Tuning of event-based industrial controllers with simple stability guarantees

Alberto Leva[a,*], Alessandro Vittorio Papadopoulos[a]

[a] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano Via Ponzio 34/5, 20133 Milano, Italy*

Abstract

This manuscript deals with the tuning of event-based controllers. By suitably constraining in a coordinated manner the controller discretisation and the event triggering rule, stability of the closed loop system is ensured, through an analysis that evidences and exploits its switching nature as induced by the event-based controller realisation. Such a sufficient condition, simple to enforce in practice, allows to take standard tuning rules, conceived for continuous-time controllers, and apply them to event-based realisations in a straightforward manner. The manuscript refers to the PI(D) controller structure, but extensions can be envisaged. Both simulation examples and an experimental test are reported.

Keywords: event-based control; controller tuning; process control.

1. Introduction

Event-based control has been gaining much interest in the last years, as shown by works like [2] and the papers quoted therein. Summarising, one may view event-based control as a means to acquire measurements, take decisions and/or apply

*Corresponding author: voice (39) 02 2399 3410, fax (39) 02 2399 3412.

Email addresses: `alberto.leva@polimi.it` (Alberto Leva), `papadopoulos@elet.polimi.it` (Alessandro Vittorio Papadopoulos)

actions "only when needed"—opposite to fixed-rate control, where those events are triggered periodically. A thorough use of event-based control can yield numerous benefits, including reduced traffic in networked systems [23], lower actuator wear, and so forth. In this work we focus on using the event-based paradigm to reduce the number of sensor transmissions, which is very important for example in the presence of battery-operated wireless devices.

As vastly discussed in the literature, a major problem with event-based control is the impossibility of analysing the system with a well established and powerful theory as that available for the fixed-rate case. Therefore, many viewpoints on the matter have been proposed. Conditions on tuning parameters for the existence of equilibrium points have been investigated, see, e.g., [4]; *ad hoc* tuning rules have been sought [21, 22], and even modifications of the most commonly used laws – typically, PI/PID-type ones – have been introduced [18]. More recently, [15] provided a relevant advance toward a unified problem treatise, by adopting a state feedback approach including a model of the continuous-time system, versus which the current plant state is evaluated, and a disturbance estimator. The approach was extended in [10] by proving asymptotic tracking properties, specialised to the PID structure, and tested experimentally.

In general, from the analysis viewpoint, one can describe fixed-rate control by saying that at a constant rate (a) a measurement of the controlled variable is taken, (b) a control signal value is computed, and (c) that value is actuated. In other words, the chain from sampling to actuating, is triggered all together by a single, periodic source of events. Correspondingly, event-based control can be seen to differ in *two* senses: (a) the time between events is not constant, and (b) there can be multiple sources of events.

Referring to Figure 1, a variety of event-based schemes is encountered, depend-

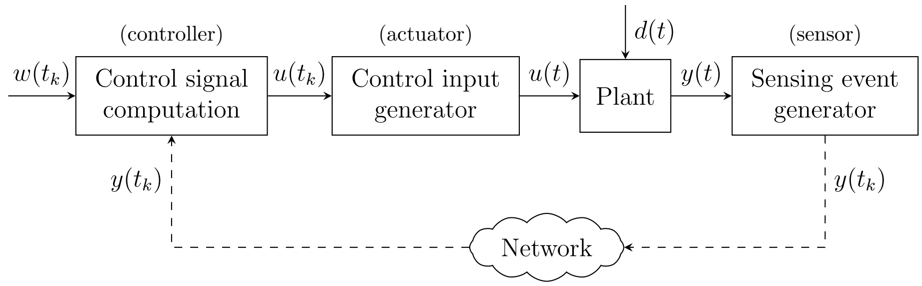

Figure 1: Event-based control loop.

ing on how the event source(s) and triggering rule(s) make the sensor, the controller and the actuator interact. The difficulty of establishing a uniform and general design paradigm is apparent, and evidenced by noticing that the application-oriented literature dominantly refers to the SISO single-loop case.

This manuscript has the same scope, and more specifically concentrates on SISO loops where a single event source is present. In such a context, the source of complexity is better qualified as the interplay between the control law, the event source, and the triggering mechanism. Even in this narrowed scope, it is in fact well known that if the controller design phase does not account for the envisaged event-based realisation, a number of undesired effects may be observed, such as performance degradation or limit cycles [6, 25].

A major point of this work is that if the mentioned scope restriction is accepted, still a lot of industrially relevant cases can be addressed, and it is possible to formulate models that allow for a straightforward analysis. This results in quite peculiar a viewpoint, that (partially) sacrifices generality in the favour of rigorous property assessment and design simplicity; its main contribution can be summarised as follows.

- Under the hypotheses of Section 2, a *sufficient* but very simple condition is presented in Section 3, that ensures asymptotic stability of the closed-loop system with an event-based controller derived from a linear, time-invariant, continuous-time one, in turn synthesised on a nominal model, of the same nature, for the process under control.

- Said condition is independent of the employed event triggering rule, as exemplified in Section 4.
- The so achieved free selection of the triggering rule is exploited in Section 5, to devise one that yields significant transmission savings, while providing protection against undesired event hauls.
- Section 6 illustrates, by providing both specific examples and a general procedure outline, how the proposed condition allows to tune event-based controllers with rules not conceived for such a realisation, thereby allowing to re-use a vast set of previous results [17].
- Sections 7 and 8 respectively report some simulation examples, to evidence the proposal strength, and an experimental test, to show its practical applicability.

As a final remark, it is worth noticing that recent literature works distinguish "event-" and "self-triggered" control (see for example [9]). The results of this work are applicable in both contexts, and in some sense orthogonal to their distinction, as the stability condition presented herein is in fact independent of the event generation mechanism.

2. General hypotheses

Any digital controller closes the loop only at some instants, while for all the remaining time it operates in open loop, no matter how the hold functionality is realised [2]. The underlying theory of fixed-rate control requires that said instants be evenly spaced, and that at each of them, sensing and actuation occur synchronously.

In this work, we introduce some industrially realistic hypotheses that lead to consider a class of event-based control systems that is in some sense the closest to the fixed-rate case. For simplicity, among the various schemes in the literature – see, e.g., [1, 11, 24] – we refer here to that in which the only event-based information flow originates from the sensor. The full set of assumed hypotheses can be stated as follows.

Hypothesis 1. *The process under control is described by the linear, time-invariant (LTI) single-input, single-output (SISO) model*

$$\begin{cases} \dot{x}_P(t) = A_P x_P(t) + b_P u(t-\tau) \\ y(t) = c_P x_P(t) \end{cases} \tag{1}$$

where t is the continuous time, $u(t) \in \mathbb{R}$ the control signal, $y(t) \in \mathbb{R}$ the controlled variable, $x_P(t) \in \mathbb{R}^{n_P}$ the process state vector, $A_P \in \mathbb{R}^{n_P \times n_P}$, $b_P \in \mathbb{R}^{n_P \times 1}$, $c_P \in \mathbb{R}^{1 \times n_P}$ constant matrices, and finally $\tau \in \mathbb{R}$, $\tau \geq 0$, a constant delay.

Note that model (1) is strictly proper, without any loss of generality for our purposes.

Hypothesis 2. *A continuous-time LTI SISO controller that stabilises the* nominal *closed-loop system containing model* (1) *is available, and has the form*

$$\begin{cases} \dot{x}_R(t) = A_R x_R(t) + b_R\,(w(t) - y(t)) \\ u(t) = c_R x_R(t) + d_R\,(w(t) - y(t)) \end{cases} \tag{2}$$

where $x_R(t) \in \mathbb{R}^{n_R}$ is the controller state, $w(t) \in \mathbb{R}$ the reference signal to be followed *by $y(t)$, $A_R \in \mathbb{R}^{n_R \times n_R}$, $b_R \in \mathbb{R}^{n_R \times 1}$, $c_R \in \mathbb{R}^{1 \times n_R}$ constant matrices, and $d_R \in \mathbb{R}$ a constant scalar.*

Controller (2) can be the result of an (auto)tuning procedure, and encompass direct input/output feedthrough, like, e.g., a PI(D).

Hypothesis 3. *Controller* (2) *is realised with digital technology, and computes the discrete-time control $u^*(k)$ at events, which occur at time instants t_k counted by an integer $k \in \mathbb{N}$, and in general not evenly spaced in time.*

Hypothesis 4. *Events are triggered by a single source (that here we assume to be the sensor).*

Hypothesis 5. *The time between two events is an integer multiple of a quantum $q_s \in \mathbb{R}, q_s > 0$.*

$$\forall t_h \leq t_k,\ t_k - t_h = \varsigma(k,h) q_s, \qquad \varsigma_{k,h} \in \mathbb{N}.$$

According to Hypothesis 4, two quantities can be defined

- the *a priori step duration* $\overline{T}_s(k)$ that is decided at the k-th event,
- and the *a posteriori step duration* $\underline{T}_s(k)$, i.e., the time actually elapsed from the k-th to the $(k+1)$-th event.

In practice, events can occur at the termination of $\overline{T}_s(k)$ or earlier, no matter why. In the former case $\underline{T}_s(k) = \overline{T}_s(k)$, while in the latter $\underline{T}_s(k) < \overline{T}_s(k)$. Notice that the event generation mechanism is in part reactive and in part proactive, the timeout being in fact the simplest way to decide when the next event has to occur (see the distinction between event- and self-triggering control mentioned above).

Furthermore, Hypothesis 5 is well consistent with the way sensor electronics is typically designed. Most frequently, in fact, the sensor has a low-power part that is always active and polls the measured variable at frequency $1/q_s$. A high-power part takes conversely care of transmitting "when deemed necessary", i.e., based on a *triggering rule*, and is kept off otherwise.

Hypothesis 6. *If process* (1) *contains a delay, this can be approximated in the control-relevant frequency band by a rational transfer function, so that one can take as* nominal *continuous-time process model one with rational dynamics only.*

Assuming this may seem peculiar, but in fact many (auto)tuning methods – like for example the well known IMC-PID one considered later on [8, 20] – rely on such models, typically obtained via Padé approximations. And even if the used tuning method is not of this type, in any non-pathological case it is possible to approximate a delay, within the control band, with simple enough a rational expression. No doubt this could somehow diminish the generality of the proposed approach, but nonetheless the variety of the usable tuning rules is still very large.

Hypothesis 7. *There is an upper bound for the time between two subsequent events, i.e.,*

$$\forall k,\ \sigma(k) \in \Sigma = \{1, \dots, N\} \subset \mathbb{N}, \quad 1 \leq N < \infty,$$

where

$$\sigma(k) := \varsigma(k+1, k).$$

This is realistic, as for safety reasons all real sensors encompass some "keep-alive" timeout, at the end of which an event is triggered unconditionally.

Hypothesis 8. *The control signal is kept constant between two subsequent events, as in the extremely frequent case where a zero-order holder is used.*

Hypothesis 9. *When an event is triggered by the sensor, this results in the computation and actuation of a new control value. The delay between the triggered event and the control actuation is either negligible or known and constant, so that it can be taken as a part of the process model.*

This is the most strict hypothesis among those introduced, but is definitely realistic in at least two cases, both of interest for process control. The first one is when sensor, controller and actuator are co-located, and the reason for using an event-based controller is to reduce the actuator wear. In this case, the delay between sensor event and actuation is practically negligible. The second case (more central to this work) is when sensor, controller and actuator communicate via a network, but the underlying communication protocol is designed in such a way to practically eliminate packet collisions, that are the primary source of network-induced (variable) delays. At present not all protocols are capable of doing that, but a great research effort is being spent on the matter, see for example [7, 16, 19, 26], and solutions suited for the addressed context are arising; for example, in [14] a synchronisation scheme is proposed that, thanks to a novel and completely control-theoretical design, permits to make virtually any existing communication protocol slotted, thus making communication delays practically invariant. When such solutions will eventually become part of industrial systems, the hypothesis under question will be safely applicable to even more real-life cases.

3. A simple sufficient stability condition

Suppose model (1) to be an exact description of the process. This section shows how a simple sufficient condition can be obtained to ensure stability of the control system (in nominal conditions) where the realisation of controller (2) is event-based.

In the first place, recall Hypotheses 1, 5 and 8, and assume that the nominal continuous-time process model contains a rational approximation of a possible delay (see Hypothesis 6). In this case, preserving the matrix notation of (1) for simplicity by just assuming that the state is conveniently augmented, the nominal process is described by

$$\begin{cases} x_P^*(k+1) = A_P^*(\underline{T}_s(k)) \cdot x_P^*(k) + b_P^*(\underline{T}_s(k)) \cdot u^*(k) \\ y^*(k+1) = c_P \cdot x_P^*(k+1) \end{cases} \tag{3}$$

where the "$*$" superscript denotes signals sampled at events – e.g., $x_P^*(k) := x_P(t(k))$ – and

$$A_P^*(\underline{T}_s(k)) := e^{A_P \underline{T}_s(k)}, \quad b_P^*(\underline{T}_s(k)) := \int_0^{\underline{T}_s(k)} e^{A_P(\underline{T}_s(k)-\xi)} b_P \, \mathrm{d}\xi. \tag{4}$$

Coming to the controller, let it be turned at the beginning of step k into a discrete-time one by some method of choice, using as discretisation period the *a posteriori* duration of the previous step, now known, so as to make $u^*(k)$ the best *replica* of its continuous-time counterpart made possible by that method. This means computing $u^*(k)$ as the output of the dynamic system

$$\begin{cases} x_R^*(k) = A_R^*(\underline{T}_s(k-1)) \cdot x_R^*(k-1) + b_R^*(\underline{T}_s(k-1)) \cdot (w^*(k-1) - y^*(k-1)) \\ u^*(k) = c_R^* \cdot x_R^*(k) + d_R^* \cdot (w^*(k) - y^*(k)) \end{cases} \tag{5}$$

where the mentioned discretisation method provides the matrix functions $A_R^*(T_s)$, $b_R^*(T_s)$, $c_R^*(T_s)$, and the scalar one $d_R^*(T_s)$—we do not explicitly indicate the dependence of those functions on the continuous-time matrices to lighten the notation. At the same instant, the process state and output are related to their values at the beginning of the previous step by (3), with the time indices shifted back by one.

Putting it all together, at step k we have an *a posteriori* closed-loop discrete-time system (the one that actually evolved) with state vector $x^*(k) := [x_P^*(k)\, x_R^*(k)]^T$, and dynamic matrix

$$A^*(\sigma(k)) = \begin{bmatrix} A_P^*(\sigma(k)q_s) - b_P^*(\sigma(k)q_s)d_R^* c_P & b_P^*(\sigma(k)q_s)c_R^* \\ -b_R^*(\sigma(k)q_s)c_P & A_R^*(\sigma(k)q_s) \end{bmatrix}. \tag{6}$$

This reveals the system's switching nature, $\sigma(k) \in \Sigma$ playing the role of the switching signal. Note that the system cannot be modelled as a piecewise linear one – which would have simplified the analysis – because the timeouts cause events to be generated also with time guards.

To guarantee stability of the event-based control system, given the unpredictability of $\sigma(k)$, it is required to prove the stability of the system with dynamic matrix (6) under arbitrary switching in Σ. To this end, based on the discussion above, an extremely simple sufficient condition can be expressed as follows.

Theorem 3.1. *A sufficient condition for the asymptotic stability of the system with dynamic matrix* (6) *under arbitrary switching in* Σ*, is that for each* $\sigma(k) \in \Sigma$ *the controller discretisation procedure make said matrix Schur, and with real distinct eigenvalues.*

Proof. Under the hypotheses, denoting by $\lambda_{j,\sigma(k)}$, $j = 1, \ldots, n_P + n_R$, the generic eigenvalue of $A^*_{\sigma(k)}$, there surely exists a nonsingular matrix $T(\sigma(k))$ such that

$$A^*_{d,\sigma(k)} := T(\sigma(k))^{-1} A^*_{\sigma(k)} T(\sigma(k)) = \text{diag}\left\{\lambda_{j,\sigma(k)}\right\}. \tag{7}$$

Let now P be a constant diagonal matrix with real elements, i.e., $P = \text{diag}\{p_j\}$,

$p_j \in \mathbb{R}$, $j = 1, \ldots, n_P + n_R$. It is immediate to show that the generic eigenvalue $\mu_{j,\sigma(k)}$ of $P - {A^*_{d,\sigma(k)}}^T P A^*_{d,\sigma(k)}$ is

$$\mu_{j,\sigma(k)} = \left(1 - \lambda^2_{j,\sigma(k)}\right) p_j. \tag{8}$$

Given that $|\lambda_{j,\sigma(k)}| < 1$ by construction, any matrix P with positive elements, thus symmetric and positive definite, makes ${A^*_{d,\sigma(k)}}^T P A^*_{d,\sigma(k)} - P$ negative definite. Provided that the law to update the controller states at step k makes the evolution of the closed-loop state-consistent with the corresponding fixed-rate running $\sigma(k)$ times with the nominal model, (8) ensures the existence of a Common Quadratic Lyapunov Function for the switching system with dynamic matrix $A^*_{\sigma(k)}$, i.e., its asymptotic stability under arbitrary switching in Σ. □

Apparently, Theorem 3.1 can be viewed as a problem-specific formulation of well known results, and the obtained condition can be quite conservative. However, the interest of that condition resides in its extreme simplicity, which makes it straightforward to ensure its validity while using a large variety of tuning rules, not conceived having an event-based realisation in mind.

Finally, concerning possible disturbances, one can notice that their influence on the stability of the closed-loop system, given its linear (switching) nature, can only be exerted by inducing a particular switching sequence. Therefore, once stability is guaranteed under arbitrary switching, it cannot be disrupted by construction.

4. An introductory application example

An introductory example of how the idea just proposed can be put to work is now given. Consider the first-order continuous-time process described by the transfer function

$$P(s) = \frac{\mu}{1+sT}, \tag{9}$$

where $T > 0$, thus limiting the scope to the asymptotically stable case, and without loss of generality also suppose $\mu > 0$. In state space form, (9) corresponds to $a_P = -1/T$, $b_P = \mu/T$, and $c_P = 1$. Applying the exact discretisation rule we have

$$a_P^* = e^{-T_s/T}, \quad b_P^* = \mu\big(1 - e^{-T_s/T}\big),$$

where T_s is the step duration, no matter for the moment whether *a priori* or *a posteriori*, thus the discrete-time switching process

$$P_{T_s}^*(z) = \mu\,\frac{1 - e^{-T_s/T}}{z - e^{-T_s/T}}, \tag{10}$$

where we adopt a transfer function notation for compactness, and evidence the switching nature by the T_s subscript.

Suppose that the tuning goal is to have the closed-loop equivalent continuous-time reference-to-output system behave like

$$T^\circ(s) = \frac{1}{1+sT_{CL}},$$

where $T_{CL} > 0$ is a desired time constant. This corresponds – adopting again the exact rule – to

$$T^{\circ *}_{\underline{T}_s(k-1)}(z) = \frac{1 - e^{-\underline{T}_s(k-1)/T_{CL}}}{z - e^{-\underline{T}_s(k-1)/T_{CL}}}$$

At the beginning of the generic k-th control step, take the *a posteriori* sampling time $\underline{T}_s(k-1)$, and obtain the discrete-time control law for the current step as

$$R^*_{\underline{T}_s(k-1)}(z) = \frac{1}{P^*_{\underline{T}_s(k-1)}(z)} \frac{T^{\circ *}_{\underline{T}_s(k-1)}(z)}{1 - T^{\circ *}_{\underline{T}_s(k-1)}(z)} \tag{11}$$

with the same notation used for (10), the $\underline{T}_s(k-1)$ subscript evidencing that also the controller has a switching nature, and additionally dictating how the switching signal is obtained from the previous closed-loop system's evolution.

Controller (11) makes the closed-loop system, viewed in the discrete time, exhibit as switching eigenvalues $e^{-\underline{T}_s(k-1)/T_{CL}}$, and the cancelled process one $e^{-\underline{T}_s(k-1)/T}$. Assuming of course $T_s, T_{CL} > 0$, that system falls in the Schur, real, positive and distinct eigenvalues case, thus being asymptotically stable under arbitrary switching as per Theorem 3.1. We omit the tedious but trivial computations on how the control law is computed to ensure the mentioned state consistence.

At this time, decide $\overline{T}_s(k)$ for the current step, apply $u^*(k)$, and let the system evolve until that time elapses, or an event is triggered by the sensor. Observe that the free motion of the closed-loop system is (in norm) strictly decreasing, thus even an event occurring *before* $\overline{T}_s(k)$ does not destroy arbitrary switching stability as seen, step by step, looking at the *a posteriori* system.

In a view to generalisation, some remarks are now in order.

- To guarantee stability of event-based control, adaptive discretisation is advantageous, at least as long as the proposed realisation approach is followed. On a similar front, conducting the analysis with k counting events – not continuous-time intervals – yields simplifications. The correspondence of the hypotheses of Section 2 to the addressed domain of (auto)tuning, particularly for process control, allows to consider the statements just made quite uniformly valid in that context.

- The sufficient stability condition of Theorem 3.1 is easy to ensure at least in cases analogous to the shown introductory example—a matter on which we shall return in Section 6.

- The previous considerations on the *a priori* and the *a posteriori* closed-loop system give sense to the idea of adopting the time elapsed from the last event to update the state, but at the same time not as the sampling period for the current control step. On the contrary, the determination of an *a priori* sampling time seems a good way to go. This can be done with techniques drawn from step size selection ones in the simulation domain [5], as in Section 5.

5. A triggering rule

Once stability is ensured *independently of the triggering rule*, it is possible to freely and safely select that rule to maximise the advantages sought when adopting the event-based framework. Basically, one wants the (*a posteriori*) control step duration

- to increase as rapidly as possible toward its allowed maximum if the sensor triggers no event, which typically occurs with the "send on delta" policy, i.e., when the controlled variable, polled by the sensor at rate $1/q_s$, differs in magnitude from the last transmitted one by more than a prescribed amount Δy;

- to allow reacting as soon as possible to an event, the minimum reaction time being q_s;

- and to avoid event hauls after the first one triggered by a controlled variable's variation.

To achieve that, once q_s and N are decided, a subset $\widetilde{\Sigma} = \{\widetilde{\sigma}_i\}$ of Σ is defined, with cardinality $\widetilde{N} < N$, so that $\widetilde{\sigma}_1$ be greater than 1, $\widetilde{\sigma}_1 q_s$ be a "small but reasonable" sampling period if adopted for a fixed-rate controller realisation, and $\widetilde{\sigma}_{\widetilde{N}} = N$. An example, assuming that a small but reasonable fixed-rate sampling time is 1, could be $q_s = 0.1$, $N = 1000$, and $\widetilde{\Sigma} = \{10, 20, 50, 100, 200, 500, 1000\}$.

The *a priori* period $\overline{T}_s$ is first initialised to $\widetilde{\sigma}_1 q_s$. Then, if a step of *a priori* duration $\widetilde{\sigma}_i q_s$ elapses, the next *a priori* period is set to $\widetilde{\sigma}_{i+1} q_s$, until $\widetilde{\sigma}_{\widetilde{N}}$ is reached. If conversely a step ends due to a sensor event, the next period is reset to $\widetilde{\sigma}_1 q_s$ and the system is forced to make it elapse. This temporary constraint results in possibly ignoring some events, which is however harmless because it was just stated that if the controller were realised as a fixed-rate one with sampling period $\widetilde{\sigma}_1 q_s$, the consequent latency – e.g., in reacting to a disturbance – would be acceptable.

The *a priori* step duration selection is summarised by the finite state automaton of Figure 2. In that figure, branches labelled with $\underline{T}_s = \overline{T}_s$ are traversed "by time-out", i.e., when the *a priori* step duration elapses; branches labelled $q_s \le \underline{T}_s < \overline{T}_s$ (se), where "se" stands for "sensor event", are traversed in the opposite case.

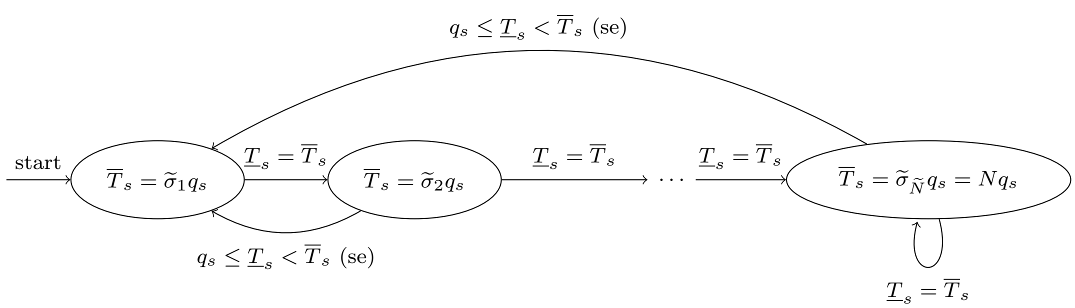

Figure 2: Finite state automaton for the *a priori* step duration selection (the time index k is dropped to lighten the notation). The acronym "se" indicates branches that are traversed due to a "sensor event".

An important remark is that, once stability is ensured as done herein, the reason for *sensor*-generated events becomes irrelevant, and thus at the sensor level one can freely introduce other event *causes* without any danger. This is relevant in the case of a set point change after the loop has settled to an equilibrium, thus causing the *a priori* step duration to be large. In this case, a set point variation will not provoke an event until said duration elapses, i.e., potentially long after the set point was varied.

To prevent such an undesired behaviour, one can take two basic countermeasures. The first one is to force the sensor to generate events based also on set point variations. This fully preserves the introduced hypotheses, but at the cost of additional communication toward the sensor. The second possibility is to force *one* control computation (not a controlled variable measurement) when the set point is modified (e.g., immediately after a step is applied, or a ramp is *started*). Doing so violates Hypothesis 4, as that value of the control signal will be computed with an "outdated" controlled variable—or equivalently, there is a second (sporadic) source of events. However, one can assume that said outdated value is close to the last transmitted one, otherwise some sensor events would have been triggered, and view the fact as an impulsive disturbance of unknown but moderate entity.

It is also worth noticing that the possible system jumps are limited. In particular, supposing that $\overline{T}_s = \widetilde{\sigma}_i$ two cases may happen

- if $\underline{T}_s < \overline{T}_s$, then the next a posteriori sampling period will necessary be $\underline{T}_s = \widetilde{\sigma}_1 q_s$;
- if $\underline{T}_s = \overline{T}_s$, then the next a posteriori sampling period will be $\underline{T}_s \in [q_s, \widetilde{\sigma}_{i+1} q_s]$.

Moreover, having given an *asymptotic* stability condition, the only possible source of limit cycles resides in the numerical quantisation effects. This makes it easy to govern said cycles by acting on the rule parameter(s), like, e.g., the threshold in the send on delta one. Incidentally, the impossibility of some jumps may allow to loosen the stability condition of Theorem 3.1.

To end the section, just a few words are in order on the choice of $\widetilde{\Sigma}$. In fact, one could set $\widetilde{\Sigma} = \{\widetilde{\sigma}_1, N\}$, not provisioning any intermediate value, which is consistent with the approach. However, if this choice is adopted, most likely the majority of control actions will be triggered by sensor events, and not by timeouts. If the send on delta triggering rule is employed, this in turn means that control actions will be almost invariantly computed in response to variations of the controlled variable that in magnitude exceed Δy. Such a situation is keen to generate larger actuator movements than that in which some control event is triggered by timeouts caused by a gradual step growth. Experience allows to conjecture that in general having some intermediate *a priori* step values favours a smoother actuator operation, and this is the reason for the adopted choice. Should this not be relevant, less intermediate values can be used without compromising the analysis.

6. Tuning the controller

In this section we focus on the PI(D) structure for convenience, although the shown ideas are more general. Also, we limit the scope to explicit model-based tuning rules. Such rules take as input a continuous-time process model $M(s, \theta_M)$, where θ_M is a parameter vector, and compute the vector θ_R of the PI(D) parameters with a *tuning formula* $\theta_R = f(\theta_M, \theta_S)$, where θ_S is a vector of specifications (e.g., a desired cutoff frequency and/or phase margin).

Quite interestingly, many rules of the addressed type operate by cancellation, and give rise to simple expressions for the open-loop nominal transfer function, thus providing a straightforward application of the proposed approach. A notable example is the well known and already mentioned IMC-PID, that starts from a continuous-time FOPDT (First Order Plus Dead Time) asymptotically stable process model

$$M_{FOPDT}(s) = \mu \frac{e^{-sD}}{1+sT}, \quad T > 0,\ D \geq 0, \tag{12}$$

and approximates the delay with a (1,1) Padé approximation, thereby adopting the nominal model

$$M(s) = \frac{\mu}{1+sT} \frac{1-sD/2}{1+sD/2}. \tag{13}$$

Taking as approximate model inverse and as IMC filter, respectively, the two transfer functions

$$Q(s) = \frac{1+sT}{\mu}, \quad F(s) = \frac{1}{1+s\lambda}, \tag{14}$$

where $\lambda > 0$ is the desired closed-loop dominant time constant, a continuous-time real PID controller is determined as

$$R(s) = \frac{Q(s)F(s)}{1-Q(s)F(s)M(s)} = \ldots = \frac{1}{\mu(D+\lambda)} \frac{(1+sT)(1+sD/2)}{s\left(1+s\dfrac{D\lambda}{2(D+\lambda)}\right)}. \tag{15}$$

Applying now the proposed idea, we discretise (13) and (14) with the exact rule. The cancelled poles of the discretised version of (13) are clearly $e^{-T_s/T}$ and $e^{-2T_s/D}$, where T_s has to be interpreted as the time-varying *a posteriori* step duration. Hence, the discrete-time IMC PID produces as closed-loop eigenvalues the triplet $(e^{-T_s/T},\ e^{-2T_s/D},\ e^{-T_s/\lambda})$, which fulfils the required stability condition provided that $\lambda \neq T$, $\lambda \neq D/2$ and $T \neq D/2$. The expression of the PID parameters is omitted for brevity, and any cancellation-based rule can be treated essentially in the same way.

If the chosen rule is not cancellation-based, computations may become complex and are in general more rule-specific, but the basic principle holds. Basically, one has to express the closed-loop eigenvalues, and check the stability condition for all the required multiples of q_s. This may be time-consuming, but it is an offline activity, and the availability of modern symbolic packages makes it affordable. We do not further delve into this matter here, to avoid presenting lengthy computations of little – if any – methodological interest.

7. Simulation examples

In this section, the proposed method is applied to some significant examples to prove its effectiveness.

7.1. Example 1

This example aims at showing the proposed controller's operation in nominal conditions. The considered process is described by the transfer function

$$P(s) = \frac{e^{-0.5s}}{1+2s}, \tag{16}$$

and an IMC PID is tuned for it with $\lambda = 2$. The obtained results are shown in Figure 3.

The upper plot reports the set point and the process variable with both the fixed-rate and the event-based controller, when the system is subject to ramp-like set point variations and to two load disturbance steps. The lower plot conversely shows the values of T_s, allowing to appreciate how the step duration grows as fast as possible, while at the same time avoiding event hauls. In particular, there are many transmissions when the set point is varying in a ramp-like manner, but less in response to a load disturbance step. To validate also the ideas about non-sensor events expressed in Section 5, two control events were forced at the beginning of

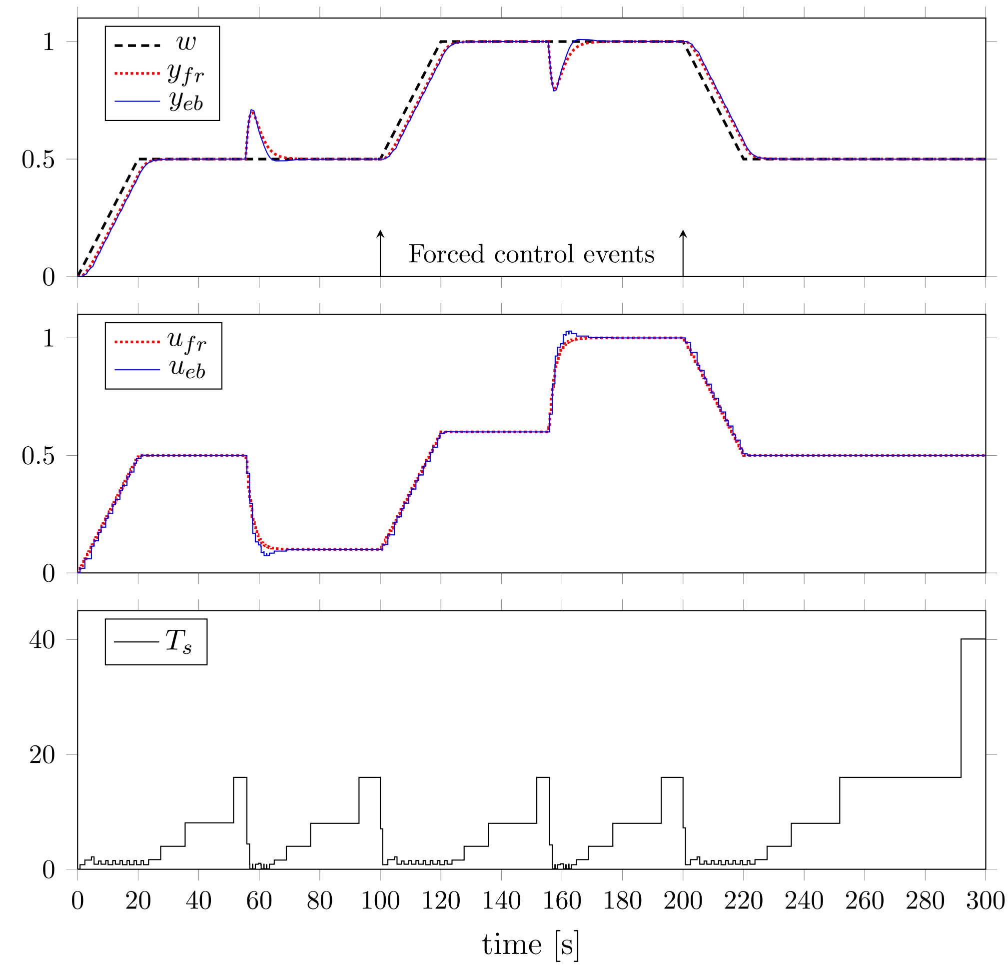

Figure 3: Results of simulation Example 1; in the two upper plots the dotted red line refers to the fixed-rate controller, the solid blue line to the event-based one.

the ramps (as marked in the figure). As expected, a single forced control event is enough to trigger the sensor-originated necessary ones. The controlled variable plots are practically identical, and no limit cycle is observed; interestingly enough, limit cycles do not appear even if the model contains the real delay term instead of the Padé approximation. Finally, evaluating the event-based transmission saving (with respect to the fixed-rate realisation with period $\lambda/5$) on the *scenario* considered in this example, leads to a result of about 90%, thereby significantly backing up the proposal. The choice of $\lambda/5$ for the fixed-rate realisation was made not to unduly favour the event based one.

It is also interesting to examine the effects of λ on the number of generated events and on the control quality loss with respect to a fixed-rate realisation. To this end, the Integral Squared Error (ISE) index is here used. Figure 4 shows the numerical results with $\lambda \in [0.2, 5]$: the left column reports the number of events (top) and the ISE (bottom) with the fixed-rate (fr) and the event-based (eb) controller, while the right column reports the ratios of the same quantities. Apparently, the gain in terms

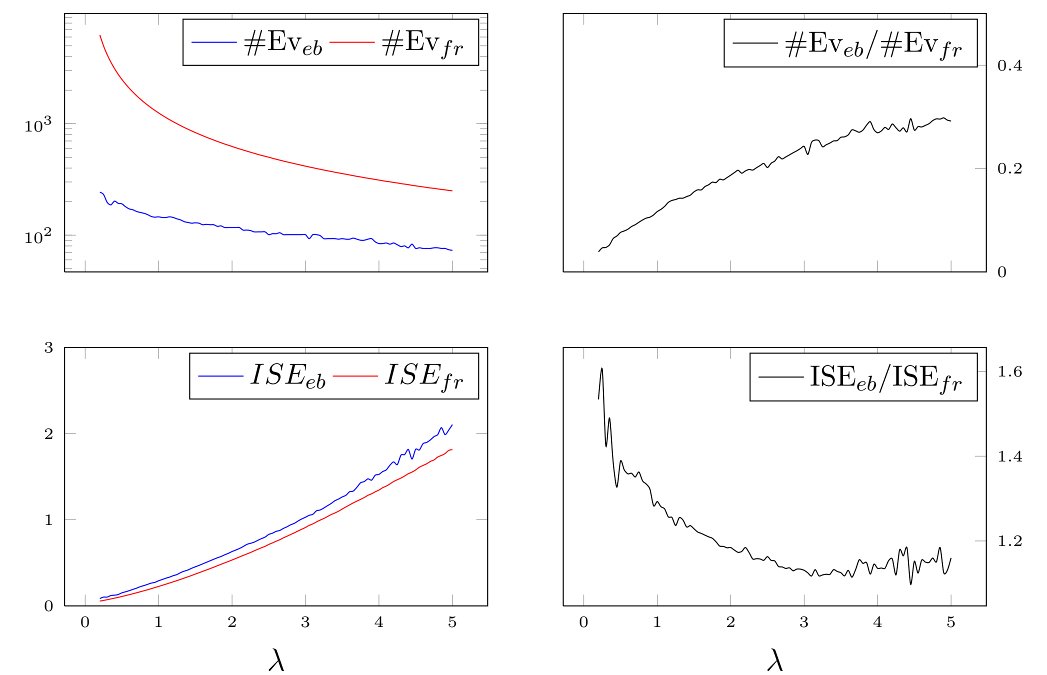

Figure 4: Results of Example 1 with different values of λ.

of saved events with the event-based realisation is significant, at the cost of a modest increment of the ISE. This synthetically indicates that the modification of the time domain transients obtained with the proposed technique, with respect to the "ideal" continuous-time ones, is comparable to that introduced by fixed-rate realisations.

7.2. *Example 2*

To better evaluate the control design effectiveness in nominal conditions on a *set* of FOPDT models, we consider the normalised delay process

$$P_n(s) = \frac{e^{-\frac{\theta}{1-\theta}s}}{1+s}$$

which corresponds to (12) with $\mu = 1$, $T = 1$ and the delay D set so as to achieve a desired value θ of the normalised delay $D/(T + D)$.

The proposed tuning method is applied with different values of θ, and selecting λ as

$$\lambda = \frac{1}{a} \cdot \frac{D + 5T}{5}$$

where a is interpreted as a required acceleration factor for the closed-loop dominant dynamics with respect to the process one; in particular, the test was conducted with $\theta \in [0.2, 0.6]$ and $a \in [0.25, 4]$.

An analysis similar to that of the previous section, produces the results shown in Figure 5. The ratio among the number of events with the event-based and the fixed-rate realisation is shown in the left plot, while the ratio among the two ISE values is in the right plot.

Also in this case, it is apparent that the improvement in terms of transmissions is significant, at the cost of a modest ISE increment.

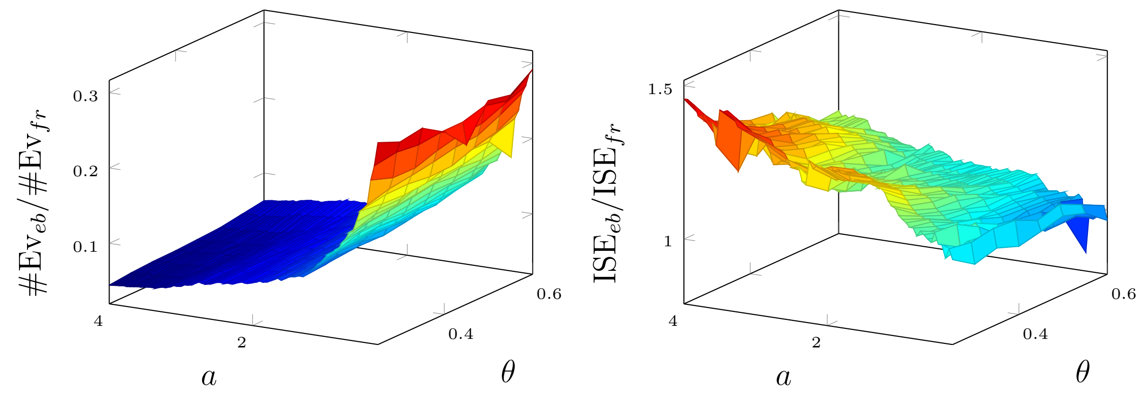

Figure 5: Results of Example 2 with different values of a and θ.

7.3. Example 3

This example applies the proposed controller to non-FOPDT processes, i.e., not in nominal conditions. Three types of process transfer functions are here used, taken from the benchmark set [3], namely

$$
\begin{aligned}
P_1(s) &= \frac{1}{(s+1)^n} && n = 2, 3, 4, 8 \\
P_2(s) &= \frac{1}{(1+s)\,(1+as)\,(1+a^2 s)\,(1+a^3 s)} && a = 0.1, 0.2, 0.5, 1.0 \\
P_3(s) &= \frac{1}{(1+sT)^2} e^{-s} && T = 0.1, 2.0, 5.0, 10.0
\end{aligned}
$$

The FOPDT models used for the tuning are parametrised together with the controller with the so-called "contextual" method applied to the PI IMC rule [13]. The reason for not using the entire batch is that the contextual IMC method is not particularly well suited for other structures, especially those with significant oscillatory and/or nonminimum phase behaviours. These issues concern the tuning method, however, and therefore are not related to the event-based realisation.

Figure 6 summarises the results, while the continuous-time controller parameters

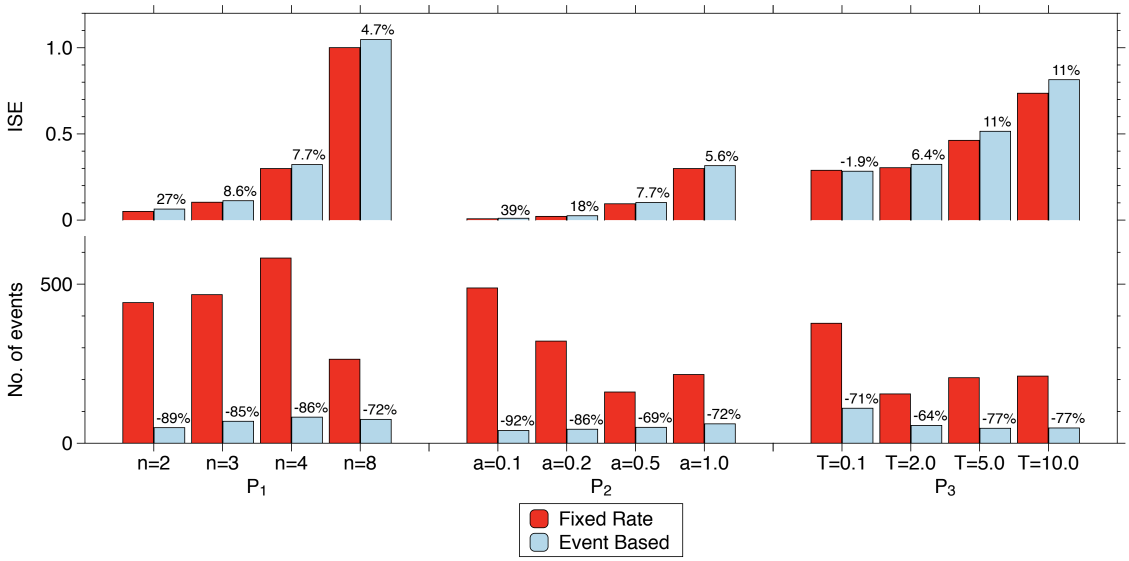

Figure 6: Results of simulation Example 3.

	$P_1(s)$			$P_2(s)$			$P_3(s)$	
n	K	T_i	a	K	T_i	T	K	T_i
2	1.621	1.195	0.1	2.505	0.410	0.1	0.681	0.910
3	1.168	2.086	0.2	1.798	0.622	2.0	1.118	3.488
4	1.031	2.921	0.5	1.198	1.267	5.0	1.292	7.238
8	0.861	6.032	1.0	1.031	2.921	10.0	1.392	13.347

Table 1: Regulator parameters in simulation Example 3.

are shown in Table 1. The used evaluation indices are the ISE for a unit step load disturbance response, and the number of events counted from the time when the disturbance step is applied till that when the set point is recovered within 1%. As can be noticed, the event-based controller realisation behaves in a comparable manner with respect to the fixed-rate one (where the sampling time was chosen again as $\lambda/5$), while the transmission saving is significant.

8. An experimental test

An event-based autotuning PID was realised in LabVIEW based on the method presented here, and tested experimentally on a laboratory plant. The used apparatus is a PT326 process trainer, produced by Feedback, and consists of a duct where the airflow induced by a fan is heated by a resistance. The apparatus is depicted in Figure 7.

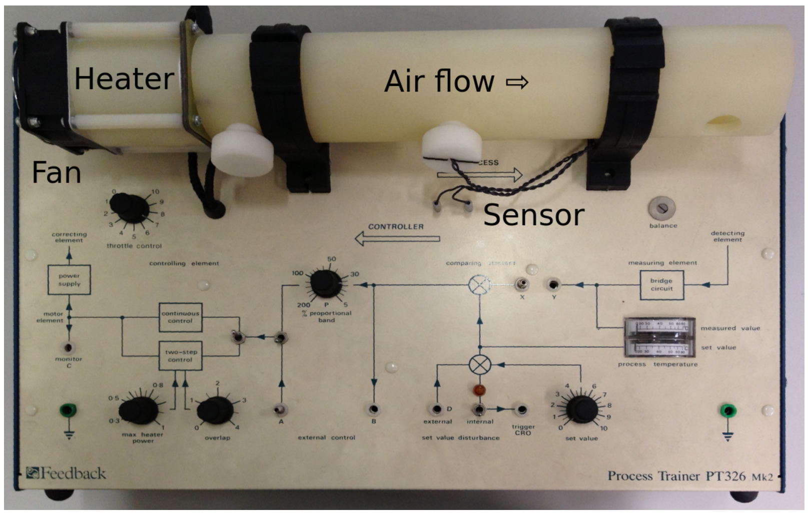

Figure 7: The PT326 apparatus.

The control objective is to regulate the outlet temperature by acting on the heater, while the fan speed can be varied to introduce a disturbance.

Figure 8 shows a test where first the PID is tuned by means of the contextual method already mentioned, having as experiment a relay plus integrator one, then two set point steps are applied, and finally the fan speed is varied twice (increased and then set back to the initial value). Signals are directly expressed in V as measured and actuated, in a (0,10) range. During the relay test the controller was switched to fixed rate, to avoid complicating the test with an event-based experiment, that could introduce artefacts, and is outside the scope of this work.

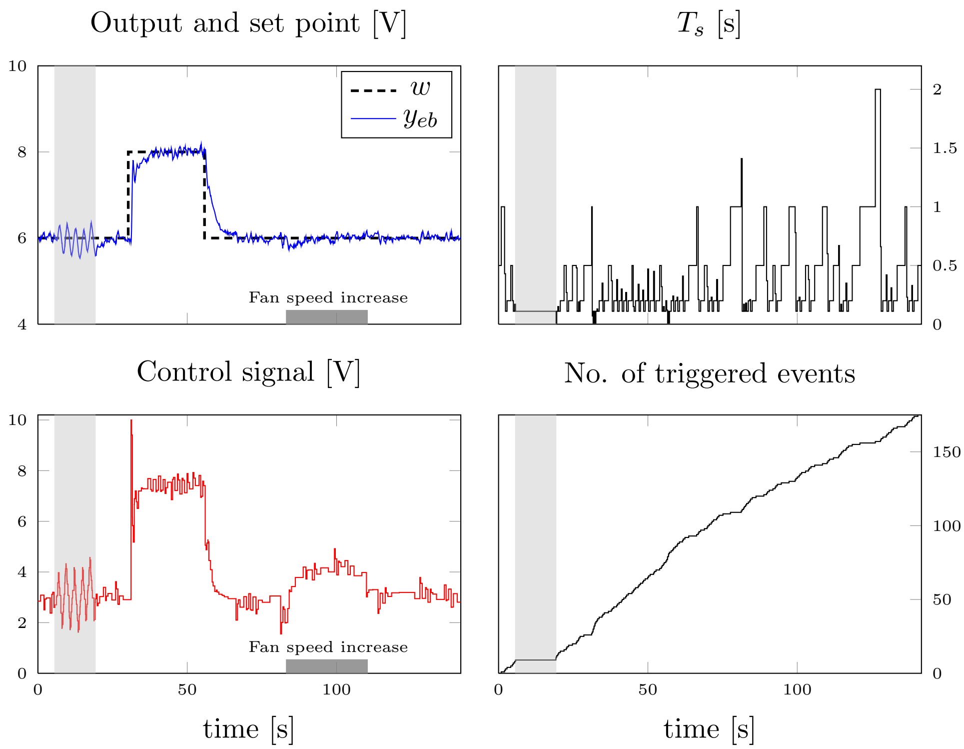

Figure 8: Experimental results with the PT326 apparatus; the vertical grey band indicates the tuning phase; events are not counted during the (fixed-rate) relay test.

The left column of Figure 8 shows the obtained transients, with the tuning phase marked by a grey bar. The right column conversely reports the behaviour of the inter-event period, which is definitely satisfactory, and the accumulated number of events. Although comparisons are never easy with experiments, one can observe that the tuned controller operates with approximately an average of 1.25 events per second. To achieve the same event frequency with a fixed-rate realisation a sampling time of 0.8s would then be required, which is definitely high for the time scale of the involved dynamics.

9. Preliminaries on robustness and performance

This work is centred on stability analysis, that is, nominal conditions. Nonetheless, the usefulness of the proposed ideas would be significantly diminished if a way of dealing with stability robustness and performance could not be at least envisioned. Dealing with robustness – and also with performance, and the tradeoff of the two – requires however a detailed and long treatise. The matter is thus necessarily deferred to future works, and only some words on it are spent in this section to sketch out a possible *modus operandi*.

Beginning with stability robustness, it is evident that any process/model mismatch that preserves the monotonically decreasing nature of the closed-loop system's state free motion is tolerable. More precisely, if the process has dynamics not described by the model, asymptotic stability under arbitrary switching is still ensured provided that said dynamics make the free motion of the closed-loop system still decreasing in norm over a continuous time horizon of length q_s. This is far from easy to ensure, however, and worth noticing just to give a methodologically grounded justification for the intuitive idea that process/model mismatches result in the impossibility of reacting to a sensor-caught event with arbitrary promptness. The ideas above are at present conjectures, but Section 7.3, where processes structurally differ from tuning models, indicates that such conjectures are reasonable.

It is also interesting to observe that with the proposed triggering rule, the sequence of *a posteriori* step durations yielding the least average inter-event time is obtained by indefinitely repeating the couple $(q_s, \widetilde{\sigma}_1 q_s)$. As such, the least possible average inter-event time is $(1 + \widetilde{\sigma}_1)q_s/2$. If some overbound on the process/model mismatch is obtained from input/output data, for example as proposed in [12], and this is used to determine a minimum required dwell time for the closed-loop switching system, then $\widetilde{\sigma}_1$ can be used as a tuning parameter for robustness, specific to the event-based controller realisation. This may conflict with selecting $\widetilde{\sigma}_1$ based on a required promptness, as done above, and to effectively handle such a tradeoff a probabilistic approach may be in order, but at least a potentially viable way to address the issue of stability robustness has been envisaged.

Coming to performance, we can observe that apart from the keep-alive timeout, N too is most frequently limited by stability requirements. This is true also in nominal conditions, by the way, as evidenced even in the fixed-rate context by the numerous sampling time selection criteria based on the induced stability degree reduction. Although stability may be preserved also for "high" (constant) sampling periods, it is well known that when the mentioned upper limit for N is approached, performance degrades. There are well known criteria also to study this issue, and porting them into the event-based context should allow to use N as a tuning knob for performance. Again, this is just a sketch of future research, but here too it seems that a possible *modus operandi* can be defined.

10. Conclusions and future work

The problem of tuning event-based industrial controllers was here addressed. Taking an essentially application-oriented attitude, a functional solution was proposed that handles both the controller discretisation and the event triggering rule. By suitably constraining the former, a sufficient stability condition was derived, and thanks to the consequent freedom in selecting the event triggering mechanism, one was devised to exploit the event-based realisation in a view to minimising sensor transmissions. The proposed technique allows to use classical continuous-time tuning rules (for the moment of the model-based type, but extensions will be addressed) in the case of an event-based realisation.

Simulation examples prove the correctness of the idea, that was also tested on a physical equipment with satisfactory results. This matter, together with possible relaxations of the stability condition, provides some clues for future methodological research. Also, the establishments of tighter relationships with neighbouring research lines, like for example that concerning the study of possible limit cycles, will be an objective. In addition, further experimenting is envisioned, as is the extension of the idea to other types of event-based control structures.

Finally, when addressing the matter of this research, we have to notice that one could have taken basically two approaches. The first is the "rule-abstracted" one used herein. The second is to specify a triggering rule and to analyse the impact of the event-based realisation on its result. No doubt the latter could lead to a lower conservatism, but at the same time the former is inherently keen to accommodate for diverse event generation mechanisms, and as proven by the examples, results in any case in an acceptable control performance. Nevertheless, following the alternative route with respect to this work is another interesting subject for future research.

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

[2] K. Åström. Event based control. In A. Astolfi and L. Marconi, editors, *Analysis and Design of Nonlinear Control Systems*, pages 127–147. Springer Berlin Heidelberg, 2008.

[3] K. Åström and T. Hägglund. Benchmark systems for PID control. In *IFAC Workshop on Digital Control – Past, present, and future of PID Control*, Terrassa, Spain, 2000.

[4] M. Beschi, A. Visioli, S. Dormido, and J. Sánchez. On the presence of equilibrium points in PI control systems with send-on-delta sampling. In *Proc. 50th IEEE Conference on Decision and Control and European Control Conference CDC-ECC 2011*, pages 7843–7848, Orlando, Florida USA, 2011.

[5] F. Cellier and E. Kofman. *Continuous system simulation*. Springer, 2006.

[6] A. Cervin and K. Åström. On limit cycles in event-based control systems. In *Proc. 46th IEEE Conference on Decision and Control*, pages 3190–3195, New Orleans, Louisiana USA, 2007.

[7] J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *SIGCOMM Comput. Commun. Rev.*, 33(1):149–154, Jan. 2003.

[8] C. E. Garcia and M. Morari. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

[9] W. Heemels, K. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3270–3285, 2012.

[10] D. Lehmann and J. Lunze. Extension and experimental evaluation of an event-based state-feedback approach. *Control Engineering Practice*, 19(2):101–112, 2011.

[11] M. Lemmon, T. Chantem, X. Hu, and M. Zyskowski. On self-triggered full-information h-infinity controllers. *Hybrid Systems: computation and control*, pages 371–384, 2007.

[12] A. Leva and A. Colombo. Estimating model mismatch overbounds for the robust autotuning of industrial regulators. *Automatica*, 36(12):1855–1861, 2000.

[13] A. Leva, S. Negro, and A. V. Papadopoulos. PI/PID autotuning with contextual model parametrisation. *Journal of Process Control*, 20(4):452–463, Apr. 2010.

[14] A. Leva and F. Terraneo. Low power synchronisation in wireless sensor networks via simple feedback controllers: the FLOPSYNC scheme. In *Proc. American Control Conference 2013*, Washington, DC, 2013 (to appear).

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

[16] L. Mottola and G. P. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.*, 43(3):19:1–19:51, Apr. 2011.

[17] A. O'Dwyer. *Handbook of PI and PID controller tuning rules.* World Scientific Publishing, Singapore, 2003.

[18] M. Rabi and K. H. Johansson. Event-triggered strategies for industrial control over wireless networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet*, WICON '08, pages 34:1–34:7, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] C. Raghavendra, K. Sivalingam, and T. Znati. *Wireless sensor networks.* Springer, 2006.

[20] D. E. Rivera, M. Morari, and S. Skogestad. Internal model control: Pid controller design. *Industrial & Engineering Chemistry Process Design and Development*, 25(1):252–265, 1986.

[21] J. Sánchez, A. Visioli, and S. Dormido. An event-based PI controller based on feedback and feedforward actions. In *Proc. 35th Annual IEEE Conference on Industrial Electronics IECON '09*, pages 1462–1467, Porto, Portugal, 2009.

[22] J. Sánchez, A. Visioli, and S. Dormido. A two-degree-of-freedom PI controller based on events. *Journal of Process Control*, 21(4):639–651, 2009.

[23] Y. Tipsuwan and M. Chow. Control methodologies in networked control systems. *Control Engineering Practice*, 11(10):1099–1111, 2003.

[24] V. Vasyutynskyy and K. Kabitzsch. Event-based control: overview and generic model. In *Proc. 8th IEEE International Workshop on Factory Communication Systems WFCS 2010*, pages 271–279, Nancy, France, 2010.

[25] V. Vasyutynskyy, A. Luntovskyy, and K. Kabitzsch. Limit cycles in PI control loops with absolute deadband sampling. In *Microwave Telecommunication Technology, 2008. CriMiCo 2008. 2008 18th International Crimean Conference*, pages 362–363, 2008.

[26] G. Zhou, Y. Wu, T. Yan, T. He, C. Huang, J. A. Stankovic, and T. F. Abdelzaher. A multifrequency mac specially designed for wireless sensor network applications. *ACM Trans. Embed. Comput. Syst.*, 9(4):39:1–39:41, Apr. 2010.