Control-driven Tasks: Modeling and Analysis*

Manel Velasco, Pau Martí
Technical University of Catalonia
Barcelona, Spain
{manel.velasco,pau.marti}@upc.edu

Enrico Bini
Scuola Superiore Sant'Anna
Pisa, Italy
e.bini@sssup.it

Abstract

The standard design of control systems is based on the periodic sampling. Every period the data is read from the input, the control law is computed, and the output is written to the actuators. However the periodicity of the sampling instants is a constraint that arises from the ease of implementation and it is not strictly necessary in the control system.

In this paper we present an execution model that samples the input "when needed". This model saves a considerable amount of computational resources. We show the schedulability analysis for a set of control-driven tasks using both Fixed Priority (FP) and Earliest Deadline First (EDF).

1 Introduction

In networked and embedded control systems, the most classic method used to read the input data of a controller is the *periodic sampling* [6]. In this case the controller is implemented by a task that is activated periodically. The implementation of controllers by periodic tasks has far become the standard method for making digital controllers. The reasons of this success are at least: the maturity of digital control theory, the ease of the implementation, and the simplicity of the schedulability analysis for these periodic tasks [26].

However the enforcement of a fixed separation between two consecutive activations can be inappropriate depending on the controller status. In some critical condition the controller may wish to sample more frequently. On the other hand, sometimes the samples could be less dense, allowing to save more computational resources.

The utility in breaking the constraint of periodic sampling is more relevant in severely limited computational resources such as in embedded systems. In fact, to make these systems cost-effective, it is mandatory to use efficiently the available computational resources [11]. Since the computational load imposed by controllers is proportional to their rate of execution, it is interesting to study the techniques to design controllers with low resource demands.

As indicated in the research literature [18], *event-driven* control systems provide interesting benefits like reducing resource utilization while providing similar control performance to the case of periodic controllers. In event-driven control systems (also known as control with adaptive sampling [14], control with Lebesgue sampling [5], interrupt-based control [19], self-triggered control [25], sporadic control [21], or state-triggered control [29]), the controller is activated upon some condition on the system status and not periodically. The condition, called *event condition* or *execution rule*, mandates to take a new control action when the measurement signal has deviated sufficiently from the desired set-point.

Unfortunately event-driven control lacks a mature system theory that prevents, for example, to estimate the computational load required by these controllers. Traditionally, event-driven approaches base their operation using some additional hardware to detect the events (e.g., analog event detector) that provokes the release of control jobs. Hence, jobs' release times are not known before system run-time, which makes schedulability analysis difficult.

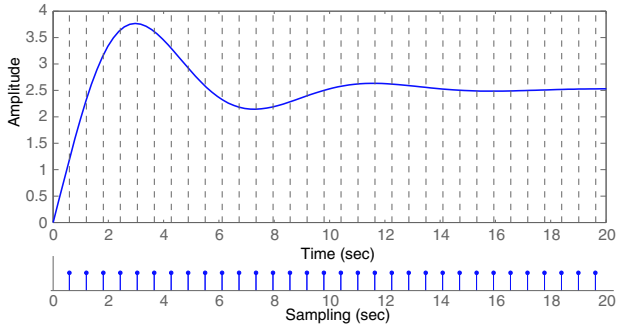
However, recent works [29, 25, 32, 3] on event-driven control approaches have focused not only on control aspects but also on real-time aspects. In particular, these works permit to derive timing properties that help characterizing when events will occur. Hence, they provide a priori knowledge on the type of sequence that event-driven control jobs will have at run time, and open the door to the problem of their schedulability analysis.

1.1 Intuitive problem formulation

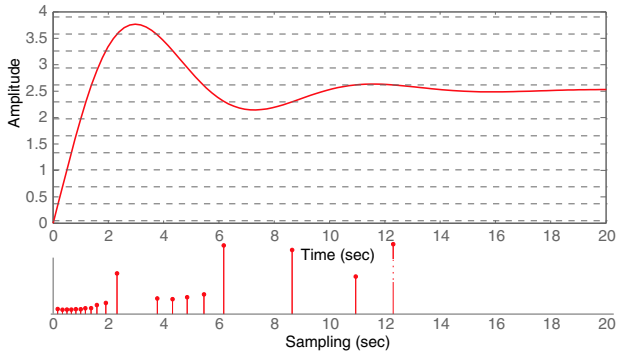
Control systems can be described by the evolution of the plant dynamics (or trajectory, response, etc.) over time. The trajectory is sometimes modeled in some space domain such

*This work was partially supported by C3DE CICYT DPI2007-61527 and by ArtistDesign NoE IST-2008-214373.

1052-8725/08 $25.00 © 2008 IEEE
DOI 10.1109/RTSS.2008.29



(a) Periodic control by discretization of the time domain



(b) Event-driven control by discretization of the space domain

Figure 1. Discretization for control systems.

as the state-space representation. Hence, control systems can be entirely described using two domains, the time and space domain.

Periodic control bases its operation by activating controller jobs according to a discretization in the time domain. Commonly, this discretization is regular, providing equidistant activation instants, i.e. constant sampling period. Figure 1-(a) illustrates this approach. In the figure we have plotted by a solid line the output of a simple linear system. The grid of vertical dashed lines represents the time discretization that marks the activation times a_i of a periodic controller. The corresponding activation pattern is reported below.

In this paper we show that the operation of various type of event-driven control approaches is based on activating controller jobs when the system trajectory crosses *boundaries* that define a discretization in the space domain, as illustrated in Figure 1-(b) by the horizontal dashed lines. These boundaries are the tolerated thresholds where the system trajectory can move without requiring control actions. Below we also draw the generated activation pattern. At each activation, the height of a line is equal to the distance to the next activation.

Looking at Figure 1-(b), many questions arise, for example: How thresholds/boundaries are defined? Is it required to have specific hardware for detecting that the trajectory reaches a boundary? What is the relation between boundaries and activation times? Is it possible to derive off-line the sequence of activation times? Given a sequence of activation times, how schedulability analysis can be performed? In the paper we will attempt to answer these questions.

1.2 Related work

Timing constraints for periodic controllers are derived in the controller design stage. Traditionally, the task period is given by the sampling period and the deadline is set to bound the input-output latency of the controller [4]. Schedulability analysis is then performed using standard techniques for periodic workloads.

Although early in the 60's [14] it was already noted the efficiency of event-driven control in terms of resource utilization, only the latest research results on this topic have covered real-time aspects.

In the approach by Johannesson *et al.* [21] it has been noted that tasks triggered by asynchronously generated events can not have guaranteed service unless a minimum inter-arrival time T is well defined. Hence, they presented the sporadic control approach where they imposed that samples can not be taken more often than T. However, schedulability issues are not addressed.

Lemmon *et al.* [25] presented the self-triggered controller approach where tasks periods are chosen using Lyapunov-based techniques to ensure robust control performance. The key idea is that at each job activation, the software tasks select themselves the next job release time according to a specific execution rule. Each next release time is then submitted to the Elastic scheduling algorithm [12], which is shown to allocate each new task utilization. Rather than imposing a minimum inter-arrival time T between consecutive controller jobs as in [21], Lemmon *et al.* provide evidences that for the defined execution rule this T exists.

Similarly, Tabuada [29] presents an event-driven approach using a similar execution rule. In addition, an accurate approximation on the minimum inter-arrival time between two consecutive control jobs is derived. This time is regarded as a lower bound for jobs release times and shown to be useful for schedulability analysis in the simple case of a workload composed by one higher priority event-driven control task sharing the processor with other periodic tasks. The drawback of this approach is that the operation of the event-driven control task requires dedicated hardware to test the defined error.

These approaches [25, 29] were further analyzed by Velasco *et al.* [31], where a graphical method to find lower and upper bounds for the timing sequences generated by these

type of event-driven control approaches was presented. Further, Velasco *et al.* showed that periodic sampled control systems is an special case of event-driven control systems.

Recently, Wang and Lemmon [32], and Anta and Tabuada [3] updated their previous approaches. Both updates reformulate the execution rule and derive approximations for the sequences of release and finishing times (or deadlines) that characterize the set of jobs for each event-driven control task. But no formal schedulability analysis is provided. However, their key contribution is that they removed the need for dedicated hardware for detecting the event condition. Instead, they have shown that detecting an analog event condition can be transformed into imposing an execution rule that sets next jobs timing constraints at each job execution.

These latest results provide the first steps towards understanding the scheduling requirements for specific event-driven control approaches. However, none of them has addressed in general terms the analysis of controller jobs activations and its integration into existing schedulability analysis as we propose in the work.

On a parallel track, the schedulability analysis has advanced significantly to capture many different kinds of task activation, and it has reached a maturity. Gresser [15] proposed an EDF analysis of a set of tasks whose activations are modeled by a sequence of interarrival times. Tindell *et al.* [30] included the burst activation task model in the FP analysis. Jeffay and Goddard [20] proposed the rate-based execution model, where it is specified the number of activation that can occur in an interval. Richter and Ernst [28] proposed to model the task activations by the maximum/minimum number of activations in any interval long Δt. Albers and Slomka [1] proposed both an exact and an approximated EDF analysis for this task model. Bini [10] reduced the set of scheduling points [24] for arbitrary activation patterns to allow a faster necessary and sufficient schedulability test. Finally, Chakraborty and Thiele [13] proposed to analyze an application modeled by the maximum time requirement using the Network Calculus [23].

1.3 Paper contributions

This paper permits to accommodate several existing event-driven control approaches in a unique framework. Specifically we contribute by:

- providing general explicit approximated solutions to compute activation times for event-driven control jobs;
- deriving the worst-case activation pattern for the control-driven tasks;
- extending both the FP and EDF schedulability analysis to the control-driven tasks.

2 Model of the Control Action

We study a system composed by m controllers, each one implemented by a control task τ_i. The controllers are activated based on events generated when some error threshold is reached. As we have remarked previously these events are not necessarily periodic. We call this model *event-driven control* task model (or *control-driven* task model) because at each activation the controller itself sets the next activation, based on some control-related events.

Since we will analyze the task model of only one controller, in the rest of this section we drop the index of the controller for better clarity.

The plant to be controlled is modeled by the following continuous-time linear control system

$$\begin{aligned} \dot{x} &= A\,x + B\,u \\ y &= C\,x \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^{n\times 1}$ represents the system state, $u \in \mathbb{R}^{m\times 1}$ is the input, $y \in \mathbb{R}$ denotes the system output, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{1\times n}$ describe the evolution of the system. Given the feedback matrix $L \in \mathbb{R}^{m\times n}$, let

$$u_i = L\,x(a_i) = L\,x_i \tag{2}$$

be the control updates computed from a sample at time a_i by a linear feedback controller designed in the continuous-time domain. Notice that we set $x_i = x(a_i)$ to denote in short the system state at the sampling time a_i.

In periodic sampling we have $a_{i+1} = a_i + T$, where T is the period of the controller. However in control-driven sampling the activations of the controller occurs at a sequence

$$\{a_i\}_{i\in\mathbb{N}} \tag{3}$$

that is not necessarily periodic. In Section 3.1 we review several possible events that can trigger the activation of the controller. Each time an event occurs, the controller samples the state, computes the control update, and applies it to the plant.

Between two consecutive control updates, the control signal is held constant, meaning that

$$\forall i \in \mathbb{N} \quad \forall t \in [a_i, a_{i+1}[\qquad u(t) = u_i. \tag{4}$$

The system trajectory, that is the system state as a function of time, after the sampling time a_i, evolves according to the dynamics [6]

$$\forall t \geq a_i \quad x(t) = (\Phi(t-a_i) + \Gamma(t-a_i)L)x_i \tag{5}$$

where $\Phi : \mathbb{R} \to \mathbb{R}^{n\times n}$ and $\Gamma : \mathbb{R} \to \mathbb{R}^{n\times m}$, that denotes the system and input matrices, are defined by

$$\Phi(t) = e^{At} \quad \text{and} \quad \Gamma(t) = \int_0^t e^{As}\mathrm{d}s B. \tag{6}$$

It can be noticed that according to (5) the state at the sampling instant $x(a_i)$ is, as expected, x_i since $\Phi(0) = I$ and $\Gamma(0) = 0$.

Note that the adopted control model of (1)–(2) assumes that the time elapsed from sampling the state to updating the control signal is negligible, that is, sampling and actuation occur at the same i^{th} time instant. In terms of timing constraints, each controller job activation and finishing time occur at the same time instant.

This assumption is taken for the sake of clarity in the exposition of the theoretical results. Nonetheless, Albertos and Crespo [2] have proposed a technique that allows to design the controller within this hypothesis. In short, all activation times can be advanced in time to give room to computation times, and then the state at the actuation times can be reconstructed from the advanced samples using an advanced observer (predictor). For further details on this technique, the interested reader is referred to [27], where its implementation and evaluation on top of a real-time kernel is presented.

3 The Control-driven Task Model

The control-driven task τ has execution time C and a deadline D that is set equal to the minimum interarrival time. The peculiarity of the control-driven task is that its jobs are activated based on an *execution rule* that triggers the execution when some control-related event is verified.

Below we show a formalization of these events.

3.1 Event Conditions

The event conditions specify the rule that triggers the execution of controller jobs. Generally speaking, the execution rule ensures that the system state does not deviate significantly from the desired/expected value, i.e. a given error is tolerated from the sampled state.

Let

$$e_i(t) = x(t) - x_i \tag{7}$$

be the error evolution between consecutive samples with $t \in [a_i, a_{i+1}[$.

For several types of event-driven control approaches, event conditions can be generalized by introducing a generic function $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ that measures the magnitude of the error. f must have the following properties:

1. $f(e, x)$ is a continuous function;
2. $f(e, x) \geq 0$ for all the errors $e \in \mathbb{R}^n$ and system states $x \in \mathbb{R}^n$;
3. $f(e, x) = 0 \;\Leftrightarrow\; e = 0$;

Once such a function is provided, the condition that must be ensured during the controller life time is

$$f(e_i(t), x_i) \leq \eta \tag{8}$$

where $\eta \in]0, 1]$ is the error tolerance.

Since we require that the condition of (8) holds during the complete lifetime of the controller, it is quite natural to set the next sampling instant a_{i+1} equal to the first instant when the state $x_{i+1} = x(a_{i+1})$ reaches the boundary of (8). This observation is condensed in the following Proposition.

Proposition 1 *If we set*

$$a_{i+1} = \min\{t \in \mathbb{R} : t \geq a_i,\ f(x(t) - x_i, x_i) = \eta\} \tag{9}$$

then we have

$$\forall i \in \mathbb{N} \quad \forall t \in [a_i, a_{i+1}[\quad f(x(t) - x_i, x_i) \leq \eta \tag{10}$$

Proof We prove it by contradiction.

First we observe that $f(x(t) - x_i, x_i)$ is continuous in t, because it is a composition of f, that is continuous by definition, and $x(t)$, that is also continuous because it represents the state trajectory (see (5)). Suppose it exists $\underline{i} \in \mathbb{N}$ and $\underline{t} \in [a_{\underline{i}}, a_{\underline{i}+1}[$ such that

$$f(x(\underline{t}) - x_{\underline{i}}, x_{\underline{i}}) > \eta$$

Since $f(x(a_{\underline{i}}) - x_{\underline{i}}, x_{\underline{i}}) = 0$ and f is a continuous function of t, for the intermediate value theorem it exists $t^* \in [a_{\underline{i}}, \underline{t}]$ such that

$$f(x(t^*) - x_{\underline{i}}, x_{\underline{i}}) = \eta$$

and this contradicts the minimality of $a_{\underline{i}+1}$ since $t^* \leq \underline{t} < a_{\underline{i}+1}$. ∎

Proposition 1 allows the selection of the next controller activation a_{i+1} such that the condition of (8) is always guaranteed. In Section A we show an explicit solution of the (9).

We stress that the execution rule set in (9) can capture the events of many real-world applications.

In some cases the execution rules are imposed to restrict the desired system dynamics, e.g. [21, 3, 32]. In some other cases it can also be intrinsic to the nature of the control setup, such as the measurement method, e.g. [17]. Many of them, although were initially developed making use of analog event detectors, can be transformed in such a way that the execution rule is expressed by (9). Therefore, the approach presented in this paper is not bounded to a single event-driven approach. It rather covers a wide range of existing results.

Although it is out of the scope of this paper to explicitly show that the rule of (9) can be used to model several types of execution rules, we examine in greater detail two possible cases for the sake of clarity. For example, the event

condition defined in the self-triggered scheme [32], reported below

$$\forall i \in \mathbb{N} \quad \forall t \in [a_i, a_{i+1}[\quad \frac{e_i(t)^T M e_i(t)}{x_i^T M x_i} \leq \eta, \qquad (11)$$

is a special case of (8).

Another significant example is provided by the encoders, position sensor widely used in many control applications. Encoders are discrete sensors that emit pulses whose frequency depends on the speed [17]. If control updates are triggered at each emitted pulse, the event condition can be defined as

$$Sx(t) = Sx_i + \eta\Delta^* \qquad (12)$$

where $\Delta^* \in \{-\Delta, 0, \Delta\}$ with Δ being the encoder resolution, and $S = [0, \ldots, 1 \ldots, 0]$ selects the k^{th} component of the state vector, which is relative to the position. If we rewrite (12) as

$$\frac{Se(t)}{\Delta^*} = \eta \qquad (13)$$

then it is clear that this condition can be seen as well in the framework of (8).

Note for the readers Starting from the next section, we present some examples. The Matlab code of these examples is available at `http://www.upcnet.es/~pmc16/`. Throughout the paper we specify the file that contains the corresponding code.

3.2 Comparison with periodic controller

The objective of presenting this example (available in `periodic_vs_event.m`) is twofold. First, it serves to show that an event-driven controller can provide the same control performance as a periodic controller while requiring less resources. Second, it illustrates the activation pattern of the controller jobs for which we provide schedulability analysis.

Let us consider the ball and beam system [6]

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x, \end{aligned} \qquad (14)$$

with state variables x and $\dot{x}$.

The periodic controller is designed using optimal control, that is, it solves the standard optimization problem [6] where the cost to be minimized is

$$\int_0^\infty x^T Q x + u^T R u + 2x^T N u \, \mathrm{d}t. \qquad (15)$$

The weighting matrices Q, N, and R are selected as follows

$$Q = \begin{bmatrix} 2.6964 & -0.3092 \\ -0.3092 & 0.0368 \end{bmatrix}, \quad N = \begin{bmatrix} 0.0030 \\ 0.00166 \end{bmatrix},$$
$$R = \begin{bmatrix} 0.0011 \end{bmatrix}.$$

By setting the sampling period to 0.05 seconds, the resulting controller gain is

$$L = \begin{bmatrix} 37.5468 & 9.7134 \end{bmatrix}.$$

The control-driven controller, with gain

$$L = \begin{bmatrix} 25.0897 & 6.8038 \end{bmatrix},$$

is heuristically designed according to the model given in Section 2 with an event condition specified by the function

$$f(e, x) = \frac{e^T Q_1 e}{x^T Q_2 x}$$

¿From Proposition 1 and the definition of f, it follows that the sequence of sampled states must obey to the equation

$$[x_{i+1} - x_i]^T Q_1 [x_{i+1} - x_i] = \eta x_i^T Q_2 x_i \qquad (16)$$

and the sampling instants a_i must sample the state, such that the sequence of states $\{x_i\}_{i\in\mathbb{N}}$ is in accordance to (16). We select

$$Q_1 = \begin{bmatrix} 1.4589 & 0.0001 \\ 0.0001 & 0.0097 \end{bmatrix}, \; Q_2 = \begin{bmatrix} 4.8051 & 0.7463 \\ 0.7463 & 0.4328 \end{bmatrix}$$

and $\eta = 0.02$ in order to reproduce a similar trajectory of the periodic controller.

In Figure 2 we illustrate the closed-loop system trajectory of the two controllers in the $(x, \dot{x})$ plane, as well as, the operation of execution rules.

In the figure, the trajectory for the case of the periodic controller is shown by a dashed line, and the trajectory for the case of the control-driven controller is shown by a solid line. From the initial state, $x_0 = [-100 \;\; 200]^T$, both trajectories tend toward the equilibrium point $[0 \;\; 0]^T$ exhibiting a very similar dynamics. Surprisingly, measuring the control performance archived by both controllers in terms of (15), the control-driven controller gives a slightly better performance than the "optimal" periodic controller.

On both trajectories we mark the job activations by *crosses* (for the periodic controller) and by *dots* (for the control-driven controller). That is, crosses correspond to the periodic activations of the optimal controller, while dots correspond to the aperiodic activations of the control-driven jobs, whose timing is partially shown in Figure 3. For the case of the control-driven controller, at each job execution we also draw the boundary of the admissible variation of the state, represented by a circle. If fact, each of the arrows that starts on a dot, mark the boundary generated by

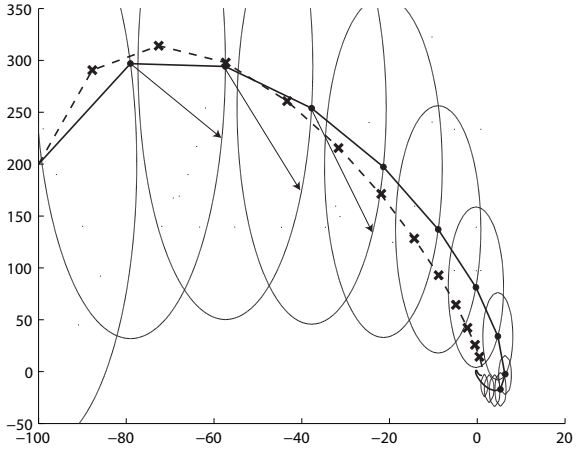


Figure 2. State dynamics for the ball and beam system.

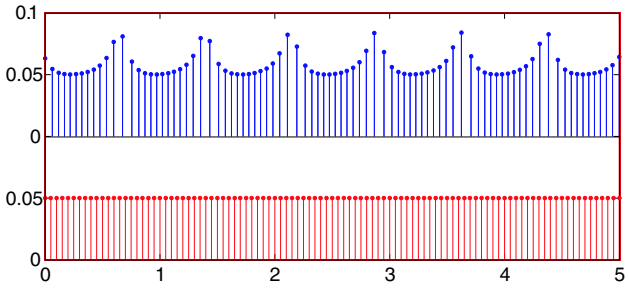


Figure 3. Activation times for the control-driven and periodic controller

the sampled state at that job activation. Looking only at the trajectory achieved by the control-driven controller, as the state moves along this trajectory and crosses the boundary, a new job is activated and a new sample is taken, according to Proposition 1.

But more interesting, the processor demand of the event-driven controller is less than the periodic controller. Figure 3 shows the pattern of the jobs' activation for both controllers. The x-axis is simulation time, and the y-axis is the sampling interval also in seconds. Each job activation time is represented by a vertical line, whose height indicates the next job release time.

As it can be seen in Figure 3, all sampling intervals for the event-driven controller are above 0.05s, which means that the control-driven controller has a varying rate of execution that demands less resources than the periodic controller executing at a constant rate given by the sampling period of 0.05s. In fact, the control-driven controller has an average period (concept that will be formally defined in Section 3.4) of 0.059s. It is also interesting to observe in Figure 3 the pattern of job activations. In the next example we will show more examples of activation patterns.

Comparing four controllers Now we present four controllers, that we call simply τ_1, τ_2, τ_3, and τ_4, that will be used in the paper to illustrate different aspects (available in the file `activation_patterns.m`). All the controllers have been designed to control the ball and beam system described by (14). The event condition is described by the quadratic function f shown in (11), as suggested by Wang and Lemmon [32]. The quadratic function f of the controller τ_i is defined by the weighting matrices M_i:

$$M_{1,2,3} = \begin{bmatrix} 1.9902 & 1.7323 \\ 1.7323 & 3.9904 \end{bmatrix} \quad M_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The first and second controller have an heuristic gain

$$L_{1,2} = \begin{bmatrix} 5 & 2 \end{bmatrix}$$

designed by pole placement [6] with desired poles at $-1 \pm 2i$, such that the closed loop has complex eigen vectors. This property, combined with the execution rule, produces an oscillating pattern for activation sequence $\{a_i\}_{i\in\mathbb{N}}$. In the controller τ_1 we set an error tolerance $\eta_1 = 0.01$, whereas the controller τ_2 we set $\eta_2 = 0.05$.

The third controller τ_3 has a gain of

$$L_3 = \begin{bmatrix} -1.001 & -1.7322 \end{bmatrix},$$

and it is designed with $\eta_3 = 0.01$ using robust control techniques [32], which grants exponential convergence of the system's state.

Finally the controller τ_4 has a gain equal to

$$L_4 = \begin{bmatrix} -2 & -3 \end{bmatrix}.$$

To further illustrate activation times, Figure 4 shows the activation times for the four controllers. It can be noticed that τ_1 and τ_2 has similar pattern, although τ_1 pattern is more dense due to a smaller value of error tolerance η. Also the third pattern oscillates although with a much longer period. Finally, the last pattern becomes stable after an initial transient.

3.3 The worst-case activation pattern

To perform the schedulability analysis of a set of control-driven tasks, it is necessary to estimate the maximum amount of computational resource required by each task τ. When all the jobs execute for the same amount of time, the scenario of maximum computing requirement occurs when

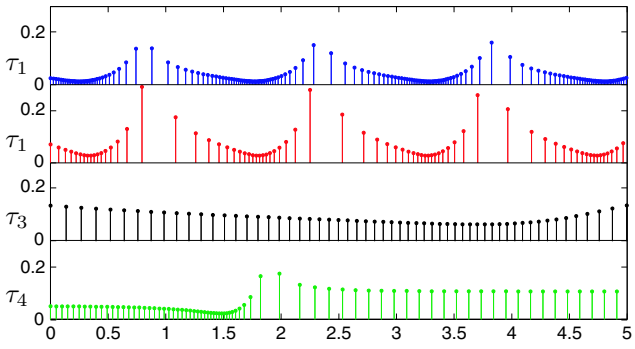


Figure 4. Examples of activation patterns.

the activations are as dense as possible [15, 23, 28, 1]. In this case it is necessary to find the length of the shortest interval I^k that can accommodate $k+1$ consecutive activations, formally defined as:

$$I^k = \min\{I \in \mathbb{R} : [t_0, t_0 + I] \text{ contains } k+1 \text{ activations}\} \tag{17}$$

For example, in the sporadic task model with minimum interarrival time T between two consecutive activations, we would simply find $I^k = kT$. If we denote the activations by a_i then we have

$$I^k = \min_{x_0}\{a_k - a_0\} \tag{18}$$

where the minimum is performed on all the possible initial states x_0.

In the rest of the paper we will call the sequence $\{I^k\}_{k\in\mathbb{N}}$ the *worst-case activation pattern* of the task τ. We highlight that the worst-case activation pattern is a sequence of interval lengths, not a sequence of activation times.

In the control-driven task model the separation between two consecutive activations depends also on the system dynamic, as explained in Section 3.1. As indicated by Proposition 1, from an activation that occurs at time a_i the next activation a_{i+1} is such that the next sampled state x_{i+1} lays on the boundary specified by the admitted error value η. Nonetheless the state evolution obeys to the law of (5). Hence we can assert that I^k is the solution of the following minimization problem

$$I^k = \min\ a_k - a_0 \tag{19}$$

$$\text{subject to } x_{i+1} = G(a_{i+1} - a_i)x_i \quad i = 0, \ldots, k-1 \tag{20}$$

$$f(x_{i+1} - x_i, x_i) = \eta \quad i = 0, \ldots, k-1 \tag{21}$$

$$a_0 = 0 \tag{22}$$

$$a_{i+1} \geq a_i \quad i = 0, \ldots, k-1 \tag{23}$$

where

- Equation (19) states that our goal is to minimize the activation of the k^{th} job, as required by the definition of I^k;
- Equation (20) describes the evolution of the system state as we have seen in Section 2. For compactness we have introduced $G : \mathbb{R} \to \mathbb{R}^{n\times n}$, defined as

  $$G(t) = e^{At} + \int_0^t e^{As}\mathrm{d}s\, BL$$

  Notice that $G(0) = I$ (the identity matrix);
- Equation (21) is the relationship between x_{i+1} and x_i, describing the condition of crossing the boundary of error;
- Equation (22) sets the initial activation equal to 0 without loss of generality;
- Equation (23) states simply that the activation a_{i+1} must follow activation a_i. This condition is necessary because, as it can be noticed in Figure 2, the state evolution crosses the error boundary also once backward in time.

  It can also be noticed that the inequality must hold strictly, otherwise from (20) it follows $x_{i+1} = x_i$, since $G(0) = I$. However in this case (21) would become $f(0, x_i) = \eta$ that is always false because $f(0, x_i) = 0$ and $\eta > 0$.

We highlight that the previous constrained minimization problem has $n(k+1)$ variables for the states ($k+1$ states in $\mathbb{R}^n$) plus $k+1$ variables for the activation instants. The constraints instead are $nk + k + 1$. Hence the free variables are n. A very interesting interpretation is achieved if we imagine the n-dimensional state x_0 as the free variable. In this case the state x_0 that solves the problem of (19)– (23) will be the initial state that generates the most dense activations $\{a_0, \ldots, a_k\}$. We label this special initial state by $\underline{x}_0^k$.

Comparing four controllers Now we use the four controllers of the ball and beam system introduced earlier to show their worst-case activation pattern (available in the file `worst_case_activation_patterns.m`). Figure 5 shows the four worst-case activation pattern on the controllers introduced earlier. In the figure, for each of the four patterns we draw the k^{th} vertical line at the position I^k. The height of the line is equal to the separation $I^{k+1} - I^k$.

Now we spend some words on the differences between the activation pattern (shown in Figure 4) and the **worst-case** activation pattern (Figure 5). In first case we are plotting the sequence of activations $\{a_i\}_{i\in\mathbb{N}}$ that we will experience, starting from some initial state x_0. Hence we have a different activation pattern for every possible starting state. In the second case, the worst-case activation pattern is the sequence $\{I^k\}_{k\in\mathbb{N}}$, where I^k is the solution of the (19)–(23). In this case the worst-case activation pattern does **not**

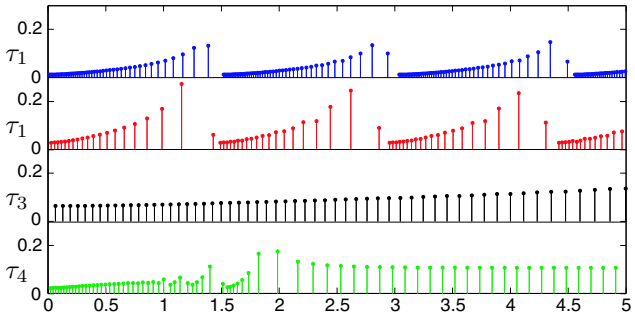


Figure 5. Example of worst-case activation patterns.

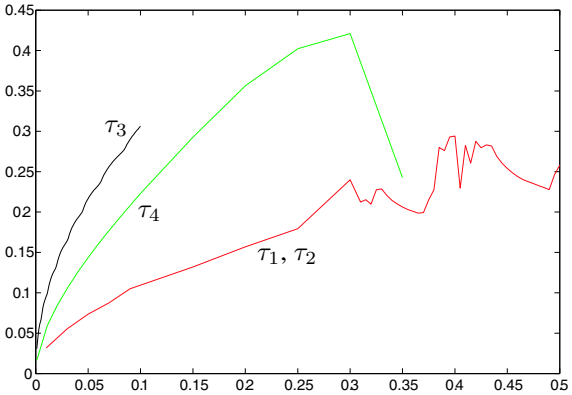


Figure 6. Average period as η varies.

depend on the initial state, because for each I^k the minimization problem search for the initial state that makes it minimum.

3.4 Asymptotic Behavior

The characterization of the controller activations by means of the worst-case activation pattern $\{I^k\}_{k\in\mathbb{N}}$ is an elegant method that allows to perform the schedulability analysis, as it will be shown in Section 4. However this representation has a clear disadvantage: it is unfeasible to compute I^k for all k. To overcome this limitation we introduce the *average period* that captures in one unique number an important feature of the worst-case activation pattern. Formally, we give the following definition.

Definition 1 *Given a task whose worst-case activation pattern is $\{I^k\}_{k\in\mathbb{N}}$, we define the* average period T *the following:*

$$T = \lim_{k\to+\infty} \frac{I^k}{k} \tag{24}$$

When this limit exists and it is finite, it is possible to define also the task *utilization* by

$$U = \frac{C}{T} \tag{25}$$

where C is the computation time of the task. Similarly as in the periodic task case, the utilization represents the fraction of the CPU that is consumed by the task.

Comparing four controllers In this example (available in `asymptotic_behavior.m`), we show how the average period varies with the admitted tolerance to the error η. Since the two controllers τ_1 and τ_2 differs only for the value of η, they will have the same behavior in this experiment. In Figure 6 we plot the three average periods as a function of η.

In general, we can see that as we tolerate a larger amount of error (large values of η) the average period increases, as one would expect. Then, above some threshold value of η, the behavior of the average period T becomes irregular.

The explanation of this irregularity can be possibly argued from an accurate observation of Figure 2. A small value of η means that the size of the ellipse that sets the next sample is small. It this case, the trajectory of the state can be well approximated by a line in a small neighborhood of the current state x_i. On the other hand, when η is large, the first-order approximation is not possible and a large amount of the trajectory falls within the ellipse $f(x_{i+1} - x_i, x_i) \leq \eta$. In this case the next activation will respond to a chaotic behavior and the limit of (24) loses significance.

4 Schedulability Analysis

If a set of m control-driven tasks is running in the computing resource then the execution of tasks can interfere with each other. Hence it is necessary to develop a proper schedulability analysis that ensures that the deadlines are not missed.

Below we develop the schedulability analysis of control driven tasks for both FP and EDF.

4.1 FP Analysis

In FP analysis all the tasks must be characterized by the maximum time demand that is the maximum computing resource that can be required in any interval of length t. The time demand can then be used either in the Response Time Analysis [16, 22, 7] to find the task response time, or in the Time Demand Analysis [24] to check the feasibility directly without computing the response time.

In the control-driven task model, since all the jobs require the same amount of time C, the time demand is given

by $\mathsf{act}(t)C$, where

$$\mathsf{act}(t) = \max_{t_0}\{\text{number of activation of } \tau \text{ in } [t_0, t_0 + t)\} \tag{26}$$

is the maximum number of activations that can fit in an interval of length t. Notice that if τ is strictly periodic with period T, we would have $\mathsf{act}(t) = \left\lceil \frac{t}{T} \right\rceil$.

Thanks to the introduction of the interval lengths I^k (see Section 3), we can easily define the number of activations as follows

$$\mathsf{act}(t) = k \quad \forall t \in (I^{k-1}, I^k] \tag{27}$$

Now we introduce again the index of the task τ_i in the notation for the number of activations $\mathsf{act}_i(t)$, the length of the intervals I_i^k, the computation time C_i, and the deadline D_i. We suppose that the tasks from τ_1 to τ_m are indexed by decreasing priority.

In the schedulability analysis of control-driven tasks scheduled by FP we must ensure that all the jobs complete before their corresponding deadline that has been set equal to the minimum separation between two activations I^1. This means we set the deadline $D_i = I_i^1$ for all the jobs of τ_i.

Hence, if we compute response time by the iterative formula

$$R_i = C_i + \sum_{j=1}^{i-1} \mathsf{act}_j(R_i) C_j \tag{28}$$

it is necessary and sufficient to test whether $R_i \leq D_i = I_i^1$ to guarantee that all the jobs of τ_i complete not later than the following activation.

On the other hand if we perform the schedulability test by the Time Demand Analysis, then we must ensure that [10]

$$\forall i = 1 \dots m \quad \exists t \in \mathcal{P}_{i-1}(I_i^1) \quad C_i + \sum_{j=1}^{i-1} \mathsf{act}_j(t) C_j \leq t \tag{29}$$

where $\mathcal{P}_i(t)$ is the set of scheduling points [24, 10] recursively defined by:

$$\begin{cases} \mathcal{P}_0(t) = \{t\} \\ \mathcal{P}_i(t) = \mathcal{P}_{i-1}\left(\max\{I_i^k : I_i^k \leq t\}\right) \cup \mathcal{P}_{i-1}(t). \end{cases} \tag{30}$$

4.2 EDF Analysis

In EDF schedulability analysis, each task must be characterized by its *demand bound function* [9, 8] that is the maximum amount of work that the task τ can require by consecutive jobs whose activation and deadline is within an interval long t. If we define $\mathsf{jobs}(t)$ as

$$\mathsf{jobs}(t) = \max_{t_0}\{\text{number of consecutive } \tau \text{ jobs whose activation and deadline is in } [t_0, t_0 + t]\} \tag{31}$$

then the demand bound function of the task τ easily becomes $\mathsf{jobs}(t)C$, because all the jobs require the same amount of computation C.

For computing the most dense number of jobs, we can take advantage of the interval lengths I^k that we found as solution of the problem in (19)–(23). In fact the sequence $\{I^k\}_{k\in\mathbb{N}}$ corresponds to the most dense sequence of activations. Moreover if we set the task deadline equal to the smallest separation between two activations (i.e. $D = I^1$), then the deadlines of the most dense activation pattern occur at the sequence $\{I^k + I^1\}_{k\in\mathbb{N}}$. It follows that the maximum number of jobs in an interval long t is

$$\mathsf{jobs}(t) = \begin{cases} 0 & \forall t \in [0, I^1) \\ k & \forall t \in [I^{k-1} + I^1, I^k + I^1),\ k \geq 1 \end{cases} \tag{32}$$

because the number of jobs makes a step at each deadline.

For the analysis of a set of m control-driven tasks we need to introduce the index i of the task τ_i in (32).

In this case a task set is schedulable by EDF if and only if [1, 13]:

$$\forall t \in \mathsf{dlSet} \quad \sum_{i=1}^{m} \mathsf{jobs}_i(t) C_i \leq t \tag{33}$$

where dlSet of a set of deadlines properly selected. If the total task set utilization $\sum_{i=1}^m U_i$ is smaller than 1 then the set of deadlines dlSet is finite. In fact, using the technique proposed by Baruah *et al.* [9] and Albers, Slomka [1], if the number of jobs can be upper estimated by a linear function as follows

$$\mathsf{jobs}_i(t) \leq b_i + \frac{t}{T_i} \tag{34}$$

then set of deadlines dlSet can be reduced to a finite set such that

$$\max \mathsf{dlSet} \geq \frac{\sum_i b_i C_i}{1 - \sum_i U_i}. \tag{35}$$

5 Conclusions and future work

The schedulability analysis of control-driven tasks opens a wide fields of exciting research in the field of both control systems and real-time systems.

For the control point of view it is interesting to investigate what is the minimum amount of error tolerance η_i that we allow for a set of controllers, given a fixed amount of computational resource. Another, indeed very challenging, problem to investigate is to design the control feedback that guarantees some desired value of performance minimizing the amount of computational resource.

For the schedulability point of view many problems remain open: is there any sufficient guarantee test that can be performed of the "utilizations" of the control-driven tasks?

What is the optimal priority assignment in control-driven tasks?

We believe that all these problems constitutes significant challenges in the field of real-time control systems.

References

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the* 16^{th} *Euromicro Conference on Real-Time Systems*, pages 187–195, Catania, Italy, June 2004.

[2] P. Albertos and A. Crespo. Real-time control of non-uniformly sampled systems. *Control Engineering Practice*, 7(4):445–458, Apr. 1999.

[3] A. Anta and P. Tabuada. Self-triggered stabilization of homogeneous control systems. In *Proceedings of the 2008 American Control Conference*, 2008.

[4] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *Proceedings of the* 39^{th} *IEEE Conference on Decision and Control*, Dec. 2000.

[5] K. J. Åström and B. Bernhardsson. Systems with lebesgue sampling. In *Directions in Mathematical Systems Theory and Optimization*, volume 286 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[6] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems. Theory and Design*. Prentice Hall, third edition, 1997.

[7] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, Sept. 1993.

[8] S. K. Baruah, D. Chen, S. Gorinsky, and A. K. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, July 1999.

[9] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), U.S.A., Dec. 1990.

[10] E. Bini. *The Design Domain of Real-Time System*. PhD thesis, Scuola Superiore Sant'Anna, Pisa, Italy, Oct. 2004. available at `http://retis.sssup.it/~bini/thesis/`.

[11] G. Buttazzo. Research trends in real-time computing for embedded systems. *SIGBED Rev.*, 3(3):1–10, 2006.

[12] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51(3):289–302, Mar. 2002.

[13] S. Chakraborty and L. Thiele. A new task model for streaming applications and its schedulability analysis. In *Design, Automation and Test in Europe Conference and Exposition*, pages 486–491, Munich, Germany, Mar. 2005.

[14] R. Dorf, M. Farren, and C. Phillips. Adaptive sampling frequency for sampled-data control systems. *IRE Transactions on Automatic Control*, 7(1):38–47, Jan. 1962.

[15] K. Gresser. An event model for deadline verification of hard real-time systems. In *Proceedings of the* 5^{th} *Euromicro Workshop on Real-Time Systems*, pages 118–123, June 1993.

[16] P. K. Harter. Response times in level-structured systems. Technical Report CU-CS-269-84, Department of Computer Science, University of Colorado, USA, 1984.

[17] W. Heemels, R. Gorter, A. van Zijl, P. van den Bosch, S. Weiland, W. Hendrix, and M. Vonder. Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, 7:1467–1482, Dec. 1999.

[18] W. Heemels, J. Sandee, and P. van den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.

[19] D. Hristu-Varsakelis and P. Kumar. Interrupt-based feedback control over a shared communication medium. In *Proceedings of the* 41^{st} *IEEE Conference on Decision and Control*, volume 3, pages 3223–3228, Dec. 2002.

[20] K. Jeffay and S. Goddard. A theory of rate-based execution. In *Proceedings of the* 20^{th} *IEEE Real-Time Systems Symposium*, pages 304–314, Phoenix, AZ, USA, Dec. 1999.

[21] E. Johannesson, T. Henningsson, and A. Cervin. Sporadic control of first-order linear stochastic systems. In *Proceedings of the* 10^{th} *International Conference on Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 4416, Pisa, Italy, Apr. 2007. Springer-Verlag.

[22] M. Joseph and P. K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, Oct. 1986.

[23] J.-Y. Le Boudec and P. Thiran. *Network Calculus*, volume 2050 of *Lecture Notes in Computer Science*. Springer, 2001.

[24] J. P. Lehoczky, L. Sha, and Y. Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., Dec. 1989.

[25] M. Lemmon, T. Chantem, X. S. Hu, and M. Zyskowski. On self-triggered full information h-infinity controllers. In *Proceedings of the* 10^{th} *International Conference on Hybrib Systems: Computation and Control*, pages 371–384, Pisa, Italy, Apr. 2007.

[26] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, Jan. 1973.

[27] C. Lozoya, M. Velasco, and P. Martí. The one-shot task model for robust real-time embedded control systems. *IEEE Transactions on Industrial Informatics*, 4(3):164–174, Aug. 2008.

[28] K. Richter and R. Ernst. Event model interfaces for heterogeneous system analysis. In *Design, Automation and Test in Europe (DATE)*, pages 506–513, Paris, France, Mar. 2002.

[29] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, Sept. 2007.

[30] K. W. Tindell, A. Burns, and A. Wellings. An extendible approach for analysing fixed priority hard real-time tasks. *Journal of Real Time Systems*, 6(2):133–152, Mar. 1994.

[31] M. Velasco, P. Martí, and C. Lozoya. On the timing of discrete events in event-driven control systems. In *Proceedings of the* 11^{th} *International Conference on Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer-Verlag, Apr. 2008.

[32] X. Wang and M. Lemmon. State based self-triggered feedback control systems with L2 stability. In *Preprints of the 17th IFAC World Congress*, 2008.

A An explicit solution to the next activation

Determining the next activation a_{i+1} from (9) requires to find the zeros of a generic function f. That is, given the sampled state x_i at the instant a_i, then a_{i+1} is given by the smallest value that solves

$$f(x(t) - x_i, x_i) = \eta \tag{36}$$

Finding the minimum t that solves (36) is a challenging task because having $x(t)$ implies that matrix exponentials (recall (5)–(6)) will appear as input arguments of f, making the equation transcendent, and preventing so to have an analytical solution.

Fortunately the following Proposition gives the means for finding a generic approximate solution.

Proposition 2 *For any linear system (1)–(2) with execution rule given by (8), and for any given state $x_i = x(a_i)$, the next sampling instant a_{i+1} is the smallest positive zero of the function on t*

$$f(\Psi(t - a_i)(A + BL)x_i, x_i) - \eta = 0, \tag{37}$$

where $\Psi : \mathbb{R} \to \mathbb{R}^{n\times n}$ is defined as

$$\Psi(t) = \int_0^t e^{As}\mathrm{d}s. \tag{38}$$

Proof Equation (36) that can be re-arranged as

$$f(x(t) - x_i, x_i) - \eta = 0. \tag{39}$$

As suggested by Åström and Wittenmark [6], the matrices $\Phi(t)$ and $\Gamma(t)$ of the system evolution (see (5)) can be both expressed as function of $\Psi(t)$

$$\Phi(t) = e^{At} = I + A\Psi(t) \tag{40}$$

$$\Gamma(t) = \int_0^t e^{As}\mathrm{d}sB = \Psi(t)B \tag{41}$$

Substituting (40)–(41) into (5) we have

$$\begin{aligned} x(t) &= (\Phi(t - a_i) + \Gamma(t - a_i)L)x_i = \\ &= (I + A\Psi(t - a_i) + \Psi(t - a_i)BL)x_i = \\ &= (A\Psi(t - a_i) + \Psi(t - a_i)BL)x_i + x_i \end{aligned} \tag{42}$$

Since matrices $\Psi(t)$ and A commute, (42) can be rewritten as

$$x(t) - x_i = \Psi(t - a_i)(A + BL)x_i \tag{43}$$

By rewriting (39) using (43), we obtain (37), which is a function on t. Therefore, we will find the next sampling time by solving (37) for t and choosing the smallest positive zero. ■

Notice that finding the zeros of (37) is still challenging. However, since $\Psi(t)$ can be efficiently approximated by the following power series [6]

$$\Psi(t) = \sum_{k=1}^{\infty} \frac{A^{k-1}t^k}{k!} \tag{44}$$

then we can simplify the problem by finding the smallest zero t of (37), replacing $\Psi(t)$ by the n^{th} order Taylor series approximation.

Proposition 2 gives the first steps toward an explicit approximate solution for finding the next activation time given a generic f. Now we examine a special cases of f.

A typical choice for f is a quadratic function [32, 3] that is defined as follows

$$f(e, x) = \frac{e^T Q_1 e}{x^T Q_2 x}, \tag{45}$$

where $Q_1, Q_2 \in \mathbb{R}^{n\times n}$ are weighting matrices. In this case it is possible to determine an explicit solution of the next sampling instant a_{i+1} that solves (9).

Given the sampled state x_i at the instant a_i, when the error is quadratic for the problem of (9), then a_{i+1} is given by the smallest value that solves the following equation

$$\frac{(x(t) - x_i)^T Q_1 (x(t) - x_i)}{x_i^T Q_2 x_i} = \eta \tag{46}$$

Fortunately the following Proposition allows to find efficiently this value.

Proposition 3 *For any linear system (1)–(2) with execution rule of the form (45), and for any given state x_i, the next sampling instant is the smallest positive zero of the function on t*

$$\begin{aligned} \left[\Psi(t - a_i)(A + BL)x_i\right]^T Q_1 \left[\Psi(t - a_i)(A + BL)x_i\right] - \\ \eta(x_i^T Q_2 x_i) = 0, \end{aligned} \tag{47}$$

where $\Psi : \mathbb{R} \to \mathbb{R}^{n\times n}$ is defined in (38).

Proof Equation (46) that can be re-arranged as

$$(x(t-a_i)-x_i)^T Q_1 (x(t-a_i)-x_i) - \eta(x_i^T Q_2 x_i) = 0. \tag{48}$$

Rewriting (48) in terms of (43) we obtain (47) which is a function on t. Therefore, we will find the next sampling time by solving (47) for t and choosing the smallest positive zero. ■

Remark 1 *Note that finding the zeros of (47) is still a challenging task. However, by using an n^{th}-order approximation of Ψ in (47), equation (47) becomes a polynomial on t of degree $2n$, and the a_{i+1} will be the smallest positive root of the polynomial.*