MASKAY 7(1), Nov 2017
ISSN 1390-6712

Recibido (Received): 2017/08/06
Aceptado (Accepted): 2017/11/14

On the Performance Improvement of the Optimal-Sampling-inspired Self-Triggered Control at Implementation Stage

Mejoramiento del Desempeño del Control Auto-Disparado Inspirado en Muestreo Óptimo en la Fase de Implementación

Carlos Xavier Rosero, *Member, IEEE*, Cristina Vaca and Juan Benavides

***Abstract*—The self-triggered control includes a sampling strategy that focuses on decreasing the use of computational resources (processor and network) while preserving the same control performance as the one obtained via a controller with periodic sampling. Within this framework it has been developed recently a self-triggered control technique inspired by a sampling pattern whose optimal density minimizes the control cost, this approach is called "optimal-sampling inspired self-triggered control". However, the strategies used to implement it on microprocessor-controlled systems working under perturbation are still unclear; this paper addresses some techniques to organize and improve the implementation on actual controllers. The proposed solution comprises both the formulation of two algorithms to organize the implementation and the insertion of a closed-loop observer to deal with the perturbation that normally appears on real plants. Regarding the former, certain computationally expensive processes involved in the implementation of this control technique are treated through their replacement by lightweight polynomials fitted at design stage. Simulations and practical experiments confirm the solution is effective and there could be an open research topic concerning observation in optimal-sampling self-triggered control strategies.**

***Index Terms*—Event-driven control, aperiodic sampling, real-time embedded control systems, aperiodic observer.**

***Resumen*—El control auto-disparado incluye una atractiva estrategia de muestreo que se enfoca en disminuir el uso de recursos computacionales (procesador y red) mientras se preserva el mismo rendimiento de control que el obtenido a través de un controlador con muestreo periódico. Dentro de este marco, se ha desarrollado recientemente una técnica de control auto-disparado inspirada en un patrón de muestreo cuya densidad óptima minimiza el costo de control, se llama "control auto-disparado inspirado en muestreo óptimo". Sin embargo, las estrategias utilizadas para implementarlo en sistemas controlados por microprocesadores que funcionan bajo perturbación aún no son claras; este documento aborda algunas técnicas para organizar y mejorar la implementación sobre controladores reales. La solución propuesta comprende la formulación de dos algoritmos para organizar la implementación y también la inserción de un observador de lazo cerrado para lidiar con las perturbaciones que normalmente aparecen en las plantas reales. En cuanto a los algoritmos, ciertos procesos computacionalmente costosos implicados en su implementación son tratados mediante la sustitución por polinomios ligeros ajustados en la fase de diseño. Tanto simulaciones como experimentos confirman que la solución es efectiva y que podría haber un tema de investigación abierto relacionado con la observación en las estrategias de control auto-disparado con muestreo óptimo.**

***Palabras Claves*—Control manejado por eventos, muestreo aperiódico, sistemas de control empotrados de tiempo real, observador aperiódico.**

C.X. Rosero and J. Benavides are with the Applied Sciences Faculty, Universidad Técnica del Norte, Av. 17 de Julio 5-21, 100105 Ibarra, Ecuador (e-mail: cxrosero@utn.edu.ec).

C. Vaca is with the Health Sciences Faculty, Universidad Técnica del Norte, Av. 17 de Julio 5-21, 100105 Ibarra, Ecuador (e-mail: cvaca@utn.edu.ec).

I. Introduction

NOWADAYS controllers are implemented on digital systems consisting of microprocessors and communication networks. Among some of the alternatives that have efficient resource consumption in a nonperiodic fashion are the self-triggered control techniques (STC), initially proposed by [1]–[5]. They solve the fundamental problem of determining both optimal sampling and efficient processing/communication strategies. Each time the control task is triggered, both the time the next sampling will be performed (sampling rule) and the control action which should be maintained until this event happens, are estimated.

Several approaches aimed at solving the problem of determining optimal sampling rules in STC have been addressed recently. An optimal sampling pattern proposed in [6] inspired the approach in [7], which is analyzed in the present study. This technique describes a sampling rule that generates approximated control actions by solving the continuous-time LQR problem [8] at each sample time. The performance guarantee is based on a number of samples over a time interval with a given sampling constraint. The sampling time is calculated by the derivative of a continuous-time LQR problem and the rule produces smaller sampling times while the control action has more variation.

Though the optimal-sampling in [6] and [7] has standard cost lower than the one obtained by periodic sampling techniques, and even than other optimal-sampling approaches i.e. [9], [10], it has many weaknesses. Since the research is still new there are many open topics, among which two stand out: *(a)* clarifying and organizing the implementation on real microprocessor systems, and *(b)* adapting the approach to cases with disturbances.

DOI: 10.24133/maskay.v7i1.344

To solve problem *(a)*, in [7] both a simulated and an experimental set-ups are described. However, a deeper explanation of the paradigm that a designer of control systems should use to put this approach on a microprocessor-based system is not shown.

With regard to problem *(b)* the approach in [6] could be restated by inserting robustness to uncertainty in the approach by developing new theory, or on the other hand by using observation techniques. A settlement applying observation in presence of unknown disturbances but on a different STC strategy to that used herein, is presented in [11], [12].

To overcome problems *(a)* and *(b)*, the contribution of this paper is twofold. First, two algorithms are formulated to organize and synthesize the implementation of the approach in [7]. Second, a time-varying closed-loop observer is applied on the approach in [7] in order to make it less sensitive to noise.

The rest of the paper is organized as follows. Section II summarizes the theory on optimal-sampling-inspired self-triggered control (OSISTC). Section III presents the insertion of state observation into the self-triggered control and also the strategies to describe the implementation. Section IV shows the simulations and experiments on a selected plant. At the end, Section V performs the analysis of results and Section VI concludes the article.

II. Revisiting the optimal-sampling-inspired self-triggered control

This section summarizes the theory on OSISTC extracted from the original works in [6] and [7], and included for better understanding of the subject of study.

A. Continuous-time dynamics

Consider the linear time-invariant system (LTI) represented in continuous-time by

$$\begin{cases} \dot{x}_{(t)} = A_c x_{(t)} + B_c u_{(t)} \\ y_{(t)} = C x_{(t)} \end{cases}, \quad \text{given } x_{(0)} = x_0 \tag{1}$$

where $x_{(t)} \in \mathbb{R}^n$ is the state and $u_{(t)} \in \mathbb{R}^m$ is the continuous control input signal. $A_c \in \mathbb{R}^{n\times n}$ and $B_c \in \mathbb{R}^{n\times m}$ describe the dynamics of the system, and $C \in \mathbb{R}^{q\times n}$ is the weight matrix used to read the state; x_0 is the initial value of the state.

B. Sampling

The control input $u_{(t)}$ in (1) is piecewise constant, meaning that it remains with the same value between two consecutive *sampling instants*, thus

$$u_{(t)} = u_{(k)} \quad \forall t \in [t_{k-1}, t_k), \tag{2}$$

where the control input $u_{(k)}$ is updated at discrete times $k \in \mathbb{N}$ and the sampling instants are represented by $t_k \in \mathbb{R}$. Consecutive sampling instants are separated by *sampling intervals* τ_k, and the relationship between instants and intervals is

$$\tau_k = t_{k+1} - t_k, \quad \text{where} \quad t_k = \sum_{i=0}^{k-1} \tau_i \quad \text{for} \quad k \geq 1. \tag{3}$$

C. Discrete-time dynamics

In periodic sampling, a constant sampling interval τ is considered. The continuous-time dynamics from (1) is discretized using methods taken from [8] by

$$A_d = e^{A_c \tau}, \quad B_d = \int_0^\tau e^{A_c(\tau - t)}\, dt\, B_c, \tag{4}$$

resulting in the discrete-time LTI system

$$\begin{cases} x_{(k+1)} = A_d x_{(k)} + B_d u_{(k)} \\ y_{(k)} = C x_{(k)} \end{cases}, \quad \text{given } x_{(0)} = x_0 \tag{5}$$

where the state $x_{(k)}$ is sampled at t_k.

The location of the system poles (or eigenvalues of the dynamics matrices A_c, A_d) is fundamental to determine/change the stability of the system [8]. Poles in continuous-time p_c become poles in discrete-time p_d through

$$p_d = e^{p_c \tau}. \tag{6}$$

State-feedback control by means of pole placement requires to assign the desired closed-loop poles by hand. Nevertheless, the LQR technique allows to place the poles automatically and optimally. LQR is used by OSISTC at each t_k considering τ_k as the sampling time.

D. Linear quadratic regulator

The LQR optimal control problem allows to find an optimal input signal that minimizes the continuous-time and discrete-time infinite-horizon cost functions in (7) and (8) respectively.

$$J_c = \int_0^\infty (x_{(t)}^T Q_c x_{(t)} + 2 x_{(t)}^T S_c u_{(t)} + u_{(t)}^T R_c u_{(t)})\, dt, \tag{7}$$

$$J_d = \sum_0^\infty (x_{(k)}^T Q_d x_{(k)} + 2 x_{(k)}^T S_d u_{(k)} + u_{(k)}^T R_d u_{(k)}) \tag{8}$$

Regarding dimensionality in (7) and (8), the weight matrices $Q_c, Q_d \succeq 0 \in \mathbb{R}^{n\times n}$ are positive semi-definite, $R_c, R_d \succ 0 \in \mathbb{R}^{m\times m}$ are positive definite, and $S_c, S_d \in \mathbb{R}^{n\times m}$. Refer to [13] to know about the transformation of the weight matrices from their continuous forms Q_c, R_c, S_c to their discrete versions Q_d, R_d, S_d.

E. Optimal sampling-inspired self-triggered control

The approach in [7] involves designing both a sampling rule as a piecewise control input, such that the LQR cost is minimized. Additionally, the periodicity of execution of the controller is relaxed so that the consumption of resources is diminishing. Then, the *sampling rule* is

$$\tau_k = \tau_{max} \frac{1}{\frac{\tau_{max}}{\eta} |K_c(A_c + B_c K_c) x_{(k)}|^\alpha + 1} \tag{9}$$

where an upper bound on the sampling intervals is given by τ_{max}; similarly η modifies the degree of density of the sampling sequence (smaller η yields denser sampling instants and vice versa). By minimizing the continuous-time cost function (7) an optimal continuous-time feedback gain K_c is found once. According to [6] and [7] there exist optimal settings for the exponent $\alpha \geq 0$ which influences the density

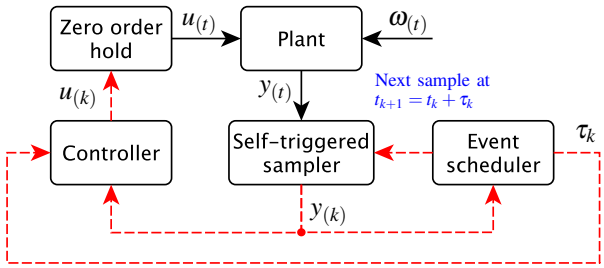

Fig. 1. Original architecture of the self-triggered feedback control. Solid lines denote continuous-time signals and dashed lines denote signals updated only at each sampling time.

of the samples set; with $\alpha = 0$ the sampling becomes regular (periodic).

Additionally, from [7] the *piecewise optimal control signal* expressed in linear state feedback form is

$$u_{(k)} = -K_{d(\tau_k)} x_{(k)}, \tag{10}$$

where $K_{d(\tau_k)}$ is calculated at each controller execution τ_k. Its value is obtained by solving the discrete-time LQR problem (8) considering a fixed sampling period τ_k.

III. On the Implementation of OSISTC

The model of the proposed approach as well as the guidelines for its implementation are explained in this section. This corresponds to the main contribution of the work.

A. Original OSISTC architecture

Figure 1 is used to ensure better understanding of the original OSISTC scheme. In this configuration the output of the plant $y_{(t)}$ is sampled by the *self-triggered sampler* at each τ_k; the measured state $y_{(k)}$ is used by both the *event scheduler* and the *controller*. The event scheduler is responsible for calculating when the next sampling time t_{k+1} will be executed by means of (9). The controller computes the control action using both (2) and (10). The control input $u_{(k)}$ is kept constant along the entire sampling interval τ_k in a zero-order hold manner.

In the same Figure 1, the bounded exogenous disturbances $\omega_{(t)}$ are not treated in any way, causing noisy states and affecting the system performance. With respect to both the event scheduler and the controller, they base their procedures on the measured state $y_{(k)}$ (or on the error $e_{(k)}$ when there is a reference). Thus, the insertion of noise into the states leads to the emergence of uncertainty in both the linear piece-wise control $u_{(k)}$ and the sampling interval τ_k.

B. Discrete-time observer

An observer constitutes a computer copy of the observed dynamic system (5) whose predicted states $\hat{x}_{(k)}$ converge to the real states $x_{(k)}$ by reducing the observer's output error $\widetilde{e}_{(k)}$. The discrete-time Luenberger observer proposed in [14] and shown in Fig. 2 is a state estimator which works properly in presence of unknown disturbances; see [8] for better understanding. Then, the system in (5) is reformulated as

$$\begin{cases} \hat{x}_{(k+1)} = A_d \hat{x}_{(k)} + B_d u_{(k)} + L_d \left(y_{(k)} - \hat{y}_{(k)} \right) \\ y_{(k)} = C x_{(k)} \end{cases}, \tag{11}$$

where $\hat{x}_{(k+1)} \in \mathbb{R}^n$ is the state estimate and $\hat{y}_{(k)} \in \mathbb{R}^q$ is the output estimate. $L_d \in \mathbb{R}^{n \times q}$ is the observer gain matrix.

In (11), if the pair (A_d, C) is completely observable, the dual system (A'_d, B'_d, C', D') is completely reachable. Then, an observer gain matrix L_d for the dual system can be designed and the eigenvalues (poles) of $(A'_d - C'L_d)$ can be arbitrarily placed [14]. Consider that the eigenvalues of a matrix $(A'_d - C'L_d)$ are equal to the eigenvalues of its transpose $(A_d - L'_d C)$.

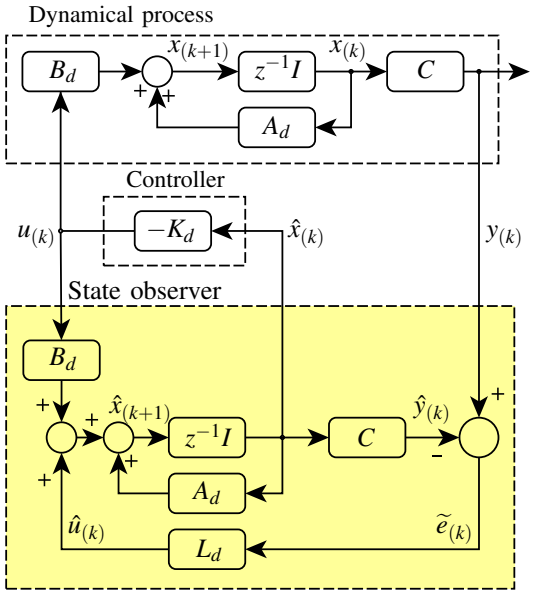

Fig. 2. Discrete-time Luenberger state observer

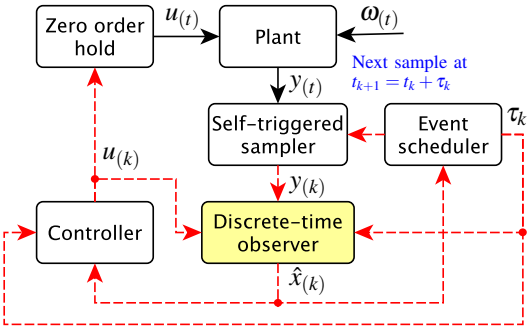

Fig. 3. Proposed architecture of the self-triggered feedback control with observation. Solid lines denote continuous-time signals and dashed lines denote signals updated only at each sampling time.

C. Proposed OSISTC architecture based on observer

Figure 3 shows the proposed self-triggered architecture in which the use of a *discrete-time observer* stands out to deal with noise $\omega_{(t)}$. Assuming that the pair $(A_{d(\tau_k)}, C)$ is observable along the set of all possible sampling intervals τ_k, the eigenvalues of $(A_{d(\tau_k)} - L'_{d(\tau_k)} C)$ can be placed arbitrarily

[14]. Notice that the dynamics now depends on τ_k because $A_{d(\tau_k)}$ is a time varying matrix. The discrete poles in (6) are also dependent on the sampling interval τ_k, as in

$$p_{d_{(\tau_k)}} = e^{p_c \tau_k}. \tag{12}$$

In this context the observer needs to solve a new pole placement at each execution, since the discrete dynamics matrices and the discrete poles are dependent on the sampling interval τ_k. This implies that the observer has a different gain matrix $L_{d(\tau_k)}$ at each execution. Then, considering the changing dynamics, the system in (11) becomes

$$\begin{cases} \hat{x}_{(k+1)} = A_{d(\tau_k)}\hat{x}_{(k)} + B_{d(\tau_k)}u_{(k)} + L_{d(\tau_k)}\left(y_{(k)} - \hat{y}_{(k)}\right) \\ y_{(k)} = Cx_{(k)} \end{cases}, \tag{13}$$

where $A_{d(\tau_k)}$ and $B_{d(\tau_k)}$ are discretized matrices for a sampling interval τ_k, $u_{(k)}$ is the linear piecewise control action calculated by (10), and $L_{d(\tau_k)} \in \mathbb{R}^{n\times q}$ is the gain matrix of the sampling-dependent observer.

D. Problems considered

There are several drawbacks in assembling both the OSISTC controller and the time-varying observer on a real-time control system.

The first issue has to do with calculation of the *controller gain matrix* $K_{d(\tau_k)}$ in (10) by solving the problem in (8) through recursive computation of the discrete algebraic Ricatti equation (DARE) until convergence [8]. The second issue is the pole placement solved by Ackermann's formula [15] in order to obtain the *observer gain matrix* $L_{d(\tau_k)}$.

Both processes are computationally expensive and must be performed at each controller execution. If the execution time of the control task is too close to the minimum sampling interval, undesirable effects such as jitter could appear [16]. Particularly in OSISTC, the worst case scenario comes out when the rate of change of the control action is maximal, causing a highest density in the emergence of samples (minimum τ_k).

E. Set of sampling intervals T

The *set of sampling intervals* $T \in \mathbb{R}^{1\times s}$ within a closed interval $[\tau_{min};\ \tau_{max}]$ is

$$T = \{\tau_{min}, \tau_{min} + \tau_g, \tau_{min} + 2\tau_g, \cdots, \tau_{max}\}, \tag{14}$$

where τ_g is the *sampling granularity* defined as the least increase-unit for the sampling intervals. Each element of the set can be addressed in this way

$$\tau_h = T_{[h]} \quad \forall h \in \mathbb{N} : 1 \le h \le s \tag{15}$$

being s the length of T.

The minimum and maximum sampling times, τ_{min} and τ_{max}, as well as τ_g are chosen following the conditions detailed in [7]

$$\begin{cases} \dfrac{\beta^\alpha}{\eta} \le \dfrac{1}{\tau_{min}} - \dfrac{1}{\tau_{max}}, \quad \beta := \sup_{x\in X} |K_c(A_c + B_cK_c)x| \\ \tau_{RTOS} \le \tau_g, \end{cases} \tag{16}$$

where X is the entire state space taken from the physical constraint of the plant, and τ_{RTOS} is the sampling granularity of the real-time operating system (RTOS) in which the technique will be implemented.

F. Strategy to calculate the controller gain matrix $K_{d(\tau_k)}$

The gain $K_{d(\tau_h)}$ is calculated by brute force for each h^{th} element of the set T in (14) by the discrete-time LQR problem (8). Therefore, we obtain a total of s controller gain matrices $K_{d(\tau_h)} \in \mathbb{R}^{m\times n}$ that have the form

$$\begin{aligned} K_{d(\tau_h)} &= dlqr\left(A_{d(\tau_h)}, B_{d(\tau_h)}, Q_{d(\tau_h)}, R_{d(\tau_h)}\right) \\ K_{d(\tau_h)} &= \begin{bmatrix} kd_{11}^{(\tau_h)} & \cdots & kd_{1n}^{(\tau_h)} \\ \vdots & \ddots & \vdots \\ kd_{m1}^{(\tau_h)} & \cdots & kd_{mn}^{(\tau_h)} \end{bmatrix}. \end{aligned} \tag{17}$$

Regrouping the elements of all gain matrices according to their position yields a group S_{K_d} that is $m \cdot n$ training sets long, where m and n are the dimensions of inputs and states respectively, then

$$\begin{aligned} S_{K_d} = \{&[kd_{11}^{(\tau_1)}, \cdots, kd_{11}^{(\tau_s)}], \cdots, [kd_{1n}^{(\tau_1)}, \cdots, kd_{1n}^{(\tau_s)}], \cdots, \\ &[kd_{m1}^{(\tau_1)}, \cdots, kd_{m1}^{(\tau_s)}], \cdots, [kd_{mn}^{(\tau_1)}, \cdots, kd_{mn}^{(\tau_s)}]\}. \end{aligned} \tag{18}$$

Each training set in (18) is defined in $\mathbb{R}^{1\times s}$ and used to perform a *polynomial curve fitting* in order to find the coefficients θ of the d-degree polynomials $K_{ij}(\tau_k)$. Therefore, we have a total of $m \cdot n$ polynomials each one following the form

$$K_{ij}(\tau_k) = \theta_1^{(ij)}\tau_k^d + \theta_2^{(ij)}\tau_k^{d-1} + \cdots + \theta_d^{(ij)}\tau_k + \theta_{d+1}^{(ij)}, \tag{19}$$

where superscript (ij) indicates the belonging of coefficients $\theta^{(ij)}$ to polynomial $K_{ij}(\tau_k)$; i-row and j-column show the position of polynomials into the gain matrix. Note the change of τ_k instead of τ_h since the former is the current sampling interval calculated online through equation (9) on a real controller. Thus, (17) to (19) become

$$K_{d(\tau_k)} = \begin{bmatrix} K_{11}(\tau_k) & \cdots & K_{1n}(\tau_k) \\ \vdots & \ddots & \vdots \\ K_{m1}(\tau_k) & \cdots & K_{mn}(\tau_k) \end{bmatrix}, \tag{20}$$

where

$$\begin{aligned} K_{11}(\tau_k) &= \theta_1^{(11)}\tau_k^d + \theta_2^{(11)}\tau_k^{d-1} + \cdots + \theta_d^{(11)}\tau_k + \theta_{d+1}^{(11)} \\ K_{1n}(\tau_k) &= \theta_1^{(1n)}\tau_k^d + \theta_2^{(1n)}\tau_k^{d-1} + \cdots + \theta_d^{(1n)}\tau_k + \theta_{d+1}^{(1n)} \\ K_{m1}(\tau_k) &= \theta_1^{(m1)}\tau_k^d + \theta_2^{(m1)}\tau_k^{d-1} + \cdots + \theta_d^{(m1)}\tau_k + \theta_{d+1}^{(m1)} \\ K_{mn}(\tau_k) &= \theta_1^{(mn)}\tau_k^d + \theta_2^{(mn)}\tau_k^{d-1} + \cdots + \theta_d^{(mn)}\tau_k + \theta_{d+1}^{(mn)}. \end{aligned}$$

G. Strategy to calculate the observer gain matrix $L_{d(\tau_k)}$

It is a process similar to that described in subsection III-F. All possible observer gain matrices $L_{(\tau_h)}$ are evaluated offline as functions of sampling intervals τ_h.

The error dynamics of the observer is given by the poles of $\left(A_{d(\tau_h)} - L_{d(\tau_h)}C\right)$. A rule of thumb considers to place the observer poles five to ten times farther to the left of s-plane than the dominant poles of the system.

By computing $A_{d(\tau_h)}$ through (4), assigning statically the continuous-time poles and discretizing them by (6) in order to have the vector $P_{d(\tau_h)}$, and finally considering C which remains

constant, we obtain a total of s observer gain matrices $L_{d(\tau_h)} \in \mathbb{R}^{n\times q}$ by the poles placement method in [15] with the form

$$L_{d(\tau_h)} = ackermann\left({A_{d(\tau_h)}}^T, C^T, {P_{d(\tau_h)}}^T\right)$$
$$L_{d(\tau_h)} = \begin{bmatrix} ld_{11}^{(\tau_h)} & \cdots & ld_{1q}^{(\tau_h)} \\ \vdots & \ddots & \vdots \\ ld_{n1}^{(\tau_h)} & \cdots & ld_{nq}^{(\tau_h)} \end{bmatrix}. \tag{21}$$

Using the same regrouping criterion as in (18) a group S_{L_d}, $n\cdot q$ training sets long, is obtained

$$S_{L_d} = \{[ld_{11}^{(\tau_1)},\cdots,ld_{11}^{(\tau_s)}],\cdots,[kd_{1q}^{(\tau_1)},\cdots,kd_{1q}^{(\tau_s)}],\cdots, [kd_{n1}^{(\tau_1)},\cdots,kd_{n1}^{(\tau_s)}],\cdots,[kd_{nq}^{(\tau_1)},\cdots,kd_{nq}^{(\tau_s)}]\}. \tag{22}$$

Subsequently a total of $n\cdot q$ polynomials are calculated with the form

$$L_{ij}(\tau_k) = \theta_1^{(ij)}\tau_k^d + \theta_2^{(ij)}\tau_k^{d-1} + \cdots + \theta_d^{(ij)}\tau_k + \theta_{d+1}^{(ij)}, \tag{23}$$

such as in (19). Finally it is obtained

$$L_{d(\tau_k)} = \begin{bmatrix} L_{11}(\tau_k) & \cdots & l_{1q}(\tau_k) \\ \vdots & \ddots & \vdots \\ L_{n1}(\tau_k) & \cdots & l_{nq}(\tau_k) \end{bmatrix}, \tag{24}$$

where

$$L_{11}(\tau_k) = \theta_1^{(11)}\tau_k^d + \theta_2^{(11)}\tau_k^{d-1} + \cdots + \theta_d^{(11)}\tau_k + \theta_{d+1}^{(11)}$$
$$L_{1q}(\tau_k) = \theta_1^{(1q)}\tau_k^d + \theta_2^{(1q)}\tau_k^{d-1} + \cdots + \theta_d^{(1q)}\tau_k + \theta_{d+1}^{(1q)}$$
$$L_{n1}(\tau_k) = \theta_1^{(n1)}\tau_k^d + \theta_2^{(n1)}\tau_k^{d-1} + \cdots + \theta_d^{(n1)}\tau_k + \theta_{d+1}^{(n1)}$$
$$L_{nq}(\tau_k) = \theta_1^{(nq)}\tau_k^d + \theta_2^{(nq)}\tau_k^{d-1} + \cdots + \theta_d^{(nq)}\tau_k + \theta_{d+1}^{(nq)}.$$

Then, on each execution of the actual controller, after calculating the next sampling interval τ_k via (9), each element of the observer gain matrix is computed through a different polynomial in the matrix $L_{d(\tau_k)}$.

H. Implementation guidelines

Through Algorithm 1 what was said in subsections III-F and III-G is summarized; this program can be performed offline by any numerical computing programming language. Algorithm 2 shows how to implement OSISTC on any processor with reduced performance features.

Data: $A_c, B_c, C, Q_c, R_c, \tau_{min}, \tau_{max}, \tau_g, \alpha, \beta, \eta, P_c$
Result: $K_c, K_{d(\tau_k)}, L_{d(\tau_k)}$
$K_c = f(A_c, B_c, Q_c, R_c)$ by continuous LQR (7);
$T = f(\tau_{min}, \tau_{max}, \tau_g)$ by (14) and (16);
for $\tau_h \in T$ **do**
  Compute $A_{d(\tau_h)}, B_{d(\tau_h)}, Q_{d(\tau_h)}, R_{d(\tau_h)}$ by (4);
  $K_{d(\tau_h)} = f\left(A_{d(\tau_h)}, B_{d(\tau_h)}, Q_{d(\tau_h)}, R_{d(\tau_h)}\right)$ by discrete LQR (17);
  Compute $P_{d(\tau_h)}$ by (6);
  $L_{d(\tau_h)} = f\left(A_{d(\tau_h)}^T, C^T, P_{d(\tau_h)}\right)$ by Ackermann (21);
end
Create S_{K_d} based on all $K_{d(\tau_h)}$ by (18);
Create S_{L_d} based on all $L_{d(\tau_h)}$ by (22);
for $i \le m$ *and* $j \le n$ **do**
  $K_{ij}(\tau_k) = f\left(S_{K_d(i\cdot j)}\right)$ by polynomial curve fitting;
end
for $i \le n$ *and* $j \le q$ **do**
  $L_{ij}(\tau_k) = f\left(S_{L_d(i\cdot j)}\right)$ by polynomial curve fitting;
end
Create $K_{d(\tau_k)}$ by ordering all $K_{ij}(\tau_k)$ as in (20);
Create $L_{d(\tau_k)}$ by ordering all $L_{ij}(\tau_k)$ as in (24);

Algorithm 1: Offline design

Data: $A_c, B_c, C, K_c, \tau_{max}, \tau_g, \alpha, \beta, \eta, K_{d(\tau_k)}, L_{d(\tau_k)}$
Result: $\tau_k, u_{(k)}$
Initialization of hardware, RTOS and variables;
$x_{(k)} := read\_input()$;
$\tau_k := f\left(\tau_{max}, K_c, \alpha, \beta, \eta, \hat{x}_{(k)}\right)$ by (9);
Set RTOS to trigger next time after τ_k;
Calculate $K_{d(\tau_k)}$ by (20);
$u_{(k)} := f\left(K_{d(\tau_k)}, \hat{x}_{(k)}\right)$ by (10);
Set actuator with $u_{(k)}$;
Calculate $L_{d(\tau_k)}$ by (24);
Compute $A_{d(\tau_k)}, B_{d(\tau_k)}$ by (4);
$\hat{x}_{(k)} := f\left(A_{d(\tau_k)}, B_{d(\tau_k)}, K_{d(\tau_k)}, L_{d(\tau_k)}, \hat{x}_{(k)}, x_{(k)}\right)$ by (13)

Algorithm 2: Online implementation

IV. Results

An experiment on a real plant is presented in order to illustrate the theory introduced in the previous section.

A. Plant

The experimental plant with form (1) is the same electronic double integrator circuit as the one used in [7], so advise with that document for further information. The state space representation is

$$A_c = \begin{bmatrix} 0 & -23.81 \\ 0 & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ -23.81 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{25}$$

In Table I most important configurations used to design both controller and observer are detailed. These values have been based on recommendations from the literature in [7]. Note that the poles of the observer have been chosen to be fast enough so that they do not slow down the dynamics of the plant

TABLE I
Experiment settings

Symbol	Description	Value
Q_c	States weighting matrix	$0.0025\cdot\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
R_c	Control weighting matrix	0.1
τ_{min}	Minimum sampling time	20ms
τ_{max}	Maximum sampling time	60ms
τ_g	Sampling granularity	1ms
α	Density of the samples set	0.667
β	-	9.4
η	Density degree of the sampling seq.	0.11
P_c	Observer continuous-time poles	$[-50+2j, -50-2j]$

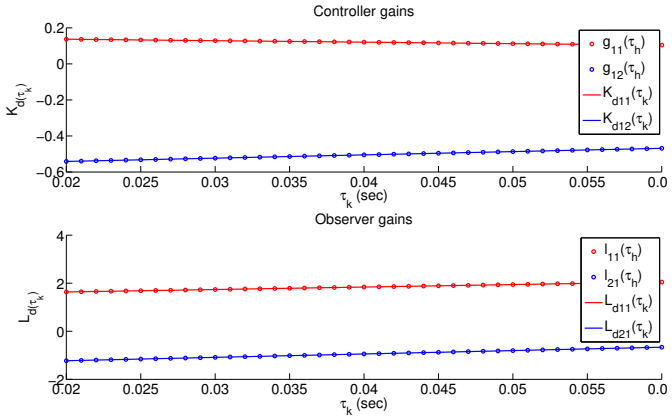

Fig. 4. Gains polynomial fitting: controller (top), and observer (bottom)

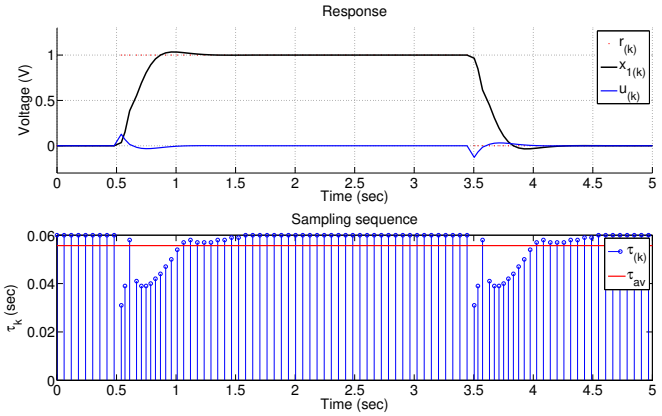

Fig. 5. Behavior of OSISTC in simulation

B. Controller and observer

Algorithm 1 has been followed step by step to perfor the offline design. In Fig. 4 the gains of both controller ar observer evaluated for the set of sampling intervals, are show by circles. Likewise, fitted curves (continuous lines) rough describe the behaviour of these gains. Additional numeric results are summarized next:

- Continuous-time feedback gain

$$K_c = [0.1581 \quad -0.5841].$$

- Controller gain matrix which consists of two polynomial as in

$$K_{d(\tau_k)} = [-0.8203\tau_k + 0.1533 \quad 1.819\tau_k - 0.5778].$$

- Observer gain matrix formed by two polynomials, as in

$$L_{d(\tau_k)} = [10.2521\tau_k + 1.431 \quad 14.094\tau_k - 1.5081]^T.$$

C. Implementation on a processor

The development platform comprises the digital signal controller (DSC) dsPIC33FJ256MC710A from Microchip which internally runs the Erika real-time kernel. To learn more about this environment, it is recommended to see the original work in [18] and its references, and the same implementation in [7].

The self-triggered controller uses rule (9) to calculate when it will activate itself next time; this value is used to set the RTOS to trigger the next sampling instant. Other functions of the controller are to read the states of the plant $x_{(k)}$ through the DSC analog/digital converter, to estimate the states $\hat{x}_{(k)}$ through the observer, and to compute the control action $u_{(k)}$ which is applied directly to the plant via pulse width modulation (PWM).

Algorithm 2 has been used to perform the implementation that works on the microcontroller. To calculate the controller gain matrix, two first-degree polynomials that are functions of τ_k are represented as $K_{11}(\tau_k)$ and $K_{12}(\tau_k)$, grouped into $K_{d(\tau_k)}$. This is done instead of minimizing DARE. Finally, the observer gain matrix is replaced by a pair of first-degree polynomials $L_{11}(\tau_k)$ and $L_{21}(\tau_k)$, framed within $L_{d(\tau_k)}$. This is done instead of using a pole placement method i.e. Ackermann.

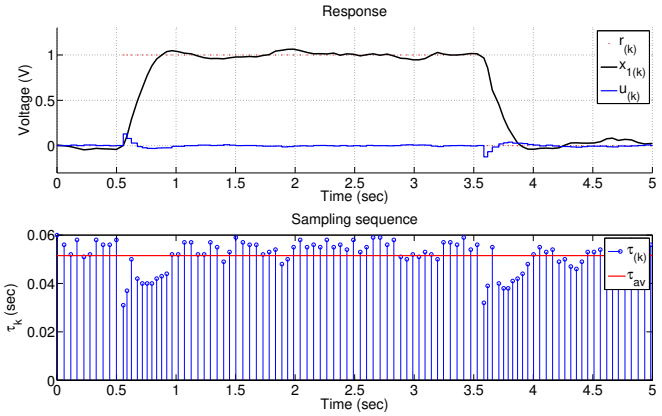

Fig. 6. Implementation of OSISTC with no observer [7]

V. Discussion

Figures 5 to 7 show the states evolution and the sampling pattern both in simulation and actual implementations when OSISTC is subjected to follow a reference. The establishment times, overshoots, and steady-state errors are almost similar for all cases.

The sampling intervals in the simulation (Fig. 5) lie within the range [31; 60]ms, in the real system without observer (Fig. 6) are within [32; 59]ms, and in the real system with observer (Fig. 7) within [31; 60]ms. The red lines in the sampling sequence graphs correspond to the average sampling, explained later through equation (26).

The observer in Fig. 7 provides noise-free states that stabilize the triggering of sampling periods τ_k at the same time. The implementation without observer in Fig. 6 tends to shake in steady state since its states have noise, which causes the oscillation of the triggering of sampling periods.

The average sampling metric τ_{av} in [7] establishes

$$\tau_{av} = \frac{1}{N}\sum_{k=0}^{N-1}\tau_k, \tag{26}$$

where N is the number of samples within the experiment/simulation time; larger values of τ_{av} indicate less resource utilization. In the simulation $\tau_{avS} = 55.7$ms, in the implementation without observer $\tau_{avNO} = 51.3$ms, and in the

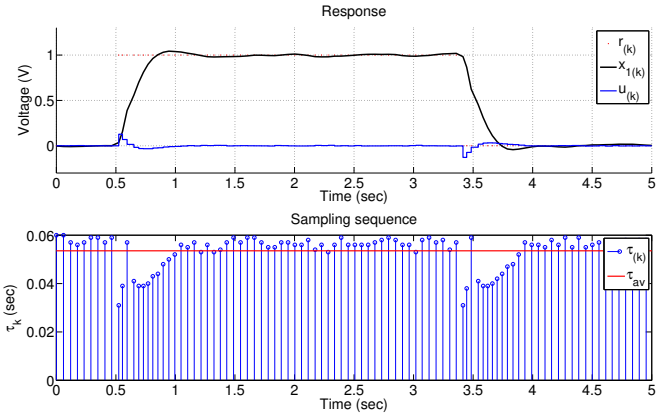

Fig. 7. Implementation of OSISTC with observer

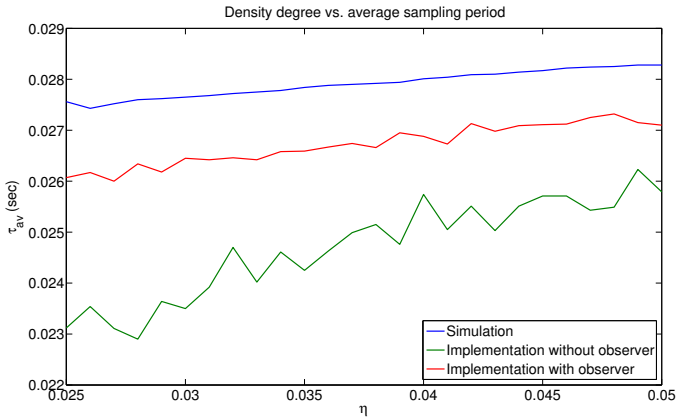

Fig. 8. Comparison of the average sampling period among simulations and implementations with and without observer, with $\tau_{min} = 3$ms, $\tau_{max} = 30$ms and $\eta \in \{0.025, 0.026, \cdots, 0.050\}$

implementation with observer $\tau_{avO} = 54.1$ms. The average sampling τ_{avNO} is less than τ_{avO}, which means that the implementation with observer has better performance than the implementation presented in [7] which has no observer, since it uses less processing resources.

Figure 8 shows how the sampling average periods behave when the density degree is changed as long as the guarantee in (16) is maintained. The behavior $\tau_{avO} > \tau_{avNO}$ is recurrent, which allows corroborating the results obtained above.

VI. Conclusions

Some techniques applied at the implementation stage to improve the performance of the method in [7] were presented. A polynomial fitted offline to calculate the discrete-time controller gains, was used to replace the online discrete-time LQR problem. A time-varying closed-loop observer has been implemented by polynomial fitting techniques while avoiding the online use of the Ackermann pole placement method.

Simulations and experiments have been confirmed the solution is effective and there could be an open research topic regarding observation techniques in OSISTC. There are interesting performance measures in the literature which could become future work for this study; metrics from [7] and [10] would allow further evaluation on a real system. A comparison between the implementation with and without observer can be made to determine the true contribution of the latter.

Acknowledgment

This work has been partially supported by the University Center for Scientific and Technological Research (CUICYT) of Universidad Técnica del Norte.

References

[1] M. Velasco, J.M. Fuertes, and P. Martí, "The Self Triggered Task Model for Real-Time Control Systems," in *Proc. RTSS*, Cancun, Mexico, Dec., 2003, pp. 67-70.

[2] A. Anta and P. Tabuada, "To Sample or Not to Sample: Self-Triggered Control for Nonlinear Systems," *IEEE Trans. Autom. Control*, vol. 55, no. 9, pp. 2030-2042, Sept. 2010.

[3] M. Mazo Jr., A. Anta and P. Tabuada, "An ISS Self-Triggered Implementation of Linear Controllers," in Automatica, vol. 46, no. 8, pp. 1310-1314, Aug. 2010.

[4] J. Almeida, C. Silvestre and A.M. Pascoal, "Self-Triggered Output Feedback Control of Linear Plants," in *Proc. ACC*, San Francisco, CA, USA, June-July, 2011, pp. 2831-2836.

[5] A. Molin and S. Hirche, "On the Optimality of Certainty Equivalence for Event-Triggered Control Systems", in *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 470-474, Feb. 2013.

[6] E. Bini and G.M. Buttazzo, "The Optimal Sampling Pattern For Linear Control Systems," in *IEEE Trans. Autom. Control*, vol. 59, no. 1, pp. 78-90, Jan. 2014.

[7] M. Velasco, P. Martí and E. Bini, "Optimal-Sampling-inspired Self-Triggered Control," in *Int. Conf. EBCCSP*, Krakow, Poland, June, 2015, pp. 1-8.

[8] K.J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd ed., Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[9] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels, "Self-Triggered Linear Quadratic Control", in *Automatica*, vol. 50, no. 4, pp. 1279-1287, Apr. 2014.

[10] C. Rosero, C. Vaca, L. Tobar and F. Rosero, "Performance of Self-Triggered Control Approaches," in *Enfoque UTE*, vol. 8, no. 2, pp. 107-120, Mar. 2017.

[11] J. Almeida, C. Silvestre and A.M. Pascoal, "Observer Based Self-Triggered Control of Linear Plants with Unknown Disturbances," in *Proc. ACC*, Montreal, Canada, June, 2012, pp. 5688-5693.

[12] X. Wang and M.D. Lemmon, "Self-Triggering Under State-Independent Disturbances", in *IEEE Trans. Autom. Control*, vol. 55, no. 6, pp. 1494-1500, June 2010.

[13] W.F. Arnold and A.J. Laub, "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations," in *Proc. IEEE*, vol. 72, no. 12, pp. 1746-1754, Dec. 1984.

[14] D. Luenberger, "An Introduction to Observers," in *IEEE Trans. Autom. Control*, vol. 16, no. 6, pp. 596-602, Dec. 1971.

[15] J. Ackermann, "On the Synthesis of Linear Control Systems with Specified Characteristics," in *Automatica*, vol. 13, no. 1, pp. 89-94, Jan. 1977.

[16] F. Paez, R. Cayssials, J. Urriza, E. Ferro and J. Orozco, "Frequency Domain Analysis of a RTOS in Control Applications," in *Cong. CASE*, Buenos Aires, Argentina, Aug., 2016, pp. 21-26.

[17] S.A. Dyer and X. He, "Least-squares fitting of data by polynomials," in *IEEE Instrum. Meas. Mag.*, vol. 4, no. 4, pp. 46-51, Dec. 2001.

[18] C. Lozoya, P. Martí, M. Velasco, J. Fuertes and E. Martin, "Resource and Performance Trade-offs in Real-Time Embedded Control Systems," in *J. Real-Time Systems*, vol. 49, no. 3, pp. 267-307, May 2013.