IMPROVING THE PERFORMANCE OF AN INADEQUATELY TUNED PID CONTROLLER BY INTRODUCING A POLYNOMIAL MODEL BASED INCREMENT IN PID CONTROL VALUE

Dushko Stavrov, Gorjan Nadzinski, Goran Stojanovski, Stojche Deskovski,
Faculty of Electrical Engineering and Information Technology – University of Ss. Cyril and Methodius in Skopje, Republic of Macedonia
dushko.stavrov@feit.ukim.edu.com.mk

Abstract: *In control literature, one can easily find a variety of different examples for industrial control, where contemporary control algorithms are implemented. Surprisingly, there are not many known examples where the state-of-the-art control algorithms have been implemented in real-time control systems. Instead, researchers usually implement algorithms that are proven to be reliable, fast and easy to implement. One control solution that has been proven to satisfy all previously mentioned attributes is PID. However, despite all its good assets it has two major deficiencies. One of them is that it can't adapt on the diversities caused by the variation occurring in the model parameters and still it can't control nonlinear systems due to multiple operating points present there. Therefore, to deal with those weaknesses an improvement in PID control structure has been introduced in the form of supervisory mechanism (SM) which as a main constitute part has a quadratic polynomial model. Thus, the control value of the newly proposed PID algorithm is formed of two terms, the first one is the value calculated by standard PID and the second one is the value calculated by the SM. The quadratic model forming part of the SM is obtained based on the past value of the error. Nevertheless, the use of quadratic model introduces additional complexity into the PID controller. Furthermore, the quadratic model should be updated fast enough and also it has to describe the data adequately. These aspects are analyzed and discussed in details in this paper. Moreover, an algorithm is introduced which will guarantee that the data used for calculation of the quadratic model is suitable.*

Keywords: PID controller, supervisory mechanism, quadratic model

1. Introduction

We are witnesses of an era in control system theory in which the scientists are in search for solutions of the problems arising in automation industry. However, there are clear evidences indicating that the existing control solutions present in this day and age are not a stand-alone solution but instead they are supported by a lot of additional procedures, which instead to shrink the complexity of the control mechanisms, they tend to add additional computational burden. As a result, the newly proposed algorithms are not fast, flexible, reliable and are difficult to implement. Even more their implementation costs a lot. All previously stated directly collides with the industry strive for creating fully automated industrial processes, since the new control solutions don't give the flexibility needed by industry in order to maximize the profit with the least possible investment.

So far the control solution that is proven to be at the same time flexible, reliable and cost effective is the PID control algorithm. To support this fact, the studies ([2], [3]) have shown that PID is most widely and most often used control mechanism in industrial processes. As a fact, the MPC control mechanism, ([7], [8]), which is supposed to reach the PID controller use in future (in industry) is a complex solution which is supported by an optimization problem and a plant model in the background. Moreover, it is a slow and expensive solution for the most of the industrial companies. Over the long history of PID controller use it was augmented with new features for improving its efficiency. Nevertheless, the thing that changed the least is its simple three term structure comprised of proportional, derivative and integral term. Also, one should make a note of the references ([5], [6]) which are indicating that there are more than a couple of hundred methods for PID parameters synthesis. The widely known method which is frequently used for obtaining PID parameters is Zeigler Nichols method ([4]). Even so, with that many procedures for selection of PID parameters, the most commonly used procedure for tuning the PID parameters is by trial and error. Evidence of this statement is given with the reference [10], which indicates that 80% of the PID controllers are poorly tuned where 30% of them operate in manual mode.

Although there are a lot of advantages of PID control, it major weaknesses are that it can't adapt to variation in parameters of control object (note CO Fig1) along with, it also struggles when it is applied to control nonlinear control object due to the existence of multiple operation points. The reason for decrease in PID performance is obvious, the PID control mechanism is calculating the control value only based on the current value of error. Hence, it can be clearly seen from Eq. 1. Therefore, to attack the diversities caused by the variation in model parameters along with the difficulties caused by the multiple operating points of the nonlinear control systems the PID is often reinforced with adaptive mechanism [11]. The main drawback of this additional mechanisms are that they are adding additional computation complexity in generation of the PID control value and at the same time seriously endanger the conservation of system stability.

In the previous work, done in [1], to charge of the previously mentioned problems, a supervisory mechanism was designed and added to the simple PID control structure. The supervisory mechanism was comprised of quadratic polynomial model estimated based on the past error values, where the data used for obtaining the model was collected in a specific moments of time. Furthermore, the comparison done between the proposed algorithm and the simple PID controller, has given the evidence that the proposed algorithm was able to beat the performance of the simple PID. However, the issue that emerged was that the model didn't adapt fast enough on the newly gathered points and also there were cases when the estimated quadratic model didn't describe effectively the data composed of past error points. Hence, in order to improve the speed of model update along with to assure that the data used for model mapping is adequate, the profile of the dynamical variable has been changed and an algorithm was introduced. The great concern of the Algorithm 1 is to guarantee the data used for model calculation is fair.

The rest of the paper is structured as follows. Firstly, the mathematical background of the simple PID and DUPID is presented; then the DUPID is discussed in more details. Secondly, the case study is shortly described. Afterwards, in the simulation part the results are given. Finally, conclusions and outlook for future work are given.

2. Formulation of simple PID and DUPID

Simple PID formulation

In this section a short explanation of the simple parallel PID controller implemented on a computer will be given. It is a well-known fact that PID control algorithm is implemented in more than 95% of the control loops in industry [2], [3]). As mentioned before, in its history of use should be noted that its basic structure, contained of three terms proportional, derivative and integral, was

never changed. And the very reason for that is simple, PID controller has been proven to be fast, reliable and easy to implement. In industry different forms of implementation of PID control mechanism can be recognized. More than ten particular forms, see ([5], [6]), in a whole. Likewise, most widely used PID control structure is the simplest parallel form given with Eq. 1.The control value of the parallel PID is produced in the form given with equation (1).

$$u_{PID}[i] = u_0 + K_p e[i] + K_i \sum_{0}^{i} e[i]\,T_i + K_d \frac{PV[i] - PV[i-1]}{T_i}, \quad \text{Eq. 1}$$

where, u_0 is a bias in the control signal, $e[i] = SP[i] - PV[i]$ represents the current error, which is calculated as difference between set point (SP) and the process value (PV). The coefficients of the PID are, K_p, K_i and K_d each represents proportional constant, integral constant and derivative constant respectively. Practically, the generation of the control signal is done in one step very fast. It can be seen, from Eq. 1, that the control value only depends on the current value of the error as well as of the difference $PV[i] - PV[i-1]$. On the Figure 1, the feedback control loop consisted of PID controller and control object is given.

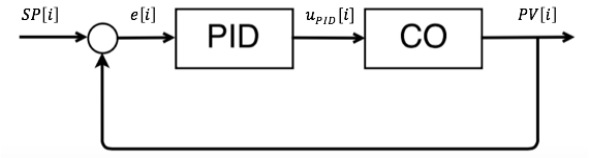

Figure 1 *PID control loop.*

DUPID formulation

The dynamically updated PID's, or shortly DUPID (see Figure 2 dashed rectangular) is comprised of two parts, the first part is the simple PID structure given with the mathematical formulation Eq. 1 which acts as a mechanism for conservation of CO stability. The second part is the supervisory mechanism depicted as SM on Figure 2.

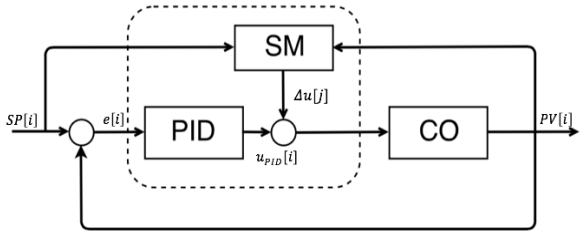

Figure 2 *DUPID control loop.*

The SM mechanism is responsible for calculation of the increment value $\Delta u[j]$. The calculation of $\Delta u[j]$ is done in multiple steps. The steps for calculation of the increment value are given in form of a flow chart shown with Figure 3. They will be described later in this part. Thus, the DUPID produces the control signal as a sum of the control value generated by PID and the increment value produced by the supervisory mechanism. In other words the control signal applied to the CO has two terms, and it is given as $u_c[i]$:

$$u_c[i] = u_{PID}[i] + \Delta u[j]. \quad \text{Eq. 2}$$

Where the second term $\Delta u[j]$ is generated by the SM at a specific moments of time.

Next, the process of calculation of the increment value will be described. The steps for obtaining the Δu value are depicted on the flow chart given on Figure 3. Most of the SM components shown on Figure 3 were introduced and explained in detail in [1]. However, for sake of clarity we will shortly describe each of the blocks given on the flow chart given on Figure 3. The first step is the initialization phase. In this phase the CO parameters as well as the SM parameters are set to its initial values. The main SM parameters are, N_{pp} and σ_{tol}, where N_{pp} defines the horizon of past error points that are used for regression and the value of the parameter σ_{tol} which defines the needed variation in the values of regression points. Both parameters are tuned by the user. After the initialization phase, the CO is controlled in the fashion given with Figure 2. until the condition $i \leq N_s$ is true. Where N_s stands for number of simulation steps. In the second phase, the value $\Delta u[j]$ is calculated in specific moment of time defined with the current value of the dynamical variable $DV[i]$. Whenever the condition given with the Eq. 3 is met, the counter of simulation steps i is divisible with the current value of the dynamical variable. In that very moment, an error point is collected Eq. 4 and the model is updated.

$$mod(i, DV[i]) = 0, \quad \text{Eq. 3}$$

Later, right after an error point is gathered the condition of how many points were collected is checked. Where, the value of counter j, emphasizes how many points where collected and in the same time it gives an information how many times the quadratic model has been updated, by current iteration step i.

$$e_s[j] = SP[i] - PV[i] \quad \text{Eq. 4}$$

If the number of gathered points is smaller or equal than 3 a simple metric given with Eq. 5 is responsible for calculation of the increment value.

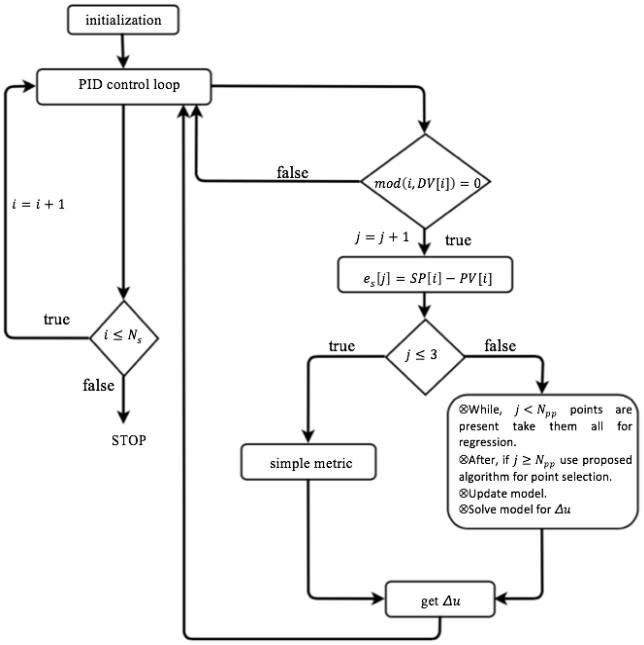

Figure 3 *Flow chart diagram of proposed DUPID algorithm.*

For purpose of the model estimation the simple metric given with Eq. 5 has been used. Using this metric, the initial model based on the minimum number of points is obtained. The parameter k_{ss} is tuned by the user.

$$\Delta u = k_{ss} \sum_{m=1}^{j} e_s(m), \quad \text{Eq. 5}$$

After the initial model is estimated based on the first 3 points the model is dynamically updated every time condition given with Eq. 3 is met. While, the number of gathered points is smaller than the value N_{pp} all collected points are used for model update. Else when $j \geq N_{pp}$ the model is dynamically updated based on the last N_{pp} points. Finally, in the last phase the value Δu is calculated in the form given with Eq. 6:

$$\Delta u = \min_{\Delta u}(f(\Delta u) = 0) \quad \text{Eq. 6}$$

To fit the model Eq. 7 the minimum number of points needed is 3. In other words, the sufficient number of points is equal to the

number of unknown parameters. The equation of the model is given by:

$$f(\Delta u) = A(\Delta u)^2 + B(\Delta u) + C \qquad \text{Eq. 7}$$

The parameters A, B and C of the model are calculated by solving the normal equation:

$$p = (M^T M)^{-1} M^{-1} \cdot E_0 \qquad \text{Eq. 8}$$

Where $p \in \mathbb{R}^3$ is a vector of parameters,

$$p = [A, B, C]^T, \qquad \text{Eq. 9}$$

$M(\Delta u) \in \mathbb{R}^{N_{pp} \times 3}$ is a matrix of N_{pp} points, which are considered to determine the parameters p.

$$M = \begin{bmatrix} (\Delta u[1])^2 & \Delta u[1] & 1 \\ \vdots & \vdots & \vdots \\ (\Delta u[N_{pp}])^2 & \Delta u[N_{pp}] & 1 \end{bmatrix} \qquad \text{Eq. 10}$$

At last, $E_0 \in \mathbb{R}^{N_{pp} \times 1}$ denotes the vector of error values, Eq. 11.

$$E_0 = \begin{bmatrix} e_s[1] & \dots & e_s[N_{pp}] \end{bmatrix}^T \qquad \text{Eq. 11}$$

Equation Eq. 8 can be solved if the inverse $(M^T M)^{-1}$ exists, which means that the matrix M should have rank equal to the number of parameters, in our case that number is 3. In other words, the points used for regression have to be distributed in a way that the rank of matrix M is not smaller than 3.

At first sight, 3 points seem to be enough to solve the equation Eq. 8. However, there are cases in which the chosen 3 points are not suitable. First of all, it is clear that two points should not be placed in the same location which leads to a reduced rank of M. Furthermore, it has to be ensured that the points are not distributed on a line. Anyway, if that is the case then the information provided by the points is not adequate to describe a quadratic function exactly.

To tackle previously mentioned issue, an algorithm was designed that will guarantee that the rank of the inverse matrix $(M^T M)^{-1}$ is greater or at least equal to 3. Namely, a rank greater or equal to 3 will be achieved if and only if the data of the last N_{pp} points consists at least 3 different points. The proposed algorithm is given with the following pseudo code, Algorithm 1. Where the vectors $\boldsymbol{\Delta U}$ and $\boldsymbol{E}$ denote all past points for increment and error value respectively. The output of this code are the vectors of adequate data U_0 and E_0 used for model update.

3. Case Study

In this paper as a CO a highly nonlinear CSTR system has been considered. The parameters of the model as well as the equations for the one are given in, ([1], [9]).

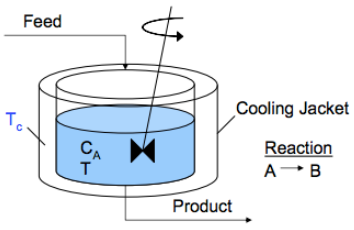

Figure 4 *CSTR system simple graphical representation.*

The chemical reactor is modelled by the following two equations:

$$\frac{dT}{dt} = \frac{q}{V}(T_f - T) + \frac{\Delta H}{\rho C_p} k_0 e^{-\frac{E}{RT}} + \frac{UA}{V \rho C_p}(T_c - T) \qquad \text{Eq. 12}$$

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{Af} - C_A) - k_0 C_A e^{-\frac{E}{RT}} \qquad \text{Eq. 13}$$

where T_c, the temperature of the cooling jacket fluid, is the manipulated variable and T, reactor temperature, is the controlled variable. For simulation purpose the equations of the model are numerically solved by the function ode23t in MATLAB with integration step T_i =0.1 min.

Algorithm 1 Selection of adequate data for regression

```
cnt0 = 0
cnt1 = 0
cnt2 = 0
U_s = []
E_s = []
for i = length(ΔU):-1: length(ΔU) - N_pp+1
        if(|ΔU(end)-ΔU(i)| > σ_tol)
                cnt0++
        end if
end for
if (cnt0≥3)
        U_0 ← U_0 ∪ ΔU(end-N_pp + 1: end)
        E_0 ← E_0 ∪ E(end-N_pp + 1: end)
else if (cnt0 == 2)
        for i = length(ΔU) - N_pp+1: -1: 1
                if(ΔU(i) ≥ ΔU(end) + σ_tol or ΔU(i) ≤ ΔU(end) − σ_tol)
                        cnt1++
                        if(cnt1==1)
                                U_0 ← U_0 ∪ append(ΔU(i), ΔU(end-N_pp + 1: end))
                                E_0 ← E_0 ∪ append(E(i), E(end-N_pp + 1: end))
                        end if
                end if
        end for
else
        for i = length(ΔU) - N_pp+1: -1: 1
                if(ΔU(i) ≥ ΔU(end) + σ_tol or ΔU(i) ≤ ΔU(end)-σ_tol)
                        cnt2++
                        U_s ← U_s ∪ append(ΔU(i))
                        E_s ← E_s ∪ append(E(i))
                        if(cnt2==2)
                                U_0 ← U_0 ∪ append(U_s, ΔU(end-N_pp + 1: end))
                                E_0 ← E_0 ∪ append(E_s, E(end-N_pp + 1: end))
                        end if
                end if
        end for
end if
return U_0, E_0 % return the arrays used for model estimation
```

4. Simulation

In this part the algorithm given in [1] reinforced by the algorithm given with the pseudo code Algorithm 1 is tested. For sake of simplicity the comparison of this algorithm with simple PID is dropped because in [1] was already proven that DUPID beat the performance of simple PID. The DUPID, was tested in two different scenarios, each one comprised of two cases. In the Scenario 1 a ramp change is introduced in the SP value. In the case 1, the desired SP changes linearly from 300 to 305 K. In the second case, case 2, the SP is linearly changing from 305 to 300 K. In Scenario 2 a S-curve change was introduced in the SP. In Case 1 of Scenario 2, a S-curve change from 300 to 305K is present. In the Case 2 a S-curve change is introduced in opposite direction, from 305 to 300 K in SP. A simulation of both scenarios was carried out over a period of 100 minutes or speaking in simulation steps $N_s = 1000$. The parameters of the algorithm given with the pseudo code are $N_{pp} = 5$ and $\sigma_{tol} = 0.09$. The speed of model update is defined by the value of DV. The evolution of variable DV in time is given on Fig5. The parameters of the PID controller used in both scenarios are the same as the ones given in Table 1 given in [1].

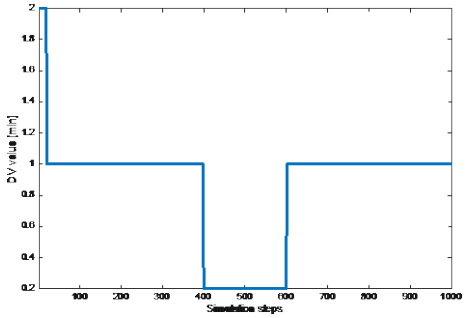

Figure 5 *Speed of point collection and model update.*

The Figure 5depicts how fast the model is updated. From the figure can be read that the first point is collected after 2 minutes and

afterwards until the change in SP value is present a point is collected and the model is updated on every 1 minute. In other words, the needed 3 points for generation of the initial model are collected after 4 minutes. Also, from the Figure 5 can be clearly seen that starting from 400 until the 600-th simulation step the speed of model update is drastically increased, speaking in time the model is updated on every 0.2 minutes.

Next, all 4 figures given below depict the temperature response of the reactor (PV), given with blue line, versus the desired temperature (SP), given with red dashed line, as the time evolve. Below the plot of T, on the second and third plots the values of the increment value Δu and the values of the error at the current iterate j are given.

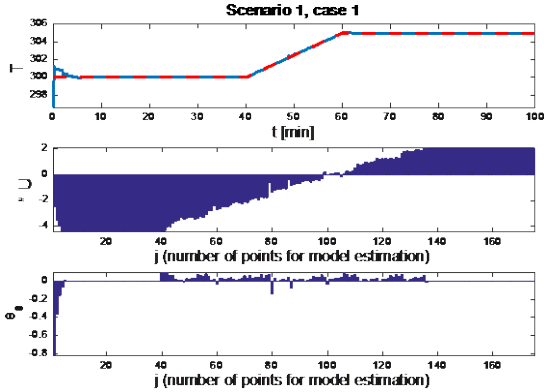

Figure 6 *Response of the system along with the values of the increment value and error in case 1 of Scenario 1.*

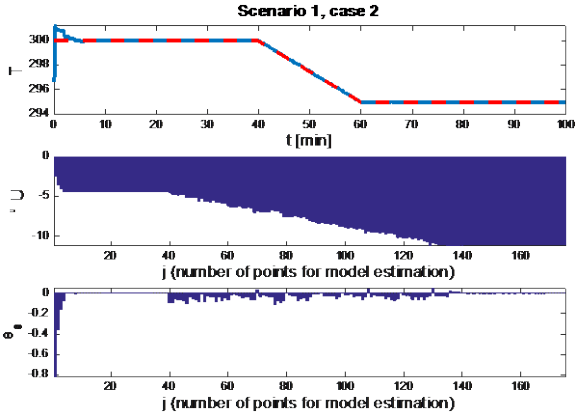

Figure 7 *Response of the system along with the values of the increment value and error in case 2 of Scenario 1.*

From Figure 6 and Figure 7, representing the case 1 and 2 respectively of Scenario 1 can be concluded that the DUPID has proven to be effective not only in steady state phase but also in the phase when linear change is present. It is easily noticeable that the CO follows the SP value with high precision.

Following figures, Figure 8 and Figure 9 show the simulation results obtained in Scenario 2. From Figure 8 and Figure 9, representing the case 1 and 2 respectively of Scenario 2 the following conclusion can be sketch. The DUPID controller has proven to be effective in dealing with the effects of the nonlinear change introduced in the desired temperature profile. Still, as in the Scenario 1 the steady state precision has been conserved.

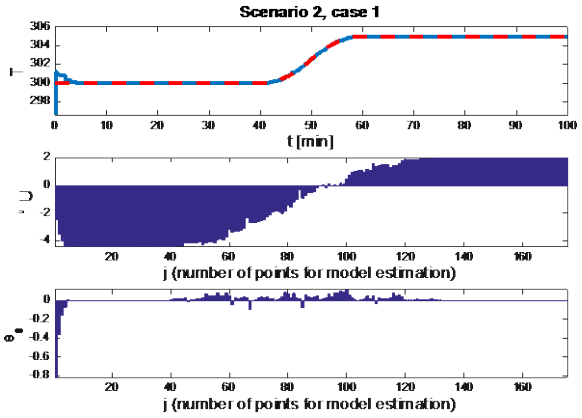

Figure 8 *Response of the system along with the values of the increment value and error in case 1 of Scenario 2.*

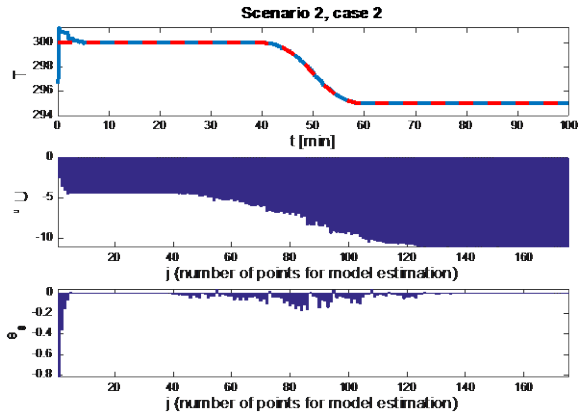

Figure 9 *Response of the system along with the values of the increment value and error in case 2 of Scenario 2.*

5. Conclusion and outlook for future work

In this paper an improvement has been introduced to the algorithm previously presented, in [1]. Previously in [1] was shown that DUPID has beaten the performance of the standard PID control algorithm when both controllers were applied to control the nonlinear CSTR system. However, the problem that had to be dealt there was that the quadratic model that is incorporated in the SM didn't update fast enough and there were cases when the points selected for regression were not adequate. So, to improve the speed of model update as well as to guarantee that the data used for regression is adequate the profile of the variable DV was changed and also Algorithm 1was introduced. Furthermore, the simulation results have shown that the DUPID has the ability to drive the system effectively to steady state and at the same time to follow the changes in desired temperature with high precision, almost with no error.

Here should be mentioned that an effort has been made to smoothen the variation in the data present with increment value Δu and e_s by introducing a simple complementary filter. However, after extensive tests, has been decided to drop this idea because the complementary filter not only has proven not to give satisfactory results but it also increased the time needed for calculation of Δu.

Future work will consist of implementing the DUPID and afterwards comparing it with linear MPC on other type of system possibly a bioengineering system. In such conditions, it is expected that the proposed control approach DUPID will achieve similar performance as MPC but the calculation of the control value in the case of DUPID will be shorter than in the MPC case.

ACKNOWLEDGEMENTS

The work was funded by the Faculty of Electrical Engineering and Information Technologies in Skopje, Republic of Macedonia, through the ERESCOP Project.

References

[1] Stavrov D., Stojanovski G. and Deskovski S., "Improving the precision of plant response by modeling the steady state error", International Scientific Journal INDUSTRY 4.0, ISSN 2543-8582, ISSUE 4/2017.

[2] Astrom, K. J. and Hagglund, T., Advanced PID control, North Carolina: ISA, 2006.

[3] Rotach, V.Ya, Teoriya avtomaticheskogo upravleniya: uchebnik dlya VUZov (Automated Control Theory: Univeristy Textbook), Moscow: MEI Publishing House, 2008.

[4] Ziegler, J.G. and Nichols, N.B., Optimum Settings for automatic controllers, Trans. ASME, 1942 vol.64, pp 759-768.

[5] O'Dwayer, A., Handbook of PI and PID controller Tuning Rules, London: Imperial College Press, 2009 3th ed.

[6] O'Dwayer, A., Handbook of PI and PID controller Tuning Rules, London: Imperial College Press, 2006 2th ed.

[7] Badwe, A. S., R. D. Gudi, R. S. Patwardhan, S. L. Shah, and S. C. Patwardhan, Detection of Model-Plant Mismatch in MPC applications, J. Process Control, 19, 1305 (2009)

[8] Camacho, E. F., and C. Bordons, Model Predictive Control 2nd ed., Springer-Verlag, New York, 2003

[9] D.E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle III, Process dynamics and control: John Wiley & Sons, 2010.

[10] P. Van Overschee and B. De Moor, in Preprints, Proc. PID '00: IFAC Workshop, Terrassa, Spain, pp. 687-692, 2000

[11] Rania A. Fahmy, R. I. Badr, Farouk A. Rahman, "Adaptive PID Controller Using RLS for SISO Stable and Unstable systems", Advances in Power Electronics, Volume 2014, Article ID 507142.