Development of a Digital Autotuning PI for First Order Plant Using RLS-PZC

Sidik Nurcahyo [1*], Fitri [2], Sungkono [3]
[1, 2, 3] Electrical Engineering, State Polytechnic of Malang, Malang, Indonesia
Email: [1] sidik.nurcahyo@polinema.ac.id, [2] fitri@polinema.ac.id, [3] sungkono@ polinema.ac.id
*Corresponding Author

***Abstract*—The fact that a real plant can be estimated as a first order and its parameters vary due to the environment has motivated this article to discuss the development of a digital autotuning PI for a first-order plant using Recursive Least Square (RLS) and Pole Zero Cancellation (PZC). Although the focus is only on first order, the methods discussed here hopefully become a basis for developing higher-order plants. Firstly, formulas for calculating PI parameters are derived using PZC and tested by simulation to verify their effectiveness. Then it is organized serially with the RLS and digital PI to form an autotuning PI algorithm. The RLS periodically reads plant input-output to estimate plant parameters. These resulting parameters are fed to PZC and finally, PZC outputs are used by digital PI to control the plant. This design is verified by Matlab simulation, where the controller is realized as an m-function containing a program code for RLS, PZC, and digital PI algorithm. The test was conducted by varying plant parameters, including DC gain and time constant. Verifying controller parameters and their response shows that RLS-PZC can effectively re-tune the digital PI parameters, proved by its response having zero steady-state error and its settling time is maintained. The proposed algorithm can also ensure that the PI controller output is always within the specified maximum limits hence the actual response does not deviate from the designed response.**

Keywords—Autotuning PI; Recursive Least Square; Pole-Zero Cancelation; First-Order Plant; Matlab.

I. INTRODUCTION

To facilitate plant analysis and controller design easily, plant transfer function is usually approximated as a low-order transfer function [1],[2]. Even a high-order system can be estimated as first-order. To control a plant, there exists several controller types, include: PID, Fuzzy [3], PSO[4], LQR [5], Sliding Mode [6], PI Sliding Mode [7], MRAC [8],[9], and many more. The PID controller is among the most widely used controllers in applications, like for regulating the speed of conveyor [10], for regulating humidity and temperature [11], for maximum power point tracker (MPPT) of photovoltaic system [12], for stabilizing maneuver of UWV (under water vehicle) [13], for controlling movement of CNC machine [14], for regulating output of SEPIC converter [15] and many more. PID is easy to implement and performs well if the parameters are chosen correctly. PID can be implemented on analog devices [16][17] or on digital devices like on a microcontroller [18], [19], [20], [21] or on an FPGA [22]. Analog type is easy to implement but it is also easily disturbed by noise and its parameter tuning must be done manually unlike digital type which allows the use of certain algorithms for parameter autotuning. The PID controller consists of three elements: proportional (P), integral (I), and derivative (D). These three elements can be combined to form several variants of controllers, such as P, PI, PD, and PID. From a mathematical point of view, the PID controller contains two zeros and one pole, making it suitable for second-order plants. Meanwhile, the PI controller contains only one pole and zero, making it ideal for first-order plants. When PI parameters (Kp, Ki) are tuned properly then pole of controlled first order plant can be canceled by zero of that PI controller, thus pole of the PI controller will take over system response thoroughly. This article will specifically discuss controlling of first-order plant using a PI controller as the plant is simpler and the number of controller parameters are fewer, making it easier to be tuned. However, in many practical applications, more complex controllers may offer better performance despite being more challenging to tune. This research chooses simplicity rather than complexity. The main issue with this controller is determining appropriate values of Kp and Ki so that the system response has zero steady state error, and its settling time equals to a value as specified. Several methods exist for tuning PID parameters, such as Ziegler Nichols [23], Cohen, GA [24], Fuzzy [27], MRAC [25], ANN [26], Firefly [27], and relay shifting[28], [29], [30], [31]. However, this article will discuss the PZC (Pole Zero Cancellation) method for tuning the PI parameters as it is simple, straightforward and more deterministic but it requires plant transfer function to be known [32], [33].

The fact indicates that plant parameters can be influenced by temperature, humidity, age, and other factors. This situation can degrade the control system performance. To address this issue, researchers have introduced autotuning controllers. This type of controller can detect changes of plant parameters [34] by several type of identification algorithm [35], and automatically re-tune the controller parameters hence the control system performance can be maintained [36]. For example, autotuning PID can be developed using the relay switch method [37]. Routh stability criteria is also used to tune integrator gain of PID [38]. Autotuning PID has been implemented for manipulator robots [39]. The short relay test method also has been applied to determine PID parameters [40]. The relay experiment method using extended Kalman filter and second order integrator as estimator was described and elaborated in [41]. Intelligent [42], Genetic algorithm [43], metaheuristic [44], machine learning [45], auto differentiation [46], computational [47] or optimization techniques [48] are also applicable methods for tuning PID parameters. Famous terms regarding to these methods are like PSO-PID [49] and Metaheuristic [50] [51].

MRAC (Model Reference Adaptive Control) is a general structure commonly used for tuning PID parameters where plant output will be forced to follow the dynamic of a chosen model [52]. In general, those autotuning methods are indeterministic and require more computational complexity.

As part of the scientific contributions, this article will discuss the development of a digital autotuning PI controller using the RLS-PZC method specifically for first-order plants. Take note that it is not for autotuning PID but autotuning PI instead, as plant to be controlled is first order. The motivation for choosing RLS-PZC over other methods is that it is simple, deterministic and performance is guaranteed. Hence, it will be applicable, or it can be implemented in applications of the real world. RLS (Recursive Least Square) is used to detect parameter changes in the first-order plant, including the plant dc gain (K) and the plant time constant (τ). Meanwhile, PZC (Pole Zero Cancelation) is assigned to calculate the best possible PI parameters for given constraints. Successfulness of PZC and overall autotuning PI will heavily depend on the estimation accuracy of RLS. As this is preliminary research then to verify the design, it is tested by Matlab simulation as described in [53] that Matlab is one of good tools for control design dan simulation. For complete verification, real world testing is emphasized. The testing is prepared to judge whether the proposed method can maintain the control system performance when plant parameters are changed. The performance to be studied will include settling time and steady state error. If time constant and dc gain of controlled plant are changed and the proposed controller capable to keep response to have specified time constant and zero steady state or its response is fit to ideal response, then it can be concluded that the proposed method is valid. The research contribution includes PZC formulation and autotuning PI based on RLS-PZC for first order plant.

II. METHOD

A. Autotuning Structure

Development of the autotuning PI controller begins with constructing autotuning PI structure as shown in Fig. 1.

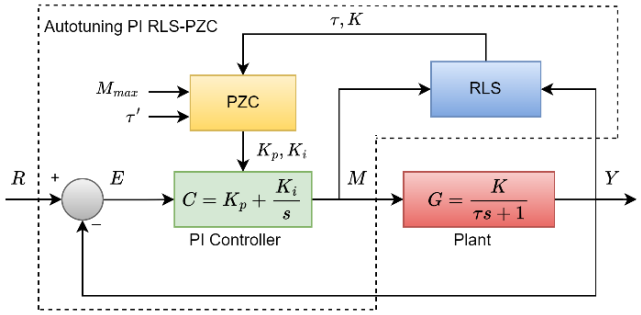

Fig. 1. Closed loop control with an RLS-PZC autotuning PI

This structure is like a regular closed loop control system except that there exist two additional blocks called RLS and PZC. RLS is assigned to estimate plant parameters include plant DC gain (K) and plant time constant (τ) based on the controller output M and plant output Y. PZC receives the estimated plant parameters to calculate PI parameters, including K_p and K_i. While calculating the PI parameters, PZC considers M_{max} and τ', which represent the maximum amplitude of M and the desired settling time, respectively. These parameters must be considered in calculation to ensure that the resulting PI parameters will exhibit response with settling time τ' and PI output or control signal does not exceed the limitation. If this limitation is violated, then the actual response will not fit to the designed response. The PI parameters are then used by controller C to process error signal E, which is the difference between setpoint R and output Y, into signal M required by plant G. This structure is applied to ensure that system performance is maintained even though plant parameters are varied during operation. The system performances include settling time and steady state error.

B. Pole Zero Cancelation (PZC)

PZC is designed to calculate PI parameter (Kp, Ki) so that plant output settles in expected duration with zero steady state error and the PI output does not exceed specified limit to guarantee that actual response will fit to designed response. As its name suggests, PZC will cancel the plant pole using PI zero hence PI pole will dominate entire closed loop response. To derive PZC formula, assume first-order plant to be controlled is as follows,

$$G = K/(\tau s + 1) \tag{1}$$

and the PI controller is as follows,

$$C = M/E = (K_p s + K_i)/s \tag{2}$$

Hence, its closed-loop transfer is stated as follows.

$$G_c = \frac{K\left(K_p s + K_i\right)}{\tau s^2 + s\left(1 + KK_p\right) + KK_i} \tag{3}$$

To make G_c as a first order with gain of one, then zero of G_c must be canceled using a pole. This can be achieved by modifying G_c denominator as follows.

$$\begin{aligned} D(s) &= \left(K_p s + K_i\right)\left(s\tau/K_p + \alpha\right) \\ &= \tau s^2 + s\left(K_p\alpha + \tau K_i/K_p\right) + K_i\alpha \end{aligned} \tag{4}$$

The comparison between (3) and (4) lower results in:

$$K\,K_i = \alpha \mathrm{K_i};\ \alpha = K \tag{5}$$

and

$$\begin{aligned} 1 + KK_p &= K_p\alpha + \tau K_i/K_p \\ K_i &= K_p/\tau \end{aligned} \tag{6}$$

Pole zero cancelation will take place by applying (4) upper as the denominator of (3), yields:

$$G_c = \frac{K\left(K_p s + K_i\right)}{\left(K_p s + K_i\right)\left(s\tau/K_p + \mathrm{K}\right)} = \frac{1}{\frac{\tau}{KK_p}s + 1} \tag{7}$$

This is a first order with dc gain of one and time constant $\tau' = \tau/KK_p$. If τ' is specified, then K_p can be calculated as follows.

$$K_p = \tau/(K\tau') \quad (8)$$

Substitution of (8) into (6) yields:

$$K_i = \frac{K_p}{\tau} = \frac{1}{K\tau'} \quad (9)$$

Kp and Ki formula depends on selected time constant τ'. The smaller time constant τ' will speed up the response, but higher amplitude of control signal is required. To know how small the time constant can be selected regarding the actuator limit, the controller output formula needs to be derived by substituting the following error formula into equation (2).

$$E = u_c\tau'/(\tau's + 1) = u_c/(s + 1/\tau') \quad (10)$$

Then obtained the following controller output.

$$M = \frac{u_c}{s + 1/\tau'}\frac{K_p s + K_i}{s} = u_c\left(\frac{K_i\tau'}{s} + \frac{K_p - K_i\tau'}{s + /\tau'}\right) \quad (11)$$

do Laplace inverse to (11), yields

$$m(t) = u_c\, K_i\tau' + u_c\big(K_p - K_i\tau'\big)e^{-t/\tau'} \quad (12)$$

Maximum $m(t)$ will occur when $e^{-t/\tau'}=1$ or at $t = 0s$, hence:

$$m_{max} = \mathrm{u_c\, K_i\tau' + u_c}\big(K_p - \mathrm{K_i\tau'}\big) = u_cK_p \quad (13)$$

Substitution (8) to (13) yields maximum value of actuator:

$$m_{max} = u_c\tau/(K\tau') \quad (14)$$

If m_{max} is determined based on actual actuator limit, then the applicable minimum time constant will be

$$\tau'_{min} = \frac{u_c\tau}{Km_{max}} \quad (15)$$

PZC algorithm consider τ'_{min} as smallest allowable value for time constant in calculating Kp and Ki using (8) and (9). If user select lower value than (15), then PZC will use τ'_{min}, instead to ensure that the actual response will fit to the designed response. Otherwise, actual response will deviate from designed response or simulation response.

C. Digital PI Controller

To enable PI implementation in a microcontroller, (2) needs to be converted into its z-equation and then into corresponding difference equation. If the s to z conversion is done using Backward Euler as follows.

$$s \approx \frac{1 - z^{-1}}{T} \quad (16)$$

where T is sampling time, then the detail steps for converting PI from Laplace to z-equation is shown in Table I.

TABLE I. LAPLACE TO Z CONVERSION FOR PI

Step	Formula	Description
1	$\frac{U}{E} = K_p + \frac{K_i}{s}$	Laplace form for PI controller
2	$U(z) = K_pE(z) + \frac{K_iE(z)T}{(1 - z^{-1}\}}$	Replacing s with (16) to get z-form of U
3	$U(z) = P(z) + I(z)$	Naming each part with P and I, respectively

Thus, by replacing z with k yields the following PI difference equation.

$$u(k) = p(k) + i(k) \quad (17)$$

where

$$P(z) = K_pE(z);\ \ p(k) = K_pe(k) \quad (18)$$

and detail derivation for I part as shown in Table II.

TABLE II. SIMPLIFICATION OF Z FORM FOR I PART

Step	Formula	Description
1	$I(z) = K_iE(z)\frac{T}{1 - z^{-1}}$	Z form for I part
2	$I(z)(1 - z^{-1}) = K_iE(z)T$	Multiply denominator to left side
3	$I(z) = I(z)z^{-1} + K_iE(z)T$	Get I(z) by moving other term to right side

Hence by converting last line from z to k results in difference equation for I part as follows.

$$i(k) = i(k - 1) + K_iTe(k) \quad (19)$$

To get better response, sampling time T should be selected as small as possible, but it must be equal to or longer than duration required to execute PI and other algorithm.

D. Recursive Least Square (RLS)

RLS is used to estimate parameters of first order plant, include dc gain K and time constant τ, based on plant input $u(k)$ and plant output $y(k)$. This is required to enable PZC calculating PI parameter accurately. Substitution (16) into (1), results in difference equation for the plant as follows.

$$\begin{gathered}\frac{Y(z)}{U(z)} = \frac{K}{\tau(1 - z^{-1})/T + 1} \\ Y(z) = (KTU(z) + \tau Y(z)z^{-1})/(\,\tau + T) \\ y(k) = au(k) - by(k - 1)\end{gathered} \quad (20)$$

where

$$a = KT/(\tau + T);\ K = a(\tau + T)/T \quad (21)$$

and

$$b = \tau/(\tau + T);\ \tau = bT/(1 - b) \quad (22)$$

If $\Phi(k) = [u(k)\ \ y(k - 1)]^T$ and $\hat{\theta} = \big[\hat{a}\ \hat{b}\big]$, where $\hat{a}$ is estimation for a and $\hat{b}$ is estimation for b, then RLS can be developed by relying calculation on these equation (23).

$$\begin{aligned} \hat{y} &= \Phi(\mathrm{k})\theta \\ e &= y - \hat{y} \\ K &= \frac{P\Phi}{\lambda + \Phi^{\mathrm{T}} P \Phi} \\ P &= (P - K\Phi^{T}P)/\lambda \\ \hat{\theta} &= \hat{\theta} + Ke \end{aligned} \tag{23}$$

where λ is adaptation rate which is freely chosen in range between 0 and 1. Insensitive behavior can be achieved by setting λ near to 1. In contrast, to get high sensitivity, λ value should be reduced approaching zero. Meanwhile P is identity matrix where its diagonal elements equal to a big number, e.g. 1000, that determines speed of estimation [7], [8], [9], [10], [11], [12], [13], [14], [15], [16].

Using data samples read from plant input $u(k)$ and plant output $y(k)$, RLS will obtain $\hat{a}$ and $\hat{b}$. Then using (21) and (22), K and τ are found effectively as both P and $\hat{\theta}$ are small size matrix, 2×2 and 1×2, respectively.

E. Autotuning PI

As mentioned earlier, the autotuning PI comprises of RLS, PZC and digital PI. Hence, to construct proposed controller simply by cascading these three processes sequentially, as follows:

- RLS take place firstly by reading $u(k)$ and $y(k)$ to produce K and τ,
- PZC uses K and τ to update K_p and K_i value,
- Digital PI uses obtained K_p, K_i and $e(k) = r(k) - y(k)$ to calculate $m(k)$ required by plant.

This process is repeated once every sampling time T until the system is disabled as shown in Fig. 2.

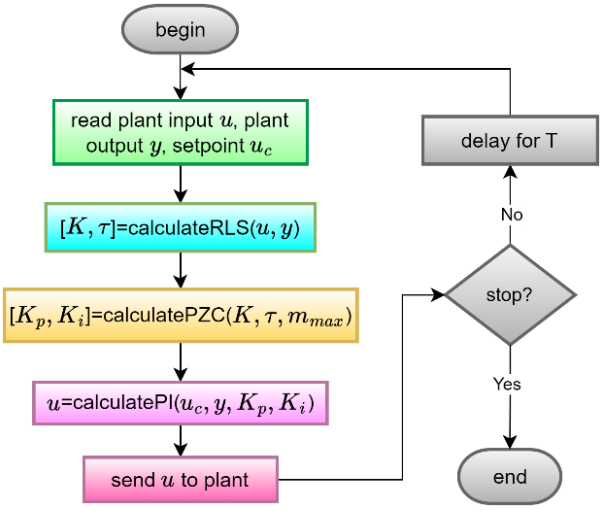

Fig. 2. Flowchart for RLS-PZC autotuning PI

III. RESULTS AND DISCUSSION

This section describes design and verification for PZC, RLS, digital PI, and autotuning PI.

A. Design and Verification for PZC

PZC is responsible for calculating Kp and Ki, using (8) and (9) respectively. These two equations require K, τ and τ'. According to (15), τ' requires u_c and m_{max}, which are the setpoint change and the maximum PI output, respectively. Thus, the PZC script can be constructed as in Listing 1.

Line 1 defines setpoint change. For example, if PI is assigned for controlling temperature where setpoint is 30°C and ambient temperature is about 25°C, then setpoint change will be 5°C. Line 2 and 3 define plant time constant and palnt dc gain, respectively. Line 4 defines maximum actuator, where it must be equal to actuator capability. Line 5 defines the required settling time for closed loop control system. Line 6 defines sampling time for PZC, RLS and PI controller. Sampling time must be equal to or longer than duration required to execute RLS-PZC and PI algorithm. Line 8 calculates (15). Line 9 calculates time constant regarding to settling time defined in line 5. Line 11-16 reassigns required time constant and required settling time with reasonable one hence controller output does not exceed maximum limit of actuator. Line 18 and 19 calculate Ki and Kp using (9) and (8), respectively. Line 20 defines plant transfer function G. Line 21 defines PI transfer function. Line 22 defines forward transfer function. Line 23 defines closed loop transfer function and Line 24 display step response information, like settling time, overshoot percentage, etc.

Listing 1. PZC script to calculate Kp and Ki.

```
1  uc = 10;                % setpoint change
2  tau = 1.2;              % plant time constant
3  K = 0.9;                % plant dc gain
4  actuator_max = 50;      % m_max in (15)
5  ts= 1.0;                % required settling time
6  T = 0.01;               % sampling time
7                          % Eq. (15)
8  tau_aksen_min = uc*tau/(K*actuator_max);
9  tau_aksen=ts/4;         % ts=4*tau
10 % ensure tau_aksen >= tau_aksen_min
11 if tau_aksen<tau_aksen_min
12     tau_aksen = tau_aksen_min;
13     ts_aktual = tau_aksen*4;
14 else
15     ts_aktual=ts;
16 end
17 ts_aktual                % display actual ts
18 Ki = 1/(K*tau_aksen)    % Eq. (9)
19 Kp = tau*Ki             % Eg. (8)
20 G=tf(K,[tau 1]);        % plant transfer fun
21 C=tf([Kp Ki],[1 0]);    % PI transfer fun
22 M=uc*feedback(C,G);     % forward path
23 Gc=feedback(C*G,1);     % closed loop
24 stepinfo(Gc)            % step response
```

Executing this scrip result in Kp=5 and Ki=1.1667. To verify this result, these values need to be entered into the following PI block and then run a simulation shown in Fig. 3.

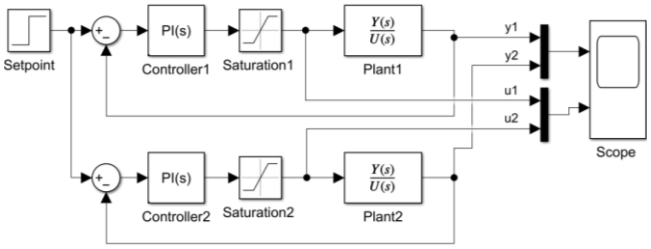

Fig. 3. Simulink for testing of PZC

Signal $y1$ is output for PI closed loop control system with saturation equal to the value set in Listing 1 line 4. While signal $y2$ is output of similar system but saturation is set to

half of previous system. Running this simulation results in response as depicted in Fig. 4.

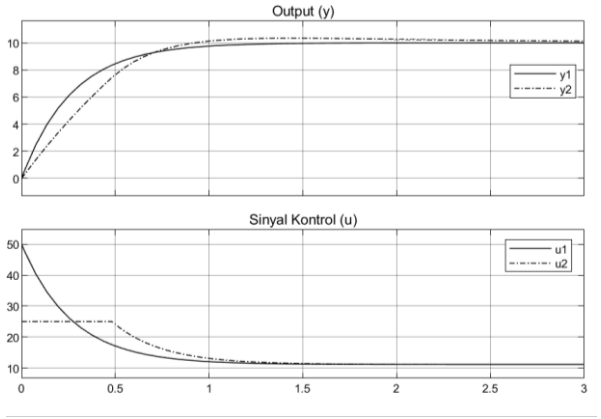

Fig. 4. Simulink for testing of PZC

y1 settles for about one second as defined by Listing 1 line 5. In contrast, *y2* requires more time for settling as Saturation2 is set to 25 which is lower than the value defined in Listing 1 line 4, in this case, 50. Control signal *u1* is also as expected. Its maximum value is equal to the value defined in Listing 1 line 4. In comparison, the control signal *u2* is saturated at 25.

B. Design and Verification for Digital PI

Digital PI described by (17), (18) and (19) can be implemented into a script as shown in Listing 2.

Listing 2. Script for digital PI.

```
1  function u = pi(r, Kp, Ki, T, y)
2
3  persistent I;  % static variable
4
5  if isempty(I)
6      I=0;       % init static variable
7  End
8
9  e = r - y;      % calculate error
10 I = I + Ki*T*e; % calculate integral
11 u = Kp*e + I;   % calculate P+I
```

Line 3 defines static variable I to keep integrator value and line 5 initiates the variable with zero. Line 9 calculates error, i.e. difference between setpoint *r* and output *y*. Line 10 does integration operation and line 11 calculates PI as in (17). This PI script is verified by attaching it into Controller2 and Controller3 of a simulation diagram shown in Fig. 5.

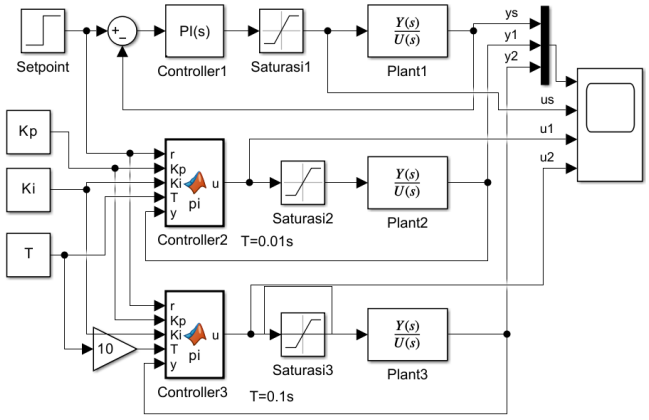

Fig. 5. Simulink for testing PZC

This diagram accompanied with three PI controllers, i.e. Controller1, Controller2, and Controller3. Controller1 is an analog type. It becomes a standard for two other digital types. Controller3 works ten times slower than Controller1. All these controllers use the same Kp and Ki value, result of Listing 1. Running this simulation yields responses as shown in Fig. 6.

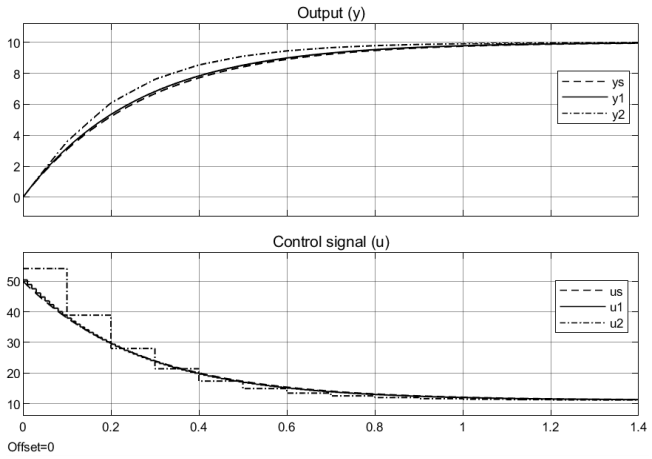

Fig. 6. Digital PI responses

As can be seen, $y1$ (response with *T=0.01s*) fits to the analog response ys, meanwhile $y2$ (response when *T=0.1s*) deviates from the analog response *ys*. This indicates that smaller sampling time makes the digital response closer to the analog response, but too small sampling time surely will burden controller work. This result also verifies that digital PI has been successfully developed since the digital response fits to the analog response (ys).

C. Design and Verification for RLS

RLS described by (21), (22) and (23) is for estimating K and τ. These equations can be implemented into a script as depicted in Listing 3.

Listing 3. RLS script to estimate K and τ.

```
1  function [tau,K] = rls(u,y,T)
2
3  persistent y1;          % define static vars
4  persistent P;           % covariance matrix
5  persistent theta_est;   % estunated theta
6
7  delta = 1000;
8  lambda = 0.95;
9
10 if isempty(y1)
11     y1 = 0;             % init static vars
12     P = delta * eye(2);
13     theta_est = zeros(2, 1);
14 end
15                         % Eq. (23)
16 phi = [u; y1];
17 e = y - phi' * theta_est;
18 K_k = (P * phi) / (lambda + phi' * P * phi);
19 theta_est = theta_est + K_k * e;
20 P = (P - K_k * phi' * P) / lambda;
21 y1 = y;
22
23 a = theta_est(1,1);    % pick a
24 b = theta_est(2,1);    % pick b
25 tau = b*T/(1-b);       % Eq. (22)
26 K=a*(tau+T)/T;         % Eq. (21)
```

To verify its functionality, this script needs to be embedded into Matlab function block named RLS as shown in Fig. 7 and its time sampling is set to T via block parameter.

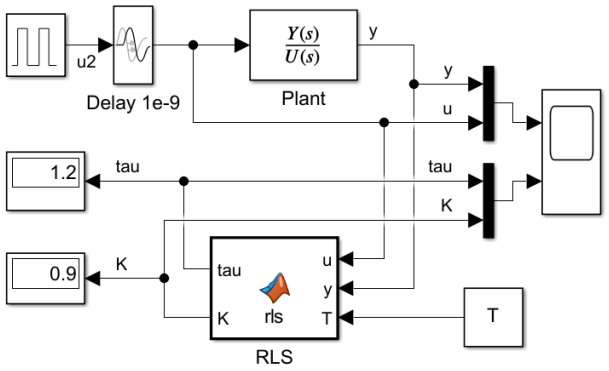

Fig. 7. Simulink for testing RLS

In this figure, Plant is a plant to be estimated by RLS; thus, its input u and output y is routed to RLS. RLS also receives T that is a sampling time, which defines an interval for reading those two signals and is required to calculate tau and K as shown in line 25-26 Listing 3. If plant dc gain and plant time constant are as defined in Listing 1 line 2-3, i.e. $K=0.9$, $\tau=1.2s$, and the sampling time is as defined in Listing 1 line 6, then running the simulation will result in response as shown in Fig. 8.

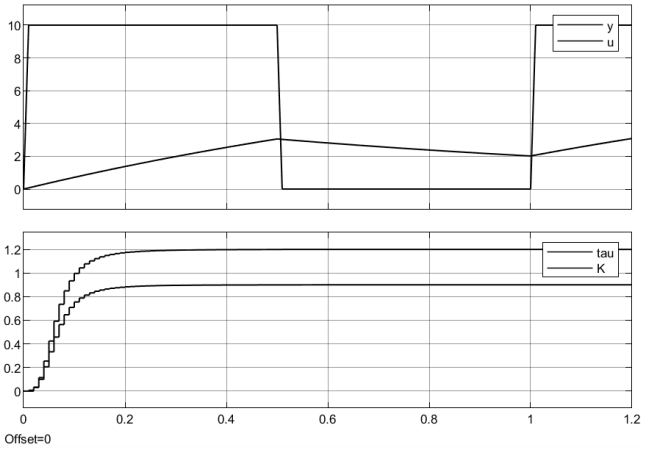

Fig. 8. RLS response

This response proves that RSL has successfully estimated K and τ in 0.2s. Exact value for this estimated K and τ are shown in two displays of Fig. 7, i.e. $K=0.9$ and τ=1.2s. These values are equal to the values described before. Input signal u that was using here, is a square wave. This type of signal does not affect the estimation result, even though its amplitude goes to zero. e.g. from $t=0.5s$ to $t=1s$, the estimated value K and τ still present correct value.

D. Design and Verification for Autotuning PI

Everything required to develop autotuning PI has been discussed. It is time to start development of autotuning PI simply by cascading RLS, PZC and digital PI as shown in Fig. 1. Hence, simulation for the autotuning PI can be constructed as shown in Fig. 9.

To investigate this simulation easily, RLS-PZC process is implemented separately from digital PI process. As can be seen, RLS-PZC receives *T*, *act_max*, ts and uc. These values are defined in Listing 1 line 1-6. Moreover, RLS-PZC needs to read plant input u and plant output y so that it can estimate plant parameters. The RLS-PZC employs RLS algorithm to estimate plant parameters (τ, K), and employs PZC to get PI parameter (Kp, Ki). Meanwhile, digital PI receives τ, Kp, Ki, T, y, *actuator_min* and *actuator_max* to produces control signal u that will be sent to the plant named Plant2. To evaluate the autotuning PI, continuous PI named Controller1 is run parallel with the autotuning PI hence their output can be compared. Running this simulation yields response as shown in Fig. 10.

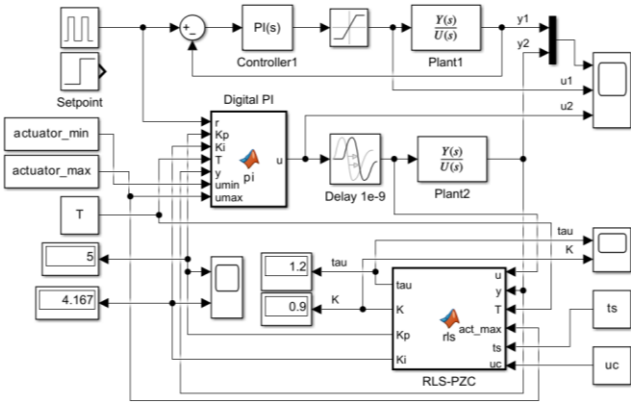

Fig. 9. Autotuning PI comprises of RLS-PZC and digital PI

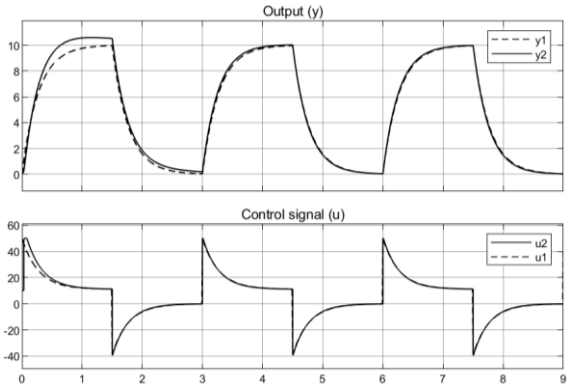

Fig. 10. Autotuning PI response when λ=0.96

The response of autotuning PI, $y2$, gradually coincides to the response of continuous PI, $y1$. This proves that autotuning PI has worked properly. If λ in (23) or lambda in Listing 3 line 8 is changed to 0.5, simulation yields response in Fig. 11.

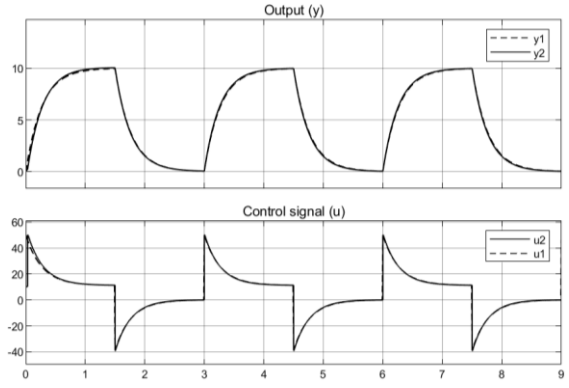

Fig. 11. Autotuning PI response when λ=0.5

When this response is compared to the response in Fig. 10, the significant difference is that $y2$ coincides to $y1$ since at the beginning response. This signifies that smaller λ will speed up or shorten the duration for parameters estimation and hence it will speed up the response of autotuning PI. The other important thing is that the absolute value of control signal $u2$ is never greater than fifty. Again, this phenomenon indicates that PZC works as expected, where the calculation of Kp and Ki has considered the *actuator_max* that is defined in Listing 1 line 4. This will guarantee that the actual response will comply with the designed response.

E. Verification for Performance

To verify whether autotuning PI algorithm can maintain its performance when the plant dc gain K is changed, the plant named Plant2 in Fig. 9 needs to be replaced with gain varying plant as shown in Fig. 12. This plant is the same as the previous plant, but now it also receives step signal routed to b. This mechanism enables it to be a varying gain plant. Here K is varied from 0.9 to 1.5 at $t = 4s$ by a step function. To enable comparison of the proposed system with an ideal system, two closed loop control systems have been provided to produce ideal response. Ideal response means response produced by PI controller where Kp and Ki is calculated using PZC with required time constant and known plant parameters.

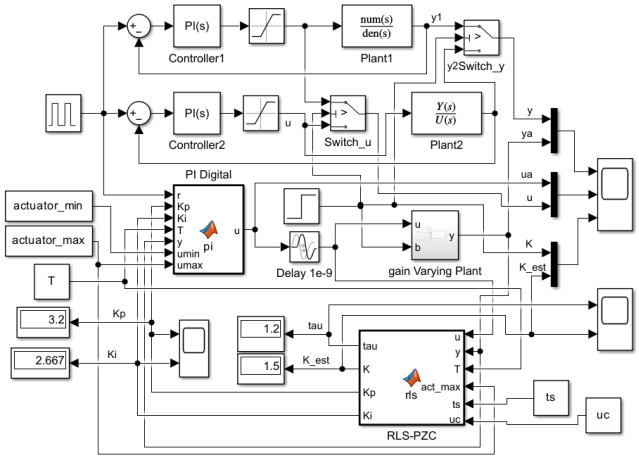

Fig. 12. Trial against changes of dc gain K

The first closed loop control system is controlled using Controller2 to produce ideal response for $t = [0\ 4]s$, while the second one is controlled using Controller1 to produce the reminding ideal response from $t = 4s$ until end of simulation. Then $y2Switch$ combines these two ideal responses. Running the simulation results in response as shown in Fig. 13. The last axis shows that K changes from 0.9 to 1.5 at $t = 4s$. This causes autotuning response, ya, to deviate from its ideal response, y.

Fortunately, autotuning PI has re-tuned PI parameters to new Kp=3.2 and new Ki=2.667 hence the response successfully recovered at t=7.5s, when ya completely coincides to y.

In the same manner, the performance against changes of time constant is tested using simulation shown in Fig. 14.

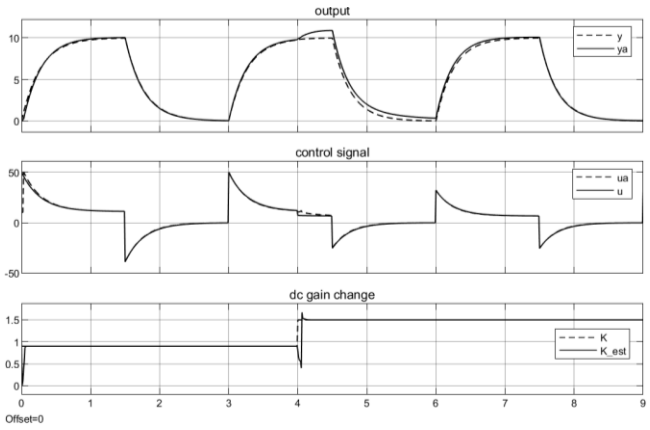

Fig. 13. Response against changes of dc gain K

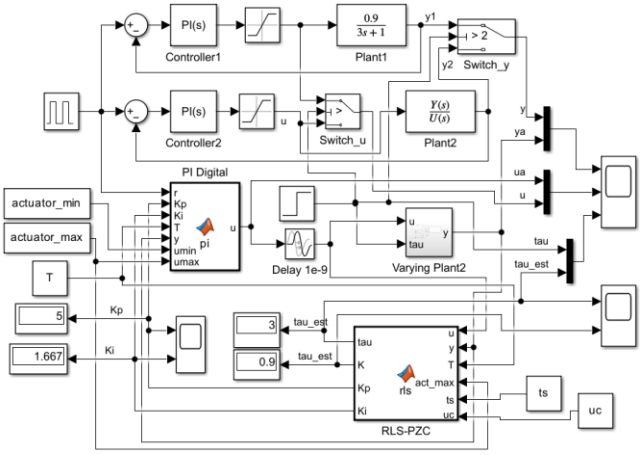

Fig. 14. Trial against changes of time constant τ

A step function is allocated to change plant time constant from 1.2s to 3s at $t = 4s$ with response as shown in Fig. 15.

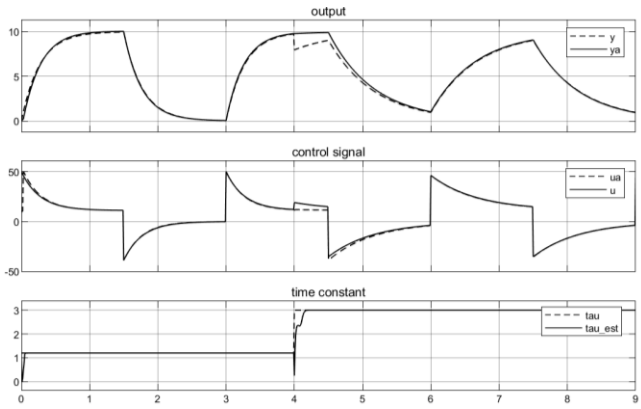

Fig. 15. Response against changes of time constant τ

As depicted in the last axis, the plant time constant was varied at $t = 4s$ from 1.2s to 3s. This causes the output of autotuning PI, ya, deviates from ideal response, y. Again, the autotuning PI successfully recovers its response to same as ideal response at t=7.5s. This response confirms that autotuning PI can maintain its performance. A controller having such capabilities will eliminate the need for manual tuning and the performance of the system or product produced will always be well maintained.

IV. CONCLUSIONS AND FUTURE WORK

This paper proposed an autotuning PI controller for first order plant which is comprises of RLS, PZC and digital PI. Simulation result proves that proposed controller works as expected, include:

- It can estimate parameters of first order plant in 0.2s.
- It can determine suitable PI parameters that will result in response without overshot, zero steady state and settling time as determined by selected time constant.
- It can generate control signal using digital PI algorithm for maintaining its performance (settling time and steady state error) even though plant parameters (dc gain and time constant) were changed. Recovery time is about 3.5s since parameters changes occurred.
- The control signal, produced by PI controller, also guaranteed never greater than its maximum value to ensure actual response fits to designed response.

All these outcomes will support the realization of simple operation and high-quality controllers.

However, this work needs to be verified in more real situation, e.g. by implementing script into C code run on a microcontroller (like AVR, ESP32, STM32, etc.) or FPGA and tested in real-world or in non-linear condition, to see and evaluate its actual performance and robustness against noise. To be more general and applicable or suitable for a particular real plant, the design requires enhancement so comply for FOPD (First Order Plant with Delay) and second order plant. The enhancement is essential since real plants are complicated higher order system and it is reasonable to estimate them as FOPD or second order than first order.

The main contribution of this article is PI parameter tuning based on PZC where resulting Kp and Ki will produce ideal first order response having required setting time and its control signal does not exceed specified limit. So, this article contributes on control domain especially about autotuning of PI controller using PZC-RLS algorithm. This method will simplify application of PI controller in real life as manual tuning by operator is not required. The main limitation of this method is that its success is dependent on the accuracy of plant transfer function, where the transfer function is obtained by RLS algorithm.

REFERENCES

[1] G. Mendonça, F. Afonso, and F. Lau, "Model order reduction in aerodynamics: Review and applications," *Proc Inst Mech Eng G J Aerosp Eng*, vol. 233, no. 15, pp. 5816–5836, 2019, doi: 10.1177/0954410019853472.

[2] K. Lu *et al.*, "Review for order reduction based on proper orthogonal decomposition and outlooks of applications in mechanical systems," *Mech Syst Signal Process*, vol. 123, pp. 264–297, 2019, doi: 10.1016/j.ymssp.2019.01.018.

[3] T. Hai, J. Zhou, and K. Muranaka, "An efficient fuzzy-logic based MPPT controller for grid-connected PV systems by farmland fertility optimization algorithm," *Optik (Stuttg)*, vol. 267, Oct. 2022, doi: 10.1016/j.ijleo.2022.169636.

[4] N. Priyadarshi, M. Bhaskar, P. Modak, and N. Kumar, "Hybrid Firefly-PSO MPPT Based Single Stage Induction Motor for PV Water Pumping With Deep Fuzzy-Neural Network Learning," *IEEE International Conference on Power Electronics, Drives and Energy Systems*, pp. 1-5, 2022, doi: 10.1109/pedes56012.2022.10080780.

[5] S. Ferahtia *et al.*, "Optimal adaptive gain LQR-based energy management strategy for battery–supercapacitor hybrid power system," *Energies (Basel)*, vol. 14, no. 6, 2021, doi: 10.3390/en14061660.

[6] A. Ma'arif and A. Çakan, "Simulation and arduino hardware implementation of dc motor control using sliding mode controller," *Journal of Robotics and Control (JRC)*, vol. 2, no. 6, pp. 582–587, 2021, doi: 10.18196/jrc.26140.

[7] R. N. Hasanah *et al.*, "Design of PI sliding mode control for Zeta DC–DC converter in PV system," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 70, no. 3, pp. 1–8, 2022, doi: 10.24425/bpasts.2022.140952.

[8] R. Rajesh and S. N. Deepa, "Design of direct MRAC augmented with 2 DoF PIDD controller: An application to speed control of a servo plant," *Journal of King Saud University - Engineering Sciences*, vol. 32, no. 5, pp. 310–320, 2020, doi: 10.1016/j.jksues.2019.02.005.

[9] S. Ciba, I. Iskender, and H. A. Ariani, "Comparing control performances of mrac and pid applied on a brushless dc motor," *International Journal on Technical and Physical Problems of Engineering*, vol. 12, no. 3, pp. 10–15, 2020.

[10] A. Latif, A. Z. Arfianto, H. A. Widodo, R. Rahim, and E. T. Helmy, "Motor DC PID system regulator for mini conveyor drive based-on matlab," *Journal of Robotics and Control (JRC)*, vol. 1, no. 6, pp. 185–190, 2020, doi: 10.18196/jrc.1636.

[11] L. H. Woldeamanuel and A. Ramaveerapathiran, "Design of Multivariable PID Control Scheme for Humidity and Temperature Control of Neonatal Incubator," *IEEE Access*, vol. 12, pp. 6051–6062, 2024, doi: 10.1109/ACCESS.2024.3349426.

[12] I. M. M. Eltigani and U. Nangia, "Intelligent PSO- PID Controller Based Maximum Power Point Tracking (MPPT) for Photovoltaic System," *2022 International Virtual Conference on Power Engineering Computing and Control: Developments in Electric Vehicles and Energy Sector for Sustainable Future (PECCON)*, pp. 1-6, 2022, doi: 10.1109/peccon55017.2022.9851173.

[13] P. Sarhadi, A. R. Noei, and A. Khosravi, "Model reference adaptive PID control with anti-windup compensator for an autonomous underwater vehicle," *Rob Auton Syst*, vol. 83, pp. 87–93, 2016, doi: 10.1016/j.robot.2016.05.016.

[14] H. Gai, X. Li, F. Jiao, X. Cheng, X. Yang, and G. Zheng, "Application of a new model reference adaptive control based on pid control in cnc machine tools," *Machines*, vol. 9, no. 11, 2021, doi: 10.3390/machines9110274.

[15] A. Gaga and B. El, "Iow cost PID controller implementation of SEPIC Converter using 8-bits Microcontroller Target for Photovoltaic Applications," *J. Electrical Systems*, vol. 18, no. 2, pp. 288-3–3, 2022.

[16] N. Roongmuanpha, J. Satansup, T. Pukkalanun, and W. Tangsrirat, "Design of Mixed-Mode Analog PID Controller with CFOAs," *Sensors*, vol. 24, no. 10, p. 3125, 2024, doi: 10.3390/s24103125.

[17] S. Mahata, N. Herencsar, and G. Maione, "Optimal approximation of analog PID controllers of complex fractional-order," *Fract Calc Appl Anal*, vol. 26, no. 4, pp. 1566–1593, 2023, doi: 10.1007/s13540-023-00168-x.

[18] A. H. Akbar, A. Ma, C. Rekik, A. J. Abougarair, and A. Molla, "Implementing PID Control on Arduino Uno for Air Temperature Optimization," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 6, no. 1, pp. 1–13, 2024, doi: 10.12928/biste.v6i1.9725.

[19] B. F. Zhou and J. L. Zhang, "Design of DC Motor PID Control System Based on STM32 Single Chip Microcomputer," *International Core Journal of Engineering*, vol. 6, no. 7, pp. 62–67, 2020, doi: 10.6919/ICJE.202007.

[20] A. F. M. Sulaiman, I. Siradjuddin, and N. R. Dzakiyullah, "The Design of a Microcontroller-based Self Balancing Robot Employing PID Control and Kalman Filter," *Webology*, vol. 19, no. 1, pp. 1194–1206, 2022, doi: 10.14704/web/v19i1/web19081.

[21] K. Sozański, "Low Cost PID Controller for Student Digital Control Laboratory Based on Arduino or STM32 Modules," *Electronics (Switzerland)*, vol. 12, no. 15, 2023, doi: 10.3390/electronics12153235.

[22] J. Kulisz and F. Jokiel, "A Hardware Implementation of the PID Algorithm Using Floating-Point Arithmetic," *Electronics*, vol. 13, no. 8, p. 1598, 2024, doi: 10.20944/preprints202401.17.

[23] I. Sukma, A. D. Suseno, P. Bakti, W. Ardiatna, and I. Supono, "Design and development of automatic voltage regulator using Ziegler-Nichols

PID for electrical irons testing," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 6, pp. 3938–3947, 2024, doi: 10.11591/eei.v13i6.7326.

[24] P. H. G. Coelho, J. F. M. Amaral, Y. C. Bacelar, E. N. Rocha, M. Bentes, and T. S. Souza, "Tuning Analog PID Controllers by Multi-Objective Genetic Algorithms with Fuzzy Aggregation," *International Conference on Enterprise Information Systems, ICEIS - Proceedings*, vol. 1, pp. 563–571, 2023, doi: 10.5220/0011976900003467.

[25] S. Mallick and U. Mondal, "MRAC based intelligent PID controller design technique for a class of dynamical systems," *Sadhana - Academy Proceedings in Engineering Sciences*, vol. 49, no. 2, 2024, doi: 10.1007/s12046-024-02457-4.

[26] E. Mendez *et al.*, "ANN based MRAC-PID controller implementation for a furuta pendulum system stabilization," *Advances in Science, Technology and Engineering Systems*, vol. 5, no. 3, pp. 324–333, 2020, doi: 10.25046/aj050342.

[27] J. Jallad and O. Badran, "Firefly algorithm tuning of PID position control of DC motor using parameter estimator toolbox," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 2, pp. 916–929, 2024, doi: 10.11591/eei.v13i2.6216.

[28] A. Dhanda and D. Niwas, "Comparison of Ziegler - Nichols, Cohen - Coon and Fuzzy Logic Controllers for Heat Exchanger Model: A Review," *International Journal of Engineering, Management, Humanities and Social Sciences Paradigms (IJEMHS)*, vol. 15, no. 01, pp. 1–7, 2015.

[29] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of PID control, tuning methods and applications," *Int J Dyn Control*, vol. 9, no. 2, pp. 818–827, 2021, doi: 10.1007/s40435-020-00665-4.

[30] M. Hofreiter, "Relay identification using shifting method for pid controller tuning," *Energies (Basel)*, vol. 14, no. 18, 2021, doi: 10.3390/en14185945.

[31] J. Sánchez Moreno, S. Dormido Bencomo, and J. M. Díaz Martínez, "Fitting of generic process models by an asymmetric short relay feedback experiment—The n-shifting method," *Applied Sciences*, vol. 11, no. 4, p. 1651, 2021.

[32] Y. Urakawa, "Vibration suppression in position control system by pole zero cancellation using limited pole placement method," *IEEJ Journal of Industry Applications*, vol. 8, no. 2, pp. 256–262, 2019, doi: 10.1541/ieejjia.8.256.

[33] Y. Wang, S. Deng, and W. Lu, "Zero-Pole Cancellation Based Iterative Learning Control Method for Speed Ripple Suppression of Single Rolling Rotor Compressor," *IEEE Access*, vol. 12, pp. 55587–55599, 2024, doi: 10.1109/ACCESS.2024.3390091.

[34] O. Miguel-Escrig and J. A. Romero-Pérez, "Improving the identification from relay feedback experiments," *Automatica*, vol. 135, p. 109987, 2022, doi: 10.1016/j.automatica.2021.109987.

[35] J. A. Romero-Pérez, O. Miguel-Escrig, J. Sánchez-Moreno, and S. Dormido-Bencomo, "Improving the relay feedback identification by using a gain-changing non-linearity," *IFAC-PapersOnLine*, vol. 58, no. 7, pp. 418–423, 2024, doi: 10.1016/j.ifacol.2024.08.098.

[36] K. J. Åström, T. Hägglund, C. C. Hang, and W. K. Ho, "Automatic tuning and adaptation for PID controllers - a survey," *Control Eng Pract*, vol. 1, no. 4, pp. 699–714, 1993, doi: 10.1016/0967-0661(93)91394-C.

[37] M. Moučka and M. Hofreiter, "PID controller with an autotuning function," *IEEE Access*, vol. PP, p. 1, 2024, doi: 10.1109/ACCESS.2024.3462090.

[38] N. Bouarroudj *et al.*, "A new tuning rule for stabilized integrator controller to enhance the indirect control of incremental conductance MPPT algorithm: Simulation and practical implementation," *Optik (Stuttg)*, vol. 268, p. 169728, Oct. 2022, doi: 10.1016/j.ijleo.2022.169728.

[39] A. Rehan, I. Boiko, and Y. Zweiri, "Autotuning of PID controller using the modified relay feedback test and optimization under uncertainty principle for robotic manipulators," *IET Control Theory and Applications*, vol. 18, no. 5, pp. 581–602, 2024, doi: 10.1049/cth2.12592.

[40] R. De Keyser, I. Birs, and C. Ionescu, "A performant PID autotuner based on a short relay test ?," *IFAC PapersOnLine*, vol. 58, no. 7, pp. 328–333, 2024, doi: 10.1016/j.ifacol.2024.08.080.

[41] D. Pavković, D. Lisjak, D. Kolar, and M. Cipek, "PI/PID Controller Relay Experiment Auto-Tuning with Extended Kalman Filter and Second-Order Generalized Integrator as Parameter Estimators," *Tehnicki Vjesnik*, vol. 30, no. 3, pp. 715–723, 2023, doi: 10.17559/TV-20221003112627.

[42] R. S. Patil, S. P. Jadhav, M. D. Patil, N. Mumbai, and C. Author, "Review of Intelligent and Nature-Inspired Algorithms-Based Methods for Tuning PID Controllers in Industrial Applications," *Journal of Robotics and Control (JRC)*, vol. 5, no. 2, pp. 336–358, 2024, doi: 10.18196/jrc.v5i2.20850.

[43] E. W. Suseno, A. Ma'arif, and R. D. Puriyanto, "Tuning Parameter Pengendali PID dengan Metode Algoritma Genetik pada Motor DC," *TELKA - Telekomunikasi Elektronika Komputasi dan Kontrol*, vol. 8, no. 1, pp. 1–13, 2022, doi: 10.15575/telka.v8n1.1-13.

[44] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems," *Heliyon*, vol. 8, no. 5, p. e09399, 2022, doi: https://doi.org/10.1016/j.heliyon.2022.e09399.

[45] Y. S. Arikusu and N. Bayhan, "Design of a Novel PID Controller Based on Machine Learning Algorithm for a Micro-Thermoelectric Cooler of the Polymerase Chain Reaction Device," *IEEE Access*, vol. 12, no. April, pp. 61959–61977, 2024, doi: 10.1109/ACCESS.2024.3392734.

[46] S. Cheng *et al.*, "DiffTune: Auto-Tuning through Auto-Differentiation," *IEEE Transactions on Robotics*, vol. 40, pp. 4085–4101, 2024, doi: 10.1109/TRO.2024.3429191.

[47] W. V Jahnavi, J. N. C. Sekhar, and A. S. Reddy, "A Review on PID Controller Tuning Using Modern Computational Algorithms," *Journal of Emerging Technologies and Innovative Research*, vol. 10, no. 8, pp. 160–168, 2023.

[48] I. A. Abbas and K. Mustafa, "A review of adaptive tuning of PID-controller: Optimization techniques and applications," *Int. J. Nonlinear Anal. Appl. In Press*, vol. 6822, pp. 2008–6822, 2023.

[49] M. J. Hasan and M. Y. Hellan, "Control and Modeling of PSO-PID based MPPT," *Int J Comput Appl*, 2022, doi: 10.5120/ijca2022922112.

[50] S. B. Joseph, E. G. Dada, A. Abidemi, D. O. Oyewola, and B. M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems," *Heliyon*, vol. 8, no. 5, p. e09399, 2022, doi: 10.1016/j.heliyon.2022.e09399.

[51] C. Aoughlis, A. Belkaid, I. Colak, O. Guenounou, and M. A. Kacimi, "Automatic and Self Adaptive P&O MPPT Based PID Controller and PSO Algorithm," *IEEE International Conference on Renewable Energy Research and Applications*, pp. 385-390, 2021, doi: 10.1109/icrera52334.2021.9598489.

[52] A. A. Ponomarev, Y. I. Kudinov, F. F. Pashchenko, and E. S. Duvanov, "Analysis and Synthesis of Adaptive PID Controller with MRAC-MIT System," *Proceedings - 2020 2nd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency, SUMMA 2020*, pp. 527–532, 2020, doi: 10.1109/SUMMA50634.2020.9280651.

[53] S. J. Hammoodi, K. S. Flayyih, and A. R. Hamad, "Design and implementation speed control system of DC motor based on PID control and matlab simulink," *International Journal of Power Electronics and Drive Systems*, vol. 11, no. 1, pp. 127–134, 2020, doi: 10.11591/ijpeds.v11.i1.pp127-134.

[54] B. Lai and D. S. Bernstein, "Generalized Forgetting Recursive Least Squares: Stability and Robustness Guarantees," *IEEE Trans Automat Contr*, pp. 1–16, 2024, doi: 10.1109/TAC.2024.3394351.

[55] A. L. Bruce, A. Goel, and D. S. Bernstein, "Recursive Least Squares with Matrix Forgetting," *Proceedings of the American Control Conference*, pp. 1406–1410, 2020, doi: 10.23919/ACC45564.2020.9148005.

[56] S. A. U. Islam and D. S. Bernstein, "Recursive Least Squares for Real-Time Implementation," *IEEE Control Syst*, vol. 39, no. 3, pp. 82–85, 2019, doi: 10.1109/MCS.2019.2900788.

[57] R. Ortega, V. Nikiforov, and D. Gerasimov, "On modified parameter estimators for identification and adaptive control. A unified framework and some new schemes," *Annu Rev Control*, vol. 50, no. 2019, pp. 278–293, 2020, doi: 10.1016/j.arcontrol.2020.06.002.

[58] A. L. Bruce, A. Goel, and D. S. Bernstein, "Convergence and consistency of recursive least squares with variable-rate forgetting," *Automatica*, vol. 119, p. 109052, 2020, doi: 10.1016/j.automatica.2020.109052.

[59] H.-S. Shin and H.-I. Lee, "A New Exponential Forgetting Algorithm for Recursive Least-Squares Parameter Estimation," *arXiv preprint arXiv:2004.03910*, 2020.

[60] V. Shaferman, M. Schwegel, T. Glück, and A. Kugi, "Continuous-time least-squares forgetting algorithms for indirect adaptive control," *Eur J Control*, vol. 62, pp. 105–112, 2021, doi: 10.1016/j.ejcon.2021.06.015.

[61] M. Bin, "Generalized recursive least squares: Stability, robustness, and excitation," *Syst Control Lett*, vol. 161, p. 105144, 2022, doi: 10.1016/j.sysconle.2022.105144.

[62] A. L. Bruce, A. Goel, and D. S. Bernstein, "Necessary and sufficient regressor conditions for the global asymptotic stability of recursive least squares," *Syst Control Lett*, vol. 157, p. 105005, 2021, doi: 10.1016/j.sysconle.2021.105005.