# On Self-Triggered Control for Linear Systems: Guarantees and Complexity

Manuel Mazo Jr., Adolfo Anta and Paulo Tabuada

*Abstract*— **Typical digital implementations of feedback controllers periodically measure the state, compute the control law, and update the actuators. Although periodicity simplifies the analysis and implementation, it results in a conservative usage of resources. In this paper we drop the periodicity assumption in favor of self-trigger strategies that decide when to measure the state, execute the controller, and update the actuators according to the current state of the system. In particular, we develop a general procedure leading to self-triggered implementations of feedback controllers, that highly reduces the number of controller executions while guaranteeing a desired level of performance. We also analyze the inherent trade-off between the computational resources required for the self-triggered implementation and the resulting performance. The theoretical results are applied to a physical example to show the benefits of the approach.**

## I. Introduction

Feedback control laws are traditionally implemented in a periodic fashion due to the ease of design and analysis. However, this approach represents a conservative solution since the controller is updated at the same rate regardless of the current state of the plant. A system could be on a stable manifold of the state space and therefore require little or no attention, while if the system lies on an unstable manifold, it might demand faster updates. Periodic implementations cannot take advantage of this fact because they disregard the information contained in the state. Moreover, the period is usually selected to guarantee a desired performance under worst case conditions even though these might rarely occur. Although there is a vast literature on periodic implementations of feedback control laws, ad hoc rules are still applied to determine stabilizing periods (for instance, 20 times the bandwidth of the system).

With the advent of embedded and networked control systems, greater functionality is expected and control loops no longer have at its disposal dedicated computational and communication resources. While traditionally the implementation aspects were ignored at the design stage, this can no longer be done. Hence, intensive research is being currently conducted to ascertain the real-time requirements of control systems. Several researchers proposed resource-aware implementations of control laws, such as event-triggered control ([1], [2], [3], [4]). Under this paradigm, the controller execution is triggered according to the state of the plant. This approach calls for resources whenever they are indeed necessary, and it provides a high degree of robustness, since the system is being permanently monitored. However, these implementations require, in general, dedicated hardware to continuously monitor the state of the plant. Moreover, event-triggered control systems lack a systems theory that facilitates the analysis of such systems.

To overcome these drawbacks of event-triggered systems, in this paper we advocate the use of self-triggered implementations. The underlying idea is to emulate the event-triggered implementation without resorting to extra hardware. In most control systems, the state of the plant has to be measured or estimated. Thus, this information can be used by the controller to decide its next execution time. The self-trigger approach was previously studied in [5], where the next execution time is treated as a new state variable. Although it represents an important step towards resource-aware implementation, no guarantees of correct operation or performance were provided. Self-triggered control was also analyzed in [6], for the particular class of full-information $\mathcal{H}_\infty$ controllers, in [7] for distributed systems and in [8] for nonlinear systems.

In this paper, we show how to obtain a self-triggered implementation of any linear state-feedback controller stabilizing a linear control system. Moreover, the resulting self-triggered implementation preserves exponential stability while reducing the number of executions of the controller. We also study the trade-off between the complexity of the implementation and the resulting control performance.

The rest of the paper is organized as follows: Section II introduces some preliminary notation and definitions. Section III defines event-triggered and self-triggered control and introduces the problem we tackle in this paper. Sections IV and V describe the proposed self-triggered implementation. The performance guarantees and complexity of the implementation are provided in Section VI. Some examples illustrating the proposed techniques are presented in Section VII. Related topics such as robustness of the self-trigger approach and non-zero computation delays are briefly discussed in Section VIII. The proof of the main theorem in the paper is included in Appendix IX.

## II. Notation

We denote by $\mathbb{N}^+$ the positive natural numbers, and $\mathbb{R}^+$ to the positive real numbers. We also use $\mathbb{N}_0^+ = \mathbb{N}^+ \cup \{0\}$ and $\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$. The usual Euclidean vector norm is represented by $|\cdot|$. The set of eigenvalues of the matrix $A$ is indicated with $\{\lambda_i(A)\}$. A function $\gamma : [0, a[ \to \mathbb{R}_0^+$, $a > 0$ is of class $\mathcal{K}$ if it is continuous, zero at zero and strictly

M. Mazo Jr, A. Anta, and P. Tabuada are with the Department of Electrical Engineering, Henry Samueli School of Engineering and Applied Sciences, University of California, Los Angeles, CA 90095-1594, {mmazo,adolfo,tabuada}@ee.ucla.edu

This work has been partially funded by the NSF awards 0712502, 0841216, and 0820061, a Spanish Ministry of Science and Education/UCLA LA2004 − 0003 fellowship and a Mutua Madrileña Automovilística scholarship.

increasing. A continuous function $\beta : \mathbb{R}_0^+ \times [0, a[ \to \mathbb{R}_0^+$ is of class $\mathcal{KL}$ if $\beta(\tau, \cdot)$ is of class $\mathcal{K}$ for each $\tau \geq 0$ and $\beta(\cdot, s)$ is monotonically decreasing to zero for each $s \geq 0$. A class $\mathcal{KL}$ function $\beta(\tau, s)$ is called exponential if $\beta(\tau, s) \leq kse^{-c\tau}$, $k > 0$, $c > 0$.

## III. Problem Statement

The problem we aim to solve is that of finding an efficient sample-and-hold implementation of a linear controller. In this context an implementation will be more efficient than another when it requires fewer executions to achieve the same stability performance. Before formally introducing the problem we tackle in this paper, we need to introduce the notions of *Event-Triggered Control* and *Self-Triggered Control*.

### A. *Event-triggered control.*

Assume a linear control system:

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

and a linear feedback controller:

$$u = Kx \tag{2}$$

rendering the closed loop asymptotically stable. For a continuous implementation we have $\dot{x}_c = (A + BK)x_c$, hence there exists a Lyapunov function $V_c(t) := x_c^T(t)Px_c(t)$ such that:

$$\begin{aligned} \dot{V}_c &= x_c^T \left( (A + BK)^T P + P(A + BK) \right) x_c && (3) \\ &= -x_c^T Q x_c && (4) \end{aligned}$$

with $P$ satisfying the Lyapunov equation:

$$-Q = (A + BK)^T P + P(A + BK) \tag{5}$$

for some positive definite matrix $Q$.

Now let the control signal be obtained from sample and hold measurements:

$$u(t) = Kx(t_k), \qquad t \in [t_k, t_{k+1}[$$

where $t_k$ and $t_{k+1}$ are two consecutive sampling instants. We will call to any implementation in which the control signal is held constant between executions, not necessarily periodic, a sampled-data system. The closed loop dynamics under a sample and hold implementation of the controller is described by:

$$\dot{x} = Ax + BKx(t_k) = (A + BK)x + BKe \tag{6}$$

where $e$ represents a measurement error defined by:

$$e(t) := x(t_k) - x(t), \quad t \in [t_k, t_{k+1}[ \tag{7}$$

Treating $e(t)$ as a new state variable we can rewrite the state space representation of the system as:

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A + BK & BK \\ -A - BK & -BK \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix} \tag{8}$$

The selection of the sequence of update times $\{t_k\}$ at which the controller is updated will determine the behavior of the sampled-data system. In a periodic implementation, we have $t_{k+1} = t_k + T$, $\forall k \in \mathbb{N}_0^+$, for some specified period $T > 0$. In contrast with a periodic implementation, an event-triggered strategy defines the sequence $\{t_k\}$ implicitly as the times at which some triggering condition is violated. We start this paper by proposing one such triggering condition.

We define the function $V(t, x_0) := x(t)^T P x(t)$, where $x(t)$ is a solution of (1), $x(0) = x_0$, and $P$ is a positive definite matrix. The specification is defined by a function $S : \mathbb{R}_0^+ \times \mathbb{R}^n \to \mathbb{R}_0^+$ upper-bounding the evolution of $V$. This function $S$ describes the desired performance for the implemented control system. Hence, the update times $t_k$ are determined by the time instants at which:

$$V(t, x_0) \leq S(t, x_0), \; t \geq t_0 \tag{9}$$

is violated. An event-triggered implementation will recompute the controller whenever $V(t_k, x_0) = S(t_k, x_0)$, in order to prevent the violation of its triggering condition (9), since $V(0, x_0) \leq S(0, x_0)$ for any $x_0$. We refer to $S(t, x_0)$ as the performance function from here on. We will drop the explicit dependence on $x_0$ of $V$ and $S$ whenever it is clear from the context.

### B. *Self-triggered control.*

The self-triggered control paradigm aims at obtaining a map relating the current state $x(t_k)$ with the next sampling time: $h : \mathbb{R}^n \to \mathbb{R}^+$, $h(x(t_k)) = t_{k+1}$. This map shall provide with an emulation of the events generated by some triggering condition, *e.g.* condition (9). If constructing such $h(\cdot)$ is possible, a self-triggered implementation provides with the same inter-execution times as an event-triggered implementation. Furthermore, the self-triggered implementation does not require continuous monitoring of the state. Now we can introduce the problem we tackle in the present paper.

*Problem 3.1 (Self-Triggered Control Design):* Given the linear control system (1), the linear state feedback controller (2), an output function $V(t, x_0)$, and a performance function $S(t, x_0)$, find a self-triggered implementation achieving a predefined performance specified as a bound on the evolution of $V$ by $S$, *i.e.* $V(t, x_0) \leq S(t, x_0)$.

## IV. Performance Specification

The first step in the design of our self-triggered implementation is to specify the desired performance. It is straightforward to conclude that an event-triggered implementation based on (9) with $S(t, x_0) \leq \beta(t, |x_0|)$, $\forall x_0 \in \mathbb{R}^n$, where $\beta(t, |x_0|) \in \mathcal{KL}$, renders the system (1) uniformly globally asymptotically stable [9]. Furthermore, if $\beta(t, |x_0|)$ is an exponential $\mathcal{KL}$ function, the system is uniformly globally exponentially stable [9]. We are only interested in specifications implying at least asymptotic stability of the system, but we do not restrict ourselves to the case when $S(t, x_0) := \beta(t, |x_0|)$, for $\beta(t, |x_0|) \in \mathcal{KL}$.

Moreover, in order for an event-triggered or self-triggered implementation to be of any use, the inter-execution times $t_{k+1} - t_k$ should be lower bounded by some positive quantity, *i.e.* $t_{k+1} - t_k \geq t_{min} > 0$. To enforce inter-execution times greater than zero it is sufficient to design $S$ satisfying $\dot{V}(t_k) < \dot{S}(t_k)$ at the execution times $t_k$. For that matter, we construct a hybrid system that describes the desired performance. Thus, a possible design of $S(t)$ satisfying the above implication is given by the Lyapunov function:

$$S(t) \;:=\; x_s(t)^T P x_s(t)$$

for the hybrid system:

$$\begin{aligned} \dot{x}_s &= A_s x_s,\ t \in [t_k, t_{k+1}[ \\ x_s(t_k) &:= x(t_k) \end{aligned} \tag{10}$$

where $A_s$ is a Hurwitz matrix satisfying the Lyapunov equation:

$$\left(A_s^T P + P A_s\right) = -R \tag{11}$$

In (11), $R$ is a positive definite matrix making $Q - R$ also positive definite, and hence guaranteeing $\dot{V}(t_k) < \dot{S}(t_k)$. Matrix $R$ describes the stability requirements for the implementation as it defines $A_s$, used to determine $S(t)$.

## V. Self-Trigger Policy

In this section, a self-trigger condition guaranteeing inequality (9) is derived. We start by finding a lower bound for the inter-execution times generated by such condition.

### A. Computation of the minimum sampling-time.

The inter-execution times of this implementation are implicitly defined by the inequality (9). By defining $y = [x^T e^T]^T$, we can rewrite system (8) in the following form:

$$\dot{y} = Fy, \qquad F := \begin{bmatrix} A+BK & BK \\ -A-BK & -BK \end{bmatrix}$$

whose solution is $y = e^{Ft} y_0$. With this notation, $S(t)$ and $V(t)$ become:

$$S(t) = (Ce^{F_s t} y_0)^T P (Ce^{F_s t} y_0) \tag{12}$$
$$V(t) = (Ce^{Ft} y_0)^T P (Ce^{Ft} y_0) \tag{13}$$

for:

$$F_s := \begin{bmatrix} A_s & 0 \\ 0 & 0 \end{bmatrix} \qquad C := \begin{bmatrix} I & 0 \end{bmatrix} \tag{14}$$

The triggering condition $V(t) \leq S(t)$ then turns into the following inequality:

$$f(t, y_0) := y_0^T (e^{F^T t} C^T P C e^{Ft} - e^{F_s^T t} C^T P C e^{F_s t}) y_0 \leq 0 \tag{15}$$

Let $t = h(x_0)$ be the relation between $t$ and $x_0$ described implicitly as the times at which the transcendental equation (15) becomes an equality, that is, $f(h(x_0), x_0) = 0$. While it is not possible to find such map $h$ in closed-form, we can find its minimum value. This will provide us with a tight lower bound for the inter-execution times under the self-trigger strategy. In order to do so, we differentiate equation (15) with respect to the initial condition $x_0$. For each coordinate we obtain:

$$\frac{df(t, y_0)}{dx_0^i} = \frac{\partial f(t, y_0)}{\partial x_0^i} + \frac{\partial f(t, y_0)}{\partial t}\frac{\partial t}{\partial x_0^i} = 0, \quad i = 1, \ldots, n$$

The extrema of the map $t = h(x_0)$ are defined by the following equation:

$$\frac{\partial t}{\partial x_0^i} = -\frac{\frac{\partial f(t, y_0^i)}{\partial x_0^i}}{\frac{\partial f(t, y_0^i)}{\partial t}} = 0, \qquad i = 1, \ldots, n \tag{16}$$

The points $(t, x_0)$ where both $\frac{\partial f(t, y_0^i)}{\partial x_0^i}$ and $\frac{\partial f(t, y_0^i)}{\partial t}$ become $0$ do not need to be considered since they do not represent points where the triggering condition is violated. Hence the system of equations that we need to solve is:

$$\frac{\partial f(t, y_0)}{\partial x_0^k} = \sum_{i=1}^{2n} \frac{\partial f(t, y_0)}{\partial y_0^i}\frac{\partial y_0^i}{\partial x_0^k} = 0, \qquad k = 1, \ldots, n \tag{17}$$

Combining (17) into matrix form we obtain:

$$M(t) x_0 = 0 \tag{18}$$

with:

$$M(t) := \begin{bmatrix} I & 0 \end{bmatrix} (e^{Ft} C^T P C e^{Ft} - e^{F_s t} C^T P C e^{F_s t}) \begin{bmatrix} I \\ 0 \end{bmatrix}$$

The solution to this equation gives us all the extrema of the map $h(x_0)$. The minimum $t$ satisfying equation (18) corresponds to the smallest inter-execution time under the triggering condition $V(t) = S(t)$. Since the left hand side of the previous equation is linear in $x_0$, it is sufficient to check when the matrix has a nontrivial nullspace. The aforementioned procedure can be summarized in the following theorem.

*Theorem 5.1:* Given the system (1) and the controller (2), the inter-execution times defined by the triggering condition (15) are lower bounded by:

$$t_{min} = \min\{t \in \mathbb{R}^+ : \det(M(t)) = 0\} \tag{19}$$

In fact, this method can also be regarded as a formal procedure to find a sampling period for periodic implementations, in contrast with the ad-hoc rules of thumb that are frequently used [10]. Moreover, this analysis can also be applied, *mutatis mutandis*, to other Lyapunov-based triggering conditions, like the ones appearing in [4] and [6].

### B. Implementation.

To simplify the presentation, we embed the state space into a higher dimensional space where both the dynamics as well as the triggering conditions become linear. This embedding also leads to a simpler implementation of the technique described below.

The desired embedding is the Veronese map of order 2 defined by:

$$\begin{aligned} \vartheta_2 &: \mathbb{R}^n \to \mathbb{R}^{\frac{n(n+1)}{2}} \\ \vartheta_2(x) &:= [x_1^2\ x_1x_2\ \ldots\ x_1x_n\ x_2^2\ x_2x_3\ \ldots\ x_n^2]^T \end{aligned}$$

The reader can easily verify that $V(t) = x(t)^T P x(t)$ can be rewritten in linear form as $V(t) = L\vartheta_2(x(t))$ for $L \in \mathbb{R}^{\frac{n(n+1)}{2}}$.

If we denote by $z, z_s \in \mathbb{R}^{\frac{2n(2n+1)}{2}}$ the vectors $z = \vartheta_2([x^T\ e^T]^T)$ and $z_s = \vartheta_2([x^T\ 0]^T)$, the triggering condition simplifies to $Lz(t) \leq Lz_s(t)$.

Furthermore, on the coordinates given by the Veronese map, the dynamics remains linear, i.e. $\dot{z} = Gz$, for some $G$ of appropriate dimensions. Hence, we can discretize the dynamics as:

$$z(t_k + r\Delta) = T^r z(t_k),\ T := e^{G\Delta}, r \in \mathbb{N}_0^+$$

where $\Delta > 0$ is the discretization step. Analogously, $\dot{z}_s = G_s z_s$ and $T_s := e^{G_s\Delta}$. As a consequence, now we can compute the value of $V(t)$ and $S(t)$ at $t = t_k + r\Delta$:

$$V(t_k + r\Delta) = LT^r z(t_k) \qquad S(t_k + r\Delta) = LT_s^r z(t_k)$$

only from the initial condition $x(t_k)$.

Using these expressions for $V$ and $S$ we can compute the next execution time as follows. Let us define $r_{min} := \lfloor \frac{t_{min}}{\Delta} \rfloor$ and $r_{max} := \lfloor \frac{t_{max}}{\Delta} \rfloor$, where $t_{max}$ is the maximum time the system is allowed to run in open loop (a design parameter), and $t_{min}$ as defined in (19). The discretized model just provides with values of $z$ at times $t_r = t_k + \Delta r$, $r \in \mathbb{N}_o^+$. The next execution time satisfying (9) could be computed as:

$$t_{k+1} = t_k + (r_{min} + \min(\{j : \nu_j > 0, \nu_{j+1} < 0\}) - 1)\Delta \quad (20)$$

where $\nu_j = S(t_k + (r_{min} + j - 1)\Delta) - V(t_k + (r_{min} + j - 1)\Delta)$ is the $j^{th}$ entry of the vector $\nu$ defined as:

$$\nu = \hat{L}z(t_k),\ \ \hat{L} = \begin{bmatrix} L(T_s^{r_{min}} - T^{r_{min}}) \\ L(T_s^{r_{min}+1} - T^{r_{min}+1}) \\ \vdots \\ L(T_s^{r_{max}} - T^{r_{max}}) \end{bmatrix} \quad (21)$$

The vector $\nu$ represents the discretized evolution of the triggering condition (9). The fact that $e(t_k) = 0$ will let us have a more efficient implementation in terms of memory usage. We will address these issues in the next section. Also notice that the matrix $\hat{L}$ is computed only once at design time, and the only terms to be computed online are $z(t_k)$ and the product $\hat{L}z(t_k)$.

The proposed policy ensures that $V(r\Delta) \leq S(r\Delta)$, $\forall r \in \mathbb{N}_0^+$, but does not establish any bound on $V(t)$ at the points $t \neq r\Delta$. At those points $V(t)$ might be greater than $S(t)$ and we could not detect it. To overcome this drawback, we develop in the following section a bound for $V(t)$ in the spirit of those in [11], in order to provide with a performance guarantee under the self-trigger policy (20).

## VI. Main Results

We establish the performance guarantees achieved by the self-trigger protocol (20) in the following theorem:

*Theorem 6.1 (Guaranteed performance):* The linear system (1), with control law (2) under the implementation defined by (20), with $S(t)$ defined as in (10) and $\Delta$ as discretization step satisfies:

$$V(t, x(t_k)) \leq g(\Delta, r_{max}) S(t, x(t_k)),\ \forall t \in [t_k, t_{k+1}[ \quad (22)$$

where:

$$g(\Delta, r_{max}) := e^{\frac{(\rho+\overline{\lambda})\mu\Delta}{\mu-\rho}} + e^{\overline{\lambda}(r_{max}-1)\Delta}\left(e^{\frac{(\rho+\overline{\lambda})\mu\Delta}{\mu-\rho}} - e^{\frac{\overline{\lambda}\mu\Delta}{\mu-\rho}}\right)$$

$$\rho := \max_i(\{\lambda_i(F)\})$$

$$\mu := \min_i(\{\lambda_i(F)\})$$

$$\overline{\lambda} := \max_i(\{\lambda_i(\tilde{R})\}),\ \tilde{R} := P^{-\frac{1}{2}} R P^{-\frac{1}{2}}$$

$$F := \begin{bmatrix} P^{\frac{1}{2}}AP^{-\frac{1}{2}} + (P^{\frac{1}{2}}AP^{-\frac{1}{2}})^T & P^{\frac{1}{2}}BKP^{-\frac{1}{2}} \\ (P^{\frac{1}{2}}BKP^{-\frac{1}{2}})^T & 0 \end{bmatrix}$$

*Corollary 6.2 (Exponential Stability):* The implementation is uniformly globally exponentially stable with:

$$V(t) \ \leq\ \beta(t, |x_s(t_0)|) := \bar{\sigma} g(\Delta, r_{max})|x_s(t_0)|^2 e^{-\underline{\lambda}t},\ \forall t \in \mathbb{R}_0^+$$
$$\underline{\lambda} \ :=\ \min_i(\{\lambda_i(\tilde{R})\}), \qquad \bar{\sigma} := \max_i(\{\lambda_i(P)\})$$

The proof to the theorem is included in the appendix. Tighter, but more convoluted, bounds than the ones presented in the theorem can be obtained through equations (33) and (34) in the appendix. The proof of the corollary is straightforward. The value $g(\Delta, r_{max})$ can be regarded as a measure of the degradation of the performance as a function of $\Delta$ and $r_{max}$. One can also see this measure of degradation in the form of a delay, *i.e.* $V(t) \leq V(t_0)e^{-\underline{\lambda}(t-\delta)}$, with the delay given by:

$$\delta = \frac{1}{\underline{\lambda}} \log(g(\Delta, r_{max})) \quad (23)$$

When discussing the space and time complexity of the proposed self-trigger strategy, we shall use $M_s$ to denote space complexity and $M_t$ to denote time complexity. Note that $M_s$ and $M_t$ only describe the space and time complexity that are needed to go from a periodic implementation to a self-triggered implementation. Regarding time complexity, we assume a unitary cost for products, additions and comparison operations. We summarize the complexity of implementing the self-trigger protocol (20) in the following Theorem:

*Theorem 6.3 (Implementation Complexity):* The $n$-dimensional linear system (1), with control law (2) under the implementation defined by (20), with $S(t)$ defined as in (10) and $\Delta$ as discretization step requires $M_s$ units of memory and $M_t$ computations, for:

$$M_s \ :=\ N(\Delta)\frac{n(n+1)}{2} \quad (24)$$

$$M_t \ :=\ 2M_s + N(\Delta) + \frac{n(n+1)}{2} \quad (25)$$

with $N(\Delta) := \lfloor \frac{t_{max}}{\Delta} \rfloor - \lfloor \frac{t_{min}}{\Delta} \rfloor$.

*Proof:*

[Space complexity]: $N(\Delta)$ vectors are needed to check the condition (9) at the different times $t_k + r\Delta$, $r = \lfloor \frac{t_{min}}{\Delta} \rfloor \ldots \lfloor \frac{t_{max}}{\Delta} \rfloor$. Since $e(t_k) = 0$, instead of storing the matrix $\hat{L} \in \mathbb{R}^{N(\Delta) \times \frac{2n(2n+1)}{2}}$ that multiplied $\vartheta_2([x^T\ e^T]^T)$, we can select the subset of its entries corresponding to only $\vartheta_2(x)$. Thus the matrix of vectors to be stored $\hat{L}_x$ has size $N(\Delta)\frac{n(n+1)}{2}$.

[Time Complexity]: The operation $\nu = \hat{L}_x z_x(t_k)$, with $z_x = \vartheta_2(x)$ requires $N(\Delta) \times \frac{n(n+1)}{2}$ products and the same

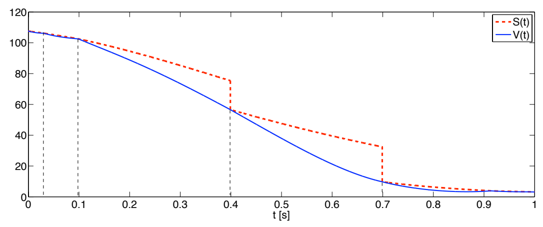

Fig. 1. Lyapunov function decay and performance function $S(t)$ for $\Delta = 1ms$.

amount of additions, besides the $\frac{n(n+1)}{2}$ products necessary to compute the embedding $\vartheta_2(x)$. Moreover, we need to find the first change of sign in the vector $\nu$ which requires $N(\Delta)$ comparisons. Adding all those terms proves the expression provided. ■

Theorem 6.3 describes the main trade-offs in this self-triggered implementation: decreasing the discretization step $\Delta$ increases the complexity linearly, but improves the performance guarantee exponentially through $g(\Delta, r_{max})$.

## VII. Example

To illustrate the performance of the proposed self-triggered implementation we borrow the Batch Reactor model from [12] with state space description:

$$\dot{x} = \begin{bmatrix} 1.38 & -0.20 & 6.71 & -5.67 \\ -0.58 & -4.29 & 0 & 0.67 \\ 1.06 & 4.27 & -6.65 & 5.89 \\ 0.04 & 4.27 & 1.34 & -2.10 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 5.67 & 0 \\ 1.13 & -3.14 \\ 1.13 & 0 \end{bmatrix} u$$

A state feedback controller placing the poles of the closed loop system at $\{-3+1.2i, -3-1.2i, -3.6, -3.9\}$ is given by:

$$K = \begin{bmatrix} 0.1006 & -0.2469 & -0.0952 & -0.2447 \\ 1.4099 & -0.1966 & 0.0139 & 0.0823 \end{bmatrix}$$

The matrix $Q$ is set to the identity $Q = I$, $R = 0.5Q$, and $P$ is obtained from solving the Lyapunov equation (5).

We compute the minimum time $t_{min} = 20ms$ using (19). We first select $t_{max} = 300ms$ and $\Delta = 1ms$. With this design the complexity becomes $M_s = 2810$ and $M_t = 5911$. The worst-case required computational speed ($M_t$ computations in $t_{min}$) does not represent an impediment for current microprocessors. Figure 1 depicts the evolution of the Lyapunov function $V(t)$ against the performance function employed in the design $S(t)$. We zoomed in the first second to better appreciate the triggering condition and the effect of setting a maximum inter-sample time. We have also marked with vertical dashed lines the instants at which a new update happens; see also Figure 2 which depicts the evolution of the inter-execution times.

We use the delay $\delta$ as the measure of the performance guarantees. Tables I and II present the different values of $\delta$ as $\Delta$ and $t_{max}$ vary. Although the performance guarantees degrade with $\Delta$, simulation results show only a modest degradation with $\Delta$ of the achieved performance. This is illustrated in Figure 3 and in Table I by looking at the number of controller updates in 10s. On the other hand we can see how increasing the time $t_{max}$ has a negative effect on the guarantees, but does provide improvements in terms of number of updates (see Table II). The implementation complexity ($M_s$ and $M_t$) for those different values of $\Delta$ and $t_{max}$, are summarized in Tables I and II.

Finally, to illustrate the effect of the discretization step $\Delta$, we raised $t_{max}$ to $900ms$, and simulated the system with initial conditions $x(\theta) = [6\sin(\theta)\ 6\cos(\theta)\ 1\ 9]^T$. In Figure 4 we present the inter-execution times produced by the self-trigger protocol with different values of $\Delta$ for initial conditions with $\theta$ ranging from 0 to 0.5 radians. We can clearly appreciate the quantizing effect of $\Delta$ on the inter-execution times.

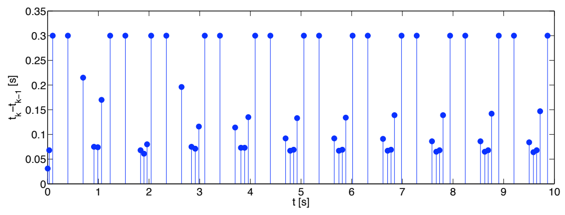

Fig. 2. Inter-execution times for $\Delta = 1ms$.

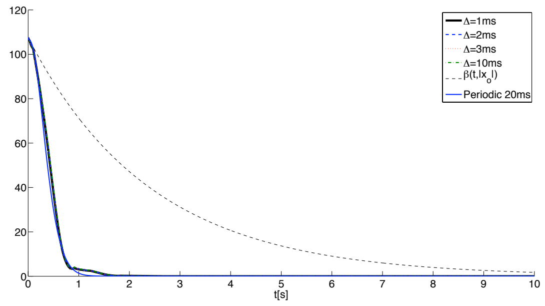

Fig. 3. Evolution of $V(t)$ for different values of $\Delta$.

TABLE I
Implementation complexity and performance as a function of $\Delta$ for $t_{max} = 0.3s$, in $10s$.

| | Periodic (T=$20ms$) | $1ms$ | $2ms$ | $5ms$ | $10ms$ |
|---|---|---|---|---|---|
| $M_s$ | 0 | 2810 | 1410 | 570 | 290 |
| $M_t$ | 0 | 5911 | 2971 | 1207 | 619 |
| $\delta[s]$ | 0 | 0.12 | 0.23 | 0.55 | 1.01 |
| $Updates$ | 500 | 62 | 62 | 62 | 63 |

TABLE II
Implementation complexity and performance as a function of $t_{max}$ for $\Delta = 5ms$, in $10s$.

| | Periodic (T=$20ms$) | $0.1s$ | $0.3s$ | $0.5s$ | $0.9s$ |
|---|---|---|---|---|---|
| $M_s$ | 0 | 170 | 570 | 970 | 1770 |
| $M_t$ | 0 | 367 | 1207 | 2047 | 3727 |
| $\delta[s]$ | 0 | 0.24 | 0.55 | 1.42 | 5.75 |
| $Updates$ | 500 | 102 | 62 | 47 | 32 |

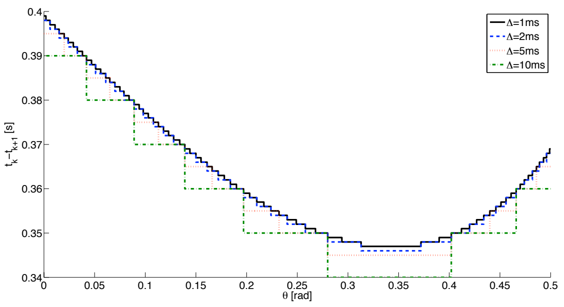

Fig. 4. Inter-execution times as a function of $\Delta$.

## VIII. Discussion

The self-triggered implementation, described in Section V, can be used for any state-feedback linear controller stabilizing a linear plant. The case of dynamic controllers, although not described in this paper, can be handled using similar techniques. Moreover, the performance degradation, described by the term $g(\Delta, r_{max})$, can be suitably modified to accommodate the influence of a non-zero delay between sensing and updating the actuators, by following the methods described in [4]. Self-triggered and event-triggered implementations are known to have some degree of robustness with respect to disturbances [13]. The authors are currently investigating the trade-offs between the computational complexity of the implementation and its robustness.

Of independent interest is the method described in Section V-A to compute the minimum inter-execution time. This method offers an alternative approach to the choice of sampling periods for periodic implementations of linear controllers.

## References

[1] K. Årzén, "A simple event based PID controller," *Proceedings of 14th IFAC World Congress*, vol. 18, pp. 423–428, 1999.
[2] K. Åström and B. Bernhardsson, "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, 2002.
[3] W. Heemels, J. Sandee, and P. van den Bosch, "Analysis of event-driven controllers for linear systems," *Int. J. of Control*, pp. 81(4), 571–590, 2008.
[4] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52(9), pp. 1680–1685, 2007.
[5] M. Velasco, J. Fuertes, and P. Marti, "The self triggered task model for real-time control systems," *Work in Progress Proceedings of the 24th IEEE Real-Time Systems Symposium*, pp. 67–70, 2003.
[6] X. Wang and M. Lemmon, "State based Self-triggered feedback control systems with L2 stability," *17th IFAC world congress*, 2008.
[7] M. Mazo Jr and P. Tabuada, "On Event-Triggered and Self-Triggered Control over Sensor/Actuator Networks," *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.
[8] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *Accepted for publication. arXiv:0806.0709*, 2008.
[9] H. Khalil, *Nonlinear systems*. Prentice Hall Upper Saddle River, NJ, 2002.
[10] K. Åström and B. Wittenmark, *Computer controlled systems*. Prentice Hall Englewood Cliffs, NJ, 1990.
[11] D. Nesic, A. Teel, and E. Sontag, "Formulas relating KL stability estimates of discrete-time and sampled-data nonlinear systems," *Syst. Control Lett*, vol. 38, pp. 49–60, 1999.
[12] G. C. Walsh and H. Ye, "Scheduling of Networked Control Systems," *IEEE Control Systems Magazine*, 2001.
[13] M. Lemmon, T. Chantem, X. Hu, and M. Zyskowski, "On Self-Triggered Full Information H-infinity Controllers," *Hybrid Systems: Computation and Control, April*, 2007.

## IX. Appendix

*Proof:* [Theorem 6.1]
From the definitions of $\mu$, $\rho$, $\underline{\lambda}$ and $\overline{\lambda}$ we have that:

$$\mu\left(V(t)+V(t_k)\right) \leq \dot{V}(t) \leq \rho\left(V(t)+V(t_k)\right) \quad (26)$$
$$-\overline{\lambda}S(t) \leq \dot{S}(t) \leq -\underline{\lambda}S(t) \quad (27)$$

and integrating the differential inequalities above we obtain:

$$V(t+s) \leq e^{\rho s}V(t)+V(t_k)\left(e^{\rho s}-1\right), \quad (28)$$
$$V(t+s) \geq e^{\mu s}V(t)+V(t_k)\left(e^{\mu s}-1\right), \quad (29)$$
$$S(t+s) \leq e^{-\underline{\lambda} s}S(t), \quad (30)$$
$$S(t+s) \geq e^{-\overline{\lambda} s}S(t) \quad (31)$$

Denoting $\tau = n\Delta = t_k + N\Delta$, an upper bound for $V(t)$ for $t \in [\tau, \tau+\Delta[$ is then provided by:

$$V(\tau+s) \leq \begin{cases} e^{\rho s}V(\tau)+V(t_k)(e^{\rho s}-1), & s \in [0,s^*] \\ e^{\mu(s-\Delta)}V(\tau+\Delta)+V(t_k)(e^{\mu(s-\Delta)}-1), & s \in [s^*,\Delta[ \end{cases} \quad (32)$$

Hence, the maximum of $V(\tau+s)$ for $s \in [0,\Delta[$, is attained at $s = s^*$, where:

$$s^* = \frac{1}{\rho-\mu}\log\left(\frac{V(\tau+\Delta)+V(t_k)}{V(\tau)+V(t_k)}\right)+\frac{\mu\Delta}{\mu-\rho} \quad (33)$$

Thus, the maximum of $V(t)$, $t \in [n\Delta,(n+1)\Delta[$ is:

$$V(\tau+s^*) = e^{\frac{\rho\mu\Delta}{\mu-\rho}}\left(\left(V(\tau)+V(t_k)\right)^{\frac{\mu}{\mu-\rho}}\left(V(\tau+\Delta)+V(t_k)\right)^{\frac{-\rho}{\mu-\rho}}\right)-V(t_k)$$

which is monotonically increasing on $V(\tau)$, $V(\tau+\Delta)$ and $V(t_k)$, and therefore in the worst case it would be bounded by:

$$V(\tau+s^*) \leq e^{\frac{\rho\mu\Delta}{\mu-\rho}}\left(\left(S(\tau)+S(t_k)\right)^{\frac{\mu}{\mu-\rho}}\left(S(\tau+\Delta)+S(t_k)\right)^{\frac{-\rho}{\mu-\rho}}\right)-S(t_k)$$

Furthermore, we can use the bounds available for the evolution of $S(t)$ to obtain $V(\tau+s^*) \leq g(\Delta,r_{max})S(\tau+s^*)$ where:

$$g(\Delta,r_{max}) = \left(e^{\frac{\rho\mu\Delta}{\mu-\rho}}\left(e^{\overline{\lambda}s^*}+e^{\overline{\lambda}(N\Delta+s^*)}\right)^{\frac{\mu}{\mu-\rho}}\left(e^{-\underline{\lambda}(\Delta-s^*)}+e^{\overline{\lambda}(N\Delta+s^*)}\right)^{\frac{-\rho}{\mu-\rho}} - - e^{\overline{\lambda}(N\Delta+s^*)}\right) \quad (34)$$
$$\leq e^{\frac{\rho\mu\Delta}{\mu-\rho}}e^{\overline{\lambda}s^*}+e^{\overline{\lambda}(N\Delta+s^*)}\left(e^{\frac{\rho\mu\Delta}{\mu-\rho}}-1\right) \quad (35)$$

with $s^*$ as in (33). The value of $s^*$ can be further bounded to obtain a cleaner expression:

$$s^* \leq \frac{\mu\Delta}{\mu-\rho} \quad (36)$$

When $s^* \leq 0$, no maximum occurs between samples and $g(\Delta,r_{max})$ becomes 1. Otherwise, substitute (36) in (35), let $N$ take its maximum possible value $N = r_{max}-1$, and note that the bound $V(t) \leq g(\Delta,r_{max})S(t)$ holds for all $t \in [n\Delta,(n+1)\Delta[$, to conclude the proof. ■