Computers and Chemical Engineering 140 (2020) 106909

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Autonomous pH control by reinforcement learning for electroplating industry wastewater

Douglas Alves Goulart*, Renato Dutra Pereira

Department of Chemical Engineering, Federal University of Rio Grande, Rio Grande, RS 96203-900, Brazil.

ARTICLE INFO

Article history:
Received 16 October 2019
Revised 1 May 2020
Accepted 3 May 2020
Available online 23 May 2020

Keywords:
Machine learning
Actor-Critic
Metaheuristic optimization
Hyperparameters
Cloud computing

ABSTRACT

The electroplating industry, due to steps such as pickling, generates acid pH wastewater. Its treatment is important for environmental preservation and the future recovery of metals. Therefore, the main objective of this work was the development of an autonomous pH controller for electroplating industry liquid effluents, based on fully automated Reinforcement Learning (RL). In order to do that, a Continuous Stirred-Tank Reactor (CSTR) neutralization simulator, and an adapted Particle Swarm Optimization (PSO) algorithm to automate the choice of RL hyperparameters were developed. The controller was developed and validated when it stabilized the effluent's pH in a neutral range in different scenarios during the regulatory and servo operations better than a Proportional Integral Derivative (PID) controller. The development of autonomous wastewater pH control systems in coated surface treatment units is a significant advancement, as it reduces human intervention and allows the monitoring of variability associated with the electroplating industry.

1. Introduction

Automation and control systems present a competitive differential in the current market context, considering several initiatives to encourage intelligent automation and the dissemination of Industry 4.0 paradigms in the industrial sector. Automatic pH control is critical in chemical, food and bioprocess industries, where many of the process steps require tools for automation of hydrogenionic concentration control, such as wastewater treatment stage (Shinskey, 1973).

Metals and chemical reagents are intrinsic to treatment and preparation processes of metal surfaces for nickel and other metals electroplating. The use of these components as pickling agents produces acidic liquid effluents. In these, pH control is crucial, not only due to damage in future contact with the environment after disposal, but also due to the recovery of the metals used in the coating (Cushnie Jr, 1985).

Industrial surface treatment units for metallic coating generate extremely toxic effluents, with different compositions depending on the production schedule and requirement of human monitoring and supervision for treatment (Chen et al., 2007). The development of autonomous wastewater pH control systems in these units is a significant advantage, as it reduces human intervention and allows monitoring the variability associated with the electroplating industry.

As mentioned by Krishnapura and Jutan (2000), the control of a reactor's pH system is often considered to be one of the benchmarks for nonlinear process control. This is due to the highly nonlinear behavior exhibited by the pH dynamics, making it a great case study for evaluating a controller.

The main objective of this work was the application of a Machine Learning technique for autonomous pH control, based on completely automated Reinforcement Learning (RL).

The secondary purposes were to build a numerical simulator of an electroplating industry wastewater neutralization reactor for the control strategy testing purposes and obtain the required data for training the model; adapt the Particle Swarm Optimization - PSO - algorithm to optimize of RL hyperparameters. In addition, our goals included - validating the autonomy of the controller based on RL, analyzing its behavior against process variations and comparing the performance with the classic Proportional Integral Derivative (PID) controller.

Abbreviations: RL, Reinforcement Learning; PSO, Particle Swarm Optimization; CSTR, Continuous Stirred-Tank Reactor; PID, Proportional Integral Derivative; ADP, Adaptive Dynamic Programming; pH, power of Hydrogen; HPO, Hyperparameter Optimization; PBT, Population-based Training; GCP, Generalized Predictive Control.

* Corresponding author.
E-mail address: douglasagoulart@gmail.com (D. Alves Goulart).

https://doi.org/10.1016/j.compchemeng.2020.106909
0098-1354/

2. Preliminaries

2.1. pH Control and acid wastewater of electroplating industry

Electroplating is considered one of the main emitters of harmful and toxic wastewaters among the chemical industries by Hosseini et al. (2016) and Bojic et al. (2007). To further elucidate how an electroplating unit can be a source of pollution, Kobya et al. (2017) classifies the electrolytic bath, washing and rinsing of the electroplated articles as the most wastewater intensive processes within the plant. The waste streams generated in these stages, waste pickle liquor and bleed streams, are of major concern due to the presence of high metals and acid concentrations, which renders them as highly corrosive and polluting.

Regarding the generation of waste pickle liquor in electroplating industries, Robson (1993) comments that during the heat treatment of various kinds of metal surfaces, for example, steel, atmospheric oxygen reacts with iron on the surface of the steel to form a crust that is made up of a mixture of iron oxides. The presence of oxide on the surface of the steel and other metal surfaces is undesirable when it is to be subsequently shaped or cold-rolled and coated.

The removal of oxide from metal surfaces by cleaning it with an acid solution is called “pickling” and is one of the main steps in the metal treatment industries. This is usually accomplished by immersing the metal in an acid bath (Kerney, 1994, Kerney, 1997). HCl, HNO_3 or H_2SO_4 are used for this purpose as pickle liquor to remove surface oxide (Kleingarn, 1988).

According to Agrawal and Sahu (2009), the acid pickling method is preferred over all other methods to remove oxide during electroplating, like abrasive blasting, tumbling, brushing and alkaline descaling because of following reasons:

- It increases service life, eliminates irregular conditions and promotes surface smoothness in the finished products.
- It allows proper alloying or adherence of metallic coatings and satisfactory adherence when a non-metallic coating or paint is used.
- It also prevents lack of uniformity and eliminates surface irregularities during the cold reduction of steel sheet and strip.

Due to the harmful effluents of the electroplating industry resulting from the pickling process, which is preferred over other alternatives, added to the variability of the industry and the variation in production demand, an autonomous pH controller will be very useful.

2.2. Automated reinforcement learning controller

The task of controlling the pH neutralization of effluents from the electroplating industry becomes difficult due to nonlinearity, time-varying characteristics, and non-steady operation. To overcome these issues and difficulties, several advanced data-driven control techniques have been analyzed for similar processes [13:21].

Data-driven control techniques rely on data collected from the process to learn and tune controllers (Hou and Wang, 2013). This makes these techniques more representative of the real process since it eliminates incompatibilities between a model and the real case. The data-driven controller learning objective can be achieved either by using highly adaptive simplified phenomenological models (Fliess and Join, 2013, Hou and Jin, 2011) or by using no model at all. Among the model-free methods, RL has received academic attention [25:34].

As reported by Radac and Precup (2019), Reinforcement Learning (Sutton and Barto, 1998) is a data-driven technique that solves optimal control problems, and is has been developed simultaneously in the machine learning and control systems communities. In latter, RL is better known as Adaptive (Approximate) Dynamic Programming (ADP) (Werbos, 1992) or neuro-dynamic programming (Bertsekas and Tsitsiklis, 1995).

The RL algorithm requires numerous pre-defined hyperparameters, which are responsible for ensuring the convergence speed and reproducibility of the learning process (Henderson et al., 2018). As they are not automatically adjusted during training, the user must select them according to his/her experience. The result of the learning process, interaction between the algorithm and the environment, as well as the required learning time strongly depends on this choice (Islam et al., 2017).

A common method is a manual search for suitable parameters. Some previous works implemented hyperparameter adaptation in RL by simple rules or manually defined schedules [41:44]. But this kind of adaptation often requires heuristic knowledge and previous experience with the algorithm. Hyperparameter Optimization (HPO), is an extremely important field in machine learning and optimization (Candelieri et al., Probst et al.). HPO is a difficult optimization problem with high computational cost (Bengio, 2012).

The introduction of an automated hyperparameter search process, according to (Liessner et al., 2019), has the greatest benefit of facilitating the industrial application of data-based control strategies, as the user does not depend on personal experience in adjustment of hyperparameters. It was also needed to make sure that the control performance obtained by these methods is the optimal response that only the best hyperparameters allow.

With the aim of making the control of chemical processes via Reinforcement Learning simpler to be applied industrially, the authors, despite being aware of the state of the art of Reinforcement Learning technology (Shin et al., 2019), see the use of Deep Reinforcement Learning as unnecessary in the studied case, since the process to be controlled is Single Input – Single Output, thus a simpler methodology, such as Actor-Critic, was chosen.

2.3. PSO for hyperparameters optimization

The work of Jaderberg et al. (2017) proposed a Population-based Training (PBT), a simple population-based hyperparameter adaptation neural network training method. Different from simple rules or heuristics, PBT adapts the hyperparameters in the whole reinforcement learning process without human intervention.

Specifically, Lorenzo et al. (2017) and Lorenzo et al. (2017) did an extensive experimental study that was backed-up with statistical tests and revealed that PSO, as a population-based metaheuristic algorithm is an effective technique for automating hyperparameters selection and it efficiently exploits computational resources. The researchers selected hyper-parameters of deep neural networks with PSO, demonstrating that it efficiently explores the solution space, allowing a network of a minimal topology to obtain competitive performance.

The use of the metaheuristic optimization tool is common in automatic pH control applications. Generalized predictive control (GPC) was used by Altiten (2007) to a pH neutralization process. The control aimed to keep the pH value at a given setpoint value when the process was subjected to variations in the feed flow rate. The model parameters were determined by using Bierman and Genetic algorithms. The efficiency of the GPC was observed by calculating the integral of the square of the error (ISE).

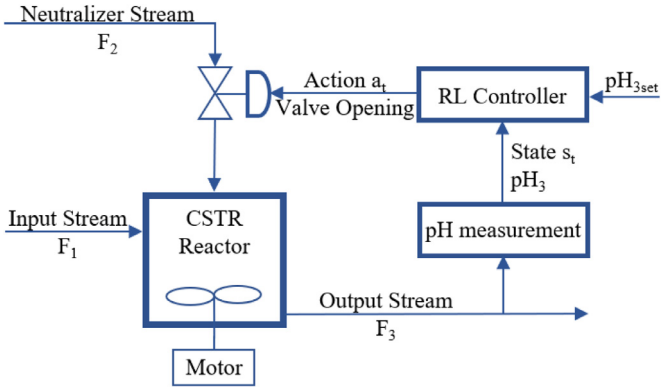

Fig. 1. Modeled process with RL-controller.

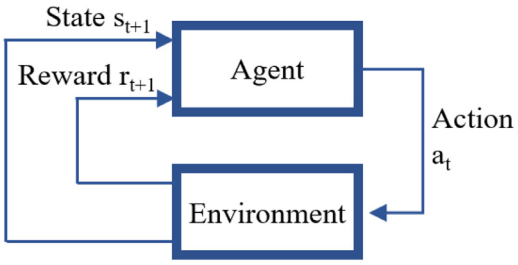

Fig. 2. Diagram of Reinforcement Learning iterations.

3. Methodology

3.1. Case study

As a benefit, Reinforcement Learning does not require an accurate model of the environment for training. However, the authors did not have a dataset or physical plant of the process, so a model was made to simulate the system and acquire the required data.

A model of a case study of effluent neutralization in an electroplating industry in a continuous stirred tank reactor (CSTR) was developed. In the development of the mathematical model, the following assumptions were made; ideal CSTR, isothermal reaction system, negligible heat of reaction, presence of perfectly ionized strong electrolytes, and a continuous operation (without pH buffering effect).

The mathematical model used for the description of the physical system in Fig. 1 was formed with the material balance based on mass conservation, and is presented in Eq. (1).

$$\frac{dC_{H^+}}{dt} = \frac{(C_{H^+,1}F_1 - C_{OH^-,2}F_2 - C_{H^+,3}F_3)}{V_{Reactor}} \quad (1)$$

Where the term hydrogen concentration accumulation over time, dC_{H^+}/dt is a function of the input stream contribution, $C_{H^+,1}F_1$; the portion consumed in the neutralization reaction with the acid effluent, $C_{OH^-,2}F_2$; the portion corresponding to the reactor output stream, $C_{H^+,3}F_3$; and the reactor volume, $V_{Reactor}$.

Defining $V_{Reactor}$=1000L, the input stream i flow, F_1=5L/min, the maximum flow of the neutralizer stream $F_{2,max}$=10L/min. To satisfy the consideration of continuous regime, $F_3=F_1+F_2$, in addition to setting the input stream concentrations: $C_{H^+,1}=10^{-5}$mol/L, $C_{OH^-,2}$=1 mol/L and the initial condition of the output stream pH_3=4. Being the time step dt = 0.07 minutes over a total simulation time of 200 minutes. Totaling 2857 iteration episodes per simulation.

This controlled the H^+ concentration in the output stream,$C_{H^+,3}$, manipulating the opening of a valve that corresponds to the flow rate of neutralizing F2 current. The Eq. (1) differential was solved by the 4th order Runge–Kutta numerical method.

3.2. RL hyperparameters

Reinforcement Learning methods are gaining popularity as an alternative approach to traditional control techniques (Lewis and Vrabie, 2009, Hoskins and Himmelblau, 1992, Syafiie et al., 2008, Shah and Gopal, 2016, Martinez, 2000, Lee and Lee, 2006, Ray Chaudhuri et al., 1996, Khan et al., 2012, Radac et al., 2017). Among the most prominent and promising approaches are model-free online learning methods, which utilizes a stepwise process by interacting with the environment to be controlled, as shown in Fig. 2. They do not require an accurate model of the environment and, most importantly, they do not require the designer to have a great experience in the specific control task.

The basic dynamic of RL is explained by Sutton and Barto (1998). In this algorithm, the agent t receives a representation of its state $s_t \in S$ from the environment, where S is the possible state space, and with this information, the agent selects an action $a_t \in A(s_t)$. $A(s_t)$ is the space of actions available in the s_t state. At the next instant of time, the agent receives a reward $r_{t+1} \in R$ and a new state s_{t+1}.

In general, the agent implements a mapping of states that are likely to perform actions. Such mapping is performed with the agent policy, denoted π_t where $\pi_t(s,a)$ is the probability of $a_t=a$, given $s_t=s$. The learning process changes the policy according to the agent's experience.

The reward signal must be defined so that by maximizing it the agent will achieve the expected goal. The reward value is defined as a function that maps an s_t state, an a_t action, and the resulting state s_{t+1} to a reward $r_{t+1}=R(s_t, a_t, s_{t+1}, a_{t+1})$.

The agent always seeks to maximize the accumulated reward over time. Thus, the agent must maximize the return R, which is the summation of actual reward the agent gets from the environment. This sum can be weighted over time so that immediate rewards are more important than future rewards. This is done by including the discount factor γ. Therefore, it is important to find a balance in γ, to find a good conciliation between future and immediate rewards.

The decision made by the agent, defined by the policy, is a function of the state signal. Thus, the state entity encompasses all kinds of information available to the agent. The state must summarize the experience compactly, but in such a way that all relevant information is retained.

Early RL algorithms assumed discrete sets of states and actions, which made them ill-suited for complex realistic control scenarios. In recent years, though, the focus of researchers has shifted to control tasks defined in multi-dimensional continuous state-action spaces. Learning continuous space domain functions is dealt with Value Function Approximation (VFA) (Busoniu et al., 2017), which provides a means of obtaining approximations of a continuous-valued function from a discrete set of samples.

VFAs use a set of activation functions called features (φ), where F is the number of features used to represent the state space. With higher F, greater precision and description of the state space is achieved, however, this is the most influential factor for the computational time in the execution of RL learning. This is a variable that, when optimized, balances performance and computational time.

A liner VFA is the approximation of a given function $f(s)$ and is denoted $\hat{f}(s)$, as in Eq. (2).

$$\hat{f}(s) = \theta\phi(s) \quad (2)$$

Where θ is the parameter vector of the linear combination, also called the feature weight vector. VFA was used to estimate the

value function V^{π} (Eq. (3)) and policy π (Eq. (4)). Thus, the learning process consists of interactively adjusting the values of the vectors θ_V and θ_π. In this study, Gaussian radial basis functions (RBF) were used.

$$\hat{V}_\pi(s_t) = \theta_V \phi(s_t) \tag{3}$$

$$\hat{\pi}(s_t) = \theta_\pi \phi(s_t) \tag{4}$$

The learning methodology applied was Actor–Critic (AC) (Cheng et al., 2013, Fernandez-Gauna et al., 2018). This design is best suited to work with continuous states and actions because it keeps a separate structure to represent the value function and the action selection policy being learned.

In this approach, at the end of each iteration, with the agent taking the action a_t and reaching the state s_{t+1}, a two-step process is accomplished. The first step is the policy evaluation and is made by the critic. It will update its estimate of the value function V_π from the new obtained data with the state transmission occurred. So that the actor can then update the policy, the critic gives it feedback, the time difference error δ_t representing the increment of the value function in s_t, according to Eq. (5), where $\hat{V}_\pi$ is the estimated value function.

$$\delta_\tau = r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t) \tag{5}$$

The evaluation of the policy uses the concept of eligibility trace, which is a memory of recent states and how many instances they were visited over the period of time (Fernandez-Gauna et al., 2018). With this, it can spread the responsibility for acquiring a certain reward throughout the visited states, with the most recent and most visited prioritized. The eligibility trace vector z_t is defined in Eq. (6).

$$z_{t+1} = \gamma \lambda z_t + \phi(s_t) \tag{6}$$

Being γ the discount factor previously presented and λ is a tuning parameter called the trace discoloration factor. The importance of the trace decay factor is noteworthy for updating the z vector, and the consequence on algorithm learning. This being another hyperparameter, optimization has a direct influence on learning performance.

Then, to perform the policy evaluation and generate a new value function estimate, the value function's weight vector θ_V must be updated, as shown in Eq. (7).

$$\theta_{V,t+1} = \theta_{V,t} + \alpha_V z_{t+1}\{r_t + \gamma \phi_V(s_{t+1})\theta_{V,t} - \phi_V(s_t)\theta_{V,t}\} \tag{7}$$

Where α_v is a learning tuning parameter that sets the value of the update step in the direction of the optimal parameter. The critic will pass the value of δ_t to the actor and the different error will be part of updating the policy's vector θ_π weight.

After the critic assesses the policy, the actor can then update the θ_π weights, toward the optimal policy. This update is defined in Eq. (8).

$$\theta_{\pi,t+1} = \theta_{\pi,t} + \alpha_\pi \nabla\theta_\pi \pi(S_t)(a_t - \pi(S_t))\delta_t \tag{8}$$

Where α_π is a learning parameter tuning, analogous toα_v, which sets the value of the update step in direction of the optimal parameter, both of which are optimizable; $\nabla\theta_\pi \pi(s_t)$ is the gradient of $\pi(s_t)$ relative to θ_π.

Another point to be defined in the RL methodology is exploration. Initially, the agent has no information about the environment or the task that it must perform. It needs to explore the state space to find out which states return good rewards and improve its policy.

This work uses the exploration technique based on the addition of noise to the policy signal that decreases over the iterations so that the learning processes may converge. So, at the beginning of learning, the agent explores a lot, not knowing the environment yet. Subsequently, it takes more advantage of the information obtained and its actions directly reflect the policy.

Table 1
RL hyperparameters.

Symbol	Description
γ	Future rewards discount factor.
F	The number of features used to represent state and action space.
λ	Decay factor of eligibility trace.
α_V	Actor learning gain.
α_π	Critic learning gain.
V_N	Noise decreasing speed.
k_r	Reward adjustment factor.

The perturbation N is a random value of a zero median range and decreasing amplitude, calculated using Eq. (9).

$$N = \text{rand}\left[-\frac{1}{V_N^{sim}}, +\frac{1}{V_N^{sim}}\right] \tag{9}$$

Where V_N^{sim} is the rate of noise decay, to the power of the number of simulations already executed. Thus, this factor ensures that throughout training simulations, noise decays are exponentially based on V_N. This rate is a parameter to be optimized, thus balancing the exploit-explorer ratio to the best of its performance.

3.3. RL as autonomous controller

The RL agent is the controller of the closed-loop process illustrated in Fig. 2. For this, it was necessary, with the defined control problem, determine which process variables would be used as states, actions in addition to establishing the control objective through the reward function

The action is the control signal, referring to F_2, and the simulation environment responding with the state, which is the controlled variable $C_{H^+,3}$. The reward will be set accordingly to the setpoint to be reached.

In this work, we aimed to neutralize the effluent output stream, the setpoint is the pH of stream 3 (pH_3) equal to 7, then, $C_{H^+,3}=10^{-7}$. Thus, the reward function, Eq. (10), was defined based on two points:

- With smaller error $|pH_3 - pH_{3,set}|$, a larger reward is generated, minimizing the error and taking the controlled variable to the setpoint.
- If $|a_t - a_{t+1}| < 0.05$ the reward is doubled. Since the action space is normalized to $a \in [0,1]$, this criterion increases the reward for smoother control strategies that minimizes the valve wear by opening and closing the frequency.

$$R_{t+1} = \begin{cases} kr[1 - (|pH3 - pH3, set|)], & |at - at+1| > 0.05 \\ 2kr[1 - (|pH3 - pH3, set|)], & |at - at+1| < 0.05 \end{cases} \tag{10}$$

Where k_r is a constant that will adjust the range of reward values. This variable also directly influences learning, since the order of magnitude of the reward determines the weight it will have in adjusting the parameters, on θ_V directly, and on θ_π via δ_t.

The RL controller in this work is autonomous, as all its hyperparameters (Table 1), which depend on the environment and the learning method, will be determined by an optimization algorithm. Thus, ensuring its best performance, and making the only knowledge of the process required for its execution of the reward function, to inform the goal to be achieved by the controller.

3.4. Particle swarm optimization

The original Particle Swarm Optimization (PSO) algorithm, developed by Kennedy and Eberhart (1995), makes use of a compu-

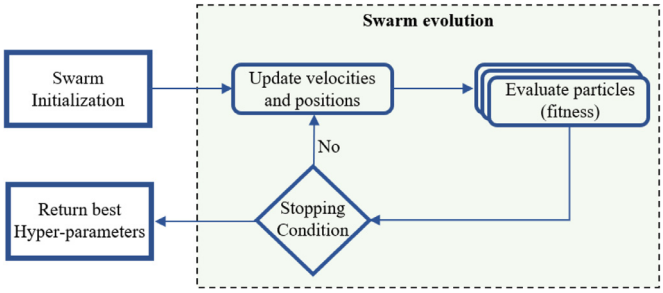

Fig. 3. Flowchart of the PSO iteration dynamic. Adapted from (Lorenzo et al., 2017).

tational paradigm based on the phenomenon of collective intelligence exhibited by swarms of insects and flocks of birds.

In PSO, each particle represents a possible solution to the optimization problem. It is initialized by the assignment of random positions and velocities in the universe where the solutions will transit.

The algorithm records the best position of each particle for an evaluation index, "pbest"; and the entire population presents a better overall candidate solution, "gbest". At each iteration, each particle stochastically accelerates towards pbest and gbest and updates its position, following Eqs. (11) and (12) (Shi and Eberhart, 1998), until a stop condition such as a maximum number of iterations is reached, as illustrated in Lorenzo et al. (2017), Fig. 3.

$$v_{i+1,d} = wv_{i,d} + c_1 rand()(pbest_{i,d} - x_{id}) + c_2 rand()(gbest_d - x_{id}) \quad (11)$$

$$x_{i+1,d} = x_{i,d} + v_{i+1,d} \quad (12)$$

where $v_{i+1,d}$ is the velocity in the next iteration in the dimension d, $v_{i,d}$ is the velocity i in the dimension d, w is the inertia, c_1 and c_2 are constants, rand() is a random fact, and $x_{i+1,d}$ and $x_{i,d}$ are, respectively, position x in the next iteration and current position in d.

3.5. Application of particle swarm optimization

The PSO is worked by varying the 7 RL hyper-parameters based on its exploration, as seen in Eqs. (11) and (12), to minimize the Integral Square Error (ISE) of the final simulation with RL controller and ensure stable responses.

Each particle at each moment represents a set of values for the 7 hyperparameters. With these variables defined, 25 RL-controlled neutralization CSTR reactor training simulations are performed, and a final simulation where exploration is zeroed, that is, the added noise to $\pi(s_t)$ is null (N=0) and RL responds only with the knowledge it has acquired. For this final simulation, ISE is calculated, which must be minimized by PSO.

This process (25 trainings simulations + 1 final simulation) is repeated 5 times, resetting θ_V and θ_π between them, thus restarting learning. The 5 ISE values of the final simulations are analyzed and if they are not stable responses, with a standard deviation of less than 10% of the mean, the particle result is discarded. If the final 5 simulations have stable ISEs, the particle has the average fitness of these values.

As a comparison, the configured PSO was also used to tune a classic PID controller in digital form, Eq. (13). The optimizer aimed to find bias and gains: proportional (Kc), integral (τ_i) and derivative (τ_D), responsible for the lowest ISE of the simulation with the respective controller.

$$u_{k+1} = bias + K_C\left(error_k + \frac{\Delta t}{\tau_i}\sum_{i=1}^{k} error_i + \tau_D\frac{\Delta error_i}{\Delta t}\right) \quad (13)$$

Optimization via PSO proved to be an efficient and better technique than other evolutionary algorithms for tuning PID controller parameters like genetic algorithms. PSO showed better time response and efficiency in determining global optima (Chen, 2007).

3.6. Deployment

The electroplating industry effluent neutralization reactor simulator, RL controller with actor-critic methodology, and PSO algorithm for optimizing hyperparameters were first developed in Scilab 6.0.0 and coupled, as illustrated in Fig. 4. It is noteworthy that the dynamics of simulations are fully represented in Fig. 1.

It was necessary to perform a methodology change when preliminary result was obtained, due to the low computational performance of Scilab. PSO worked with 7 dimensions and each particle performed a total of 130 simulations (25 trainings + 1 final, 5 times). This made it impossible to execute the final algorithm in Scilab 6.0.0 on a machine without high computational power. The alternative chosen to enable the algorithm execution in this developed work was:

1. Translate the code to C++, a lower-level language, closer to the machine language, which decreases the computational time required for execution.
2. The code was executed in the cloud. Cloud9 of Amazon Web Services (AWS), a Cloud-based IDE was used, which offers a good experience in the development of serverless applications and ensures stable program execution (Amazon Cloud9).

The virtual processing instance on AWS was free t2.micro instance (Amazon EC2), which consists of 1 virtual CPU with 6 CPU/hour credits, 1GiB memory and storage at Amazon Elastic Block Store. With this machine, the algorithm took about 6 days and 4 hours to complete.

4. Results

The first analysis considers the configuration of the PSO used. Maximum number of iterations was set to 100 and the population size was varied from 5 to 150 particles. Each configuration was performed 3 times, the average of the determined gBests are shown in Fig. 5.

The non-reproducibility was observed due to the high variance in the configurations with a few particles (over 75 particles) as the convergence was guaranteed, finding results closer to the optimum repeatedly. The future analyses presented in this work were carried out in the configuration of 100 particles in the population since this is a point of balance between convergence and computational time.

The convergence of PSO can be evidenced in Fig. 6, by the stability of gBest throughout the iterations, being the smallest fitness found among all particles. Its convergence means the proximity of the particles with the optimal set of hyperparameters, the one that brings the RL controller to the best performance.

The best swarm particle had its best position in the iteration 75, which is consistent with the best set of parameters found as shown in Table 2. These were the parameters responsible for generating the lowest average of ISEs with a standard deviation within an acceptable range. Average was 8778.6 with a standard deviation of 0.0022%.

The pH behavior over time in electroplating industry wastewater treatment, based on neutralization in a CSTR, controlled by RL with the parameters found by the PSO can be seen for the training and final simulations of Fig. 7.

It can be observed that the RL has learned by manipulating the neutralizer input to drive and maintain the output current pH at 7, thus achieving the goal given to it by the reward function.

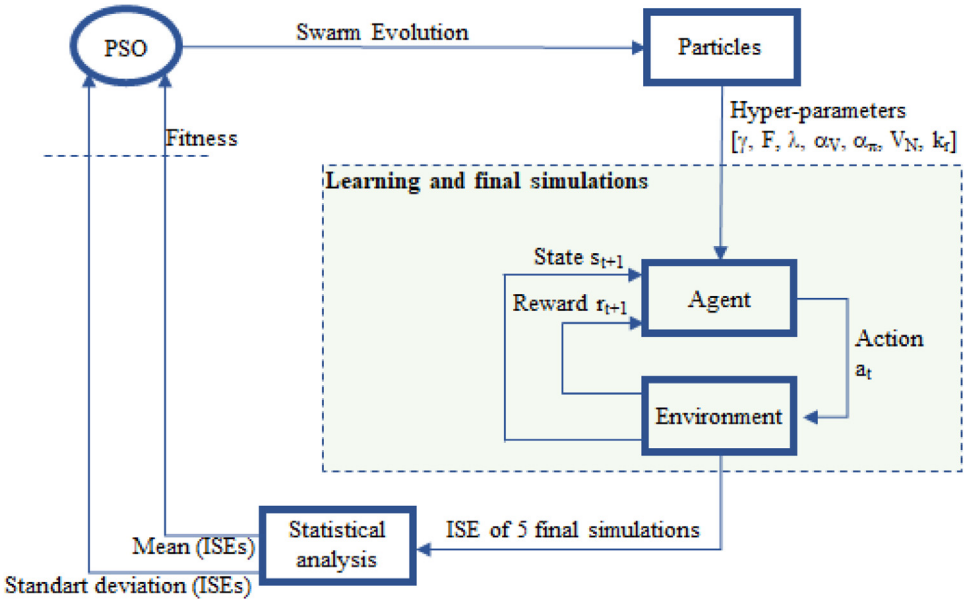

Fig. 4. Simulator-RL Controller-PSO coupling algorithm.

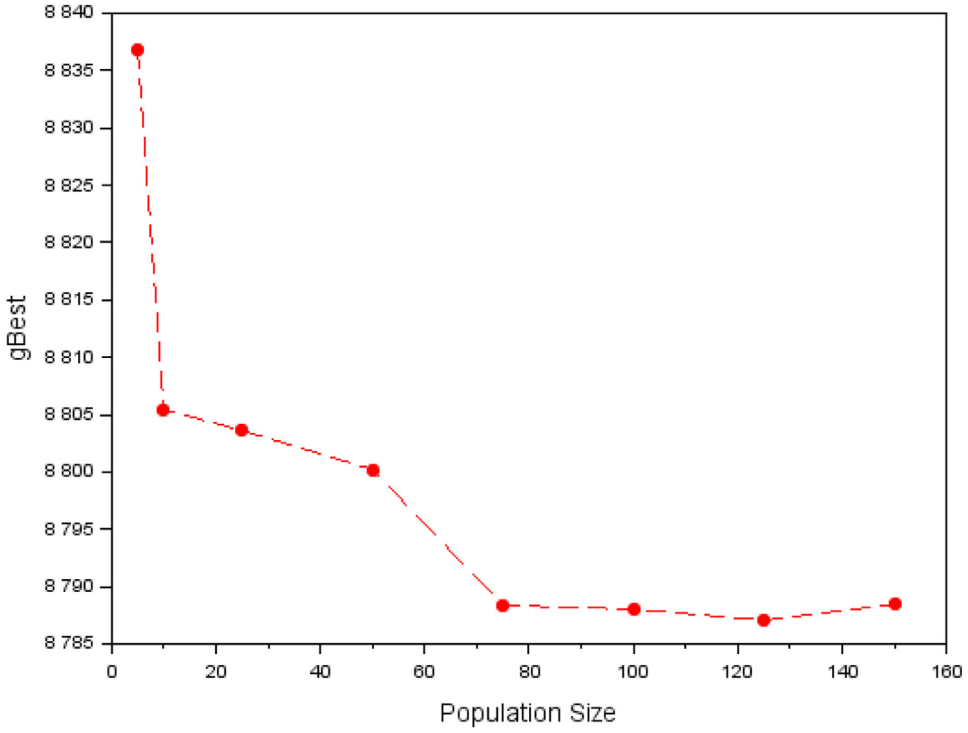

Fig. 5. Minimum ISE (gBest) in different configurations of PSO, with 100 maximum iterations.

To evaluate the optimal hyperparameters determined in Table 2, an analysis was made of how the algorithm behaves in an extensive range of combinations of learning gains, both for the actor (α_π) and for the critic (α_V). These hyperparameters were chosen because they have the biggest influence on the RL algorithm in all stages. These parameters were varied by multiplying the optimum values (α_π = 2.2051E-6, α_V= 1.1534E-4) by different factors, stating all other hyperparameters were fixed. The variation of the ISE of the final simulation in face of this combination of factors is shown in Fig 8. It is observed that the optimum founded (alfa_P factor = alfa_V factor =1) is really in a global minimum.

Fig. 7 shows the evolution of the 25 learning simulations. Fig. 9 shows more clearly the importance of the number of training simulations, an extremely important factor in time and computational cost of the controller training. In this step, performance and reproducibility analysis, tests were performed with different amounts of learning simulations, always with a final simulation, which has its ISE evaluated. Each set of simulations (training plus final) was performed 10 times, the ISE's averages are plotted in Fig. 9. The standard deviation was very small, meeting the optimization objective, at a maximum of 0.034% and decreases with the number of simulations, guaranteeing the reproducibility of the controller performance for the same set of hyper-parameters.

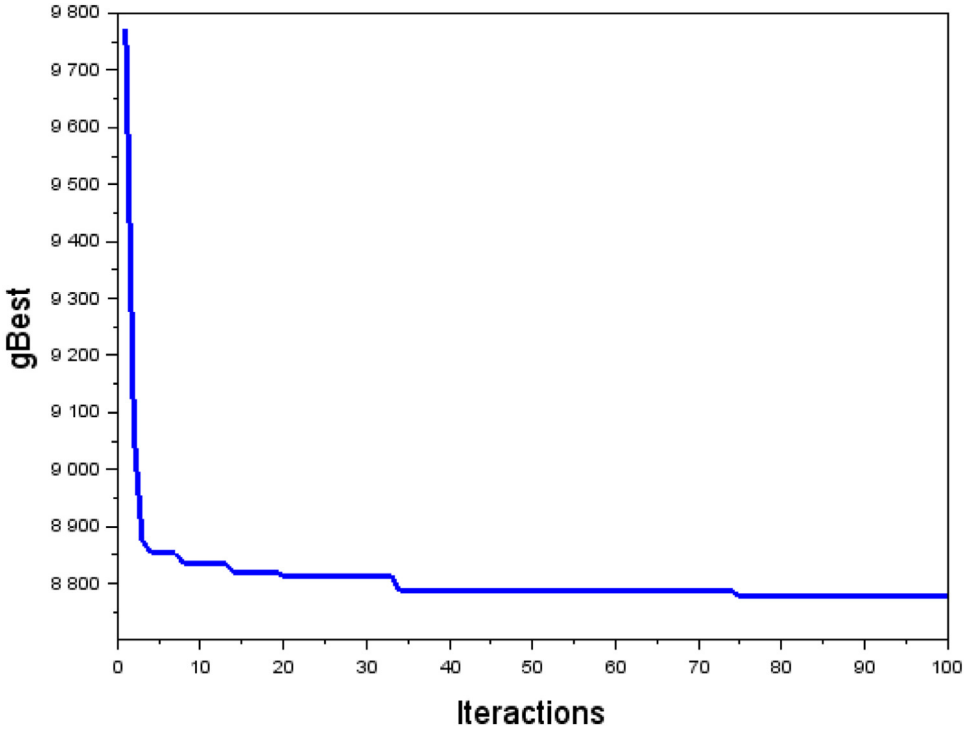

Fig. 6. Evolution of gbest over PSO iterations for RL hyperparameters optimization.

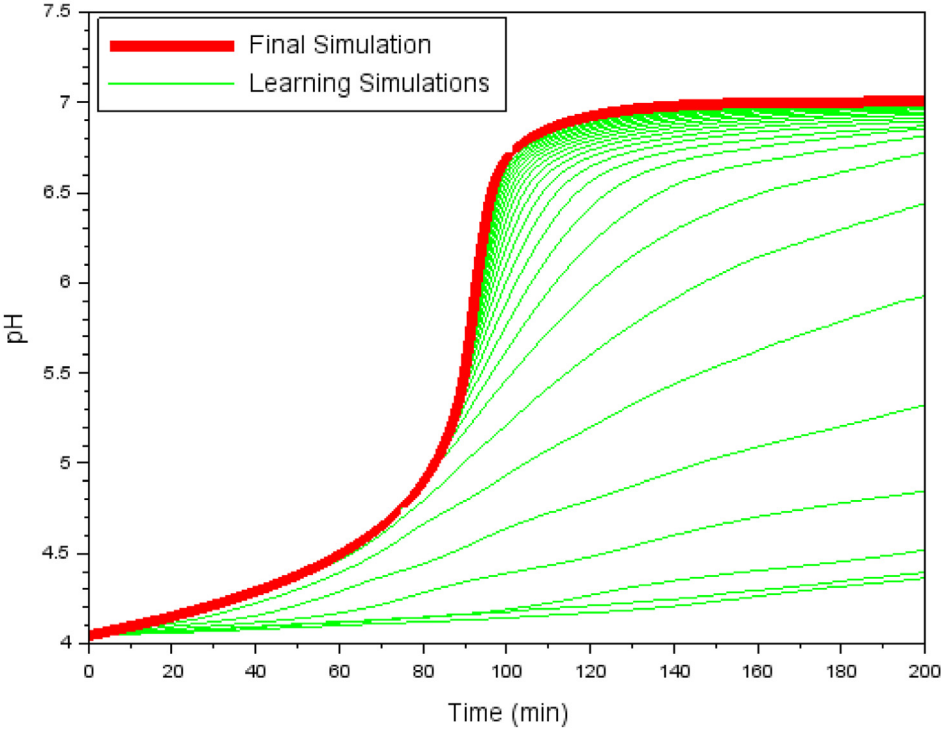

Fig. 7. Controlled pH by RL in training and final simulations.

In the optimization of the classical controller, two suboptimal responses were found when tuning the PID controller, which are presented in Table 3. The particle responsible for the PSO gBest, with the smallest ISE, is presented as the first solution. However, this response sets the bias to zero. The second scenario, with bias, showed a difference of only 0.04% compared to the first scenario and the presence of bias in the PID controller configuration can make it more effective in some situations. Therefore, this paper will compare the RL controller with these two PID configurations.

The performance comparison between classic PID and RL based controllers is shown in Fig. 10. In the first analysis by ISE, the PID controller performed better due to the shorter rising time of the controlled variable, however, it has an offset of 1.4%, while the RL controller showed no offset. The performance of the PID controller

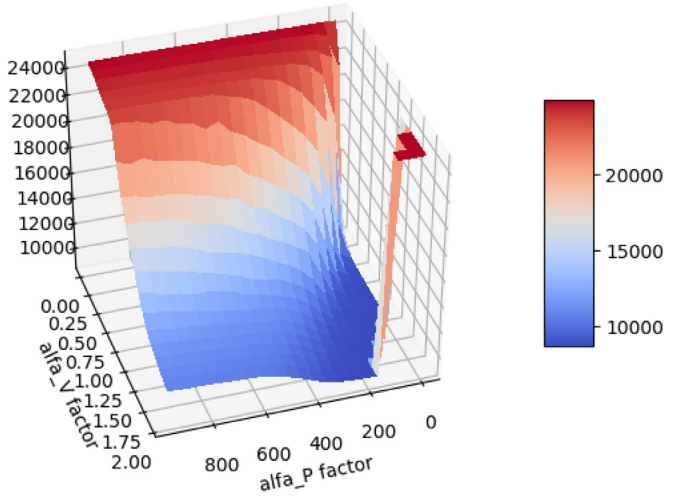

Fig. 8. ISE of the final simulation controlled by RL with different combinations of learning gains.

Table 2
Best particle position by PSO for RL controller optimization.

Hyper-parameter	Best value
γ	0.3619
F	32
λ	0.9041
α_V	1.1534E−4
α_π	2.2051E−6
V_N	1.9654
k_r	0.84

Table 3
The best answers for PID parameters optimization by PSO.

	Bias	Kc	τ_i	τ_D	ISE
1	0	−531.27	9459.2	1.6907	8724.6
2	0.8206	−2962.4	3235.8	1.6567	8728.1

represents both versions (with and without bias), as its graphical difference is imperceptible.

Up to this point, the RL controller has only been shown to work when it has been optimized. To evaluate the intelligent controller's autonomy capacity, it was submitted to situations with variations in the process in regulatory and servo operation.

In the regulatory operation, the simulation time was increased to 800 minutes, and after 200 minutes there was a variation in the pH of the process input stream. Fig. 11 shows the performance of RL, PID controllers with and without bias with input current pH equal to 2.

This scenario shows the ability of the RL controller to learn from the environment, even after the optimization and learning simulations, where it perceives changes and learns from it, adapting to process variations inherent to the real-world control problems. The poor and oscillating PID performance shows the difficulty of a classic controller in high nonlinear processes such as pH control (Shinskey, 1973). Both controllers went through the same optimization step for the process with pH 5 as input current. This result demonstrates the robustness and autonomy of the Reinforcement Learning-based intelligent controller.

This latter analysis was redone to intermediate input stream pH's. The ISE of each simulation controlled by the three controllers as a function of input current pH is shown in Fig. 12. The bias PID controller was the one that showed the worst performance under new conditions. The intelligent RL-based controller is the one that demonstrates the lowest performance degradation. In the very different situations from the primary conditions, pH 2 and 2.5 for the input current becomes the one with the lowest ISE.

It is worth mentioning that the control was made with variations in the input pH, as this is the most impact variable in the

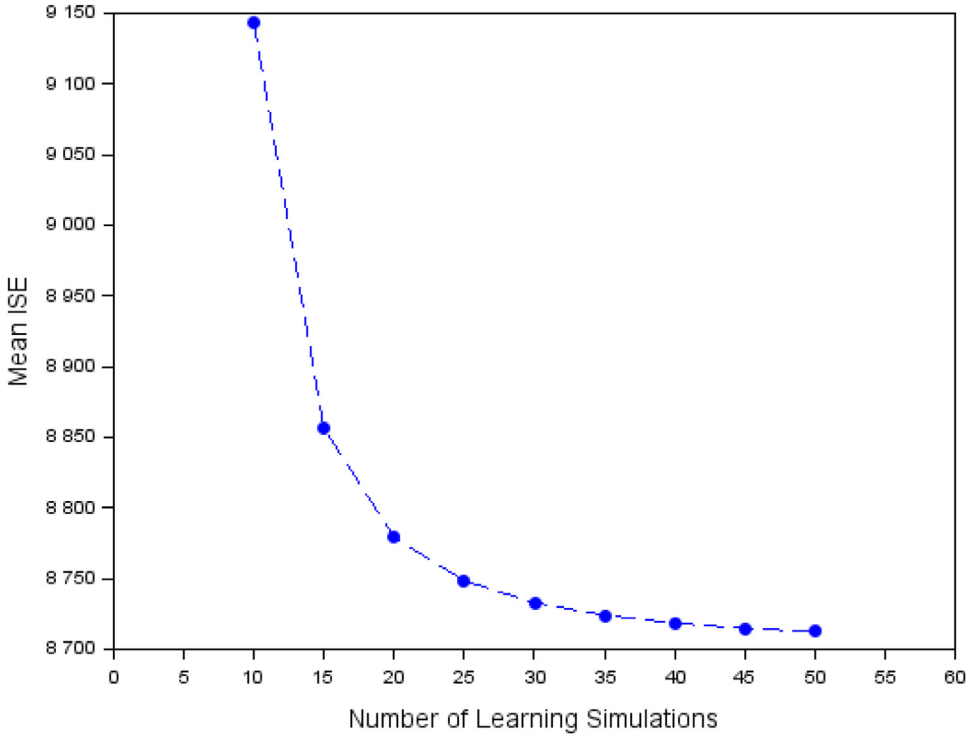

Fig. 9. Performance analysis of the RL controller with different amounts of learning simulations.

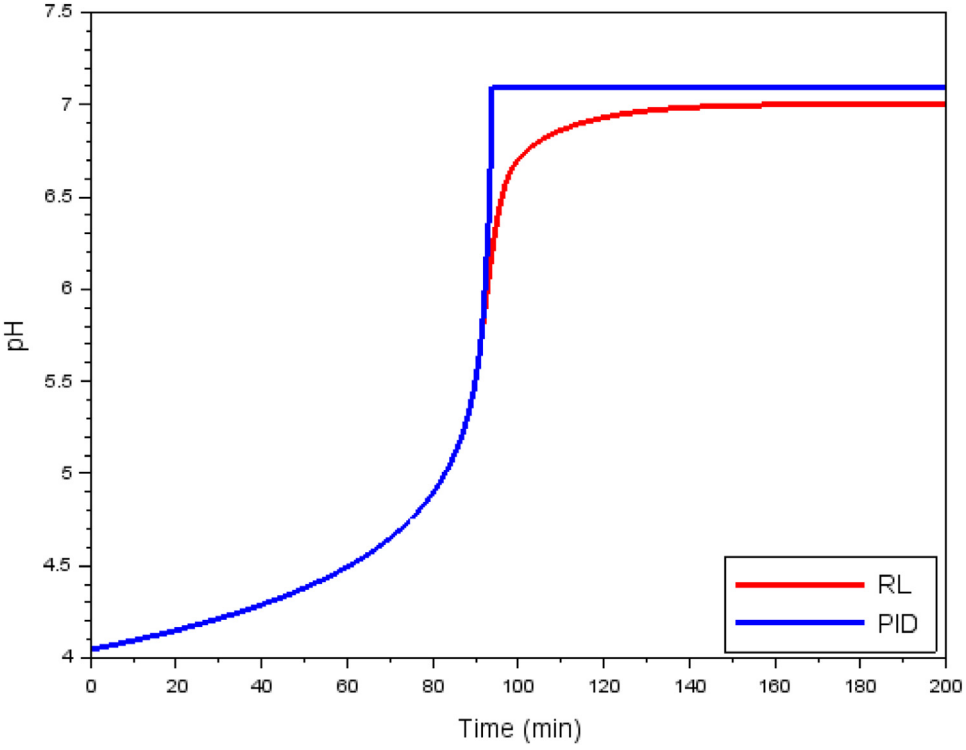

Fig. 10. pH controlled by RL and PID.

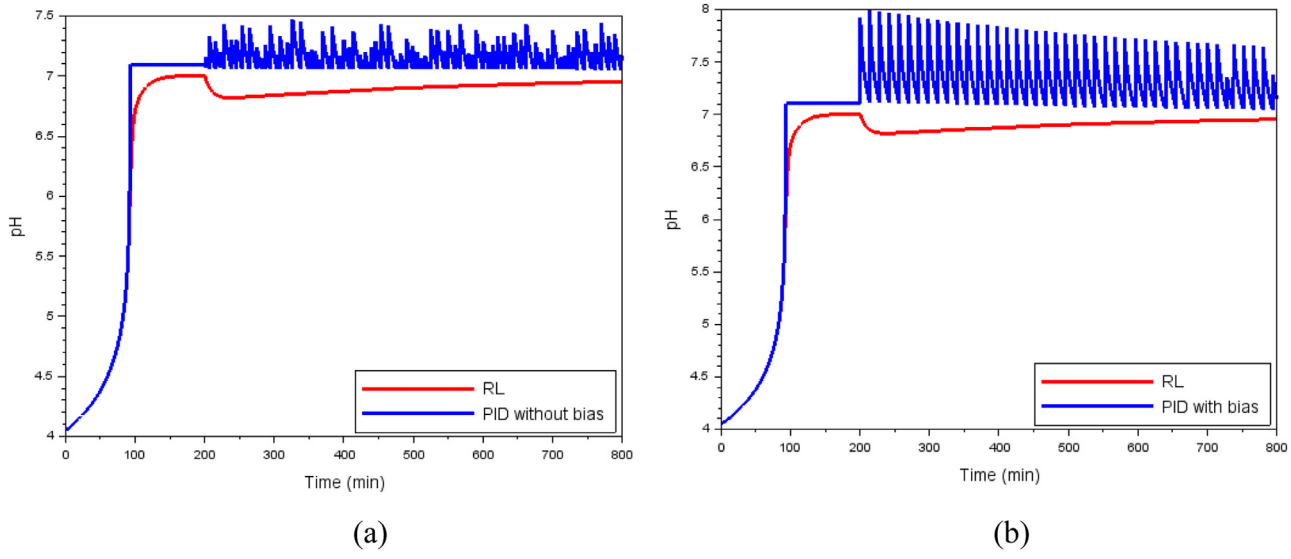

Fig. 11. pH dynamics in regulatory operation with RL and PID (a) without and (b) with the bias controller.

process, and is the variation most likely to occur in the electroplating industry. The variations went to more acidic pH's to test the neutralization case study.

In servo operation, simulations of 800 minutes were also done, but with small variations in the desired pH of the output current, the process setpoint, starting from 200 minutes. Figure 13 shows the performance of RL, PID controllers with and without bias with the new pH setpoint equal to 7.4.

Again, the robustness of the RL controller over classic PID controllers is explicit. In this case the PID control, upon entering the new operating condition, responds with an overshoot, requiring a long time to return to the setpoint. PID without bias responds with a smaller overshoot but still represents 12.6%. While the RL control realizes the change, it explores the space of actions and states and reaches the new setpoint. This analysis has been redone for intermediate setpoints. The ISE of each simulation with the different controllers is presented in Fig. 14.

The PID with bias controller performs better than without bias in all test scenarios, but both, when out of the initial optimization situation, perform worse than the RL controller, again showing robustness, autonomy and ability to continue learning and adapting throughout the process.

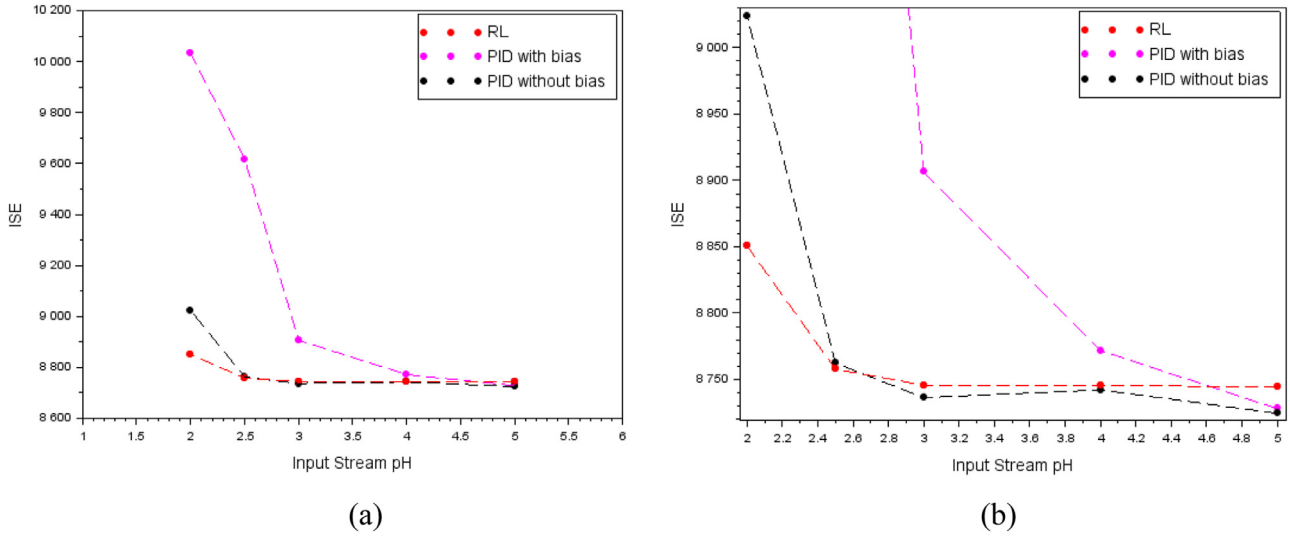

Fig. 12. ISE's from regulatory operation simulations to RL and PID without and with bias controllers. (a) Full Graphic (b) Zoom for comparison between RL and PID with bias.

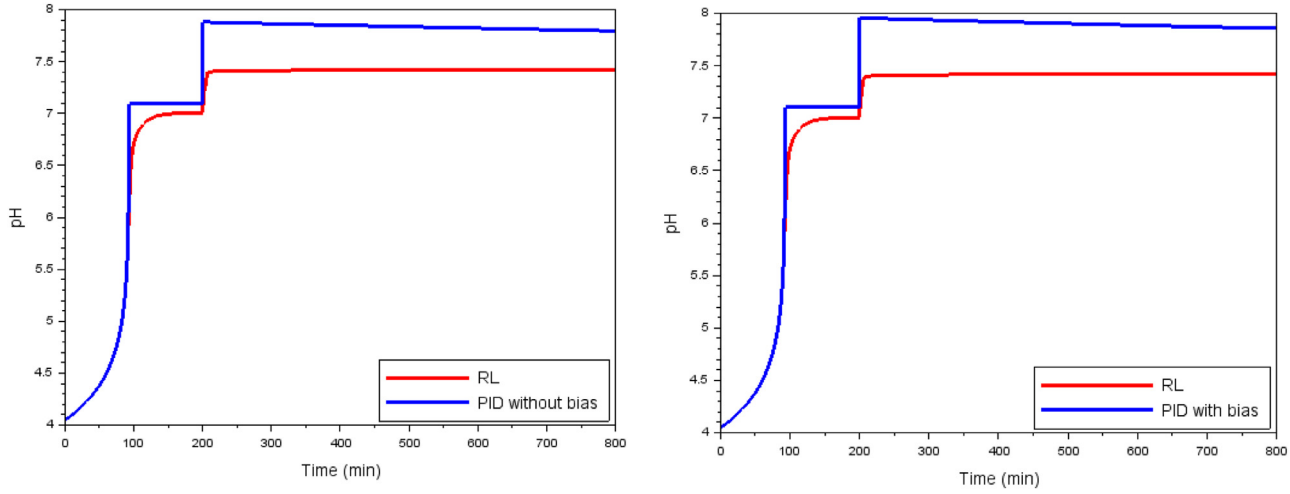

Fig. 13. pH dynamics in servo operation with RL and PID without and with bias controllers.

5. Final considerations and conclusions

This work developed, applied and validated a Machine Learning technique for autonomous pH control via Reinforcement Learning (RL) for an electroplating industry wastewater neutralization reactor.

To do so, a numerical simulator of an effluent neutralization reactor for electroplating industry was modeled, where the control strategies were tested. The actor-critic RL methodology was chosen and adapted with VFAs stochastic policy to work with a continuous space of states and actions.

A PSO algorithm has been adapted and executed on the cloud to determine the optimal controller hyperparameters. Thus, the only knowledge of the process to be introduced in this technique is the inherent characteristic of control, which is the goal to be achieved via a reward function for the RL.

The RL-based controller proved its autonomy in regulatory and servo operations, in addition to having its performance validated when compared to the classic controller. The PID algorithm presented an oscillatory control performance and a 12.6% offset in the regulatory and servo operation, respectively. In both scenarios, RL controller had smooth offset-free control.

Future research steps will involve decreasing the simplifications of the dynamic simulator. By disregarding the considerations of isothermal reaction system, with negligible reaction heat, presence of perfectly ionized strong electrolytes, without buffering effect, thus bringing the simulator even closer of the real case.

Analyses were also performed to evaluate the importance of the optimization algorithm configuration and how the RL behaves with poor hyperparameters, which proved the efficiency of the PSO in finding the optimal set of hyperparameters for the RL controller.

In the future, the authors intend to start testing the RL autonomous controller in pilot plants of the process, in the treatment of real effluents from electroplating industry.

An interest was generated in studying possibilities of RL methodologies, and other ways of working with continuous state and action spaces, such as the use of the Deep Reinforcement Learning methodology associated with the transformation of the current model into a Multiple Inputs Multiple Outputs - MIMO

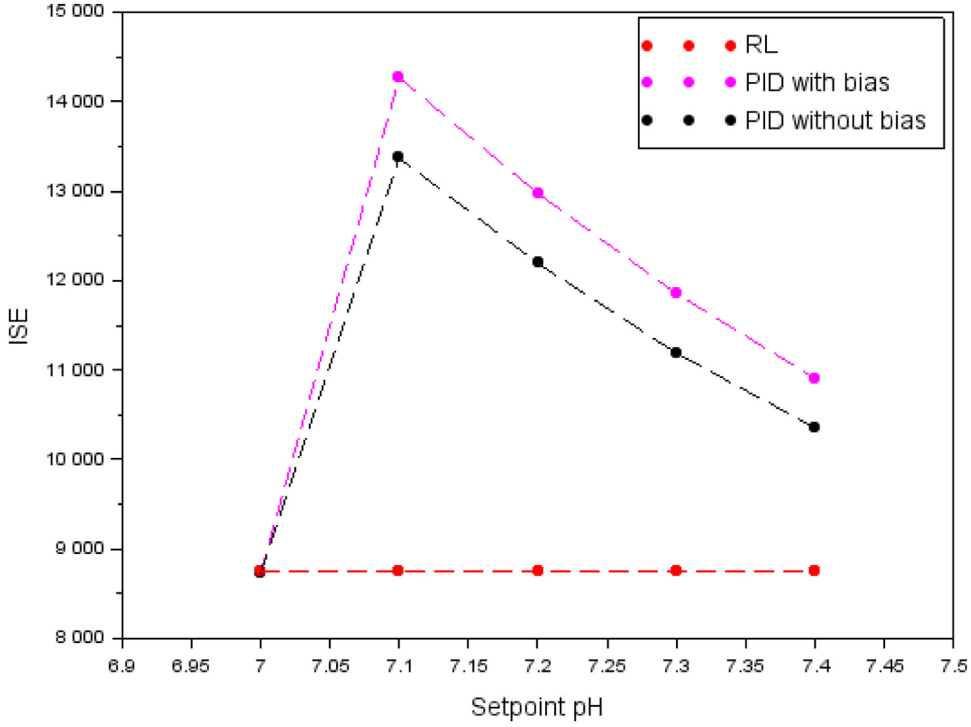

Fig. 14. ISE's from servo operation simulations to the RL and PID without and with bias controllers.

system. Allowing more opportunities for process manipulation and control.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.106909.

CRediT authorship contribution statement

Douglas Alves Goulart: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Renato Dutra Pereira:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

References

Agrawal, A., Sahu, K.K., 2009. An overview of the recovery of acid from spent acidic solutions from steel and electroplating industries. J. Hazard. Mater. 171 (1-3), 61–75.

Altiten, A., 2007. Generalized predictive control applied to a pH neutralization process. Comput. Chem. Eng. 31 (10), 1199–1204.

Amazon Cloud9, https://aws.amazon.com/cloud9.

Amazon EC2, https://aws.amazon.com/ec2.

Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. Neural Netw. 437–478.

Bertsekas, D.P., Tsitsiklis, J.N., 1995. Neuro-dynamic programming: an overview. In: Proceedings of 34th IEEE Conference on Decision and Control, 1, pp. 560–564.

Bojic, A.L., Bojic, D., Andjelkovic, T., 2007. Removal of Cu^{2+} and Zn^{2+} from model wastewaters by spontaneous reduction-coagulation process in flow conditions. J. Hazard. Mater. 168, 813–819.

Busoniu, L., Babuska, R., De Schutter, B., Ernst, D, 2017. Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC press.

Candelieri, A., Giordani, I., Archetti, F., Barkalov, K., Meyerov, I., Polovinkin, A., Sysoyev, A., Zolotykh, N., "Tuning hyperparameters of a SVM-based water demand forecasting system through parallel global optimization", Comput. Oper. Res., vol. 106, pp. 202-209.

Chen, S.F., 2007. Particle swarm optimization for PID controllers with robust testing. In: International Conference on Machine Learning and Cybernetics IEEE, 2, pp. 956–961.

Chen, S.S., Cheng, C.Y., Li, C.W., Chai, P.H., Chang, Y.M., 2007. Reduction of chromate from electroplating wastewater from ph 1to 2 using fluidized zero valent iron process. J. Hazard. Mater. 142 (1-2), 362–367.

Cheng, Y., Feng, H., Wang, X., 2013. "Efficient data use in incremental actor–critic algorithms. Neurocomputing 116, 346–354.

Cushnie, G.C. Jr, "Electroplating wastewater pollution control technology", 1985.

Fernandez-Gauna, B., Osa, J.L., Grana, M., 2018. "Experiments of conditioned reinforcement learning in continuous space control tasks". Neurocomputing 271, 38–47.

Fliess, M., Join, C., 2013. Model-free control. Int. J. Control 86 (12), 2228–2252.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D., 2018. Deep Reinforcement Learning that matters. Thirty-Second AAAI Conference on Artificial Intelligence.

Hoskins, J., Himmelblau, D., 1992. Process control via artificial neural networks and reinforcement learning. Comput. Chem. Eng. 16 (4), 241–251.

Hosseini, S.S., Bringas, E., Tan, N.R., Ortiz, I., Ghahramani, M., Shahmirzadi, M.A, 2016. Recent progress in development of high-performance polymeric membranes and materials for metal plating wastewater treatment: a review. J. Pro. Eng. 9, 78–110.

Hou, Z.S., Wang, Z., 2013. From model-based control to data-driven control: survey, classification and perspective. Inf. Sci. 235, 3–35.

Hou, Z., Jin, Z., 2011. Datta-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems. IEEE Trans. Neural Netw. 22 (12), 2173–2188.

Islam, R., Henderson, P., Gomrokchi, M., Precup, D., "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control", 2017 arXiv:1708.04133.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., "Population based training of neural networks", 2017 arXiv:1711.09846.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization (pso). In: Proc. IEEE International Conference on Neural Networks. Perth, Australia, pp. 1942–1948.

Kerney, U., 1994. Treatment of spent pickling acid from hot dip galvanizing. Resour. Conservation Recycl. 10 (1-2), 145–151.

Kerney, U., 1997. Treatment of zinc containing spent pickle acids. In: Proceedings: 18th International Galvanizing Conference, European General Galvanizers Association.

Khan, S.G., Herrmann, G., Lewis, F.L., Pipe, T., Melhuish, C., 2012. Reinforcement

learning and optimal adaptive control: An overview and implementation examples. Ann. Rev. Control 36 (1), 42–59.
Kleingarn, J.P., 1988. Pickling in hydrochloric acid. In: Proceedings: 15th International Galvanizing Conference, European General Galvanizers Association.
Kobya, M., Demirbas, E., Ozyonar, F.U.A.T., Sirtbas, G., Gengec, E.R.H.A.N., 2017. Treatments of alkaline non-cyanide, alkaline cyanide and acid zinc electroplating wastewaters by eletrocoagulation. Process Safety Environ. Protect. 105, 373–385.
Krishnapura, V.G., Jutan, A., 2000. A neural adaptive controller. Chem. Eng. Sci. 55 (18), 3803–3812.
Lee, J.H., Lee, J.M., 2006. Approximate dynamic programming based approach to process control and scheduling. Comput. Chem. Eng. 30 (10-12), 1603–1618.
Lewis, F.L., Vrabie, D., 2009. "Reinforcement learning and adaptive dynamic programming for feedback control". IEEE Circuits Syst. Mag. 9 (3), 32–50.
Liessner, R., Schimitt, J., Dietermann, A., Bäker, B., 2019. Hyperparameter optimization for deep reinforcement learning in vehicle energy management. ICAART.
Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., R.Pastor, J., 2017. "Particle swarm optimization for hyper-parameter selection in deep neural networks". In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM, pp. 481–488.
Lorenzo, P.R., Nalepa, J., Ramos, L.S., Pastor, J.R., 2017. "Hyper-parameter selection in deep neural networks using parallel particle swarm optimization". In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, pp. 1864–1871.
Martinez, E.C., 2000. Batch process modeling for optimization using reinforcement learning. Comput. Chem. Eng. 24 (2-7), 1187–1193.
Probst, P., Wright, M.N., Boulesteix, A.L., "Hyperparameters and tuning strategies for random forest", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 3, pp. 1301.
Radac, M.B., Precup, R.E., 2019. Data-driven model-free tracking reinforcement learning control with VRFT-based adaptive actor-critic. Appl. Sci. 9 (9), 1807.
Radac, M.B., Precup, R.E., Roman, R.C., 2017. Model-free control performance improvement using virtual reference feedback tuning and reinforcement q-learning. Int. J. Syst. Sci. 48 (5), 1071–1083.
Ray Chaudhuri, T., Hamey, L.G., Bell, R.D., 1996. "From conventional control to autonomous intelligent methods". IEEE Control Syst. Mag. 16 (5), 78–84.
Robson, J., 1993. Steel pickling: a profile. Draft Report, EPA Contract (68-D1) 0143.
Shah, H., Gopal, M., 2016. "Model-free predictive control of nonlinear processes based on reinforcement learning". IFAC Pap. OnLine 49 (1), 89–94.
Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, pp. 69–73.
Shin, J., Badgwell, T.A., Liu, K.H., Lee, J.H., 2019. Reinforcement learning–overview of recent progress and implications for process control. Comput. Chem. Eng. 127, 282–294.
Shinskey, F.G., 1973. pH and pION Control in Process and Waste Streams, 1. Wiley.
Sutton, R.S., Barto, A.G., 1998. Introduction to reinforcement learning. MIT Press Cambridge 135.
Syafiie, S., Tadeo, F., Martinez, E., Weber, C., Elshaw, M., Mayer, N., 2008. Model-free learning control of chemical processes. Reinforcement Learn. 295.
Werbos, P.J., 1992. Approximate dynamic programming for real-time control and neural modeling. In: Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, pp. 493–525.