1\}$, the transition function is visualized by all the edges

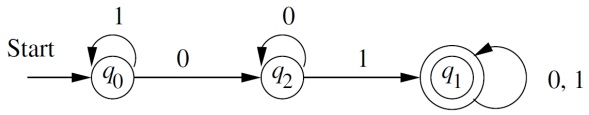


Figure 1-2: The transition diagram for the DFA accepting all strings with a substring 01. Inherited from [25, p. 48]

in the diagram, the set of starting states is visualized by the 'Start' arrow $\mathcal{Q}_0 = \{q_0\}$ and, lastly, the set of accepted states is visualized by the double circled node $\mathcal{F} = \{q_1\}$.

Since the definition of an automaton is very general, there exist a lot of different applications and variations on the definition. Since not all alterations on the automaton are useful for out purpose, only the timed and game automata will be treated in more detail. These will become useful later on with the abstraction of ETC systems and the schedulers which uses these abstractions. Notice the set of accepting is sometimes regarded when the emphasis is on the (infinite) dynamics instead of the (finite) sequences.

1-3-6 Timed Automaton

While the automaton only considered discrete dynamics and choices, the Timed Automata (TA) has both continuous and discrete dynamics and therefore is a representation of a hybrid system. The continuous dynamics are introduced with the use of clocks. A clock is a real variable $c \in \mathbb{R}_{\geq 0}$, starting at a given initial value, which derivative is set to 1. These clocks can be used to describe when a system is allowed to change from state or for how long the system is allowed to stay in a certain state. In order to specify this we need to introduce clock constraints, which is no more than a clock followed by an (in)equality and a real variable, e.g. '$c \leq 3$'. The set of all possible clock constraints, using the clocks from a set $\mathcal{X}$, will be denoted by $\Phi(\mathcal{X})$. Using these clocks and constraints, the definition of a timed automaton is given by Definition (1.12).

Definition 1.12 (Timed Automaton)**.** A Timed Automaton TA is given by the tuple $TA = (\Sigma,\ \mathcal{L},\ \mathcal{L}_0,\ \mathcal{X},\ \mathcal{F},\ \mathcal{E})$ with:

- $\mathcal{L}$ the finite set of states
- $\mathcal{L}_0 \subseteq \mathcal{L}$ the set of initial states
- Σ a finite set, the alphabet, containing all possible actions
- $\mathcal{X}$ a finite set of real variables, the set of clocks
- $Inv : L \to \Phi(\mathcal{X})$ gives every location a clock constraint, while true the system is allowed to stay at the state and the system should transition before the clock constraint becomes false.
- $\mathcal{F} \subseteq \mathcal{L}$ the set of final states
- $\mathcal{E} \subseteq \mathcal{L} \times \Sigma \times \mathcal{P}(\mathcal{X}) \times \Phi(\mathcal{X}) \times \mathcal{L}$ the set of transitions, which maps from some states to others with corresponding element from the alphabet. A transition is only allowed to be taken if the clock constraint is true, the new clock values are given towards the new state.

1-4 Minimum Cycle Mean

Finding the minimum mean weight of a cycle for a given weighted directed graph is a highly studied subject. However, the most iconic at this time is still Karp's algorithm given in [26], which can find the minimum cycle mean in $\mathcal{O}(mn)$ time. Here, $n = |V|$ is the number of nodes/vertices and $m = |E|$ is the number of edges in the graph. Ever since, small improvements are made based on this algorithm as discussed by A. Dasdan and R. Gupta in [27] for specific conditions. Recently, M. Chaturvedi and R. McConnell have also shown some shortcomings in the original proof of Karp's algorithm [28]. However, all are just small extensions and yield the same worst case $\mathcal{O}(mn)$ time. Nevertheless, it is widely used as the to go algorithm for finding the minimum cycle mean. Karp's algorithm works for both positive and negative weights, it can also be used to construct the maximum cycle mean by multiplying all weights by -1. In appendix B-1 the algorithm is described in more details

Given a directed graph $G = (E, V)$, here V is the set vertices and $E \subseteq V \times V$ is the set of edges. Let $w : E \to \mathbb{R}$ be a weighting function on G, mapping a weight to every edge of G. Let C_G be the set of all possible cycles in the graph G. Notice that a cycle $c \in C_G$ is given by a finite sequence of edges, e.g. $e_1,\ e_2,\ e_3$. The cycle mean of a cycle $c \in C_G$ is defined by $\lambda(c) = \frac{1}{|c|}\sum_{e\in c} w(e)$. Here $|c|$ describes the number of edges in the cycle c. Define λ^* to be the minimum cycle mean of a graph G. Then $\lambda^* = \min_{c\in C_G} \lambda(c)$ is given by Eq. (1-12).

$$\lambda^* = \min_{c\in C_G} \frac{\sum_{e\in c} w(e)}{|c|} \tag{1-12}$$

1-4-1 Negative cycle relation

Notice that the problem of finding the minimum cycle mean is closely related to determining if the graph has a negative cycle. This can be demonstrated by an example. Say we have a $G = (E, V)$ with a cycle set C_G and weight mapping w. Say we know the minimum cycle mean of the graph to be α, meaning that for all $c \in C_G$ it holds that $\lambda(c) \geq \alpha$. Furthermore, there exists at least one $c' \in C_G$ such that $\lambda(c') = \alpha$.

When only interested in if there exists a negative cycle in the graph, one can calculate the exact value of α using Karp and see if it is positive or negative. However, there are faster negative cycle detection algorithms since we do not need to know the exact value of α, just its sign.

1-5 Minimum Weight to Time Cycle

Next to Karp's algorithm for finding the minimum cycle mean, another problem of interest is finding the minimum weight to time ratio of all cycles of a given weighted timed directed graph, also referred to as minimum ratio cycles. Next to a weight $w : E \to \mathbb{R}$, there will also be a timing function $t : E \to \mathbb{R}_{\geq 0}$ associated to each edge of the graph. Which can be interpreted as the time an edge takes before moving to the next vertex. Notice that the timing, instead of the weight, is enforced to be non-negative.

The problem of finding the minimum ratio cycle of a graph can be seen as a generalization of the minimum cycle mean problem by setting the timing function to 1 for every edge: $t(e) = 1$ for all $e \in E$. This is shown in Eq. (1-13) and describes how they are related.

$$\begin{aligned} \lambda_r^* &= \min_{c \in C_G} \frac{\sum_{e \in c} w(e)}{\sum_{e \in c} t(e)} \\ &= \min_{c \in C_G} \frac{\sum_{e \in c} w(e)}{\sum_{e \in c} 1} \\ &= \min_{c \in C_G} \frac{\sum_{e \in c} w(e)}{|c|} \\ &= \lambda^* \end{aligned} \tag{1-13}$$

Since the discovery of Karp's algorithm which runs in $\mathcal{O}(nm)$ time, with $n = |V|$ and $m = |E|$, algorithms for finding the minimum ratio cycle have tried to come close to this performance. There has been strongly polynomial algorithms like Burn's primal-dual algorithm [29] which runs in $\mathcal{O}(mn^2)$ time, weakly polynomial algorithms like Lawler's binary search approach [30] combined with a negative cycle detection algorithm like Bellman-Ford [31, 32]in $\mathcal{O}(nm \log(nCT))$[1] , or a pseudopolynomial algorithm like Hartmannand Orlin [33] which is running in $\mathcal{O}(mnT)$ time. Most recent improvement on the existing algorithms was done by K. Bringmann et. al. [34] which claims to solve the problem in $\mathcal{O}(n^{\frac{3}{2}} m^{\frac{3}{4}} \log^2(n))$, approaching the strongly polynomial upper bound of Karp's algorithm for the simpler problem which only solves the minimum cycle mean.

Given a directed graph $G = (E, V)$, here V is the set vertices and $E \subseteq V \times V$ is the set of edges. Let $w : E \to \mathbb{R}$ be a weighting function on G, mapping a weight to every edge of G. Let $t : E \to \mathbb{R}_{\geq 0}$ be a timing function on G, mapping a non-negative timing to every edge of G. Let C_G be the set of all possible cycles in the graph G. Notice that a cycle $c \in C_G$ is given by a finite sequence of edges, e.g. $e_1,\ e_2,\ e_3$. We will define the ratio cycle of a cycle $c \in C_G$ by $\lambda_r(c) = \frac{\sum_{e \in c} w(e)}{\sum_{e \in c} t(e)}$. Define λ_r^* to be the minimum ratio cycle of a graph G. Then $\lambda_r^* = \min_{c \in C_G} \lambda_r(c)$ is given by Eq. (1-14).

$$\lambda_r^* = \min_{c \in C_G} \frac{\sum_{e \in c} w(e)}{\sum_{e \in c} t(e)} \tag{1-14}$$

As stated above, their is a wide variety of algorithms that can be used to solve the problem of finding the minimum ratio cycle. Which algorithm is preferred depends on additional information on the weight and timing functions and which type of scaling of the problem is expected. Examples of additional information being:

- Weight function only mapping to non-negative values
- Weight and/or timing function mapping to integer values
- Known bounds on the weight and/or timing function

[1] Notice that Lawler's binary search approach only works for weight functions of the form $w : E \to \mathbb{N}_{\leq C}$ and timing function of the form $t : E \to \mathbb{N}_{\leq T}$, i.e. both functions map to a finite set of natural numbers

1-6 Pareto Frontier

In this section, we will introduce the concept of the Pareto front and give an example relevant later in this thesis.

In multi-objective optimization problems, it can be impossible to give priority to one of the optimization quantities. Likewise, striking a balance between these quantities is an arbitrary choice and therefore one can not determine which solution is preferred.

In classic optimization theory, the multi-objective optimization problem is mostly described when minimizing all quantities. Say we want to minimize two quantities $y_1, y_2 \in \mathbb{R}$ where $(y_1, y_2) = f(x)$ is given by $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$. It could be the case there is a global minimum, e.g. there exists a x^g such that both optimization quantities y_1 and y_2 are minimal. However, the solution space could also be more complex. Since f is an arbitrary function, a different input x' may lead to a decrease of y_1 but increase of y_2. Pareto optimal (i.e. efficient) points are defined as solutions which are optimal in some sense. Meaning that decreasing one quantity leads to an increase in one of the other quantities.

In Figure 1-3 Boyd and Vandenberghe visualize the idea of the Pareto frontier for the multi-objective optimization of a continuous function. The set $\mathcal{O}$ visualizes the set of all possible solution $f(x)$ for $x \in \mathbb{R}_{\geq 0}$. The highlighted point $f_0(x^{po})$ is a Pareto efficient solution. The set of all Pareto efficient solutions is called the Pareto frontier (i.e. front) and is given by the function $f_0 = f|\mathcal{X}^{po}$ limited to the set $\mathcal{X}^{po} \subseteq \mathbb{R}_{\geq 0}$ of input values that lead to Pareto efficient points. In the figure, both optimization quantities y_1 and y_2 are visualized on respectively the x and y axis. [35][p. 177-184]

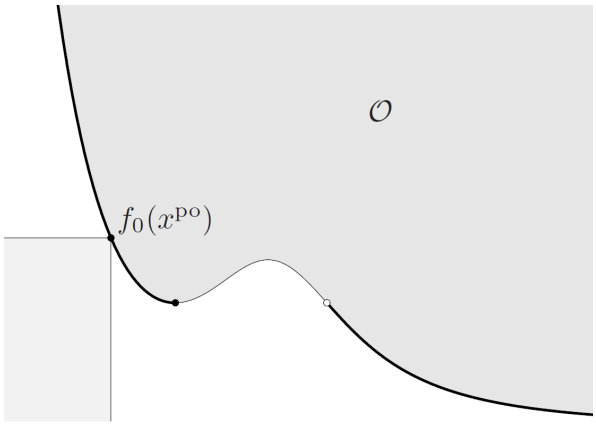


Figure 1-3: Illustration of the Pareto efficient points (visualized by the dark line) of the continuous solution space $\mathcal{O}$ (light gray) when minimizing optimization quantities on both axis. [35][p. 178]

Finding all Pareto optimal points can be hard, depending on the problem. One strategy is scalarization, which evolves around introducing a new condition (which sets a desired slope) such that the problem becomes a optimization problem of one optimization variable; i.e. introduce a vector $\lambda \in \mathbb{R}^n_{\geq 0}$ and minimize the new objective $\lambda^\top f(x)$. In the example of Figure 1-3 this would result in minimizing $\lambda_1 y_1 + \lambda_2 y_2$. Notice that by varying the choice of λ, different Pareto optimal points can be found, but not necessary the whole Pareto frontier. If the optimization problem is convex, then all Pareto optimal solutions can be found using this method and the whole Pareto frontier can be constructed.

1-6-1 Discrete Pareto Frontier

Later on in this thesis, the solutions will not be drawn from a continuous solution space, but from a discrete space, generated by a finite (but very large) distinct set of inputs. For context, we will be interested in maximizing two quantities: the average IST $\hat{\tau}$ and the average control performance $\hat{\sigma}$. The exact details will be discussed later on. Different strategies will result in different solutions shown in Figure 1-4, each point representing a scheduling strategy. Since we are maximizing, the set of Pareto efficient points are given by those for which there does not exist a point which increases in both $\hat{\tau}$ and $\hat{\sigma}$ as shown in Figure 1-5.

When handling discrete solutions, the Pareto frontier is not defined as only the set of Pareto optimal points, but instead as for which region they are optimal. Therefore, the Pareto front describes the boundary of where the Pareto efficient points are optimal as shown in Figure 1-6. For discrete systems, the Pareto frontier may not be convex, then the scalarization method will also not result in all Pareto efficient solutions.

1-6-2 Conditioning

From an engineering point of view, calculating the complete Pareto frontier (besides it may not even be easily computable) may not be that useful. One has to look at all Pareto efficient solutions manually and make a dedicated choice from there. What can be of more use is the introduction of an extra condition.

Instead of calculating the whole Pareto frontier. One could state a hard constraint for all but one of the optimization quantities (e.g. $\hat{\tau} \geq \tau_{min}$). The multi-objective optimization problem then gets reduced to a single optimization quantity as shown in Figure 1-7. All solutions at least have to suffice the hard constraint(s), then the Pareto efficient point that just satisfies this condition is the optimal solution. With this method, from an engineering point of view, one could have more of a feel for the desired solution.

1-7 System Abstractions

In this section, we will inherit the system representation from Tabuada [36, Chapters 1-6] in order to sketch a new perspective on control systems from a Transition System (TS) point of view. The TS as shown in Definition (1.9), is a more general definition then the automaton introduced in Definition (1.11). The definition of the TS forms the basis of the discrete system representations. A TS T is described by a set of states X and the set of all possible transitions E. Whereas the Labeled Transition System (LTS), defined in Definition (1.10), has every transition labeled by an element from the label set U.

1-7-1 System definition

A LTI control system in state space representation is mostly given by $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t)$. One could also think of it as an LTS with a infinite amount of states, connected by transitions, labeled with the corresponding inputs $u(t)$. Notice an autonomous system could be described by a TS instead. The definition of LTS is extended in order to incorporate output behavior

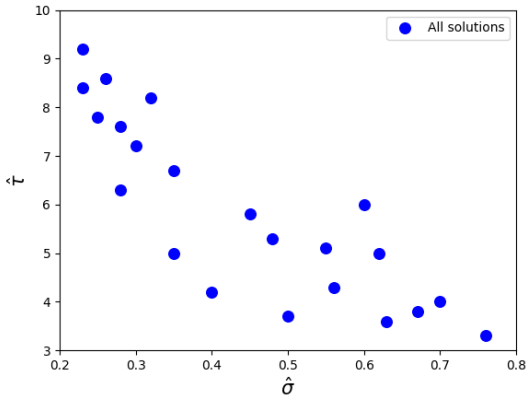


Figure 1-4: Every point, marked with the color 'blue', describes a solution. From such a solution one can measure the worst case performance on average IST [$\hat{\tau}$] and average decay rate [$\hat{\sigma}$] and place it in the figure accordingly.

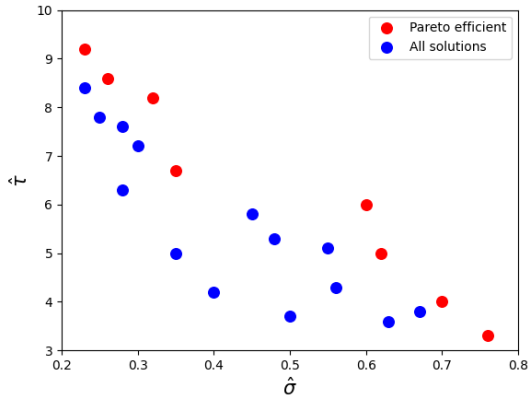


Figure 1-5: The Pareto efficient solutions, marked with the color 'red', describe the set of optimal solutions. Due to being unbiased in optimizing $\hat{\tau}$ or $\hat{\sigma}$, all Pareto efficient solutions are all optimal in some sense.

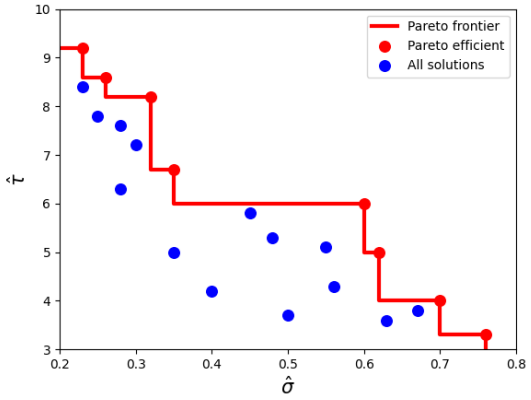


Figure 1-6: The Pareto frontier, visualized by the 'red' line, encloses the area where the Pareto efficient solutions are optimal. No solutions exists above or to the right of this line, given the Pareto efficient solutions are complete.

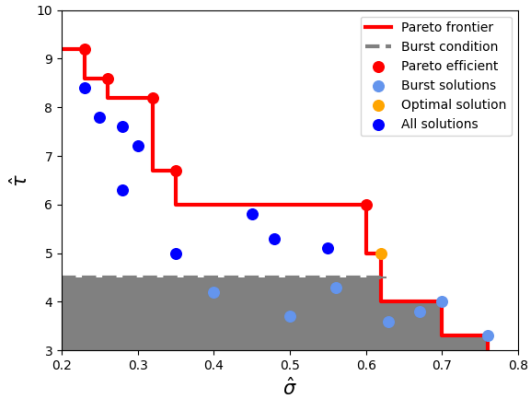


Figure 1-7: By setting an extra condition: a minimum value for $\hat{\tau}$, marked with the color 'gray', our problems becomes a non-multi optimization problem.

and initial starting states of a control system. Therefore, a LTI system S can be described by Definition (1.13) instead of the original dynamics. Notice, all the behavior of the original system is captured. In the literature, the transition set E is also referred to by $\rightarrow$. However, we will often refer to this set as E since the symbol $\rightarrow$ is also used in function declarations.

Definition 1.13 (System)**.** A system S can be represented as a tuple $(X,\ X_0,\ U,\ E\ ,\ Y,\ H)$, where:

- X the set of states
- $X_0 \subset X$ the set of initial states
- U the set of inputs
- $E \subset X \times U \times X$ the set of transitions
- Y the set of outputs
- $H : X \rightarrow Y$ the output map

Remark. We will abuse the notation and overload the U by also defining it as an function $U : X \rightarrow U$. For a given $x \in X$, we will define $U(x) \subseteq U$ by the set of possible inputs at the given state x. This will be useful later on.

1-7-2 System generated behavior

So far we have talked about the behavior of a system, without specifying exactly what is meant by this. We distinguish between two types of behavior, namely internal and external. A *finite internal behavior* from a state $\mathbf{x}_0$ is given by a finite sequence of transitions, i.e. a sequence of states with specified inputs. Likewise, a *finite external behavior* is given by a sequence of outputs $\{y_i\}_{i=0}^{N}$, corresponding by the output map to the sequence of states from the finite internal behavior. All possible finite external behavior, starting from a state $\mathbf{x}_0$, of a system S is denoted by $\mathcal{B}_{\mathbf{x}_0}(S)$, which includes all possible input sequences of finite length. The finite external behavior generated the system S is given by $\mathcal{B}(S)$, further defined by Definition (1.14), which includes all finite external behavior from every possible initial condition.

Definition 1.14 (Finite External Behavior)**.** Given a system $S = (X,\ X_0,\ U,\ E,\ Y,\ H)$. The generated finite external behavior by S is given by: $\mathcal{B}(S) = \cup_{\mathbf{x} \in X_0} \mathcal{B}_{\mathbf{x}}(S)$

Next to the finite sequences, given a system S the *infinite external behavior* from a starting state $\mathbf{x}_0$ is given by $\mathcal{B}^{\omega}_{\mathbf{x}}(S)$ and contains all possible infinite output sequences a system can endure starting from $\mathbf{x}_0$. The infinite external behavior generated by the system S is defined in Definition (1.15) and will be denoted by $\mathcal{B}^{\omega}(S)$.

Definition 1.15 (Infinite External Behavior)**.** Given a system $S = (X,\ X_0,\ U,\ E,\ Y,\ H)$. The generated infinite external behavior by S is given by: $\mathcal{B}^{\omega}(S) = \cup_{\mathbf{x} \in X_0} \mathcal{B}^{\omega}_{\mathbf{x}}(S)$

Remark. The infinite external behavior $\mathcal{B}^{\omega}(S)$, generated by S will also be referred to as just the behavior generated by S, since this will be of interest the rest of the time.

Notice that $\mathcal{B}^{\omega}(S)$ is empty if no infinite behaviors exists, as opposed to $\mathcal{B}(S)$. This framework can be used to incorporate more details about the system S. For example, one may not only be interested in the output sequences, but also wants to incorporate the corresponding input sequence in the behavior. First introduce an element $* \notin U$ that denotes if no input is given. From the system S, create a system $S_o = (X_o,\ X_{o0},\ U_o,\ E_o,\ Y_o,\ H_o)$ given by:

- $X_o = X \times U_o$
- $X_{o0} = X_0 \cup *$
- $U_o = U \cup *$
- $E_o = \{((\mathbf{x}, u), u', (\mathbf{x}', u')) \mid (\mathbf{x}, u', \mathbf{x}') \in E\}$
- $Y_o = Y \times U$
- $H_o : X_o \to Y_o$, given by $H_o(\mathbf{x}, u) = (H(\mathbf{x}), u)$

Generated behavior by S_o, $\mathcal{B}^\omega(S_o)$ will consists of infinite sequences $\{y_i, u_i\}_{i=0}^{\infty}$ instead of $\{y_i\}_{i=0}^{\infty}$.

1-7-3 Synchronization

Given two separate systems S_a and S_b. One can use an *interconnection relation* $\mathcal{I} \subseteq X_a \times X_b \times U_a \times U_b$ to compose both systems, i.e. the interconnection relation describes the interaction of S_a with S_b. The interconnection relation can be used to compose two systems to create an bigger system, or to synchronize the systems for control purposes and is denoted by $S_a \times_{\mathcal{I}} S_b$ as defined in Definition (1.16). The interconnection relation describes the synchronization between both systems on how they proceed.

Definition 1.16 (System Composition)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with an interconnection relation $\mathcal{I} \subseteq X_a \times X_b \times U_a \times U_b$. Let $S_a \times_{\mathcal{I}} S_b = (X_{ab}, X_{ab0}, U_{ab}, E_{ab}, Y_{ab}, H_{ab})$ be the composition of S_a and S_b with:

- $X_{ab} = \pi_X(\mathcal{I})$
- $X_{ab0} = X_{ab} \cap (X_{a0} \times X_{b0})$
- $U_{ab} = U_a \times U_b$
- $E_{ab} = \left\{ ((x_a, x_b), (u_a, u_b), (x'_a, x'_b)) \mid \text{such that: } \begin{array}{l} (x_a, u_a, x'_a) \in E_a, \\ (x_b, u_b, x'_b) \in E_b, \\ (x_a, x_b, u_a, u_b) \in \mathcal{I} \end{array} \right\}$
- $Y_{ab} = Y_a \times Y_b$
- $H_{ab}(x_a, x_b) = (H_a(x_a), H_b(x_b))$

Remark. here the function π_X is the projection from $X_a \times X_b \times U_a \times U_b$ to $X_a \times X_b$.

Due to the construction of the system composition, it holds that the behaviors of the composition are only a subset of the possible behaviors of the separate systems $\mathcal{B}(S_a \times_{\mathcal{I}} S_b) \subseteq \mathcal{B}(S_a) \times \mathcal{B}(S_b)$. Therefore, the composition can limit what behaviors a system endures. Later we will show hoe this can be used for control purposes since we first need to introduce some more concepts.

1-7-4 Behavioral Equivalence

There are different notions of equivalences between systems. We will focus on the behavioral pre-order (i.e. behavioral inclusion) and behavioral equivalence. The behavioral pre-order in systems is described on the different infinite behaviors a system can endure as described by

Definition (1.17).If the behaviors of system S_a are only a subset of the behaviors of system S_b, meaning $\mathcal{B}^\omega(S_a) \subseteq \mathcal{B}^\omega(S_b)$, we say $S_a \preceq S_b$. Behavioral inclusion can be used in order to make sure a system captures at least all the behaviors of another system.

A stricter requirement is the behavioral equivalence of two systems. Two systems S_a and S_b are said to be behavioral equivalent in they produce the exact same infinite behaviors $\mathcal{B}^\omega(S_a) = \mathcal{B}^\omega(S_b)$ as defined in Definition (1.18), denoted by $S_a \cong_{\mathcal{B}} S_b$.

Definition 1.17 (Behavioral Inclusion)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$. Define $S_a \preceq S_b$ if S_a is behaviorally included in S_b, meaning $\mathcal{B}^\omega(S_a) \subseteq \mathcal{B}^\omega(S_b)$

Definition 1.18 (Behavioral Equivalence)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$. Define $S_a \cong_{\mathcal{B}} S_b$ if S_a is behaviorally included in S_b and vice versa, meaning $S_a \preceq S_b$ and $S_b \preceq S_a$.

On the other hand, behavioral inclusion and equivalence can be used in control design. Say we have a system S_a. However, there is another system S_b which has more desirable behaviors over S_a. If one can construct a controller S_c and a interconnection relation $\mathcal{I}$ such that the behaviors of $S_c \times_{\mathcal{I}} S_a$ are limited or equivalent to the desired behaviors S_b (i.e. $S_c \times_{\mathcal{I}} S_a \preceq_{\mathcal{B}} S_b$ or $S_c \times_{\mathcal{I}} S_a \cong_{\mathcal{B}} S_b$), then we controlled our system such that it only exhibits desired behaviors.

For finite state systems, behaviors can be checked on a experimental bases. Since there are only a finite amount of states, all infinite behavior sequences will end up in a (combination of) cycle(s). Therefore, many conclusions can be made from analyzing all primitive cycles in the system. However, for infinite state systems this technique is not so straight forward and checking behavioral inclusion or equivalence can be more difficult.

1-7-5 Reachability

Using behavioral equivalences, we make a small detour in order to define the reachability of a system. Starting from an initial state, with Reach(S) we will denote the possible outputs the system can even endure as shown in Definition (1.19). The set of reachable states can be useful once one is playing safety or reachability games.

Definition 1.19 (Reachable States)**.** Given a system $S = (X,\ X_0,\ U,\ E,\ Y,\ H)$. A state $x \in X$ is a reachable state by S if there exists an $x_0 \in X_0$ such that there is a finite path in S starting from x_0 to x. Moreover, then $y = H(\mathbf{x})$ is said to be a reachable output.

Remark. The set of all reachable outputs of a system S is given by Reach(S).

To put this in the context of behavioral equivalences. Notice that if the behavior of a system S_a is included in a system S_b, the set of reachable outputs must also be included, i.e.
$S_a \preceq_{\mathcal{B}} S_b \implies \text{Reach}(S_a) \subseteq \text{Reach}(S_b)$. This argument follows simply by taking an element in Reach(S_a) and by definition of behavioral inclusion, follows that this element must also be present in Reach(S_b).

1-7-6 Similarity relations

To tackle the difficult part of comparing infinite state systems. We will introduce more tools besides the behavioral equivalence and inclusion. Namely, similarity relations. Given two systems S_a and S_b, a relation $R \subset X_a \times X_b$ relates states from one system to another. First of is a *simulation relation* as defined in Definition (1.20). We say S_a is simulated by S_b via the simulation relation $R \subset X_a \times X_b$. Meaning, if a state is part of the relation $(x_a, x_b) \in R$ then x_a from S_a is simulated by x_b from S_b. When there exists a simulation relation R such that S_b simulates S_a, we denote this by $S_a \preceq_{\mathcal{S}} S_b$. Notice that behaviors are a result of the output sequences, therefore a simulation is a stronger requirement then behavioral inclusion $S_a \preceq_{\mathcal{S}} S_b \implies S_a \preceq_{\mathcal{B}} S_b$. The inverse is only true if S_a is non-blocking and S_b is output deterministic.

Definition 1.20 (Simulation Relation)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$ and a relation $R \subseteq X_a \times X_b$. R is a simulation relation from S_a in relation to S_b if and only if:

- $\forall x_{a0} \in X_{a0} \exists x_{b0} \in X_{b0}$ such that $(x_{a0}, x_{b0}) \in R$
- $\forall (x_a, x_b) \in R$ it holds that $H_a(x_a) = H_b(x_b)$
- $\forall (x_a, x_b) \in R$ it holds that $\forall u_a in U_a(x_a),\ x'_a \in X_a$ such that $(x_a, u_a, x'_a) \in E_a$, $\exists (x_b, u_b, x'_b) \in E_b$ such that $(x'_a, x'_b) \in R$

Remark. Some small remarks on the three bullets in the definition. The first bullet describes that it is necessary to relate all initial states. The second bullet denotes that related states must have the same output. The last bullet describes the progression, if two states are related, then there must exists a successor in S_b related to the successor in S_a.

If there exists a relation R such that R is a simulation relation from S_a towards S_b and vice verse R^{-1} is a simulation relation towards S_b, then R is a bisimulation relation and S_a is bisimilar to S_b as shown in Definition (1.21). The notion of two systems being bisimilar is one of the strongest relations there is. It is no surprise that a bisimulation implies behavioral equivalence: $S_a \cong_{\mathcal{S}} S_b \implies S_a \cong_{\mathcal{B}} S_b$. Recall from the simulation relation, if S_a and S_b are non-blocking and output deterministic, behavioral equivalence will also imply the two systems are bisimilar.

Definition 1.21 (Bisimulation)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$ and a relation $R \subseteq X_a \times X_b$. If R is a simulation relation such that S_b simulates S_a and R^{-1} from is a simulation relation such that S_a simulates S_b, then S_a is bisimilar to S_b and will be denoted with $S_a \cong_{\mathcal{S}} S_b$. Furthermore, R is called a bisimulation relation.

Let us make an useful observation, given two distinctive simulation relations R and R' from S_a to S_b such that S_b simulates S_a, then $R \cup R'$ is also a simulation relation. This result is especially useful in finding a bisimulation. Say we have two systems S_a and S_b and two distinctive relations R and R' such that R is a simulation relation from S_a to S_b and R' is a simulation relation from S_b to S_a. It follows from the previous result that $R \cup R'$ is a bisimulation relation on S_a and S_b, and therefore $S_a \cong_{\mathcal{S}} S_b$.

1-7-7 Quotient system

Previously, we have seen behavioral and simulation relations between two different systems. This theory can also be applied on one and the same system, which may not be as interesting at first. However, can be used as a tool. Given a system $S = (X, X_0, U, E, Y, H)$ and a relation $Q \subseteq X \times X$. This relation Q is said to be an *equivalence relation* on S if it groups states with equivalent output, meaning for all $(x, x') \in Q$ it holds that $H(x) = H(x')$.

An equivalence relation can be used to group states with equivalent outputs, the resulting system is called a quotient system. The reduction can be as critical that an infinite state system can be reduced to only a finite state quotient system while describing the same output behavior. A precise definition of the quotient system is given in Definition (1.22).

Definition 1.22 (Quotient System)**.** Given a system $S = (X, X_0, U, E, Y, H)$ and a equivalence relation $Q \subseteq X \times X$ on X. Let $S_{/Q} = (X_{/Q}, X_{/Q0}, U_{/Q}, E_{/Q}, Y_{/Q}, H_{/Q})$ be the quotient system of S by Q with:

- $X_{/Q} = X/Q$
- $X_{/Q0} = \left\{x_{/Q} \in X_{/Q} \mid x_{/Q} \cap X_0 \neq \varnothing\right\}$
- $U_{/Q} = U$
- $E_{/Q} = \{(x_{/Q}, u, x'_{/Q}) | \exists x \in x_{/Q},\ x' \in x'_{/Q} \text{ such that } (x, u, x') \in E\}$
- $Y_{/Q} = Y$
- $H_{/Q}\left(x_{/Q}\right) = H(x)$ for some $x \in x_{/Q}$

In the literature, the quotient system $S_{/Q}$ is also referred to as a symbolic system. Notice by construction that there exists a simulation relation from $S_{/Q}$ to S, meaning $S \preceq_{\mathcal{S}} S_{/Q}$. The converse is only true when Q becomes a bisimulation on S and itself.

1-7-8 Alternating relations

As Tabuada states beautifully in [36, p. 40] "*Simulation relations require the matching of transitions while in problems of control we require the existence of inputs enforcing desired transitions.*", the simulation relations from Definition (1.20) are slightly misaligned for control purposes. Namely, if a controller is in relation to the system, we want the input of the controller always being able to stay inside the relation. Namely, given a state and input with multiple nondeterministic possible post states. The simulation relation can allow only one of the resulting post states instead of including all of them.

For this purpose, they introduce the *alternating simulation relation* as shown in Definition (1.23). The difference in comparison to the simulation relation from Definition (1.20) is in the last bullet of the definition.

Definition 1.23 (Alternating Simulation Relation)**.** Given two systems
$S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$ and a relation $R \subseteq X_a \times X_b$. R is an alternating simulation relation from S_a to S_b if and only if:

- $\forall x_{a0} \in X_{a0} \exists x_{b0} \in X_{b0}$ such that $(x_{a0}, x_{b0}) \in R$
- $\forall (x_a, x_b) \in R$ it holds that $H_a(x_a) = H_b(x_b)$
- $\forall (x_a, x_b) \in R$ it holds that $\forall u_a \in U_a(x_a) \exists u_b \in U_b(x_b)$ such that for all possible $(x_b, u_b, x_b') \in E_b$ there exists a $(x_a, u_a, x_a') \in E_a$ such that $(x_a', x_b') \in R$

Remark. The last bullet in the definition differs from the definition of simulation relation in Definition (1.20). From a relation point (x_a, x_b) and an input of S_a, there should exist an input in S_b such that every reachable state x_b' in S_b has a relatable state x_a' in S_a. Notice that when the systems S_a and S_b are deterministic, the notion of simulation and alternating simulation are equivalent.

Given two systems S_a and S_b with equivalent output sets $Y_a = Y_b$ and there exists an alternating simulation relation R from S_a to S_b, then S_b alternatingly simulates S_a and is denoted by $S_a \preceq_{\mathcal{AS}} S_b$. The concept of an alternating simulation can be extended in the same manner as before to define an alternating bisimulation as shown in Definition (1.24).

Definition 1.24 (Alternating Bisimulation)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$ and a relation $R \subseteq X_a \times X_b$. If R is an alternating simulation relation such that S_b alternatingly simulates S_a and R^{-1} from is an alternating simulation relation such that S_a alternatingly simulates S_b, then S_a is alternatingly bisimilar to S_b and will be denoted with $S_a \cong_{\mathcal{AS}} S_b$. Furthermore, R is called an alternating bisimulation relation.

In the prospect of control, it is useful to make a small extension of the alternating simulation relations since it will be useful to incorporate relating the inputs as well in the relation. Therefore, the *extended alternating simulation relation*, as shown in Definition (1.25), is defined as an alternating simulation relation in both the states and the inputs.

Definition 1.25 (Extended Alternating Simulation Relation)**.** Given two systems $S_a = (X_a, X_{a0}, U_a, E_a, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, E_b, Y_b, H_b)$ with equivalent output sets $Y_a = Y_b$ and an alternating simulation relation $R \subseteq X_a \times X_b$ from S_a to S_b. Define the extended alternating simulation relation by $R^e \subseteq X_a \times X_b \times U_a \times U_b$ such that for all $(x_a, x_b, u_a, u_b) \in R^e$ the following conditions hold:

- $(x_a, x_b) \in R$
- $u_a \in U_a(x_a)$
- $u_b \in U_b(x_b)$ and for all $x_b' \in X_b$ such that $(x_b, u_b, x_b') \in E_b$ there exists a $x_a' \in X_a$ such that $(x_a, u_a, x_a') \in E_a$ and $(x_a', x_b') \in R$

1-7-9 Control

Let S be our original system we would like to control towards some desired behaviors described by a system S_b. However, S does not meet these desired properties $S \not\preceq S_b$. In the control problem, we want to design an system S_c and an interconnection relation $\mathcal{I}$, such that the controlled system does meet the required behaviors $S_c \times_{\mathcal{I}} S \preceq S_b$ or becomes even bisimilar to them $S_c \times_{\mathcal{I}} S \cong S_b$.

The type of control we will elaborate on in the feedback composition, which essentially determines the next input from a given sequence of previous states and can act as a state-feedback controller. Given a system S_c, this system is *feedback composable* to S if there exists an alternating simulation relation R such that $S_c \preceq_{\mathcal{AS}} S$. When two systems are feedback composable, we can construct the *feedback composition* as defined in 1.26.

Definition 1.26 (Feedback Composition)**.** Given two systems S and S_c such that S_c is feedback composable with S by an alternating simulation relation R. Let $\mathcal{F} = R^e$ be an interconnection relation, defined by the extended alternating simulation relation of R. Then the feedback composition of S_c and S is given by $S_c \times_{\mathcal{F}} S$

A small illustration can clarify the working procedure of such a control process by the feedback composition. Given two systems $S = (X, X_0, U, E, Y, H)$ and $S_c = (X_c, X_{c0}, U_c, E_c, Y_c, H_c)$ such that S_c is feedback composable to S by an alternating simulation relation R. Let $\mathcal{F} = R^e$ be the extended relation such that the feedback composition of S_c and S is given by $S_c \times_{\mathcal{F}} S$. Starting from some state $(x_c, x) \in R$ the control scheme works as follows:

1. The controller chooses any of their possible inputs $u_c \in U_c(x_c)$
2. The system S chooses any input $u \in U(x)$ that is corresponding to the interconnection relation $(x_c, x, u_c, u) \in \mathcal{F}$
3. System S progresses to a new state x' for which there exists a transition $(x, u, x') \in E$
4. The controller S_c needs to measure the new state x' in S and take any transition $(x_c, u_c, x_c') \in E_c$ such that the new state x_c' is related by R, i.e. $(x', x_c') \in R$

A feedback composition can thus reduce the possible behaviors the original system can exhibit by excluding possible inputs from the controller. Moreover, even initial states of the original system can be nullified by not including them in the alternating simulation relation. Theoretically, the controlled system is simulated by the controller $S_c \times_{\mathcal{F}} S \preceq_{\mathcal{S}} S_c$, and therefore the controlled system can only experience behaviors the controller has as well.

This notion of control in the TS perspective can be applied to playing safety games, reachability games, behavioral games or simulation games. However, the required knowledge for these topics in this thesis project will be given outside of the system definition of Tabuada. The reader is referred to [36] for more profound examples.

1-8 Traffic Model

Given an autonomous system $S_{\mathcal{Q}}$ which describes the IST behavior of the original system S for a given static feedback gain K and triggering condition $\mathcal{Q}$. Recall the original system S is a LTI with measurable states. The system $S_{\mathcal{Q}} = (X_{\mathcal{Q}}, X_{\mathcal{Q}0}, U_{\mathcal{Q}}, E_{\mathcal{Q}}, Y_{\mathcal{Q}}, H_{\mathcal{Q}})$ is given by:

- $X_{\mathcal{Q}} = \mathbb{R}^n$
- $X_{\mathcal{Q}0} = X$
- $U_{\mathcal{Q}} = \varnothing$

- $E_{\mathcal{Q}} = \left\{(x, u, x') \mid x' = e^{A\tau(\mathbf{x})}x(t_0)\right\}$
- $Y_{\mathcal{Q}} = T$
- $H_{\mathcal{Q}}(\mathbf{x}) = \tau(\mathbf{x})$

Remark. Let $T = \{\tau = k\Delta \mid k \in \mathbb{N},\ 1 \leq \tau \leq \tau_{\max}\}$ is the set of all possible IST. Recall $\Delta \geq 0$ is the sampling frequency and $\tau_{\max}$ is the maximum IST, i.e. the heartbeat of the system. Define $\tau : X \to T$ to be the function which describes at a state $\mathbf{x}$ the time $\tau(\mathbf{x})$ the system can flow until the triggering condition is violated. This will be defined more clearly later in this section in Eq. (1-17).

By modeling the traffic of IST behavior of the system $S_{\mathcal{Q}}$, one can construct a quotient system $S_{\mathcal{Q}/Q}$ by introducing a equivalence relation $Q \subseteq X_{\mathcal{Q}} \times X_{\mathcal{Q}}$. The benefit being that the infinite system $S_{\mathcal{Q}}$ can be simulated by a finite state system $S_{\mathcal{Q}/Q}$, i.e. $S_{\mathcal{Q}} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q}$. Therefore, also all behaviors of $S_{\mathcal{Q}}$ are captured in $S_{\mathcal{Q}/Q}$, recall $S_{\mathcal{Q}} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q} \implies S_{\mathcal{Q}} \preceq_{\mathcal{B}} S_{\mathcal{Q}/Q}$.

1-8-1 Quotient model

In order to construct the quotient model $S_{\mathcal{Q}/Q}$ from $S_{\mathcal{Q}}$, we need to partition the state space in a smart manner and adjust the transition and output map accordingly.

Notice that for the system $S_{\mathcal{Q}}$ with a given state $\mathbf{x}_0$, the triggering time $\tau(\mathbf{x}_0)$ is invariant under a multiplication with the initial state, i.e. for $\alpha \in \mathbb{R}_{\neq 0}$ holds $\tau(\mathbf{x}_0) = \tau(\alpha\mathbf{x}_0)$, see appendix A-3 for a more detailed proof. Therefore, we can group all points that lay on a line thought the origin (except the origin itself) into a single state, since they have the same output $H_{\mathcal{Q}}(\mathbf{x}_0) = H_{\mathcal{Q}}(\alpha\mathbf{x}_0)$. Due to this scaling, only rotation needs to be accounted for in the transitions $E_{\mathcal{Q}}$.

Partitioning the system this way, still results in an infinite system. Therefore, we make the critical observation that for PETC systems, the IST behavior is discrete and when inducing a maximum $\tau_{\max}$, we are guaranteed to only have a finite amount $|T|$ of possible IST. If we now group all states that have equivalent IST, we are guaranteed to obtain a quotient system with a finite amount of states.

Introducing notations

Now the idea is clear, let us define the system S dynamics flowing for τ time by $M(\tau)$ as shown in Eq. (1-15). Moreover, we can use M in the following way: $M(\tau)\mathbf{x}(t_0) = \mathbf{x}(t_0 + \tau)$, for all $\tau \in \mathbb{R}_{\geq 0}$ and $\mathbf{x}(t_0) \in X$ as long the triggering condition is not met.

Recall the original triggering condition $\xi^\top(t)\mathcal{Q}\xi(t) > 0$ for $\xi(t) = \left[\mathbf{x}^\top(t),\ \hat{\mathbf{x}}^\top(t)\right]^\top$. Now, the difference between $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ can be expressed by M. Therefore, the triggering condition can be rewritten to $\mathbf{x}^\top(t)N(\tau)\mathbf{x}(t) > 0$, where N is given by Eq. (1-16) and τ the time after t, meaning $\mathbf{x}(\tau + t) = M(\tau)\mathbf{x}(t)$. This N can then be used to express the next IST at a given state by $\tau(\mathbf{x})$ defined in Eq. (1-17). Notice, in the condition $\mathbf{x}^\top(t)N(\tau)\mathbf{x}(t) > 0$, the state $\mathbf{x}(t)$ is stationary, while τ can be slowly increased. This in contrast to the condition $\xi^\top(t)\mathcal{Q}\xi(t) > 0$, where $\hat{\mathbf{x}}(t)$ is stationary and $\mathbf{x}(t)$ is slowly increased.

$$M(\tau) := \mathrm{e}^{A\tau} + \int_0^\tau \mathrm{e}^{At}\mathrm{d}tBK \tag{1-15}$$

$$N(\tau) := \begin{bmatrix} M(\tau) \\ \mathbf{I} \end{bmatrix}^\top \mathcal{Q} \begin{bmatrix} M(\tau) \\ \mathbf{I} \end{bmatrix} \tag{1-16}$$

$$\tau(\mathbf{x}) = \min\left\{\tau = k\Delta \mid k \in \mathbb{N}, k\Delta \leq \tau_{\max}, \mathbf{x}^\top N(\tau)\mathbf{x} > 0 \text{ or } k\Delta = \tau_{\max}\right\} \tag{1-17}$$

We can now mathematically define the sets of states which have equivalent IST output behavior. First, introduce the sets $\boldsymbol{\mathcal{K}}_k$, describing which states have already triggered after $\tau = k\Delta$ time, as shown in Eq. (1-18). The set of states which will trigger exactly after $k\Delta$ time is defined by $\boldsymbol{\mathcal{Q}}_k$ by taking the set $\boldsymbol{\mathcal{K}}_k$ and remove all states that have already triggered in $\boldsymbol{\mathcal{K}}_{k'}$ for all $k' < k$, as shown in Eq. (1-19).

$$\boldsymbol{\mathcal{K}}_k := \begin{cases} \left\{\mathbf{x} \in \mathcal{X} \mid \mathbf{x}^\top N(k\Delta)\mathbf{x} > 0\right\}, & \tau < \tau_{\max} \\ \mathbb{R}^{n_x}, & k\Delta = \tau_{\max} \end{cases} \tag{1-18}$$

$$\boldsymbol{\mathcal{Q}}_k := \boldsymbol{\mathcal{K}}_k \backslash \left(\bigcup_{j=1}^{k-1} \boldsymbol{\mathcal{K}}_j\right) = \boldsymbol{\mathcal{K}}_k \cap \bigcap_{j=1}^{k-1} \overline{\boldsymbol{\mathcal{K}}}_j \tag{1-19}$$

Furthermore, define the natural projection of a relation $R \subseteq X_a \times X_b$ by $\pi_R : X_a \to X_b$, or more precisely by $\pi_R(X) = \{x_b \in X_b \mid (x_a, x_b) \in R, x_a \in X\}$.

Constructing the quotient system

For each non-empty set $\boldsymbol{\mathcal{Q}}_k$, we can create a quotient state, resulting in quotient states $X_{\mathcal{Q}/Q_1} = \{\boldsymbol{\mathcal{Q}}_k \mid k \in \mathbb{N}, \boldsymbol{\mathcal{Q}}_k \neq \varnothing\}$. Now we can create a relation Q_1 that relates all state-space states with equivalent IST to the associated quotient state:

$$Q_1 = \left\{(x, \boldsymbol{\mathcal{Q}}_k) \mid x \in X_{\mathcal{Q}}, \boldsymbol{\mathcal{Q}}_k \in X_{\mathcal{Q}/Q_1}, x \in \boldsymbol{\mathcal{Q}}_k\right\}$$

This relation $Q_1 \subseteq X_{\mathcal{Q}} \times X_{\mathcal{Q}}$ can be used to construct the quotient system. Notice, in the process of grouping states together, we are likely to introduce non-determinism in the quotient system. The resulting quotient system is given by
$S_{\mathcal{Q}/Q_1} = (X_{\mathcal{Q}/Q_1}, X_{\mathcal{Q}/Q_10}, U_{\mathcal{Q}/Q_1}, E_{\mathcal{Q}/Q_1}, Y_{\mathcal{Q}/Q_1}, H_{\mathcal{Q}/Q_1})$ with the following properties:

- $X_{\mathcal{Q}/Q_1} = \{\boldsymbol{\mathcal{Q}}_k \mid k \in \mathbb{N}, \boldsymbol{\mathcal{Q}}_k \neq \varnothing\}$
- $X_{\mathcal{Q}/Q_10} = X$
- $U_{\mathcal{Q}/Q_1} = U_{\mathcal{Q}} = \varnothing$
- $E_{\mathcal{Q}/Q_1} = \{(\boldsymbol{\mathcal{Q}}_k, u, \boldsymbol{\mathcal{Q}}'_k) \mid \exists x, x' \text{ such that } (x, \boldsymbol{\mathcal{Q}}_k), (x', \boldsymbol{\mathcal{Q}}'_k) \in Q_1 \text{ and } (x, u, x') \in E_{\mathcal{Q}}\}$
- $Y_{\mathcal{Q}/Q_1} = T$
- $H_{\mathcal{Q}/Q_1}(\boldsymbol{\mathcal{Q}}_k) = k\Delta$

Remark. This is the minimal representation of the quotient system. Later on, when we are introducing the l-complete abstractions, the transitions will be defined by the domino rule, resulting in a non-minimal representation. If we construct the transitions this way, it would be defined by transition relation $E'_{\mathcal{Q}/Q_1} = \left\{(x, u, x') \mid x, x' \in X_{\mathcal{Q}/Q_1}\right\}$.

By construction, the quotient system $S_{\mathcal{Q}/Q_1}$ simulates $S_{\mathcal{Q}}$, i.e. $S_{\mathcal{Q}} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_1}$. Therefore, one can use the quotient system for control since it includes all behaviors of $S_{\mathcal{Q}}$.

1-8-2 Expanding to l-complete models

Nevertheless the quotient system $S_{\mathcal{Q}/Q_1}$ simulates $S_{\mathcal{Q}}$, it also may include behavior that the original system can not endure. By introducing l-complete abstractions $S_{\mathcal{Q}/Q_l}$ by a equivalence relation Q_l, one can 'refine' the quotient system to become a closer and closer simulation of the system $S_{\mathcal{Q}}$ by increasing l using the bisimulation algorithm of [36]. Moreover, ideally it may even find a bisimulation from some l and rendering the behavior of $S_{\mathcal{Q}}$ equivalent to the behavior of $S_{\mathcal{Q}/Q_l}$.

First introduced by Moor [10], the strongest l-complete abstraction of a system captures all behaviors of the original system, while having minimal behavior of the l-complete abstraction and able to use output sequences of length l as internal states. This reasoning, applied on PETC in order to model the IST behaviors, is constructed by [37, 38].

The key behind the l-complete abstractions for PETC systems is to extend the quotient states to a sequence of ISTs. Resulting in eliminating behaviors that the system $S_{\mathcal{Q}}$ can not endure. The corresponding transitions now have to conform the domino rule.

Quotient states

We want to relate quotient states to sequences of IST in order to identify more complex behavior of the system $S_{\mathcal{Q}}$. We can relate states to such a sequence of ISTs $\tau_1\tau_2\dots\tau_l$ by the inter-sample sequence relation R_l given by Definition (1.27). Notice that we will often switch between time sequences $\tau_1\tau_2\dots\tau_l$ and discrete inter-sample time sequences $k_1k_2\dots k_l$ related to each other by $\tau_i = k_i\Delta$.

Definition 1.27 (Inter-Sample Sequence Relation)**.** From [38]. Given a IST sequence of length l, denote the inter-sample sequence relation $R_l \subseteq X \times \mathbb{N}^l$ by $(x, k_1k_2\dots k_l) \in R_l$, where $k_i = \frac{\tau_i}{\Delta}$, if and only if:

$$\begin{aligned} x &\in \boldsymbol{\mathcal{Q}}_{k_1}, \\ M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_2}, \\ M(k_2\Delta)\, M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_3}, \\ &\vdots \\ M(k_{l-1}\Delta)\dots M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_l} \end{aligned} \tag{1-20}$$

Remark. The relation R_l relates state $\mathbf{x}$ uniquely to an inter-sample sequence.

Define $\boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l} = \{x \in X_Q \mid (x, k_1k_2\dots k_l) \in R_l\}$. Then create the equivalence relation $Q_l \subseteq X_{\mathcal{Q}} \times \subseteq X_{\mathcal{Q}}$, given by $Q_l = \{(x, \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}) \mid x \in \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}, k_i\Delta \in T \text{ for } i \in \{1, 2, \dots, l\}\}$. By construction, $S_{\mathcal{Q}/Q_l}$ simulates $S_{\mathcal{Q}}$. Moreover, due to the increasing sequences, $S_{\mathcal{Q}/Q_{l-1}}$ simulates $S_{\mathcal{Q}/Q_l}$. Therefore, the following observation is shown mathematically in Eq. (1-21).

$$S_{\mathcal{Q}} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_l} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_{l-1}} \preceq_{\mathcal{S}} \cdots \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_1} \tag{1-21}$$

Domino game

Given the equivalence relation Q_l, we automatically construct the quotient system with states, transitions and output map as shown in Definition (1.22). However, an easy way to compute the transitions is to play the domino game (apply the domino rule). Here, a quotient state $\boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}$ can only have transitions to $\boldsymbol{\mathcal{Q}}_{k_2\dots k_lk_{l+1}}$ for variable k_{l+1} as long as the set $\boldsymbol{\mathcal{Q}}_{k_2\dots k_lk_{l+1}}$ is non-empty, i.e. the last $l-1$ discrete inter-sample times of the first quotient state need to correspond to the first $l-1$ discrete inter-sample times of the second quotient state. This is best visualized with an example.

Example 1.1. *Consider a system that can only exhibit behaviors $\mathcal{B} = \{(aab)^*\}$. The quotient model $l = 2$ describes all behaviors of length 2. In Figure 1-8 is on the left a domino game visualized for $l = 2$. Notice that the resulting sequence of symbols from this domino game does not exists inside the behaviors $\mathcal{B}$, since the quotient model simulates the original system. On the right is a domino game for $l = 3$ visualized.*

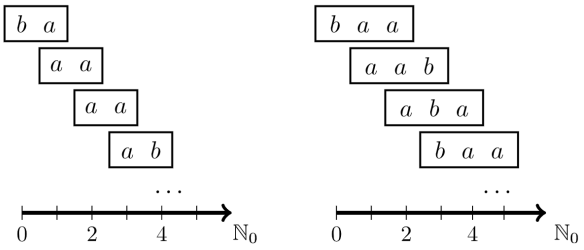


Figure 1-8: Domino game for $l = 2$ (left) and $l = 3$ (right) corresponding to Example 1.1, by Schmuck [12][p. 68]

Remark. Applying the inter-sample sequence relation R_l to the states and the domino rule on the transitions leads to a non-minimal representation of the l-complete abstraction. For example, calculating if a state $\tau_1\tau_2\tau_3$ exists is equivalent to calculating if there exists a transition from $\tau_1\tau_2$ to $\tau_2\tau_3$. The strongest l-complete abstraction is generated by applying R_l on the transitions and use the domino rule to derive the quotient states. In the literature, both forms are used. In order to stay in line with the work by Mazo, we will use the former representation.

1-9 Two Player Games

In game theory, there is a variety of games one can play. In this section we will highlight a few 2-player games, useful in the rest of the thesis.

Two player games are mostly played on an arena $\mathcal{A} = (V_0, V_1, E)$. An arena is a graph where the vertices are divided in two distinct sets V_0 and V_1, each set of vertices is dedicated to respectively player 0 and player 1. The game is played from the perspective of player 0. Therefore the vertex set V_0 is also referred to as the controllable set of vertices and V_1 as the set of uncontrollable vertices. The set of all vertices is described as $V = V_0 \cup V_1$. Furthermore, the edge map $E \subseteq V \times V$ describes which transitions exist in the graph.

A game is played starting from a random starting node. The state of game progresses through the graph, depending if the game state is in a controllable node or uncontrollable node, respectively player 0 and player 1 decides which outgoing edge the game state takes. The game is played indefinitely resulting in an infinite path. Notice both players have knowledge of the full graph when they decide on a transition.

A special form of arena, which often occurs, is the bipartite graph or arena. A arena $\mathcal{A} = (V_0, V_1, E)$ is called bipartite if there exists only edges between V_0 and V_1, i.e. every time a transition is taken, the state of the system switches between being in V_0 and V_1.

1-9-1 Systems and games

We will briefly show the relation between systems and 2-player games and how one can switch between them. Given a system $S = (X, X_0, U, E, Y, H)$ with $X_0 = X$ and $U \neq \varnothing$, one can play an 2-player game on the internal dynamics of S by constructing a bipartite arena. The output map H can be used to determine the objective set as will explained later with reachability and safety games. Create an arena $\mathcal{A} = (V_0, V_1, E')$ where $V_0 = X$ describes all controllable states. In a controllable state $v \in V_0$, one can choose any input $u \in U(v)$. Therefore, for each input $u \in U(v)$ create a connected state in V_1, i.e. $V_1 = \{(v, u) \mid v \in V_0, u \in U(v)\}$ and $E'_0 = \{(v, u, (v, u)) \mid \forall v \in V_0, u \in U(v)\}$ representing all transitions from V_0 to V_1.

Notice, for a given $x \in X$ and $u \in U(x)$, there may multiple $x' \in X$ such that there exists a transition $(x, u, x') \in E$. This non-determinism of the system S is captured in V_1, the set of uncontrollable states. For all $(v, u) \in V_1$, create a transition for all v' such that $(v, u, v') \in E$, i.e. all transitions from V_1 to V_0 are given by $E'_1 = \{(v, u, v') \mid \forall (v, u) \in V_1, v' \in V_0 : (v, u, v') \in E\}$. All transition in the arena are given by $E' = E'_0 \cup E'_1$ and the original system S is now visualized as a bipartite arena $\mathcal{A} = (V_0, V_1, E')$.

From a bipartite arena, one can construct the internal behavior of a system by similar computations as shown above. All states are given V_0. All different inputs at a state $x \in V_0$ are given by the possible edges from V_0 in E. The transitions of the system are given by a combinations of edges from V_0 and V_1 due to edges from V_1 describing the possible non-determinism.

1-9-2 Reachability and safety games

The type of reachability game of interest are played on a non-blocking finite arena $\mathcal{A} = (V_0, V_1, E)$ and a reachable set (objective set) $W \subseteq V = V_0 \sup V_1$. The goal is to construct the attractor set $Att(\mathcal{A}, W)$, defined by all states $v \in V$ for which a strategy exists, that starting from v guarantees to enter the set W at some point.

A simple algorithm to construct the attractor set, given a finite arena is given in the Appendix B-5. The idea is to start with the reachable set $W_0 = W$ and slowly make sets W_i of increasing

size, for which one can guarantee a strategy to reach W_{i-1} in one step. There are more detail given in the appendix, following Gradel et. al.[39]. Notice the difference with Gradel et. al. since we only concern about non-blocking and finite arena's. They also show that for finite arena's, there exists a memoryless strategy and the attractor set is unique given $\mathcal{A}$ and W.

The safety game is the dual problem to the reachability game. A safety game also concerns a non-blocking finite arena $\mathcal{A} = (V_0, V_1, E)$ and a safety set (objective set) W. The goal is to construct the trap set $Trap(\mathcal{A}, W)$, defined by all states $v \in V$ for which no strategy exists, that starting from v guarantees to stay inside W for ever. Since the safety game is the dual problem to the reachability game, we can reuse the reachability algorithm. Construct a new arena $\mathcal{A}' = (V_1, V_0, E)$. Playing the safety game on $\mathcal{A}$ with W is equivalent to playing the reachability game on $\mathcal{A}'$ with $\overline{W} = V \backslash W$, i.e. one can construct the desired trap set by using the reachability algorithm $Trap(\mathcal{A}, W) = Att(\mathcal{A}', \overline{W})$.

Once the trap set $Trap(\mathcal{A}, W) \subseteq V$ is constructed, the safety game is solved. Define the limited safety set $W^e = V \backslash Trap(\mathcal{A}, W)$ as the set of all states, where as long as the play starts in $v \in W^e$, there exists a strategy for player 0 to guarantee to stay outside the trap set $Trap(\mathcal{A}, W)$ for ever. Designing such a strategy can be done in various ways, as long as no edges from W^e to $Trap(\mathcal{A}, W)$ are included.

1-9-3 Mean-payoff games

One of the well studied 2-player games is called the Mean-Payoff Game (MPG). Classically, a player 0 competes against a player 1, by respectively minimizing and maximizing a global quantity. A MPG is played on an arena extended by a weight function $w : E \to \mathbb{R}$. The complete description of a MPG is given by $\Gamma = (V_0, V_1, E, w)$.

The global quantity, which player 0 wants to minimize and player 1 wants to maximize, is given by the average transition weight taken along a path as shown in Eq. (1-22) for a path $r = \{e_i\}_{i=0}^{\infty}$. The global quantity $\nu(r)$ is called the value of the path. Player 0 wants to maximize this quantity, knowing player 1 will play optimally, i.e. maximize $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} w(e_i)$. Identically, player 1 wants to minimize $\limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} w(e_i)$.

$$\nu(r) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} w(e_i) \tag{1-22}$$

Ehrenfeucht and Mycielski shown in [40] that the optimal strategy for both players are positional strategies, meaning at each vertex each player makes the same decision, no matter the history. Moreover, they shown that while both players have different interest, both are only able to guarantee the same value of the game ν when they both play optimally, meaning $\nu = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} w(e_i) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} w(e_i)$. These two findings form the ways one can solve a MPG problem. Since the number of vertices is in general smaller then the possible number of infinite paths possible in a MPG graph, one can focus on the decisions at each independent vertex. Furthermore, a new fundamental question arises, without computing the strategies of both players, can one compute the value of the game ν.

Currently, one of the state of the art methods is Brim et. al. [41] which uses an efficient way to solve energy games as sup-process of solving the MPG. More methods were compared,

however this was the algorithm that could next to calculating the value of the MPG in $\mathcal{O}(nmW)$, create a strategy in quasipolynomial time of $\mathcal{O}(mn^2W(\log n + \log W))$. Here the number of vertices is given by $n = |V|$, the number of edges by $m = |E|$ and the maximum absolute weight by $W = \max_{e \in E}(|w(e)|)$. A more detailed insight of the algorithm is given in the appendix B-4. Notice that recently there has been some good results as well with the use of separating automata and universal graphs as shown in [42]. In specific situation where either the maximum absolute weight W is large, they show a algorithm of time complexity $\mathcal{O}(nm(nW)^{1-\frac{1}{n}})$ for calculating the value of the MPG. Furthermore, if the number of different weights is limited by k, they also show an algorithm of time complexity $O(mn^k)$. However, both algorithms are not more efficient in our situation since we have a lot of small weights, and mainly n and m are the critical factors and only work for solving the decision problem.

So far there has not been many requirements on a MPG $\Gamma = (V_0, V_1, E, w)$. However, it is important to note some underlying assumptions.

- Mainly, the guarantee that a path is always of infinite length and can never end up in a dead-end, i.e. the graph is non-blocking. Some algorithms even require the stricter condition of the graph being strongly connected, meaning every node can be reached from every other node. However, Brim et. al. does not imply this condition.
- From the definition of MPG, one could create an MPG with an infinite amount of nodes. However, as shown by the time complexities of some of the algorithms, this will result in a non-computational problem. Therefore, we require only a finite amount of vertices.
- The most efficient algorithms for solving MPG assume the weight function only maps to a finite countable set, e.g. Brim et. al. uses $w : E \to \{-W, \ldots, W\}$ for some $W \in \mathbb{N}$, which seems like a harsh restriction. However, in some situations the weight function can be adjusted to fit this description by multiplication and addition.
- Not necessarily a requirement, but for some algorithms a property that speeds up computation time, is the MPG graph being bipartite, i.e. every edge form V_0 ends in V_1 and vice versa. Effectively, the players alternate moves.

Chapter 2

Scheduling

In this chapter we will introduce the framework which will be used throughout the thesis, combining knowledge gained from the previous chapter about theoretical preliminaries. With the introduced framework we can make the question we would like to answer more precise and will comment on noteworthy choices made during the thesis project.

2-1 System Introduction

We start with a controllable Linear Time-Invariant (LTI) system that has one equilibrium in the origin. Let $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ for some $n, m \in \mathbb{N}$. Furthermore, let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. Assume all states are measurable (or the system is at least observable) such that the state of the system can be used for state feedback control.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t)$$

For the LTI system, one has determined a stabilizing state feedback gain $K \in \mathbb{R}^{m \times n}$. Therefore, the closed loop system is stable with the input $u(t) = K\mathbf{x}(t)$, i.e. $A + BK$ is Hurwitz.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + BK\mathbf{x}(t) = (A + BK)\mathbf{x}(t)$$

For completeness, we will also describe our system, and all feature permutations, in the notation first introduced by Tabuada in [36] and described by us in Section 1-7. Define the system $S = (X, X_0, U, E, Y, H)$ by

- $X = \mathbb{R}^n$
- $X_0 = X$
- $U = \mathbb{R}^m \times \mathbb{R}_{\geq 0}$
- $E = \{(x, (u, t), x') \mid x' = e^{At}x(t_0) + \int_0^t e^{A(t-\tau)}Bud\tau\}$
- $Y = X$

- $H(\mathbf{x}) = \mathbf{x}$

Remark. The set X contains all state-space coordinates. The input set U is for construction given in a zero-order hold format, meaning elements of $(u, t) \in U$ consists of an input value u with a time t for which the input is held. For our purpose, this input set definition suffices. For continuously changing inputs, a different system must be constructed. The output map H gives information about the internal state since we assume the states are fully measurable.

2-1-1 Periodic Event-Triggered Control of the System

The system is controlled in a Periodic Event-Triggered Control (PETC) manner. The state $\mathbf{x}$ is measured periodically every $\Delta > 0$ seconds. We presume Δ is properly chosen such that all the system behavior is captured and, on the other hand, sampling with time intervals of Δ does not over-sample the system dynamics. The measurement times are given by the time sequence $\{t_k\}_{k\in\mathbb{N}_0}$, described by $t_k = k\Delta$. The new measurement of the state $\mathbf{x}(t)$ is only passed on if $t \in \{t_k\}_{k\in\mathbb{N}_0}$.

Instead of updating the control input every time the state is measured, the control input is only updated in certain events. The sampling times are given by the strictly increasing sequence $\{\hat{t}_i\}_{i\in\mathbb{N}_0}$ which is a subset of the measurement sequence $\{t_k\}_{k\in\mathbb{N}_0}$. Furthermore, the starting times of both sequences are the same: $0 = t_0 = \hat{t}_0$. The Inter-Sample Time (IST) is given by $\tau_i = \hat{t}_{i+1} - \hat{t}_i \geq \Delta > 0$.

The last state known to the controller is given by $\hat{\mathbf{x}}(t)$ and is used to compute the control input. The state $\hat{\mathbf{x}}$ is only updated in one of the triggering times as shown in Eq. (2-1) and afterwards the control input is updated since $u(t) = K\hat{\mathbf{x}}(t)$.

$$\hat{\mathbf{x}}(t) = \begin{cases} \mathbf{x}(t) & \text{if } t \in \{\hat{t}_i\}_{i\in\mathbb{N}_0} \\ \hat{\mathbf{x}}(t) & \text{if } t \notin \{\hat{t}_i\}_{i\in\mathbb{N}_0} \end{cases} \tag{2-1}$$

The closed loop system is therefore given below for $t \in [\hat{t}_i,\ \hat{t}_{i+1})$. Notice the input is updated in a zero-order hold fashion.

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + BK\hat{\mathbf{x}}(t) = A\mathbf{x}(t) + BK\mathbf{x}(\hat{t}_i)$$

In section Section 1-1 some examples of different triggering conditions are given. In this thesis project, a mix of different triggering conditions can be used, with the only requirement being that each triggering condition has a guaranteed decay rate on the Common Lyapunov Function (CLF).

One well studied field in control theory is the use of a quadratic Lyapunov function and triggering conditions. First, we allocate a quadratic function of the form $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$. We choose P positive definite and make sure the continuous Lyapunov condition is met $(A + BK)^\top P + P(A + BK) = -Q$ for some positive definite matrix Q, recall Theorem (1.3). Then we can conclude that the V is a Lyapunov function for our system.

Once the Lyapunov function is fixed, one can use the matrices P and Q to construct triggering mechanisms. As shown in section Section 1-1, there are multiple ways to use these matrices

to construct triggering conditions that guarantee an exponential decay rate of the CLF V. A guaranteed decay rate $\kappa\sigma$ will enforce the Lyapunov function to decay with a rate of $e^{-\kappa\sigma\tau}$ over τ time as shown in Eq. (2-2), as long the corresponding triggering condition is applied. Here, κ denotes the maximum possible Lyapunov decrease of the system with the corresponding Lyapunov function, which is a constant in respect to these quantities. Therefore, we will refer to the guaranteed Lyapunov decay rate as $\sigma \in (0, 1]$ alone, since this is the varying factor per triggering condition. Furthermore, a triggering condition that has a guaranteed Lyapunov decay of σ will be denoted by $\mathcal{Q}_\sigma$ and have the property shown in Eq. (2-2). How one can calculate κ for given system dynamics and Lyapunov function V is shown in appendix A-1.

$$\forall \mathbf{x}(t_0) \in \mathbb{R}^n, \tau \in \mathbb{R}_{\geq 0} : V(\mathbf{x}(t_0 + \tau)) \leq e^{-\kappa\sigma\tau} V(\mathbf{x}(t_0)) \tag{2-2}$$

The triggering time sequence $\{\hat{t}_k\}_{k\in\mathbb{N}_0}$ is determined by the system S, feedback gain K, applied triggering condition $\mathcal{Q}_\sigma$ and initial state $\mathbf{x}_0$, i.e. given a static feedback gain K and triggering condition $\mathcal{Q}_\sigma$, the system S becomes autonomous and is denoted by $S_{\mathcal{Q}\sigma} = (X_{\mathcal{Q}\sigma}, X_{\mathcal{Q}\sigma 0}, U_{\mathcal{Q}\sigma}, E_{\mathcal{Q}\sigma}, Y_{\mathcal{Q}\sigma}, H_{\mathcal{Q}\sigma})$ with the following properties:

- $X_{\mathcal{Q}\sigma} = \mathbb{R}^n$
- $X_{\mathcal{Q}\sigma 0} = X_{\mathcal{Q}\sigma}$
- $U_{\mathcal{Q}\sigma} = \varnothing$
- $E_{\mathcal{Q}\sigma} = \{(x, u, x') \mid x' = M(t)x, \text{ where } t = \tau_\sigma(x)\}$
- $Y_{\mathcal{Q}\sigma} = T$
- $H_{\mathcal{Q}\sigma}(\mathbf{x}) = \tau_\sigma(\mathbf{x})$

Here, we overload again τ to also be the function $\tau_\sigma : X \to T$ which maps every state to the corresponding IST, given the trigger condition $\mathcal{Q}_\sigma$. This function can be mathematically defined likewise as shown in Section 1-8, and is given in Eq. (2-5). First define $M : T \to \mathbb{R}^{n\times n}$ as the state progression after a certain time, described as shown in Eq. (2-3). Our function M can be used as follows: $M(\tau)\mathbf{x}(t_0) = \mathbf{x}(t_0+\tau)$ for some $\tau \in T$ and $\mathbf{x}(t_0) \in \mathbb{R}^n$. Furthermore, M can be used to rewrite the triggering condition $\xi^\top(t)\mathcal{Q}_\sigma\xi(t) > 0$, where $\xi(t) = \left[\mathbf{x}(t)^\top, \hat{\mathbf{x}}^\top(t)\right]^\top$, in to the following form $\mathbf{x}^\top(t)N_\sigma(\tau)\mathbf{x}(t) > 0$. N_σ is given in Eq. (2-4), for more detail I refer to Section 1-8.

$$M(\tau) := \mathrm{e}^{A\tau} + \int_0^\tau \mathrm{e}^{At}\mathrm{d}tBK \tag{2-3}$$

$$N_\sigma(\tau) := \begin{bmatrix} M(\tau) \\ \mathbf{I} \end{bmatrix}^\top \mathcal{Q}_\sigma \begin{bmatrix} M(\tau) \\ \mathbf{I} \end{bmatrix} \tag{2-4}$$

$$\tau_\sigma(\mathbf{x}) = \min\left\{\tau = k\Delta \mid k \in \mathbb{N}_\sigma, k\Delta \leq \tau_{\max}, \mathbf{x}^\top N_\sigma(\tau)\mathbf{x} > 0 \text{ or } k\Delta = \tau_{\max}\right\} \tag{2-5}$$

Notice the difference in the definition of N_σ and τ_σ as to Section 1-8, since the triggering condition can differ.

2-1-2 Switched System

The scheduler operates as a supervisor that is able to switch between triggering conditions at any point. However, since we do not incorporate early triggering in this thesis project, we only evaluate the triggering condition choice at the triggering instances. Given a finite set of triggering conditions $\{\mathcal{Q}_{\sigma_1}, \mathcal{Q}_{\sigma_2}, \dots, \mathcal{Q}_{\sigma_s}\}$ with each a different guaranteed decay rate of the CLF V given by respectively $\sigma_1, \sigma_2, \dots, \sigma_s$. Define the set of all triggering decay rates by $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_s\}$. Since the triggering conditions $\mathcal{Q}_\sigma$ are unique liked with their respective Lyapunov decay rates σ, we can use σ as a designator, uniquely matching to a triggering condition. Notice that all following theory does also apply if one wants to use two different triggering conditions with the same guaranteed decay rate of V, only the unique triggering matrices should be used as designators.

The scheduling problem describes how switching between triggering conditions can benefit the overall desired properties of the system. Recall from Section 1-3 that we can model this system dynamics as a switched system. Therefore, the scheduler needs to control a switched system, given by $S_\mathcal{Q} = (X_\mathcal{Q}, X_{\mathcal{Q}0}, U_\mathcal{Q}, E_\mathcal{Q}, Y_\mathcal{Q}, H_\mathcal{Q})$ with the following properties:

- $X_\mathcal{Q} = \mathbb{R}^n \times \Sigma$
- $X_{\mathcal{Q}0} = X_{\mathcal{Q}\sigma}$
- $U_\mathcal{Q} = \Sigma$
- $E_\mathcal{Q} = \{((x, \sigma), u, (x', \sigma')) \mid x' = M(t)x,\ u = \sigma', \quad \text{and } t = \tau_\sigma(x)\}$
- $Y_\mathcal{Q} = T$
- $H_\mathcal{Q}(\mathbf{x}, \sigma) = \tau_\sigma(\mathbf{x})$

Remark. The new set of states $X_\mathcal{Q}$ hold the state-space coordinates together with a current trigger condition identifier. For now, in the initial state, the system can have every triggering condition possible associated. More information on controller synthesis is given in Section 2-4. The finite set $T = \{\tau_\sigma(\mathbf{x}) \mid (\mathbf{x}, \sigma) \in X_\mathcal{Q}\}$ consists of all possible IST the system can endure given any state and triggering condition.

The switched system $S_\mathcal{Q}$ will be the bases for future computations as finding a controller for $S_\mathcal{Q}$ describes the scheduling problem. Recall from Section 1-3-4 the nice possibility of switching triggering conditions while preserving decay of the CLF and thus preserving stability. This fundamental property is the reason why we can schedule a controller. In essence, we use quadratic homogeneous triggering conditions and there nice properties, in order to construct a more complex triggering condition for the system which enhances the performance.

2-2 Scheduling a controller

In the previous Section 2-1 we have introduced our system dynamics and the possibility to apply a variety of triggering conditions that share a CLF. In this section we will show how this applies to scheduling a controller.

First of, in the literature of control systems, there are two methods denoted by scheduling. While one focuses on the planning of multiple control loops over the same network, the other interpretation of scheduling means the planning of the triggering conditions regarding a single

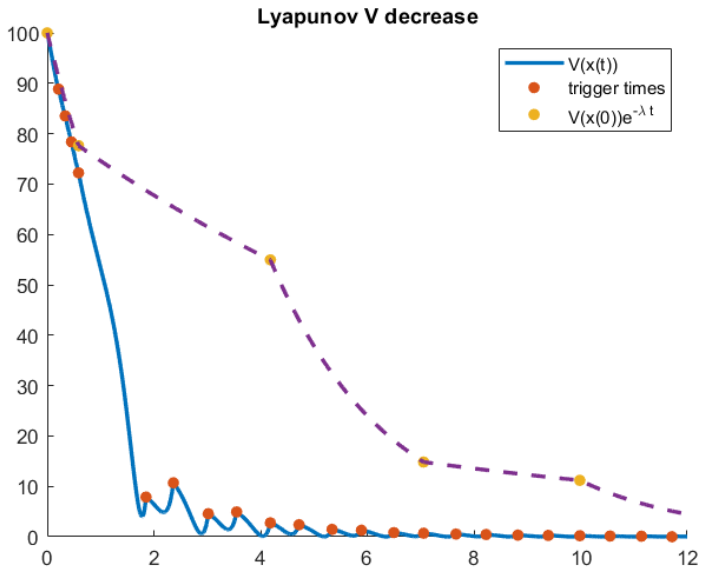


Figure 2-1: Simulation of switched system using Dynamic Triggering Mechanism (DTM) for PETC strategy. Applied to the system dynamics introduced in Chapter 5. Using decay rates $\sigma = 0.95$ or $\sigma = 0.2$ and initial condition $x(0) = [10\ 0]^\top$.

control loop. In this thesis project, we will focus on the latter definition of scheduling and are interested in how to switch triggering conditions in a clever way to increase desired properties.

Recall from Section 1-3 that switching between triggering condition preserves stability as long a CLF can be guaranteed. By the construction of our system as shown in previous Section 2-1, all triggering conditions are obligated to use the same Lyapunov function $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$ and guarantee a rate of decay with respect to this Lyapunov function. Therefore, V is a CLF from a switched system perspective and by switching between triggering conditions, stability is guaranteed.

This form of scheduling is in literature of hybrid systems also described as supervisory control [20, p. 38]. Here, a supervisor determines which controller is allowed to operate the plant, depending on the state of each controller and the state of the plant. They give an example where they map each controller to a region in state space where it operates. In our case, the scheduler (/supervisor) will be capable of more complex behavior and will run an automata next to the physical system. Moreover, instead of switching between controllers, only the triggering conditions will be switched. Therefore, the state progression does not differ between triggering conditions, only the times at which the control input is updated.

2-3 Traffic Abstraction

Recall from Section 2-1 that $S_{\mathcal{Q}}$ is the switched system which combined a set of different triggering condition from a finite set $\{\mathcal{Q}_{\sigma_1}, \mathcal{Q}_{\sigma_2}, \ldots, \mathcal{Q}_{\sigma_s}\}$, each applied in a PETC fashion. The scheduling problem now yields, how to switch between triggering conditions in order to optimize some performance. before we can answer that question, we will construct a quotient

system $S_{\mathcal{Q}/Q_l}$ as l-complete abstraction for the switched system $S_{\mathcal{Q}}$ by the equivalence relation $Q_l \subseteq X_{\mathcal{Q}} \times X_{\mathcal{Q}}$. Moreover, we will make a simulation between the infinite state system and the finite quotient system, which includes all behavior the switched system can endure. Recall the switched system is given by $S_{\mathcal{Q}} = (X_{\mathcal{Q}}, X_{\mathcal{Q}0}, U_{\mathcal{Q}}, E_{\mathcal{Q}}, Y_{\mathcal{Q}}, H_{\mathcal{Q}})$ with the following properties:

- $X_{\mathcal{Q}} = \mathbb{R}^n \times \Sigma$
- $X_{\mathcal{Q}0} = X_{\mathcal{Q}\sigma}$
- $U_{\mathcal{Q}} = \Sigma$
- $E_{\mathcal{Q}} = \{((x,\sigma), u, (x',\sigma')) \mid x' = M(t)x,\ u = \sigma', \text{ and } t = \tau_\sigma(x)\}$
- $Y_{\mathcal{Q}} = T$
- $H_{\mathcal{Q}}(\mathbf{x}, \sigma) = \tau_\sigma(\mathbf{x})$

Before we can construct the equivalence relation Q_l, we have to introduce the sets of states which have triggered $\boldsymbol{\mathcal{K}}$ and who trigger exactly $\boldsymbol{\mathcal{Q}}$, likewise to Section 1-8. However, unlike the traditional traffic models, the (deterministic) inter-sample sequences go now pared with an triggering condition sequence since it will be important to know for how long τ we can guarantee a certain decay rate σ.

We extend inter-sample sequences $\tau_1\tau_2\ldots\tau_l$ to a pairing of two equally long sequences $(\tau_1\tau_2\ldots\tau_l, \sigma_1\sigma_2\ldots\sigma_l)$, where $\tau_i \in T$ and $\sigma_i \in \Sigma$ for all i. likewise for deterministic inter-sample sequences: $(k_1k_2\ldots k_l, \sigma_1\sigma_2\ldots\sigma_l)$, where $\tau_i = k_i\Delta$ for all i. Notice, while we will refer to both as the *(deterministic) inter-sample sequence*, when elaborating on the differences, we will denote the pairing with guaranteed decay rate by the *extended (deterministic) inter-sample sequence*.

Likewise to Section 1-8, we can mathematically define the sets of states which have equivalent IST output behavior, given the input sequence of triggering conditions. First, introduce the sets $\boldsymbol{\mathcal{K}}_k^\sigma$, describing which states have already triggered after $\tau = k\Delta$ time- when applying triggering condition $\mathcal{Q}_\sigma$, as shown in Eq. (2-6). The set of states which will trigger exactly after $k\Delta$ time when applying $\mathcal{Q}_\sigma$ is defined by $\boldsymbol{\mathcal{Q}}_k^\sigma$ by taking the set $\boldsymbol{\mathcal{K}}_k^\sigma$ and remove all states that have already triggered in $\boldsymbol{\mathcal{K}}_{k'}^\sigma$ for all $k' < k$, as shown in Eq. (2-7).

$$\boldsymbol{\mathcal{K}}_k^\sigma := \begin{cases} \left\{\mathbf{x} \in \mathcal{X} \mid \mathbf{x}^\top N_\sigma(k\Delta)\mathbf{x} > 0\right\}, & \tau < \tau_{\max} \\ \mathbb{R}^{n_{\mathrm{x}}}, & k\Delta = \tau_{\max} \end{cases} \tag{2-6}$$

$$\boldsymbol{\mathcal{Q}}_k^\sigma := \boldsymbol{\mathcal{K}}_k^\sigma \setminus \left(\bigcup_{j=\underline{k}}^{k-1} \boldsymbol{\mathcal{K}}_j^\sigma\right) = \boldsymbol{\mathcal{K}}_k^\sigma \cap \bigcap_{j=1}^{k-1} \overline{\boldsymbol{\mathcal{K}}}_j \tag{2-7}$$

2-3-1 Constructing the quotient model

The sets $\boldsymbol{\mathcal{Q}}_k^\sigma$ can be used to construct the extended inter-sample sequence relation $\tilde{R}_l$, given by Definition (2.1). The purpose of $\tilde{R}_l$ is to relate states to possible extended inter-sample sequences. These sequences will form the basis of the quotient states and quotient relation.

Definition 2.1 (Extended Inter-Sample Sequence Relation)**.** Given a IST sequence of length l, denote the extended inter-sample sequence relation $\tilde{R}_l \subseteq X \times \mathbb{N}^l$, where elements $(x, k_1k_2 \dots k_l, \sigma_1\sigma_2 \dots \sigma_l) \in \tilde{R}_l$ with $k_i = \frac{\tau_i}{\Delta}$, if and only if:

$$\begin{aligned} x &\in \boldsymbol{\mathcal{Q}}_{k_1}^{\sigma_1}, \\ M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_2}^{\sigma_2}, \\ M(k_2\Delta)\, M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_3}^{\sigma_3}, \\ &\vdots \\ M(k_{l-1}\Delta) \dots M(k_1\Delta)\, x &\in \boldsymbol{\mathcal{Q}}_{k_l}^{\sigma_l} \end{aligned} \tag{2-8}$$

Remark. Unlike the relation R_l from Section 1-8, the relation $\tilde{R}_l$ relates a state $\mathbf{x}$ to multiple different extended inter-sample sequences. Moreover, for each possible input sequence of triggering conditions, it will relate to one inter-sample time sequence. Therefore, the amount of related sequences by $\tilde{R}_l$ from a state $\mathbf{x}$ is $|\Sigma|^l$.

Using the extended inter-sample sequence relation $\tilde{R}_l$, we can construct quotient states: $\boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}^{\sigma_1\sigma_2\dots\sigma_l} = \left\{(x, \sigma_1) \in X_{\mathcal{Q}} \mid (x, k_1k_2 \dots k_l, \sigma_1\sigma_2 \dots \sigma_l) \in \tilde{R}_l\right\}$ for any extended deterministic inter-sample sequence $(k_1k_2 \dots k_l, \sigma_1\sigma_2 \dots \sigma_l)$, which groups a set of states from $X_{\mathcal{Q}}$.

The chosen equivalence relation simply couples the states to their future inter-sample sequences for varying input sequences of triggering conditions. The resulting equivalence relation $Q_l \subseteq X_{\mathcal{Q}} \times X_{\mathcal{Q}}$ is given by Eq. (2-9).

$$Q_l = \left\{\left((\mathbf{x}, \sigma_1), \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}^{\sigma_1\sigma_2\dots\sigma_l}\right) \mid (\mathbf{x}, \sigma_1) \in \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}^{\sigma_1\sigma_2\dots\sigma_l}\right\} \tag{2-9}$$

By the definition of a quotient system in Section 1-7-7, the quotient system $S_{\mathcal{Q}/Q_l}$ is defined purely by the corresponding equivalence relation Q_l and system $S_{\mathcal{Q}}$. However, for computation purposes one can use the domino rule to construct the transitions $E_{\mathcal{Q}/Q_l}$. For completeness, the resulting quotient system of $S_{\mathcal{Q}}$ by Q_l is given by $S_{\mathcal{Q}/Q_l} = \left(X_{\mathcal{Q}/Q_l}, X_{\mathcal{Q}/Q_l 0}, U_{\mathcal{Q}/Q_l}, E_{\mathcal{Q}/Q_l}, Y_{\mathcal{Q}/Q_l}, H_{\mathcal{Q}/Q_l}\right)$ with the following properties:

- $X_{\mathcal{Q}/Q_l} = \pi_{X_{\mathcal{Q}}}(Q_l) = \left\{x_{/Q} = \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}^{\sigma_1\sigma_2\dots\sigma_l} \mid x_{/Q} \neq \varnothing \text{ and } \sigma_i \in \Sigma, \tau_i \in T \forall i\right\}$
- $X_{/Q0} = \left\{x_{/Q} \in X_{\mathcal{Q}/Q} \mid x_{/Q} \cap X_{\mathcal{Q}0} \neq \varnothing\right\} = X_{\mathcal{Q}/Q_l}$
- $U_{\mathcal{Q}/Q_l} = U_{\mathcal{Q}} = \Sigma$
- $E_{\mathcal{Q}/Q_l} = \left\{\left(x_{/Q}, u, \hat{x}_{/Q}\right) \mid x_{/Q} = \boldsymbol{\mathcal{Q}}_{k_1k_2\dots k_l}^{\sigma_1\sigma_2\dots\sigma_l}, \hat{x}_{/Q} = \hat{\boldsymbol{\mathcal{Q}}}_{k_2k_3\dots k_{l+1}}^{\sigma_2\sigma_2\dots\sigma_{l+1}}, u = \sigma_1\right\}$
- $Y_{\mathcal{Q}/Q_l} = Y_{\mathcal{Q}} = T$
- $H_{\mathcal{Q}/Q_l}\left(x_{/Q_l}\right) = H_{\mathcal{Q}}\left((\mathbf{x}, \sigma)\right) = \tau_\sigma(\mathbf{x})$ for $x_{/Q} \in X_{\mathcal{Q}/Q_l}$ and for any $(\mathbf{x}, \sigma) \in x_{/Q}$

Remark. Note we use the domino rule to construct the transitions instead of practically calculating the $l+1$ states, see example 1.1 and the remark for more details. Moreover, notice $S_{\mathcal{Q}/Q_l}$ will be non-blocking, and even internally strongly connected from a graph perspective.

By construction, the quotient system $S_{\mathcal{Q}/Q_l}$ simulates the switched system $S_{\mathcal{Q}}$ and for increasing values of l, simulates more narrowly. Recall that $S_a \preceq_{\mathcal{S}} S_b \implies S_a \preceq_{\mathcal{B}} S_b$. The number of

transitions for every state in $S_{\mathcal{Q}/Q_l}$ is at least the amount of triggering conditions $|\mathcal{Q}|$. When there are exactly $|\mathcal{Q}|$ transitions for every state, the system has become deterministic.

$$\begin{aligned} S_{\mathcal{Q}} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_l} \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_{l-1}} \preceq_{\mathcal{S}} \cdots \preceq_{\mathcal{S}} S_{\mathcal{Q}/Q_1} \\ \Longrightarrow \\ S_{\mathcal{Q}} \preceq_{\mathcal{B}} S_{\mathcal{Q}/Q_l} \preceq_{\mathcal{B}} S_{\mathcal{Q}/Q_{l-1}} \preceq_{\mathcal{B}} \cdots \preceq_{\mathcal{B}} S_{\mathcal{Q}/Q_1} \end{aligned} \tag{2-10}$$

Notice that the choice to use future inter-sample sequences, limits the amount states to incorporate in the abstraction in comparison to interpreting the inter-sample sequences as the history a state has already endured. Both interpretations are slightly different in creating the quotient system. Mainly, when fixing the history of a state, one needs additional states when starting up the simulation, since there exists no present history. However, this leaves synchronizing the controller as an open question, which will be dealt with in the next section.

As an alternative to the l-complete transitions defined earlier by $E_{\mathcal{Q}/Q_l}$. We will also define the transition set $\mathbf{E}_{\mathcal{Q}/Q_l}$, which distinguishes inputs with the last guaranteed decay rate σ_l in the sequence instead of the first σ_1. While this transition set does not oblige as the l-complete abstraction, it will be useful once we will apply some control on the abstracted system.

$$\mathbf{E}_{\mathcal{Q}/Q_l} = \left\{ \left(x_{/Q}, u, \hat{x}_{/Q} \right) \mid x_{/Q} = \boldsymbol{\mathcal{Q}}^{\sigma_1\sigma_2\ldots\sigma_l}_{k_1k_2\ldots k_l}, \hat{x}_{/Q} = \hat{\boldsymbol{\mathcal{Q}}}^{\sigma_2\sigma_2\ldots\sigma_{l+1}}_{k_2k_3\ldots k_{l+1}}, u = \sigma_l \right\} \tag{2-11}$$

2-3-2 Controlling the scheduler

We can design a controller (e.g. S_c) for the finite state l-complete abstraction $S_{\mathcal{Q}/Q_l}$ as shown in Section 1-7-9, by a feedback composition. Let S_c and $S_{\mathcal{Q}/Q_l}$ be feedback composable, i.e. there exists an alternating simulation relation R such that $S_{\mathcal{Q}/Q_l}$ alternatively simulates S_c: $S_c \preceq_{\mathcal{AS}} S_{\mathcal{Q}/Q_l}$. This requirement allows us to pick the desired input in S_c, while the system behavior is described by $S_{\mathcal{Q}/Q_l}$, which will be a necessary condition explained in more detail in Section 1-7-9 and applied to the current circumstances in Section 2-4-1. Let $F = R^e$ be an interconnection relation, the extended alternating simulation relation of R to incorporate the matching inputs. Then the controlled dynamics of the scheduler are described by: $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$. Moreover, S_c can limit the behaviors $S_{\mathcal{Q}/Q_l}$ can endure, as well as the behaviors of $S_{\mathcal{Q}}$. A properly chosen controller S_c can even optimize quantities which are deemed desirable.

2-4 Controller Synthesis

Controller synthesis describes the connection between the abstractions we design and the 'physical' system S. So far in this chapter, we have done some work to construct the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ of the switched system $S_{\mathcal{Q}}$ which describes the scheduling problem. In this section we will practically describe how one can use a controller designed on a scheduler. Furthermore, there is some nuance in the initial synchronization of the system and the scheduler which will we will address.

2-4-1 Process of synthesis

Recall the example given in Section 1-7-9 on how a feedback composition of a controller and system operate. This line of thinking can be easily applied to the scheduling problem. As shown in Section 2-3-2, we would like to design a system S_c, controlling the abstraction of the scheduler $S_{\mathcal{Q}/Q_l}$ by a interconnection relation F. The new behavior the controlled scheduler can endure is then given by $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$.

Following the example of Section 1-7-9, while relating the system S to the l-complete abstraction $S_{\mathcal{Q}/Q_l}$, describes the controller synthesis. The process can be divided in an *update* and *implement* phase. First, the state $\mathbf{x} \in X$ is measured in the system S when the current triggering condition is met and the update phase starts, where $S_{\mathcal{Q}/Q_l}$ and later S_c, get updated on which transition to take from the previous state. Secondly, S_c decides on a preferable input, and this gets implemented back to an input in $S_{\mathcal{Q}/Q_l}$ and the next triggering condition is set on S. Awaiting the next triggering instance. This process is more elaborate demonstrated below:

Update phase

1. When the current triggering condition $\mathcal{Q}_{\sigma_i}$ triggers, measure the state $\mathbf{x} \in X$
2. Automatically update $\hat{\mathbf{x}}_i = \mathbf{x}$ and the input signal $u = K\hat{\mathbf{x}}_i$ to system S
3. In $S_{\mathcal{Q}/Q_l}$, from the previous state $x_{i-1/Q} \in X_{\mathcal{Q}/Q}$ related to $(\hat{\mathbf{x}}_{i-1}, \sigma_{i-1})$ by Q_l, take any transition labelled with σ_i such that we arrive at a new state $x_{i/Q} \in X_{\mathcal{Q}/Q}$ related to $(\hat{\mathbf{x}}_i, \sigma_i)$ by Q_l.
4. In S_c, take a transition labelled with σ'_i (the previous iteration chosen input) from the previous state $x_{i-1,c}$, to the new state $x_{i,c}$, which is uniquely related to $x_{i/Q}$ by R, i.e. $\left(x_{i/Q}, x_{i,c}\right) \in R$.

Implement phase

5. In S_c, choose a new control input $\sigma'_{i+1} \in U_c(x_{i,c})$
6. Relate the control input σ'_{i+1} back to a unique $\sigma_{i+1} \in U_{\mathcal{Q}/Q_l}(x_{i/Q})$ by F, i.e. $\left(x_{i/Q}, x_{i,c}, \sigma_{i+1}, \sigma'_{i+1}\right) \in F$
7. Apply σ_{i+1} and set the new triggering condition $\mathcal{Q}_{\sigma_{i+1}}$

Remark. This explanation is still from the theoretical view form Section 1-3. Some practical remarks for implementation are given below.

- This looks like an time consuming process. However, it consists of lookup tables and in the case of relating states, it will usually have very few options, sometimes only even one when deterministic.
- Regarding the third bullet, relating to a quotient state $x_{i/Q}$, i.e. $\left((\hat{\mathbf{x}}_i, \sigma_i), x_{i/Q}\right) \in Q_l$, simply comes down to checking if $\hat{\mathbf{x}}_i \in x_{i/Q}$ for the possible quotient states $x_{i/Q}$.
- Regarding the fourth bullet, the states for our controller S_c will be a subset of $S_{\mathcal{Q}/Q_l}$, i.e. $X_c \subseteq X_{\mathcal{Q}/Q_l}$. Therefore, relating states in this step is a trivial task.
- Regarding the sixth bullet, the inputs σ' in S_c represent the input applied to S after $l+1$ events, while the inputs σ in S and $S_{\mathcal{Q}/Q_l}$ represent the current triggering condition.

- Moreover the seventh bullet, due to the l-complete abstractions, the next l triggering conditions are already fixed at every state in $S_{\mathcal{Q}/Q_l}$ and S_c and some implementation computational costs can be saved.

It is important to note that there is some nuance for the first $l-1$ events at startup since a state $\mathbf{x} \in X$ will not be uniquely linked to a state of $S_{\mathcal{Q}/Q_l}$. This will be discussed in next on how to synchronize the scheduler and the system.

2-4-2 Synchronization

At startup, a measured state $\mathbf{x}_0 \in X$ of the system S has $|\Sigma|^l$ possible quotient states in the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ due there being $|\Sigma|^l$ possible input sequences, each rendering exactly one related quotient state. When no controller S_c is applied to the scheduler, all these possible input sequences still generate viable quotient states. Even when a controller S_c is designed there may be multiple possible viable quotient states $\{x_{1/c}, x_{2/c}, \ldots, x_{q/c}\} \subseteq X_c$ for some $q \leq |\Sigma|^l$ for an initial measured state $\mathbf{x}_0 \in X$, i.e. for all $x_{/c} \in \{x_{1/c}, x_{2/c}, \ldots, x_{q/c}\}$ it holds that $\mathbf{x}_0 \in x_{/c}$.

This problem arises due to our choice of quotient states representing future inter-sample sequences and triggering conditions, instead of them describing the past and incorporate extra states to handle the startup face for when there is no past l events so far. However, we argue that having multiple quotient states $x_{/Q}$ related to a single initial measured state x_0 is no problem for the abstracted scheduler $S_{\mathcal{Q}/Q_l}$ and neither it is when a controller S_c is designed for the scheduler.

Regarding the l-complete abstraction $S_{\mathcal{Q}/Q_l}$, the system just describes all possible behaviors, and therefore may include multiple possible future input sequences. For designing a controller S_c, we give a more elaborate explanation.

When a controller S_c is designed and $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$ still relates multiple quotient states, with one initial measured state x_0, as long as S_c guarantees the desired properties, one can freely choose any to relate $x_{/c} \in \{x_{1/c}, x_{2/c}, \ldots, x_{q/c}\}$ as a starting point. Since S_c guarantees the desired properties for all starting quotient states, every one is a valid option. When implementing, one could make an arbitrary decision to prioritize some quotient states over others by altering X_{c0} or just pick a random quotient state from the set of viable states $\{x_{1/c}, x_{2/c}, \ldots, x_{q/c}\}$.

Chapter 3

Cost Function Approach

The first approach to designing a controller for the scheduler, was to design a common cost function in which one could prioritize Inter-Sample Time (IST) and control performance accordingly. Reducing the multi objective optimization problem, to a single objective optimization problem. The idea being that by using different balances of IST and control performance, one could estimate the Pareto frontier.

Unfortunately, this approach failed. However, I am still going to introduce the concepts and explain why it ended up not working. In this chapter, we will mathematically define control performance and will state which quantities are of interested. We will keep it short, and let it be an introduction to the next chapter where we come up with a solution by slightly adjusting our objective.

3-1 Objective Description

We are interested in the infinite behavior patterns of the controlled system, i.e. the infinite horizon problem. Since the behaviors of the scheduler $S_{\mathcal{Q}/Q}$ includes the behavior of the switched system $S_{\mathcal{Q}}$, we want to limit the behaviors $S_{\mathcal{Q}/Q}$ can endure by designing a controller S_c as described in Section 2-3-2. Since $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q}$ is a finite state system, we can view the internal dynamics as a finite directed graph and use some of the results from Section 1-2. Namely, when interested in an infinite path of behaviors, it is sufficient to focus on the separate primitive cycles in the graph, i.e. $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q}$.

Given a primitive cycle c, denote by $|c|$ the amount of unique vertices in c (or equivalent, the amount of unique edges in c). Since cycles are now taken from states and transitions in $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q}$, elements of c will be in $X_{\mathcal{Q}/Q} \subseteq X \times \Sigma$. The quantities of interest are the average inter-sample times and the average control performance.

3-1-1 Average IST and control performance

The average IST of a cycle is simply given by the sum of all inter-sample times divided by the number of edges in the cycle: $\frac{\sum_{(\mathbf{x},\sigma)\in c}\tau_\sigma(\mathbf{x})}{|c|}$. Due to non-determinism in the graph and unknown initial condition, it is uncertain in which cycle we will end up in. Therefore, the worst primitive cycle gives us our guaranteed average IST $\hat{\tau}$ as shown in Eq. (3-1).

For control performance it does not suffice to do the same and take the average σ value for the transitions. This can most easily be demonstrated by a small example. Imagine we have a 2-cycle the controlled scheduler $S_c \times_{\mathcal{F}} S_{\mathcal{Q}/Q}$: with the two nodes $(\mathbf{x}_1, \sigma_1)$ and $(\mathbf{x}_2, \sigma_2)$ such that $\sigma_1 = 0.2$ has a low guaranteed decay rate and $\sigma_2 = 0.8$ guarantees a fast decay rate. Depending on the times each guaranteed decay rate is held, this cycle may guarantee a decay rate close to either σ_1 or σ_2. Therefore, it will be important to use a timed average on the decay rates σ. A detailed derivation with the Common Lyapunov Function (CLF) decay is given in appendix A-4. Again, the guaranteed decay rate of the graph $\hat{\sigma}$ is given by the worst cycle as shown in Eq. (3-2). Notice the worst cycle of $\hat{\tau}$ may be a different from the cycle used to determine $\hat{\sigma}$.

$$\hat{\tau} = \min_{c\in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} H_{\mathcal{Q}/Q}(\mathbf{x},\sigma)}{|c|} = \min_{c\in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \tau_\sigma(\mathbf{x})}{|c|} \tag{3-1}$$

$$\hat{\sigma} = \min_{c\in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma H_{\mathcal{Q}/Q}(\mathbf{x},\sigma)}{\sum\limits_{(\mathbf{x},\sigma)\in c} H_{\mathcal{Q}/Q}(\mathbf{x},\sigma)} = \min_{c\in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma\tau_\sigma(\mathbf{x})}{\sum\limits_{(\mathbf{x},\sigma)\in c} \tau_\sigma(\mathbf{x})} \tag{3-2}$$

3-1-2 Multi-objective optimization problem

The task now is to design a controller S_c for the scheduler to maximize the quantities $\hat{\tau}$ and $\hat{\sigma}$. However, in general, maximizing the IST will result in a decrease of possible guaranteed decay rate. Therefore, the optimization problem has multiple objectives. As shown in Section 1-6, a way to describe if a solution is 'optimal' is by the introducing Pareto efficient solutions. One could try every possible controller S_c and create the discrete Pareto frontier as shown below in Figure 3-1. However, trying every possible controller is a tedious task, even impossible as the size of the abstraction or system increases. That is why we need a smart way to compute the optimal controllers. How we attempted to do this, we will discuss in the next section.

3-2 Cost Function Generation

A common tactic to estimate the Pareto front is to, instead of trying to optimize two contradicting quantities, create a new optimization objective, which is a linear combination of the two desired quantities. This method is called scalarization as described in Section 1-6.

However, before we can create such a thing, for neediness we scale both $\hat{\tau}$ and $\hat{\sigma}$ such that one does not overwhelm the other by default. Both quantities lay within a known closed and

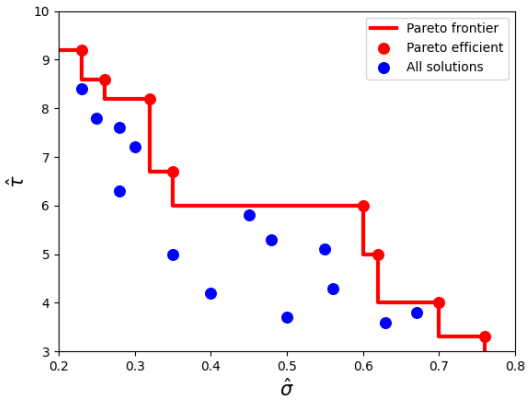


Figure 3-1: The Pareto frontier, visualized by the 'red' line, encloses the area where the Pareto efficient solutions are optimal. No solutions exists above or to the right of this line, given the Pareto efficient solutions are complete.

confined range. Therefore, both can be scaled such that both ranges map to the interval of $[0, 1]$ as shown in Eq. (3-3).

$$\begin{aligned} \hat{\tau}' &= \frac{\hat{\tau} - \tau_{min}}{\tau_{max} - \tau_{min}} \\ \hat{\sigma}' &= \frac{\hat{\sigma} - \sigma_{min}}{\sigma_{max} - \sigma_{min}} \end{aligned} \tag{3-3}$$

The optimization objective will take the form of maximizing $\lambda\hat{\tau}' + (1-\lambda)\hat{\sigma}'$ for different values of $\lambda \in [0, 1]$. By varying λ we hope to find different Pareto efficient solution and get a better and better estimation of the Pareto frontier. Recall from Section 1-6 that the scalarization method only guarantees to find all Pareto efficient solutions if the solution space is convex, which is not guaranteed in our case.

3-2-1 Fundamental Problem

However, a problem arises. Recall we do not want to try every controller S_c separately but want to use some algorithm that can, with some efficiency, calculate the Pareto efficient solutions. Due to the fundamental difference in $\hat{\tau}$ and $\hat{\sigma}$. Namely, $\hat{\tau}$ being the average and $\hat{\sigma}$ being the timed average.

Regarding the algorithms that can efficiently solve the two problems:

- Maximize the minimum cycle mean in a graph, to calculate $\hat{\tau}$
- Maximize the minimum weight to time cycle in a graph, to calculate $\hat{\sigma}$

Both fundamentally work differently due to the difference in the nature of the problems as shown in Section 1-4 and Section 1-5. When wanting to calculate these quantities, one has

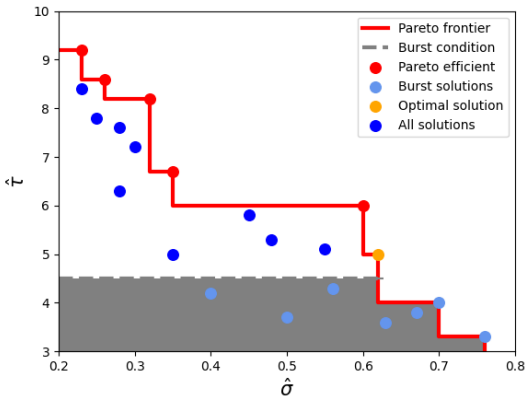


Figure 3-2: The Pareto frontier. By setting an extra condition: a minimum value for $\hat{\tau}$, marked with the color 'gray', our problem becomes a single optimization problem since we are only maximizing $\hat{\sigma}$

to assign weights to every edge in the graph. For both problems, the weight assigning is different. Therefore, it is fundamentally impossible to assign a combined weight to the edges in order to maximize the minimum value for $\lambda\hat{\tau}' + (1-\lambda)\hat{\sigma}'$. For more details on the weight assessment of maximizing the minimum weight to time cycle in a graph, see Section 4-5. In contrast to maximizing the minimum cycle mean in a graph, which just takes the quantity of interest as edge weights.

3-3 Key Findings

While the combined cost function method was not successful, we did learn a lot about different algorithms, our optimization objectives and how one can shift perspective when tackling a problem. It became clear we need a different approach and we debated on what would be a useful tool for a control engineer to design a scheduler. Instead of estimating the entire Pareto front, would it not be more useful to get more grip on a minimal average IST $\hat{\tau}$ or control performance $\hat{\sigma}$? We have chosen to further investigate the case where we set a minimal average IST $\hat{\tau}$, and implement a slightly stricter practical condition: design for optimal control performance $\hat{\sigma}$ while avoiding IST bursts in the system.

Effectively reducing the multi-objective optimization problem to a single objective. Regarding the Pareto efficient solutions, this can be visualized by Figure 3-2. We will keep the definitions of $\hat{\tau}$ and $\hat{\sigma}$ as they were motivated in this chapter, given in Eq. (3-1).

Chapter 4

Controller Design

In this chapter, the final controllers are designed. In chapter 2, we first introduced the Linear Time-Invariant (LTI) system S which represents the physical system. The system S can be controlled in a Periodic Event-Triggered Control (PETC) manner by choosing a quadratic triggering conditions $\mathcal{Q}_\sigma \in \{\mathcal{Q}_{\sigma_1}, \mathcal{Q}_{\sigma_2}, \dots, \mathcal{Q}_{\sigma_s}\}$ and having a fixed stabilizing static feedback gain K. Each triggering condition $\mathcal{Q}_\sigma$ guaranteeing a decay rate of $\kappa\sigma$ and when implemented, rendering the system autonomous $S_{\mathcal{Q}_\sigma}$.

We introduced the scheduling problem, which is capable of switching between different triggering conditions $S_\mathcal{Q}$ after each triggering event. A quotient model with increasing behavioral precision was created by introducing the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ by an equivalence relation Q_l, which construct a finite state system while behaviorally includes the switched system $S_\mathcal{Q}$.

In the previous chapter we described the two quantities of interest, $\hat{\tau}$ and $\hat{\sigma}$, when designing a controller S_c on $S_{\mathcal{Q}/Q_l}$. Using the Pareto frontier, we motivated how we can change our approach to the problem.

In this chapter, we will first clearly define our burst constraint and how this in relation to $\hat{\tau}$. Using the burst constraint, we will define a safety set W which will only contain safe behaviors. The safety set is used to construct the Maximal Permissive Controller (MPC) S_{mpc} which should limit the behavior of $S_{mpc} \times_\mathcal{F} S_{\mathcal{Q}/Q_l}$ to the set W.

On the MPC S_{mpc}, we will define a greedy controller S_{gc} which optimizes performance in a greedy manner. Finally, we will introduce our infinite horizon controller S_{ihc}, which solves the infinite horizon problem and produces the optimal control strategy.

4-1 Burst Setup

First we will mathematically define the burst condition and give some concrete examples of what is and what is not considered a burst sequence. Second, the comparison is made between the worst average Inter-Sample Time (IST) cycle in the system $\hat{\tau}$ and the burst

condition. Afterwards, the burst condition is transformed to a safety set, practical for use of the l-complete abstraction.

4-1-1 Defining bursts

Recall, the PETC sampling time is Δ and we will be using the discrete inter-sample time sequences with $k_i = \frac{\tau_i}{\Delta}$, first introduced in Section 1-7-7. Define $\mathcal{K}_N$ as the set of all possible discrete inter-sample time sequences of length N, i.e. $\mathcal{K}_N = \left\{ \{k_i\}_{i=1}^N \mid \forall i : k_i \in \mathbb{N} \right\}$. A burst of length N will be defined by a collection of discrete inter-sample sequences of length N. Define the burst function $b_N : \mathbb{N} \to \mathcal{K}_N$, for some discrete timing k_b and number of events N. Let $b_N(k_b)$ describe all possible discrete IST sequences of N events that trigger within k_b time, defined by $b_N(k_b) = \left\{ \{k_i\}_{i=1}^N \mid \forall i : k_i \in \mathbb{N}, \sum_{i=1}^N k_i \leq k_b \right\}$.

The use of the burst function b_N is best illustrated with a small example. Let $b_3(12)$ be a burst set describing the sequences which would trigger at least 3 events within 12 time, e.g. the following sequences are included $(2, 2, 2) \in b_3(12)$ or $(6, 4, 2) \in b_3(12)$, while sequences $(5, 14, 12) \notin b_3(12)$, $(6, 4, 3) \notin b_3(12)$ or $(2, 2) \notin b_3(12)$ are not included. In order to effectively implement a burst condition, we require the depth of the abstraction to be at least the size of the burst $N \leq l$. Otherwise, it will not be possible to mark states according to the burst condition.

4-1-2 Relation bursts and average IST

A possible problem in Networked Control System (NCS) is that one control system requires many control updates in a small time window in order to keep the system stable, i.e. a burst of communications on the shared network is necessary to guarantee the stability of a single control system. This may be at the expense of other control systems. Moreover, when multiple systems exhibit burst behavior, network congestion can occur, leading to increased latency or even package losses. Therefore, we argue that setting a burst condition b_N as a hard constraint is more useful then implementing a minimum value for the worst average IST cycle $\hat{\tau}$. However, the downside with both methods is how to choose the parameters: which value for $\hat{\tau}$ or N and k for a burst $b_N(k)$.

Despite the differences, there are some similarities. Most noticeable when the size of a burst $b_N(k)$ is of equal length then the depth of the l-complete abstraction, i.e. when one chooses $N = l$ and $k = l\hat{\tau}$. In this case, setting a minimum average IST will mark states of equal length to the burst condition, and thereby both marking the same set of states. Moreover, both methods become effectively equivalent when one chooses the size of a burst to equal to the depth of the abstraction. Recall the purpose of setting a minimum $\hat{\tau}$ from the Pareto frontier perspective from Figure 3-2. Note this is only the case for when $N = l$. If $N < l$, both methods will not necessary be marking the same set of states.

4-1-3 Safety set

Marking burst patterns within a burst set $b_N(k) \in \mathcal{K}_N$ describes the patterns we would like to avoid. Due to having an l-complete abstraction, which not necessarily has the same depth as

the burst sequences, we would like to describe which states are 'safe' and do not foresee a burst in the next l events. These states are described by the safety set, the safety set is a function $W_{N,l} : \mathcal{K}_N \to \mathcal{K}_l$ where the depth of the abstraction l is at least the size of the burst N, i.e. $N \leq l$. The safety set is defined by $W_{N,l}(b) = \left\{\{k_i\}_{i=1}^{l} \mid \exists j \in \mathbb{N}_{\leq l-N+1} : \{k_i\}_j^{N+j-1} \in b\right\}$ where $b \in \mathcal{K}_N$ describes a burst set as first seen in Section 4-1-1.

The next objective will be to design a MPC S_{mpc} which will guarantee that the l-complete abstraction will only exhibit safe behaviors, i.e. Reach $\left(S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q}\right) \subseteq W$, while preserving as much input sequences as possible. Inside this safe behavior, we will design another controller S_c which will use the remaining input sequences to optimize the control performance $\hat{\sigma}$, i.e. the resulting system will have the following structure $S_c \times_{\mathcal{F}'} S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q}$.

4-2 Safety Game

Given a burst condition $b_N(k_b)$ for triggering at least N events within k_b time and a corresponding safety set $W_{N,l}\left(b_N(k_b)\right)$ describing all safe behaviors of lengths l. As described in the previous section, we would like to design a MPC S_{mpc} and an extended alternating simulation relation $F = R^e$ such that $S_{\mathcal{Q}/Q_l}$ is alternatingly simulating S_{mpc} by R, i.e. $S_{mpc} \preceq_{\mathcal{AS}} S_{\mathcal{Q}/Q_l}$. The MPC S_{mpc} is designed such that all possible behaviors from the controlled l-complete abstraction $S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$ are included within the safety set $W_{N,l}(b_N(k_b))$, i.e. to guarantee Reach $\left(S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}\right) \subseteq W$, while preserving as much input sequences as possible.

We construct S_{mpc} by playing a safety game on the arena equivalent of $S_{\mathcal{Q}/Q_l}$ with the safety set $W_{N,l}(b_N(k_b))$. Recall from 1-9-1 how we can transform the internal dynamics of a system to an arena $\mathcal{A} = (V_0, V_1, E)$ to play the safety game on. Recall, when converting $S_{\mathcal{Q}/Q_l}$ to an arena, the arena will have the following properties $V_0 = X_{\mathcal{Q}/Q_l}$, $V_1 = \{(v,u) \mid v \in V_0, u \in U(v)\}$ and $E = E_0 \cup E_1$. Here, $E_0\{(v,u,(v,u)) \mid \forall v \in V_0, u \in U(v)\}$ describes the controllable transitions from V_0 to V_1 and $E_1 = \{(v,u,v') \mid \forall (v,u) \in V_1, v' \in V_0 : (v,u,v') \in E_{\mathcal{Q}/Q_l}\}$ the uncontrollable non-determinism in the transitions.

4-2-1 Defining the maximal permissive controller

The safety game is played as described in 1-9-2 by using the reachability algorithm on the dual problem. After playing the safety game, we obtain the trap set $Trap(\mathcal{A}, W)$ and can construct the complementary set: the limited safety set $W^e = X_{\mathcal{Q}/Q_l} \backslash Trap(\mathcal{A}, W)$. Since W^e is defined on the arena instead of a system, the new states of the controller S_{mpc} are defined by limiting W^e to only V_0, i.e. taking $X_{mpc} = V_0 \cap W^e = X_{\mathcal{Q}/Q_l} \cap W^e$ since $V_0 = X_{\mathcal{Q}/Q_l}$. The transitions of the maximal permissive controller are defined by all transitions for which the systems stays inside these safe states X_{mpc} a complete description of the MPC $S_{mpc} = (X_{mpc}, X_{mpc0}, U_{mpc}, E_{mpc}, Y_{mpc}, H_{mpc})$ is given below:

- $X_{mpc} = X_{\mathcal{Q}/Q_l} \cap W^e$
- $X_{mpc0} = X_{mpc}$
- $U_{mpc} = U_{\mathcal{Q}/Q_l}$
- $E_{mpc} = \{(x,u,x') \in \mathbf{E}_{\mathcal{Q}/Q_l} \mid x, x' \in X_{mpc}\}$

- $Y_{mpc} = Y_{\mathcal{Q}/Q_l}$
- $H_{mpc} = H_{\mathcal{Q}/Q_l}|_{X_{mpc}}$ the function $H_{\mathcal{Q}/Q_l}$ limited to the input set X_{mpc}

Recall the alternative transition set of the l-complete abstraction $\mathbf{E}_{\mathcal{Q}/Q_l}$ introduced in Eq. (2-11). In contrast to $E_{\mathcal{Q}/Q_l}$ which uses the first guaranteed decay rate in the sequence as input, for control purposes, the last guaranteed decay rate in the sequence is needed. An input now representing which decay rate comes next to the existing sequence which is already fixed.

Let us now show there exist an alternating simulation relation $R \subseteq X_{mpc} \times X_{\mathcal{Q}/Q_l}$. By construction of S_{mpc}, the states X_{mpc} are just a subset of $X_{\mathcal{Q}/Q_l}$. Moreover, the transitions follow the requirements of an an alternating simulation relation from Definition (1.23). Therefore, the trivial relation which relates equivalent states $R = \{(x, x) \mid x \in X_{mpc}\}$, is an alternating simulation relation from S_{mpc} to $S_{\mathcal{Q}/Q_l}$, i.e. $S_{mpc} \preceq_{AS} S_{\mathcal{Q}/Q_l}$. Recall Definition (1.23) for the requirements for an alternating simulation relation. This relation R can be used to construct a feedback composition to let S_{mpc} control the system $S_{\mathcal{Q}/Q_l}$ since they are feedback composable. Let $F = R^e$ be the extended alternating simulation relation of R. The feedback composition is then given by $S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$ and describes the safely controlled system.

4-2-2 Reduced initial conditions

Notice, it may be the case that S_{mpc} has less (initial) states then the system $S_{\mathcal{Q}/Q_l}$. Meaning, S_{mpc} is only a valid controller for $S_{\mathcal{Q}/Q_l}$ if we initialize $S_{\mathcal{Q}/Q_l}$ in a state that relates to S_{mpc} by R. If this is not the case, no valid control input can be generated. Since S_{mpc} is the maximal permissive controller, if the system is initialized outside X_{mpc}, e.g. $(x_0, \sigma_0) \in X_{\mathcal{Q}/Q_l} \backslash X_{mpc}$, there is no combination of future triggering conditions for which a burst can be avoided staring from x_0 and first applying σ_0.

This can be problematic if there is a $x_0 \in X$ in the original system S for which there is no triggering condition $\sigma \in \Sigma$ such that (x_0, σ) can be controlled by S_{mpc}, i.e. there exists a $x_0 \in X$ such that $\forall \sigma \in \Sigma$ it holds that $(x_0, \sigma) \in X_{\mathcal{Q}/Q_l} \backslash X_{mpc}$. There are three lines of reasoning when this is the case. Firstly, one could ignore these possible initial states $x_0 \in X$ since the system is guaranteed to exhibit burst behavior no matter the input sequence of possible $\sigma \in \Sigma$. Secondly, one could include and extra triggering condition σ' which is very conservative and has a slow decay rate of the Lyapunov function. However, this does not guarantee that no bursts occur, only make it more unlikely. Thirdly, the burst condition may be chosen too hash and needs to revised for the system S.

4-2-3 Properties of the maximal permissive controller

Notice, unlike $S_{\mathcal{Q}/Q_l}$, the system S_{mpc} will not necessarily be strongly connected due to a possible reduction of transitions. Since this is requirement for some algorithms, we will have to make smart adjustments in the future. However, do note that S_{mpc} is still non-blocking. This can be reasoned by the following contradiction: if S_{mpc} is blocking, there is a state $x_0 \in X_{mpc}$ such that $U(x_0) = \varnothing$. However, by construction of the reachability algorithm shown in appendix B-5, every state that has no possible output such that it stays outside the trap set, is included in the trap set. Since X_{mpc} includes all states not included in the trap set, we have $x_0 \notin X_{mpc}$, concluding our contradiction.

On the MPC S_{mpc}, we will implement a controller which optimizes control performance, i.e. maximizes $\hat{\sigma}$. First we will show a greedy approach to constructing a controller S_{gc}, which fails in optimizing $\hat{\sigma}$, but does give decent results. Thereafter, we will give the infinite horizon controller S_{ihc}, which does maximizes $\hat{\sigma}$, but requires much more computing power.

4-3 Greedy Optimizer

Given the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ of the switched system $S_{\mathcal{Q}}$ describing the scheduling problem, the Maximal Permissive Controller (MPC) S_{mpc} and the alternating simulation relation R_{gc} between them. We would like to design a new controller on S_{mpc} which optimize for control performance. Since S_{mpc} is maximally permissive, at a given state $x \in X_{mpc}$ there may still be multiple valid inputs, i.e. $|U_{mpc}(x)| > 1$.

A greedy optimizer can be used as a low computational method to construct a sub-optimal greedy controller S_{gc}, by taking the optimal decay rates as inputs locally at every state. Therefore, the greedy controller S_{gc} has the same states and output mapping as the MPC S_{mpc}, only limiting the possible transitions to the maximum valid guaranteed decay rate at every state. The greedy controller $S_{gc} = (X_{gc}, X_{gc0}, U_{gc}, E_{gc}, Y_{gc}, H_{gc})$ is given by:

- $X_{gc} = X_{mpc}$
- $X_{gc0} = X_{gc}$
- $U_{gc} = U_{mpc}$
- $E_{gc} = \{(x, u, x') \in E_{mpc} \mid u = \max(U_{mpc}(x))\}$
- $Y_{gc} = Y_{mpc}$
- $H_{gc} = H_{mpc}$

Recall the overloaded notation of $U : X \to U$ throughout this thesis, mapping each state to the set of possible inputs at the given state, e.g. given a state $x \in X_{mpc}$, $U_{mpc}(x) = \{u \mid \exists x' \in X_{mpc} : (x, u, x') \in E_{mpc}\}$. Since the states in X_{gc} use the same as S_{mpc}, let us try the trivial relation $R_{gc} = \{(x, x) \mid x \in X_{gc}\}$. Due to the nature of the transitions generation of E_{gc}, indeed R_{gc} is an alternating simulation relation from S_{gc} to S_{mpc} by definition, given in Definition (1.23). Since $S_{gc} \preceq_{\mathcal{AS}} S_{mpc}$, the system is feedback composable. Define the extended simulation relation $F_{gc} = R^e_{gc}$ as interconnection relation according to Definition (1.25). The greedy controller can then control the MPC controller as follows: $S_{gc} \times_{\mathcal{F}_{gc}} S_{mpc}$. This can be applied as before, as shown in Section 4-2-1, as a controller in itself to control the l-complete abstraction by the extended simulation relation F. We obtain $S_{gc} \times_{\mathcal{F}_{gc}} S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$ as the controlled l-complete abstraction of $S_{\mathcal{Q}}$. Recall from Section 2-4-1 the process of synthesizing the controlled l-complete abstraction and the switched system $S_{\mathcal{Q}}$.

4-3-1 Sub-optimality

The sub-optimality of S_{gc} comes from the greedy optimizing approach of only optimizing the control performance of a single step in the system, instead of considering the possible infinite paths. As discussed before in 1-2 the infinite paths of a finite system are captured within the primitive cycles in the system due to states not in a primitive cycle, are visited at most

once in an infinite path. How to efficiently calculate the guaranteed control performance is shown later in Section 4-5. That the greedy optimizer does not necessarily find the optimal controller for control performance, is best visualized in a small example.

In Figure 4-1, a small part of an arena $\mathcal{A} = (V_0, V_1, E)$ is given, complemented with details from S_{mpc} as guaranteed decay rate sequences, IST sequences and inputs corresponding to the controllable transitions. Only considering this part of the arena, one can see that the greedy optimizer does not find the optimal control strategy to optimize control performance.

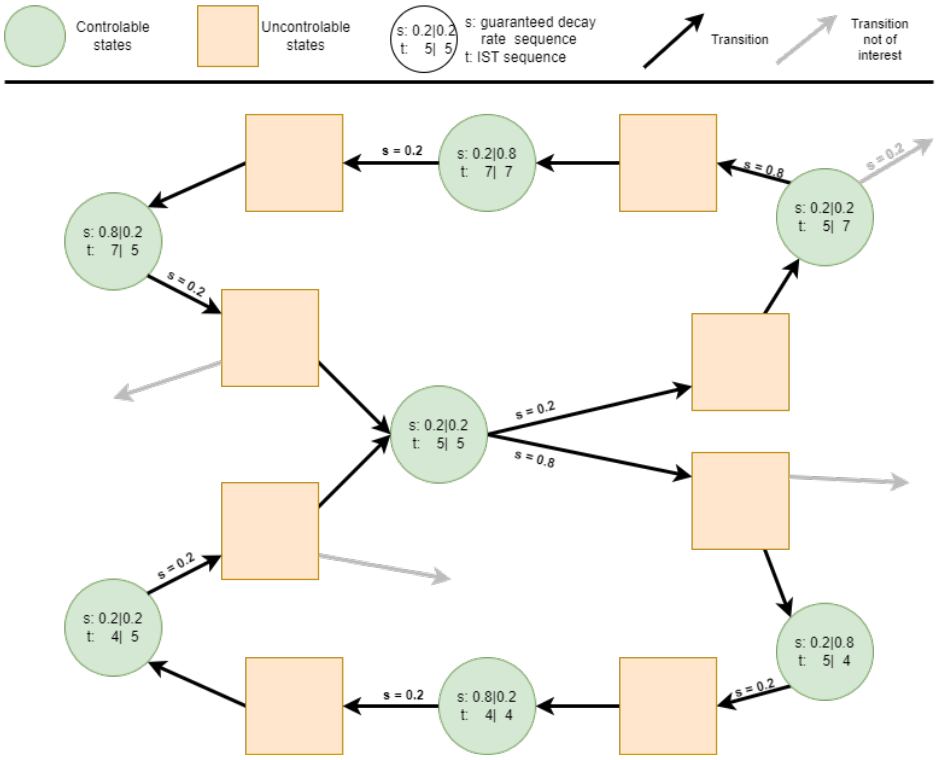


Figure 4-1: A sketch of a situation when the greedy controller does not find the optimal strategy for optimizing control performance in an arena.

The example given in Figure 4-1 is generated using an l-complete abstraction of depth $l = 2$ and using some burst condition to construct S_{mpc}. For this small part of the arena, one can see that there is only a single controllable state for which a decision needs to be made, the middle state. Naturally, the greedy controller S_{gc} will choose the highest guaranteed decay rate as it only looks at the local state to make a decision, i.e. $\sigma = 0.8$ (equivalent to $s = 0.8$ in the figure). However, this will result in a cycle performance of:

$$\frac{\sum_{e \in c} \sigma_e \tau_e}{\sum_{e \in c} \tau_e} = \frac{0.2 \cdot 5 + 0.2 \cdot 5 + 0.8 \cdot 4 + 0.2 \cdot 4}{5 + 5 + 4 + 4} = \frac{1}{3}$$

However, when one chooses locally the worst option of $\sigma = 0.2$ ($s = 0.2$) at the middle controllable node, the resulting cycle performance will increase as seen below:

$$\frac{\sum_{e \in c} \sigma_e \tau_e}{\sum_{e \in c} \tau_e} = \frac{0.2 \cdot 5 + 0.2 \cdot 5 + 0.2 \cdot 7 + 0.8 \cdot 7}{5 + 5 + 7 + 7} = \frac{3}{8} > \frac{1}{3}$$

Naturally, these numbers are chosen arbitrary to represent realistic values. In theory, one could enhance these numbers to artificially increase the gap the two obtained performances up to the difference in maximum and minimum decay rate of the Common Lyapunov Function (CLF), i.e. $\sigma = 0.2$ and $\sigma = 0.8$ in this example.

Our last goal will be to design a controller S_{ihc} which solves the infinite horizon problem of maximizing the worst case control performance primitive cycle. This is shown in the next section.

4-4 Optimal Control Performance

Instead of using a greedy optimizer to obtain a controller S_{gc} on the MPC S_{mpc}. We would like solve the infinite horizon problem and produce the controller S_{ihc} which optimizes the control performance $\hat{\sigma}$ of $S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$, first introduced in Eq. (3-2), but also shown below in Eq. (4-1). Likewise to the construction of S_{gc}, the infinite horizon controller S_{ihc} will be equivalent to S_{mpc} except it will limit the possible transitions.

$$\hat{\sigma} = \min_{c \in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma H_{\mathcal{Q}/Q}(\mathbf{x},\sigma)}{\sum\limits_{(\mathbf{x},\sigma)\in c} H_{\mathcal{Q}/Q}(\mathbf{x},\sigma)} = \min_{c \in C_G} \frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma \tau_{\sigma}(\mathbf{x})}{\sum\limits_{(\mathbf{x},\sigma)\in c} \tau_{\sigma}(\mathbf{x})} \tag{4-1}$$

The next sections are describing the two different problems we still have to solve. First we will discuss, given a controller on S_{mpc}, how one can verify the associated control performance $\hat{\sigma}$ efficiently. Here, we will give the details of how to apply which algorithms. The method used will be a neat starting point for the second problem. In the second problem we will describe how one can generate the optimal strategy S_{ihc} as controller on S_{mpc} to solve the infinite horizon problem.

4-5 Verifying Control Performance

Given a strategy/controller on S_{mpc}, one can verify the control performance by calculating $\hat{\sigma}$. In this section we will show how one can do this more efficiently then determining all primitive cycles, construct C_G and calculating the fraction from Eq. (4-1) for every cycle $c \in C_G$. As one can imagine, this is a tedious, time consuming task. Therefore, the problem we would like to solve is: how to efficiently calculate the minimum weight to time cycle in a graph.

Luckily, this is a well studied problem as described in the preliminary knowledge of Section 1-5. There are multiple choices for different algorithms. Since Lawler's binary search approach will be beneficial later on as well, we have chosen to implement this method, complemented with either Floyd-Warshall (FW) (see appendix B-3) or Bellman-Ford (BF) (see appendix B-2).

Unfortunately, BF can only be used on strongly connected graphs. Recall, the MPC S_{mpc} is connected and non-blocking, but not necessarily strongly connected. Therefore, we need some additional adjustments before we can use BF. The requirement of the graph being strongly connected, does not apply to the FW algorithm.

4-5-1 Lawler's method

The key insight why Lawler's method [43] is able to efficiently calculate $\hat{\sigma}$ in our case, is because we have tight bounds on the possible values of $\hat{\sigma}$. Since $\hat{\sigma}$ is the time average of all possible decay rate $\sigma \in \Sigma$, we know it must be in the closed interval $\hat{\sigma} \in [\min(\Sigma), \max(\Sigma)]$ between the minimum and maximum decay rates. For clarity, recall from Section 2-1 we took out the maximum Lyapunov decay rate κ and for all $\sigma \in \Sigma$ it is just a scalar $\sigma \in (0, 1]$.

Recall the definition of $\hat{\sigma}$ from Eq. (4-1), of being a minimum weight ($\sigma\tau_\sigma(\mathbf{x})$) to time ($\tau_\sigma(\mathbf{x})$) cycle problem. Mathematically, one can construct a weighted timed graph $G_1 = (V_1, E_1, w_1, t_1)$, with a weight function $w_1 : E \to \mathbb{R}$ and a timing function $t_1 : E \to \mathbb{R}_{\geq 0}$, all defined by:

- $V_1 = X_{mpc}$
- $E_1 = E_{mpc}$
- $t_1(e) = t_1(v, v') = H_{mpc}(v)$
- $w_1(e) = w_1(v, v') = U_{mpc}(v)H_{mpc}(v) = \sigma_e\tau_e$

Given a limit range of possible values for the weight to time cycle, Lawler's method consist of taking a guess of the minimum weight to time cycle in the graph and adjusting it in a binary search fashion. Given a guess $\lambda \in [\min(\Sigma), \max(\Sigma)]$ for $\hat{\sigma}$, one can create the weighted graph $G_2^\lambda = (V_2, E_2, w_2^\lambda)$ with equivalent states and edges, only with an adjusted weight function $w_2^\lambda : E \to \mathbb{R}$, given by:

- $V_2 = V_1$
- $E_2 = E_1$
- $w_2^\lambda(e) = w_2^\lambda(v, v') = U_{mpc}(v)H_{mpc}(v) - \lambda H_{mpc}(v) = \sigma_e\tau_e - \lambda\tau_e$

The interesting thing about G_2^λ, is that we translated the minimum weight to time cycle problem in G_1 to a minimum mean cycle problem. If we find a cycle with a mean cycle equal to zero (same reasoning can be applied to smaller or bigger then zero), then λ equals the fraction of interest with respect to $\hat{\sigma}$ as shown in Eq. (4-2). If the cycle mean of every cycle in G_2^λ is bigger then 0, i.e. there are no negative cycles, then initial guess was too low $\lambda < \hat{\sigma}$. On the contrary, if there is a negative mean cycle in G_2^λ, our initial guess was too high $\lambda > \hat{\sigma}$. Therefore, we only need a negative cycle detection algorithm, like BF or FW, and can with a binary search approximate $\hat{\sigma}$ as precisely as needed.

$$\begin{aligned}
\frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma\tau_\sigma(\mathbf{x}) - \lambda\tau_\sigma(\mathbf{x})}{|c|} &= 0 \\
\sum_{(\mathbf{x},\sigma)\in c} \sigma\tau_\sigma(\mathbf{x}) - \lambda\tau_\sigma(\mathbf{x}) &= 0 \\
\sum_{(\mathbf{x},\sigma)\in c} \sigma\tau_\sigma(\mathbf{x}) &= \lambda \sum_{(\mathbf{x},\sigma)\in c} \tau_\sigma(\mathbf{x}) \\
\frac{\sum\limits_{(\mathbf{x},\sigma)\in c} \sigma\tau_\sigma(\mathbf{x})}{\sum\limits_{(\mathbf{x},\sigma)\in c} \tau_\sigma(\mathbf{x})} &= \lambda
\end{aligned} \tag{4-2}$$

Remark. Unfortunately, it is likely there is a mismatch between the fraction and λ, i.e. in Eq. (4-2) the fraction is bigger $>$ or less $<$ then zero instead of equal $=$. So far, this mismatch can not be used (other then the information of being < 0 or > 0) to make a highly educated next guess of λ.

4-5-2 Negative cycle detection algorithm

As stated before, we implemented both BF and FW. Theoretically, BF has a time complexity of $\mathcal{O}(V \cdot E)$ and FW a time complexity of $\mathcal{O}(V^3)$. The benefit of implementing FW is that the graph does not have to be strongly connected and if there is indeed a negative cycle in the graph G_2^λ, the process can prematurely terminate. Despite BF not having both benefits, the time complexity is better then FW in general due to $|V| \leq |E| \leq |V|(|V|-1)$[1] and we will show it is well suited for parallel programming. Both methods have there own benefits, when running simulations, we will comment on the speed comparisons between BF and FW.

Let us comment on how to apply BF in parallel fashion on a graph G_2^λ which is not strongly connected. The key is that the precise criteria for applying BF is not that the graph is strongly connected, but we have a source node $s \in V$ which can reach all other nodes in the graph. By assuming the graph is strongly connected, every node can be a source node. However, for negative cycle detection, it is sufficient to split up the graph in sub-graphs $G_{/i} = (V_{/i}, E_{/i}, w_{/i})$ with related source node $s \in V$, defined by:

- $V_{/i} = \{v \in V_2 \mid \forall j < i : v \notin V_j, v \in \mathit{Reach}(G_2^\lambda, s)\}$
- $E_{/i} = \{(v, v') \in E_2 \mid v, v' \in V_{/i}\}$
- $w_{/i}^\lambda = w_2^\lambda|_{E_{/i}}$

Recall $\mathit{Reach}(G, s)$ being the set of all vertices which can be reached from s. By constructing $G_{/i}$ with an associated s for increasing values of i by choosing $s \notin V_j$ for $j < i$, we obtain a partition of the original graph, i.e. in our case G_2^λ. If a node in a cycle is reachable from a starting node $s \in V$, then all nodes of that cycle are reached from s. Therefore, nodes of cycles always being in the same G_i of the partition. Since all nodes in a $G_{/i}$ are reachable from the starting node, we can apply BF to each one separately. As soon a negative cycle is detected in one of the sub-graphs $G_{/i}$, we conclude there is a negative cycle in G_2^λ and terminate the process.

4-6 Infinite Horizon Controller

Given a strategy/controller on the MPC S_{mpc}, one can verify the control performance $\hat{\sigma}$ as shown in the previous section. The question now is, how to design a controller on the MPC S_{mpc} in order to obtain the maximum control performance.

[1] Since G_2^λ is non-blocking, we know $|V| \leq |E|$. Furthermore, if every vertex is connected to every other vertex, the maximum number of edges is reached and we have $|E| = |V|(|V|-1)$. Note, if a strategy is applied such that every vertex has exactly one outgoing edge, then $|E| = |V|$.

4-6-1 Use of mean-payoff games

As demonstrated in Section 4-5-1, Lawler's method can be used to construct a weighted graph $G_2^\lambda = (V_2, E_2, w_2^\lambda)$, where λ is an improving guess for the value of $\hat{\sigma}$. Recall the motivation for the weight adjustment from Eq. (4-2), showing that if the minimum cycle is equal to zero (/lower/higher) then λ equals (/is bigger then/is lower then) $\hat{\sigma}$. Therefore, we can translate our problem to the following: given a guess of λ, does there exists a strategy on G_2^λ such that there are no negative cycle. If no such strategy exists, we will say G_2^λ has an *unavoidable negative cycle*.

Luckily, this is a problem for which we can solve somewhat efficiently. The existence of an unavoidable negative cycle in a graph, can be answered by determining if the value ν of a Mean-Payoff Game (MPG) is positive or negative. As shown by Brim et. al. [41] and mentioned in Section 1-9-3, this question can relatively efficiently be solved by playing several energy games in a total time complexity of $\mathcal{O}(E \cdot V \cdot W)$. More details are given in appendix B-4. This can be used in Lawler's method to close down on the real value of $\hat{\sigma}$ in a binary search fashion.

As shown in Section 1-9-3, a MPG is given by $\Gamma = (V_0, V_1, E, w)$, where (V_0, V_1, E) is an arena extended with a weight function $w : E \to \mathbb{R}$. Recall from Section 1-9-1 how one can convert systems or graphs to arena's and vice versa. From the graph G_2^λ, we construct the MPG $\Gamma_{G\lambda} = (V_{G0}, V_{G1}, E_G, w_{G\lambda})$ with:

- $V_{G0} = V_2$
- $V_{G1} = \{(v, v') \mid v, v' \in V_{G0} : (v, v') \in E_2\}$
- $E_G = E_{G0} \cup E_{G1}$
  $E_{G0} = \{(v, (v, v')) \mid v \in V_{G0}, (v, v') \in V_{G1}\}$
  $E_{G1} = \{((v, v'), v') \mid v' \in V_{G0}, (v, v') \in V_{G1}\}$
- $w_{G\lambda}(e) = \begin{cases} w_{G\lambda}(v, (v, v')) & = w_2^\lambda(v, v') \quad \text{if } e \in E_{G0} \\ w_{G\lambda}((v, v'), v') & = w_2^\lambda(v, v') \quad \text{if } e \in E_{G1} \end{cases}$

Notice the extended weight function to the non-deterministic states from V_{G1} in order to keep the average of a cycle consistent with that of G_2^λ. The value of MPG $\Gamma_{G\lambda}$ is denoted by $\nu_{G\lambda}$ and can be calculated by Brim et. al. first algorithm as stated before.

4-6-2 Constructing the strategy

Once the value of λ estimates $\hat{\sigma}$ with a desired precision and there are no unavoidable negative cycles in the MPG $\Gamma_{G\lambda}$ for the given $\lambda < \hat{\sigma}$, i.e. the MPG $\Gamma_{G\lambda}$ has positive value $\nu_{G\lambda} \geq 0$. One can use the second algorithm of Brim et. al. [41], given in appendix B-4, to construct a strategy of non-negative value $\nu_{G\lambda}$ (since there are no unavoidable negative cycles), with a time complexity of $\mathcal{O}\left(E \cdot V^2 \cdot W \cdot (\log V + \log W)\right)$. Notice a strategy is only valid if the value of the game $\nu_{G\lambda}$ is non-negative due to our choice of weights, motivated in Eq. (4-2).

From the second algorithm of Brim et. al., we obtain a strategy function $\eta : V_{G0} \to E_G$ which describes at every controllable state $v \in V_{G0}$ the optimal transition $e \in E_G$ to take in order to obtain the MPG value $\nu_{G\lambda} \geq 0$. Recall, by construction $V_{G0} = X_{mpc}$.

Therefore, one can apply the strategy η to obtain the infinite horizon controller $S_{ihc} = (X_{ihc}, X_{ihc0}, U_{ihc}, E_{ihc}, Y_{ihc}, H_{ihc})$, given by:

- $X_{ihc} = X_{mpc}$
- $X_{ihc0} = X_{mpc0}$
- $U_{ihc} = U_{mpc}$
- $E_{ihc} = \{(x, x') \in E_{mpc} \mid x, x' \in X_{mpc}, (x, (x, x')) \in eta(x)\}$
- $Y_{ihc} = Y_{mpc}$
- $H_{ihc} = H_{mpc}$

4-6-3 Relating the controller

Likewise to the procedure of the greedy optimizer from 4-3, we will take the trivial relation $R_{ihc} = \{(x, x) \mid x \in X_{mpc}\}$. By construction of S_{ihc}, it is easy to see that R_{ihc} is an alternating simulation relation from S_{ihc} to S_{mpc}, i.e. $S_{ihc} \preceq_{\mathcal{AS}} S_{mpc}$, since S_{ihc} only limits the amount of transitions in S_{mpc}. Let $F_{ihc} = R^e_{ihc}$ be the extended alternating simulation relation as an interconnection relation between S_{ihc} and S_{mpc}. Then S_{ihc} is feedback composable to S_{mpc} and the controlled system will be denoted with $S_{ihc} \times_{\mathcal{F}_{ihc}} S_{mpc}$.

Recall MPC S_{mpc} is a controller for the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ by a relation R and $F = R^e$. Since S_{mpc} was the maximal permissive controller, as long as the system is initialized inside S_{mpc}, all behaviors in $S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$ are rendered safe, i.e. without bursts. On these safe behaviors, S_{ihc} optimizes the control performance and producing our final control strategy on the l-complete abstraction: $S_{ihc} \times_{\mathcal{F}_{ihc}} S_{mpc} \times_{\mathcal{F}} S_{\mathcal{Q}/Q_l}$.

For the connection between the controlled l-complete abstraction $S_{\mathcal{Q}/Q_l}$ and the physical system S, we refer back to the process of synthesis in 2-4-1.

4-6-4 Process speed up

In order to speed up the process. One can first generate the greedy controller S_{gc} and determines the associated control performance $\hat{\sigma}_{gc}$. Generating S_{gc} is computationally cheap. Given a controller, determining $\hat{\sigma}_{gc}$ can be done efficiently as demonstrated in Section 4-5. This value of $\hat{\sigma}_{gc}$ can be used as a new lower bound for the starting guess λ of $\hat{\sigma}_{ihc}$ in the process of Section 4-6-1. Moreover, one can check if there even exists a improved strategy over S_{gc}, or S_{gc} is already the best possible strategy.

Start with the guess $\lambda = \hat{\sigma}_{gc}$ for $\hat{\sigma}_{ihc}$. If the value of the associated MPG is exactly $\nu = 0$, we know there is no improved strategy. Adjusting the weights by increasing λ, would result in a loss of value $\nu < 0$ and there would no valid strategy be for guaranteeing this increased λ. Therefore, in this case $\hat{\sigma}_{ihc} = \hat{\sigma}_{gc}$ and the greedy controller S_{gc} has optimal control performance.

Chapter 5

Simulations and results

In this chapter, the general outline of the simulations is given and the results are discussed. All the plots can be found in appendix C, together with some general info about the simulations and a link to the gitlab, here one can find the code that is used to generate the simulations.

5-1 Simulation Setup

For comparison reasons, we will use the same system first introduced by Tabuada [6] since this is widely used as shown in Eq. (5-1). The system is stabilized with a static state-feedback gain $K = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$ as shown in Eq. (5-2).

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{5-1}$$

$$u(t) = \begin{bmatrix} 1 \\ -4 \end{bmatrix} \mathbf{x}(t) \tag{5-2}$$

Next to the system, we also inherit from Tabuada the Lyapunov function $V = \mathbf{x}^\top P \mathbf{x}$ and the corresponding Q from the continuous Lyapunov equation $(A+BK)^\top P + P(A+BK) = -Q$, where P and Q are given in Eq. (5-3).

$$P = \begin{bmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{2} \end{bmatrix} \tag{5-3}$$

The sampling rate of the system is chosen to be $\Delta = 0.05$ and describes the rate at which the triggering condition is periodically checked.

The triggering condition chosen is the monotone Lyapunov triggering condition for variable guaranteed convergence rate $\sigma \in (0, 1]$ as expressed in Eq. (5-4) and first introduced in Section 1-1-5.

$$\mathcal{Q}_\sigma = \begin{bmatrix} (\sigma - 1)Q - 2PBK & PBK \\ (PBK)^\top & \mathbf{0} \end{bmatrix} \tag{5-4}$$

Our Lyapunov function will be $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$. Determining the maximum decay rate of the Periodic Event-Triggered Control (PETC) methods following the Linear Matrix Inequality (LMI) optimization from appendix A-1, gives a value of $\kappa = 0.48$. For an initial condition of the system I ended up settling for: $\mathbf{x}(0) = [4.574, -8.892]^\top$, which may seem oddly specific, but I'll support my choice more in Section 5-1-1. All simulations are run for 10 time t. In this time, the Lyapunov and state responses all decayed to approximately the origin, but it gives some time to show the Inter-Sample Time (IST) behaviors.

5-1-1 Interesting initial state

As stated before, the choice of initial condition is oddly specific. The initial condition has to suffice a two criteria to be interesting:

- For every simulation, we want $\mathbf{x}(0)$ to be in the safety set, specified by the corresponding S_{mpc}
- For every simulation, we want the quotient state in the corresponding l-complete abstraction $S_{\mathcal{Q}/Q_l}$ of $\mathbf{x}(0)$ to not exhibit only maximum decay of σ_{max} and maximum IST of $\tau_{max} = k_{max}\Delta$ for the first l triggering instances.

Notice, mainly due to the weak burst condition from the first simulation in Figure C-5 in combination with the relatively low k_{max}, this system configuration is critical and we use this to set $\mathbf{x}(0)$. Notice that the second criteria for interesting starting states, in combination with this simulation, is very limiting for possible starting positions. Therefore, we set the second criteria in order to show interesting controller behavior. Namely, avoid the possible self-loop in the system for maximum discrete IST at maximum control control performance.

In order to find possible starting states, $\mathbf{x}_\alpha(0) = [\sin(\alpha) \ \cos(\alpha)]^\top$ for values of α in an interval $\alpha \in [0, \pi)$. In this interval is split equally in 5000 values for α, and non-trivial solutions are extruded. Roughly speaking, this gives a range $\mathbf{x}_{\alpha,0}(0) \in [0.435, 0.475]$ and $\mathbf{x}_{\alpha,1}(0) \in [-0.9, -0.88]$ around an angle of $\alpha \approx \frac{5\pi}{6}$. We choose one of these starting conditions at random to be $\mathbf{x}(0) = [4.5743078, -8.8924523]^\top$, corresponding to $\alpha = \frac{4255\pi}{5000}$. Notice this interval is very small due to the low k_{max} of the simulation.

Notice, as long as the initial condition is in the corresponding Maximal Permissive Controller (MPC) S_{mpc}, it is a valid starting condition for the simulations. For the other simulations, which use a larger k_{max}, we could have generated an different $\mathbf{x}(0)$ in similar manner. However, we decided for comparison reasons to keep $\mathbf{x}(0)$ constant among the different simulations.

5-2 Comparing Simulations

First of all, for the simulations see Figure C-5, Figure C-6 and Figure C-7 in appendix C. Furthermore, of the first simulation, the quotient state progression is given in Table **??**. The controller S_{ihc} is able to schedule the PETC triggering conditions to obtain a guaranteed exponential decay of the Lyapunov function in between the decay rates of the original triggering conditions. All while being on the edge of inducing a burst to occur. Notice the difference in burst condition for each simulation, therefore the allowed IST patterns are also different. The controller S_{ihc} can guarantee a proper decay rate, even for abstractions of low depth and a minimum variety in IST, and shows a better performance than I would have thought.

Regarding an abstraction of depth $l = 2$, the controller sometimes runs into the problem of a safe state in S_{mpc} having an undesirable self-loop (with minimum control performance) regardless of the input of S_{ihc}, and therefore can not guarantee any decay rate of V other then the minimum control performance. By increasing the depth of the abstraction. The amount of self-loops in the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ gets reduced to only the states with maximum control performance, and the problem is mitigated. Therefore, for other problems, I recommend to verify the self-loop behavior of the quotient states in the l-complete abstraction by increasing the depth l.

In general, the difference in maximum IST is clearly visible in the simulations. The resulting state-space responses all converge to the origin and the Lyapunov decay shows clearly the triggering times and the moments there is a switch between triggering conditions.

Regarding the first two simulations, one of depth $l = 3$ in Figure C-5 and one of depth $l = 4$ in Figure C-6. Due to applying a more aggressive burst condition in the second simulation, the obtained guaranteed exponential decay is slightly lower then the first simulation. However, due to the increment in depth of the l-complete abstraction and adding more details by increasing k_{max} from 8 to 15, the controller is able to exhibit more interesting behaviors. Resulting in only a slightly lower guaranteed control performance.

Regarding the second an third simulations, one of depth $l = 4$ and a maximum discrete IST $k_{max} = 15$ in Figure C-6 and one of depth $l = 3$, $k_{max} = 11$ and an additional triggering condition. Both simulations now have the same minimum Average Inter Sample Time (AIST) condition, only for different depths of the abstraction. The third controller, of only depth $l = 3$, with about half the amount of states, outperforms the second controller, due to having a different set of triggering conditions available. Showing that control in triggering conditions is just as, or even more, useful than increasing the depth of the abstraction.

5-2-1 Difference greedy and optimal controller

in appendix C, a few simulations are given. As one can see, these are only simulations for the infinite horizon controller S_{ihc} and not for the greedy controller S_{gc}. Unfortunately, I did not find a difference in performance between them. Therefore, the infinite horizon controller finds the same strategy as the greedy optimizer and has the exact same control performance. My suspicion is that the infinite horizon controller S_{ihc} can still outperform the greedy controller S_{gc}, but only when the abstraction $S_{\mathcal{Q}/Q_l}$ can be constructed of sufficient depth. As one can see in appendix C, the size of the abstraction grows very fast as the depth of the abstraction

increases. Even more so then previous Event-Triggered Control (ETC) methods due to the interconnections between the different triggering conditions, but we will elaborate more about that in Section 5-3.

It could also be the case that I have been unlucky with the choice of initial conditions and other settings of the simulations. However, I have tried many more configurations. The initial condition is chosen to be in S_{mpc} for all burst conditions in the simulations, but also that even the small controller (with a maximum discrete IST of $k_{max} = 8$) exhibits interesting behavior.

For example, regarding the simulation of Figure C-5 in appendix C. Due to the quotient state associated with $(0.8, 0.8, 0.8|8, 8, 8)$ taking up most of the state space, the controller does not have much control in steering the system. Moreover, interesting initial conditions can only be taken from a small set (small slice of physical state-space states), such that it takes at least 3 IST's before possibly getting in $(0.8, 0.8, 0.8|8, 8, 8)$. More details are given in Section 5-1-1.

Another possibility, the greedy controller may always find the strategy to the infinite horizon problem. Recall from Section 4-3 the counter example shown in Figure 4-1. Maybe one could show that with how the abstraction is created, this situation can not occur. However, I doubt it. Due to the construction of the situation in Figure 4-1, the system could diverge to different states with independent behaviors. Naturally, the system dynamics are more intertwined than the abstraction shows at first sight.

5-2-2 Scheduling performance

To answer the question: why even use scheduling and not just apply a triggering condition which just avoids the burst condition. In Table 5-1, minimum occurring discrete IST's are given for the first guaranteed decay rate they do occur. These values are found by constructing the individual traffic models for 1000 different σ values using the same system and triggering conditions throughout this chapter.

For example, if one would like to have a guaranteed minimum AIST of $\hat{\tau} > 4$, one must choose the control performance $\sigma < 0.443$. While using a simple scheduling simulation shown in Figure C-5, constructed by using a minimum k_{max} and depth l, one can obtain a control performance of $\sigma = 0.545$.

This result gets even more extreme when inducing a minimum AIST of $\hat{\tau} > 5$, where normally one can only choose control performances $\sigma < 0.074$. However with scheduling, as shown in Figure C-6 and Figure C-7, we can still obtain a control performance of $\sigma = 0.508$ or $\sigma = 0.531$ respectively.

Remark that these assumptions are made by the existence of a k_{min} in the traffic model, i.e. the l-complete abstraction of base depth $l = 1$. Here, there exists transitions between all regions. Of increasing depths l, these performances without scheduling may lay higher when no loops exists of guaranteed AIST $\hat{\tau} = k_{min}$. However, this topic exceeds the aim of this thesis.

Minimum decay rates	
σ_{min}	k_{min}
0.001	6
0.074	5
0.443	4
0.673	3
0.824	2

Table 5-1: Shown are the minimum decay rates to exhibit behaviors of decreasing values of k_{min}. Tested for values of σ for 3 decimals.

5-3 Growth of the abstraction

As shown in the two tables: Table C-1 and Table C-2, the size of the abstraction increases drastically for increasing values of depth l, maximum discrete IST k_{max} or number of triggering conditions $|\Sigma|$.

For comparison reasons, I also visualized the ratio between transitions and regions. If this ratio equals the number of triggering conditions $|\Sigma|$ of the l-complete abstraction $S_{\mathcal{Q}/Q_l}$, the abstraction has become deterministic (every quotient state has at least $|\Sigma|$ outgoing transitions). However, notice on the ratio in transitions over regions. It may seem small, and if every state only has that in amount of transitions, there is not much non-determinism in the abstraction. While in reality, the amount of transitions per state is not ideally distributed.

Clarifying, there is a significant increase of transitions when increasing the number triggering conditions. While there is a relative decrease of transitions when increasing the depth of the abstraction. However, when increasing the maximum discrete IST k_{max}, it can both increase or decrease the ratio. Moreover, increasing k_{max} splits the regions with a sequence containing k_{max} into multiple regions. The outgoing transitions of such states gets split over multiple states, decreasing the average ratio of transitions over regions. However, the incoming transitions are only increased, due to states now mapping to more regions. This is best visualized with a sketch in Figure 5-1.

Note that the rapidly increasing size of the abstraction is especially a problem when playing energy and mean-payoff games, since they are computationally most expensive. Recall the time complexity of mean-payoff games to be $\mathcal{O}(|E||V|^2W(\log|V| + \log W))$ of a graph $G = (V, E)$ where W is the maximum number of different weights for outgoing transitions of a state. In our case, namely V and E are crucial for computation efficiency, e.g. doubling the amount of states and transitions leads to approximately 8 times the computation time needed.

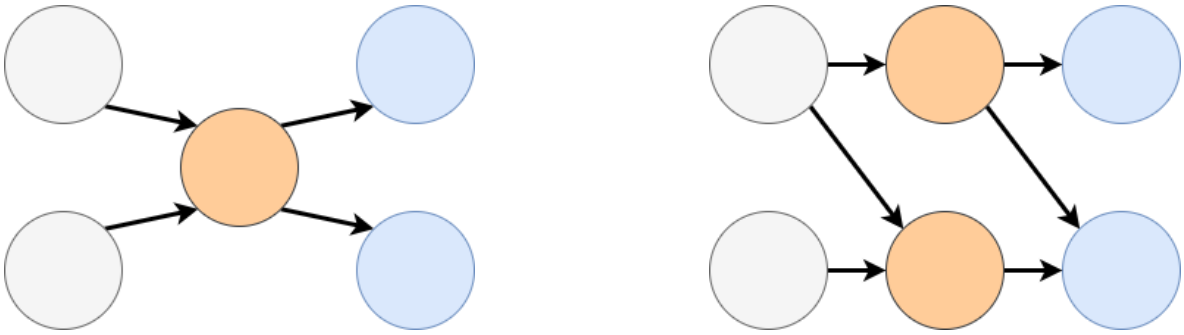


Figure 5-1: A situation is sketched where the orange state has a discrete IST sequence containing k_{max}. In gray are states that have outgoing transitions towards an orange state and in blue are states which have an incoming transition from an orange state. On the Left, the starting situation is visualized. On the right, k_{max} is increased and the orange state is split up in two new states. The average outgoing transitions per orange state is decreased, while the average outgoing transitions per gray state in increased.

Chapter 6

Discussion and Findings

In this thesis project, I have taken a look at the balancing of control performance and Average Inter Sample Time (AIST) of Periodic Event-Triggered Control (PETC) systems by scheduling different triggering conditions at trigger instances. Instead of traditionally setting an required decay rate of the Lyapunov function, we started by setting a burst condition or minimum AIST, and then maximizing for control performance (the decay rate of the Lyapunov function). I have shown that by scheduling, one can obtain a higher control performance then originally possible while avoiding a burst or requiring a minimum AIST.

Unfortunately, due to the size scaling of the l-complete abstraction, I was not able to design controllers for larger depths of l and including more dynamics of the system. Therefore, the controller designed by a greedy optimizer obtains the same strategy as the controller designed by optimizing the infinite horizon problem.

6-1 Future work

Despite already being able to give some results in schedulers that are able to achieve a relative high control performance while avoiding bursts, there are more ideas that could be investigated. My directions for future work are given below:

1. It is still the choice of the engineer to choose the starting set of triggering conditions/decay rates of the Lyapunov function. If this is a free choice, there may exists some optimal selection. Moreover, one may construct a method which could vary the decay rate indefinitely to gain an optimal triggering condition uniquely for each state in the abstraction.
2. In the simulations so far, we only used the same type of (monotone Lyapunov) triggering condition, only with varying the parameter σ. While one could use different types of triggering conditions as long as one could guarantee a fixed Lyapunov decay rate per triggering condition, e.g. the Dynamic Triggering Mechanism (DTM) introduced in Section 1-1-5.

3. The realizations of the l-complete abstraction are not in the minimal representation. One could, computationally almost for free, increase the depth by 1 of the l-complete abstractions. Since the amount of states and edges in the abstraction gets immense quickly, this could save computation time for constructing the scheduler. However, would only save some memory once implemented as a controller.

4. Another way one may reduce the size of the abstraction. Since a state $\mathbf{x}$ in state-space may lay in multiple overlapping regions of the abstraction, it may be possible to remove several regions from the abstraction, without loss of usability. This will be more noticeable when using a larger variety of triggering conditions or when applying a burst condition which only marks very few nodes, since both situations lead to increased overlap of the regions. Remark, this approach may benefit from a shift in perspective for creating the abstractions using the pasts inputs instead of the future inputs for the construction of the quotient states.

5. So far the the greedy strategy has shown the same performance guarantees as the infinite horizon strategy. From a theoretical perspective, the greedy strategy does not guarantee to find the optimal strategy for the infinite horizon problem. That is why the infinite horizon strategy is determined using the computational heavy mean-payoff games. See the debate in Chapter 5. If one could show that the greedy controller does find the optimal strategy, much computation time is saved.

6. The construction of the abstraction could be extended to Event-Triggered Control (ETC) systems by approximating the regions of the abstraction, where each region represents an range of triggering times instead of a single Inter-Sample Time (IST). The latter is only possible due to the use of PETC system.

7. One could extend this framework to allow the varying of triggering conditions in between trigger instances as well. This has some analogy with early triggering in ETC, which would allow more control options for the scheduler.

Appendix A

Proves

Some proves do not fit well inside the main body of the thesis. For some of these proves we can refer to other papers. However, for other proves, and for completeness, we will put them in this appendix.

A-1 Calculating κ

Not necessarily a proof, but a short derivation of κ and how one can calculate κ with respect to the system dynamics $\dot{\mathbf{x}}(t)$ and a Lyapunov function V with corresponding P and Q matrices. Choose $\kappa \in \mathbb{R}_{\geq 0}$ as the maximum value such that $Q - \kappa P$ is still positive semi-definite. Notice that such $\kappa > 0$ must exists since P and Q are both positive definite. Per definition, this implies that $\mathbf{x}^\top (Q - \kappa P)\, \mathbf{x} \geq 0$ holds for all $\mathbf{x} \in \mathbb{R}^n$. Rewriting this gives: $\mathbf{x}^\top Q \mathbf{x} \geq \kappa \mathbf{x}^\top P \mathbf{x}$.

It is given that for the quadratic Lyapunov function $V(\mathbf{x}) = \mathbf{x}^\top P \mathbf{x}$ it holds that $\dot{V}(\mathbf{x}(t)) \leq -\mathbf{x}(t)^\top Q \mathbf{x}(t)$. This can be used to derive at the following:

$$\begin{aligned} \dot{V}(\mathbf{x}(t)) &\leq -\mathbf{x}(t)^\top Q \mathbf{x}(t) \\ &\leq -\kappa \mathbf{x}(t)^\top P \mathbf{x}(t) \\ &= -\kappa V(\mathbf{x}(t)) \end{aligned} \tag{A-1}$$

Solving the differential equation $\dot{V}(\mathbf{x}(t)) = -\kappa V(\mathbf{x}(t))$ given the initial condition of the state $\mathbf{x}(0)$ results in $V(\mathbf{x}(t)) = e^{-\sigma \kappa t} V(\mathbf{x}(0))$. However, recall that $\dot{V}(\mathbf{x}(t)) \leq -\kappa V(\mathbf{x}(t))$ instead of being equal, meaning the decay of V is at least the exponential decay rate of $e^{-\kappa t}$, i.e. $V(\mathbf{x}(t)) \leq V(\mathbf{x}(0)) e^{-\kappa t}$ for all times $t \geq 0$. Rendering κ as the maximum guaranteed decay rate of V.

One can calculate κ efficiently since it is a Linear Matrix Inequality (LMI) of maximizing $\kappa > 0$ such that $Q - \kappa P$ is still positive semi-definite.

A-2 Monotone Lyapunov Triggering Condition

In this section, the derivation of the monotone Lyapunov triggering matrix is given. First we construct an alternative derivation of the time derivative of a function $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$. This result is afterwards used to arrive at the matrix representation of the desired monotone Lyapunov triggering condition.

Theorem A.1 (Time derivative of V)**.** *Given a $Q > 0$ and $P > 0$ such that $(A+BK)^\top P + P(A+BK) + Q = 0$. For $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$ we can proof that $\frac{dV(\mathbf{x}(t))}{dt} = -\mathbf{x}^\top Q\mathbf{x}$*

Proof. The proof is a bit straightforward. The time derivative of V is worked out. Furthermore, the identity $(A+BK)^\top P + P(A+BK) + Q = 0$ is substituted in order to arrive at the desired expression.

$$\begin{aligned}\frac{dV(\mathbf{x}(t))}{dt} &= \frac{d}{dt}\mathbf{x}^\top(t)P\mathbf{x}(t)\\ &= \dot{\mathbf{x}}^\top(t)P^\top\mathbf{x}(t) + \mathbf{x}^\top(t)P\dot{\mathbf{x}}(t)\\ &= \Big((A+BK)\mathbf{x}(t)\Big)^\top P^\top\mathbf{x}(t) + \mathbf{x}^\top(t)P(A+BK)\mathbf{x}(t)\\ &= \mathbf{x}^\top(t)\Big((A+BK)^\top P^\top + P(A+BK)\Big)\mathbf{x}(t)\\ &= \mathbf{x}^\top(t)(-Q)\mathbf{x}(t) = -\mathbf{x}^\top(t)Q\mathbf{x}(t)\end{aligned} \tag{A-2}$$

□

Theorem A.2 (Monotone Lyapunov triggering condition)**.** *The monotone Lyapunov triggering condition $\dot{V}(\mathbf{x}(t)) > -\sigma\mathbf{x}^\top(t)Q\mathbf{x}(t)$ is equivalent to the expression $\xi^\top(t)\mathcal{Q}\xi(t) > 0$ with $\mathcal{Q} = \begin{bmatrix}(\sigma-1)Q - 2PBK & PBK\\ (PBK)^\top & \mathbf{0}\end{bmatrix}$.*

Proof. for $t \in [\hat{t}_i,\ \hat{t}_{i+1})$ we want the condition $\dot{V}(\mathbf{x}(t)) > -\sigma\mathbf{x}^\top(t)Q\mathbf{x}(t)$ to always hold for some $\sigma \in (0, 1]$. The idea being that $t = \hat{t}_{i+1}$ would violate the triggering condition and the system would update the control input.

First in Eq. (A-3) the basic system dynamics are rewritten with the sample-and-hold method. In Eq. (A-4) the rewritten system dynamics are used to calculate $\frac{d}{dt}V(\mathbf{x}(t))$ for the sample-and-hold method. Notice that the continuous version of the derivative $\frac{d}{dt}V(\mathbf{x}(t)) = -\mathbf{x}^\top(t)Q\mathbf{x}(t)$ is used in the derivation. Afterwards, in Eq. (A-5), the triggering condition $\frac{d}{dt}V(\mathbf{x}(t)) > -\sigma\mathbf{x}^\top(t)Q\mathbf{x}(t)$ is combined with the acquired $\frac{d}{dt}V(\mathbf{x}(t))$ from Eq. (A-4). This expression is

organized and structured such that it becomes clear it is equivalent to Eq. (A-6).

$$\begin{aligned}\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + BK\hat{\mathbf{x}}(t)\\ &= A\mathbf{x}(t) + BK\mathbf{x}(\hat{t}_i)\\ &= A\mathbf{x}(t) + BK\mathbf{x}(t) + BK\Big(\mathbf{x}(\hat{t}_i) - \mathbf{x}(t)\Big)\end{aligned} \tag{A-3}$$

$$\begin{aligned}\frac{d}{dt}V(\mathbf{x}(t)) &= \frac{dV(\mathbf{x})}{d\mathbf{x}}\frac{d\mathbf{x}(t)}{dt}\\ &= \frac{dV(\mathbf{x})}{d\mathbf{x}}\Big(A\mathbf{x}(t) + BK\mathbf{x}(t) + BK(\mathbf{x}(\hat{t}_i) - \mathbf{x}(t))\Big)\\ &= \frac{dV(\mathbf{x})}{d\mathbf{x}}\Big(A\mathbf{x}(t) + BK\mathbf{x}(t)\Big) + \frac{dV(\mathbf{x})}{d\mathbf{x}}\Big(BK(\mathbf{x}(\hat{t}_i) - \mathbf{x}(t))\Big)\\ &= -\mathbf{x}^\top(t)Q\mathbf{x}(t) + 2\mathbf{x}^\top(t)PBK\Big(\mathbf{x}(\hat{t}_i) - \mathbf{x}(t)\Big)\\ &= \mathbf{x}^\top(t)\Big(-Q - 2PBK\Big)\mathbf{x}(t) + 2\mathbf{x}^\top(t)PBK\mathbf{x}(\hat{t}_i)\end{aligned} \tag{A-4}$$

$$-\sigma\mathbf{x}^\top Q\mathbf{x} < \frac{d}{dt}V(\mathbf{x}(t))$$

$$-\sigma\mathbf{x}^\top Q\mathbf{x} < \mathbf{x}^\top(t)\Big(-Q - PBK\Big)\mathbf{x}(t) + 2\mathbf{x}^\top(t)PBK\mathbf{x}(\hat{t}_i) \tag{A-5}$$

$$0 < \mathbf{x}^\top(t)\Big((\sigma-1)Q - PBK\Big)\mathbf{x}(t) + 2\mathbf{x}^\top(t)PBK\mathbf{x}(\hat{t}_i)$$

$$0 < \xi^\top(t)\begin{bmatrix}(\sigma-1)Q - 2PBK & PBK\\ (PBK)^\top & \mathbf{0}\end{bmatrix}\xi(t) \tag{A-6}$$

□

A-3 Invariant States On Ray

In this section, a general proof is given that states which lay on a line trough the origin have equivalent triggering times when using a quadratic homogeneous triggering condition on an Linear Time-Invariant (LTI) system. This follows from the fact that scaling around the origin for a LTI system is multiplying by a constant factor.

Theorem A.3 (Equivalent triggering times $\tau(\alpha\mathbf{x}_0) = \tau(\mathbf{x}_0)$)**.** *Let the system dynamics be given by* $\dot{\mathbf{x}}(t) = (A\mathbf{x}(t) + BK\hat{\mathbf{x}}(t))$*. Define* $\xi(t) = [\mathbf{x}^\top(t),\ \hat{\mathbf{x}}^\top(t)]^\top$ *and let* $\mathcal{Q}$ *be a quadratic homogeneous triggering condition of the form* $\xi^\top(t)\mathcal{Q}\xi(t) > 0$*. Let* $\mathbf{x}_0 \in \mathbb{R}^n$ *be the initial state not equal to the origin. Then* $\forall\alpha \in \mathbb{R}_{\neq 0}$ *it holds that* $\tau(\alpha\mathbf{x}_0) = \tau(\mathbf{x}_0)$*.*

Proof. Let $\alpha \in \mathbb{R}_{\neq 0}$ be a scalar. Let $\mathbf{x}_0 \in \mathbb{R}^n$ be the initial state such that $\mathbf{x}_0 \neq \mathbf{0}$. We describe two systems, both having the same system dynamics $\dot{\mathbf{x}}(t) = (A\mathbf{x}(t) + BK\hat{\mathbf{x}}(t))$. The first system will be denoted by $\mathbf{x}$ and initialized at $\mathbf{x}(t_0) = \mathbf{x}_0$, while the second system will be denoted by $\mathbf{x}_a$ and initialized somewhere on the ray of $\mathbf{x}_0$ through the origin (except the origin itself) i.e. at $\mathbf{x}_a(t_0) = \alpha\mathbf{x}_0 = \alpha\mathbf{x}(t_0)$.

Firstly, while looking at both systems before the first triggering time notice the systems can be related to each other, even when initialized at different points as shown in Eq. (A-7). Together with the initial condition $\mathbf{x}_a(t_0) = \alpha\mathbf{x}(t_0)$, we can integrate the equation $\dot{\mathbf{x}}_a = \alpha\dot{\mathbf{x}}$ from t_0 to

an arbitrary t to show that $\mathbf{x}_a(t) = \alpha\mathbf{x}(t)$ for all $t \geq t_0$. Remark that $\hat{x}(t) = \hat{\mathbf{x}}_a(t) = \mathbf{x}(t_0)$ since the we analyzed the system before the first triggering time.

Secondly, using this newly acquired relation of $\mathbf{x}_a(t) = \alpha\mathbf{x}(t)$, we are interested in the triggering times of both systems. Define $\xi_a(t) = [\mathbf{x}_a^\top(t),\ \hat{\mathbf{x}}_a^\top(t)]^\top$. Notice, while not triggered so far we know $\hat{\mathbf{x}}_a(t) = \mathbf{x}(t_0)$. Furthermore, since $\mathbf{x}_a(t) = \alpha\mathbf{x}(t)$ we derive at $\xi_a(t) = \alpha\xi(t)$. Now it follows from the triggering condition being quadratic and homogeneous that the triggering times are equivalent as shown in Eq. (A-8) by dividing a positive factor α^2 from both sides.

$$\begin{aligned}\dot{\mathbf{x}}_a(t_0) &= A\mathbf{x}_a(t_0) + BK\alpha\hat{\mathbf{x}}_a(t_0)\\ &= A\alpha\mathbf{x}(t_0) + BK\alpha\mathbf{x}(t_0)\\ &= \alpha\dot{\mathbf{x}}(t_0)\end{aligned} \tag{A-7}$$

$$\begin{aligned}0 &< \alpha\xi^\top(t)\mathcal{Q}\alpha\xi(t)\\ 0 &< \alpha^2\xi^\top(t)\mathcal{Q}\xi(t)\\ &\equiv\\ 0 &< \xi^\top(t)\mathcal{Q}\xi(t)\end{aligned} \tag{A-8}$$

□

A-4 Average Decay Rate

Given a LTI system, a stabilizing static feedback gain and different triggering conditions that each guarantee a different decay rate of a Common Lyapunov Function (CLF) $V(\mathbf{x}) = \mathbf{x}^\top P\mathbf{x}$. Given a finite path (possibly a cycle) in the l-complete abstraction, represented by the extended sequence of both inter-sample times and guaranteed decay rates $(\tau_1\tau_2\ldots\tau_N, \sigma_1\sigma_2\ldots\sigma_N)$. Since each guarantees a decay rate of of the CLF V, one can calculate the total decay factor of the whole path as shown in A-9. The optimization quantity of $\hat{\tau} = \frac{\sum_{i=1}^N \tau_i}{N}$ is already determined, but we want to establish a mathematical motivation to optimize the performance.

$$\begin{aligned}V(\mathbf{x}(t_0+\tau_1)) &= e^{-\kappa\sigma_1\tau_1}V(\mathbf{x}(t_0))\\ V(\mathbf{x}(t_0+\sum_{i=1}^N\tau_i)) &= e^{-\kappa\sigma_1\tau_1}e^{-\kappa\sigma_2\tau_2}\ldots e^{-\kappa\sigma_N\tau_N}V(\mathbf{x}(t_0))\\ &= e^{-\kappa\sum_{i=1}^N\sigma_i\tau_i}V(\mathbf{x}(t_0))\end{aligned} \tag{A-9}$$

Therefore, the first quantity of interest is minimizing the decay factor $e^{-\kappa\sum_{i=1}^N\sigma_i\tau_i}$ of the complete path. Notice κ is a positive constant factor among V and the triggering conditions. Therefore, minimizing $e^{-\kappa\sum_{i=1}^N\sigma_i\tau_i}$ is equivalent to maximizing $\sum_{i=1}^N\sigma_i\tau_i$ along the entire path.

Setting the weight of each edge equal to $\sigma_i\tau_i$ and taking the average along the edges would result in an optimization quantity of $\frac{1}{N}\sum_{i=1}^N\sigma_i\tau_i$. However, this quantity is not desired,

since it symbolizes the absolute decay factor of the V, and not the decay rate of V! When optimizing this quantity, a low σ_i which is held for a long time, may be preferred over a faster σ_j value. Therefore, we will instead focus on the decay rate of V.

Again the quantity of interest is $\sum_{i=1}^{N} \sigma_i \tau_i$. Set the weight of each edge equal to $\sigma_i \tau_i$, but now take the timed average instead of just the average. This would result in an optimizing quantity of $\frac{1}{\sum_{i=1}^{N} \tau_i} \sum_{i=1}^{N} \sigma_i \tau_i$. In contrast to before, this does prefer a higher value for σ_i over a lower value σ_j since it is just the timed average of all σ_i values. The resulting descriptions for $\hat{\sigma}$ and $\hat{\tau}$ are given in Eq. (A-10). Note, these are for a single path, and thus not the values for the complete graph by minimizing over them as is done in Eq. (3-1).

$$\begin{aligned} \hat{\tau} &= \frac{\sum_{i=1}^{N} \tau_i}{N} \\ \hat{\sigma} &= \frac{\sum_{i=1}^{N} \sigma_i \tau_i}{\sum_{i=1}^{N} \tau_i} = \frac{\sum_{i=1}^{N} \sigma_i \tau_i}{N \hat{\tau}} \end{aligned} \tag{A-10}$$

Notice, by using this description of $\hat{\sigma}$ and $\hat{\tau}$, the decay factor of V simplifies and is shown in Eq. (A-11).

$$\begin{aligned} V(\mathbf{x}(t_0 + \sum_{i=1}^{N} \tau_i)) &= e^{-\kappa \sum_{i=1}^{N} \sigma_i \tau_i} V(\mathbf{x}(t_0)) \\ &= e^{-\kappa N \hat{\sigma} \hat{\tau}} V(\mathbf{x}(t_0)) \end{aligned} \tag{A-11}$$

Appendix B

Algorithms

In this appendix, some algorithms are described in a bit more details. Since they are used throughout the thesis project, I did want to include them, but found that they did not belong in text.

B-1 Karp's Algorithm

Given a directed strongly connected graph $G = (E, V)$, here V is the set vertices and $E \subseteq V \times V$ is the set of edges. Let $w : E \to \mathbb{R}$ be a weight function on G and C_G the set of all cycles in G. Karps algorithm [26] is used for calculating the exact minimum cycle mean λ^* of G. Define the cycle mean of a cycle $c \in C_G$ by $\lambda(c) = \frac{1}{|c|} \sum_{e \in c} w(e)$. The minimum cycle mean is then given by $\lambda^* = \min_{c \in C_G} \lambda(c)$. Notice that it is equivalent to only consider the primitive cycles in the cycle set C_G.

Choose a starting source node v_0 in the graph G. Since G is strongly connected, every other edge can be reached from this node, and vice versa end up in v_0 again. Define a minimum edge progression function $F_k : V \to \mathbb{R} \cup \infty$ parameterized by k where $F_k(v)$ represents the minimum distance from v_0 to a node v (i.e. minimum total weight of the edges between v_0 and v) while the path contains exactly k edges. If such a path does not exists for the given amount of edges, the value of $F_k(v)$ is set to ∞.

Notice that computing $F_0(v)$ for all $v \in V$ is trivial since only the source node v_0 can be reached using no edges and will have a distance of $F_0(v_0) = 0$. Karp's algorithm is so efficient because the other distances $F_k(v)$ for $k \in \mathbb{N}_0$ van be calculated recursively by Eq. (B-1). The distances $F_k(v)$ for all $v \in V$ and for all $k \leq n$ can be computed in $\mathcal{O}(nm)$ time. Here, n is the amount of nodes $n = |V|$ and m the amount of edges $m = |E|$ in the graph G. Notice, while only $k \leq n$, in the minimization of the recursion still all edges are traversed, hence the m term in the time complexity.

$$d^k(v) = \min_{(v^*, v) \in E} d^{k-1}(v^*) + w(v^*, v) \tag{B-1}$$

Once all distances are calculated. the value for the minimum mean cycle is found by Eq. (B-2). Which can be computed in n^2 comparisons.

$$\lambda^* = \min_{v \in V} \max_{0 \leq k \leq n-1} \frac{F_n(v) - F_k(v)}{n-k} \tag{B-2}$$

A detailed proof of Karp's original algorithm can be found in [26]. It's important to notice some shortcomings of Karp's algorithm due to shortcomings in the proof as shown by [28]. Chaturvedi and McConnell give an alternate method using backpointers to construct the edge progressions $F_k(v)$.

B-2 Bellman-Ford Algorithm

The Bellman-Ford algorithm is an algorithm for finding the shortest path (path of minimum weight) in a directed graph. It was first introduced by Shimbel [44] and only later by Bellman [31] and Ford [32].

Given a directed strongly connected graph $G = (E, V)$, here V is the set vertices and $E \subseteq V \times V$ is the set of edges. Let $w : E \to \mathbb{R}$ on G be a weight function. The goal of the Bellman-Ford algorithm is to compute the shortest distance from a source node, to all other nodes in V. The algorithm can handle negative weights, but if a negative cycle is present, the distances can get arbitrary low. Luckely, this algorithm can detect if a negative cycle is present.

Introduce a distance function $d : V \to \mathcal{R} \cup \infty$ which will describe the distance from a source node v_0 to every node in V. This distance function is build recursively. First, initialize the distance function zero for the source node $d(v_0) = 0$ and infinity elsewhere $d(v) = \infty$ for $v \neq v_0$.

The distance function is adjusted by looping over all edges in $(u, v) \in E$, with if the simple rule shown in Eq. (B-3) holds, the distance function $d(v)$ is adjusted by setting $d(v) = d(u) + w((u, v))$. This procedure of looping over all edges needs to be repeated $|V| - 1$ times in order to ensure all minimum distances are found from the source node. Therefore, the time complexity of this algorithm is $\mathcal{O}(|V||E|)$.

$$d(v) > d(u) + w((u, v)) \tag{B-3}$$

This algorithm can also be used to check if there are negative cycles in the graph. Namely, if one more iteration is applied of looping over all edges and checking if Eq. (B-3) applies, all distances should not change if there is no negative cycle (since all correct distances from the source node should be found). If it is the case that distances get reduced, then there must be a negative cycle in the graph G.

B-3 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is an algorithm for finding all shortest paths (path of minimum weight) in a directed graph. The current interpretation is given by Floyd in [45]. However,

the essence is equivalent to the traversing algorithms already found by Warshall [46] and Roy [47].

Given a directed strongly connected graph $G = (E, V)$, here V is the set vertices and $E \subseteq V \times V$ is the set of edges. Let $w : E \to \mathbb{R}$ on G be a weight function. The goal of the Floyd-Warshall algorithm is to construct the shortest path between every pair of nodes. Notice this algorithm will only result in correct results if there do not exists negative cycles in the graph (while negative weights are permitted).

Let us define a distance function $d : V \times V \to \mathbb{R} \cup \infty$ which describes the minimum distance (sum of weights along the edges) along the possible paths from the first node to the second. The goal is to complete this function, but first it can be initialized trivially by setting $d(v, v) = 0$ for all $v \in V$ and inserting all direct edges given by E, all other distances are for now set to ∞ since they are unknown or impossible.

The smart procedure is in recursively updating the distance function d, which only has to be done $|V|$ times exactly. Each iteration, take a $v^* \in V$ until every vertex has been once. Use v^* and see which path use this vertex in order to construct new paths. For all $v \in V$ such that $d(v, v^*) \neq \infty$ and for all $v^{**} \in V$ such that $d(v^*, v^{**}) \neq \infty$, if the current distance $d(v, v^{**}) > d(v, v^*) + d(v^*, v^{**})$, update the current distance since we have shown there exists a shorter path. After using every helper node $v^* \in V$ once, the distance function is complete and all shortest paths between all possible nodes are stored in d. The construction of d does take computationally $\mathcal{O}(|V|^3)$ time.

Some important notices have to be made about the Floyd-Warshall algorithm. Since a cycle is just a path where the starting and ending node are the same, Floyd-Warshall can also be used for minimum cycle detection by looking at $d(v, v)$ for all $v \in V$. Furthermore, it can also be used as a negative cycle detection algorithm since as soon a $v \in V$ has negative distance $d(v, v) < 0$, there exists a negative cycle in the graph.

B-4 Brim's Algorithm

The algorithm described by Brim et. al. in [41] consist of two parts. Firstly they give an efficient way to solve energy games and secondly they show how Mean-Payoff Game (MPG)s can be reduced to solving multiple energy games. We are interested in the latter. Currently this is still one of the fastest computational ways to solve MPGs with a quasipolynomial time complexity of $\mathcal{O}(nmW)$ (for the decision, synthesis and three-way partition problem) and $\mathcal{O}(mn^2W(\log n + \log W))$ (for the value problem and optimal strategy synthesis), in which we are interested in the latter. Here the number of vertices is given by $n = |V|$, the number of edges by $m = |E|$ and the maximum absolute weight by $W = \max_{e \in E}(|w(e)|)$.

Let $\Gamma = (V_0, V_1, E, w)$ be a MPG with V_0 the set of controllable nodes, V_1 the set of uncontrollable nodes such that $V_0 \cap V_1 = \varnothing$ and denote $V = V_0 \cup V_1$ the set of all nodes. Furthermore, let $E \subset V \times V$ be the transition function and $w : E \to \{-W, \ldots, W\}$ the weight function bounded by some $W \in \mathbb{R}_{>0}$. We will only illustrate the outline of Brim et. al. second algorithm, since we are interested in the value problem and the optimal strategy synthesis. However, it does make use of the three-way partition problem which is solved by there first algorithm. For context, the first algorithm, called: *Value-iteration algorithm for energy games*, can determine the set of states for which a no positive mean-payoff can be guaranteed.

The three-way partition problem aims to divide V in three sets depending on a guessed game value ν such that $|\nu| \leq W$. Create two new MPGs $\Gamma' = (V_0, V_1, E, w - \nu)$ and $\Gamma'' = (V_1, V_0, E, -w + \nu)$. Using there first algorithm on Γ', one can construct the set of states that can not guarantee the game value ν, denoted by $V_{\leq \nu}$. Likewise, applying algorithm 1 on Γ'' results in the set $V_{>\nu}$. Similar combinations can be used to get the desired sets: $V_{<\nu}$, $V_{=\nu}$ and $V_{>\nu}$.

Their second algorithm uses this to divide the original MPG Γ into smaller and smaller MPG problems recursively. The trick is that the nodes in $V_{=\nu}$ can be assigned the value ν, while the nodes in the other sets can each have their own new mean-payoff problem with a new estimate of ν. Since the exact possible values of the weights are known beforehand (recall that w maps to a finite set of values), the possible game values ν must a fraction $\nu = \frac{q}{l}$ where $1 \leq l \leq |V|$ (since ν comes from a cycle in the MPG with a maximum of $|V|$ nodes) and $q \in \{-W, \ldots, W\}$. The possible values for ν can be iterated by a binary search over the search space.

B-5 Reachability Games

Given a non-blocking finite arena $\mathcal{A} = (V_0, V_1, E)$ and a reachable set $W \subseteq V = V_0 \sup V_1$. We would like to construct the attractor set $Att(\mathcal{A}, W)$, defined by all states $v \in V$ for which a strategy exists, that starting from v guarantees to enter the set W at some point. This set $Att(\mathcal{A}, W)$ is constructed by induction. We start by setting our initial set $W_0 = W$ and increment it step-wise. For this, let us introduce the pre function $pre : \mathcal{P}(V) \to \mathcal{P}(V)$ of a subset in V. The $pre(X)$ function is defined below by taking all vertices in V_0 which can reach X the next step and all vertices in V_1 which can only reach X the next step.

$$pre(X) = \{v \in V_0 \mid \exists x \in X : (v, x) \in E\} \cup \{v \in V_1 \mid \forall (v, x) \in E : x \in X\}$$

Now, one can extend the set for which one knows a strategy exists to end up in W by inductively creating $W_{i+1} = W_i \cup pre(W_i)$. Since for all i, we know $W_i \subseteq W_{i+1} \subseteq V$ and V only has a finite amount of vertices, we are guaranteed to converge. As soon as we reach a value for i such that $pre(W_i) = W_i$, the algorithm has converged and we have constructed the attractor set $Att(\mathcal{A}, W) = W_i$.

Appendix C

Simulations

In this appendix, simulations and statistics are given. At first, some details about the l-complete abstraction $S_{\mathcal{Q}/Q_l}$ are given for different settings in order to visualize the rate of growth of the abstraction, see Table C-1 and Table C-2. A resulting l-complete abstraction is visualized in Figure C-1. As one can see, the readability of the states is low. Therefore, we switch to a number reference representation of the quotient states in Figure C-2. For more details about the interconnections between the states, I will refer to the gitlab page: gitlab.tudelft.nl/sync-lab/ETCetera/-/tree/menno_thesis.

Before we play the safety (dual reachability) game to construct the Maximal Permissive Controller (MPC) S_{mpc}, one has to define a burst condition or minimum Average Inter Sample Time (AIST) $\hat{\tau}$ in order to mark states to avoid. In Figure C-3, such a marking is done. Once the safety game is finished, some additional states are marked as losing and some transitions which may lead to losing states are removed. This is visualized in Figure C-4. However, notice that the losing states (marked with red) are not included in S_{mpc} and are there pure for illustration purposes.

In the rest of the figures, some simulation are made using different settings for the infinite horizon controller S_{ihc}. We visualized the state-space progressions of the internal states. Furthermore, the Lyapunov function, guaranteed Lyapunov decay and triggering times are given.

Decay rates used: $0.2, 0.5$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
8	225	940	4.18
9	403	1732	4.30
10	523	2286	4.37
11	736	3176	4.32
Decay rates used: $0.2, 0.5, 0.8$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
8	917	5489	5.99
9	1377	8501	6.17
10	1754	10912	6.22
11	2320	14258	6.15
Decay rates used: $0.2, 0.5, 0.6, 0.8$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
8	2288	18396	8.04
9	3433	28402	8.27
10	4435	37601	8.48
11	5809	47976	8.26

Table C-1: Increasing the number of triggering conditions in the abstraction and the maximum discrete Inter-Sample Time (IST) k_{max}. Used monotone Lyapunov decay triggering conditions, sampling frequency $\Delta = 0.05$ with abstraction depth $l = 3$.

Depth $l = 2$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
9	82	666	8.12
11	147	1397	9.50
13	201	2028	10.09
15	250	2587	10.35
Depth $l = 3$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
9	331	1419	4.29
11	569	2512	4.41
13	883	4022	4.55
15	1175	5429	4.62
Depth $l = 4$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
9	1036	3556	3.43
11	1597	5508	3.45
13	3084	10880	3.53
15	4701	19397	4.13
Depth $l = 5$			
$\mathbf{k_{max}}$	**regions**	**transitions**	**ratio**
9	2820	8576	3.04
11	4048	12384	3.06
13	10119	34221	3.38
15	18330	71498	3.90

Table C-2: Increasing the depth of the l-complete abstraction and the maximum discrete IST k_{max}. Used monotone Lyapunov decay triggering conditions with decay rates: $0.1, 0.8$, sampling frequency $\Delta = 0.05$.

Figure C-1: The l-complete abstraction $S_{\mathcal{Q}/Q_3}$ of depth $l = 3$ visualized as Transition System (TS), using two monotone Lyapunov triggering conditions with decay rates: $0.1, 0.8$ and a maximum discrete IST of $k_{max} = 8$. In the quotient states, one can see the trigger condition sequence on the left and the IST sequence on the right.

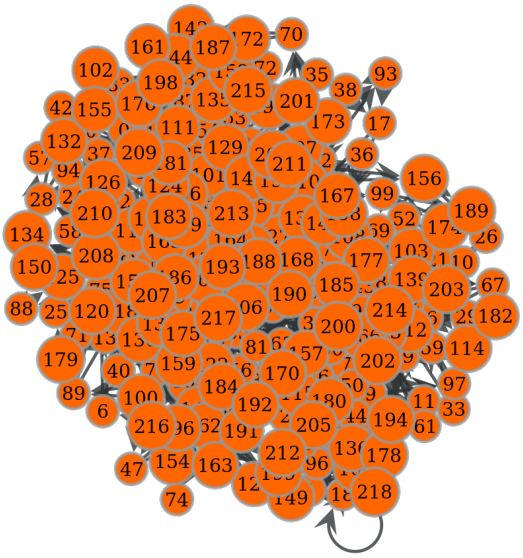


Figure C-2: The l-complete abstraction $S_{\mathcal{Q}/Q_3}$ from Figure C-1 with integer representation of the quotient states.

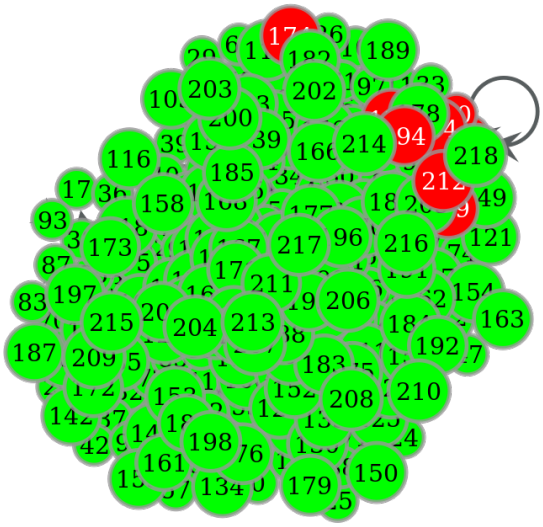


Figure C-3: Using the l-complete abstraction of Figure C-1. Marked states according to the burst condition: at least 3 triggering instances within 12 discrete time ($12\Delta = 0.6$ time)

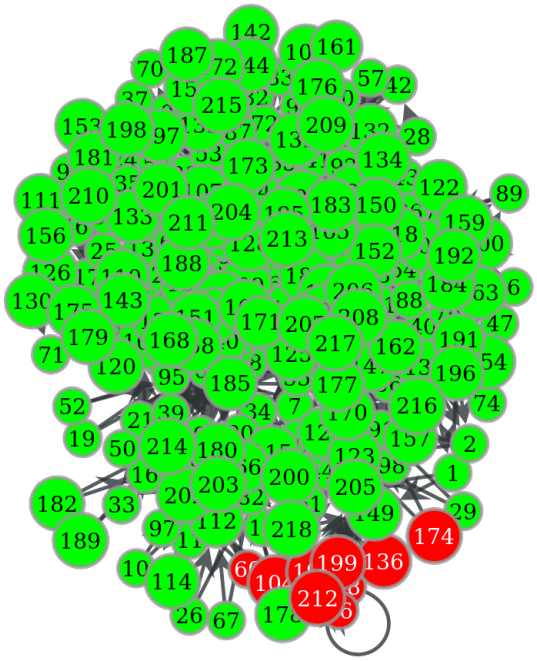


Figure C-4: Finished playing the safety game on Figure C-3. Marked additional losing states and deleted possible losing transitions. Notice the marked states are left in for illustration purposes, they are not present in the MPC S_{mpc}.

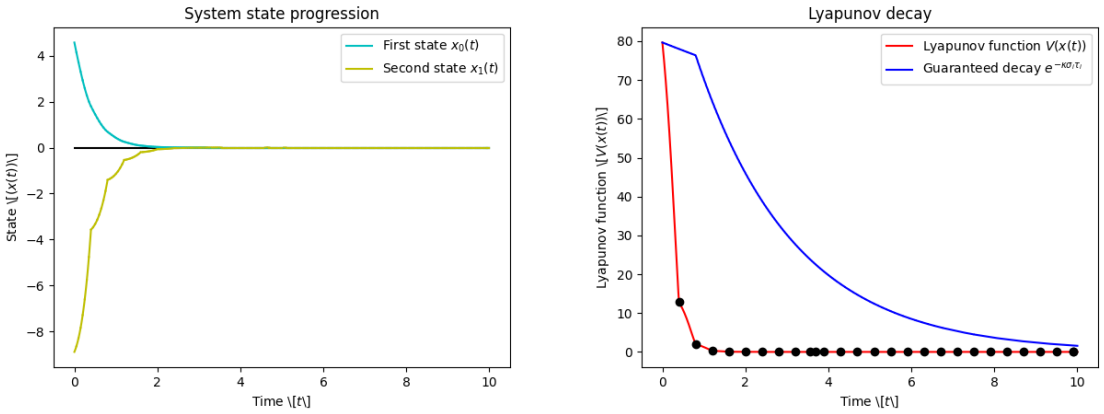


Figure C-5: Simulation of the infinite horizon controller S_{ihc}. Left the state-space response and on the right the Lyapunov decay. Depth of l-complete abstraction is $l = 3$, maximum discrete IST is $k_{max} = 8$, minimum average discrete IST of $\hat{\tau} > 4\Delta$. Using two monotone Lyapunov triggering conditions with decay rate: $0.1, 0.8$. The controller consists of 219 safe regions (excluding 9 losing states) and guarantees an exponential decay of V with a rate of 0.545.

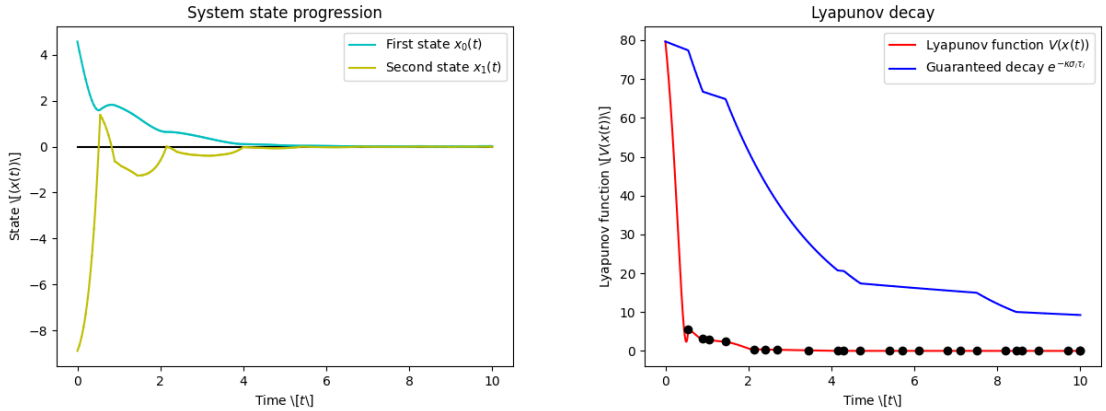


Figure C-6: Simulation of the infinite horizon controller S_{ihc}. Left the state-space response and on the right the Lyapunov decay. Depth of l-complete abstraction is $l = 4$, maximum discrete IST is $k_{max} = 15$, minimum average discrete IST of $\hat{\tau} > 5\Delta$. Using two monotone Lyapunov triggering conditions with decay rate: $0.1, 0.8$. The controller consists of 4651 safe regions (excluding 50 losing states), and guarantees an exponential decay of V with a rate of 0.508

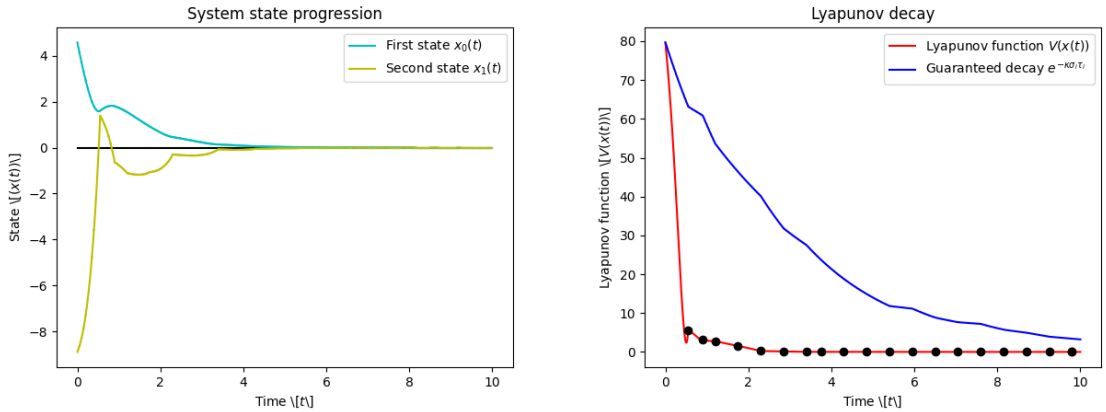


Figure C-7: Simulation of the infinite horizon controller S_{ihc}. Left the state-space response and on the right the Lyapunov decay. Depth of l-complete abstraction is $l = 3$, maximum discrete IST is $k_{max} = 11$, minimum average discrete IST of $\hat{\tau} > 5\Delta$. Using three monotone Lyapunov triggering conditions with decay rate: $0.2, 0.5, 0.8$. The controller consists of 2160 safe regions (excluding 160 losing states), and guarantees an exponential decay of V with a rate of 0.531

First simulation from Figure C-5		
Quotient state	IST sequence	triggering sequence
165	(0.1, 0.1, 0.1)	(8, 8, 8)
213	(0.1, 0.1, 0.8)	(8, 8, 8)
4	(0.1, 0.8, 0.8)	(8, 8, 8)
14	(0.8, 0.8, 0.8)	(8, 8, 8)
14	(0.8, 0.8, 0.8)	(8, 8, 8)
⋮	⋮	⋮
14	(0.8, 0.8, 0.8)	(8, 8, 8)
28	(0.8, 0.8, 0.8)	(8, 8, 3)
37	(0.8, 0.8, 0.8)	(8, 3, 4)
21	(0.8, 0.8, 0.8)	(3, 4, 8)
58	(0.8, 0.8, 0.8)	(4, 8, 8)
14	(0.8, 0.8, 0.8)	(8, 8, 8)
⋮	⋮	⋮

Table C-3: Controller state progressions of the simulations. Some insight in the IST and triggering sequences generated by the simulations.

Bibliography

[1] C. Bissell, "A history of automatic control," in *Springer handbook of automation* (S. Y. Nof, ed.), no. LXXVI in Springer handbook series, pp. 53–69, Heidelberg, Germany: Springer Verlag, 2009. In: Part A Development and Impacts of Automation.

[2] T. Schouten, "Stability analysis of the equations of motion for the 2014 delft solar boat," B.S. Thesis, Technical University of Delft, 2015.

[3] K. Johan Åström and B. Bernhardsson, "Comparison of periodic and event based sampling for first-order stochastic systems," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 5006–5011, 1999. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.

[4] K.-E. Åarzén, "A simple event-based pid controller," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 8687–8692, 1999. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.

[5] J. Yook, D. Tilbury, and N. Soparkar, "Trading computation for bandwidth: reducing communication in distributed control systems using state estimators," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 4, pp. 503–518, 2002.

[6] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1680–1685, 9 2007.

[7] W. P. Heemels, M. C. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Transactions on Automatic Control*, vol. 58, pp. 847–861, 2013.

[8] A. Cervin and T. Henningsson, "Scheduling of event-triggered controllers on a shared network," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3601–3606, 2008.

[9] M. Mazo, A. Sharifi-Kolarijani, D. Adzkiya, and C. Hop, "Abstracted models for scheduling of event-triggered control data traffic," *Lecture Notes in Control and Information Sciences*, vol. 475, pp. 197–217, 2018.

[10] T. Moor and J. Raisch, "Supervisory control of hybrid systems within a behavioural framework," *Systems & Control Letters*, vol. 38, no. 3, pp. 157–166, 1999.

[11] A.-K. Schmuck, P. Tabuada, and J. Raisch, "Comparing asynchronous l-complete approximations and quotient based abstractions," 2015.

[12] A.-K. Schmuck and J. Raisch, "Asynchronous l-complete approximations," *Systems & Control Letters*, vol. 73, pp. 67–75, 2014.

[13] A. Lyapunov, "The general problem of stability of motion," Master's thesis, University of Kharkov, 1892.

[14] M. U. Bikdash and R. A. Layton, "An energy-based lyapunov function for physical systems," *IFAC Proceedings Volumes*, vol. 33, no. 2, pp. 81–86, 2000. IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control, Princeton, NJ, USA, 16-18 March 2000.

[15] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[16] A. Girard, "Dynamic triggering mechanisms for event-triggered control," *IEEE Transactions on Automatic Control*, vol. 60, pp. 1992–1997, 1 2013.

[17] W. P. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3270–3285, 2012. Lyapunov triggering condition.

[18] X. Wang and M. D. Lemmon, "Event design in event-triggered feedback control systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 2105–2110, 2008.

[19] M. Velasco, P. Martí, and E. Bini, "On lyapunov sampling for event-driven controllers," *IEEE Conf. Decision Control*, pp. 6238–6243, 2009.

[20] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems: Robust stability and control for systems that combine continuous-time and discrete-time dynamics," *IEEE Control Systems*, vol. 29, pp. 28–93, 2009.

[21] R. Goebel and A. R. Teel, "Solutions to hybrid inclusions via set and graphical convergence with stability theory applications," *Automatica*, vol. 42, pp. 573–587, 2006.

[22] A. Benzaouia, *Introduction to Switched Systems*, pp. 77–94. London: Springer London, 2012.

[23] G. H. Mealy, "A method for synthesizing sequential circuits," *The Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955.

[24] E. F. Moore, "Gedanken-experiments on sequential machines," vol. 34, pp. 129–154, 1958.

[25] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. USA, Boston: Addison-Wesley Longman Publishing Co., Inc., 3rd ed., 2006.

[26] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Mathematics*, vol. 23, pp. 309–311, 1 1978.

[27] A. Dasdan and R. K. Gupta, "Faster maximum and minimum mean cycle algorithms for system-performance analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 889–899, 1998.

[28] M. Chaturvedi and R. M. McConnell, "A note on finding minimum mean cycle," *Information Processing Letters*, vol. 127, pp. 21–22, 11 2017.

[29] S. M. Burns and A. J. Martin, "Performance analysis and optimization of asynchronous circuits," in *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI*, (Cambridge, MA, USA), p. 71–86, MIT Press, 1991.

[30] E. L. Lawler, *Optimal Cycles in Graphs and the Minimal Cost-To-Time Ratio Problem*, pp. 37–60. Vienna: Springer Vienna, 1972.

[31] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[32] L. R. Ford, *Network Flow Theory.* Santa Monica, CA: RAND Corporation, 1956.

[33] M. Hartmann and J. B. Orlin, "Finding minimum cost to time ratio cycles with small integral transit times," *Networks*, vol. 23, no. 6, pp. 567–574, 1993.

[34] K. Bringmann, T. D. Hansen, and S. Krinninger, "Improved algorithms for computing the cycle of minimum cost-to-time ratio in directed graphs," 4 2017.

[35] S. P. Boyd and L. Vandenberghe, *Convex Optimization.* USA, New York: Cambridge University Press, 7th ed., 2004.

[36] P. Tabuada, "Verification and control of hybrid systems: A symbolic approach," *Verification and Control of Hybrid Systems: A Symbolic Approach*, pp. 1–202, 2009.

[37] A. S. Kolarijani and M. M. Jr., "Traffic characterization of LTI event-triggered control systems: a formal approach," *CoRR*, vol. abs/1503.05816, 2015.

[38] G. A. D. Gleizer and M. Mazo, "Computing the sampling performance of event-triggered control," *HSCC 2021 - Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (part of CPS-IoT Week)*, 3 2021.

[39] E. Gradel, W. Thomas, and T. Wilke, "Automata logics, and infinite games: A guide to current research," *Lecture Notes in Computer Science*, vol. 2500, pp. 265–274, 2002.

[40] 1979.

[41] L. Brim, . J. Chaloupka, . L. Doyen, . R. Gentilini, . J. F. Raskin, J. Chaloupka, L. Doyen, J. F. Raskin, and R. Gentilini, "Faster algorithms for mean-payoff games," *Form Methods Syst Des*, vol. 38, pp. 97–118, 2011.

[42] N. Fijalkow, P. Gawrychowski, and P. Ohlmann, "The complexity of mean payoff games using universal graphs," 2019.

[43] E. Lawler, *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[44] L. B. Alfonso Shimbel, "Structural parameters of communication networks.," *Bulletin of Mathematical Biophysics*, vol. 15, pp. 501–507, 1953.

[45] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, p. 345, 6 1962.

[46] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM (JACM)*, vol. 9, pp. 11–12, 1 1962.

[47] B. Roy, "Transitivité et connexité," *C. R. Acad. Sci. Paris*, vol. 249, p. 216–218, 1959.

Glossary

List of Acronyms

AIST	Average Inter Sample Time
BF	Bellman-Ford
CETC	Continuous Event Triggered Control
CLF	Common Lyapunov Function
DTM	Dynamic Triggering Mechanism
ETC	Event-Triggered Control
FW	Floyd-Warshall
IST	Inter-Sample Time
LMI	Linear Matrix Inequality
LTI	Linear Time-Invariant
LTS	Labeled Transition System
MPG	Mean-Payoff Game
MPC	Maximal Permissive Controller
NCS	Networked Control System
PETC	Periodic Event-Triggered Control
SAlCA	Strongest (Asynchronous) l-Complete Approximations
STC	Self-Triggered Control
TA	Timed Automata
TS	Transition System