An improved self-triggered implementation for linear controllers

J. Araújo [*], **H. Fawzi** [**] **M. Mazo Jr.** [***] **P. Tabuada** [**] **K. H. Johansson** [*]

[*] *ACCESS Linnaeus Center, KTH Royal Institute of Technology, Sweden*
[**] *University of California at Los Angeles, USA*
[***] *INCAS*^{3} *& University of Groningen, The Netherlands*

Abstract: Research in networked control systems raised the importance of understanding what are the timing requirements for control. In recent years this problem has been attacked from multiple angles including the computation of Maximal Allowable Transmission Intervals, event-triggered, and self-triggered controller implementations. In a self-triggered implementation the controller is responsible for computing the next time instant at which the actuator values should be updated by evaluating the control law on fresh sensor measurements. One of the main challenges in self-triggered control is how to perform the exact calculation of the time at which these updates should take place. In this paper we present a new technique to compute lower bounds on the self-triggered update times in a computationally light manner. We evaluate the algorithm on numerical examples and we observe that the algorithm performs well when compared to other existing methods and provides tight lower bounds to the exact update times. Additionally, we propose a Semidefinite Programming-based (SDP) technique that produces triggering conditions that are less conservative than the existing ones and for which the update times are larger.

1. INTRODUCTION

Recently, there has been a great interest in the use of wireless networks as the communication medium for control applications. In this scenario the communication medium is usually shared among several sensors, controllers and actuators. Furthermore, sensor nodes are typically powered by small batteries. Transmitting measurements too often, as it is done in today's digital control systems, can not only empty the energy reserves of battery-driven sensors in a matter of hours but complicates the shared usage of the communication network.

One of the most important aspects in a modern control system is to decide how often should sensors measure the physical system and transmit information to the control unit. A shift in perspective was brought by the introduction of a new control paradigm known as event-triggered control by Åström and Bernhardsson [1999], Årzén [1999]. Although these references pointed to the benefits of event-triggered control in specific examples, the first systematic approach to obtain event-triggered implementations of control laws only appeared in Tabuada [2007] and was then improved by many researchers Heemels et al. [2008], Cervin and Henningsson [2008], Lemmon [2011], Lunze and Lehmann [2010], Mazo Jr. and Tabuada [2011], Garcia and Antsaklis [2011], Dimarogonas et al. [2012]. While in a traditional sampled-data paradigm (Åström and Wittenmark [1990]) new controller updates are performed periodically, regardless of the state of the system, event-triggered control is based on events triggered when stability or a pre-specified control performance is about to be lost. This approach has proved to reduce the number of control-loop executions, while providing a high degree of robustness, since the system is permanently monitored. However, by requiring the continuous monitoring of a certain triggering condition, event-triggered controllers still require a large amount of energy consumption. To overcome this drawback, several researchers proposed the use of self-triggered control technique Velasco et al. [2003], Wang and Lemmon [2009], Anta and Tabuada [2010], Mazo Jr. et al. [2009]. The underlying idea is to emulate the event-triggered implementation and estimate the time at which the next event takes place using the knowledge of the system's dynamics. The sensor nodes are then scheduled for transmission at the expected triggering time. In between triggering times, sensor nodes can simply be in an idle mode thereby greatly reducing its energy consumption. However, this reduces the robustness of the system to external disturbances Mazo Jr. and Tabuada [2009]. One of the main challenges in self-triggered control is how to perform the exact calculation of the next update time.

The main contribution of this paper is a new algorithm to compute the next update time for a certain type of event-triggering conditions. This problem has been considered earlier in, e.g., Wang and Lemmon [2009], Anta and Tabuada [2010] and Mazo Jr. et al. [2009]. In Anta and Tabuada [2010] a general technique was developed to compute the inter-execution times for any nonlinear polynomial system. Since this technique focused on nonlinear systems, it becomes conservative when used for linear systems. The paper Mazo Jr. et al. [2009] proposes, in the context of linear systems, to simply integrate the dynamics using some discretization method in order to find the update time. This method has the drawback that the choice of discretization step greatly influences the complexity of the implementation. The method proposed by Wang and Lemmon [2009] computes a conservative lower bound of the next update time while providing performance guarantees in terms of the closed-loop system's induced $\mathcal{L}_2$ gain. The computed self-triggered update time is shown to be an order of magnitude smaller than the exact event time. Our goal is to propose a new method to compute the update times for self-triggered control of diagonalizable linear systems. The method builds upon an idea from Mazo Jr. and Tabuada [2011] and provides a computationally light manner to estimate the update times of event-triggered implementations.

An additional contribution of this paper is to design event-triggering conditions which provide larger inter-execution times than the technique proposed in Tabuada [2007], guaranteeing the same level of performance, and allowing the application of the self-triggered mechanism proposed in this paper. We propose a Semidefinite Programming-based (SDP) technique in order to accomplish this goal.

2. SELF-TRIGGERED CONTROL FOR LINEAR SYSTEMS

We consider networked linear control systems consisting of several wireless sensors and actuators and a central controller node. The controller node is responsible for computing the control signal based on the measurements transmitted by the sensor nodes. The *control-loop execution* is defined by: the transmission of measurements by the sensor nodes to the controller node, the computation of the control signal by the controller node and the posterior dissemination of the control signal to the

actuator nodes. All these actions are performed instantaneously. [1] Additionally, the controller node is responsible for computing the triggering times of the self-triggered control implementation and schedule the sensor nodes for future transmission.

We begin by revisiting event-triggered control of linear systems which serves as the basis for the self-triggered control scheme we propose.

2.1 Event-triggered control for linear systems

Consider a linear control system

$$\dot{x} = Ax + Bu, \;\; x \in \mathbb{R}^n, \; u \in \mathbb{R}^m. \tag{1}$$

If the A matrix is diagonalizable we can transform (1) into:

$$\dot{x'} = M^{-1}AMx' + M^{-1}Bu \tag{2}$$

where $x' = M^{-1}x$ and M is the eigenvector matrix of $\bar{A}$. Therefore, we directly assume system (1) to have a diagonal A matrix.

In a sample-and-hold implementation of the control signal u, the input signal is held constant between control loop executions, i.e., $u(t) = u(t_k)$, for $t \in [t_k, t_{k+1})$, where $t_1, t_2, t_3, ...$ is a sequence of update times at which a new measurement is transmitted to the controller and input signals are applied to the system. We define *inter-execution time* as $\Delta T = t_k - t_{k-1}$ representing the time between control-loop executions.

Let $u = -Kx$ be a stabilizing controller for the system (2) and let $V = x^T Px$ be a Lyapunov function for the corresponding closed-loop stable system. The matrix P is the solution of the Lyapunov equation $(A + BK)^T P + P(A + BK) = -Q$, where Q is a chosen symmetric positive definite matrix.

If $x(t_k)$ is the current state of the system at time $t = t_k$, we know that the input $u = -Kx(t_k)$ makes $\dot{V}(x(t_k)) < 0$, since $\dot{V}(x(t_k)) = -x(t_k)^T Q x(t_k) < 0$.

In order to guarantee asymptotic stability of the closed-loop system, in an event-triggered implementation of the controller $u = -Kx$, the input $u = u(t_k)$ is kept constant until condition:

$$\dot{V} < -x^T Sx \tag{3}$$

is violated for $S > 0$. The time at which the condition above is violated is denoted by t_{k+1}. The matrix S defines the desired rate of decay of the Lyapunov function V.

The expression (3) can be further represented as:

$$\begin{aligned} \dot{V} + x^T Sx &= \dot{x}^T Px + x^T P\dot{x} + x^T Sx \\ &= (Ax + Bu)^T Px + x^T P(Ax + Bu) + x^T Sx \\ &= x^T(A^T P + PA + S)x + 2x^T PBKx(t_k) \\ &= z^T \Phi z, \end{aligned} \tag{4}$$

where $z(t) := [x(t)^T \;\; x^T(t_k)]^T$ and

$$\Phi := \begin{pmatrix} A^T P + PA + S & PBK \\ K^T B^T P & 0 \end{pmatrix}.$$

With these notations, we have $z(t_k)^T \Phi z(t_k) < 0$ and the control-loop execution takes place as soon as $z(t)^T \Phi z(t) = 0$.

In Tabuada [2007], the author provides another triggering condition that also guarantees $z(t)^T \Phi z(t) < 0$ along the trajectory, but has the benefit of being simpler. The rule given by Tabuada [2007] is to update the control input as soon as

$$z(t)^T \widetilde{\Phi} z(t) \geq 0, \tag{5}$$

where

$$\widetilde{\Phi} = \begin{pmatrix} I - \sigma^2 I & -I \\ -I & I \end{pmatrix} \tag{6}$$

and σ is a parameter which depends on the choice of A, B, K, Q and S. The constant σ is chosen so that the following implication holds for any z:

$$z^T \Phi z \geq 0 \Rightarrow z^T \widetilde{\Phi} z \geq 0. \tag{7}$$

This guarantees that if we use the triggering rule given by $\widetilde{\Phi}$, we have $\dot{V} = z(t)^T \Phi z(t) < 0$ along the trajectory. Other matrices $\widetilde{\Phi}$ exist that satisfy the above implication for all z. Throughout the paper we will say that a triggering matrix $\widetilde{\Phi}$ is a *valid bound* to Φ if the implication (7) holds for all z.

2.2 Problem statement

The main problem we address in this paper is:

- How can we design an *efficient* method that computes the next event time for the simple triggering rule given by $\widetilde{\Phi}$ of Tabuada [2007]?

In fact, we solve the preceding problem not just for the triggering condition defined by $\widetilde{\Phi}$ but also for any quadratic triggering condition that has a nice separability structure. This method is presented in detail in Sec. 2.

In the second part of this paper, we propose an Semidefinite Programming-based (SDP) technique to find quadratic triggering conditions that best bound Φ and have the required separability properties to enable the application of the previous method. Thus, these techniques allow us to obtain larger inter-execution times of a self-triggered implementation that still guarantee $\dot{V} < -x^T Sx$.

Note that, in principle, computing the next event time t_{k+1} given by any quadratic triggering condition can always be done since a closed form expression for the evolution of the state is available. However, this is a computationally intensive task. Instead, we seek a computationally efficient method that could (ideally) be implementable on a microcontroller.

3. MAIN ALGORITHM

We now present our main algorithm to compute a lower bound on the triggering time t_{k+1}, which is the earliest time t for which

$$z^T \widetilde{\Phi} z \geq 0$$

is true. Our algorithm requires the control system (1) to be diagonalizable, and the triggering matrix $\widetilde{\Phi}$ to be a $2n \times 2n$ symmetric matrix for which the upper-left block is diagonal. This guarantees that the triggering condition is separable. In other words, if we let the *decision gap* G be [2] $G(x) = \begin{pmatrix} x \\ x(t_k) \end{pmatrix}^T \widetilde{\Phi} \begin{pmatrix} x \\ x(t_k) \end{pmatrix}$, then G has a separable structure: $G(x) = \sum_{i=1}^n G_i(x_i)$. This separability together with the fact that the system is diagonal (i.e., that $\dot{x}_i$ only depends on x_i and the input) allows for the efficient computation of the triggering time t_{k+1}.

3.1 An insight from Mazo Jr. and Tabuada [2011]

Our method uses an idea by Mazo Jr. and Tabuada [2011] proposed in the context of decentralized event-triggered control.

If we consider $\theta_1, ..., \theta_n \in \mathbb{R}$, such that $\sum_{i=1}^n \theta_i = 0$, we can rewrite the decision gap function $G(x)$ as

$$G(x) = \sum_{i=1}^{n} (G_i(x_i) - \theta_i). \tag{8}$$

[1] For clarity of presentation we assume that computation or transmission delays are inexistent. However, Tabuada [2007] has shown that the techniques presented here can accommodate delays by modifying the triggering conditions appropriately.

[2] Since $x(t_k)$ is fixed, we are only interested in the dependence of G on x.

The following implication then holds since the sum of non-positive quantities is still a non-positive quantity:

$$\forall i = 1, \dots, n \;\; (G_i(x_i) - \theta_i \leq 0) \Rightarrow G(x) \leq 0 \qquad (9)$$

Thus, if we define

$$t_i(\theta_i) := \min\{t \in \mathbb{R} : G_i(t) - \theta_i = 0, t > t_i\}, \qquad (10)$$

then we clearly have that

$$\min_{i=1,\dots,n} t_i(\theta_i) \leq t_{k+1}.$$

Note that this is true for any choice of θ such that $\sum_{i=1}^n \theta_i = 0$. Thus if we denote by T the quantity: $T(\theta) := \min_{i=1,\dots,n} t_i(\theta_i)$ we have that for any $\theta_1, \dots, \theta_n$ such that $\sum_{i=1}^n \theta_i = 0$,

$$T(\theta) \leq t_{k+1}.$$

Hence, in order to obtain a good lower bound on t_{k+1}, we need to find the value of θ for which $T(\theta)$ is maximal.

Observe that since the system is assumed to be diagonal, and the triggering matrix has a separable structure, one can compute explicitly the value of $T(\theta)$ for any given θ (closed form expressions are provided in Appendix A). We will rely on these formulas for the algorithm we present next.

Remark 3.1. Finding the value of θ for which $T(\theta)$ is maximal is a non-trivial problem that has no analytical solution, in general. In Mazo Jr. and Tabuada [2011] the authors proposed to equalize $G_i(x_i) - \theta_i = 0$, for all i in order to find θ. However, such approach does not take into account additional conditions that must be satisfied to correctly find θ and $T(\theta)$, as it will be shown in the next subsection. ◁

3.2 Finding the best value of θ

In this section we propose an iterative method to compute the value of θ that makes $T(\theta)$ maximal. The method starts with an initial value of θ, say $\theta^{(0)}$, and then keeps improving it at each iteration so that $T(\theta^{(0)}) < T(\theta^{(1)}) < T(\theta^{(2)}) < \dots$ where $\theta^{(\ell)}$ is the value of θ at iteration ℓ of the algorithm.

We now explain how one can improve the value of θ from one iteration to the next, i.e., how one can find $\theta^{(\ell+1)}$ such that $T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$. The update rule for $\theta_i^{(\ell)}$ will take the form

$$\theta_i^{(\ell+1)} = \theta_i^{(\ell)} - \delta_i$$

for some appropriately chosen δ_i's. We now show how to choose the values of δ_i's.

Observe that for a given $\theta^{(\ell)}$ we have, for any [3] i

$$G_i(T(\theta^\ell)) - \theta_i^{(\ell)} \leq 0 \qquad (11)$$

with equality for at least one subsystem i (this is the "bottleneck" subsystem for which $t_i(\theta_i^{(\ell)})$ is minimal, see Fig. 1).

We observe that if we can make the inequalities (11) *strict for all i* then we can make progress, in the sense that there exists $\theta^{(\ell+1)}$ such that $T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$.

Indeed assume that we can find $\delta_1, \dots, \delta_n$, $\delta_i \in \mathbb{R}$ such that :

$$\sum_{i=1}^n \delta_i = 0, \qquad (12)$$

and such that *for all i*:

$$G_i(t) - \theta_i^{(\ell)} + \delta_i < 0 \quad \forall t \in [t_k, T(\theta^{(\ell)})]. \qquad (13)$$

The important property about the δ's here is that the inequalities above are *strict* for all i. The coefficients δ_i correspond to a vertical displacement of the graphs of $G_i - \theta_i^{(\ell)}$, as depicted for example in Fig. 2. If such values of δ exist, then by defining $\theta_i^{(\ell+1)} = \theta_i^{(\ell)} - \delta_i$, we have that

$$\forall t \in [t_k, T(\theta^\ell)], \; G_i(t) - \theta_i^{(\ell+1)} < 0,$$

for all i, and so this means that *for all i*, $t_i(\theta_i^{(\ell+1)}) > T(\theta^{(\ell)})$ which in turn means that $T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$.

The question now remains of how to find values $\delta_1, \dots, \delta_n$ that satisfy conditions (12) and (13). Note that condition (13) on the δ_i's can be rewritten as

$$\max_{t \in [t_k, T(\theta^{(\ell)})]} G_i(t) - \theta_i^{(\ell)} + \delta_i < 0$$

or equivalently

$$\delta_i < \overline{\delta_i}$$

where $\overline{\delta_i} := \theta_i^{(\ell)} - \max_{t \in [t_k, T(\theta^{(\ell)}))} G_i(t) - \theta_i$. Since we assumed our system to be diagonal, the quantity $\overline{\delta_i}$ can be computed explicitly and the formulas are given in appendix A. The problem of finding δ_i's therefore corresponds to solving the following feasibility problem:

$$\text{find } \delta_1, \dots, \delta_n \text{ such that } \delta_i < \overline{\delta_i} \text{ and } \sum_{i=1}^n \delta_i = 0$$

If there are no solutions to this problem, we stop the algorithm. Otherwise, there can be many solutions to this feasibility problem, and so it would be preferable to find the "best" possible δ. For example one can introduce an objective function that makes the inequalities $G_i(T(\theta^{(\ell)})) - \theta_i^{(\ell)} + \delta_i < 0$ as strict as possible. This therefore yields the following optimization problem for the δ's:

$$\begin{aligned} \underset{\delta_1,\dots,\delta_n}{\text{minimize}} \;\; & \max_{i=1,\dots,n} G_i(T(\theta^{(\ell)})) - \theta_i^{(\ell)} + \delta_i \qquad (14) \\ \text{subject to} \;\; & \delta_i \leq \overline{\delta_i} \text{ for all } i = 1, \dots, n \\ & \sum_{i=1}^n \delta_i = 0 \end{aligned}$$

This optimization problem can be solved efficiently using a simple bisection algorithm. The details of the algorithm are given in Appendix B. The complete iterative algorithm outlined in this section to compute a value of θ such that $T(\theta)$ is as large as possible is given in Algorithm [4] 1.

From the previous discussion, the behavior of the algorithm can be summarized in following proposition:

Proposition 1. Consider a diagonal system $\dot{x} = Ax + Bu$ where A is diagonal, and a triggering matrix $\widetilde{\Phi} \in \mathbb{R}^{2n \times 2n}$ whose upper-left block is diagonal. Let $x(t_k)$ be the state at time t_k satisfying $\begin{pmatrix} x(t_k) \\ x(t_k) \end{pmatrix}^T \widetilde{\Phi} \begin{pmatrix} x(t_k) \\ x(t_k) \end{pmatrix} < 0$. Let now t_{k+1} be

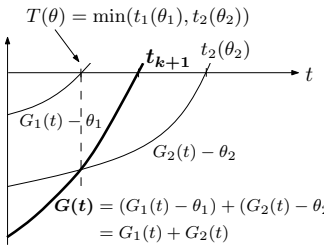

Fig. 1. For a given θ, we have $G_i(T(\theta)) - \theta_i \leq 0$ for all i, with equality for at least one subsystem i (in the figure, this is subsystem 1 since $t_1(\theta_1) < t_2(\theta_2)$)

[3] To lighten the notations we write $G_i(t)$ instead of $G_i(x_i(t))$ and $G(t)$ instead of $G(x(t))$.

[4] The initial value $\theta^{(0)}$ in Alg. 1 satisfies $\sum \theta_i^{(0)} = 0$ and $G_i(t_k) - \theta_i^{(0)} < 0$

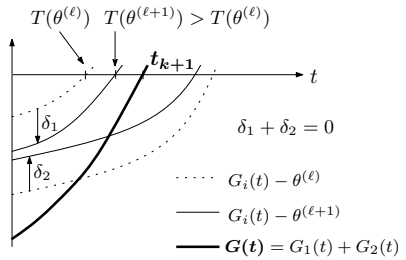

Fig. 2. This figure shows how to improve the value of the θ's from iteration ℓ to iteration $\ell+1$, in the sense that $T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$. Observe that δ_1 and δ_2 on the figure are such that $\delta_1 + \delta_2 = 0$ (condition (12)), and $G_i(T(\theta_i^{(\ell)})) - \theta_i^{(\ell)} + \delta_i < 0$ (condition (13)). As can be seen in the figure, these conditions guarantee that for $\theta^{(\ell+1)} := \theta^{(\ell)} + \delta$, we have $T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$.

the earliest time $t > t_k$ such that $\begin{pmatrix} x(t) \\ x(t_k) \end{pmatrix}^T \widetilde{\Phi} \begin{pmatrix} x(t) \\ x(t_k) \end{pmatrix} = 0$.

The sequence $\theta^{(\ell)}$ generated by Algorithm 1 satisfies

$$T(\theta^{(\ell+1)}) > T(\theta^{(\ell)})$$

and also $T(\theta^{(\ell)}) < t_{k+1}$ for all ℓ such that $\theta^{(\ell)}$ is defined. In other words, the algorithm produces a sequence of increasingly tight lower bounds to t_{k+1}.

Algorithm 1 Iterative algorithm to approach the value of θ for which $T(\theta)$ is maximal. This yields a lower bound on the triggering time since $T(\theta) < t_{k+1}$ for any θ.

Initialize $\theta_i^{(0)} = G_i(t_k) - \sum_j G_j(t_k)/n$
for $\ell = 0$ to `numiter` **do**
  $t_i(\theta_i^{(\ell)}) \leftarrow \min\{t \geq t_k \colon G_i(t) - \theta_i^{(\ell)} = 0$ and $G_i(s) - \theta_i^{(\ell)} < 0 \ \forall s \in [0,t)\}$
  $T(\theta^{(\ell)}) \leftarrow \min_{i=1,\ldots,n} t_i(\theta_i^{(\ell)})$
  $\overline{\delta}_i \leftarrow \min\{|G_i(t) - \theta_i| \colon t \in [0, T(\theta^{(\ell)})]\}$
  $\delta \leftarrow$ solution to optimization problem (14), if feasible
  if problem (14) infeasible **then**
    break
  end if
  $\theta_i^{(\ell+1)} = \theta_i^{(\ell)} - \delta_i$
end for

Algorithm 1 is typically run for a constant number of iterations `numiter` equal to 10. Furthermore, each iteration of the algorithm has a computational cost that is linear in n (where n is the number of states of the system) and hence the algorithm has a total cost that scales linearly [5] with n.

4. IMPROVED TRIGGERING CONDITIONS FOR SELF-TRIGGERED CONTROL

The algorithm presented in the previous section works for any triggering matrix $\widetilde{\Phi}$ that has its upper-left block diagonal. Recall from Sec. 2 however that ideally we would like to compute the earliest time t_{k+1} for the *original* triggering matrix Φ which corresponds to the triggering rule $\dot{V} = -x^T S x$. Since in general Φ does not have the required diagonal structure, we need to find a valid bound $\widetilde{\Phi}$ of the original Φ before using the algorithm of the previous section. One valid bound $\widetilde{\Phi}$ that has the required structure is the one proposed in Tabuada [2007] that we mentioned earlier (cf. equation (6)). In this section however we will try to find somehow "optimal" valid approximations $\widetilde{\Phi}$ that have the required diagonal structure using semidefinite-programming.

Recall that $\widetilde{\Phi}$ is a valid bound to Φ if the following implication holds for any z:

$$z^T \Phi z \geq 0 \Rightarrow z^T \widetilde{\Phi} z \geq 0.$$

This guarantees that the triggering time associated with $\widetilde{\Phi}$ will always be *smaller* than the triggering time associated with Φ. By the S-lemma, the constraint above is equivalent to the LMI condition

$$\exists \gamma_1 \geq 0 \text{ such that } \widetilde{\Phi} \succeq \gamma_1 \Phi.$$

From this observation, one way of finding a valid bound $\widetilde{\Phi}$ with the required structure is by solving the following SDP problem:

$$\begin{aligned} \underset{\widetilde{\Phi},\gamma_1}{\text{minimize}} \ & \text{trace}(\widetilde{\Phi}) \\ \text{subject to} \ & \widetilde{\Phi} \succeq \gamma_1 \Phi, \ \gamma_1 \geq 0 \\ & \widetilde{\Phi} \preceq \begin{pmatrix} I - \sigma^2 I & -I \\ -I & I \end{pmatrix}, \\ & \text{Upper-left block of } \widetilde{\Phi} \text{ diagonal} \end{aligned} \tag{15}$$

The second constraint of the SDP above guarantees that the optimal $\widetilde{\Phi}$ will be at least as good as the one of Tabuada [2007] given in equation (6). Also, in the SDP above we are interested in the minimization of trace$(\widetilde{\Phi})$ since we aim at finding the *tightest* triggering matrix which guarantees the given constraints. Other objective functions (e.g., like trace$(N\widetilde{\Phi})$ with positive semidefinite matrix N) can of course be envisaged.

5. NUMERICAL EXAMPLES

We now present examples illustrating the effectiveness of the proposed techniques. The first example will serve to verify the performance of the self-triggered implementation with respect to its accuracy to compute the event times. Moreover, we evaluate how the inter-execution times are enlarged by using the improved triggering condition of Sec. 5. A time-response analysis of a well known system is performed to assess the performance of both contributions.

5.1 Unstable second-order system

We consider the following unstable second-order system from Garcia and Antsaklis [2011] given by

$$A = \begin{pmatrix} 0.55 & -0.4 \\ 0.3 & -0.7 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

and $K = -(1.3424 \ 0.0095)$, $Q = I$ and P obtained from solving the Lyapunov equation. The performance parameter S and σ are set to $S = 0.5I$ and $\sigma = 0.22$. The number of iterations for algorithm 1 is set to $numiter = 10$ and $numIterDelta = 15$.

We compare the inter-execution times computed using three different methods: the main algorithm from Sec. 3, the algorithm proposed in Mazo Jr. and Tabuada [2011] and an exact computation of the times. The comparison is performed using the same triggering condition from Tabuada [2007] in (6) for all computation methods. We evaluate these implementations for different initial conditions $x(0) = [\cos(\alpha) \ \ \sin(\alpha)]^T$ for $\alpha \in [\frac{\pi}{30}, 2\pi]$ with $\frac{\pi}{30}$ increments. The results are shown in Fig. 3. The maximum relative error and average relative error of the main algorithm to the exact computation was $9.6 \times 10^{-3}\%$ and $6.5 \times 10^{-4}\%$, respectively. On the other hand the algorithm proposed in Mazo Jr. and Tabuada [2011] achieves a maximum relative error of 16% and an average relative error of 1.8%.

[5] The bisection algorithm to find the δ's does have complexity linear in n because it is in practice run for a constant number of iterations (say 15), and each iteration has complexity linear in n.

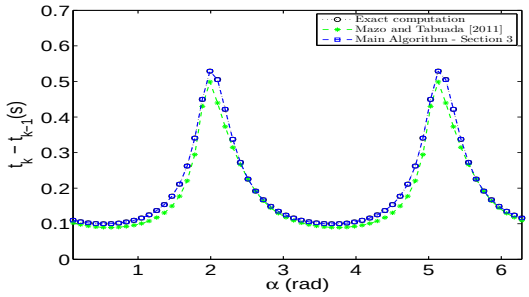

Fig. 3. Inter-execution time computation for the triggering condition Tabuada [2007] in (6) for different computation methods. Times for different initial conditions $x(0) = [\cos\alpha \;\; \sin(\alpha)]^T$, $\alpha \in [\frac{\pi}{30}, 2\pi]$ with $\frac{\pi}{30}$ increments.

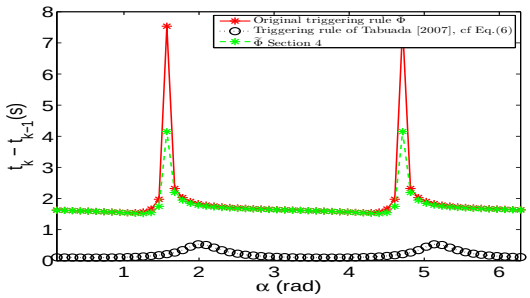

Fig. 4. Exact computation of the inter-execution times for different triggering rules: Φ in (4), $\widetilde{\Phi}$ from Tabuada [2007] in (6) and $\widetilde{\Phi}$ from Sec. 4.

Hence, the main algorithm proposed in Sec. 3 is able to provide a very tight bound to the exact inter-execution time for this system. Additionally, the inter-execution time converged to its final value in a maximum of 4 iteration steps of Algorithm 1 for any of the evaluated initial conditions.

We now evaluate the method proposed in Sec. 4. Fig. 4 depicts the results for the same set of initial conditions as in the previous example. We compare the inter-execution time given by an exact computation for different triggering rules: original triggering condition Φ in (4), the triggering condition $\widetilde{\Phi}$ from Tabuada [2007] in (6) and $\widetilde{\Phi}$ from Sec. 4. The maximum relative error and average relative error of $\widetilde{\Phi}$ from Sec. 5 to Φ in (4) was 44.5% and 14.5%, respectively. The triggering condition given by $\widetilde{\Phi}$ from Tabuada [2007] on the other hand achieves a maximum relative error of 97.2% and an average relative error of 169.8%. Clearly, there is an advantage of performing the SDP technique and compute $\widetilde{\Phi}$ since it is able to greatly enlarge the inter-execution times. This is achieved while guaranteeing the same rate of decay of the Lyapunov equation V as for the other triggering conditions.

5.2 Batch reactor

To illustrate the performance of the proposed techniques in a time-response experiment, we borrow the Batch Reactor model from Walsh et al. [1999]. The same system parameters are used and the state-feedback controller gain is:

$$K = \begin{pmatrix} -0.2713 & -0.9600 & 0.2645 & 0.0214 \\ 2.4207 & -0.3256 & 0.0275 & 0.1062 \end{pmatrix}.$$

The initial condition is set $x(0) = [-15\ 14\ -23\ 15]^T$. Matrix Q is set to be the identity $Q = I$ and P is obtained from solving the Lyapunov equation. The performance parameter S and $\sigma = 0.387$ were $0.1I$ and 0.387, respectively. Additionally, $numiter = 10$ and $numIterDelta = 15$. The closed-loop system is set to run for 1 second and Table 1 presents

Table 1. Event and self-triggered control of the Batch Reactor.

Scheme / Number of control executions	
Event-triggered Φ	5
Event-triggered of $\widetilde{\Phi}$ given by (6)	10
Event-triggered with improved $\widetilde{\Phi}$ from Sec. 4	5
Our algorithm with $\widetilde{\Phi}$ given by (6)	11
Our algorithm with improved $\widetilde{\Phi}$ from Sec. 4	5
Algorithm of Mazo and Tabuada [2011] with $\widetilde{\Phi}$ given by (6)	17

the number of control-loop executions performed by different implementations. We compare the event- and self-triggered implementations for three different triggering rules: Φ in (4), $\widetilde{\Phi}$ from Tabuada [2007] in (6) and $\widetilde{\Phi}$ from Sec. 4. The self-triggered implementations are performed using our algorithm from Sec. 3 and the method in Mazo Jr. and Tabuada [2011].

As the results show, the self-triggered implementation using our algorithm is able to achieve approximately the same number of control-execution as the event-triggered implementation for both triggering conditions $\widetilde{\Phi}$ from Tabuada [2007] in (6) and $\widetilde{\Phi}$ from Sec. 5. Additionally, the event-triggered implementation of $\widetilde{\Phi}$ from Sec. 5 achieves the same performance as Φ in (4).

6. CONCLUSIONS

In this paper, we developed a computationally efficient method to compute the inter-execution times for event-triggered implementations of diagonalizable systems, where the triggering conditions are quadratic and have a nice separability structure. This method uses an idea from Mazo Jr. and Tabuada [2011] which was proposed in the context of decentralized event-triggered control. Through numerical examples we have shown that a self-triggered implementation using this approach is able to provide a tight bound to the exact inter-execution times. Additionally, in the second part of this paper, we proposed an SDP technique to find quadratic triggering conditions that best bounded Φ as well as possessed the required separability properties to enable the application of the previous method. We have shown that this technique allows us to obtain larger inter-execution times of a self-triggered implementation of the triggering condition proposed in Tabuada [2007], while providing the same performance guarantees with respect to the rate of decay of the Lyapunov function.

Appendix A. SOME CLOSED-FORM EXPRESSIONS

A.1 Calculation of the triggering time

For the diagonalized linear system, the solution of the $i-th$ state linear differential equation at a given time t_e is given by:

$$x(t_e) = e^{at_e}x(t_0) + \frac{v}{a}\left(e^{at_e} - 1\right) = e^{at_e}\left(x(t_0) + \frac{v}{a}\right) - \frac{v}{a}, \tag{A.1}$$

where $v = [Bu]_i$ is the input signal contribution $i-th$ state and $a = A_{ii}$.

The triggering condition for each state x_i is defined as

$$G_i(x_i) - \theta_i = 0, \tag{A.2}$$

as discussed in Sec. 3.1. Consequently, developing (A.2) we achieve:

$$\bar{a}x(t_e)^2 + \bar{b}x(t_e) + \bar{c} = 0. \tag{A.3}$$

Roots q_1, q_2 of the quadratic equation are the solutions for $x(t_e)$.

Equalizing (A.1) to the two solutions of (A.3) and developing that expression we arrive to the following expression for the triggering times:

$$t_e = \frac{1}{a}\log\left(\frac{q_1 + \frac{v}{a}}{x(t_k) + \frac{v}{a}}\right) \bigvee \; t_e = \frac{1}{a}\log\left(\frac{q_2 + \frac{v}{a}}{x(t_k) + \frac{v}{a}}\right) \tag{A.4}$$

A.2 Finding $\bar{\delta}$

The value $\bar{\delta}$ is given by:

$$\bar{\delta} = \max_{t\in[t_k,T(\theta)]} G_i(t) - \theta_i. \tag{A.5}$$

In order to evaluate (A.5) we must investigate the solution of $\frac{d(G_i(t)-\theta_i)}{dt} = \frac{dG_i(t)}{dt} = 0$. Substituting (A.1) in $G_i(t)$ and calculating the derivative we get

$$\frac{dG_i(t_e)}{dt} = 2e^{at_e}\bar{a}\left(x(t_0)+\frac{v}{a}\right)^2 + \left(x(t_0)+\frac{v}{a}\right)\left(\bar{b}-2\bar{a}\frac{v}{a}\right) \tag{A.6}$$

We can now solve (A.6) to find the time at which there exists a maximum/minimum of the gap $G_i(t)$, which has the following solution

$$t_e^* = \frac{1}{a}\log\frac{2\bar{a}\frac{v}{a}-\bar{b}}{2\bar{a}\left(x(t_0)+\frac{v}{a}\right)} \tag{A.7}$$

Since there only exists one maximum/minimum of the $G_i(t)$, the solution to $\bar{\delta}$ is then computed as:

$$\bar{\delta} = \max\left\{G_i(t)-\theta_i : t = \{t_k, t_e^*, T(\theta)\}\right\} \tag{A.8}$$

Appendix B. SOLVING OPTIMIZATION PROBLEM (14)

In this section we show how one can efficiently solve the optimization problem (14) for the δ, which we rewrite here:

$$\begin{aligned} \underset{\delta_1,\ldots,\delta_n}{\text{minimize}} \quad & \max_{i=1,\ldots,n}(c_i+\delta_i) \\ \text{subject to} \quad & \delta_i \le \overline{\delta_i} \text{ for all } i = 1,\ldots,n \\ & \sum_{i=1}^{n}\delta_i = 0 \end{aligned} \tag{B.1}$$

We will assume that $\overline{\delta_i} \ge 0$ in this section, which is true in our case (cf. equation (A.5)). Note that the problem can be written as a linear program (LP) and so can be solved by any LP solver. One can however use a simple bisection algorithm to solve this problem more efficiently without resorting to LP packages.

The algorithm relies on the observation that given a real number X, there is a simple explicit method to decide whether the optimal value of the problem is less than or equal to X. This decision problem can be written as follows:

Given $X \in \mathbb{R}$, does there exist $\delta_1,\ldots,\delta_n$ such that:

(1) $\max_{i=1,\ldots,n}(c_i+\delta_i) \le X$

(2) $\delta_i \le \overline{\delta_i}$ for all $i = 1,\ldots,n$

(3) $\sum_{i=1}^{n}\delta_i = 0$

This decision problem can be solved as follows: Let $I_> = \{i \mid c_i \ge X\}$ and $I_< = \{i \mid c_i < X\}$. If δ is a feasible solution to the decision problem above, then from conditions (1) and (2) we necessarily have, for $i \in I_>$, $\delta_i \le X - c_i \le 0$, and for $i \in I_<$, we will have $0 \le \delta_i < \min(X-c_i, \overline{\delta_i})$. It is therefore easy to see such that a vector δ satisfying (1), (2) and (3) exists if, and only if

$$\sum_{i\in I_>}(c_i - X) \ge \sum_{i\in I_<}\min(X-c_i, \overline{\delta_i}).$$

Moreover, if this condition is true and $s = \sum_{i\in I_<}\min(X - c_i, \overline{\delta_i}) - \sum_{i\in I_>}|X-c_i| > 0$, then the vector δ defined by:

$$\delta_i = \begin{cases} X - c_i - s/|I_>| & \text{if } i \in I_> \\ \min(X-c_i, \overline{\delta_i}) & \text{otherwise} \end{cases}$$

will satisfy conditions (1), (2) and (3) above.

The complete bisection algorithm to solve the optimization problem (B.1) is given in Algorithm 2.

Algorithm 2 Bisection algorithm to solve the optimization problem (B.1). The algorithm reaches an accuracy of $2^{-\texttt{numIterDelta}}$ after `numIterDelta` iterations.

Input: $c, \overline{\delta} \in \mathbb{R}^n$, `numiter` ($\approx 15$)
Output: $\delta^* := \text{argmin}\{\max_i(c_i+\delta_i) : \delta_i \le \overline{\delta}_i, \sum_i \delta_i = 0\}$
$X_1 \leftarrow \min\{c_i, i = 1,\ldots,n\}$
$X_2 \leftarrow \max\{c_i, i = 1,\ldots,n\}$
while $k \le$ `numIterDelta` **do**
  $X \leftarrow (X_1 + X_2)/2$
  $I_> \leftarrow \{i \in \{1,\ldots,n\} : c_i \ge X\}$
  $I_< \leftarrow \{i \in \{1,\ldots,n\} : c_i < X\}$
  if $\sum_{i\in I_>}(c_i - X) \ge \sum_{i\in I_<}\min(X-c_i, \overline{\delta_i})$ **then**
    $X_1 \leftarrow X$
  else
    $X_2 \leftarrow X$
  end if
  $k \leftarrow k+1$
end while
$\delta^* \leftarrow \min(X-c, \overline{\delta})$

REFERENCES

A. Anta and P. Tabuada. To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030 –2042, 2010.

K.E. Årzén. A simple event-based PID controller. *Preprints 14th World Congress of IFAC. Beijing, China*, 1999.

K.J. Åström and B.M. Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. In *14th IFAC World Congress*, 1999.

K.J. Åström and B. Wittenmark. *Computer controlled systems*. Prentice Hall Englewood Cliffs, NJ, 1990.

A. Cervin and T. Henningsson. Scheduling of event-triggered controllers on a shared network. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2008.

D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 2012.

E. Garcia and P. Antsaklis. Model-based event-triggered control with time-varying network delays. In *IEEE CDC-ECC*, 2011.

W.P.M.H. Heemels, J.H. Sandee, and P.P.J. van den Bosch. Analysis of event-driven controllers for linear systems. *Int. J. of Control*, pages 81(4), 571–590, 2008.

M. Lemmon. Event-triggered feedback in control, estimation, and optimization. In *Networked Control Systems*, volume 406. Springer Berlin / Heidelberg, 2011.

J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46(1):211–215, January 2010.

M. Mazo Jr. and P. Tabuada. On the robustness of self-triggered control for sensor/actuator networks. 2009.

M. Mazo Jr. and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *Automatic Control, IEEE Transactions on*, 56(10), oct. 2011.

M. Mazo Jr., A. Anta, and P. Tabuada. On self-triggered control for linear systems: Guarantees and complexity. 2009.

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

M. Velasco, J. Fuertes, and P. Martí. The self triggered task model for real-time control systems. *24th IEEE Real-Time Systems Symposium (work in progress)*, 2003.

G. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, pages 2876–2880, 1999.

X. Wang and M. Lemmon. Self-triggered feedback control systems with finite-gain l2 stability. *IEEE Transactions on Automatic Control*, 45:452–467, 2009.