Computing the sampling performance of event-triggered control

De Gleizer, Gabriel A.; Mazo, Manuel

Computing the sampling performance of event-triggered control

Gabriel de A. Gleizer
g.gleizer@tudelft.nl
TU Delft
Delft, The Netherlands

Manuel Mazo Jr.
m.mazo@tudelft.nl
TU Delft
Delft, The Netherlands

ABSTRACT

In the context of networked control systems, event-triggered control (ETC) has emerged as a major topic due to its alleged resource usage reduction capabilities. However, this is mainly supported by numerical simulations, and very little is formally known about the traffic generated by ETC. This work devises a method to estimate, and in some cases to determine exactly, the minimum average inter-sample time (MAIST) generated by periodic event-triggered control (PETC) of linear systems. The method involves abstracting the traffic model using a bisimulation refinement algorithm and finding the cycle of minimum average length in the graph associated to it. This always gives a lower bound to the actual MAIST. Moreover, if this cycle turns out to be related to a periodic solution of the closed-loop PETC system, the performance metric is exact.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Theory of computation** → *Abstraction*; • **Networks** → *Cyber-physical networks*.

1 INTRODUCTION

Nowadays, most control systems are implemented using networks as the communication medium between sensors, controllers, and actuators. The classic method of performing control through digital media is called periodic *sample-and-hold control*, where sensor data is gathered at a fixed sampling rate, control commands are then immediately updated and sent to actuators, which hold this command for the sampling period time. Choosing the sampling period is based fundamentally on a worst-case analysis across the state-space of the system. Disrupting this periodic paradigm, event-triggered control (ETC) works by sampling only when a significant condition happens, thus adapting the sampling rate to the system state: this gives it the potential to drastically reduce communications in comparison to periodic control. This concept dates back from [1], with the formal methods to design event conditions to achieve

desired stability properties presented in [2]. Since then, many authors (e.g. [3–5]) worked on event design to improve sampling performance while guaranteeing stability and control performance properties, or to improve the practical implementation aspects of ETC, such as the periodic event-triggered control (PETC) of [6]. In PETC, event conditions are checked periodically, but sensor values are only communicated upon the condition satisfaction. It is important to note, however, that most of the evidence of superiority of ETC in comparison to periodic control, in what concerns bandwidth usage, is only supported through numerical case studies. In some cases, such as PETC, one can obtain a straightforward *qualitative* assertion of non-inferiority by setting the *event-checking period* equal to the largest periodic sampling time one can obtain. Nonetheless, no *quantitative* measurement of this superiority has been established; e.g., in PETC, doing this strategy often leads to periodic triggering of the events, bringing no benefits at all.

The present work is concerned precisely with measuring the sampling performance of a given PETC system. More specifically, we aim at computing the *minimum average inter-sample time* (MAIST) of a PETC system: this translates directly to its expected network load or resource utilization. We focus on PETC due to its practical relevance, but also because, as observed in [7], it enables exact traffic abstractions—a central tool for the results of this paper. Literature related to this objective can be categorized in two main approaches. The first [8] focuses on understanding the qualitative asymptotic trends of the inter-sample times of planar linear systems. The authors conclude that, for small enough triggering condition parameters, the inter-sample times eventually converge to a fixed value or exhibit a periodic pattern. Despite providing very interesting insights, the results are limited to two-dimensional state-spaces, do not provide the quantitative information we are interested in, and, perhaps most importantly, are only valid for small triggering parameters: this way, ETC provides the least benefit. The second category is the use of symbolic abstractions [7, 9], which follow on the extensive work on partitioning and aggregation for abstractions, see [10]. These works are concerned with short-term prediction of inter-sample times in order to develop a scheduler that can, e.g., request sensor data before events are triggered; they do not capture long-term properties of the sampling behavior of ETC like the MAIST, which, we argue, provide a more definitive information about the sampling performance. Still in the same category, [11] has recently given a step towards understanding longer-term traffic patterns of PETC, by proposing the usage of a bisimulation-like algorithm which determines the m next inter-sample times from a given state. This allows a very conservative estimate of the MAIST by taking the minimum average of all such m-length sequences.

The present work builds upon the bisimulation-like algorithm of [11], which can be seen as a modified l-complete abstraction [12, 13], to compute the MAIST of PETC for linear time-invariant

systems. The main insight is that computing the MAIST once there is a finite-state simulation (abstraction) of the PETC traffic is easy, as it reduces to finding the cycle of minimum average length of the associated weighted graph [14]. Then, as we show, if we can *verify* that the minimum cycle in the abstraction exists in the actual closed-loop system, we will have obtained an exact value for the MAIST. This observation gives rise to the concept of *minimum-average-cycle-equivalent simulation*. If the minimum cycle is not exhibited by the PETC system, the value obtained through the abstraction is still a lower bound, and further refinements will eventually break it, providing tighter bounds.

1.1 Notation

We denote by $\mathbb{N}_0$ the set of natural numbers including zero, $\mathbb{N} :=$ $\mathbb{N}_0 \setminus \{0\}$, $\mathbb{N}_{\leq n} := \{1, 2, ..., n\}$, by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{R}_+$ the set of non-negative reals. We denote by $|x|$ the norm of a vector $x \in \mathbb{R}^n$, but if s is a sequence or set, $|s|$ denotes its length or cardinality, respectively. For a square matrix $A \in \mathbb{R}^{n \times n}$, we write $A > 0$ ($A \geq 0$) if A is positive definite (semi-definite). The set $\mathbb{S}^n$ denotes the set of symmetric matrices in $\mathbb{R}^n$. For a set $X \subseteq \Omega$, we denote by $\bar{X}$ its complement: $\Omega \setminus X$; We often use a string notation for sequences, e.g., $\sigma = abc$ reads $\sigma(1) = a, \sigma(2) = b, \sigma(3) = c$. Powers and concatenations work as expected, e.g., $\sigma^2 = \sigma\sigma = abcabc$. In particular, σ^ω denotes the infinite repetition of σ. For a relation $\mathcal{R} \subseteq X_a \times X_b$, its inverse is denoted as $\mathcal{R}^{-1} = \{(x_b, x_a) \in X_b \times X_a : (x_a, x_b) \in \mathcal{R}\}$. Finally, we denote by $\pi_{\mathcal{R}}(X_a) := \{x_b \in X_b \mid (x_a, x_b) \in \mathcal{R}$ for some $x_a \in X_b\}$ the natural projection of X_a onto X_b.

2 PROBLEM STATEMENT

Consider a linear time-invariant plant controlled with sample-and-hold state feedback [15] described by

$$\dot{\xi}(t) = A\xi(t) + BK\hat{\xi}(t), \qquad (1)$$

where $\xi(t) \in \mathbb{R}^{n_x}$ is the plant's state with initial value $x_0 := \xi(0)$, $\hat{\xi}(t) \in \mathbb{R}^{n_x}$ is the state measurement available to the controller, $K\hat{\xi}(t) \in \mathbb{R}^{n_u}$ is the control input, n_x and n_u are the state-space and input-space dimensions, respectively, and A, B, K are matrices of appropriate dimensions. The holding mechanism is zero-order: let $t_i \in \mathbb{R}_+, i \in \mathbb{N}_0$ be a sequence of sampling times, with $t_0 = 0$ and $t_{i+1} - t_i > \varepsilon$ for some $\varepsilon > 0$; then $\hat{\xi}(t) = \xi(t_i), \forall t \in [t_i, t_{i+1})$.

In ETC, a *triggering condition* determines the sequence of times t_i. In PETC, this condition is checked only periodically, with a fundamental checking period h. Figure 1 shows a simple diagram depicting the ETC scheme. We consider the family of static *quadratic triggering conditions* from [6] with an additional maximum inter-event time condition below:

$$t_{i+1} = \inf \left\{ kh > t_i, k \in \mathbb{N} \; \middle| \; \begin{matrix} \begin{bmatrix} \xi(kh) \\ \xi(t_i) \end{bmatrix}^\mathsf{T} Q \begin{bmatrix} \xi(kh) \\ \xi(t_i) \end{bmatrix} > 0 \\ \text{or } kh - t_i \geq \bar{k}h \end{matrix} \right\}, \quad (2)$$

where $Q \in \mathbb{S}^{2n_x}$ is the designed triggering matrix, and $\bar{k}$ is the chosen maximum (discrete) inter-event time.[1] Observing this equation,

---

[1] Typically, a maximum inter-event time exists naturally for a system with (P)ETC (see [16]). Still, one may want to set a smaller maximum inter-event time so as to establish a "heart beat" of the system. In any case, this is a necessity if one wants to obtain a finite-state abstraction of the system.
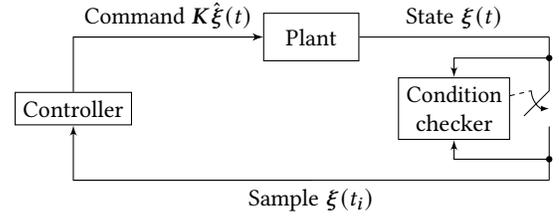

Figure 1: Block diagram of an ETC system.

note that the inter-event time $t_{i+1} - t_i$ is a function of $x_i := \xi(t_i)$; denoting $\kappa := (t_{i+1} - t_i)/h$ as the discrete inter-sample time, it follows that

$$\kappa(x_i) = \min \left\{ k \in \{1, 2, ...\bar{k}\} \mid x_i^\mathsf{T} N(k) x_i > 0 \text{ or } k = \bar{k} \right\},$$

$$N(k) := \begin{bmatrix} M(k) \\ I \end{bmatrix}^\mathsf{T} Q \begin{bmatrix} M(k) \\ I \end{bmatrix}, \qquad (3)$$

$$M(k) := A_\mathrm{d}(k) + B_\mathrm{d}(k)K := \mathrm{e}^{Ahk} + \int_0^{hk} \mathrm{e}^{A\tau} \mathrm{d}\tau BK.$$

where I denotes the identity matrix. Thus, the event-driven evolution of sampled states can be compactly described by the recurrence

$$\xi(t_{i+1}) = M(\kappa(\xi(t_i)))\xi(t_i). \qquad (4)$$

With this, each initial condition $x_0 \in \mathbb{R}^{n_x}$ leads to a sequence of samples x_i and inter-sample times $k_i(x_0)$ defined recursively as

$$x_{i+1} = M(\kappa(x_i))x_i$$
$$k_i(x_0) := \kappa(x_i),$$

for which one may attribute an *average inter-sample time* (AIST):

$$\mathrm{AIST}(x) := \liminf_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^n h k_i(x).$$

As usual, we use $\liminf$ instead of $\lim$ to obtain the limit lower bound in case the regular limit does not exist.

The goal of this paper is to devise a method to compute, for a given periodic event-triggered controlled system (1)–(2), its *minimum average inter-sample time* (MAIST), which is the minimal AIST across all possible initial conditions:

$$\mathrm{MAIST} := \inf_{x \in \mathbb{R}^{n_x}} \liminf_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^n h k_i(x). \qquad (5)$$

Calculating or even estimating the quantity above is challenging. How can one choose a sufficiently large n, or how can one exhaustively search for states to obtain one that yields the MAIST? This direct approach is unpromising, and thus we propose to find the value of Eq. (5) through finite-state abstractions.

3 BACKGROUND

The strategy to solve the problem posed in the previous section is to *abstract* the infinite-state system given by Eqs. (1)–(2) to a finite-state system, compute the equivalent to a MAIST in this abstraction, and establish a relation between the quantities of the original system and its abstraction. For that, we introduce the framework of [10] to formally relate systems of different natures, e.g., those described by

differential equations with those described by finite-state machines. Later, we present the notion of quantitative automata from [14] and how to compute the MAIST of a (priced) automaton.

3.1 Transition systems and abstractions

In [10], Tabuada gives a generalized notion of transition system:

DEFINITION 1 (TRANSITION SYSTEM [10]). *A system S is a tuple $(X, X_0, \mathcal{E}, \mathcal{Y}, H)$ where:*

- *X is the set of states,*
- *$X_0 \subseteq X$ is the set of initial states,*
- *$\mathcal{E} \subseteq X \times X$ is the set of edges (or transitions),*
- *$\mathcal{Y}$ is the set of outputs, and*
- *$H : X \to \mathcal{Y}$ is the output map.*

Here we have omitted the action set $\mathcal{U}$ from the original definition because we focus on autonomous systems. A system is said to be finite (infinite) state when the cardinality of X is finite (infinite). A transition in $\mathcal{E}$ is denoted by a pair (x, x'). We define $\text{Post}_S(x) := \{x' \mid (x, x') \in \mathcal{E}\}$ as the set of states that can be reached from x in one step. System S is said to be *non-blocking* if $\forall x \in X, \text{Post}_S(x) \neq \emptyset$. We call $x_0 x_1 x_2 ...$ an *infinite internal behavior*, or *run* of S if $x_0 \in X_0$ and $(x_i, x_{i+1}) \in \mathcal{E}$ for all $i \in \mathbb{N}$, and $y_0 y_1 ...$ its corresponding *infinite external behavior*, or *trace*, if $H(x_i) = y_i$ for all $i \in \mathbb{N}$. We denote by $B_S(r)$ the external behavior from a run $r = x_0 x_1 ...$ (in the case above, $B_S(r) = y_0 y_1 ...$), by $\mathcal{B}_x^\omega(S)$ the set of all infinite external behaviors of S starting from state x, and by $\mathcal{B}^\omega(S) := \bigcup_{x \in X_0} \mathcal{B}_x^\omega(S)$ the set of all infinite external behaviors of S.

The concepts of simulation and bisimulation are fundamental to establish formal relations between two transition systems.

DEFINITION 2 (SIMULATION RELATION [10]). *Consider two systems S_a and S_b with $\mathcal{Y}_a = \mathcal{Y}_b$. A relation $\mathcal{R} \subseteq X_a \times X_b$ is a simulation relation from S_a to S_b if the following conditions are satisfied:*

  i) *for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in \mathcal{R}$;*
  ii) *for every $(x_a, x_b) \in \mathcal{R}, H_a(x_a) = H_b(x_b)$;*
  iii) *for every $(x_a, x_b) \in \mathcal{R}$, we have that $(x_a, x_a') \in \mathcal{E}_a$ implies the existence of $(x_b, x_b') \in \mathcal{E}_b$ satisfying $(x_a', x_b') \in \mathcal{R}$.*

We say $S_a \preceq S_b$ when S_b simulates S_a, which is true if there exists a simulation relation from S_a to S_b.

DEFINITION 3 (BISIMULATION [10]). *Consider two systems S_a and S_b with $\mathcal{Y}_a = \mathcal{Y}_b$. S_a is said to be bisimilar to S_b, denoted $S_a \cong S_b$, if there exists a relation $\mathcal{R}$ such that:*

- *$\mathcal{R}$ is a simulation relation from S_a to S_b;*
- *$\mathcal{R}^{-1}$ is a simulation relation from S_b to S_a.*

Weaker but relevant relations associated with simulation and bisimulation are, respectively, *behavioral inclusion* and *behavioral equivalence*:

DEFINITION 4 (BEHAVIORAL INCLUSION AND EQUIVALENCE [10]). *Consider two systems S_a and S_b with $\mathcal{Y}_a = \mathcal{Y}_b$. We say that S_a is behaviorally included in S_b, denoted by $S_a \preceq_\mathcal{B} S_b$, if $\mathcal{B}^\omega(S_a) \subseteq \mathcal{B}^\omega(S_b)$. In case $\mathcal{B}^\omega(S_a) = \mathcal{B}^\omega(S_b)$, we say that S_a and S_b are behaviorally equivalent, which is denoted by $S_a \cong_\mathcal{B} S_b$.*

(Bi)simulations lead to behavioral inclusion (equivalence):

THEOREM 1 ([10]). *Given two systems S_a and S_b with $\mathcal{Y}_a = \mathcal{Y}_b$:*

- *$S_a \preceq S_b \implies S_a \preceq_\mathcal{B} S_b$;*
- *$S_a \cong S_b \implies S_a \cong_\mathcal{B} S_b$.*

3.2 Quantitative automata

While much of the field of formal methods in control is concerned with qualitative analyses, such as safety, stability, and reachability, often quantitative computations are of interest, like the problem we set ourselves to solve in this paper. In [14], Chatterjee et al. established a comprehensive framework for quantitative problems on finite-state systems, from which we borrow some definitions and results, trying to be consistent with the notation from the previous section as much as possible.

DEFINITION 5 (WEIGHTED AUTOMATON (ADAPTED FROM [14])). *A weighted automaton S is the tuple $(X, X_0, \mathcal{E}, \mathcal{Y}, H, \gamma)$, where*

- *$(X, X_0, \mathcal{E}, \mathcal{Y}, H)$ is a non-blocking transition system;*
- *$\gamma : \mathcal{E} \to \mathbb{Q}$ is the weight function.*

The adaptation we have made is that we include outputs to comply with previously introduced notation; again, we ignore the action set as we are interested in autonomous systems. For a given run $r = x_0 x_1 ...$ of S, $\gamma(r) = v_0 v_1 ...$ is the sequence of weights defined by $v_i = \gamma(x_i, x_{i+1})$.

A *value function* $\text{Val} : \mathbb{Q}^\omega \to \mathbb{R}$ attributes a value to an infinite sequence of weights $v_0 v_1 ... \in \mathbb{Q}^\omega$. Among the well-studied value functions, the one we are interested in is

$$\text{LimAvg}(v) := \liminf_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^{n} v_i.$$

A *LimAvg-automaton* is a weighted automaton equipped with the LimAvg value function. We define the value of a LimAvg-automaton as $V(S) := \inf\{\text{LimAvg}(\gamma(r)) \mid r \text{ is a run of } S\}$.[2] The following result is essentially an excerpt from Theorem 3 in [14], which uses the classical result from Karp [17]:

THEOREM 2. *Given a finite-state LimAvg-automaton S with $|X| = n$ and $|\mathcal{E}| = m$, $V(S)$ can be computed in $O(nm)$. Moreover, system S admits a cycle $x_0 x_1 ... x_k$ satisfying $x_i \to x_{i+1}, i < k$, and $x_k \to x_0$ s.t. $\text{LimAvg}(\gamma((x_0 x_1 ... x_k)^\omega)) = V(S)$.*

The cycle mentioned above is a *minimum average cycle* of the weighted digraph defined by S, and can be recovered in $O(n)$ using the algorithm in [18].

4 COMPUTING THE MAIST

From Theorem 2, we have an indication that it would be relatively straightforward to compute the minimum average inter-sample time of the PETC system (1)–(2) if we could represent it as a weighted automaton. Let us investigate how we can do this. We start by describing the evolution of samples of a PETC system,

---

[2]In [14], sup is used instead of inf because they consider it a more natural choice in general quantitative decision problems, considering the convention used in qualitative decisions (e.g., acceptance). As it is remarked in [14] itself, using inf is also a valid choice, and it is the most natural for our problem.

cf. Eq. (4), as a generalized transition system following Def. 1:

$$\mathcal{S} = (\mathbb{R}^n, \mathbb{R}^n, \mathcal{E}, \mathcal{Y}, H), \text{ where}$$
$$\mathcal{E} = \{(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^n \times \mathbb{R}^n \mid \boldsymbol{x}' = \boldsymbol{M}(\kappa(\boldsymbol{x}))\boldsymbol{x}\}$$
$$\mathcal{Y} = \{1, 2, ..., \bar{k}\} \tag{6}$$
$$H = \kappa.$$

The first feature that we see by inspecting Eqs. (6) and (5) in view of the definition of a LimAvg-automaton is that the weight of a transition is in fact h times the output of its outbound state. Hence, for any run r of $\mathcal{S}$, it holds that $\gamma(r) = h \cdot B_{\mathcal{S}}(r)$; that is, we can characterize weight sequences, hence run values, exclusively by external behaviors. Considering the notion of behavioral inclusion, this gives a straightforward result:

PROPOSITION 1. *Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\mathcal{Y}_a = \mathcal{Y}_b \subset \mathbb{Q}$. Attribute to each system the weight function $\gamma_s(x_s, x'_s) \equiv H_s(x_s)$, where $s \in \{a, b\}$. If $\mathcal{S}_a \preceq_{\mathcal{B}} (\cong_{\mathcal{B}}) \mathcal{S}_b$, then $V(\mathcal{S}_a) \geq (=) V(\mathcal{S}_b)$.*

PROOF. By definition, $V(\mathcal{S}_s) = \inf\{\text{LimAvg}(\gamma_s(r)) \mid r \text{ is a run of } \mathcal{S}_s\} = \inf\{\text{LimAvg}(y) \mid y \in \mathcal{B}^\omega(\mathcal{S}_s)\}$. Since $\mathcal{B}^\omega(\mathcal{S}_a) \subseteq (=) \mathcal{B}^\omega(\mathcal{S}_b)$, the desired result follows. □

Proposition 1 hints that obtaining a finite-state (bi)simulation of Eq. 6 provides means to compute a lower bound (or the actual value) for the MAIST. This is promising, because works in [7, 11] provides methods to find simulations of a PETC traffic. However, on the one hand, a simulation alone does not provide how conservative the lower bound may be; on the other hand, a finite-state bisimulation of an infinite system is often impossible to be obtained. In fact, bisimulation is too strong, in the sense that all behaviors and their fragments are exactly captured. As hinted by Theorem 2, the LimAvg value is determined by a minimum average cycle of the system. If one such cycle happens to have a correspondence with the concrete system, this is sufficient to obtain the exact value for the MAIST. For the rest of this paper, we assume every transition system is equipped with the weight function equal to the outbound state output, i.e., $\gamma(x, x') \equiv H(x)$. Let us proceed with formalities.

DEFINITION 6 (MINIMUM-AVERAGE-CYCLE-EQUIVALENT SIMULA-TION). *Consider two transition systems $\mathcal{S}_a$ and $\mathcal{S}_b$ satisfying $\mathcal{S}_a \preceq \mathcal{S}_b$. Denote by $\text{MAC}(\mathcal{S}_b)$ the set of minimum average cycles of $\mathcal{S}_b$. If $dc^\omega \in \mathcal{B}^\omega(\mathcal{S}_a)$ for some finite-length sequence d and some $c \in \text{MAC}(\mathcal{S}_b)$, then $\mathcal{S}_b$ is a minimum-average-cycle-equivalent (MACE) simulation of $\mathcal{S}_a$.*

A MACE simulation is a standard simulation with the additional requirement that one of the minimum average cycles of the abstraction must be observed on the concrete system, after possibly a finite number of transitions from the initial state. It should be clear that MACE simulation is stronger than simulation, but it is significantly weaker than bisimulation. The following result is a straightforward conclusion from Proposition 1 and Theorem 2.

PROPOSITION 2. *Let $\mathcal{S}_b$ be a finite-state MACE simulation of $\mathcal{S}_a$; then, $V(\mathcal{S}_a) = V(\mathcal{S}_b)$.*

PROOF. From Def. 6, take $dc^\omega \in \mathcal{B}^\omega(\mathcal{S}_a)$ for some finite-length sequence d with $c = k_0 k_1 ... k_N \in \text{MAC}(\mathcal{S}_b)$. The associated LimAvg

is

$$v := \text{LimAvg}(dc^\omega) = \text{LimAvg}(c^\omega) = \frac{1}{N} \sum_1^N k_i.$$

From Theorem 2, $v = V(\mathcal{S}_b)$. As v is also the value of a behavior from $\mathcal{S}_a$, it holds that $v \geq V(\mathcal{S}_a)$. Since Prop. 1 gives that $V(\mathcal{S}_a) \geq V(\mathcal{S}_b)$, we have that

$$V(\mathcal{S}_b) = v \geq V(\mathcal{S}_a) \geq V(\mathcal{S}_b),$$

and thus $V(\mathcal{S}_b) = V(\mathcal{S}_a)$. □

4.1 MACE simulation of PETC traffic

The challenge now resides on obtaining a MACE simulation of the system (6). For this we need to be able to (i) build a finite-state simulation of the system; (ii) check if its minimum mean cycle exists in the actual system; if not, (iii) refine the simulation until the cycle breaks; and (iv) repeat the process. This method is essentially the same as the bisimulation algorithm from a quotient model, presented in [10], which was used for PETC in [11], but with a different stopping criterion. Therefore, let us recover the simulation relation in [11], with a simplification that suits our purpose:

DEFINITION 7 (INTER-SAMPLE SEQUENCE RELATION (ADAPTED FROM [11])). *Given a sequence length l, we denote by $\mathcal{R}_l \subseteq \mathcal{X} \times \mathcal{Y}^l$ the relation satisfying $(\boldsymbol{x}, k_1 k_2 ... k_l) \in \mathcal{R}_l$ if and only if*

$$\boldsymbol{x} \in Q_{k_1}, \tag{7a}$$
$$\boldsymbol{M}(k_1)\boldsymbol{x} \in Q_{k_2}, \tag{7b}$$
$$\boldsymbol{M}(k_2)\boldsymbol{M}(k_1)\boldsymbol{x} \in Q_{k_3}, \tag{7c}$$
$$\vdots$$
$$\boldsymbol{M}(k_{l-1})...\boldsymbol{M}(k_1)\boldsymbol{x} \in Q_{k_l}, \tag{7d}$$

where

$$Q_k := \mathcal{K}_k \setminus \left(\bigcap_{j=\underline{k}}^{k-1} \mathcal{K}_j\right) = \mathcal{K}_k \cap \bigcap_{j=1}^{k-1} \bar{\mathcal{K}}_j,$$
$$\mathcal{K}_k := \begin{cases} \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{x}^\mathsf{T} \boldsymbol{N}(k) \boldsymbol{x} > 0\}, & k < \bar{k}, \\ \mathbb{R}^{n_\mathsf{x}}, & k = \bar{k}. \end{cases} \tag{8}$$

Eq. (8), taken from [7], defines the sets Q_k, containing the states that trigger exactly with inter-sample time hk. Eq. (7) simply states that a state $\boldsymbol{x} \in \mathbb{R}^n$ is related to a state $k_1 k_2 ... k_l$ of the abstraction if the inter-sample time sequence that it generates for the next l samples is $hk_1, hk_2, ..., kh_l$. The simplification with respect to [11] is that, here, we are not concerned with the state reaching a ball around the origin, but only with the sequence of sampling times it generates.

REMARK 1. *Setting $l = 1$ gives a quotient state set of Eq. 6, while larger values of l can be seen as refinements using the bisimulation algorithm of [10]. This relation can also be seen as a method to construct the strongest l-complete approximation [12] of the system (6). As a consequence of [13, Corollary 2], for autonomous deterministic systems, both methods lead to the same abstraction. Hence, here we can use both terms interchangeably, but for purposes of exposition we will use the term l-complete in what follows.*

DEFINITION 8. *Given an integer $l \geq 1$, the l-complete PETC traffic model is the system $\mathcal{S}_l := (\mathcal{X}_l, \mathcal{X}_l, \mathcal{E}_l, \mathcal{Y}, H_l)$, with*
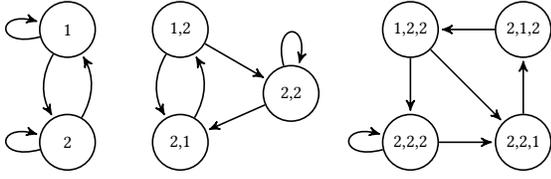
Figure 2: Example of l-complete PETC traffic models, for $l = 1$ (left), $l = 2$ (middle), and $l = 3$ (right).

- $\mathcal{X}_l \coloneqq \pi_{\mathcal{R}_l}(\mathcal{X})$,
- $\mathcal{E}_l = \{(k\sigma, \sigma k') \mid k, k' \in \mathcal{Y}, \sigma \in \mathcal{Y}^{l-1}, k\sigma, \sigma k' \in \mathcal{X}_l\}$,
- $H_l(k_1 k_2 ... k_m) = k_1$.

The model above partitions the state-space $\mathbb{R}^{n_x}$ of the PETC into subsets associated with the next l inter-sample times these states generate. Computing the state set requires solving the quadratic inequality satisfaction problems of Eq. (7), which can be done exactly using a satisfiability-modulo-theories (SMT) solver[3] such as Z3 [19], or approximately through convex relaxations as proposed in [7].[4] The output map is the next sample alone. The transition relation is simply what is called in [13] the domino rule: a state associated with a sequence $k_1 k_2 ... k_l$ must naturally lead to a state whose next first $l - 1$ samples are $k_2 k_3 ... k_l$, because the system is deterministic, autonomous, and time-invariant. Hence, any state in $\mathcal{X}_l$ that starts with $k_2 k_3 ... k_l$ is a possible successor of $k_1 k_2 ... k_l$. An example for $l = 1, 2, 3$ is depicted in Fig. 2.

The following fact is a direct consequence of Theorems 6 and 7 from [13], and gives the desired simulation refinement properties:

PROPOSITION 3. *Consider the system $\mathcal{S}$ from Eq. (6) and $\mathcal{S}_l$ from Definition 8, for some $l \geq 1$. Then, $\mathcal{S} \preceq \mathcal{S}_{l+1} \preceq \mathcal{S}_l$.*

Periodic set solutions of a PETC. The result above allows us to construct finite-state abstractions of system (6) with increased precision by increasing l; for each l, Karp's algorithm [17] (with the corrections from [18]) can be used to detect the mean average cycles. What is missing is a method to verify any such cycle in the concrete system. To this end, we need to confirm whether there is a subset of $\mathbb{R}^n$ that is a periodic solution with the desired pattern. Given that the recursion (4) is piecewise linear, and due to the homogeneity of the sets $\mathbb{Q}_k$ of Eq. (8), linear subspaces suffice to characterize these periodic solutions in the general case.[5]

DEFINITION 9 (PERIODIC SUBSPACE SOLUTION). *A linear subspace $\mathcal{A} \subseteq \mathbb{R}^{n_x}$ is a periodic subspace solution of Eq. (4) if there exists $J \in \mathbb{N}$ such that, for every $x \in \mathcal{A} \setminus \{0\}$, $\xi(t_i) = x \implies \xi(t_{i+J}) \in \mathcal{A}$.*

REMARK 2. *We remove the origin from the desired set solution (hence abusing the term linear subspace) because it is the only point along a ray that does not trigger like the others, due to the strict inequality in Eq. (2). Since controllers are typically designed for asymptotic stability, solutions starting away from the origin never reach*

---

[3]For that, the variable is $x \in \mathbb{R}^{n_x}$ and the query is $\exists x$:Eq. (7) holds.
[4]Using relaxations implies finding inter-sample sequences that may not be exhibited by the real system. This still generates a simulation relation, but not the strongest l-complete approximation.
[5]A set Q is homogeneous if $x \in Q \implies cx \in Q$, $\forall c \in \mathbb{R} \setminus \{0\}$. In the general case, every solution of a linear recurrence $\xi(k+1) = M\xi(k)$ converges asymptotically to an invariant linear subspace of M, hence verifying these invariants suffice. A formal proof is left out due to space limitations.

it. If $\xi(0) = 0$, it will always trigger at $\bar{k}$, so $\bar{k}^\omega$ is an obvious behavior, although very unlikely to be the minimum-in-average.

PROPOSITION 4. *A linear subspace $\mathcal{A}$ is a periodic subspace solution of Eq. (4) if there exists a sequence $k_1, k_2, ..., k_J$ such that (i) $\mathcal{A}$ is an invariant of $M(k_J)...M(k_2)M(k_1)$, and (ii)*

$$
\begin{aligned}
\mathcal{A} \setminus \{0\} &\subseteq Q_{k_1}, \\
M(k_1)\mathcal{A} \setminus \{0\} &\subseteq Q_{k_2}, \\
&\vdots \\
M(k_{J-1})...M(k_1)\mathcal{A} \setminus \{0\} &\subseteq Q_{k_J}.
\end{aligned}
\tag{9}
$$

PROOF. First, $\mathcal{A}$ must be a fixed set of Eq. (4) iterated J times, hence condition (i). Second, for a particular sequence $k_1, k_2, ..., k_J$, all points in this set must satisfy Eq. (7), which is what is displayed in Eq. (9). □

Proposition 4 shows how one can find if a sequence associated with the minimum average cycle of a similar model is indeed a periodic solution of the PETC system. First, determine the invariants of $M(k_J)...M(k_2)M(k_1)$, then, verify if they satisfy Eq. 9. For this latter part, first remember that each Q_k is an intersection of quadratic sets (see Eq. (8)). Then, the following simple result can be used to check if the linear space is a subset of a given quadratic set:

PROPOSITION 5. *Let $\mathcal{A}$ be a linear subspace with basis $v_1, v_2, ..., v_m$, and let V be the matrix composed of the vectors v_i as columns. Let $Q \in \mathbb{S}^n$ be a symmetric matrix and define $Q_n \coloneqq \{x \in \mathbb{R}^n \mid x^\mathsf{T} Q x \geq 0\}$ and $Q_s \coloneqq \{x \in \mathbb{R}^n \mid x^\mathsf{T} Q x > 0\}$. Then, $\mathcal{A} \setminus \{0\} \subseteq Q_n$ (resp. Q_s) if and only if $V^\mathsf{T} Q V \succeq 0$ (resp. $V^\mathsf{T} Q V \succ 0$).*

PROOF. For brevity, let us consider the strict inequality case (the other is analogous). First, note that $\mathcal{A} = \{Va \mid a \in \mathbb{R}^m\}$. Hence, if we want all points in $\mathcal{A}$ to belong to Q_s, we need that

$$\forall a \in \mathbb{R}^m \setminus \{0\}, \ a^\mathsf{T} V^\mathsf{T} Q V a > 0,$$

which is exactly the definition of $V^\mathsf{T} Q V \succ 0$. □

To have a finite collection of subspaces to be checked, we rely on the following assumption:

ASSUMPTION 1. *Let $k_1, k_2, ..., k_J \in \mathrm{MAC}(\mathcal{S}_l)$. Then, the matrix $M(k_J)...M(k_2)M(k_1)$ has no repeated eigenvalues.*

REMARK 3. *If Assumption 1 holds, then $M(k_J)...M(k_2)M(k_1)$ has at most 2^{n_x} real invariant subspaces, which are the linear combinations of its eigenvectors.[6]*

The MACE simulation algorithm. Algorithm 1 summarizes the method to obtain a MACE simulation of a given PETC system (1)–(2). In the outer loop, the relation $\mathcal{R}_l$ and corresponding finite-state system $\mathcal{S}_l$ are built, followed by the computation of one of its minimum average cycles. Then, an inner loop looks for linear subspaces associated with this cycle that satisfies Prop. 4; if one is found, the algorithm terminates. Otherwise, l is incremented and the main loop is repeated. An important remark is that, if a cycle σ^ω does not exist in the concrete system, after sufficiently many

---

Algorithm 1 MACE simulation algorithm

Input: $h, \mathcal{Y}, \ M(k), Q_k, \forall k \in \mathcal{Y}$
Output: $l, \mathcal{S}_l$, value, cycle, MAIST

1: $l \leftarrow 1$
2: **while** true **do**
3:     Build $\mathcal{R}_l$ and $\mathcal{S}_l$ ▷ (Defs. 7 and 8)
4:     value $\leftarrow V(\mathcal{S}_l)$, cycle $\leftarrow$ MAC$(\mathcal{S}_l)$ ▷ [17, 18]
5:     $\mathcal{V} \leftarrow$ eigenvecs$(M_{k_m}...M_{k_2}M_{k_1})$ ▷ $k_1 k_2...k_m =$ cycle
6:     **for** $\mathcal{V}' \in 2^{\mathcal{V}} \setminus \{\emptyset\}$ **do**
7:        $\mathcal{A} \leftarrow$ span$(\mathcal{V}')$
8:        **if** $\mathcal{A}$ satisfies Prop. 4 with $k_1, k_2, ..., k_m$ **then**
9:           MAIST $\leftarrow h \cdot$ value
10:           **return**
11:        **end if**
12:     **end for**
13:     $l \leftarrow l + 1$
14: **end while**

---

iterations it will not be exhibited in the abstraction, as for some N, σ^N will not be a sequence satisfying Eq. (7).

For an example, refer to Fig. 2, and assume the cycle $(1, 2, 2)^\omega$ is a trace of the concrete system. For $l = 1$, the only MAC is 1^ω, but it is not verified in the PETC. For $l = 2$, this cycle is broken, and the MAC becomes $(1, 2)^\omega$, again unverified. Finally, for $l = 3$, the MAC is $(1, 2, 2)^\omega$, which is verified and the algorithm terminates.

REMARK 4. *Algorithm 1 is in fact a semi-algorithm: it terminates if the concrete system exhibits a periodic trace whose LimAvg value is the smallest among all of its solutions. This is not always the case with PETC, as it may exhibit non-periodic traces. E.g., consider system* (1)–(4) *with* $\bar{k} = 2, M(1) = M(2) = \alpha \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}, N(1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. *Matrices* $M(k)$ *have eigenvalues equal to* $\alpha(1 \pm 2i)$, *which means that solutions rotate (up to rescaling) on the plane by the irrational angle* arctan(2). $N(1)$ *selects the first and third quadrants to output 1, whereas the others output 2. Identifying* $x \sim \lambda x, \ \forall \lambda \in \mathbb{R} \setminus \{0\}$, *this system is topologically equivalent to an irrational rotation on the circle, which does not exhibit periodic behavior.*

5 NUMERICAL EXAMPLE

Consider the system (1) with

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, K = \begin{bmatrix} 0 & -5 \end{bmatrix},$$

and the triggering condition of [2], $|\xi(t) - \hat{\xi}(t)| > \sigma|\xi(t)|$ for some $0 < \sigma < 1$, which can be put in the form Eq. (2). Checking time was set to $h = 0.05$, and maximum inter-sample time to $\bar{k} = 20$. We implemented Algorithm 1 in Python using Z3 [19] to solve Eq. (7), and attempted to compute its MAIST through a MACE simulation for $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Table 1 presents the MAIST for each σ, as well as the l value (Def. 8) where it was obtained. Only for $\sigma = 0.1$ the algorithm did not terminate before $l = 50$: for this case, the actual $\bar{k}$ of the system was 3, and all $M(k), k \leq 3$, have complex eigenvalues. Thus, it is likely that it does not have periodic

Table 1: MAIST values for the numerical example

σ	0.1	0.2	0.3	0.4	0.5
l	50*	15	26	12	10
MAIST	0.0786	0.137	0.171	0.25	0.3
CPU time [s]	327	41	147	29	45

* Algorithm interrupted before finding a verified cycle.

behaviors, similarly to what is discussed in Remark 4.[7] Nonetheless, the LimAvg value of the *maximum* average cycle of $\mathcal{S}_l$ is 0.0798 (after multiplying by h); since this is an upper bound for the MAIST (same arguments as Prop. 1), we know that the estimate is within only 0.0012 of the real value. For the other cases, trivial cycles were found for $\sigma = 0.4(5^\omega)$ and $\sigma = 0.5(6^\omega)$, but it took a few iterations to break, e.g., the 2^ω loop. Interestingly, the simplest cycles for $\sigma = 0.2$ and $\sigma = 0.3$ had length, respectively, 27 and 28, showing that PETC can often lead to very complex recurring patterns.

6 CONCLUSIONS

We have presented a method to compute the sampling performance of PETC, namely its minimum average inter-sample time, by means of abstractions. For that, we observed that limit average metrics of finite-state priced automata are (easily) computable; then, using behavioral inclusion properties of abstractions, we introduced the concept of minimum-average-cycle-equivalent simulations, together with a semi-algorithm that can compute the MAIST of linear PETC systems whenever their limit behaviors are periodic. When behaviors are aperiodic, the algorithm still provides good approximations, increasingly with higher values of l.

Future work is aimed at how to use these methods to design better triggering mechanisms particularly for sampling performance and schedulability. As we have noted in the previous section, computing the maximum AIST is also possible with the same tools; since a simulation model has embedded in it which states lead which cycles, one could consider modifying the sampling strategy for the states that lead to the MAIST, in order to steer them towards the system's most sample-efficient behaviors.

REFERENCES

[1] K. J. Åström and B. Bernhardsson, "Comparison of riemann and lebesgue sampling for first order stochastic systems," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, vol. 2, pp. 2011–2016, IEEE, 2002.
[2] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.
[3] X. Wang and M. D. Lemmon, "Event design in event-triggered feedback control systems," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 2105–2110, IEEE, 2008.
[4] A. Girard, "Dynamic triggering mechanisms for event-triggered control," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1992–1997, 2015.
[5] W. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 3270–3285, IEEE, 2012.

---

[6]The special case of repeated eigenvalues require some technicalities and is left out of this paper due to space considerations.

---

[7]It cannot be easily proven, however, whether there is no admissible product of such matrices with real eigenvalues.

[6] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 847–861, 2013.

[7] G. A. Gleizer and M. Mazo Jr., "Scalable traffic models for scheduling of linear periodic event-triggered controllers," *21st IFAC World Congress (accepted)*, 2020. https://arxiv.org/abs/2003.07642.

[8] R. Postoyan, R. G. Sanfelice, and W. P. M. H. Heemels, "Inter-event times analysis for planar linear event-triggered controlled systems," in *Decision and Control, 2019. CDC 2019. 58th IEEE Conference on*, 2019.

[9] A. S. Kolarijani and M. Mazo Jr, "A formal traffic characterization of LTI event-triggered control systems," *IEEE Transactions on Control of Network Systems*, 2016.

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

[11] G. A. Gleizer and M. Mazo Jr., "Towards traffic bisimulation of linear periodic event-triggered controllers," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 25–30, 2021.

[12] 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.

[13] A.-K. Schmuck, P. Tabuada, and J. Raisch, "Comparing asynchronous l-complete approximations and quotient based abstractions," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 6823–6829, IEEE, 2015.

[14] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Quantitative languages," *ACM Transactions on Computational Logic (TOCL)*, vol. 11, no. 4, pp. 1–38, 2010.

[15] K. J. Aström, "Event based control," in *Analysis and design of nonlinear control systems*, pp. 127–147, Springer, 2008.

[16] G. A. Gleizer and M. Mazo Jr., "Self-triggered output feedback control for perturbed linear systems," *IFAC-PapersOnLine*, vol. 51, no. 23, pp. 248–253, 2018.

[17] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete mathematics*, vol. 23, no. 3, pp. 309–311, 1978.

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

[19] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.