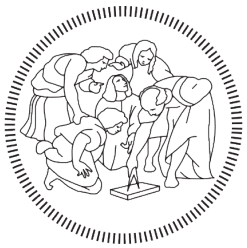
POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria dell'Automazione
Dipartimento di Elettronica, Informazione e Bioingegneria - DEIB



POLITECNICO
MILANO 1863

Multi-Sample Transmission in EB Control

Tesi di Laurea di:
Yusef AL Mawajdeh
Matricola:
850450

Relatore:
Prof. Alberto Leva

Anno Accademico 2016-2017

Abstract

This dissertation is about periodic event-based control, and has a twofold purpose. From a methodological viewpoint, the possibilities offered by having the sensor transmit past values of the controlled variable in addition to the one that triggered the event, are explored, and the consequent stability analysis is formalized. From the technological and implementation-related point of view, a Modelica library is presented to experiment with the theoretical ideas just sketched. The library comprehends several controller structures and event triggering techniques, with or without past samples transmission, and structures the represented control blocks so as to separate sensor, event generators and controllers, for maximum flexibility. The presented ideas look promising, and future work will be devoted to further investigate the stability of event-based control loops encompassing past samples transmission. Also, there is much room to expand the library by adding new models for the different event-based components.

Sommario

Questo lavoro riguarda il controllo ebent-based periodico e ha un duplice scopo. Da un punto di vista metodologico, vengono esplorate le possibilità offerte dal fatto che il sensore trasmetta valori passati della variabile controllata in aggiunta a quello che ha innescato l'evento, e la conseguente analisi di stabilità viene formalizzata. Dal punto di vista tecnologico e di implementazione, viene presentata una libreria Modelica per sperimentare le idee teoriche descritte sopra. La libreria comprende diverse strutture di controllori e diverse tecniche per la generazione di eventi, con o senza la trasmissione di campioni passati; inoltre, la libreria struttura i blocchi di controllo rappresentati in modo da separare sensori, generatori di eventi e controlori, per ottenere la massima flessibilità. Le idee presentate sembrano promettenti e in futuro la ricerca si concentrerà su ulteriori indagini a proposito della stabilità dei loop di controllo event-based (sempre di tipo periodco) comrpendenti la trasmissione dei campioni passati. Inoltre, c'è molto spazio per espandere la libreria aggiungendo nuovi modelli per i diversi componenti di un loop event-based.

Contents

List of Figures

Chapter 1

Introduction

For decades, the time triggered (periodic) control has been always the traditional choice for implementing the feedback control laws in digital controlled system. Using a fixed sampling time q_s to acquire measurements and calculate the control signal periodically, then update the actuator using zero-order holder (ZOH). Furthermore, the periodic control is the standard method for discretizing the continuous controller with q_s sampling time, thanks for the solid theories that periodic control has established through all the past years.
However, in the informatics world where the amount and size of transferred data is contentiously increasing specially in a big scaled plant where it has a large number of control components (sensors, controllers, actuators) that need to communicate to each others. Hence the periodic control becomes less efficient where the communication traffic is excessively increased and cause bandwidth problem and waste in the energy resources. Therefore, many researches recently start going in the direction of Event Based Control (EBC). In matter of fact, EBC is not a new topic, but the lack of the theories have prevented using it in the digital controlled system. Even though, many applications use EBC naturally. For example, encoders are event based sensors, Relay systems with on-off control and satellite thrusters are event based control and many others [1].

1.1 EB Control (Aperiodic Control)

Different from the periodic control, EBC acquires measurements, takes decisions and/or applies actions only when needed. As a result, there are numerous benefits come from using EBC. So by reducing the number of sampling, not only the communication traffic in a networked system is reduced and the bandwidth is saved, but also helps to efficiently utilize the system's resources. For instance, less number of transmitted measurement for a battery powered sensors in wire/wireless network implies a reduction of the sensor's battery consumption. Therefore, the battery lifetime significantly increases and obtains saving not just in energy, but also in the cost and maintenance effort specially for wireless sensors in remote places. It is farther assumed that the energy consumption of message transmission by the wireless sensor is much greater than energy consumption of calculation inside the sensor [2]. In addition, EBC reduces the number of tasks that has been computed by the controller's CPU in order to avoid computing a control action that brings no improvement to the process which is clearly a waste of the CPU computational resources. Another key point is that updating the actuators only when it is necessary increases the life cycle like reducing the wear in valves.

1.2 Main Contribution

In spite of the EBC's advantages, there are still many open problems that prevent us from fully use this powerful technique. One of the open problems is how to keep the state of the controller consistent with that of the controlled plant in the case of a long time span between two subsequent control events. This problem has many facets, and it is particular important if the control is implemented by means of state feedback, but also need to be consider in the case of any other standard industrial controller. The proposed solution is to have a sensor transmit not only the value of the measured variable when the event was triggered, but also a convenient number of past ones at fixed internal sampling rate. Consequently, it will slightly increase the transmitted information, but can be used straightforwardly to reconstruct the plant state, or simply to update that of a generic output- feedback controller, provided some conditions are met. Unfortunately, using this method rises a new problem, this time is related to stability. The EBC will have two

dynamic states matrices and based on violating the chosen trigger rule it will contentiously switch between them. However, it is important to proof the stability under arbitrarily number of switchings hence the system is stable in regardless of the selected trigger rule.

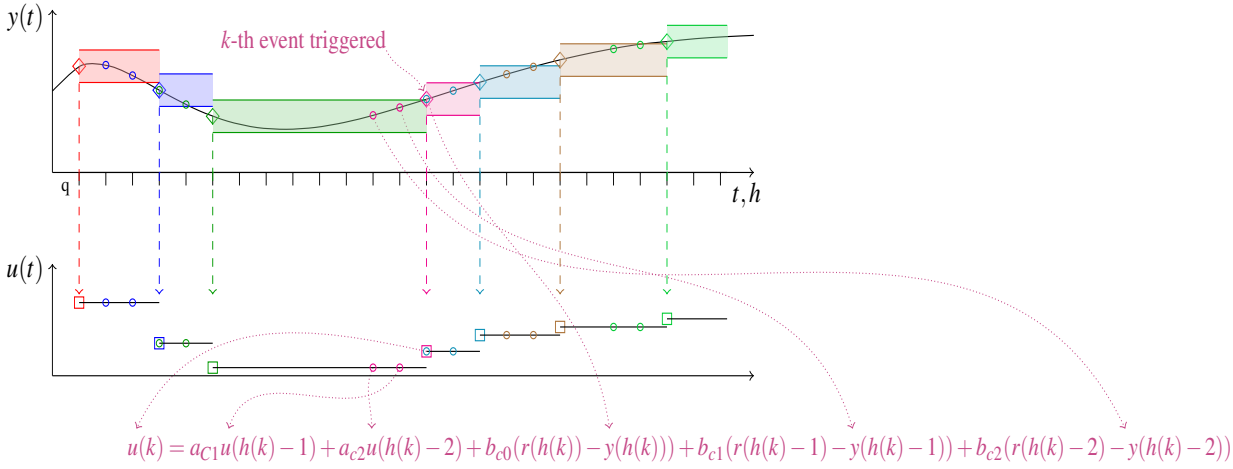


Figure 1.1: Operation of the proposed event-based scheme coupled to send-on-delta triggering with threshold Δ: diamonds indicate the controlled variable samples that trigger events, circles the past samples (two in this example) that are transmitted together with the triggering one, squares the new controls computed based also on past control samples, clearly known to the controller (two in the example) and denoted by circles as well; the semi-transparent bands indicate the $\pm\Delta$ ranges around the last transmitted output, and ticks on the upper horizontal axis mark the sensor sampling at step q.

As can be noticed, Figure 1.1 shows how the multi-sampling sensor operates. In this example particularly, the sensor is the single source of events coupled with send-on- delta policy. When an event is generated, as indicated by the dashed coloured arrows, the sensor transmits the present control variable y and the required number of the past samples of it, based on which (and on its internal state memory) the controller obtains a new value for the control command u, that is held till the next event. Although it is not shown in the figure in order to not impair clarity, it may happen that two subsequent events occur at distance in the h sampling less than the number of past control samples. In other words, the past control samples are not necessary all equal. Optionally, the sensor could be farther optimised to not

transmit replicated samples when events are close enough to generate some. It is important to mention that with past values' transmission works best with controller has order higher than one because it is safer to implement controllers that need more than one past value of controller variable, i.e. to go beyond the PID controllers.

Chapter 2

Literature Review

EBC has gained much of interests in the last years as mean to act on a controlled system "only when this is necessary", see e.g. [1] and the papers quoted therein. Therefore the literature offers a wide variety of works on a wide range of applications from solar collectors [3] to room temperature control [4] and thermal management of high-density micro processor [5].

In [6], the author states that the resurgence of *EB* debate is due to the increasing popularity of (shared) wired and wireless networked control systems (*NCS*) that rise the importance of explicitly addressing energy, consumption, and communication constraints when designing feedback control loops.

In regards to resources utilization, there are exist many works that focus in reducing the sampling numbers and calculation with saving resources [7] [8] [9] [10].

Other reasons that attract the researcher to *EBC* are mentioned in [1] and [7]. For instance, *EB* sampling is used in the process industry when statistical process control (*SPC*) is implemented in closed loop. In order not to disturb the process, a new control action is only calculated when a statistically significantly deviation has occurred. Also in the process industry that has many production units where buffer tanks are typically used for smoothing the production variations. *EB* can be applied to avoid any frequent change in production rates that cause upsets. Furthermore, modern distributed control systems impose system architectural constraints that make it difficult to stick to the time-triggered paradigm thus *EB* provides an alternative. This is specifically the case when control loops are closed over computer networks or buses, e.g., field buses, local area network or ATM network. Another reason why *EBC* is interesting is that it is closer in nature to

the way a human behaves as a controller. For example, the human motion control and also the manual control is EB rather than time triggered.

The major problem with EBC is the impossibility of analysing the system with well established and powerful theory as that available for the fixed rate case [9]. Therefore, there are many viewpoints on the matter have been proposed. Conditions on tuning parameters for the existence of equilibrium points have been investigated, see, e.g. [11], and modifications of the most commonly used laws, typically PI/PID and their tuning [7] [8] [12].

In particular many researches focused on PIDs controller. For example, [13] shows that the increased use of wireless sensors and actuators in industrial plants have stimulated a significant interest in event-based PI(D) controllers from academic and industrial researchers [7], [8], [14], [9], [15], [16], [17], [18], [19], [20] (in many cases the derivative part is not used). In fact, some works shows that this kind of controllers is able to reduce the power consumption of the devices and therefore to increase the battery life time. Further, the reduction of the communication load allows the reduction of the risk of lost data and stochastic time delays [21], [22], [23].

The author of this work [18] states that generally all the control laws should be adapted to the irregular intervals in order to use them in EB. Moreover, sensors and controllers in closed loop compose four different sampling configurations vary between time based and event based. The most promising configuration is when the both controller and sensor are event based what is called pure event based. However, in practical the event based mechanisms are completed with time triggered event to reduce the uncertainty that can be created from the pure event based system in some cases.

The work [2] highlights a relationship between the energy efficiency of wireless sensor and tuning PI controller in the case of level crossing sampling. It has been found that the best trade off between energy efficiency and control quality is reached by a setting with nearly no overshoot, while for periodic controller more overshoot is usually advantageous.

Another relevant problem in EB realisation is showed in [24] where Leva poses a challenge of how to keep the state of controller consistent with that of the controlled plant in the case of a long time span between two subsequent control events. the proposed solution in the paper is suggesting that the sensor transmits not only the value of the measured variable, but also a convenient number of that past ones at a fixed internal sampling rate. The transmitted information then can be used to reconstruct the plant state, or simply to update that of a generic output-feedback controller.

Regarding the stability, [9] provides a simple (sufficient) stability condition for EB realization of a PID and it is assumed that any event could be triggered only on a time instant that is an integer multiple of quantum q_s. In addition, it showed that there exists a close relationship between EB and switching systems. Moreover, the paper [9] states that the stability must be ensured independently of the triggering rule, therefore, it is done under arbitrary number of switching regardless of te triggering rule.

In Leva work [24], the dynamic matrix (state matrix) demonstrates the switching nature by taking two values when the control signal is updated or maintained.

There are several ways to design the control signal generator was discussed. In [1] , one method has advantage that the EB system is equivalent to a non linear system in case if it is free from modelling error and disturbance.

The design of triggering rule for EB systems are also widely discussed and there exist in literature many results which present a new event triggering rule scheme that guarantee the resulting event-triggering feedback system to be asymptotic stable. For example in [25], it represents non quadratic Lyapanuv-based triggering rule that ensure the asymptotic stability provided that the continuous systems are stabilizable.

This work [26] proposes a simultaneous design of the state feedback law and the event-triggering conditions ensuring local exponential stability and LQ performance in the presence of plant input saturation and of a communication channel between the controller output and the saturated plant input.

In this brief review, a various topics on EB have been through such as open problems in EB and the different triggering rules and schemes. But there does not exist in the literature a simulation library that contain all the different EB schemes and the triggering rules hence allow us to do a comparative studies, also there is no such rule in EB that takes into account using the past values of the control variables which they are going to be the core of this research.

Chapter 3

Modeling EB Control Loop

EB has various schemes, depending on how the event source(s) and triggering rule(s) make the sensor, the way that the controller and the actuator interact. In the next sections, we are going to talk about the triggering rules that used to build the Event-Generator (*EG*), the control laws and then build the scheme of the considered model and use the multi- sampled sensor and show the advantages and the consequences issue in stability.

3.1 Event Generators (EG)

Starting with the event generator (*EG*) which is the Essence of *EBC* system. The *EG* is responsible of firing events that are used with both the sensors to acquire measurements and the controllers to calculate a new control signal (*CS*). These events are triggered based on a specific triggering rules. In fact, there are four rules that have been modeled in a separate blocks, each rule will be discussed next in details. Another point to remember, *EG* is time based where it test the violation of these rules periodically with the same sample period corresponding to the time triggered controller and sensor [7].

3.1.1 Triggering Rules

The triggering rules generally vary between two policies, *SOD* (Send on Delta) and *SOA* (Send on Area), and vary in the used signal whether error signal or controlled variable. Farther more, each rule can be merged with

timeout condition where EG force an event after a certain amount of time elapse in regardless of triggering rule's violating state.

3.1.1.1 SOD Error Difference

SOD Error Difference basically uses the error signal (the difference between CV and set point (SP), $e = SP - CV$) and takes the absolute difference between the current error and the last triggered error event. Then, a new event is fired when the absolute difference is equal or greater than a predefined threshold as shown below:

$$abs(e - e_{Last}) >= threshold \tag{3.1}$$

Briefly, SOD Error Difference has relatively shown a higher sampling number comparing to other triggering rules such as SOD error. However, it has shown also a smaller steady sate error [12].

3.1.1.2 SOD Error

This triggering rule is similar to SOD Error Difference, but instead of comparing the difference between current error and the last triggered error, only the current error is taken into account and compared with a predefined threshold. The triggering rule is shown in equation 3.2:

$$abs(e) >= threshold \tag{3.2}$$

That is to say, SOD Error has shown less sampling number comparing to SOD error difference but has a larger steady sate error [12].

3.1.1.3 SOD Control Variable Difference

The main difference here lies in using the control variable instead of the error where the absolute difference between the current CV and last transmitted value CV_{Last} is calculated. The equation 3.3 below illustrates the triggering rule:

$$abs(CV - CV_{Last}) >= threshold \tag{3.3}$$

Generally speaking, SOD CV Difference has a very low sampling error and that explains the high oscillation in the plant response that occur due to the decreased number of the updated measured CV.

3.1.1.4 SOA Control Variable Difference

SOA CV Difference triggers a new event when the integrated area (Π) under the difference between the current $CV(t)$ and the last transmitted control variable ($CV_{Last}(t)$) along time is greater or equal than the predefined threshold (see equations 3.4 and 3.5). Subsequently, the area Π is reinitialize to zero after each time a new event is triggered.

$$\Pi\left(t_{last}, t\right) = \int_{t_{last}}^{t} \left(CV(t) - CV_{last}\right)\ dt \tag{3.4}$$

$$abs(\Pi) >= threshold \tag{3.5}$$

Moreover, SOA has shown a better performance comparing to SOD in term of sampling numbers in a highly noisy systems where it avoids unnecessary sensor data transmissions [27]. Having said that, SOD does not detect signal oscillation or steady-state error very efficiently if the difference (Δ) remains within the threshold range for a long time, but this problem has been countered in *SOA* [28].

3.2 Controllers (C)

PID controller is the most used in the industrial field. In order to use it in EBC system, a modification in PID structure is necessary. Therefore, there are several different models that vary mainly in the used method of discretizing the integral part and also in the used EG. The control algorithm is integrated with a various control signal generators, in this work we are implementing the Zero Order Hold (ZOH) and Impulse Hold (IH). In absence of a new event, the ZOH holds the value of the CS from the last triggered event till the next event occurs. While the IH sets the CS to zero all the time except for those instants when an event is triggered, then CS can has any value. However, all the controller models that are discussed next are represented in its digital form after discretization.

3.2.0.1 Fixed Rate PID

Fixed rate PID is a simple time triggered based. The control signal is computed periodically with a fixed sampling period (cycle time). Indeed, the CS is the sum of three parts, each part is demonstrated shown below:

1. Proportional part (u_p):

$$u_p = K_p \left(\beta SP - CV\right) \tag{3.6}$$

where K_p is the proportional gain and β is the SP weighting. In matter of fact, discretizaion for proportional part is straightforward [7].

2. Integral part (u_i):

$$u_i = u_i + \left(\frac{K_p \; cycletime}{T_i}\right) (SP - CV) \tag{3.7}$$

T_i represents the integral time. In order to discretize u_i, backward approximation with most recent error value is used [8].

3. Derivative part (u_d):

$$u_d = \left(\frac{T_d}{T_d + N \; cycletime}\right) u_d - \left(\frac{K_p \; T_d \; N}{T_d + N \; cycletime}\right) (CV - CV_{Last}) \tag{3.8}$$

where T_d is the derivative time and N is the low pass filter cut off frequency. Backward approximation is used for discretizing the derivative part [2].

After calculating each part, the new CS will be the sum of tall the three parts as shown below:

$$CS = u_p + u_i + u_d \tag{3.9}$$

3.2.0.2 Arzen's PID

Now shifting to EB controllers and starting with Arzen which is going to be discussed first and it will be the starting point for the rest of EB controllers. More accurately saying, the used Arzen model is an improved model where the difference from the known standard Arzen lies in using the backward discretzing method for the integral part instead of the forward discretzing method. The reason behind that is for anticipating the next event for forward method is not possible in practice. Opposite the fixed rate

controllers, the interval between two subsequent events (I) is not fixed and equal to an integer multiple (h) of the nominal sampling time (cycle time):

$$I = h\ cycletime \tag{3.10}$$

In addition, Arzen can internally force a control event independent from the violation of the EG triggering rule. In other words, Arzen Implies the time out safety condition where a controller forces a new event after satisfying the following rule:

$$I >= h_{max} \tag{3.11}$$

where h_{max} is the maximal safety interval time.
A new CS (see eq 3.9) is generated each time an event is triggered by EG or forced event by the safety condition. Likewise, the proportional part remains same as in FR controller (see eq 3.6) while the integral and the derivative parts differ by only replacing the nominal cycle time with varying interval (I) as shown below:

$$u_i = u_i + \left(\frac{K_p I}{T_i}\right)(SP - CV) \tag{3.12}$$

$$u_d = \left(\frac{T_d}{T_d + NI}\right)u_d - \left(\frac{K_p T_d N}{T_d + NI}\right)(CV - CV_{Last}) \tag{3.13}$$

3.2.0.3 Durand's PID

Durand is merely an improvement on Arzen to avoid a repetitive computations, specially in the long steady state where a very few new events are triggered from the violation in the EG's triggering rules [8]. The time out safety condition is removed and only EG remains responsible to generate events which effects directly in reducing the number of event, but overshoots appear after long steady state intervals. To solve this issue, modifications in the integral part are introduced to Durand in the next following models.

3.2.0.4 Durand's PID with Saturation

The only difference from Durand is the modification on the integral part to solve the above mentioned problem. The idea of this modification is bounding the product between the sampling interval (I) and the error ($e = SP - CV$) after a long steady state. Thus, when the sampling interval becomes too

large, the product $(I * e)$ is saturated. Otherwise, the I_e will takes its real value. In this case, u_i will be same as in equation 3.7. The algorithm is shown below:

if $I >= h_{maxI}$ *then*

$$Ie = (I - cycletime)threshold + cycletime\ e \tag{3.14}$$

else

$$I_e = Ie \tag{3.15}$$

end

where h_{maxI} is maximal time interval for Integral part, and beyond it the product Ie is saturated. $threshold$ equals to the value of the same parameter in EG. After all, the integral part can be written like this:

$$u_i = u_i + \frac{K_p Ie}{T_i} \tag{3.16}$$

3.2.0.5 Durand's PID with Exponential Forgetting Factor

Similar to Durand with saturation, but the sampling interval will be reduced after long steady state. This time, by adding an exponential forgetting factor. Then a new interval (I_{exp}) will be computed and used in the integral part u_i.

$$I_{exp} = Ie^{(cycletime - I)} \tag{3.17}$$

$$u_i = u_i + \left(\frac{K_p I_{exp}}{T_i}\right) e \tag{3.18}$$

3.2.0.6 Durand's Hybrid PID

Hyprid algorithm uses the both modification, saturation and exponential forgetting factor. As a result, the product (Ie) will be modified as following:

if $I >= h_{maxI}$ *then*
  $I_{exp} = Ie^{(cycletime - I)}$
  $Ie = (I_{exp} - cycletime)threshold + cycletime\ e$
else
  $Ie = Ie$
end

And the integral part u_i is written as following:

$$u_i = u_i + \frac{K_p I e}{T_i} \tag{3.19}$$

3.3 EBC Modelling

In this work, we are considering single input, single output ($SISO$) EB loop with single source of events. More precisely, we assume that the sensor is the single source of events. And the triggered event cause the computation of a control action which is actuated contextually.

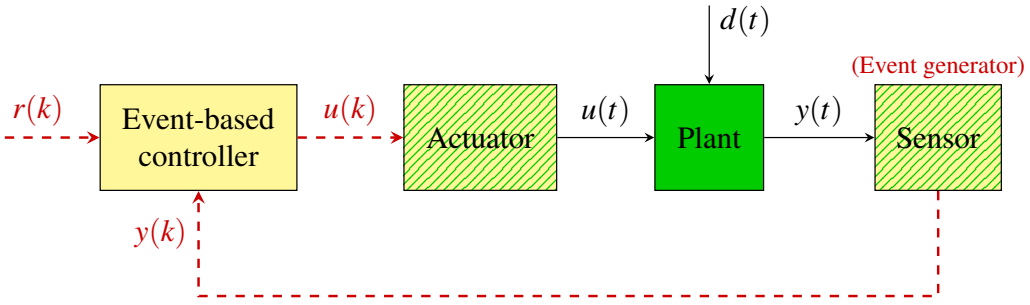


Figure 3.1: Block diagram of the considered EB loop

The considered scheme of EB loop is shown in Figure 3.1. The black solid lines represent contentious time signal t or could be FR (fixed rate) sampling time q where $t = Nq$ and N is an integer. Moreover, red dashed lines represent signals sampled at event and k is an integer index counting the events. r,y,u and d are respectively the set point, the controlled variable, the control signal and the disturbance. The hatched blocks are the interfaces between the event based world of the controller, and the contentious time one of the plant.
However, EB system is hybrid control loop where it contentiously switches between two operating modes. In other words, there are two dynamic behaviours (two state matrices) representing the EB system. The first mode is when an event is triggered by the sensor then it sends the controller the latest measurement in order to compute a new control signal ($u(k+1)$ and update the actuator with the new value by using ZOH (zero order holder) or any used control signal generator. And the state matrix for this system called A_{run}. the second mode happens in case of no events (absent of events),

then the system runs in open loop and the sensor does not transmit any new values to the controller; hence the controller holds the latest control signal computed from the last event using ZOH and the state matrix for this system called A_{hold}.

3.3.1 Hypothesises

Before starting to write the system's equations, it is important to state the hypothesises that system has been modelled in. Listed below:

- The process plant is $SISO$ (single-input single-output) LTI (linear and time-invariant) with a transfer function $P(s)$ and it is always assumed to be strictly proper, $d_p = 0$.
- The contentious time controller is $SISO$ LTI that has been already synthesised to fulfil the closed loop specifications as stability and performance. And it has a transfer function $C(s)$ where it is assumed to be strictly proper (it is not necessary for the controller to be strictly proper but in this work it is assumed to be).
- The plant and the controller are discretized using a sampling time q_s in such way that the stability degree and other properties remain acceptable, suitable for fixed rate realisation of the controller.

3.3.2 State Space Representation

The process plant is represented in discrete state space equations (state equation 3.20a and the output equation 3.20b). Furthermore, the system has n_p order an integer number and $x_p(k)$ is a column vector of state variables that has n_p elements such that $x_{pi}(k)$ is the ith element where i=1,2...,n_p, and k is an integer counts the events.

$$x_p(k) = A_p x_p(k-1) + b_p u(k-1) \tag{3.20a}$$
$$y(k-1) = c_p x_p(k-1) \tag{3.20b}$$

Similarly, the controller is represented in discrete state space (state equation 3.21a and output equation 3.21b). Also the controller has order n_c an integer

number and $x_c(k)$ is the state variables column vector with n_c elements such that x_{ci} where i=1,2...n_c.

$$x_c(k) = A_c x_c(k-1) + b_c e(k-1) \tag{3.21a}$$
$$u(k-1) = c_c x_c(k-1) + d_c e(k-1) \tag{3.21b}$$

Where $e(k) = r(k) - u(k)$ is the error at the k^{th} event
However, a $SIMO$ (single-input multi-output) system with order equal to $n_p + n_c$ arises from connecting the process plant and the controller together.

3.3.3 EB in Closed Loop (Run)

Starting with the first operating mode (run) that we enter when an event is triggered. The state space equations for the closed loop where a new control signal u is computed, the system is shown as following:

$$\begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix} = A_{run} \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} + \begin{bmatrix} b_p d_c \\ b_c \end{bmatrix} r(k-1) \tag{3.22a}$$

$$\begin{bmatrix} y(k-1) \\ u(k-1) \end{bmatrix} = \begin{bmatrix} c_p & 0_{1\times n_c} \\ -d_c c_p & c_c \end{bmatrix} \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ d_c \end{bmatrix} r(k-1) \tag{3.22b}$$

where

$$A_{run} = \begin{bmatrix} A_p - b_p d_c c_p & b_p c_c \\ -b_c c_p & A_c \end{bmatrix} \tag{3.23}$$

Note that A_{run} is obtained directly by simply substitute equation 3.21b in equation 3.20a. In contrast, we will see that deriving A_{hold} is more complex.

3.3.4 EB in Open Loop (Hold)

In the second operating mode (hold), there is no triggered event, the system is in open loop and the controller holds the control signal from the last event, see equation 3.24.

$$u(k) = u(k-1) \tag{3.24}$$

Furthermore, the evolution of the plant's state $x_p(k)$ is not effected and remain in same way as in run mode, only the controller's state $x_c(k)$ here is

needed to be computed and determined instead of depending on it as usual. Accordingly, we need to write n_c scalar equations where n_c is the order of the controller. And one way is to propose that the sensor transmits not only the value of the present measured variable when the event was triggered, but also a convenient number of past ones at its fixed internal sampling rate in order to use them in the absent of event, hold mode, to write the necessary n_c equations.
However, let $C(z^{-1})$ be the transfer function obtained by taking the Z-transform for the controller state space, and write

$$\begin{aligned} C(z^{-1}) &= \frac{B_c(z^{-1})}{1 - A_c(z^{-1})} \\ &= \frac{b_{c0} + b_{c1}z^{-1} \cdots + b_{cm_c}z^{-m_c}}{1 - a_{c1}z^{-1} \cdots - a_{cn_c}z^{-n_c}} \end{aligned} \tag{3.25}$$

Assume we are at time k in he hold mode thus we need to determine the controller state $x_c(k)$ by writing n_c singular equations using the following information: the sensor provides $1 + m_c$ measured variables transmitted at time instant $k-1$ $\{y(k-1) \dots y(k-1-m_c)\}$ and a similar set is also saved for the reference signal in an internal buffer $\{r(k-1) \dots r(k-1-m_c)$ where the present one $r(k)$ is known as well. Also n_c past control signals $\{u(k-1) \dots u(k-n_c)\ \}$ is saved in an internal buffer. Later we are going to explain how these information are exploited to write the n_c singular equations.
Back to the main argument, in the hold mode where the value of the control signal $u(k)$ is held constant equal to the previous one $u(k-1)$ until a new event is generated.
Starting from equation 3.24 we obtain the following:

$$\begin{aligned} c_c x_c(k) &= u(k-1) - d_c e(k) \\ &= u(k-1) - d_c(r(k) - c_p x_p(k)) \\ &= u(k-1) - d_c r(k) + d_c c_p(A_p x_p(k-1) + b_p u(k-1)) \end{aligned} \tag{3.26}$$

It is worth to mention that solving equation 3.26 and determining $x_c(k)$ is not always feasible neither trivial task.
For example in the case of a non strictly proper controller, if the set point has changed then we must be in run mode not in hold mode because it interferes with the event triggering mechanism. Consequently, there is no solution in the hold mode in this particular case.

On the other hand, if we are in the hold mode and the set point is fixed, $r(k) = r(k-1)$, equation 3.26 is solvable and become as following:

$$c_c x_c(k) = u(k-1)(1 + d_c c_p b_p) + d_c c_p A_p x_p(k-1) - d_c r(k-1) \tag{3.27}$$

In the case of a strictly proper controller where $d_c = 0$, a solution is exist and equation 3.26 is simplified as shown below.

$$c_c x_c(k) = u(k-1) \tag{3.28}$$

For sake of simplicity, we are going to proceed in the hypothesis of strictly proper controller. Now to compute $x_c(k)$, we need n_c singular equation and 3.28 represents the first equation. For the rest of $n_c - 1$ equations, we are going to show how to derive them using the controller past outputs.
Starting with deriving the second equation:

$$c_c x_c(k-1) = u(k-1) \tag{3.29a}$$

where $u(k-1)$ is known, exogenous signal, and we need to express $x_c(k-1)$ as a function of our unknown state variable(s) $x_c(k)$. Therefore, solve the state equation 3.21a for $xc(k-1)$. And to do so, the state matrix A_c must be non singular and has no delay in the controller which is reasonable.

$$x_c(k-1) = A_c^{-1}(x_c(k) - b_c r(k-1) + b_c c_p x_p(k-1)) \tag{3.29b}$$

Now, substitute equation 3.29b in equation 3.29a and solve for $x_c(k)$.

$$c_c A_c^{-1} x_c(k) = -c_c A_c^{-1} b_c c_p x_p(k-1) + c_c A_c^{-1} b_c r(k-1) + u(k-1) \tag{3.29c}$$

The second equation is shown above in 3.29c where $c_c A_c^{-1} b_c r(k-1) + u(k-1)$ are exogenous input signals.
In similar manner, the third equation is derived:

$$c_c x_c(k-2) = u(k-2) \tag{3.30a}$$

$$\begin{aligned} x_c(k-2) &= A_c^{-1}(x_c(k-1) - b_c r(k-2) + b_c c_p x_p(k-2)) \\ &= A_c^{-1}\left(A_c^{-1}\left(x_c(k) - b_c r(k-1) + b_c c_p x_p(k-1)\right) - b_c r(k-2) + b_c c_p x_p(k-2)\right) \end{aligned} \tag{3.30b}$$

As we did for $x_c(k-1)$ in 3.29b, solve the plant state equation for $x_p(k-2)$, holding that A_p is non singular.

$$x_p(k-2) = A_p^{-1}(x_p(k-1) - b_p u(k-2)) \tag{3.30c}$$

Then substitute 3.30c in 3.30b.

$$\begin{aligned} x_c(k-2) &= A_c^{-2}x_c(k) - A_c^{-2}b_c r(k-1) + A_c^{-2}b_c c_p x_p(k-1) - A_c^{-1}b_c r(k-2) \\ &\quad + A_c^{-1}b_c c_p A_p^{-1}x_p(k-1) - A_c^{-1}b_c c_p A_p^{-1}b_p u(k-2) \end{aligned} \tag{3.30d}$$

Now substitute 3.30d in 3.30a and solve for $x_c(k)$

$$\begin{aligned} c_c A_c^{-2}x_c(k) &= -c_c(A_c^{-2}b_c c_p + A_c^{-1}b_c c_p A_p^{-1})x_p(k-1) + c_c A_c^{-2}b_c r(k-1) \\ &\quad + c_c A_c^{-1}b_c r(k-2) + (1 + c_c A_c^{-1}b_c c_p A_p^{-1}b_p)u(k-2) \end{aligned} \tag{3.30e}$$

In the same way, the fourth equation is written as following:

$$\begin{aligned} c_c A_c^{-3}x_c(k) &= -c_c(A_c^{-3}b_c c_p + A_c^{-2}b_c c_p A_p^{-1} + A_c^{-1}b_c c_p A_p^{-2})x_p(k-1) \\ &\quad + c_c A_c^{-3}b_c r(k-1) + c_c A_c^{-2}b_c r(k-2) + c_c A_c^{-1}b_c r(k-3) \\ &\quad + (1 + c_c A_c^{-1}b_c c_p A_p^{-1}b_p)u(k-3) \\ &\quad + (c_c A_c^{-2}b_c c_p A_p^{-1}b_p + c_c A_c^{-1}b_c c_p A_p^{-2}b_p)u(k-2) \end{aligned} \tag{3.31}$$

In conclusions, to write n_c singular equations we always set the first equation to be $c_c xc(k-1) = u(k-1)$ and the rest $n_c - 1$ equations (when $nc > 1$) are obtained recursively using the general law below.

$$\begin{aligned} c_c A_c^{-h}x_c(k) &= -c_c\left(\sum_{l=0}^{h-1} A_c^{l-h}b_c c_p A_p^{-l}\right)x_p(k-1) + c_c\sum_{l=1}^{h} A_c^{l-h-1}r(k-l) \\ &\quad + u(k-h) + c_c\sum_{l=2}^{h} A_c^{l-h-1}b_c c_p A_p^{-1}b_p u(k-l) + c_c\sum_{l=2}^{h-1} A_c^{l-h}b_c c_p A_p^{-2}b_p u(k-l) \\ &\quad + c_c\sum_{l=2}^{h-2} A_c^{l-h+1}b_c c_p A_p^{-3}b_p u(k-l) + \ldots\ldots \\ &\quad \cdots + c_c\sum_{l=2}^{h-s} A_c^{l-h-1+s}b_c c_p A_p^{-1-s}b_p u(k-l) \end{aligned} \tag{3.32}$$

Where h and s are integer numbers such that $h = 1 \ldots n_c - 1$ where $n_c > 1$ and $s = 0, 1, 2 \ldots h-2$ when $h > 1$.
Writing the n_c equations together in a matrix form , we can obtain the following:

$$Mx_c(k) = Qx_p(k-1) + Inputs \tag{3.33a}$$

Where M, Q and the $Inputs$ are matrices such that:

$$M = \begin{bmatrix} c_c \\ c_c A_c^{-1} \\ c_c A_c^{-2} \\ \vdots \\ c_c A_c^{-h} \end{bmatrix} \tag{3.33b}$$

$$Q = \begin{bmatrix} 0_{1\times n_p} \\ -c_c A_c^{-1} b_c c_p \\ -c_c (A_c^{-2} b_c c_p + A_c^{-1} b_c c_p A_p^{-1}) \\ \vdots \\ -c_c \left(\sum_{l=0}^{h-1} A_c^{l-h} b_c c_p A_p^{-l} \right) \end{bmatrix} \tag{3.33c}$$

$$\begin{aligned} Inputs \;=\; & \begin{bmatrix} 0 \\ c_c A_c^{-1} b_c r(k-1) \\ c_c A_c^{-2} b_c r(k-1) + c_c A_c^{-1} b_c r(k-2) \\ \vdots \\ c_c \sum_{l=1}^{h} A_c^{l-h-1} b_c r(k-l) \end{bmatrix} \\ & + \begin{bmatrix} u(k-1) \\ u(k-1) \\ (1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p) u(k-2) \\ \vdots \\ u(k-h) + c_c \sum_{s=0}^{h-2} \sum_{l=2}^{h-s} A_c^{l-h-1+s} b_c c_p A_p^{-1-s} b_p u(k-l) \end{bmatrix} \end{aligned} \tag{3.33d}$$

Obviously, the equations are still not ready to write the controller's state equation. First, simplify the $Inputs$ term by substituting all the control signal $u(k-1) \dots \; u(k-h)$ with its known output equation and write them in term of $x_c(k-1)$ in such way $u(k-1) = c_c x_c(k-1)$, but it gets more complicated for $u(k-2)$ and $u(k-3) \dots u(k-h)$ because it applies recursive

formula as it is shown below:

$$
\begin{aligned}
u(k-2) &= \frac{c_c A_c^{-1} x_c(k-1) - c_c A_c^{-1} b_c r(k-2) + c_c A_c^{-1} b_c c_p A_p^{-1} x_p(k-1)}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \\
\frac{u(k-3)}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} &= \left(c_c A_c^{-2} - \frac{c_c A_p^{-1} b_p c_c A_c^{-1}}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \right) x_c(k-1) + \\
&\quad \left(-c_c A_c^{-2} b_c + \frac{c_c A_p^{-1} b_p c_c A_c^{-1} b_c}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \right) r(k-2) - c_c A_c^{-1} r(k-3) + \\
&\quad \left(c_c A_c^{-2} b_c c_p A_p^{-1} + c_c A_c^{-1} b_c c_p A_p^{-2} - \frac{c_c A_p^{-1} b_p c_c A_c^{-1} b_c c_p A_p^{-1}}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \right) x_p(k-1)
\end{aligned}
$$

Then, write all the terms in $r(k-1)$, $r(k-2)$, $x_c(k-1)$ and $x_p(k-1)$ and obtain the following structure

$$Inputs = I_{r1} r(k-1) + I_r r(k-) + Q_c xc(k-1) \tag{3.33e}$$

Where

$$I_{r1} = \begin{bmatrix} 0 \\ c_c A_c^{-1} b_c \\ c_c A_c^{-2} b_c \\ c_c A_c^{-3} b_c \\ \vdots \end{bmatrix} \tag{3.33f}$$

$$I_{r2} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ c_c A_c^{-1} b_c \\ \hline 1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p \\ \vdots \end{bmatrix} \tag{3.33g}$$

$$Q_c = \begin{bmatrix} c_c \\ c_c \\ c_c A_c^{-1} \\ c_c A_p^{-1} b_p c_c A_c^{-1} \\ c_c A_c^{-2} - \frac{}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \\ \vdots \end{bmatrix} \tag{3.33h}$$

And for $x_p(k-1)$, the Q matrix becomes Q_p where

$$Q_p = \begin{bmatrix} 0_{1\times n_p} \\ -c_c A_c^{-1} b_c c_p \\ -c_c A_c^{-2} b_c c_p \\ -c_c A_c^{-3} b_c c_p - \dfrac{c_c A_p^{-1} b_p c_c A_c^{-1} b_c c_p A_p^{-1}}{1 + c_c A_c^{-1} b_c c_p A_p^{-1} b_p} \\ \vdots \end{bmatrix} \tag{3.33i}$$

Next, equation 3.33a is rewritten in this way:

$$Mx_c(k) = Q_p x_p\,(k-1) + Q_c C x_c(k-1) + I_{r1} r(k-1) + I_{r2} r(k-2) \tag{3.33j}$$

Solving 3.33j for $x_c(k)$ where M matrix must be invertible and it is satisfied when the controller is detectable and strictly proper. And in hold mode, the reference point is assumed to be fixed where $r(k-1) = r(k-2)$.

$$\begin{bmatrix} x_p\,(k) \\ x_c\,(k) \end{bmatrix} = A_{hold} \begin{bmatrix} x_p\,(k-1) \\ x_c\,(k-1) \end{bmatrix} + \begin{bmatrix} 0_{n_p\times 1} \\ M^{-1}(I_{r1} + I_{r2}) \end{bmatrix} r(k-1) \tag{3.34a}$$

Where the state matrix in the hold mode for the strictly proper controller is written as following:

$$A_{hold} = \begin{bmatrix} A_p & b_p c_c \\ M^{-1} Q_p & M^{-1} Q_c \end{bmatrix} \tag{3.34b}$$

Note that the evaluation of the plant's state has not effected and it is same as in the run mode for the strictly proper controller case, $d_c = 0$, and there is no change in the output equations.

$$\begin{bmatrix} y\,(k) \\ u\,(k) \end{bmatrix} = \begin{bmatrix} c_p & 0_{1\times n_c} \\ 0_{1\times n_p} & c_c \end{bmatrix} \begin{bmatrix} x_p\,(k-1) \\ x_c\,(k-1) \end{bmatrix} \tag{3.34c}$$

Finally, the equations 3.34a and 3.34c are the state space representation for the EBC in hold mode for strictly proper controller.

3.3.5 The Switching Nature for EB Control Loop

Apparently, EB control loop can be seen as a switching system where the controller is either in the run mode computing a new control signal or in

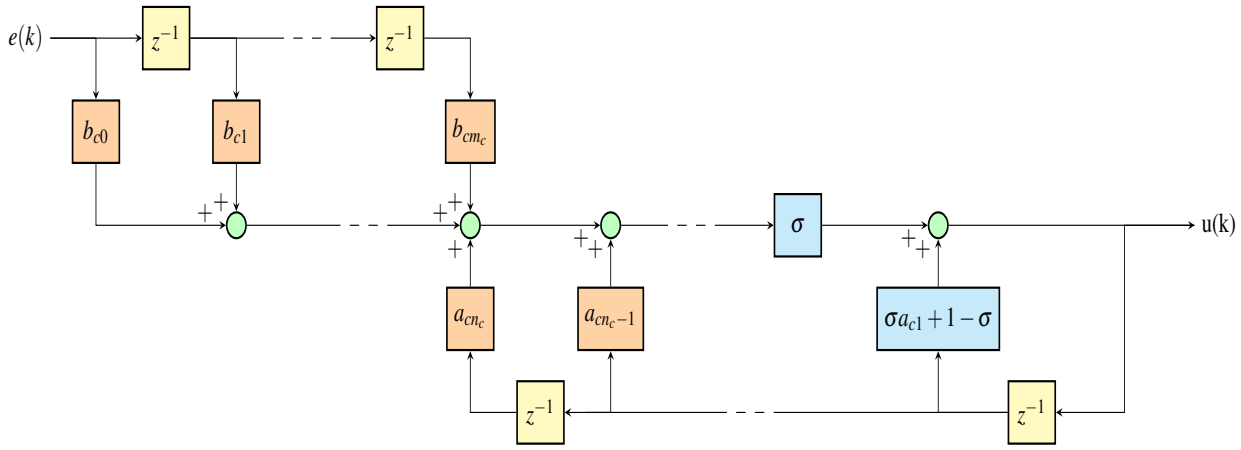


Figure 3.2: Block diagram for the EB controller representing the switching between the two modes, Run mode ($\sigma = 1$) and Hold mode ($\sigma = 0$).

the hold mode holding the last control signal till a new event occurs. This switching mechanism can be represented a in block diagram where the value of σ decides in which mode the controller is in. The computation of the control signal is actuated contextually, see figure 3.2.

The next step is to verify the compatibility between the block diagram and the equation obtained from our model. At the end we should be able to obtain a dynamic matrix A in such way:

$$A = \sigma A_{run} + (1 - \sigma) A_{hold} \tag{3.35}$$

Taking a general example of a strictly proper plant with order $n_c = 3$ and a strictly proper controller with order $n_c = 3$. And using the observable canonical form for the both, plant and controller. Then, the state space

representation for the plant will be as following:

$$\begin{aligned}
x_p(k) &= A_p x_p(k-1) + b_p u(k-1) \\
y(k-1) &= c_p x_p(k-1) \\
&where \\
A_p &= \begin{bmatrix} 0 & 0 & -a_{p_3} \\ 1 & 0 & -a_{p_2} \\ 0 & 1 & -a_{p_1} \end{bmatrix} \quad b_p = \begin{bmatrix} b_{p_3} \\ b_{p_2} \\ b_{p_1} \end{bmatrix} \\
c_p &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad x_p(k) = \begin{bmatrix} x_{p_1}(k) \\ x_{p_2}(k) \\ x_{p_3}(k) \end{bmatrix}
\end{aligned}$$

And for the controller, the state space representation is written below:

$$x_c(k) = A_c x_c(k-1) + b_c e(k-1) \tag{3.37a}$$

$$u(k-1) = c_c x_c(k-1) \tag{3.37b}$$

$$\begin{aligned}
&where \\
A_c &= \begin{bmatrix} 0 & 0 & -a_{c_3} \\ 1 & 0 & -a_{c_2} \\ 0 & 1 & -a_{c_1} \end{bmatrix} \quad b_c = \begin{bmatrix} b_{c_3} \\ b_{c_2} \\ b_{c_1} \end{bmatrix} \\
c_c &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad x_c(k) = \begin{bmatrix} x_{c_1}(k) \\ x_{c_2}(k) \\ x_{c_3}(k) \end{bmatrix} \\
e(k) &= r(k) - y(k)
\end{aligned}$$

Now writing the state space for the EB control loop in the run mode using the equations in section 3.3.3.

$$\begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix} = A_{run} \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} + b_{run} r(k-1) \tag{3.38}$$

$$\begin{bmatrix} y(k-1) \\ u(k-1) \end{bmatrix} = C \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} \tag{3.39}$$

where

$$A_{run} = \begin{bmatrix} 0 & 0 & -a_{p_3} & 0 & 0 & b_{p_3} \\ 1 & 0 & -a_{p_2} & 0 & 0 & b_{p_2} \\ 0 & 1 & -a_{p_1} & 0 & 0 & b_{p_1} \\ 0 & 0 & -b_{c_3} & 0 & 0 & -a_{c_3} \\ 0 & 0 & -b_{c_2} & 1 & 0 & -a_{c_2} \\ 0 & 0 & -b_{c_1} & 0 & 1 & -a_{c_1} \end{bmatrix} \quad b_{run} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{c_3} \\ b_{c_2} \\ b_{c_1} \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The state equations are written as:

$$\begin{aligned} x_{p_1}(k) &= -a_{p_3} x_{p_3}(k-1) + b_{p_3} x_{c_3}(k-1) \\ x_{p_2}(k) &= x_{p_1}(k-1) - a_{p_2} x_{p_3}(k-1) + b_{p_2} x_{c_3}(k-1) \\ x_{p_3}(k) = y(k) &= x_{p_2}(k-1) - a_{p_1} x_{p_3}(k-1) + b_{p_1} x_{c_3}(k-1) \\ x_{c_1}(k) &= -a_{c_3} x_{c_3}(k-1) - b_{c_3} x_{p_3}(k-1) + b_{c_3} r(k-1) \\ x_{c_2}(k) &= x_{c_1}(k-1) - a_{c_2} x_{c_3}(k-1) - b_{c_2} x_{p_3}(k-1) + b_{c_2} r(k-1) \\ x_{c_3}(k) = u(k) &= x_{c_2}(k-1) - a_{c_1} x_{c_3}(k-1) - b_{c_1} x_{p_3}(k-1) + b_{c_1} r(k-1) \end{aligned}$$

By simplifying $u(k)$ and writing everything in terms of u and e we obtain the following:

$$u(k) = -a_{c_1} u(k-1) - a_{c_2} u(k-2) - a_{c_3} u(k-3) + b_{c_1} e(k-1) + b_{c_2} e(k-2) + b_{c_3} e(k-3) \tag{3.40}$$

Note that equation 3.40 corresponds to the block diagram in figure 3.2 in the run mode ($\sigma = 1$).
In the hold mode, the EB control loop state space representation is shown below using the equations in section 3.3.4.

$$\begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix} = A_{hold} \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} + b_{hold} r(k-1) \tag{3.41}$$

$$\begin{bmatrix} y(k-1) \\ u(k-1) \end{bmatrix} = C \begin{bmatrix} x_p(k-1) \\ x_c(k-1) \end{bmatrix} \tag{3.42}$$

where

$$A_{hold} = \begin{bmatrix} 0 & 0 & -a_{p_3} & 0 & 0 & b_{p_3} \\ 1 & 0 & -a_{p_2} & 0 & 0 & b_{p_2} \\ 0 & 1 & -a_{p_1} & 0 & 0 & b_{p_1} \\ 0 & 0 & -b_{c_3} & 0 & 0 & -a_{c_3} \\ 0 & 0 & *^1 & \frac{-1}{a_{c_3}} & 0 & a_{c_1} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad b_{hold} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_{c_3} \\ b_{c_2} - b_{c_1} a_{c_1} \\ 0 \end{bmatrix}$$

$$*^1 = (-b_{c_3} + \frac{b_{c_2} a_{c_2}}{a_{c_3} a_{p_3}} - \frac{-b_{c_1} a_{c_1}}{a_{c_3} a_{p_3}})$$

In like manner, writing the state equations we obtain that $u(k)$ is hold to the previous value as it should and the equations correspond to the block diagram in figure 3.2 in the hold mode ($\sigma = 0$).

$$x_{c_3}(k) = x_{c_3}(k-1)$$

Also can be written as

$$u(k) = u(k-1)$$

To sum up, the EB control loop can be represented as switching system as has been shown in the block diagram in figure 3.2. And the equations are compatible with the block diagram.

Chapter 4

The Modelica Library

This chapter presents the motivations of establishing the *EBC* Open Modelica library. Also showing the components that the library contains and how they are organized where there are different models for each component. At the end, simulations are presented as an example of using the library. Then, the chapter ends with a conclusion.

4.1 Library Motivations

The motivations of building the *EBC* library come form the increasing interests in *EBC* systems in the past years and due to the lack of theories comparing to the fixed rate (*FR*) control systems. The library meant to be designed and organized to have all the *EBC* system's components (event generators, controllers and sensors) and a variety of models for each component in one library. Hence, it gives the user a freedom to build his own *EBC* scheme by simply dragging and dropping blocks then connect them. Furthermore, building and simulating different *EBC* schemes with different parameters will be a great help to understand the theory behind. However, there is no single standard scheme of event-based. Instead, there are many different schemes encountered, depending on which event generators are used with sensor and controller, also on the type of control signal generator. That variety in event-based schemes is also a motivation to have such a library in order to try a new combinations and compare the performance among the different schemes. Hopefully, it will lead us to build a unitary *EBC* scheme.

4.2 Organization and Components

This section discusses the EBC system's components separately and how it related to the library organization since the components were grouped based on EBC structure. We are going to discuss them in the same order that they are organized in the library.

4.2.1 Event Generators (EG)

Choosing the EG decides the triggering mechanism in EBC system. The EG is designed as a separate entity and it connect with either EB controller or EB sensor. The inputs for the EG are the set point (SP) signal and the control variable (CV). The output from EG is boolean (EV) where its value is toggled each time an event is triggered and the other output is the CV which it only passes through the EG to the sensor or the controller. Overall, there are four different EG modules, each one represents a triggering rule.

1. SOD Error Difference
2. SOD Error
3. SOD Control Variable Difference
4. SOA Control Variable Difference

Each rule is modelled twice, one time for the controller and another for the sensor where the only difference between them is that the one for controller has additional output for the SP signal where it is only passes through.

4.2.2 Controllers (C)

The library has concentrated in the PID controllers and established different PID models as we have shown in section 3.2. Besides to the basic inputs, SP and CV, there is an additional input in the block related to EB nature called $TRIG$. This input is connected to EG's output EV where the change in its value indicates a new event has been triggered hence a new control signal will be calculated. In additional to the control signal, there is a new output EV has been added to the control module. Similar to the variable EV in the EG's output but used for different purpose. This boolean

output in controller is useful in case of a network system where there are exist a several controllers. Then, the output EV is used to announce that the corresponding controller has updated the CS, thus it will enable whatever comes in sequence after. However, all the modules of PID controllers are listed below have been modelled for each control signal generators, namely ZOH and IH.

1. Fixed Rate PID
2. Arzen's PID
3. Durand's PID
4. Durand's PID with Saturation
5. Durand's PID with Exponential Forgetting Factor
6. Durand's Hybrid PID
7. Past Values General Controller

And a special controller model is established for the past values technique that has been mentioned in the previous chapter 3. The past values general controller takes any controller that is entered as a transfer function. Note that this type of controller is exclusively works with the past values sensor that transmits the past measured variables and the controller save the past control signals in an internal buffer. The reference signals have been also saved in an internal buffer.

4.2.3 Sensors (S)

In the library, four different sensors have been modelled. They all provide the feedback signal to the controller but in a different mechanisms, depends on the type of the closed loop system whether it is EB or FR and if a time-out safety condition has been added or not.

4.2.3.1 Fixed Rate Sensor

In FR sensor, the CV is fed to the controller periodically (fixed time based). Moreover, the contentious CV is discretized with a fixed sample time equal to q_s without taking into account any other factors.

4.2.3.2 EB Sensor

In EBC system, the sensor is selected in the most of time as the single source of events. Particularly, EB sensor transmits a new CV based on event where the connected EG is periodically checking the violation of the selected triggering rules. The sensor has a boolean input $TRIG$, as in the controller, that is used to detect the coming new events from the EG's output. Moreover, the sensor has the EV boolean outputs just as in the controller that is similarly used to enable the next sequence action.

4.2.3.3 EB Sensor with Safety Time-Out

This model is exactly same as EB sensor with an extra added feature. In order to keep the sensor alive, unconditional events are triggered after passing a certain amount of time with no receiving any event by EG (the $TRIG$ value has not changed).

4.2.3.4 EB Sensor Past Values

It has the exact concept of EB sensor with safety time-out, but with some difference in the output form. However, when an event occurs whether by the EG or forced by the safety time-out, the sensor will not only transmits the present value of the controller variable $CV(t)$, but also an integer number (k) of the past values $(CV(t-k))$ and it is used exclusively in the library with the past value general controller

4.3 Example and Conclusion

A simulation consists of FR system and EBC system compared with contentious time (CT) system is shown as an example of creating EB control loops using the components in the library by dragging and connecting them, see figure 4.1. In this example, a first order plant and a step input reference are used in all the systems, but each system has a different controllers (PID). In addition, the FR and EBC systems use different sensors. The system's components parameter conflagration are going to be discussed next.

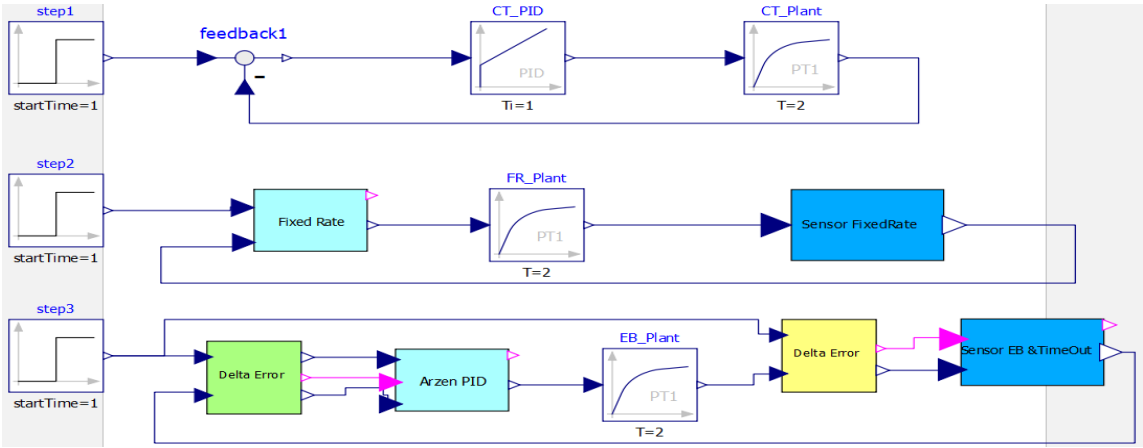


Figure 4.1: Block Diagram of the simulated system

4.3.1 Common Used Blocks in all Systems

The following components and their conflagration parameters are exactly same in all the systems.

4.3.1.1 Plant

The plant is a first order system with a gain (k) and a time constant (τ) equal to 1 and 2, respectively.

4.3.1.2 Input Reference

From OpenModelica blocks built-in library, a step input is used as an input reference (Set Point, SP). The step height is equal to 1 and it start at time equals to 1 second.

4.3.2 Blocks are Used Only in FR System

Beside to the common components, the controller and the sensor are different in the FR system. The used controller is fixed rate PID with ZOH as a CS generator. The configuration parameter is shown in figure 4.2. For the sensor, FR sensor is used with nominal sampling time ($q_s = 0.1\ sec$).

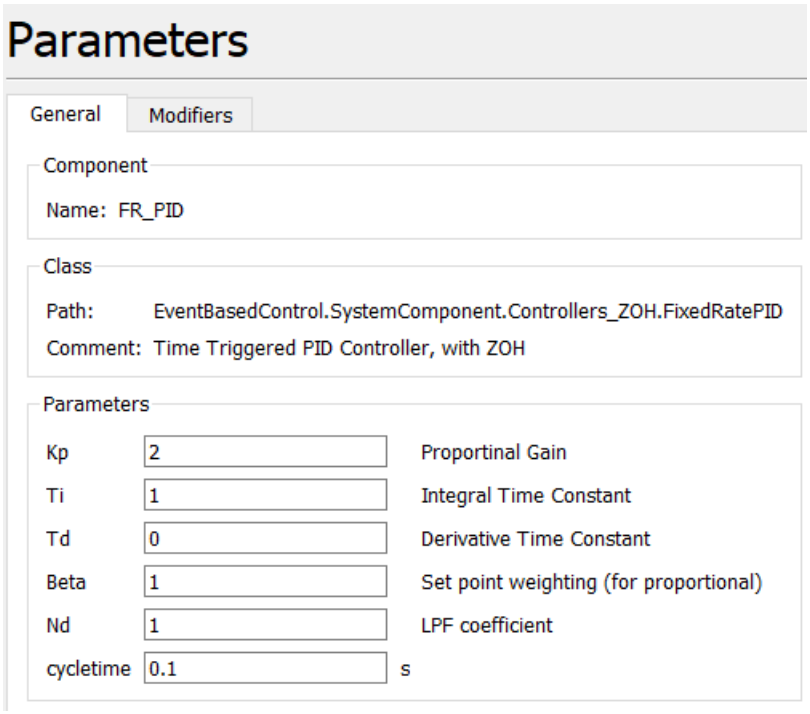


Figure 4.2: *Fixed Rate PID* configuration

4.3.3 Blocks are Used Only in EB System

Different controller and sensor are used in EB and FR systems.

4.3.3.1 Arzen PID

Arzen PID model is the chosen controller for EB system with the same common configuration parameters in Fixed Rate PID, shown in figure 4.2. The additional parameter in Arzen is for safety timeout ($h_{max} = 5$) to decide number of instants that allowed to be without event before force one. Since the controller is EB, it is necessary to select an event generator and connect it with controller. Delta Error EG is chosen with following parameters, $cycletime = 0.1$ sec and $threshold = 0.1$.

4.3.3.2 EB Sensor with Time-out

The configuration parameters for the sensor are showed as following: Nominal sampling time ($q_s = 0.1\ sec$). For safety time-out, the integer multiple of q_s ($N = 10$). EG must connect with EB sensor. The Delta Error EG is selected with ($cycletime = 0.1\ sec$ and $threshold = 0.1$).

4.3.3.3 Blocks are Used Only in CT System

All the used blocks in CT system have been taken from OpenModelica built-in library. Moreover, no special sensor model is needed to provide the feedback signal where the plant output (CV) is directly connected as an input to Feedback block which takes also the input reference as input too. the output of Feedback block ($error$) is the input to the CT controller (PID block from OpenModelica). The PID's parameters are the same as in FR and EB $PIDs$ (k=2, T_i=1, T_d=0 and N_d=1).

4.3.4 Simulation Result

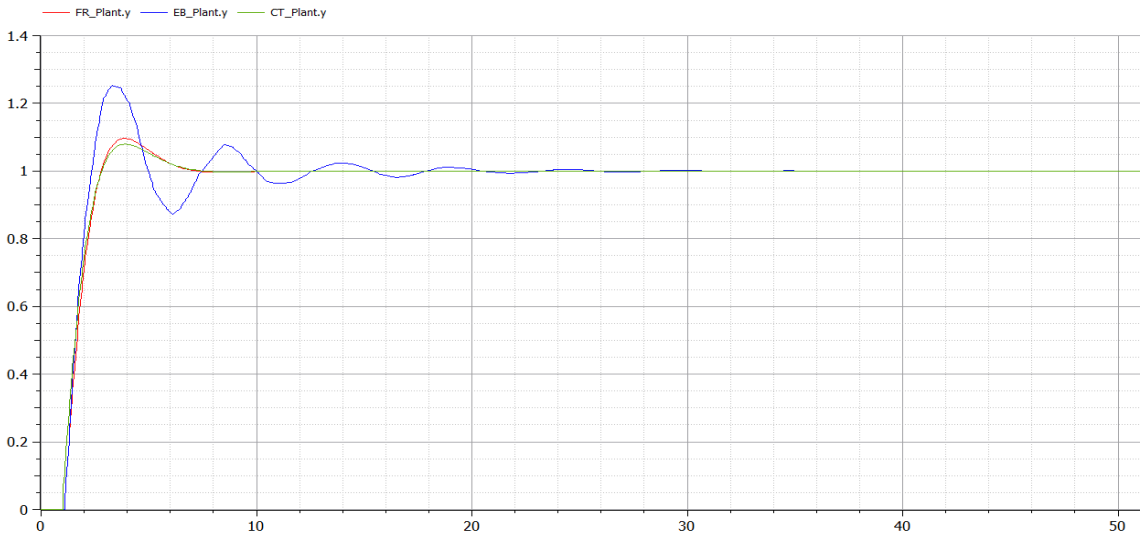


Figure 4.3: Response for FR,EB and CT systems with sampling times ($cycletime = 0.1\ sec$, $q_s = 0.1\ sec$)

After running the simulation for 60 seconds. The plant response of the three closed loop systems are shown in figure 4.3. The figure shows that

CT controller provides the best response, less overshoot and the fastest to settle down, and so it will be our reference for comparison. The response of FR system comes in the second place, relatively close to the CT system's response with slightly bigger overshoot. The FR sensor sent feedback values periodically (600 events) and the controller needed to compute a new control signals also periodically (600 events). Obviously, the EB system's response has higher overshoot and the longest settling time. On the bright side, the EB system has significantly reduced the number of events for both the controller and the sensor where they triggered only 150 and 70 events, respectively. Comparing to FR system, the number of events in EB is 75% less for the controller and 83% for the sensor.
The figure 4.4 marks all the events triggered in the EB sensor where each event transmits data to the controller. And distinguishing between those events that originated by the violation of the threshold, $EventTimeStamp = 1$, from the ones that has been triggered due to the safety time-out expiring, $EventTimeStamp = -1$. In the absence of events, $EventTimeStamp = 0$. Notably, all the events that were originated by violation the threshold are triggered before the system is settled down and when it settles, after 25 seconds, all the events afterwards are originated by the time-out condition in order to keep the system alive.

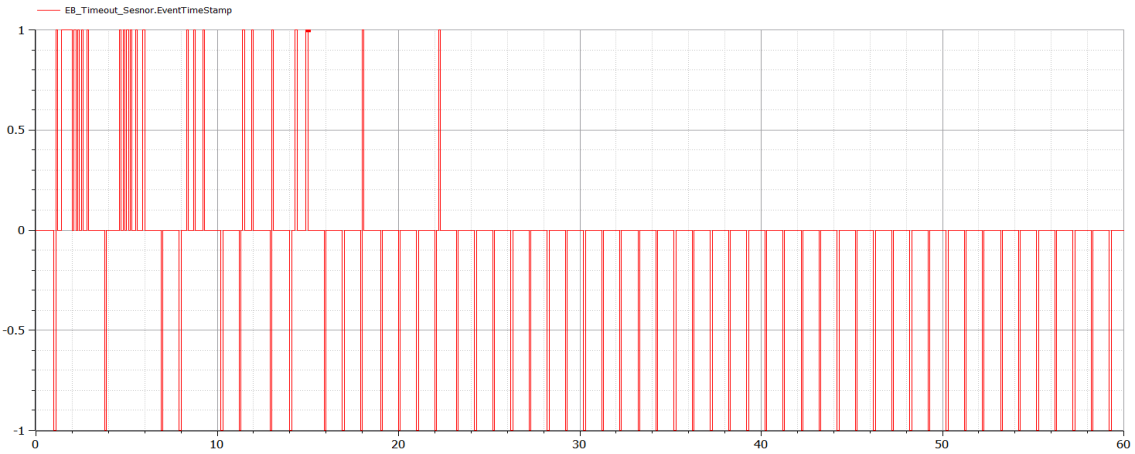


Figure 4.4: Event time-stamp for EB sensor with sampling time $q_s = 0.1$ *sec*

It can easily vary the parameters and re-simulate the system. For instance, the sensor's sampling time in FR and EB is decreased to 0.05 *sec*. Figure 4.5 shows that the FR system's response will act very similar to the

CT system, but the cost is increasing in the number of events to double (1200 events). The figure 4.5 shows also that EB has now a better performance (less osculation and smaller settling time). Again, the pay off is the increased number of events. The sensor has triggered 128 events while the number of the control events has not changed (150 events). Despite of the increasing in the number of events, the sensor in EB system has triggered much less events compared with the number of events triggered in FR, 89% less.

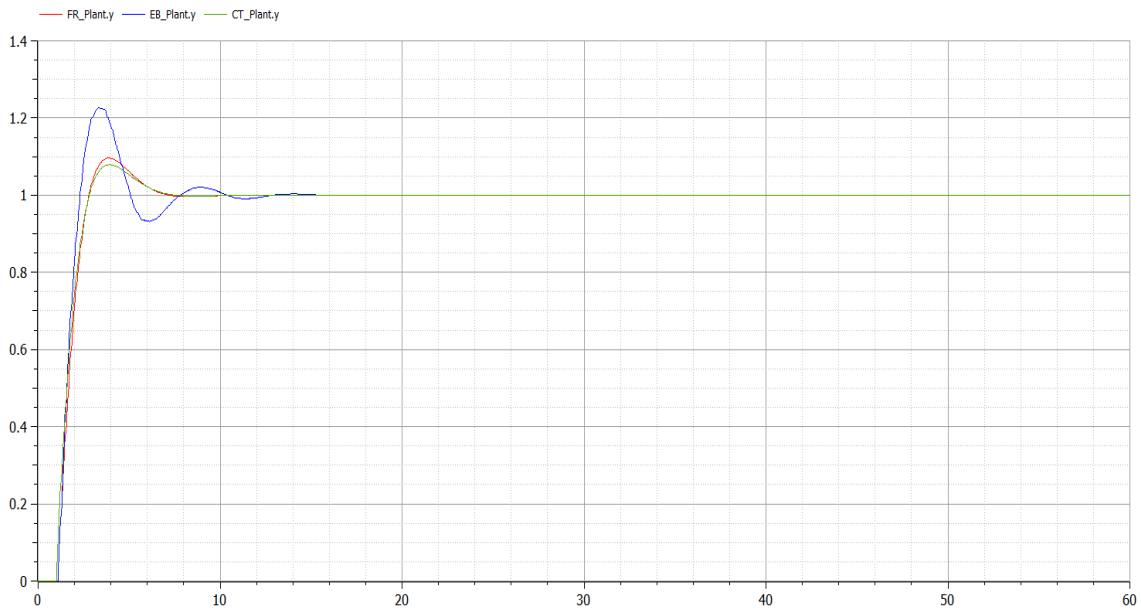


Figure 4.5: Response for FR,EB and CT systems after reducing the sampling times ($cycletime = 0.1\ sec$, $q_s = 0.05\ sec$)

4.3.5 Conclusion

The increased need to a tool that allows building and simulating EBC systems is significantly crucial to understand the theory behind. Moreover, the flexibility to build a wide range of different EBC schemes, not only the familiar ones is one of the keys behind creating the EBC library. Importantly, modifying the system's components is fairly simple by the library. where the user needs only to remove and add blocks. Also the library has made it simple to vary system parameters as it has been shown in the first example. Above

all, the library enhance studying and comparing several EBC components as well as analysing the system behaviour.

Chapter 5

Conclusions

This dissertation dealt with the problem of inconsistency between the controller and plant states. Background on the Event Based (EB) control paradigm was first given and a literature studies was reviewed.

In the beginning of chapter 3, it has been introduced the main general components in the EB control loop, including the EG with a various triggering rules and also a several models for the PID controllers. Then it presented the analysis and development of the EB control model that was resulted from using the past values. Writing the state space representation for the EB control loop has revealed two state matrices, run and hold modes, and showed the switching nature between these two modes. The plant states evaluate in the same manner whether it was in the run or hold mode. In contrast, the controller states is evaluate differently. And it is not always possible to determine the state of the controller in case of hold mode. For this reason, only strictly proper controller was considered in this work.

This chapter 4 was dedicated to the technological and implementation-related point of view. First, it presented the motivations of creating the Modelica library. And the library components were shown and structures the represented control blocks so as to separate sensor, event generators and controllers, for maximum flexibility. Then, an example on using the library was demonstrated.

Concerning future developments of this work, a further investigate the stability of event-based control loops encompassing past samples transmission and study the case of non strictly proper controller. Also, there is much room to expand the library by adding new models for the different event-based components.

Dedication

In dedication to all the people that helped me in the realization of this work, in particular, prof. Alberto Leva for the opportunity of developing this research activity, guidance, and unrelenting support throughout this process. A special dedication goes to my family, in particular, my mother, father and grandparents: it is thanks to their support and encouragement if today I have managed to achieve this goal. Also I am thankful for my friends, the affection and support they have shown me make this goal even more precious.

I would like to thank my partner for her unremitting encouragement that started from the very beginning of this journey till the end. Put simply, I have never met anyone who believes in me more. Thank you for making me more than I am.

Bibliography

[1] Astrom and J. Karl, *Event Based Control*, pp. 127–147. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[2] B. Hensel, J. Ploennigs, V. Vasyutynskyy, and K. Kabitzsch, "A simple pi controller tuning rule for sensor energy efficiency with level-crossing sampling," 2012.

[3] J. Chac´on, J. S´anchez, L. Yebra, A. Visioli, and S. Dormido, "Experimental study of two event-based pi controllers in a solar distributed collector field," in *12th European Control Conference, Z¨urich, Switzerland*, pp. 626–631, 2013.

[4] J. Chac´on, J. S´anchez, L. Yebra, A. Visioli, and S. Dormido, "An adaptive pi controller for room temperature control with level-crossing sampling," in *2012 UKACC International Conference on Control, Cardiff, United Kingdom*, pp. 197–204, 2012.

[5] A. Leva, F. Terraneo, and W. Fornaciari, "Event-based control as an enabler for high power density processors," in *2nd International Conference on Event-based Control, Communication, and Signal Processing, Krakow, Poland,*, pp. 1–8, 2016.

[6] W. P. M. H. Heemels, K. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control,"

[7] Arzen and Karl-Erik, "A simple event-based pid controller," no. 2, pp. 423–428, 1999.

[8] S. Durand and N. Marchand, "Further results on event-based pid controller," in *European Control Conference 2009*, pp. 1979–1984, 2009.

[9] A. Leva and A. V. Papadopoulos, "Tuning event-based industrial controllers with smapling stability guarantees," *Journal of Process Control*, vol. 23, pp. 1251–1260, 2013.

[10] S. Durand and N. Marchand, "An eventbased pid controller with low computational cost," 2009.

[11] M. Beschi, A. Visioli, J. Sánchez, and S. Dormido

[12] B. Hensel, J. Ploennigs, V. Vasyutynskyy, and K. Kabitzsch, "Some improvements on event-based pid controllers," in *32nd Chinese Control Conference*, 2013.

[13] M. Beschi, S. Dormido, J. Sánchez, and A. Visioli, "An event-based pi controller autotuning technique for integral processes," in *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–6, 2015.

[14] C. Diedrich, L. Winkel, and T. Blevinsd, "Function block applications in control systems based on iec 61804," vol. 43, no. 1, pp. 123–131, 2004.

[15] M. Rabi and K. H. Johansson, "Event-triggered strategies for industrial control over wireless networks," in *4th Annual International Conference on Wireless Internet, Maui, Hawaii, USA*, 2008.

[16] J. Sanchez, A. Visioli, and S. Dormido, "A two-degree-of-freedom pi controller based on events," *Process Control*, vol. 21, pp. 639–651, 2011.

[17] J. Sanchez, A. Visioli, and S. Dormido, "Pid control in the third millennium (r. vilanova and a. visioli (eds.))," pp. 495–526, 2012.

[18] V. Vasyutynskyy and K. Kabitzsch, "Implementation of pid controller with send-on-delta sampling," 2006.

[19] V. Vasyutynskyy and K. Kabitzsh, "First order observers in eventbased pid controls," in *5IEEE Conference on Emerging Technologies and Factory Automation*, pp. 1–8, 2009.

[20] V. Vasyutynskyy and K. Kabitzsh, "A comparative study of pid control algorithms adapted to send-on-delta sampling," in *IEEE International Symposium on Industrial Electronics*, pp. 3373–3379, 2010.

[21] M. Miskowicz, "Send-on-delta concept: An event-based data reporting strategy," vol. 6, pp. 49–63, 2006.

[22] J. Otanez, P.and Moyne and D. Tilbury, "Using deadbands to reduce communication in networked control systems," in *American Control Conference, Anchorage, USA*, pp. 3015–3020, 2002.

[23] V. Pawlowski, J. L. Guzm´an, F. Rodr´ıguez, M. Berenguel, J. Sanchez, and S. Dormido, "Simulation of greenhouse climate monitoring and control with wireless sensor network and event-based control," in *13th IEEE International Conference on Emerging Technologies and Factory Automation*, 2008.

[24] A. Leva, F. Terraneo, and S. Seva, "Periodic event-based control with past measurments transmission," 2017.

[25] X. Wang and M. D. Lemmon, "Event design in event-triggered feedback control systems," IEEE, 2008.

[26] A. Seuret, C. Prieur, S. Tarbouriech, and L. Zaccarian, "Lq-based event-triggered controller co-design for saturated linear systems," *Automatica*, vol. 74, pp. 47–54, 2016.

[27] V. H. Nguyen and Y. S. Suh, "Networked estimation with an area-triggered transmission method," 2008.

[28] M. Miskowicz, "Asymptotic effectiveness of the event-based sampling according to the integral criterion," *Sensor*, 2007.