

An Event-Based PID Controller With Low Computational Cost

Sylvain Durand, Nicolas Marchand

▶ **To cite this version:**

Sylvain Durand, Nicolas Marchand. An Event-Based PID Controller With Low Computational Cost. SampTA 2009 - 8th International Conference on Sampling Theory and Applications, May 2009, Marseille, France. Special session on Sampling and Industrial Applications. hal-00393031

HAL Id: hal-00393031
https://hal.science/hal-00393031

Submitted on 18 Feb 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Event-Based PID Controller With Low Computational Cost

Sylvain Durand and Nicolas Marchand

NeCS Project-Team, INRIA - GIPSA-lab - CNRS, Grenoble, France.
sylvain.durand@inrialpes.fr, nicolas.marchand@gipsa-lab.inpg.fr

Abstract:

In this paper, some improvements of event-based PID controllers are proposed. These controllers, contrary to a time-triggered one which calculates the control signal at each sampling time, calculate the new control signal only when the measurement signal *sufficiently* changes. The contribution of this paper is a low computational cost scheme thanks to a minimum sampling interval condition. Moreover, we propose to reduce much more the error margin during the steady state intervals by adding some extra samples just after transients. A cruise control mechanism is used for simulations and a noticeable reduction of the mean control computation cost is finally achieved with similar closed-loop performances to the conventional time-triggered ones.

1. Introduction

The classical so-called discrete time framework of controlled systems consists in sampling the system uniformly in the time with some constant sampling period h_{nom} and in computing and updating the control law every time instants $t = kh_{nom}$. This field, denoted time-triggered (or synchronous in sense that all the signal measurements are synchronous), has been widely investigated [6] even in the case of sampling jitter or measure loss that can be seen as some asynchronicity. However, some works addressed more recently event-based sampling where the sampling intervals are event-triggered (also called asynchronous), as for example when the output crosses a certain level. Thus the term *sampling period* denotes a time interval between two consecutive level crossings and the sampling periods are hence not equidistant in time anymore.

Event-triggered notion is taking more and more importance in the signal processing community with now various publications on this subject (see for instance [1] and the references therein). In the control community, very few works have been done. In [3], it is proved that such an approach reduces the number of sampling instants for the same final performance. In [8], it is shown that controlling an asynchronous sampled system or a continuous time system with quantized measurements and a constant control law over sampling periods are equivalent problems.

Many reasons are motivating event-based systems and in particular because more and more asynchronous systems or systems with asynchronous needs are encountered. Actually, the demand of low power electronic components in all embedded applications encourages companies to develop asynchronous versions of the existing time-triggered components, where a significant power consumption reduction can be achieved by decreasing the samplings and consequently the CPU utilization: about four times less power than its synchronous counterpart for the 80C51 microcontroller of Philips Semiconductors in [12]. Note that the sensors and the actuators based on level crossing events also exist, rendering a complete asynchronous control loop now possible. But the most important contributions come from the real-time control community. Indeed, real-time synchronous control tasks are often considered as hard tasks in term of time synchronization, requiring strong real time constraints. Efforts are so carried on the co-design between the controller and the task scheduler in order to soften these constraints. The adopted approach is often either to change dynamically the sampling period related to the load [10, 11] or to use event-driven control where the events are generated with a mix of level crossings and a maximal sampling period [9, 2].

This maximal sampling period seems to be added for stability reasons in order to fulfill the condition of Nyquist-Shannon sampling theorem: a new control signal is performed when the time elapsed since the last sample exceeds a certain limit. We first proposed in [7] to remove it because, thanks to the level detection, the Nyquist-Shannon sampling condition is no more consistent. The CPU cost is hence considerably reduced without performance loss. We now focus on the improvement of event-based control by reducing even more the computational cost with a controller based on a fully asynchronous level detection. The next two sections recall the conventional time-triggered structure and the existing event-based algorithms. The main contribution is developed in section 4 where an event-driven controller with low computational cost is detailed. All controllers are finally compared (in terms of performances and CPU needs) in section 5.

Notations:
e^- will denote the value of e at the last sampling time.

2. Time-Based Control

The textbook PID controller is given as follows:

$$U(s) = K\left(E(s) + \frac{1}{T_i s}E(s) + T_d s E(s)\right)$$

This equation can be divided into a proportional, an integral and a derivative parts, i.e. U_p, U_i and U_d respectively, which are then modified to improve performances [4]. First, set point weighting is applied on U_p and U_d for a more flexible structure, giving the PID two dimensions of freedom. Moreover, a low-pass filter is added in the derivative term to avoid problems with high frequency measurement noise.

$$\begin{aligned} U_p(s) &= K\left(\beta Y_{sp}(s) - Y(s)\right) \\ U_i(s) &= \frac{K}{T_i s} E(s) \\ U_d(s) &= \frac{K T_d s}{1 + T_d s/N}\left(\gamma Y_{sp}(s) - Y(s)\right) \end{aligned}$$

A discrete time controller is finally obtained: the proportional part is straightforward and the backward difference approximation is used for integral and derivative parts.

3. Event-Based Control

The basic setup of an event-based PID controller, introduced in [2], consists of two parts: *a time-triggered event detector* used for level crossings and *an event-triggered PID controller* which calculates the control signal. The first part runs with the sampling period h_{nom} (that is the same as for the corresponding conventional time-triggered PID) whereas the second part runs with the sampling interval h_{act} which depends on the requests sent by the event detector when a new control signal has to be calculated. This is required either when the relative error between the measured signal and the desired one crosses a certain level, i.e. $abs(e - e^-) > e_{lim}$, or if the maximal sampling period is achieved, i.e. $h_{act} \geq h_{max}$.

We proposed in [7] to remove this maximal sampling period underlying a primordial fact in asynchronous control that is that the Nyquist-Shannon sampling condition is no more consistent thanks to the level detection. However, the integral part, i.e. $u_i = u_i^- + K/T_i \cdot h_{act} \cdot e$, leads to important overshoots after the steady states because the sampling period h_{act} becomes huge due to the absence of event. In fact, this time interval between *the last sample before the steady state* and *the first sample of the transient* can be divided into a "real" steady state interval which is equal to $h_{act} - h_{nom}$, plus the detection time period h_{nom}. During the first part the error is very small (lower than e_{lim} else the steady state is not achieved) and so is the product he (lower than $(h_{act} - h_{nom})\, e_{lim}$). As regards the second part, when the set point changes the error becomes large but only during the event detection and therefore the product is $h_{nom}e$. From this observation, several control algorithms were proposed in [7] and we will use the hybrid one which gives good performances with the minimum of samplings.
The hybrid algorithm is a mix between **i)** a controller with a saturation of he which is bounded in $(h_{act} - h_{nom}) \cdot e_{lim} + h_{nom} \cdot e$ when $h_{act} \geq h_{max}$ and **ii)** a controller with an exponential forgetting factor of h_{act} to decrease its impact after a long steady state interval, with $h_{act}^i = h_{act} \cdot \exp\left(h_{nom} - h_{act}\right)$ corresponding to the new sampling period used in the integral part. This mix leads to bound the exponential forgetting factor:

$$\begin{aligned} &\text{if } h_{act} \geq h_{max} \\ &\quad he = \left(h_{act}^i - h_{nom}\right) \cdot e_{lim} + h_{nom} \cdot e \\ &\text{else} \\ &\quad he = h_{act} \cdot e \\ &\text{end} \\ &u_i = u_i^- + K/T_i \cdot he \end{aligned} \tag{1}$$

A first improvement could be obtained by changing the level crossing detection because only one level is really required. Indeed, the control signal needs to be calculated when the measurement is too far from the set point, i.e. as soon as $abs(e) > e_{lim}$. Of course, with this method the number of samples increases during the transients but, at least, the error between the system and the set point is now sure to be lower than e_{lim} during the steady state intervals, which was not the case before with the level detection of the relative error $abs(e - e^-) > e_{lim}$.
A second improvement could be done on the time-triggered event detector which is currently a discrete time system: an event could only be detected at the time instants $t = kh_{nom}$ thereof several levels could miss if they appear between two sampling instants. We propose to use a continuous time event detector which is in fact closer to the real case because a sensor based on level crossing events will send a request as soon as a level is crossed.

Afterwards, the hybrid controller with these improvements is called the asynchronous event-based controller.

4. Event-Based Control with Low Computational Cost

The asynchronous event-based controller is interesting but the number of samples is still important during transients. Indeed, a new request is sent as soon as the error is upper than the detection limit, i.e. $abs(e) > e_{lim}$, which means (quasi)-continuously during the whole transient. To avoid that, we propose to add a minimum sampling interval condition to lighten the transients in order that a new control signal is performed only if a certain time was elapsed since the last sample, i.e. $h_{act} > h_{min}$. This minimum sampling interval could be chosen as the discrete sampling period h_{nom} corresponding to the conventional time-triggered controller or not, but it does have to satisfy the Nyquist-Shannon sampling condition. The choice $h_{min} = h_{nom}$ leads to a discrete-time event detector when the dynamics is important and to a continuous-time event detector when the dynamics is slow (quasi-steady state). Thus, when an event occurs after a steady state configuration, a new control signal is instantaneously computed.

Whatever that may be the h_{min} value, an important reduction of the computational cost is achieved. Nevertheless, we propose to improve the event-based scheme again by adding a few number of samples more. The idea here is to decrease much more the error during the steady state intervals. Currently, one could assure that the error is lower than the limit e_{lim} but cannot know how much lower. Moreover, one could not know if the measured signal is going closer or moving away from the set point.

Therefore, we propose to add some extra samples after a transient while an event-based controller would do not do anything because the condition $abs(e) > e_{lim}$ is wrong. Thus, an extra event is sent to the controller if nothing appends after the last time a control signal was calculated plus a certain sampling interval h_{extra}. Then, this is repeated while the error is upper than a desired minimum level e_{min}. One only needs to define his desired error margin and some extra samples will be added to achieve that. Note that the lower e_{min} is chosen the higher the number of extra samples will be.

5. Simulation Results: Application to a Cruise Control Mechanism

Event-based controller is a good solution, more especially for all the systems which do not need to be constantly controlled. We chose to illustrate our proposals with the cruise control mechanism depicted in [5] because the desired speed of the car is constant most of the time and a new control signal is so only required when the set point changes or when the load (i.e. the slope of the road) varies.

The equation of motion of the car (ν is the velocity) is:

$$m\dot{\nu} = F - F_d$$

The force F is generated by the engine, whose torque is proportional to a control signal $0 \leq u \leq 1$ that controls the throttle position and depends on engine velocity too.

$$F = \alpha_n u T_m \left(1 - \beta \left(\frac{\alpha_n \nu}{\omega_m} - 1\right)^2\right)$$

where α_n depends on the gear ratio n.
The disturbance force F_d has three major components due to the gravity F_g, to the rolling friction F_r and to the aerodynamic drag F_a.

$$\begin{aligned} F_d &= F_g + F_r + F_a \\ \text{with} \quad F_g &= mgsin(\theta) \\ F_r &= mgC_r sgn(\nu) \\ F_a &= \tfrac{1}{2}\rho C_d A \nu^2 \end{aligned}$$

where θ is the slope of the road, i.e. the disturbance.

As regards the control law, an anti-windup mechanism is added to consider the saturation of the control signal u. Thus the integral part consists on the integral of the error plus a *reset* based on the saturation of the actuator (in order to prevent windup when the actuator is saturated).

$$u_i = u_i^- + \frac{K}{T_i} x - \frac{h_{act}}{T_a}(u - u_{sat})$$

where $x = h_{act} \cdot e$ for the time-triggered controller and $x = he$ defined by (1) for the event-based controllers.
Parameter values are $K = 0.8$, $T_i = 1.4$ and $T_a = 0.7$. The nominal and maximal sampling intervals used for the hybrid algorithm are $h_{nom} = 0.1s$ and $h_{max} = 0.5s$ and those used for the low computational cost and the extra samples ones are $h_{min} = 0.1s$ and $h_{extra} = 0.5s$. The detection levels are $e_{lim} = 0.1$ and $e_{min} = 0.01$ for crossing events and for extra samples respectively.

The simulations run during $50s$ with the following test bench: at time 0 the set point is set to $25m/s$ $(90km/h)$, then at time $2s$ it is changed to $30.6m/s$ $(110km/h)$ and changed again to $36.1m/s$ $(130km/h)$ at time $30s$. The gear ratio is chosen accordingly to the speed range, i.e. $n = 5$, and no disturbance is applied, i.e. $\theta = 0$.

The first simulation results are shown on Figure 1 where the conventional time-triggered PI controller is compared to the asynchronous event-based one (see section 3). The top plot shows the set point and both measured signals, the bottom plot shows the sampling intervals (i.e. this signal changes each time the controller calculates a new control signal). The asynchronous event-based controller permits to obtain a system response as quick as the time-triggered one, by calculating a control signal about four time less only (with this benchmark). However, the number of samples remains important during the transients. Our proposal, i.e. the event-based PI controller with a low computational cost, avoids that because the number of samples is dropped by a ratio of 30, as shown on Figure 2.

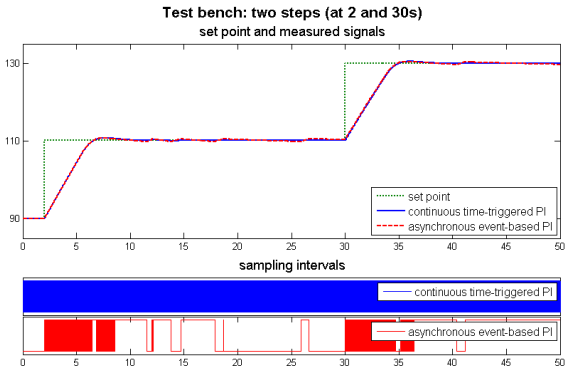


Figure 1: *A conventional time-triggered PI controller* (15000 sampling intervals) vs. *the asynchronous event-based one* (3703 sampling intervals, that is 24.7%).

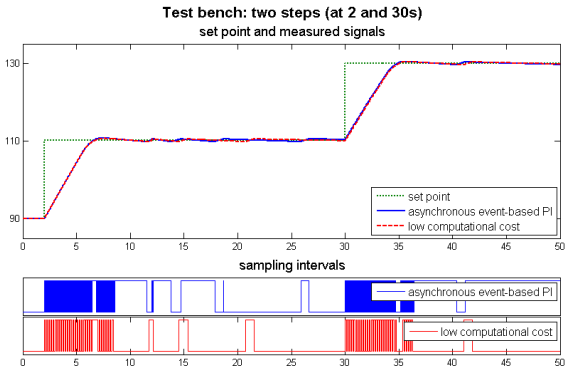


Figure 2: *The asynchronous event-based PI controller* (3703 sampling intervals) vs. *the one with a low computational cost* (126 sampling intervals, that is 3.4%).

Whatever the achieved gain with the low computational cost controller, we propose to improve the error during the steady state intervals by adding some samples just after the transients. Results are shown on Figure 3 where one could see that, by adding extra samples, the sampling number is

finally reduced and the steady state intervals are not oscillating anymore. These are thanks to a measurement signal closer to the set point during the steady state intervals.

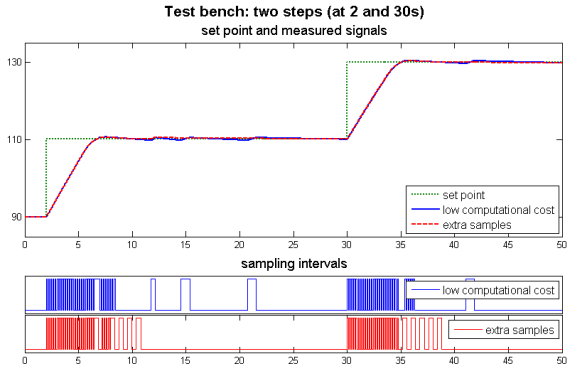


Figure 3: *The asynchronous event-based PI controller with a low computational cost* (126 sampling intervals) vs. *the one with extra samples* (120 sampling intervals).

Finally the integral of the norm of the error are compared for the whole controllers to verify if the responses are not too far from the conventional time-triggered one. All measurements on Figure 4 have a similar behavior with some differences during the steady state intervals because of the allowed error margin e_{lim}. The final values are 74.67 for the reference, 78.2 for the asynchronous event-based controller, 78.63 for the low computational cost one and 77.12 for the extra samples one. Moreover, as regards the last one, it is possible to be much more closer to the time-based value by reducing the minimum value e_{min}.

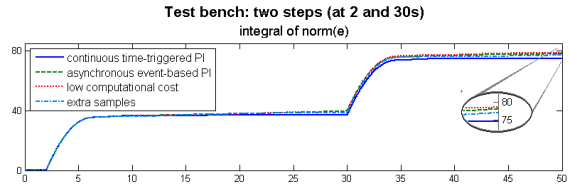


Figure 4: Integral of the norm of the error.

6. Conclusions and Future Works

In this paper we propose to improve the event-based PID controllers depicted in [2] and [7]. The first improvement consists on a minimum sampling interval condition used to decrease the number of samples during the transients. The second one comes from the wishing to reduce much more the error margin during the steady state intervals. Based on these ideas, event-based PID controllers with low computational cost and with extra samples are proposed.
A cruise control mechanism is used to compare them (in simulation) with the conventional time-triggered and with the classical event-based controllers. Both proposals clearly give good performances with a minimum of sampling intervals and the controller with extra samples permits to reduce the error margin as low as desired to achieve a response very closed to the conventional one.

Next steps in this research is naturally to test these controllers in practice and develop other event-based methods for more general types of control.

7. Acknowledgments

This research has been supported by the NeCS Project-Team (INRIA, GIPSA-lab, CNRS) in the FeedNetBack project context, which aims to close the control loop over wireless networks by applying a co-design framework that allows the integration of communication, control, computation and energy management aspects in a holistic way.

References:

[1] F. Aeschlimann, E. Allier, L. Fesquet, and M. Renaudin. Asynchronous FIR filters: towards a new digital processing chain. In *Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems*, pages 198–206, 2004.

[2] K-E Årzén. A simple event-based PID controller. In *Preprints of the 14th World Congress of IFAC*, 1999.

[3] K.J. Åström and B. Bernhardsson. Comparison of Riemann and Lebesque sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[4] K.J. Åström and T. Hägglund. *PID controllers: theory, design, and tuning, 2nd Edition*. The Instrumentation, Systems, and Automation Society, 1995.

[5] K.J. Åström and R.M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[6] K.J. Åström and B. Wittenmark. *Computer Controlled Systems, 3rd Edition*. Prentice Hall, 1997.

[7] S. Durand and N. Marchand. Further results on event-based PID controller. In *Proceedings of the European Control Conference*, 2009.

[8] N. Marchand. Stabilization of Lebesgue sampled systems with bounded controls: the chain of integrators case. In *Proceedings of the 17th IFAC World Congress*, 2008.

[9] J.H. Sandee, W. Heemels, and P.P.J. van den Bosch. Event-driven control as an opportunity in the multidisciplinary development of embedded controllers. In *Proceedings of American Control Conference*, pages 1776–1781, 2005.

[10] O. Sename, D. Simon, and D. Robert. Feedback scheduling for real-time control of systems with communication delays. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, volume 2, 2003.

[11] D. Simon, D. Robert, and O. Sename. Robust control/scheduling co-design: application to robot control. In *Proceedings of the IEEE Symposium on Real-Time and Embedded Technology and Applications*, pages 118–127, 2005.

[12] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor, and G. Stegmann. An asynchronous low-power 80C51 microcontroller. In *Proceedings of the 4th International Sympsonium on Advanced Research in Asynchronous Circuits and Systems*, pages 96–107, 1998.