© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

Improved Event-Based Pi Controller for Limit Cycles Avoidance

Noraide Md Yusop[1], Rosbi Mamat[1*]

[1]School of Electrical Engineering, Faculty of Engineering,
Universiti Teknologi Malaysia, 81310 Skudai, Johor, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2021.13.04.006
Received 7 October 2020; Accepted 22 December 2020; Available online 30 April 2021

Abstract: One of the issues in designing event-based proportional-integral (PI) controller using aggressive tuning rules is the possible occurrence of limit cycles. To date, there is no adequate simple event-based PI controller technique able to explicitly use aggressive tuning rule. In this paper an improved simple event-based PI controller is proposed to address this issue. By analysing the discrete PI algorithm, an improved triggering condition is introduced. To test the effectiveness of the approach, extensive simulations are carried out by introducing the proposed method to process control under various sampling period, triggering limit, and different tuning rules namely, AMIGO, SIMC and One-Third tuning rules. The performances are evaluated based on two standard criteria: ability to imitate the time-triggered system and computation load reduction. The results show that the performance of the proposed method able to surpass others simple event-based PI controller approaches by giving a closest response to the time-triggered system, lowest computational load and able to avoid the limit cycles occurrences. It is envisaged that the proposed method can be useful in designing a simple event-based PI controller that compatible with any type of tuning rules.

Keywords: Event-based, PI control, limit cycles, tuning constraints

1. Introduction

The application of proportional-integral (PI) controller in low-power microcontroller is getting attentions from the control community because of their capability to provide a satisfactory performance for many processes with a relatively easy design [1]. Low-power microcontroller gives an advantage in term of their small size, low cost, and simple programming. Discrete control system is based on the sampling time, where the system output is sampled and control signal is updated periodically [2]. This time-triggered strategy unfortunately is not efficient in energy usage because it will remain consuming energy and computation resources, by updating the control signal even the system has reached a steady state.

An event-based PI controller can be an alternative to have more resources-aware system. In the event-based strategy, the controller only computes control signals changes when certain conditions are satisfied [3-5]. Such strategy gives an advantage on energy saving in the computation load of all the digital electronics involved in the control loop. In context of real-time system, event-based technique can reduce the CPU utilization which will make the processor to be available to do more tasks.

The first event-based control was introduced by Årzén [6]. The event-based PI controller was designed with two logical conditions which were absolute relative error and time-triggered safety maximal sampling period condition, to reduce the control update while maintaining good output performance. This technique is also known as level crossing sampling [7], send-on-delta (SOD) sampling [8], or deadband sampling [9]. In [10] Durant and Marchand managed to reduce more control update by removing the safety maximal period and improved the integral discretization algorithm. On parallel work, Beschi et. al [11] extended the event-based SOD sampling strategy to symmetry send-on-delta

*Corresponding author: rosbi@utm.my

(SSOD) where the error signal was quantized first before computed the control signal. However, SSOD technique introduced new computation for error quantization and required new tuning rule, consequently, increasing the computational complexity [12-14]. The latest and simplest algorithm can be found in [15] where simple modification was made on integral discretization algorithm to produce event-based response close to time-triggered response.

Although even-based PI controller offers a good strategy in saving the computational resources, this strategy has a tendency to produce a limit cycles response, especially when the control parameters are taken directly from continuous tuning rules [14,16-17]. Limit cycles is an oscillatory response at the steady-state region which can create problems such as quick wear out of actuators. Existing simple event-based PI controller [6,10,15] cannot be used arbitrarily with the aggressive tuning rule to have the good response with free limit cycles. Therefore, the aim of this work is to introduce new simple event-based algorithm that can avoid the limit cycles even by using control parameters from the continuous tuning rule.

The paper is structured as follows. Section 2 explains the event-based PI controller setup. Section 3 outlines how the proposed method is developed. In section 4 the simulation setup and comparison results of proposed method with other simple event-based PI controllers on various event-based control parameters and tuning rules are presented. Finally, in Section 5, the conclusion is provided.

2. Event-based PI Controller

Generally, feedback control system has three main components: 1) controller; 2) process/system/plant with actuators; 3) sensors. Once the controller is a type of digital controller, an additional component of analog-to-digital converter and digital-to-analog converter are compulsory. Basically, event-based PI controller is a digital controller which receives and computes a control signal based on the presence of event, instead of triggering periodically. Fig. 1 shows the setup for event-based PI controller.

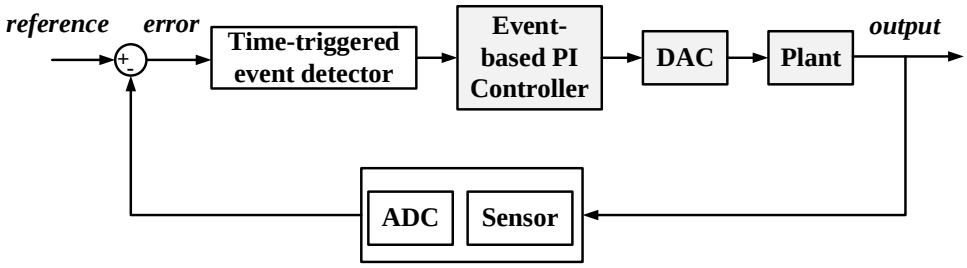


Fig. 1 – Block diagram of event-based PI controller

Event-based PI controller uses an event detector to decide control signal update. The event detector checks the event periodically which is equal to the sampling period (h_{nom}) of the sensor. Event detector has a logical rule to determine the event. In the Årzén's setup [6] the event triggering condition is given by

$$|e(k) - e(j)| > \bar{e} \text{ or } h_{act} \geq h_{max} \tag{1}$$

where the event is detected based on two rules: a relative measurement between current error $e(k)$ and the previous used error $e(j)$ by controller crosses a predefined limit level $\bar{e}$; and an interval time (h_{act}) without a control update crosses maximal sampling period limit h_{max}. This setup resulted in the controller will be updating control signal at h_{nom} during transient (i.e., set point changes and load disturbances) and reducing controller execution to h_{max} when the system reaches steady state.

3. Proposed Method

The PI controller in Laplace domain is formulated as

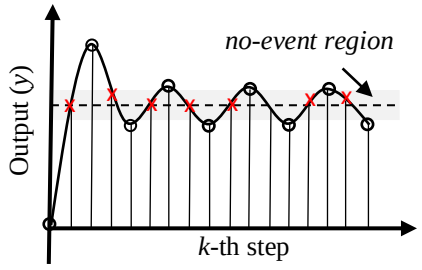
$$U(s) = K.E(s) + K/T_i s.E(s) \tag{2}$$

Where, U and E are control and error signals respectively, K and T_i are proportional and integral terms of PI controller. The discrete PI control with backward difference algorithm is calculated as

$$u_p(k) = K(e(k)) \tag{3}$$
$$u_i(k) = u_i(k\text{-}1) + (K/T_i).\ h(k).\ e(k) \tag{4}$$
$$u(k) = u_p(k) + u_i(k) \tag{5}$$

Where, $u(k)$, $u_p(k)$ and $u_i(k)$ are control signal, proportional control and integral control at k-th step respectively, while $e(k)$ is error at k-th step and h is an interval time between step. In event-based technique the interval time is sporadic and every event-based approach has different $h(k)$ used in their PI algorithm. Consequently, due to the aperiodic control update in event-based PI controller, forward approximation discretization as suggested in [2] for time-triggered controller is not accurate for event-based controller because the value of $h(k+1)$ cannot be predetermined [10].

Another significant effect of event-based controller is that the system will experience two states of condition; 1) event triggered during transient state; 2) no-event during steady state. If the event is triggered, which indicate the system is in transient state, the system will act as a closed-loop system, while it will be an open-loop system if there is no event, i.e., steady state. Note that, during the transition from the event state (closed-loop system) to no-event state (open-loop system), the last control signal which is generated by closed-loop system will be inherited. This is crucial in open-loop system, due to that control signal will determine the direction of the output response. As open-loop system is a condition at the steady state, the control signal should drive the output response within the boundary limit. If the response crosses the limit, the system will return to close-loop system which will drive back to open-loop system and this bouncing condition will result in oscillatory response which is called limit cycles.



o – event triggered
x – no event

Fig. 2 – Step response with limit cycles due to event-based control

Fig. 2 shows the example of the limit cycles occurrences due to the event-based control. Circle mark indicate the event is triggered and the control is updated while cross mark shows no control update due to it is in no-event region. Noted after the response reaches near the set point, the response keeps moving to the limit of event-condition resulting in new event and control update. This phenomenon repeated and caused the limit cycles happened. As explained before, this limit cycles can be avoided if the proper control signal is achieved before the response enter the no-event region.

Typically, event conditions that are used for simple event-based PI controller to determine the event, are absolute error level crossing $|e(k)| > \bar{e}$ and absolute relative error level crossing $|\Delta| > \bar{e}$, where Δ is a value of the difference between the current error and the error where the last event is triggered ($e(k)$-$e(j)$), and $\bar{e}$ is the triggering level limit. In velocity form PI control signal can be calculated as

$$u(k) = u(k\text{-}1) + K.\ (e(k)\text{-}e(k\text{-}1)) + (K/T_i).\ h(k).\ e(k) \tag{6}$$

In event-based PI controller, the control signal $u(k\text{-}1)$ and error $e(k\text{-}1)$ are the last control signal and error where the event is triggered written as $u(j)$ and $e(j)$. This is due to the control signal is updated based on the event rather on the sequence of k-th step. The resulting control signal can be expressed as

$$u(k) = u(j) + K.\ (e(k)\text{-}e(j)) + (K/T_i).\ h(k).\ e(k) \tag{7}$$

By substituting $\Delta = e(k)\text{-}e(j)$, equation (7) is given by

$$u(k) = u(j) + K.\ \Delta + (K/T_i).\ h(k).\ e(k) \tag{8}$$

Undertake $h(k)$, K, T_i as constant, hence, $u(k)$ can be manipulated using the values of $u(j)$, Δ, and $e(k)$. It is important to remark that $h(k)$, K, T_i are assumed as a constant to indicate these parameters are explicitly taken from continuous tuning rule. Fig. 3 depicts an example of step response for the plant output (Fig. 3(a)), with its error trajectory (Fig. 3(b)) and control signal in Fig. 3(c). As shown in Fig. 3(c), first control signal $u(1)$, reduces the error from e_1 to e_2 with

Δ_1 amount. The relation between control signal and error show that control signal plays a significant role in the direction of error and the value of relative error Δ. Thus, by examining the current relative error, a behavior of last control signal can be identified. This information is important to ensure the open-loop system (no-event state) receives an appropriate control signal. Next section will analyse the effect of conventional triggering condition to the open-loop control signal for response in Fig. 3, and propose the improved triggering condition.

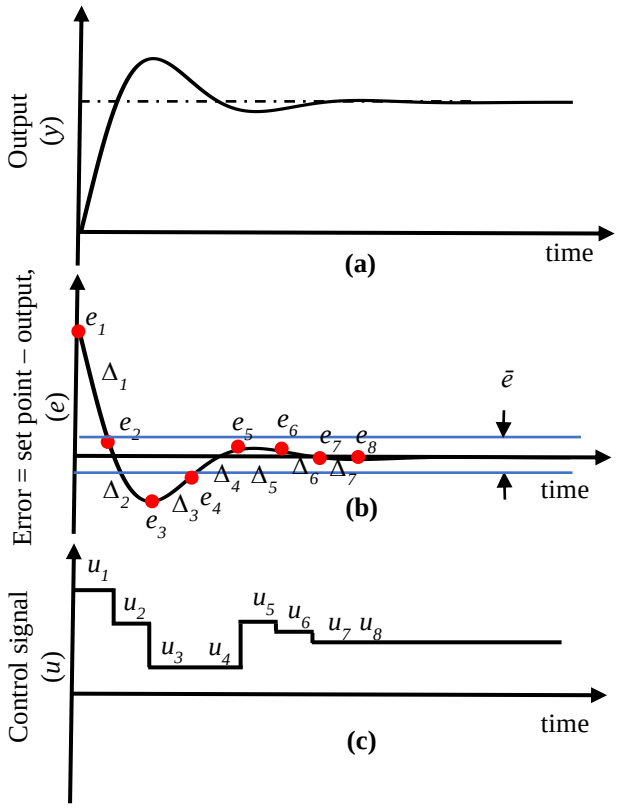


Fig. 3 - Output, error and control responses for step response

3.1 Absolute Error Level-crossing Triggering

Level crossing absolute error event condition will stop the control update when current error is equal or lower than the triggering-limit $\bar{e}$. Fig. 3(b) shows that first error that within the $\bar{e}$ boundary limit is e_2, however, the value of Δ_1 is huge and as logical explanation this is because the control signal $u(1)$ is in aggressive mode. Therefore, if control update starts to stop at e_2, it will generate limit cycles where control signal will drive the output response toward the limit $\bar{e}$ before reverse the direction toward another boundary limit. Fig. 4 illustrates the limit cycles response caused by absolute error triggering condition. This conclude that absolute error triggering condition cannot give an appropriate control signal when the system changes from closed-loop to open-loop system. Despite the oscillation, the output response is closed to the set point**.**

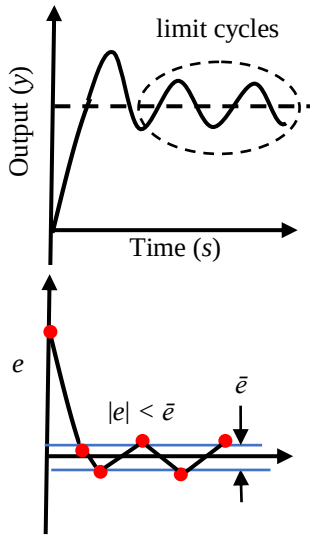


Fig. 4 - Output and error for event-based PI controller with absolute error level-crossing triggering

3.2 Absolute Relative Error Level-crossing Triggering

Event condition using absolute relative error most probably will give a control signal that contributes to the steady error to the system (small Δ). However, the steady error is not guarantying a small error, in this case sticking problem will be generated. Sticking problem is a phenomenon where the system stops updating the control updates even the response is far from reference point. As illustrated in Fig. 3(b), the earliest low relative error can be found at point e_4, by assuming Δ_3 value is lower than $\bar{e}$. At this point the system will start running using open-loop system and as shown in Fig. 5, the system is experienced a sticking problem when level-crossing absolute relative error is applied as event condition. The output responses were able to maintain its position but with the cost of huge steady state error.

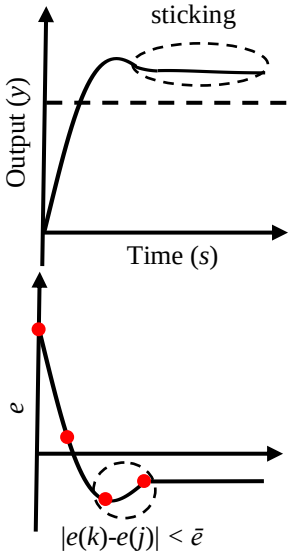


Fig. 5 - Output and error for event-based PI controller with absolute relative error level-crossing triggering

3.3 Proposed Improvement for Event Triggering Condition

Analysis of the conventional event triggering conditions in Section 3.1 and 3.2 indicates that, the absolute error rule can guaranty lower output response. On the other hand, absolute relative rule gives smooth output response with a steady error. Based on this evidence, a combination both triggering conditions is proposed as an improved triggering condition which is define as

$$|e(k)| > \bar{e} \text{ or } |e(k) - e(j)| > \delta \quad (9)$$

where δ is triggering limit for relative error. The first condition will force the system continuously update when the error is outside the $\pm \bar{e}$ region and the later will make sure the rate of error is small before the system stop updating. As illustrated in Fig. 3, assume error and relative error at point e_5 are within the boundary limit $\bar{e}$ and δ (assuming $\delta = \bar{e}$) respectively. Despite a small steady state error as depicted in Figure 5, the output response is smooth and close to the set point. Based on Fig. 4, 5 and 6, the improved triggering condition is believed can deliver a good output response for event-based PI controller because the improved triggering condition start to stop the control update closest to the equilibrium point which is at e_5 as shown in Fig. 3 and Fig. 5 whereas the absolute error and absolute relative error triggering conditions are stopped update at e_2 at e_4 respectively. It is suggested the δ is chosen at least 10 times smaller than $\bar{e}$ to guarantee the system has reach the settling region before stop updating.

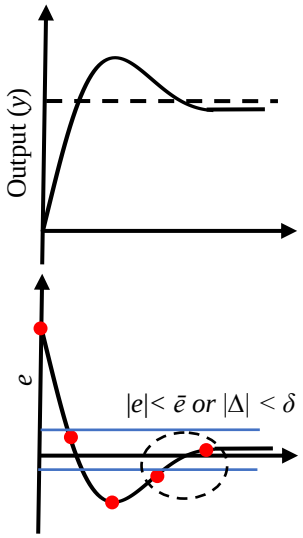


Fig. 6 - Output and error responses for event-based PI with improved triggering condition

In measuring the output response, it is hard for the sensor to avoid a measurement noise. This measurement noise can eventually generate an event if its value reaches the threshold triggering limit $\bar{e}$. Therefore, the selection of triggering limit $\bar{e}$ is important to avoid/reduce the event triggered by measurement noise. This condition can be remedied by choosing triggering limit $\bar{e}$ greater than measurement noise. The triggering limit is typically tuned according to a trade-off between the number of events per time unit and the control performance [16].

4. Simulation Examples

In this section, performance of the proposed method (PM) will be compared with conventional time-triggered PI controller (TT) and two simple event-based PI controllers, the Durand and Marchand saturation (DMS) method [10], and Yusop and Mamat fixed period algorithm (FPA) [15]. The DMS and FPA methods were selected due to both methods have a less algorithm in their event-based PI computation load [15]. Table 1 shows the comparison of the integral discretization algorithm and triggering condition for all three methods. As described in Table 1, both DMS and FPA used similar triggering condition but different $h(k)$ for integral discretization algorithm in (4). In DMS and FPA, second Årzén triggering condition rule in (1) was removed which led to huge integral impact when there was a large interval time without control update h_{act}. Thus, DMS used saturation h_e as $h(k)$, while FPA used h_{nom} to ease the integral impact. PM algorithm was based on FPA integral discretization algorithm but the triggering condition used a combination of absolute error and absolute relative error (as proposed in Section 3.3). Although there is an additional subtraction and logic check at the triggering condition algorithm in PM, this addition does not give huge complexity to the controller computation. The coding for DMS, FPA and PM are illustrated in Fig. 7, 8 and 9 respectively.

Table 1 - Comparison of event-based PI controller methods

Method	Integral discretization Algorithm	Triggering condition
DMS [10]	$u_i(k) = u_i(j) + (K/T_i).\ h_e$ if $h_{act} > h_{max}$ $h_e = (h_{act} - h_{nom}).(\bar{e}) + h_{nom}.\ (e(k))$ else $h_e = h_{act}$	$\lvert e(k)\text{-}e(j)\rvert > \bar{e}$
FPA [15]	$u_i(k) = u_i(j) + (K/T_i).\ h_{nom}.\ e(k)$	$\lvert e(k)\rvert > \bar{e}$
PM	$u_i(k) = u_i(j) + (K/T_i).\ h_{nom}.\ e(k)$	$\lvert e(k)\rvert > \bar{e}$ or $\lvert (e(k)\text{-}\ e(j)\rvert > \delta$

```
% inputs
ysp = u(1);
y = u(2);
e = ysp - y;

% calculate control signal
hact = hact + hnom;
if abs(e - e_old)- > elim

   if hact >= hmax
      he=(hact-hnom)*elim + K/Ti*hnom *e;
   else
      he = hact *e;
   end
       up = K*e
       ui = ui_old + K/Ti*he;
        u = up + ui
% update
hact = hnom;
e_old = e;
ui_old = u;

end
```

Fig. 7 - Codes for DMS event-based PI controller

```
% inputs
ysp = u(1);
y = u(2);
e = ysp - y;

% calculate control signal

if abs(e) > elim
       up = K*e
       ui = ui_old +  K/Ti*hnom*e;
        u = up + ui

% update
ui_old = u;

end
```

Fig. 8 - Codes for FPA event-based PI controller

```
% inputs
ysp = u(1);
y = u(2);
e = ysp - y;

% calculate control signal

if (abs(e) > elim OR abs(e - e_old) > δ )
       up = K*e
       ui = ui_old +  K/Ti*hnom*e;
        u = up + ui

% update
ui_old = u;

end
```

Fig. 9 - Codes for PM event-based PI controller

A good event-based PI controller should be adaptable for any type of tuning rules. For this purpose, three tuning rules namely, AMIGO tuning rule [19], SIMC tuning rule [20], and One-third tuning rule [18] were tested on all three event-based PI controllers for three simulation setups. These three tuning rules are selected due to their difference behavior of tuning, where AMIGO gives moderate tuning gain, SIMC is aggressive tuning, and One-third is the slowest tuning among them.

4.1 Simulation Setup

The simulation was conducted on two processes/plants. The process control can be a good benchmark, where PI controllers are widely employed. The plants were approximated to first order plus deadtime (FOPDT) as

$$P(s) = \frac{K_p}{(1+sT)} e^{-sL} \tag{10}$$

where K_p is the static gain, T the lag or time constant, and L the time delay. In process control the behaviour of the plants are categorised as lag-dominant, balanced and delay-dominant. This dominant behaviour is an important guideline in selecting an appropriate tuning rule [18]. The three categories are determined by the normalized time delay $\tau = L / (L+T)$. Commonly, the limits between the three categories are $0 \leq \tau \leq 0.2$ for lag dominated, $0.2 < \tau < 0.6$ for balanced, and $0.6 \leq \tau \leq 1.0$ for delay-dominated processes [14, 18]. In this work similar plants used in [14] specifically lag-dominant and delay-dominant plants were studied, given as

$$P_1(s) = \frac{1}{(1+s)(1+0.1s)(1+0.01s)(1+0.001s)} \tag{11}$$

and

$$P_2(s) = \frac{1}{(s+1)^4} e^{-5s} \tag{12}$$

where $K_p = 1$, $L = 0.075$ and $T = 1.04$ and $K_p = 1$, $L = 6.3$ and $T = 2.92$ for processes $P_1(s)$ and $P_2(s)$ respectively. $P_1(s)$ is considered as lag-dominant process due to lag T is much longer than time delay L, while $P_2(s)$ is delay-dominant process since time delay L is longer.

Simulation 1

The first simulation used process $P_1(s)$ with the control parameters listed in Table 2. The control parameters were calculated from the three tuning rules based on FOPDT parameters.

Table 2 - Controller parameters for process (9)

Rule	K	T_i
AMIGO	4.13	0.539
SIMC	6.93	0.60
One-Third	0.33	1.04

The simulation was established as in [14] where the set point was set to 0 to illustrate the system is in equilibrium before load disturbance with magnitude 1 was introduced at time 1*s*. The event-based parameters $\bar{e}$ and δ were chosen equal to 0.1 and 0.01, respectively, and for DMS algorithm, h_{max} was equal to 1. The 0.01 sampling period h_{nom} was used for the event-detector.

Simulation 2.
The second simulation used process $P_2(s)$ with the control parameters listed in Table 3. The set point, load disturbance, event-based parameters and sampling period were used similar as in Simulation 1.

Table 3 - Controller parameters for process (10)

Rule	K	T_i
AMIGO	0.21	3.61
SIMC	0.23	2.92
One-Third	0.33	4.25

Simulation 3
The third simulation used process $P_1(s)$ and parameters similar as Simulation 1, however the step reference change with value 1 was given at time 1*s* and load disturbance with magnitude 1 was introduced at time 30*s* with 60*s* simulation time. The triggering limit and sampling period were tested for several values, i.e. 0.01 and 0.1. Similar performance index in [15] was used in this study; an integrated absolute difference between the time-triggered and event-based system response (IAEP). The IAEP was calculated as follows:

$$\int_0^\infty |y_{time\text{-}triggered}(t) - y_{event\text{-}based}(t)|\, dt \tag{11}$$

Where small IAEP means the response is close to the time-triggered response and large IAEP implies poor response due to the response is far from the time-triggered response. Computational load for the response was calculated to measure the reduction of computational cost. The computational load was obtained as follows:

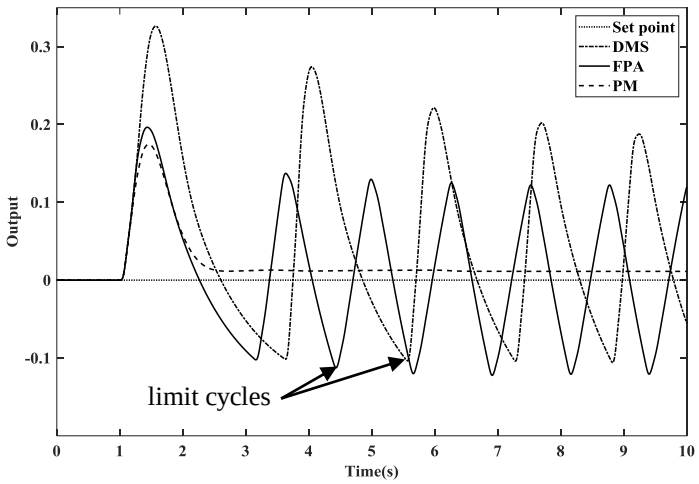
$$\frac{\text{number of updates}}{\text{number of updates for time-triggered}} \text{ x } 100 \tag{12}$$

4.2 Simulation Results

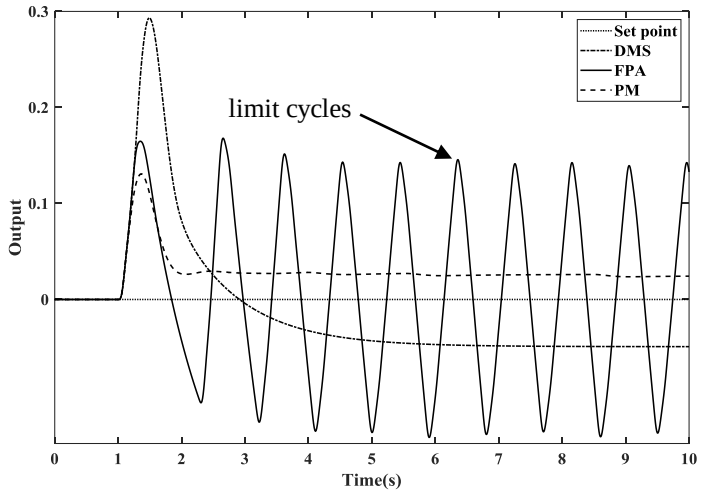
The objective for Simulation 1 and 2 was to verify the effectiveness of PM in limit cycles avoidance, while Simulation 3 was purposely to further investigate the performance and computation load for event-based PI controller using different sampling period h_{nom} and triggering limit $\bar{e}$.

Fig. 10 shows the responses for the Simulation 1 where the process $P_1(s)$ was simulated with three event-based PI control algorithms (DMS, FPA and PM) using tuning rules listed in Table 2. As seen in Fig. 10, PM algorithm managed to avoid limit cycles for all tuning rules. On the contrary, DMS and FPA produced inconsistence results, where limit cycles occurred to DMS response when using AMIGO, while FPA failed to avoid limit cycles for AMIGO and SIMC. This implies that, there is no guarantee to have a smooth response when directly applied a continuous tuning rule on DMS and FPA. In addition, slow tuning rule, i.e., One-Third which suggested lowest controller gains, provided a smooth steady-state response for all three even-based PI controllers. Such tuning offers slow response resulting in small rate of change for the error which is favorable for open-loop PI switching state in event-based controller. Thus, all three event-based PI control algorithms managed to reach open-loop state with appropriate control signal when using One-Third tuning rule.

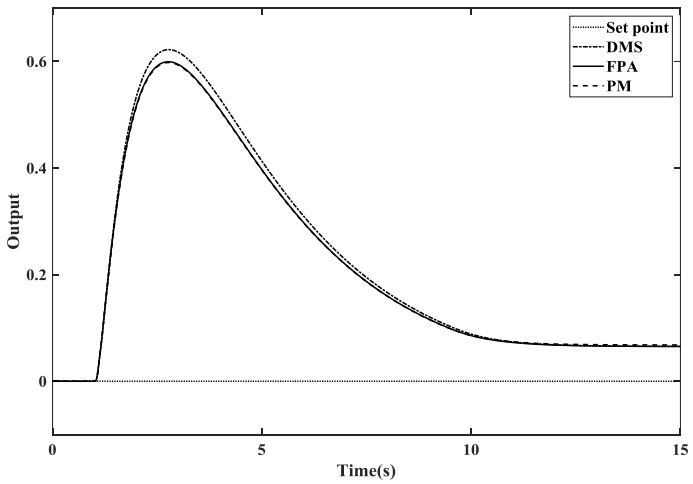
Fig. 11 illustrates the response for Simulation 2 on process $P_2(s)$ with the same parameters' setup as in Simulation 1 and control parameters listed in Table 3. It was found that, all the tested event-based strategies able to produce a response without limit cycles as shown in Fig. 11. This is due to the delay-dominant process produced slow output response which contributed to small error-rate change. The small error-rate change resulting in smooth switch from close-loop system to open-loop system in event-triggering which offer high chance to avoid limit cycles. Another interesting finding from the results is the proposed method PM algorithm managed to maintain a good performance even for slow-response process where PM gave almost identical output response as DMS and FPA for AMIGO (Fig. 11(a) and One-Third (Fig. 11c)) tuning rules, while for SIMC tuning rule, PM gave better output response which closest to the set point as shown in Fig. 11(b).



(a) AMIGO

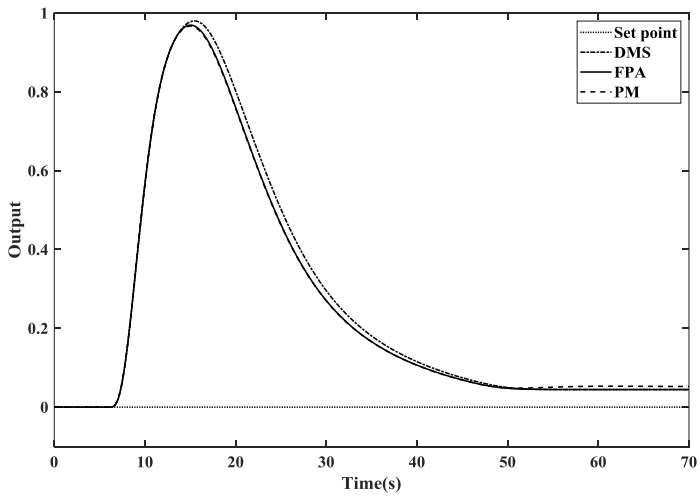


(b) SIMC

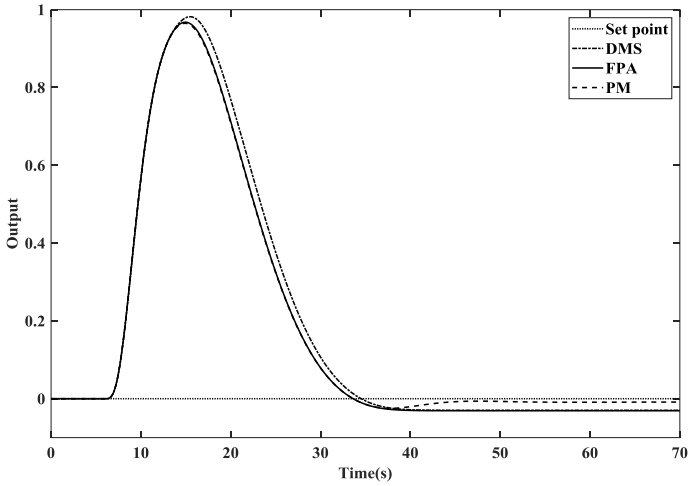


(c) One-Third

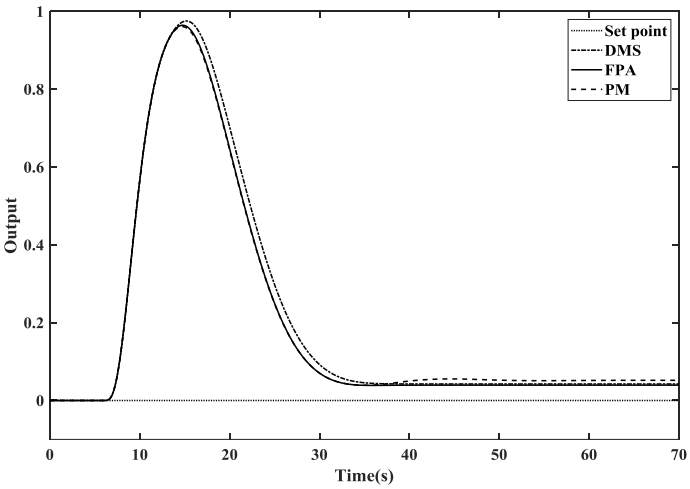
Fig. 10 - Step load disturbances obtained from the lag-dominant process in Simulation 1



(a) AMIGO



(b) SIMC



(c) One-Third

Fig. 11 - Step load disturbances obtained from the delay-dominant process in Simulation 2

Fig. 12 demonstrates one of the responses for the Simulation 3 where the limit cycles are occurred. As shown in Fig. 12, DMS and FPA event-based method generated the limit cycles around the set point. Contrary result was showed for PM algorithm where smooth response without limit cycles was achieved. The rest of Simulation 3 results obtained for different event-based controllers with different parameters are listed in Table 4. As depicted in Table 4, PM algorithm produced a smooth response without limit cycles for all tested parameters, which is the reason for PM algorithm producing lowest IAEP and computational load compared to DMS and FPA with limit cycles responses. This supports the fact that limit cycles will deteriorates the system performance and increases the control updates. It is important to emphasize that DMS and FPA required proper selection of parameters i.e., tuning rule method, sampling period and triggering limit to guarantee good performance without limit cycles. For instance, at lower sampling time and triggering limit i.e., 0.01, AMIGO tuning rule seems to be acceptable employed with FPA, but as the triggering limit was increased, the limit cycles started to appear and constituted poor performance with more control update. In addition, aggressive tuning rule like SIMC is unfeasible with DMS and FPA, where even at lower triggering limit, SIMC still produced limit cycles for DMS and FPA responses.

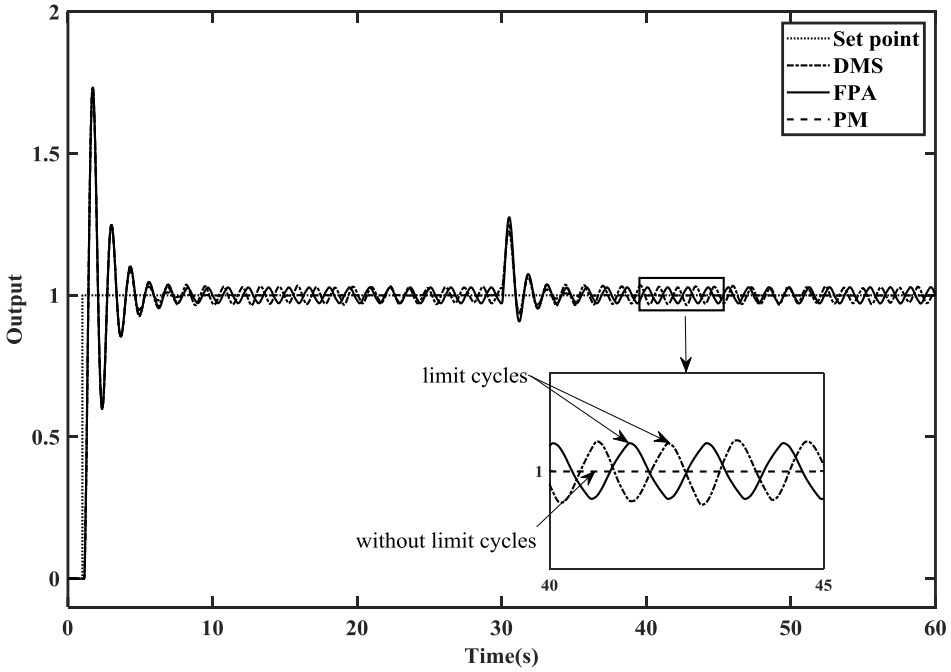


Fig. 12 - Output response for Simulation 3 using AMIGO tuning with triggering limit $\bar{e}$ = 0.01 and sampling period 0.1

Table 4 - Performance index of event -based PI controller for Simulation 3

Tuning rule	Sampling period	Triggering limit $\bar{e}$	Control Scheme	Number of updates	Limit Cycles	Computa -tional load (%)	IAEP
AMIGO	0.01	-	TT	6000	-	100	
		0.01	DMS	1201	Yes	20.02	0.64
			FPA	297	No	4.95	0.32
			PM	313	No	5.22	0.34
		0.1	DM	2044	Yes	34.07	4.47
			FPA	1630	Yes	27.17	3.93
			PM	281	No	4.68	0.54
	0.1	-	TT	600	-	100	
		0.01	DMS	457	Yes	76.17	0.97
			FPA	441	Yes	73.50	0.90
			PM	154	No	25.67	0.02
		0.1	DMS	427	Yes	71.17	10.32
			FPA	425	Yes	70.83	9.52
			PM	129	No	21.50	0.04
SIMC	0.01	-	TT	6000	-	100	
		0.01	DMS	1463	Yes	24.38	0.45
			FPA	1356	Yes	22.60	0.43
			PM	342	No	5.70	0.32
		0.1	DM	1282	Yes	21.37	3.34
			IA	2369	Yes	39.48	4.65
			PM	220	No	3.67	1.09
One-Third	0.01	-	TT	6000	-	100	
		0.01	DMS	2988	No	49.80	0.26
			FPA	2974	No	49.57	0.17
			PM	2983	No	49.72	0.18
		0.1	DM	1681	No	28.02	4.23
			FPA	1507	No	25.12	2.56
			PM	1594	No	26.57	2.55
	0.1	-	TT	600	-	100	
		0.01	DMS	294	No	49.00	0.27
			FPA	292	No	48.67	0.16
			PM	298	No	49.67	0.17
		0.1	DMS	167	No	27.83	4.14
			FPA	148	No	24.67	2.51
			PM	158	No	26.33	0.49

Overall finding suggests that the proposed event condition combination in PM algorithm is able to be directly employed with the continuous tuning rule and avoid limit cycles occurrences. Although with a simple algorithm PM manages to reduce the computational load more than 70% from the time-triggered approach.

5. Conclusion

In this paper, an improved triggering rule of event-based PI controller has been presented and tested using three tuning rules, namely AMIGO, SIMC and One-Third rules against a load disturbance response. By considering the limitations in directly applying continuous tuning rules to event-based PI controller, a good event condition rule has been proposed to significantly avoid the unwanted limit cycles oscillations around the set point. The simulation results and the comparative analysis of the proposed algorithm with its counterparts proved the advantages of the proposed

method in terms of: the ability to imitate time-triggered performance without a limit cycles no matter what tuning rules is used and the reduction in the computational load.

Acknowledgement

The authors would like to thank Universiti Teknologi Malaysia (UTM) and Public Service Department Malaysia (PSD) for their support.

References

[1] M. Nithyasree, and K. V. Kandaswamy, "A Generic PID Controller Based on ARM Processor," *Procedia Engineering*, vol. 38, pp. 1044-1049, 2012
[2] K. J. A˚ström, B. Wittenmark, Computer-controlled Systems, Theory and Design, Prentice-Hall, 1996
[3] A. Visioli. "Research trends for PID controllers,". *Acta Polytechnica*, vol. 52, pp. 144-150, 2012
[4] J. Sánchez, A. Visioli, S. Dormido, "Event-based PID control," in: PID Control in the Third Millennium, Advances in Industrial Control, Springer London, pp. 495–526. 2012
[5] Q. Liu, Z. Wang, X. He, & D. H. Zhou, "A survey of event-based strategies on control and estimation," *Systems Science & Control Engineering*, vol. 2(1), pp. 90–97, 2014
[6] K. -E. Årzén, "A simple event-based PID controller," *Proc. 14th IFAC World Congress*, vol. 18, pp. 423–428, 1999
[7] V. Vasyutynskyy, K. Kabitzsh, "Towards comparison of deadband sampling types," *Proceedings of IEEE International Symposium on Industrial Electronics*, 2007
[8] E. Kofman, J. Braslavsky, "Level crossing sampling in feedback stabilization under data rate constraints," *Proceedings of 45th IEEE International Conference on Decision and Control*, 2006
[9] J. Sánchez, M.A. Guarnes, S. Dormido, A. Visioli, "Comparative study of event-based control strategies: An experimental approach on a simple tank," *Proceedings of European Control Conference*, 2009
[10] S. Durand and N. Marchand, "Further results on event-based PID controller," *Proceeding of the European Control Conference*, pp. 1979–1984, 2009
[11] M. Beschi, S. Dormido, J. Sánchez, A. Visioli, "Characterization of symmetric send-on-delta PI controllers," *Journal of Process Control*, vol. 22, no. 10, pp. 1930–1945, 2012
[12] M. Beschi, S. Dormido, J.Sánchez, A. Visioli, "Tuning of symmetric send-on-delta proportional-integral controllers," *IET Control Theory & Applications*, vol. 8, no. 4, pp. 248-259, 2014
[13] J.A. Romero, R. Sanchís, "A new method for tuning PI controllers with symmetric send-on-delta sampling strategy," *ISA Trans.*, vol. 64, pp. 161–173. 2016
[14] J. Sánchez, M. Guinaldo, S. Dormido, A. Visioli, "Validity of continuous tuning rules in event-based PI controllers using symmetric send-on-delta sampling: an experimental approach," *Computers and Chemical Engineering*, vol. 139, 2020
[15] N. M. Yusop and R. Mamat, "Analysis of Event-Based PI Controller and Some Proposed Improvements," *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, Shah Alam, Selangor, Malaysia, pp. 170-175, 2020
[16] A. Cervin, K. J. Åström, "On limit cycles in event-based control systems," *Proceedings of 46th IEEE International Conference on Decision and Control*, New Orleans, LA, USA, pp. 3190–3195, 2007
[17] V. Vasyutynskyy, A. Luntovskyy, K. Kabitzsch, "Limit cycles in PI control loops with absolute deadband sampling," *Proceedings of 18th IEEE International Conference on Microwave & Telecommunication Technology*, pp. 362–363, 2008
[18] T. Hägglund, "The one-third rule for PI controller tuning," *Computer and Chemical Engineer*, vol. 127, pp. 25–30, 2019
[19] K. J. Aström, T. Hägglund, "Advanced PID Control," *ISA - The Instrumentation, Systems, and Automation Society*, Research Triangle Park NC 27709, 2006
[20] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Journal of Process Control* vol. 13 (4), pp. 291–309, 2003