Advances in Science and Technology Research Journal

*Advances in Science and Technology Research Journal*, 2026, 20(8), 295–307
https://doi.org/10.12913/22998624/221712
ISSN 2299-8624, License CC-BY 4.0

Received: 2026.04.02
Accepted: 2026.06.01
Published: 2026.07.01

# Application of Q-learning controller for processes with dead time

Jakub Musiał[1]

[1] Silesian University of Technology, Faculty of Automatic Control, Electronics and Computer Science, Department of Automatic Control and Robotics, 44-100 Gliwice, Poland
E-mail: jakub.musial@polsl.pl

## ABSTRACT

This paper presents an extension of a self-improving, model-free Q-learning controller for industrial processes characterized by significant dead time. While conventional Q-learning-based control approaches have demonstrated effectiveness for systems without delay, their direct application to time-delay processes is difficult by the mismatch between control actions and their delayed observable effects. To address this limitation, the proposed method introduces a modified Q-learning update mechanism based on FIFO buffers that delay Q-value updates in accordance with the process dead time, ensuring proper correlation between state–action pairs and resulting system responses. Additionally, a reward policy is reformulated for the delayed update structure to support stable and convergent learning. The controller preserves key practical advantages of the original Q2d framework, including model-free operation, bumpless initialization from existing PI controller parameters, and the ability to learn online during normal operation without additionally applied external excitations. The approach is validated through simulation studies using the benchmark first-order plus dead time (FOPDT) processes with different delay times. Results demonstrate that the proposed method enables effective online performance improvement in set-point tracking and disturbance rejection over a range of time delay value and for different accuracies of the delay time estimation. Overall, the proposed modification extends the applicability of Q-learning-based control to a wider class of industrial processes with time delay, providing a practical possibility for applying reinforcement learning controllers in systems where transport delay is unavoidable.

**Keywords:** Q-learning algorithm, reinforcement learning, process control, online learning, intelligent control.

## INTRODUCTION

In industrial process control, conventional PI and PID controllers remain the dominant technology. It is estimated that over 90% of all industrial control loops rely on some form of PID-based control [1]. Despite the well-established theoretical foundations of PID control, surveys consistently report that the majority of these controllers are poorly tuned in practice [2, 3]. A significant part of their implementation operates with default or near-default tuning parameters, which results in sluggish or oscillatory closed-loop behavior that leads to increased energy consumption, material waste, and accelerated degradation of process equipment [2]. Improving the performance of these control loops without requiring detailed process knowledge or costly retuning procedures is therefore of considerable practical importance. Among the most desirable solutions is a self-improving controller capable of replacing an existing, poorly tuned PI controller and progressively enhancing its control performance through online learning during normal process operation.

Q-learning, originally proposed by Watkins [4, 5], is one of the most widely investigated model-free reinforcement learning (RL) algorithms. Over the past decade, Q-learning and its variants have been successfully applied across a remarkably broad spectrum of engineering domains. In optimization and path planning, Q-learning has been employed for multimodal transportation routing under time uncertainty [6], mobile robot local path planning [7], and energy-efficient trajectory optimization for unmanned aerial vehicles [8]. In the field of robotics and

autonomous systems, neurofuzzy reinforcement learning architectures have been developed for optimized dynamical performance of robotic platforms [9], while Q-learning-based reward shaping strategies have been applied to the swing phase control of semi-active prosthetic knee systems [10]. This wide applicability demonstrates that Q-learning provides a flexible and effective framework for sequential decision-making problems where an explicit model of the environment is either unavailable or difficult to derive.

Within the specific domain of feedback control of dynamical systems, Q-learning has attracted growing attention both from theoretical and application-oriented perspectives. On the theoretical side, output feedback Q-learning formulations for the discrete-time linear quadratic regulator problem have been developed [11], establishing important convergence and optimality guarantees. From the application viewpoint, adaptive fuzzy Q-learning controllers have been designed for grid-tied power electronic inverters [12], and experimental studies have demonstrated Q-learning-based control of airfoil trailing-edge flow separation using plasma synthetic jets [13]. Fuzzy Q-learning strategies have been applied to temperature regulation systems [14], while studies on rotary inverted pendulum systems have compared classical Q-learning with deep Q-network approaches for balancing control [15]. A particularly active research direction concerns the use of Q-learning for online tuning of PID controller parameters. In this context, Q-learning frameworks have been proposed for online PID tuning in continuous dynamic systems [16], deep Q-learning has been applied for automated PID autotuning [17], adaptive Q-learning-based PID design has been validated for maglev train levitation control [18], and Q-learning-based approaches for PI controller autotuning have been investigated [19]. Additional contributions include reinforcement learning approaches to autonomous PID tuning for chemical processes [20] and deep reinforcement learning architectures proposed for self-driving process control [21].

However, most of the works employ Q-learning either as a supervisory mechanism for tuning the parameters of a conventional PID controller or as a controller for discrete-state systems such as robotic platforms. Comparatively few studies investigate the direct application of Q-learning as a standalone controller for continuous-process closed-loop systems, where the algorithm itself generates the manipulating variable at each sampling instant without relying on an underlying PID structure. This important distinction motivates the research presented in this paper. In previous works, the concept of a self-improving Q-learning controller for industrial process automation was introduced in [22], where the state space was redefined based on a closed-loop reference trajectory and the Q-matrix was initialized using the tuning parameters of the existing PI controller to ensure bumpless switching. Implementation aspects of this approach, including reduction of the Q-matrix dimensionality from three to two dimensions (the Q2d approach), were presented in [23]. A comprehensive treatment of the Q2d controller, including its detailed design methodology, exponential state space discretization, modified exploration policy, and systematic experimental validation on an electric drive system, was published in [24]. The Q2d approach demonstrated that a model-free Q-learning controller, properly initialized by PI tuning parameters, can progressively improve closed-loop performance through online learning while maintaining bumpless operation throughout the transition from the existing PI controller.

Despite the demonstrated effectiveness of the Q2d approach, its validation was limited to processes without significant dead time in their dynamics. In industrial practice, however, a substantial number of processes are characterized by the presence of transport delay (dead time), which constitutes one of the most challenging aspects of process control. The dead time introduces a time desynchronization between the control action and its observable effect on the process output, which is in fundamental contradiction with the standard Q-learning update policy that assumes that the effect of an action taken at time $t$ is observable at the next time step $t$+1. When the dead time exceeds the sampling interval, the Q-learning controller rewards actions whose actual effect on the process output will only become apparent after the dead time has elapsed, leading to incorrect state-action pair correlations and degraded or failed learning. While several theoretical studies have addressed reinforcement learning for time-delay systems from an optimal control perspective, including learning-based adaptive optimal control formulations for linear time-delay systems [25] and investigations of control delay effects in reinforcement learning for real-time dynamical systems [26], these approaches typically

assume knowledge of system dynamics, operate in a model-based framework, or address the delay problem at the level of the learning algorithm rather than at the level of a practical, model-free controller suitable for industrial implementation. Model-based reinforcement learning methods for systems with delay time have also been proposed for continuous control problems [27], but they rely on learned environment models and are not directly applicable to the model-free, PI-initialized control framework considered in this work. Existing approaches to reinforcement learning in systems with delayed effects typically address the problem by augmenting the state space with past actions, thereby transforming the delayed process into an equivalent Markov decision process [28]. Another widely used approach relies on model-based or predictive frameworks, where the system dynamics are explicitly learned or estimated to compensate for delay effects [29]. While effective, these methods often increase the dimensionality of the problem and the computational complexity of the controller, which may limit their applicability in resource-constrained industrial environments. In contrast, the approach proposed in this work introduces a minimal modification to the standard Q-learning framework by delaying Q-value updates using a FIFO buffer, without increasing the state dimension or requiring an explicit model of the process except the knowledge about the time delay itself. This makes the method particularly suitable for implementation in industrial control systems with limited computational resources. To the best of the author's knowledge, no existing work addresses the practical extension of a model-free Q-learning controller to industrial processes with significant dead time while preserving bumpless initialization from an existing PI controller. This paper addresses this gap by proposing an extension of the previously proposed Q2d control algorithm for processes with dead time. The major contributions of this work are as follows:

1. A modification of the Q-learning update policy that introduces delayed Q-value updates using FIFO buffers synchronized with the process dead time, enabling correct correlation between control actions and their delayed effects on the process output.
2. A reward strategy specifically designed for the delayed update framework that ensures proper convergence of the Q-matrix in the presence of dead time.
3. A systematic analysis of the sensitivity of the proposed approach to the accuracy of dead time estimation, including the effects of both underestimation and overestimation.
4. Simulation-based validation demonstrating effective online learning for processes characterized by small, moderate, and significant dead times, confirming that the proposed method preserves all essential advantages of the original Q2d approach, including model-free operation and bumpless initialization from existing PI controller tunings.

## SHORT INTRODUCTION TO Q-LEARNING ALGORITHM

Q-learning is a model-free reinforcement learning algorithm that learns optimal control policy which brings the system to the goal-state, through interaction with its environment. The algorithm observes the current state $s$, selects an action $a$ that moves the system to the new state and determines the reward for applied action. The actions are selected from a set of predefined action table based on the information collected in Q-matrix, whose elements indicate what action will move the process from its current state as close as possible to the goal state. In standard approach, at the initial stage of operation the Q-learning has no knowledge about its environment and learns it by random applying actions and rewarding those of them which bring the system to the goal state. After that, actions that lead to non-goal state that is close to the goal-state are partially rewarded. This policy leads to the following formula describing the update of the Q-matrix elements:

$$Q(s_t, a_t) \leftarrow Q_p(s_t, a_t) + \alpha \left[R + \gamma \times \max_a Q(s_{t+1}, a_{t+1}) - Q_p(s_t, a_t)\right] \quad (1)$$

In Equation 1, $t$ denotes a discrete-time instant, $s$ represents the system state belonging to a predefined set of states $s \in S$, and $a$ denotes an action selected from a predefined set of actions $a \in A$. The action is chosen either as the optimal one during the exploitation phase or drawn randomly during exploration phase and is applied in the current state. The term $Q(s_t, a_t)$ denotes Q-value representing the corresponding element of the Q-matrix, which represents weights for applying an action $a$ when system is in the state $s$. Additionally, in

Equation 1 $Q_p(s_t, a_t)$ denotes the Q-value prior to the update, while $Q(s_{t+1}, a_{t+1})$ corresponds to the Q-value of the subsequent state-action pair after executing action $a_t$. The parameter $\alpha \in [0, 1]$ is the learning rate, $R$ is the reward associated with the state $s_t$, and $\gamma \in [0, 1]$ denotes the discount factor. The operator $max_a$ selects the action that yields the maximum expected reward at the next time step. For further details on Q-learning algorithm, its theoretical foundation and a comprehensive review of its applications, the reader is referred to [30].

## STATEMENT OF THE PROBLEM

In control engineering applications, one of the most promising approaches is the direct implementation of the single-agent Q-learning algorithm as a controller within a conventional single-loop closed-loop control system. In this framework, the Q-learning controller (agent) learns directly through interactions with controlled *Process* (environment), based on current state of the closed-loop system, which is defined using the control error and the dynamics of its decay.

In most industrial closed loop systems, control is usually performed by a PI controller that is often tuned conservatively. In such cases, the objective is to apply the self-improving Q-learning-based controller capable of immediate replacing the existing PI controller in the real applications. Its objective is stabilization, which aims to maintain the process output $Y$ at its desired setpoint $Y_{SP}$ (tracking) by adjusting the manipulated variable $U$ in the presence of varying load disturbance $d$. This situation is illustrated in Figure 1.

It is assumed that the existing PI controller is pre-tuned, with known tuning parameters and provides acceptable but too sluggish control performance. Because it is also assumed that any process model is unknown, the Q-learning controller must initially preserve the control performance that is very similar to the performance of the existing PI controller. While controlling the process, during the exploration phase, the Q-learning controller is expected to progressively improve its performance and achieve the desired target closed-loop behavior by learning directly from interaction with the dynamic process.

The only available information about the existing close-loop system is the PI controller tuning parameters, which should be used to initialize the Q-learning properly to ensure bumpless switching between both controllers. Additionally, it is assumed that the Q-learning controller has a predefined and discretized state space and a set of available actions. Both sets should be determined based only on the very general information about the operating conditions of the closed-loop system.

The concept presented in this work is based on the Q2d approach described in details in [24] where it was shown that the Q2d controller provides satisfying improvement in the control performance for processes without time delay. Its performance was validated both by simulation and experimentally in the application to the electric drive system. In this section, only the most important background of Q2d approach is presented to allow the reader to follow the concept proposed in this paper.

## SHORT INTRODUCTION TO Q2d CONTROLLER DESIGN

Q2d approach is dedicated to control the single-input single-output (SISO) dynamic process with control signal $U(t)$, process output $Y(t)$, desired setpoint $Y_{SP}(t)$ and control error $e(t)=Y_{SP}(t) - Y(t)$. The control objective is to keep the control error within the predefined range *(-precision, precision)*. The desired closed-loop behavior is defined by the first order trajectory:

$$\dot{e} + \left(\frac{1}{T_e}\right) e = 0 \tag{2}$$

where: $T_e > 0$ is desired closed-loop time constant and $\dot{e}$ is the control error derivative.

This formulation of the closed-loop trajectory provides an intuitive link to the existing PI controller represented as:

$$\Delta U_{PI}(t) = K_{PI} T_s \left(\dot{e}(t) + \frac{1}{T_I} e(t)\right) \tag{3}$$

where: $K_{PI}$ is the proportional gain, $T_I$ is the integral time and $T_s$ is sampling time. Based on that the control output for Q2d controller is:

$$\Delta U_Q(t) = K_Q T_s \cdot s = K_Q T_s \left(\dot{e}(t) + \frac{1}{T_e} e(t)\right) \tag{4}$$

and after assuming $K_Q = K_{PI}$ the similarities between formula (3) and (4) become clear.

In Q2d approach, the states are created by merging control error and its time derivative into a single state based on target trajectory:

$$s = \dot{e} + \frac{1}{T_e} e \tag{5}$$

This state formulation has a clear physical interpretation because $s$ quantifies the deviation from desired first-order trajectory (2). When $s = 0$, the error evolves according to $\dot{e} = -\left(\frac{1}{T_e}\right) e$ trajectory which means that the closed-loop system accurately follows the target trajectory. Positive values $s > 0$ show that the error is decreasing too slowly and when $s < 0$, the situation is opposite. State vector S accommodates both negative and positive values and represents the discrete state space as:

$$[-s_N, -s_{N-1}, \ldots, -s_1, s_1, \ldots, s_N] \tag{6}$$

where: $N$ is the total number of states, typically even number to include symmetrical states on both sides of $s = 0$.

To provide efficient scaling and desired precision application, in Q2d approach it is proposed to apply the exponential distribution of control error, which ensures that the accuracy of the state distinction increases as the absolute value of the control error decreases. Based on this approach, two vectors are generated: one for the discretized control error and one for its discretized time derivative. Both are generated, assuming maximal possible error value $e_{max}$ and the *precision* determining accuracy of reaching the goal state. Next, based on these two vectors, the vector representing the boundaries of the states intervals $S$ in the form (6) is computed. Finally, the vector of the mean values of two neighboring state boundaries is generated and this vector is directly used for generating the vector of the accessible Q2d controller actions $A$:

$$A = \begin{bmatrix} -a_N, -a_{N-1}, \ldots, -a_2, a_1 \\ = 0, a_2, \ldots, a_{N-1}, a_N \end{bmatrix} \tag{7}$$

where: $a_i = K_Q \cdot T_s \cdot s_i^m$ and $s_i^m = \frac{s_{i-1} + s_i}{2}$ denotes the mean value of two neighboring state boundaries. It ensures that these actions are identical to those that the PI controller would take in the same state computed by Equation 3, with the accuracy of discretization.

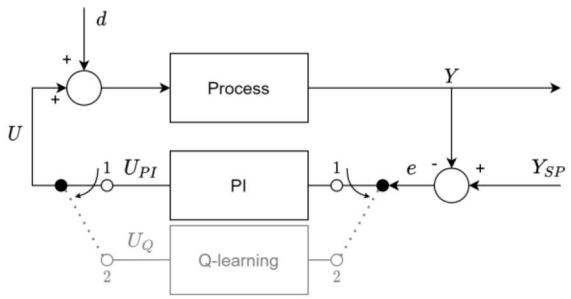


**Figure 1.** Schematic diagram of the closed-loop control system with PI initialization ad subsequent Q-learning-based control

By assumption, in Q2d approach the reference trajectory defined by $T_e$ differs significantly from the closed-loop trajectory ensured by the existing PI controller with its integral time $T_I$. This fact requires application of the projection function that ensures the simple Q2d initialization providing bumpless switching between the existing PI controller and the designed Q2d controller. For this purpose, it is assumed that $K_{PI} = K_Q$ and combining Equations 3 and 4 leads to:

$$\Delta U_Q = K_Q T_s \left(\dot{e} + \frac{1}{T_e} e\right) - e \left(\frac{1}{T_e} - \frac{1}{T_I}\right) \tag{8}$$

which finally allows to define Q2d control law modified by the projection function:

$$\Delta U_Q = a - e\left(\frac{1}{T_e} - \frac{1}{T_I}\right) \tag{9}$$

where: $a$ is the action selected for current state $s$. From practical reasons, the control signal must be also saturated as $U_Q \epsilon [U_{min}, U_{max}]$.

Such an approach to generating state and action vectors results in the 2-dimentional Q-matrix (2$N$-1) × (2$N$-1) that stores the correlation between the current systems state and the corresponding so-far optimal action. At the same time, due to similarities of the discretized state-action space to the PI controller, the initialization of Q2d controller is very easy. Adjusting Q matrix as the identity one ensures that after switching between the existing PI controller and its substitute (Q2d controller), the switching is practically bumpless because Q2d controller performs practically the same as the existing PI controller.

To ensure learning abilities of Q2d controller, its exploration/exploitation problem is addressed

using the modified ε-greedy method [24]. In conventional Q-learning approach, the exploration phase requires an unrestricted range of randomized actions. However, this is unacceptable in industrial closed-loop systems where learning must be carried out during normal operation and thus the closed loop system cannot be disturbed by wide variety of randomized action. In Q2d approach the range of drawing actions is limited by introducing the parameter *RD* that directly limits the number of drawn actions around the optimal action represented by the highest value in the Q-matrix. This approach reduces the disturbances intentionally applied to the closed loop system only by drawing actions whose values significantly differ from the current optimal action.

In addition, the action randomness was additionally limited by limiting the permissible range of randomly selected actions to only those whose sign matches the sign of the action with the highest Q-value one. This prevents changing $\Delta U$ sign near goal state which can lead to oscillatory behavior near goal state.

## MODIFICATION OF Q2d APPROACH FOR PROCESSES WITH DEAD TIME

Q2d was validated by simulation and experimentally with the assumption that the control process has no dead time in its dynamics [24]. In this section it is proposed how to extend the Q2d approach to the processes with a significant dead time.

### Difficulties with process dead time

The presence of dead time $T_0$ in the process dynamics poses a fundamental challenge for Q2d controller because the basic Q-learning policy (1) assumes that the direct effect of action $a_t$ is observable in the next state $s_{t+1}$, When dead time exceeds the single sampling interval ($T_0 >> T_s$), the control action $\Delta U_Q$ affects the measured process output at time ($t + T_0$). This fact provides a significant difficulty with reward policy because in the presence of process dead time, Q2d controller rewards the actions whose effect on the process output will be seen in the future after dead time $T_0$. Without proper modifications in the reward policy, Q2d controller is unable to ensure proper update of the state-action pairs which leads to poor or even failed learning abilities. Such a modification is proposed in the next section.

### Dead time compensation for Q2d approach

For Q2d approach, the major challenge with the presence of the process dead time $T_0$. is to ensure that Q-value update policy can reward the correlated state-action pairs because the effect of the action $a_t$ taken at time $t$ becomes visible at time ($t + T_0$).

To deal with this difficulty, at this stage it is assumed the dead time $T_0$ is known. Such approaches have been investigated in literature, for example through practical time-delay estimation techniques designed for industrial implementation [31]. Then it is proposed to use the FIFO buffers that store state-action pairs and thus delay the Q-value updates until the effect of the applied action is visible. The buffer size is determined based on the knowledge of dead time:

$$N_{buffer} = trunc\left(\frac{T_0}{T_s}\right) \tag{10}$$

and the application of this buffer results in the modification of the Q-matrix update policy:

$$\begin{aligned} &Q\left(s_{t-N_{buffer}}, a_{t-N_{buffer}}\right) \\ &\leftarrow Q_p\left(s_{t-N_{buffer}}, a_{t-N_{buffer}}\right) + \\ &+\alpha\begin{bmatrix} R + \gamma \times \max_a Q(s_{t+1}, a_{t+1}) \\ -Q_p\left(s_{t-N_{buffer}}, a_{t-N_{buffer}}\right) \end{bmatrix} \end{aligned} \tag{11}$$

which ensures that the properly delayed state/action pair is rewarded.

Another important issue is the proper choice of reward function because it significantly influences convergence of the Q-matrix and the online learning process. Thus, the reward strategy is defined as follows:

$$R = \begin{cases} 1 \; if \; s_{t-N_{buffer}} = \\ = \; s_{goal} \; and \; a_{t-N_{buffer}} = a_{goal} \\ 0 \; otherwise \end{cases} \tag{12}$$

to ensure the reward is applied only if after the dead time the closed-loop system reaches the goal state $s_{goal} = 0$ and reaching this state is the result of taking the action $a = a_{goal} = 0$ which is equivalent to that the control signal remains unchanged.

This reward policy ensures that:

- Direct reward is applied only when the closed-loop system is at the goal state; the value $Q(s_{goal}, a_{goal})$ is always the highest value in the whole Q-matrix, no other state-action pair can achieve the same maximal value.
- Temporal-difference learning is applied; when the Q-value $Q(s_{goal}, a_{goal})$ is rewarded, it is propagated to all other Q-values by the bootstrap term $(\gamma \times \max_a Q(s_{t+1}, a_{t+1}))$ existing in (11). States from which the closed-loop system is most often transferred to states for which Q-values are large, themselves also obtain large Q-values by backward propagation through the state space.
- It is preferred that the closed-loop system remains in the goal state; because the direct reward is applied only when the closed-loop system is in the goal state, the Q-learning controller prefers to maintain the closed-loop closed-loop system in its goal state (at the target trajectory) rather than maintaining it in off-trajectory states.
- Because the reward is applied only when the action $a = a_{goal} = 0$ was applied, it prevents the Q-learning controller from selecting non-zero action in the goal state, which would unnecessarily disturb the closed-loop system.
- With progress in the on-line learning, the closed loop system will increasingly be in states closer to the target state. Consequently, the states corresponding to the fact that the system deviates more significantly from the target state obtain lower Q-values based on the expected cumulative discounted reward for following the optimal trajectory towards the goal state. This creates the gradient distribution in the Q-matrix that forces selecting the actions leading toward the goal state.

## VALIDATION AND RESULTS

Validation of the proposed Q2d controller with the proposed dead time compensation was done through simulation using the FOPDT dynamics representing many industrial processes, such as simple thermal processes, flow processes or hydraulic processes where the dynamics is characterized by a single time constant $T$ and the dead time $T_0$

$$G_1(s) = \frac{1}{sT + 1} e^{-T_0 s} \qquad (13)$$

For the process (13), three cases are considered:

- $T = 5$, $T_0 = 0.5$, strictly dominating first order dynamics with small but significant dead time
- $T = 5$, $T_0 = 2$, first order dynamics with a moderate dead time
- $T = 5$, $T_0 = 4$, first order dynamics with a more significant dead time

Because in practical cases the estimation of the real value of the dead time can be inaccurate, for each case, three different scenarios are considered representing different accuracies on the knowledge on the process dead time (the proposed dead time compensation was applied using the estimated values $T_{0,controller}$):

- underestimation, $T_{0,controller} = 0.8\ T_0$,
- accurate estimation, $T_{0,controller} = T_0$,
- overestimation, $T_{0,controller} = 1.2\ T_0$.

The simulation experiments for each case are carried out with the same scenario. Q2d controller always starts with the initialization ensuring its initial performance equivalent to the performance of PI controller with Siemens PLC default tunings $K_{PI} = 1$, $T_I = 20$. They represent PI tunings that even for the considered FOPDT processes provide conservative closed-loop performance. Thus, there is significant potential for the improvement of this performance due to on-line learning abilities of Q2d controller.

All experiments were conducted in MATLAB/Simulink environment with sampling time $T_s = 0.1$ s and Q2d parameters adjusted as shown in Table 1.

In each case, the on-line learning was performed by successive disturbing the closed loop system by applying the randomly drawn load

**Table 1.** List of Q2d parameters used in verification

| Parameter | Symbol | Value |
|---|---|---|
| Controller gain | $K_Q$ | 1 (=KPI) |
| Goal trajectory time constant | $T_e$ | 5s |
| PI Integral time | $T_I$ | 20 s |
| Learning rate | $\alpha$ | 0.1 |
| Discount factor | $\gamma$ | 0.99 |
| Exploration rate | $\in$ | 0.3 |
| Random deviation | RD | 3 |
| Precision | precision | 0.1% |
| Number of states | N | 50 |
| Control limits | $[U_{min}, U_{max}]$ | [0%, 100%] |
| Set point | $Y_{SP}$ | 50% |

disturbance step changes of the normal distribution with $\sigma_d = \frac{0.5}{3}$ (three sigma rule) and $\mu = 0$ amplitude. Time for which the randomly selected disturbance is kept at a constant value was also randomized using normal distribution $\sigma_N = 3000$ and $\mu = 3000$. It represents the realistic situation when the real process is disturbed by external disturbances appearing in a natural way and when not every disturbance can be fully compensated before the new disturbing signal is randomized and applied. The applied normal distribution represents the most common industrial disturbances and ensures consistency of the obtained results, all drawings were carried out using the same random number generator.

Figures 2–4 show the results of the on-line learning of the proposed Q2d controller with a dead time compensation for three considered FOPDT processes (for three different dead time values $T_0 = 0.5$, $T_0 = 2$ and $T_0 = 4$) with three different cases representing accuracy of dead time estimation. The results are presented using the verification experiments performed after 3000 epochs of learning (3000 generated disturbances) for Q2d controller switched into the exploitation mode. The number of learning epochs was selected based on preliminary studies, which indicated that further learning beyond this point does not yield significant performance improvement. Each experiment consists of the step change of the set-point value (from 20% to 50%) and consecutive step change of the load disturbance $d = -0.3$ at t = 200, both applied to the closed loop system.

Apart from visual validation, Table 2 summarizes the quantitative performance indicators obtained for the presented results. These indicators include:

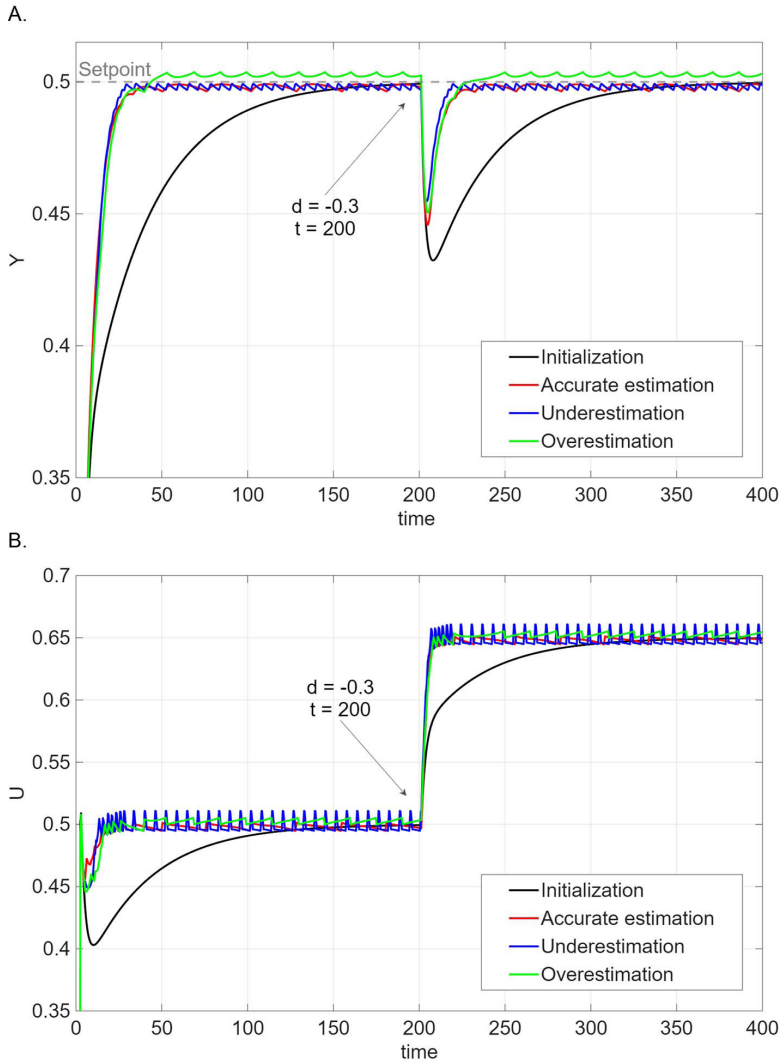


**Figure 2.** Closed-loop responses after 3000 learning epochs ($T_0 = 0.5$), (A) output $Y$, (B) control signal $U$, for different dead-time estimation scenarios

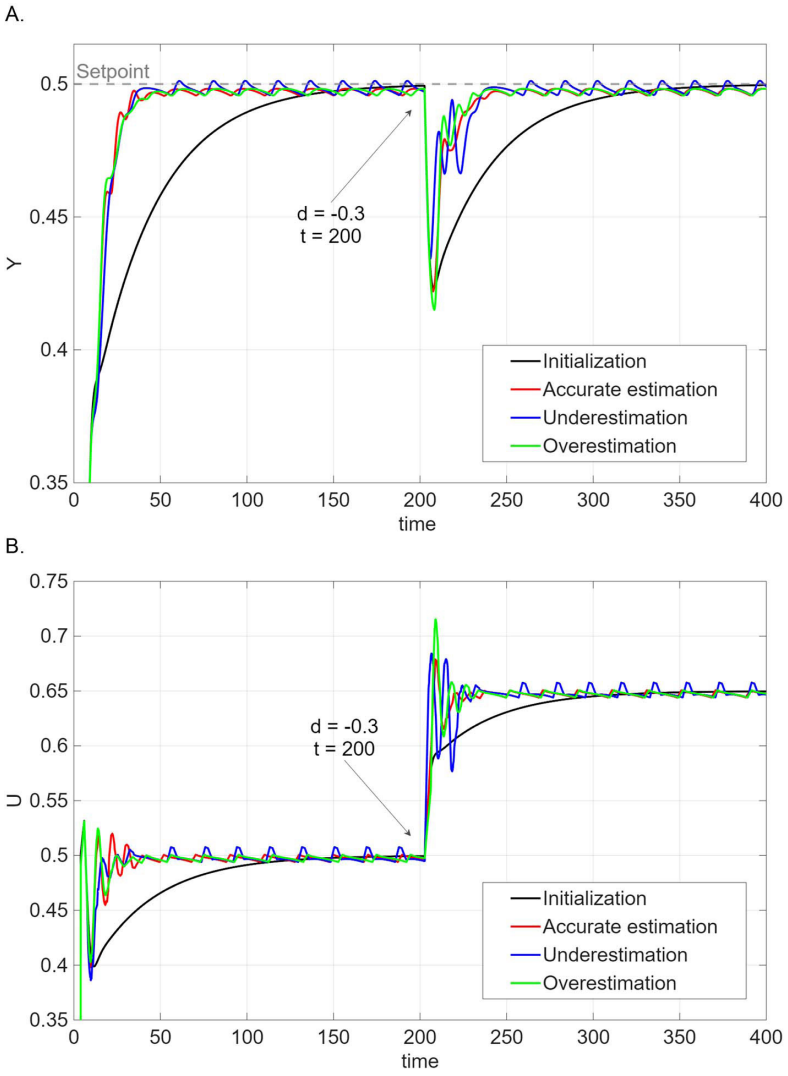


**Figure 3.** Closed-loop responses after 3000 learning epochs ($T_0$ = 2), (A) output $Y$, (B) control signal $U$, for different dead-time estimation scenarios

- Integral absolute error (IAE): $\int_0^T |e(t)|dt$
- Maximum overshoot: $\max_t |Y(t) - Y_{SP}|$
- Maximum control increment (Max u): $\max_t |\Delta U_Q(t)|$

For the smallest dead time, $T_0$ = 0.5, all compensation strategies lead to a significant improvement compared to the closed-loop performance obtained for initialization. However, accurate estimation consistently provides the most balanced performance. For tracking, it ensures the lowest IAE (268.5) with a very small overshoot and control effort comparable to the case representing initialization. Although underestimation yields a slightly lower IAE for disturbance rejection, this improvement is obtained by the price of more significant oscillations in the manipulating variable, as visible in Figure 2(B). Accurate estimation, in contrast, provides a strong reduction in IAE value (83.6) while preserving smoother control action, indicating a better trade-off between performance improvement and control aggressivity. However, in each case of dead time uncertainty, the closed-loop is improved compared to initialization case and acceptable from practical viewpoint if small chattering in control action is acceptable.

For moderate dead time $T_0$ = 2, the advantages of accurate estimation become more apparent. While all learning-based strategies reduce the IAE for $Y_{SP}$ changes to a similar level, accurate estimation ensures negligible overshoot and moderate control effort. In disturbance rejection, underestimation provides the lowest IAE, however, this is achieved at the cost of increased control signal oscillations and higher values of maximum

control increment. Overestimation further amplifies these effects, leading to increased overshoot. Accurate estimation offers more acceptable closed-loop performance, providing substantial improvement over the initialization case while avoiding excessive oscillations, as confirmed by the smoother trajectories in Figure 3.

For the largest dead time $T_0 = 4$ and the most significant nominal differences between over and underestimation of dead time, the difficulties resulting from an improper dead time compensation become clearly visible. Underestimation leads to a significant degradation of control performance, with IAE exceeding even the initialization case. Overestimation improves disturbance-related IAE but introduces noticeable oscillatory behavior in both the controlled and manipulating variables. Although the accurate dead time estimation does not guarantee the minimal IAE, it provides the most acceptable closed loop performance preserving moderate overshoot and avoiding the pronounced oscillations observed for the inaccurate dead time estimation cases. This behavior is clearly illustrated in Figure 4, where accurate estimation provides the smoothest closed-loop response among the depicted cases.

Overall, the results demonstrate the expected effect that accurate dead-time estimation yields the most robust and reliable on-line learning compared to the considered inaccuracies in the dead-time estimation. However, the results also show that the proposed approach is robust to slight uncertainty on the dead time estimated value, but this robustness decreases with the increase of dead time in the real process.

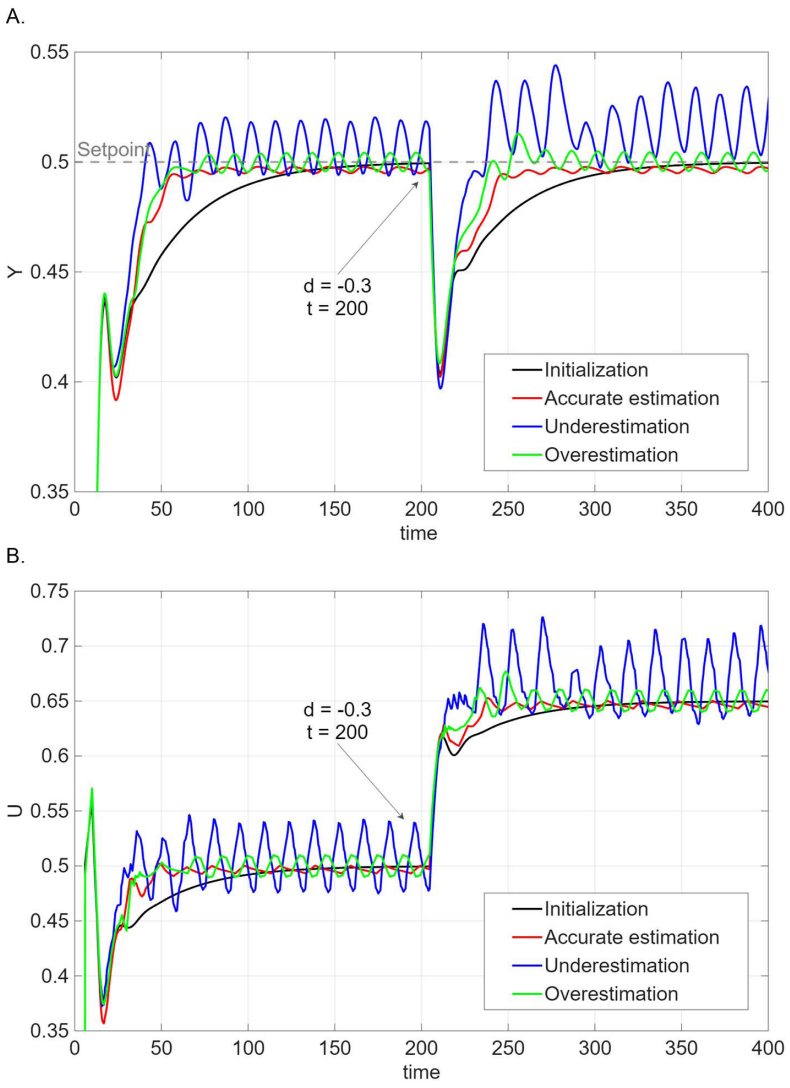


**Figure 4.** Closed-loop responses after 3000 learning epochs ($T_0 = 4$), (A) output $Y$, (B) control signal $U$, for different dead-time estimation scenarios

**Table 2.** Quality indicators comparison for initialization and after 3000 epochs of learning for different FOPDT processes and dead time value estimations

| Scenario | Metric | Initialization | Accurate est. | Underest. | Overest |
|---|---|---|---|---|---|
| T0=0.5 | | | | | |
| SP change | IAE | 634.9 | **268.5** | 271.5 | 294.8 |
| | Max u | 30.2 | 29.2 | 29.2 | 29.2 |
| | Max overshoot | **0.0** | **0.0** | **0.0** | 0.4 |
| Disturbance | IAE | 301.1 | 83.6 | **71.3** | 89.3 |
| | Max u | 0.3 | 0.4 | 0.4 | 0.4 |
| | Max overshoot | 6.8 | 5.4 | **4.5** | 5.0 |
| T0=2 | | | | | |
| SP change | IAE | 635.2 | **340.3** | 344.1 | 343.0 |
| | Max u | 30.2 | 29.2 | 29.2 | 29.2 |
| | Max overshoot | **0.0** | **0.0** | 0.1 | 0.0 |
| Disturbance | IAE | 301.0 | 135.4 | **113.6** | 125.6 |
| | Max u | 0.3 | 0.5 | 0.8 | 0.8 |
| | Max overshoot | 7.7 | 7.8 | **6.6** | 8.5 |
| T0=4 | | | | | |
| SP change | IAE | 635.6 | 526.7 | 547.9 | **490.4** |
| | Max u | 30.2 | 29.2 | 29.2 | 29.2 |
| | Max overshoot | **0.0** | **0.0** | 2.0 | 0.4 |
| Disturbance | IAE | 300.8 | 237.5 | 437.2 | **198.7** |
| | Max u | 0.3 | 0.3 | 0.3 | 0.3 |
| | Max overshoot | 9.6 | 9.8 | 10.3 | **9.2** |

## CONCLUSIONS

Based on the general Q-learning framework, this paper presents an extension of the Q2d control algorithm for processes with significant dead time. The primary contribution consists in modifying the standard Q-learning reward policy by introducing delayed Q-value updates using FIFO buffers synchronized with the process dead time. This modification enables correct correlation between applied control actions and their delayed effects on the process output, which is a fundamental requirement for successful learning in systems with transport delay.

The proposed solution preserves all essential advantages of the original Q2d approach. In particular, the controller does not require any knowledge of the mathematical model of the controlled process except the estimation of its dead time, and the control objective is defined by a reference trajectory formulated in terms of the control error and its time derivative decay. The Q2d initialization ensures practically bumpless switching from an existing PI controller, using only the PI tuning parameters and closed-loop sampling time. As a result, the extended Q2d method preserves the assumptions of being a self-improving controller also in the presence of dead time.

Simulation results confirm that the proposed method of compensation of dead time enables effective online learning for processes characterized by small, moderate, and significant dead time. For all investigated scenarios, the Q2d controller initialized from PI tuning preserves the initial closed-loop performance both for set-point tracking and disturbance rejection. After switching to exploration mode, the controller progressively improves performance as quantified by the reduction of IAE and, in most cases, by reducing overshoot and improving disturbance rejection. The results also demonstrate that the efficiency of on-line learning strongly depends on the accuracy of the estimation of the process dead time. For small nominal differences there are insignificant differences in closed-loop performance after learning but as the absolute difference increases, the effectiveness of online learning becomes increasingly limited.

The obtained results indicate that the proposed approach exhibits robustness to moderate inaccuracies in dead-time estimation (at the level of ±20%), particularly for processes with small and moderate delay. This suggests that in practical applications the proposed approach is resistant to small variations of dead time.

For small and moderate dead times, the learning process leads to substantial closed loop performance improvements compared to the initialization case. For larger dead times, as the plant dynamics become more challenging, improvement is still visible but at the cost of increased oscillations and chattering of the control signal becomes increasingly evident.

An important advantage of the proposed method is that learning still can be performed during normal closed-loop operation and in practice, it does not require any external excitation signals. The controller can learn from naturally occurring disturbances, which makes the approach more suitable for industrial systems where intentional perturbation of the process is often unacceptable. At the same time, the restriction of the exploration range through the *RD* parameter ensures that online learning does not introduce excessive changes in the control action required for efficient learning.

From the implementation point of view, the proposed modification does not significantly increase computational complexity or memory requirements compared to the original Q2d controller. The only additional element is the FIFO buffer used for delayed updates, which requires storing a limited number of past samples. Therefore, the method remains suitable for real-time implementation on industrial control hardware.

The obtained results demonstrate that the Q2d approach can be successfully extended to processes with dead time while preserving its computational efficiency, partial model-free character, and bumpless initialization from existing PI controllers. The proposed modification enables practical application of reinforcement learning-based control for a broader class of industrial processes where time delay is unavoidable.

### Acknowledgements

This work was supported by the grant from SUT – subsidy for maintaining and developing the research potential in 2026.

## REFERENCES

1. Jelali M. An overview of control performance assessment technology and industrial applications. Control Eng Pract. 2006; 14(5): 441–466.
2. Van Overschee P, De Moor B. RaPID: the end of heuristic PID tuning. IFAC Proc Vol. 2000; 33(4): 595–600.
3. Jelali M. Control performance management in industrial automation. Berlin: Springer; 2013.
4. Watkins CJC. Learning from delayed rewards [PhD dissertation]. Cambridge: University of Cambridge; 1989.
5. Watkins CJC, Dayan P. Q-learning. Mach Learn. 1992; 8: 279–292.
6. Zhang T, Cheng J, Zou Y. Multimodal transportation routing optimization based on multi-objective Q-learning under time uncertainty. Complex Intell Syst. 2024; 10: 3133–3152. https://doi.org/10.1007/s40747-023-01308-9
7. Zhou Q, Lian Y, Wu J, Zhu M, Wang H, Cao J. An optimized Q-learning algorithm for mobile robot local path planning. Knowl Based Syst. 2024; 286: 111400. https://doi.org/10.1016/j.knosys.2024.111400
8. Saeed RA, Ali ES, Abdelhaq M, Alsaqour R, Ahmed FRA, Saad AME. Energy efficient path planning scheme for unmanned aerial vehicle using hybrid genetic algorithm-based Q-learning optimization. IEEE Access. 2024; 12: 1170–1184. https://doi.org/10.1109/ACCESS.2023.3344455
9. Abouheaf M, Gueaieb W. Neurofuzzy reinforcement learning control schemes for optimized dynamical performance. In: Proceedings of the IEEE International Symposium on Robotic and Sensors Environments (ROSE); 2019; Ottawa, Canada. p. 1–7. https://doi.org/10.1109/ROSE.2019.8790424
10. Hutabarat Y, Ekkachai K, Hayashibe M, Kongprawechnon W. Reinforcement Q-learning control with reward shaping function for swing phase control in a semi-active prosthetic knee. Front Neurorobot. 2020; 14: 565702. https://doi.org/10.3389/fnbot.2020.565702
11. Rizvi SAA, Lin Z. Output feedback Q-learning control for the discrete-time linear quadratic regulator problem. IEEE Trans Neural Netw Learn Syst. 2019; 30(5): 1523–1536. https://doi.org/10.1109/TNNLS.2018.2870075
12. Gheisarnejad M, Sharifzadeh M, Khooban MH, Al-Haddad K. Adaptive fuzzy Q-learning control design and application to grid-tied nine-level packed E-cell inverter. IEEE Trans Ind Electron. 2023; 70(3): 2721–2731. https://doi.org/10.1109/TIE.2022.3153803
13. Zong H, Wu Y, Liang H, Su Z, Li J. Experimental study on Q-learning control of airfoil trailing-edge flow separation using plasma synthetic jets. Phys Fluids. 2024; 36: 025175. https://doi.org/10.1063/5.0185853

14. Chen YC, Hung LC, Syamsudin M. Fuzzy Q-learning control for temperature systems. In: Proceedings of the IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD); 2021; Taichung, Taiwan. p. 184–188. https://doi.org/10.1109/SNPD51163.2021.9704994
15. Ben Hazem Z. Study of Q-learning and deep Q-network learning control for a rotary inverted pendulum system. Discov Appl Sci. 2024; 6: 56. https://doi.org/10.1007/s42452-024-05690-y
16. Ibarra-Perez D, Garcia-Nieto S, Sanchis Saez J. Q-learning for online PID controller tuning in continuous dynamic systems: an interpretable framework for exploring multi-agent systems. Mathematics. 2025; 13(21): 3461. https://doi.org/10.3390/math13213461
17. Raj M, Sharma S, Gupta G, Yadav S. Auto tuning PID using deep Q-learning. In: Proceedings of the International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC); 2025; Bhubaneswar, India. p. 1–5. https://doi.org/10.1109/ASSIC64892.2025.11158657
18. Shou B, Zhang H, Long Z, Xie Y, Zhang K, Gu Q. Design and applications of Q-learning adaptive PID algorithm for maglev train levitation control system. In: Proceedings of the Chinese Control and Decision Conference (CCDC); 2023; Yichang, China. p. 1947–1953. https://doi.org/10.1109/CCDC58219.2023.10326815
19. Brzeziński S, Stebel K. Q-learning algorithm for PI controller autotuning. Prz Elektrotechniczny. 2025; 101(5): 44–48. https://doi.org/10.15199/48.2025.05.10
20. Dogru O, Velswamy K, Huang B. Reinforcement learning approach to autonomous PID tuning. Comput Chem Eng. 2022; 161: 107760. https://doi.org/10.1016/j.compchemeng.2022.107760
21. Spielberg S, Tulsyan A, Lawrence NP, Loewen PD, Gopaluni RB. Toward self-driving processes: a deep reinforcement learning approach to control. AIChE J. 2019; 65(10): e16689. https://doi.org/10.1002/aic.16689
22. Musial J, Stebel K, Czeczot J. Self-improving Q-learning based controller for a class of dynamical processes. Arch Control Sci. 2021; 31(3): 527–555. https://doi.org/10.24425/acs.2021.138691
23. Musial J, Stebel K, Czeczot J. Implementation aspects of Q-learning controller for a class of dynamical processes. In: Proceedings of the International Conference on Methods and Models in Automation and Robotics (MMAR); 2022; Międzyzdroje, Poland. p. 319–324. https://doi.org/10.1109/MMAR55195.2022.9874270
24. Musial J, Stebel K, Czeczot J, Nowak P, Gabrys B. Application of self-improving Q-learning controller for a class of dynamical processes: implementation aspects. Appl Soft Comput. 2024; 152: 111250. https://doi.org/10.1016/j.asoc.2024.111250
25. Cui L, Pang B, Jiang ZP. Learning-based adaptive optimal control of linear time-delay systems: a policy iteration approach. IEEE Trans Autom Control. 2024; 69(1): 629–636. https://doi.org/10.1109/TAC.2023.3273786
26. Schuitema E, Busoniu L, Babuska R, Jonker P. Control delay in reinforcement learning for real-time dynamic systems: a memoryless approach. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2010; Taipei, Taiwan. p. 3226–3231. https://doi.org/10.1109/IROS.2010.5650345
27. Chen B, Xu M, Li L, Zhao D. Delay-aware model-based reinforcement learning for continuous control. Neurocomputing. 2021; 450: 119–128. https://doi.org/10.1016/j.neucom.2021.04.015
28. Agarwal M, Aggarwal V. Blind decision making: reinforcement learning with delayed observations. Pattern Recognit Lett. 2021;150:176–182. https://doi.org/10.1016/j.patrec.2021.06.022
29. Chen B, Xu M, Li L, Zhao D. Delay-aware model-based reinforcement learning for continuous control. Neurocomputing. 2021; 450: 119–128. https://doi.org/10.1016/j.neucom.2021.04.015
30. Jang B, Kim M, Harerimana G, Kim JW. Q-learning algorithms: a comprehensive classification and applications. IEEE Access. 2019; 7: 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229
31. Grelewicz P, Nowak P, Czeczot J, Musial J. Increment count method and its PLC-based implementation for autotuning of reduced-order ADRC with Smith predictor. IEEE Trans Ind Electron. 2021; 68(12): 12554–12564. https://doi.org/10.1109/TIE.2020.3045696