Implementation Guidelines for the Optimal-Sampling-inspired Self-Triggered Control

Juan Benavides and Carlos Xavier Rosero
Universidad Técnica del Norte, 100102, Ibarra, Ecuador
Email: {jpbenavidesp,cxrosero}@utn.edu.ec

Abstract—**In a self-triggered scenario the controller is allowed to choose when the next sampling time should occur and which control action will be maintained until that happens. This emerging control type is aimed at decreasing the use of computational resources (processor and network) while preserving the same performance control 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 real microprocessor-controlled systems working under disturbances are still unclear, then this paper addresses some implementation guidelines to make this theory applicable to actual controllers. The proposed solution comprises a new conception of this technique based on a closed-loop observer as well as making strategies for implementation of computationally expensive processes 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 techniques.**

I. Introduction

Nowadays controllers are implemented on digital systems made up of microprocessors and communication networks. Their general functions are sampling the states, calculating the control action and finally performing it over the plant along the runtime.

In a periodic fashion the execution of various control tasks guaranteeing closed-loop performance, needs faster microprocessors and/or higher-speed networks. Some alternatives for efficient resource consumption within the nonperiodic control paradigm are the event-triggered control (ETC), and the self-triggered control (STC). They solve the fundamental problem of determining both optimal sampling and processing/communication strategies.

In STC, proposed by [1] and [2], each time the control task is triggered, both the time the next sampling should be performed (sampling rule) and the control action which should be maintained until this event happens, are estimated. The design of the sampling rule follows several strategies handled by control and/or resource constraints. Within these paradigms the optimal-sampling rules are oriented to optimal control performance, in the sense of producing a sequence of control inputs which are capable to deliver a control cost less than an optimal control cost. Consider the latter as the one corresponding to the minimum cost produced by the periodic discrete-time linear quadratic regulator (LQR) in [3].

Several approaches aimed at solving the problem of determining optimal sampling rules have been addressed recently. An optimal sampling pattern proposed in [4] inspired the approach in [5] which describes a sampling rule that generates approximated control actions by solving the continuous-time LQR problem 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 [4] and [5] has standard cost lower than the one obtained by periodic sampling techniques, and even than other optimal-sampling techniques, it has still many weaknesses. Since research is still new, there are many open topics among which stand out two: *(a)* making rules for implementation on actual microprocessor systems, and *(b)* adapting the approach to cases with disturbances.

To solve the problem *(a)* in [5] both a simulated and an experimental set-up are described. They consider a plant to be controlled and a real-time kernel over certain hardware, however a deeper explanation of the paradigm that a designer of control systems should use to write code on a microprocessor system is not shown.

With regard to the solution to problem *(b)* the approach in [4] could be restated by inserting disturbances in the model and developing new theory, or otherwise assuming known or unknown disturbances and applying observation techniques. A settlement is presented in [6] where the control of a linear plant in presence of unknown disturbances is described considering a different self-triggered strategy to that used herein.

To overcome the problems mentioned above, the contribution of this paper is twofold. On the one hand, the implementation of the algorithm from [5] on microprocessor-based controllers, replacing the online use of the LQR problem by a lightweight polynomial fitted at design stage (offline). On the other hand the application of a time-varying closed-loop observer on the approach in [5], in order to make it less sensitive to noise.

The rest of the paper is organized as follows. Section II introduces certain theoretical knowledge involving optimal-sampling-inspired self-triggered control (OSISTC) as well as state observation. Section III presents strategies and models used to develop the approach, and also describes guidelines for implementing the solution. Section IV evaluates the numerical results obtained. At the end, section V concludes the article.

II. Background

A. Continuous-time dynamics

Consider the continuous-time linear time-invariant system (LTI) described in state space representation 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. Variables m, n and q denote the dimensions of states, inputs and outputs respectively.

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

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

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 thus the relationship between instants and intervals is described by

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

C. Discrete-time dynamics

In periodic sampling a constant sampling interval τ is considered, then the continuous-time dynamics from (1) is discretized using methods taken from [3] 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 following 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 matrix when state space representation is used) is fundamental to determine/change the stability of the system. Poles in continuous-time p_c become poles in discrete-time p_d through

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

technique also taken from [3].

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.

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

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

Regarding dimensionality in (7) and (8), 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 $S_c, S_d \in \mathbb{R}^{n\times m}$.

E. Optimal sampling-inspired self-triggered control

The approach in [5] involves designing both a sampling rule as a piecewise control input at the same time, such that the LQR cost is minimized. Additionally the execution periodicity of the controller is relaxed thereby decreasing the consumption of resources. 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 viceversa). Minimizing the continuos-time cost function (7) an optimal continuous-time feedback gain K_c is found once. According to [4] and [5] there exist optimal settings for the exponent $\alpha \geq 0$ which influences the density of the samples set; with $\alpha = 0$ the sampling becomes regular (periodic).

Additionally, from [5] 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. Its value is obtained by solving the discrete-time LQR problem with (8) considering a fixed sampling period τ_k.

F. Discrete-time observer

To implement any state feedback controller like that seen in (10) knowledge on the complete state vector $x_{(k)}$ is required, however often sensors only provide measurements of the output $y_{(k)}$, without considering disturbances as shown in (5). Thus, it appears the need to reconstruct missing state-variable information necessary for control; that is achieved by estimating the state $x_{(k)}$ only measuring the output $y_{(k)}$ and knowing the control input $u_{(k)}$ applied to the system.

An observer constitutes a computer copy of the dynamic system fed in parallel by the same signal $u_{(k)}$; for instance the Luenberger observer [7] is a state estimator which works properly in presence of unknown disturbances. See [3] for better understanding.

The Luenberger observer in Fig. 1 is widely used to correct the estimation $\hat{x}_{(k+1)}$ with a feedback from the observer's output error $\tilde{e}_{(k)}$; the deduced equation is

$$\hat{x}_{(k+1)} = A_d \hat{x}_{(k)} + B_d u_{(k)} + L_d \left[y_{(k)} - \hat{y}_{(k)} \right], \tag{11}$$

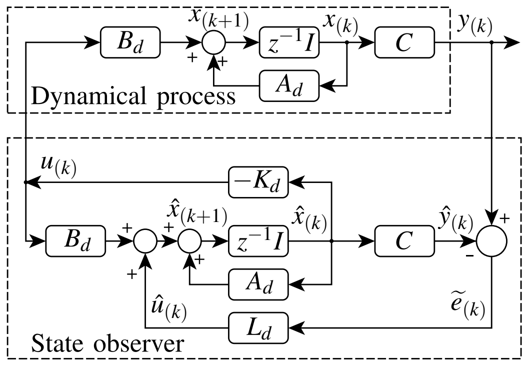

Fig. 1. Discrete-time Luenberger state observer

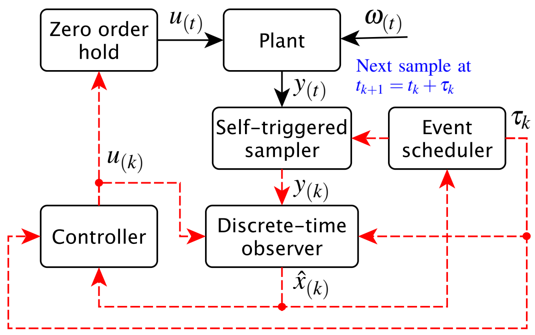

Fig. 2. Architecture of the self-triggered feedback control system with observation. Solid lines denote continuous-time signals and dashed lines denote signals updated only at each sampling time.

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; A_d, B_d, and C have the appropriate dimensions and are explained in greater detail through (4) to (5). $L_d \in \mathbb{R}^{n \times q}$ is the observer gain matrix.

III. Implementation Guidelines

A. Optimal-sampling-inspired self-triggered control in practice

Theory on OSISTC does not consider noise ($\omega_{(t)}$ in Fig. 2), therefore it is necessary to use some additional strategies for its practical application on a plant under perturbations.

Figure 2 shows the proposed self-triggered architecture in which the use of a *discrete-time observer* stands out. This configuration has as disadvantage that 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 at each execution. Considering the changing dynamics the system in (11) becomes

$$\hat{x}_{(k+1)} = A_{d(\tau_k)}\hat{x}_{(k)} + B_{d(\tau_k)}u_{(k)} + L_{d(\tau_k)}\left[y_{(k)} - \hat{y}_{(k)}\right], \quad (12)$$

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 from self-triggered strategy in (10), and $L_{d(\tau_k)} \in \mathbb{R}^{n \times q}$ is the gain matrix of the sampling-dependent observer.

B. Problems considered

There are several drawbacks in assembling both OSISTC controller and 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 discrete LQR problem in (8) through recursive computation of the discrete algebraic Ricatti equation (DARE) until convergence [3]. The second issue is the pole placement solved by Ackermann's formula [8] 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; they are difficult to implement in real time on a embedded system with low computational capacity. If the execution time of the control task is too close to the minimum sampling interval, they appear undesirable effects such as jitter [9]. 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).

To solve the problems above it is proposed to perform certain offline mechanisms which produce approximations to be used online in order to optimize the processor utilization.

C. Set of sampling intervals T

The proposed strategy includes offline calculation of a set of all possible controller gain matrices $K_{d(\tau_k)}$, for later inferring and using polynomial functions to imitate their behaviour.

The first step is based on describing a *set of sampling intervals* $T \in \mathbb{R}^{1 \times s}$ within a closed interval $[\tau_{min}, \tau_{max}]$ as follows

$$T = \{\tau_{min}, \tau_{min} + \tau_g, \tau_{min} + 2\tau_g, \cdots, \tau_{max}\} \quad (13)$$

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

$$\tau_h = T_{[h]} \quad \forall h \in \mathbb{N} : 1 \leq h \leq s \quad (14)$$

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 [5]

$$\begin{cases} \dfrac{\beta^{\alpha}}{\eta} \leq \dfrac{1}{\tau_{min}} - \dfrac{1}{\tau_{max}}, & \beta := \sup\limits_{x \in X} |K_c(A_c + B_c K_c)x| \\ \tau_{RTOS} \leq \tau_g \end{cases} \quad (15)$$

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 algorithm will be implemented.

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

$K_{d(\tau_h)}$ is calculated by brute force for each h^{th} element from the set of sampling intervals T in (13) addressed by (14). The above involves each time first calculating the discrete matrices $A_{d(\tau_h)}$, $B_{d(\tau_h)}$, $Q_{d(\tau_h)}$ and $R_{d(\tau_h)}$ by (4) with which the discrete-time LQR problem is solved later; the latter stipulates the minimization of the cost function in (8). Therefore, we obtain a total of s controller gain matrices of the $K_{d(\tau_h)} \in \mathbb{R}^{m\times n}$ type because they are evaluated for each of the s possible values of τ_h within the set T. These matrices have the form

$$K_{d(\tau_h)} = dlqr\left(A_{d(\tau_h)}, B_{d(\tau_h)}, Q_{d(\tau_h)}, R_{d(\tau_h)}\right) = \begin{bmatrix} kd_{11}^{(\tau_h)} & \cdots & kd_{1n}^{(\tau_h)} \\ \vdots & \ddots & \vdots \\ kd_{m1}^{(\tau_h)} & \cdots & kd_{mn}^{(\tau_h)} \end{bmatrix} \tag{16}$$

where the superscript (τ_h) indicates the belonging of the corresponding gain element to matrix $K_{d(\tau_h)}$.

If the elements from all gain matrices are regrouped according to their position we have a group S_{K_d}, $m \cdot n$ training sets long, where n and m are the dimensions of states and inputs respectively, then

$$\begin{aligned} S_{K_d} = \{&[kd_{11}^{(\tau_1)}, \cdots, kd_{11}^{(\tau_s)}], \cdots, [kd_{1n}^{(\tau_1)}, \cdots, kd_{1n}^{(\tau_s)}], \cdots, \\ &[kd_{m1}^{(\tau_1)}, \cdots, kd_{m1}^{(\tau_s)}], \cdots, [kd_{mn}^{(\tau_1)}, \cdots, kd_{mn}^{(\tau_s)}]\}. \end{aligned} \tag{17}$$

Each training set in (17) is defined in $\mathbb{R}^{1\times s}$ and used to perform a *polynomial curve fitting* in order to find the coefficients θ of d-degree polynomials $K_{ij}(\tau_k)$. Therefore, we have a total of $m \cdot n$ polynomials each one following the form

$$K_{ij}(\tau_k) = \theta_1^{(ij)} \tau_k^d + \theta_2^{(ij)} \tau_k^{d-1} + \cdots + \theta_d^{(ij)} \tau_k + \theta_{d+1}^{(ij)}, \tag{18}$$

where superscript (ij) indicates the belonging of coefficients θ 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 will be the current sampling interval calculated online by equation (9) on a real controller. Thus (16) to (18) become

$$K_{d(\tau_k)} = \begin{bmatrix} K_{11}(\tau_k) & \cdots & K_{1n}(\tau_k) \\ \vdots & \ddots & \vdots \\ K_{m1}(\tau_k) & \cdots & K_{mn}(\tau_k) \end{bmatrix}, \tag{19}$$

where

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

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

It is a process similar to that described in subsection III-D. 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 $[A_{d(\tau_h)} - L_{d(\tau_h)}C]$. A rule of thumb considers to place the observer poles at least five to ten times farther to the left of s-plane than the dominant poles of the system. Therefore, the location of the continuous-time poles is statically assigned and these ones are discretized using (6) for all τ_h, obtaining $P_{d(\tau_h)}$.

Computing again $A_{d(\tau_h)}$ by (4) for all τ_h, assigning statically continuous-time poles and discretizing them by (6) also for all τ_h to have $P_{d(\tau_h)}$, 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 [8] with the form

$$L_{d(\tau_h)} = ackermann\left({A_{d(\tau_h)}}^T, C^T, {P_{d(\tau_h)}}^T\right) = \begin{bmatrix} ld_{11}^{(\tau_h)} & \cdots & ld_{1q}^{(\tau_h)} \\ \vdots & \ddots & \vdots \\ ld_{n1}^{(\tau_h)} & \cdots & ld_{nq}^{(\tau_h)} \end{bmatrix}. \tag{20}$$

Using the same regrouping criterion as in (17) a group S_{L_d}, $n \cdot q$ training sets long, it can be obtained

$$\begin{aligned} S_{L_d} = \{&[ld_{11}^{(\tau_1)}, \cdots, ld_{11}^{(\tau_s)}], \cdots, [kd_{1q}^{(\tau_1)}, \cdots, kd_{1q}^{(\tau_s)}], \cdots, \\ &[kd_{n1}^{(\tau_1)}, \cdots, kd_{n1}^{(\tau_s)}], \cdots, [kd_{nq}^{(\tau_1)}, \cdots, kd_{nq}^{(\tau_s)}]\}. \end{aligned} \tag{21}$$

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

$$L_{ij}(\tau_k) = \theta_1^{(ij)} \tau_k^d + \theta_2^{(ij)} \tau_k^{d-1} + \cdots + \theta_d^{(ij)} \tau_k + \theta_{d+1}^{(ij)}, \tag{22}$$

such as in (18). Finally, using (22) we obtain

$$L_{d(\tau_k)} = \begin{bmatrix} L_{11}(\tau_k) & \cdots & l_{1q}(\tau_k) \\ \vdots & \ddots & \vdots \\ L_{n1}(\tau_k) & \cdots & l_{nq}(\tau_k) \end{bmatrix}, \tag{23}$$

where

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

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 $L_{d(\tau_k)}$.

F. Implementation guidelines

Through Algorithm 1 what was said in subsections III-D and III-E 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.

IV. Experimental Test

A. Controlled plant

The experimental plant to be considered with form (1) is an electronic version of a double integrator circuit (critically

```
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 (13) and (15);
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 (16);
    Compute $P_{d(\tau_h)}$ by (6);
    $L_{d(\tau_h)} = f\left(A^T_{d(\tau_h)}, C^T, P_{d(\tau_h)}\right)$ by Ackermann (20);
end
Create $S_{K_d}$ based on all $K_{d(\tau_h)}$ by (17);
Create $S_{L_d}$ based on all $L_{d(\tau_h)}$ by (21);
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 (19);
Create $L_{d(\tau_k)}$ by ordering all $L_{ij}(\tau_k)$ as in (23);
```

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 $\tau_k$;
Calculate $K_{d(\tau_k)}$ by (19);
$u_{(k)} := f\left(K_{d(\tau_k)}, \hat{x}_{(k)}\right)$ by (10);
Set actuator with $u_{(k)}$;
Calculate $L_{d(\tau_k)}$ by (23);
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 (12)
```

Algorithm 2: Online implementation

stable); advise with [5] for further information. Its dynamics is

$$A_c = \begin{bmatrix} 0 & -23.81 \\ 0 & 0 \end{bmatrix}, B_c = \begin{bmatrix} 0 \\ -23.81 \end{bmatrix}, C = [1\ 0]. \tag{24}$$

In Table I most important configurations used to design both controller and observer are detailed. Their values have been based on recommendations from literature on the original approach [5].

B. Controller and observer

Numerical values obtained are summarized next:

- Continuous-time feedback gain $K_c = [0.1581, -0.5841]$.
- Controller gain matrix which consists of two polynomials $K_{d(\tau_k)} = [K_{11}(\tau_k), K_{12}(\tau_k)]$ where $K_{11}(\tau_k) = -8.8668\tau_k + 0.1548$ and $K_{12}(\tau_k) = 1.8819\tau_k - 0.5799$.

TABLE I
EXPERIMENT SETTINGS

Parameter	Value	Parameter	Value
Q_c	$0.0025 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	R_c	0.1
τ_{min}	15ms	τ_{max}	50ms
τ_g	1ms	α	0.667
β	9.4	η	0.11
P_c	$[-5+2j, -5-2j]$		

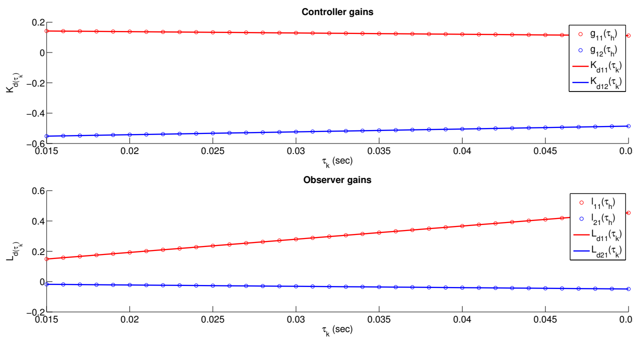

Fig. 3. Gains behaviour: controller (top), and observer (bottom)

- Observer gain matrix formed by two polynomials $L_{d(\tau_k)} = [L_{11}(\tau_k), L_{21}(\tau_k)]^T$ where $L_{11}(\tau_k) = 8.7071\tau_k + 0.0185$ and $L_{21}(\tau_k) = -0.8751\tau_k - 0.0048$.

In Fig. 3 gains of both controller and observer evaluated for the set of sampling intervals, are shown by circles. Likewise, fitted curves (continuous lines) roughly describe the behaviour of these gains.

C. Implementation on a processor

The development platform comprises a digital signal controller (DSC) with its respective RTOS and a double integrator electronic circuit characterized in (24). The controller is a Microchip dsPIC33FJ256MC710A, internally runs the Erika real-time kernel and is physically mounted on the Full Flex board. To learn more about this environment, it is recommended to see the original work on [11] and the references that exist within it.

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. Another function of the controller is to read the states of the plant $x_{(k)}$ through the DSC's analog/digital converter, to then 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).

The controller gain matrix, instead of minimizing DARE, uses two first-degree polynomials that are functions of τ_k and are represented as $K_{11}(\tau_k)$ and $K_{12}(\tau_k)$, grouped into $K_{d(\tau_k)}$. Finally, instead of using a pole placement method such as Ackermann, the observer gain matrix is replaced by a pair of first-degree polynomials $L_{11}(\tau_k)$ and $L_{21}(\tau_k)$, framed within $L_{d(\tau_k)}$.

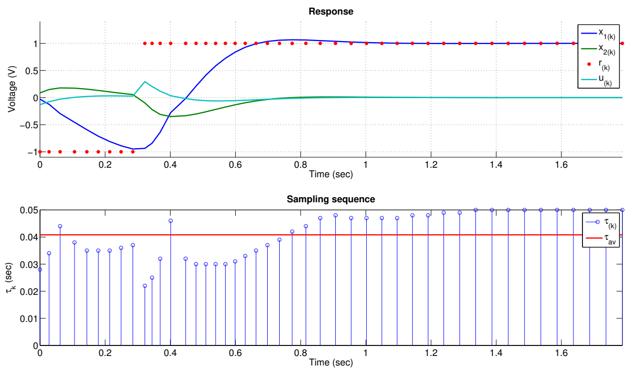

Fig. 4. Simulation results considering step response

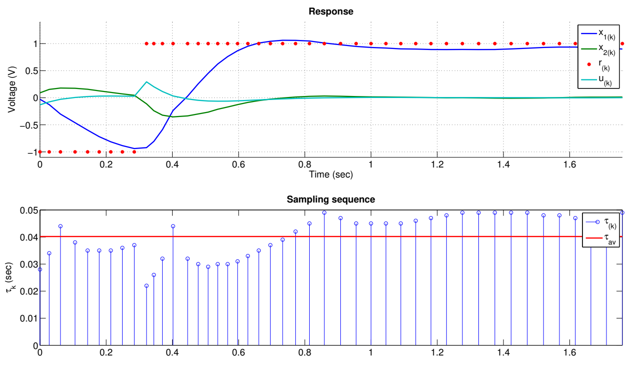

Fig. 5. Implementation results considering step response

D. Analysis of results

In Figures 4 and 5 the step response of OSISTC is evidenced both in simulation and actual implementation on a DSC, respectively; additionally both figures show the behaviour of the sampling sequence.

Considering same duration of both simulation and experiment, the criteria for determining correspondence between them are: time-response, minimum and maximum sampling intervals, and average sampling interval.

Within the time domain analysis a very close correspondence for both cases has been obtained, except that in the real controller there are slight variations due to uncertainty in process and sensors. These are reflected in the establishment time, overshoot, and steady-state error. In addition, the sampling interval tends to oscillate around τ_{max} in steady state.

Sampling intervals $\tau_{(k)}$ in the simulation lie within the range $[22,50]$ms, while in the real system are within $[21,50]$ms. It was ensured through guarantee in (15) that both meet the range $[\tau_{min},\tau_{max}]$.

The average sampling metric τ_{av} is taken from [5] and establishes that

$$\tau_{av} = \frac{1}{N}\sum_{k=0}^{N-1}\tau_k, \tag{25}$$

where N is the number of samples within the experiment/simulation time. Based on this it can be determined that for the simulation $\tau_{av} = 40.82$ms, and for the implementation $\tau_{av} = 39.96$ms. These results are very close and the slight difference may be due to uncertainty. It is necessary to point out that a larger τ_{av} indicates less resource utilization, which is a primary objective in self-triggered control.

V. Conclusion

Some implementation guidelines to make the theory in [5] applicable to real controllers were presented. For this, the implementation of the algorithm on microprocessor-based controllers was made by replacing the online use of the LQR problem by a lightweight polynomial fitted at design stage (offline). Additionally, a time-varying closed-loop observer has been implemented by polynomial fitting techniques while avoiding the online use of pole placement methods like Ackermann. The coherence between theory and practice has been demonstrated so that implementation can be assumed to be effective. Simulations and practical experiments on a real processor confirm 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 implementation; metrics from [5] and [12] 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. In addition, one could replace the Luenberguer observer with another observer with uncertainty information such as Kalman.

References

[1] M. Velasco, J.M. Fuertes, and P. Marti, "The Self Triggered Task Model for Real-Time Control Systems", in *Proc. IEEE Real-Time Systems Symposium*, pp. 67-70, 2003.

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

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

[4] E. Bini and G.M. Buttazzo, "The Optimal Sampling Pattern For Linear Control Systems", in *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 78-90, January 2014.

[5] M. Velasco, P. Martí, and E. Bini, "Optimal-Sampling-Inspired Self-Triggered Control", in *1st IEEE International Conference on Event-based Control, Communication, and Signal Processing*, Krakow, Poland, June 2015.

[6] J. Almeida, C. Silvestre, A.M. Pascoal, "Observer Based Self-Triggered Control of Linear Plants with Unknown Disturbances", in *American Control Conference*, pp. 5688-5693, 2012.

[7] D. Luenberger, "An introduction to observers", in *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 596-602, December 1971.

[8] J. Ackermann, "On the synthesis of linear control systems with specified characteristics", in *Automatica*, vol. 13, pp. 89-94, 1977.

[9] F. Paez, R. Cayssials, J. Urriza, E. Ferro and J. Orozco, "Frequency domain analysis of a RTOS in control applications", in *Seventh Argentine Conference on Embedded Systems (CASE)*, pp. 21-26, 2016.

[10] E. Weisstein, "Least Squares Fitting-Polynomial", from *MathWorld–A Wolfram Web Resource*, http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html

[11] C. Lozoya, P. Martí, M. Velasco, J. Fuertes and E. Martin, "Resource and performans trade-offs in real-time embedded control systems", in *Real-Time Systems*, vol. 49, no. 3, pp. 267-307, 2013.

[12] 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, 2017.