1 Introduction

Classical feedback controllers are an established technology in modern process control. Although more complex control systems have been developed in the past decades, the proportional–integral–derivative (PID) controller is still the most widely used controller throughout the process industry due to its simplicity and wide range of use [1][2]. In order to achieve a certain degree of performance, PID controllers do, however, require some kind of tuning in relation to the specific process at hand. A common approach of such tuning is to adjust the controller's settings in situ i.e. during the initial commissioning of the process plant. A second, more advanced and complex approach consists of creating a dynamic mathematical model of the process which can then be used to determine the PID tuning parameter values [3]. Creating such a sufficient approximative mathematical model can, however, be challenging or even practically unfeasible.

On the other hand, there has been a regained interest in the field of reinforcement machine learning (RL) during the last decade, partly due to the successful implementation of artificial neural networks (NN). This has allowed RL agents to learn complex control tasks like playing computer games on a human-like level and without any prior knowledge of the game environment [4]. The successful use of reinforcement learning evidently imposes the question if and how RL algorithms can add additional value to the existing industrial process control toolkit. Can reinforcement learning agents be implemented as a process controller without the need for any manual tuning or the availability of a mathematical model of the process under consideration?

Some research has already been conducted in the past regarding the use of RL models in selecting and updating the PID tuning parameters [5]. Other research proposes the use of meta-learning to create controllers which can easily and automatically adjust to new and unknown processes [6]. Although these research examples address specific problems regarding the adaptive tuning of PID controllers under changing circumstances or the implementation of a generic PID controller that is able to auto-tune on a new unknown process, an interesting question remains: how would a pure RL agent compare, performance wise, to a classic PID controller in terms of responsiveness, the reduction of steady-state errors and the capability of dampening the process under impulse of any setpoint changes and/or occurring disturbances?

In this paper, we will restrict ourself to these basic performance related questions. We will start by proposing a reinforcement learning algorithm, inspired by the work of Spielberg et al. [7]. Subsequently, we will test the agent's capability of learning to control two basic and common process applications in order to finally compare the RL-agent with a PID controller, set to control these same two control applications.

2 Classical process control and PID controllers

The general objective of any process control system is to bring and maintain a given process to its desired operating conditions [3]. A frequently used setup to obtain this objective, is to establish a closed-loop feedback control system as shown in Figure 1. The closed-loop system consists of a measurement of the process variable $y(t)$ which we want to control, a reference or setpoint signal $r(t)$ which represents the desired value for the process variable, the error signal $e(t) = r(t) - y(t)$ and the manipulated variable value $u(t)$ generated by the controller. The closed-loop feedback control system will allow the controller to execute corrective actions in order to reach the desired reference value regardless of any disturbances $d(t)$ influencing the process [3]. The controller does this by using the error signal $e(t)$ as input in order to calculate a controller output signal $u(t)$ which in its turn is used to control as certain actuator in the process[1].

Standard PID controllers combine three different control actions to achieve its output signal $u(t)$, namely a proportional, an integrating and a derivative control action. The proportional action will use the error signal $e(t)$ and multiply this by a given proportional gain K_p so that a larger error signal will result in an increase in the actuator signal $u(t)$. The mere use of the proportional component will, however, not always suffice to reach the reference value. This is due to the fact that when the error signal decreases, the output signal $u(t)$ follows and therefore the feedback loop will settle to a new state of equilibrium, often with a remaining non-zero

[1]It should be noted that the actual actuator and sensor for measuring the process variable $y(t)$ are not included in the diagram. The actuator and sensor dynamics are often neglected in controller design [2] as they will be in this paper for the purpose of simplicity.

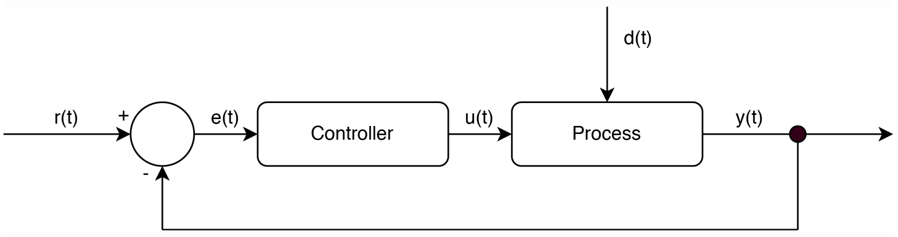


Figure 1: Schematic overview of a closed-loop feedback control system.

steady state error signal $e(t)$ [8]. To mitigate this steady state error, an integrating component with a separate integral gain K_i is added to the controller. This will allow the controller and thus the process variable $y(t)$, to close in on the reference signal until the steady state error has disappeared [8]. Finally, the derivative component with a separate derivative gain K_d, will allow the controller to anticipate to any alteration in the error signal by considering its rate of change [3]. The combination of these three components results in the standard mathematical equation for a PID controller as shown in (1).

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{de(t)}{dt} \tag{1}$$

Given this equation, tuning a PID controller comes down to selecting appropriate values for the parameters K_p, K_i and K_d. This is, however, non-trivial due to the fact that several often contradictory objectives need to be met, as shown in Figure 2.

First of all, the controller should be able to respond quickly to any setpoint changes or disturbances occurring. Furthermore, the steady state error between setpoint and process variable and the overshoot and settling time of the process variable should be limited. Finally, the controller should prevent excessive wear of the controlled equipment by avoiding excessive/aggressive alterations of the actuator signal $u(t)$ [9].

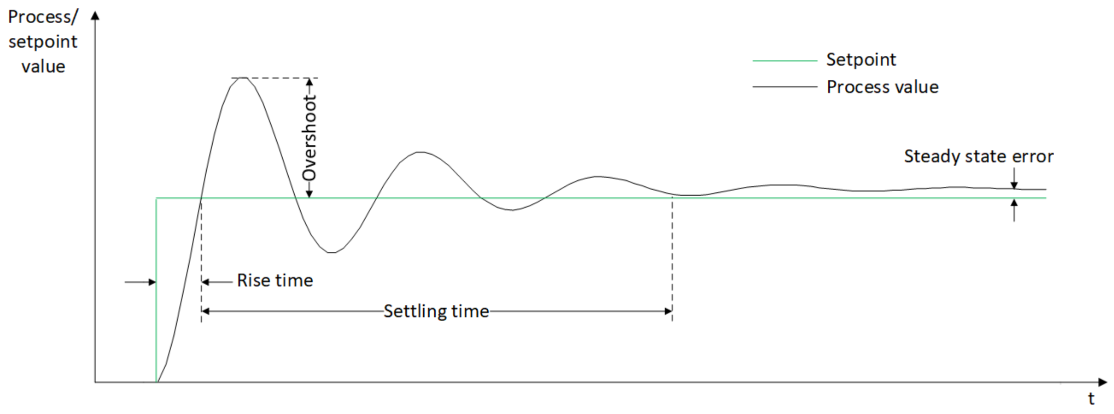


Figure 2: Visual representation of some of the controller objectives.

In order to establish satisfactory PID-parameter settings, several tuning methods have been developed over the years, from which the Ziegler-Nichols method is perhaps the best known [10]. It is, however, a well established fact that many of the PID controllers in industrial applications are tuned merely based on the experience of the engineer and/or the use of a trial-and-error approach [10], which clearly invigorates the necessity of some sort of self-tuning mechanism to reach optimal controller settings.

3 Experimental setup

3.1 Used environments

To test the agent's performance, we selected two fairly simple but common process scenarios for which both scenarios have the advantage that their dynamics are fairly intuitive. The first scenario consists of a computer model representing a cylindrical liquid tank level control, as shown in Figure 3 (left).

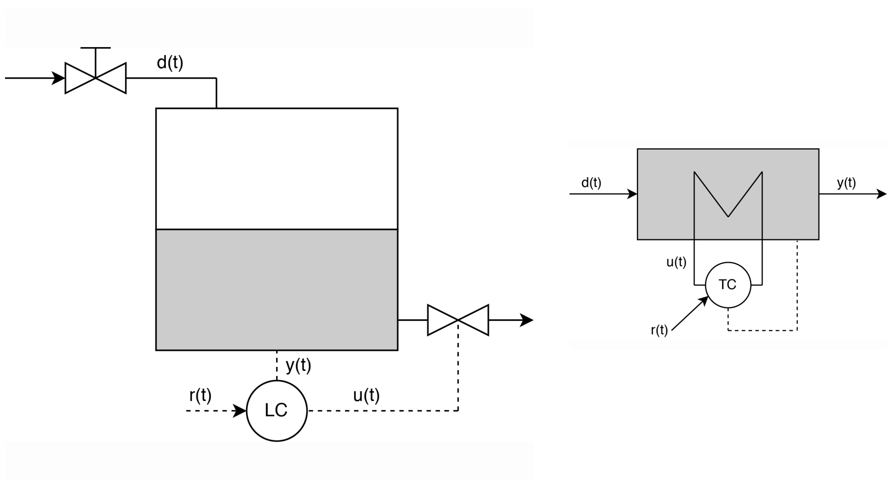


Figure 3: Schematic overview of the tank level controlling process (left) and the air heater controlling process (right).

The goal of the agent in this scenario, is to keep the liquid level in the tank[2], which acts as the process variable $y(t)$, as close as possible to a given setpoint $r(t)$ by manipulating a control valve through the actuator signal $u(t)$. The disturbance signal $d(t)$ is a manual operated valve which controls the inward flow of the liquid in the tank. The differential equation defining the change in liquid level is formulated as shown in (2).

$$\frac{d(h)}{dt} = \frac{1}{\pi r^2} \left\{ f(v_{in}) A_{in} \sqrt{\frac{2p_{in}}{\rho}} - f(v_{out}) A_{out} \sqrt{2gh + \frac{2 \triangle p}{\rho}} \right\} \tag{2}$$

In this equation the tank height and floor radius are given by the variables h and r and the input and output valve cross areas are represented by variables A_{in} and A_{out}. p_{in} is the pressure of the liquid before the input valve and $\triangle p$ is the difference between the pressure on the liquid surface in the tank and the pressure directly behind the output valve. Finally, variable ρ represents the mass density of the liquid. The equation also contains two functions $f(v_{in})$ and $f(v_{out})$ which represents the relationship between the respective actuator signal and the ratio of opening for the input and output valve. For simplicity, both valves in our environment are modelled as linear valves where the flow through the valve increases linearly with the input signal [9]. The

[2]For this specific environment the diameter of the tank was set to 2 with a height of 5. Both valves have a diameter of 0.2 and the mass density of the liquid is set to 1000.

values for v_{in} and v_{out} respectively correspond with the signals $d(t)$ and $u(t)$ in our feedback loop model. The actual liquid level in the tank, in its turn, concurs with the process signal $y(t)$.

The second scenario represents an air heater system as shown in Figure 3 (right) and is based on the example provided in [11]. The air will enter the heater with a constant flow and will be heated up by a heating coil. The heat dispersion of the coil is controlled by the agent by means of the actuator signal $u(t)$ and thus will allow the agent to achieve a required temperature for the outgoing air $y(t)$ as set by the reference signal $r(t)$. As a potential disturbance signal $d(t)$, the temperature of the incoming air can be altered. To model this scenario the differential equation, as shown in (3), is used.

$$\frac{d(T_{out})}{dt} = \frac{1}{\tau_t}\left(-T_{out} + K_h u(t) + T_{in}\right) \tag{3}$$

The variable τ_t and K_h in this equation, respectively represent the time constant of the system and the heater gain in $[°C/V]$. T_{in} and T_{out} are the temperature variables for the air entering and leaving the system for which the former equivalents to the disturbance signal $d(t)$ and the latter corresponds to the process signal $y(t)$.

For both simulations, the differential equation is discretized by using Euler's method as described in [12] by using formula (4) where Δt is the timestep or sample time.

$$\frac{dx}{dt} \approx \frac{f(x_{t+\Delta t}) - f(x_t)}{\Delta t} \tag{4}$$

At each timestep, the process will take the controller's output signal $u(t)$ as an input variable, which will in its turn, control the heating coil and thus result in a process output variable y_t.

For both scenarios, the output variable alone is not sufficient to describe the actual state of the process to the agent. Therefore, the process state is defined as a tuple $s_t = \langle y(t), \nabla y(t), e(t)\rangle$ containing the process variable value, the gradient of the process variable obtained through the differential equation and the error signal denoting the discrepancy of the process signal in reference to the setpoint signal.

Furthermore, to allow the agent to find the optimal policy, the process incorporates a specific reward function as defined by (5).

$$r(s_t, a_t, s_{t+1}) = \begin{cases} 1 \text{ if } |e(t)| < 0.01 \cdot (y_{max} - y_{min}) \\ 0 \text{ if } |e(t+1)| < |e(t)| \\ -1 \text{ otherwise} \end{cases} \tag{5}$$

The agent will, in other words, receive a reward, equal to one, if the error signal is smaller than one percent of the total range of the process variable. If this requirement is not met, the agent can still be rewarded with a reward of zero if the absolute value of the error signal decreases in reference to the previous timestep. In case none of the above prerequisites are met, the agent's reward is equal to minus one.

3.2 Agent architecture

As follows from sections 2 and 3.1, we are dealing with continuous state and action spaces. Therefore, we will use an agent with a deterministic off-policy actor-critic architecture as proposed in [13][7] and for which the schematic overview is shown in Figure 4.

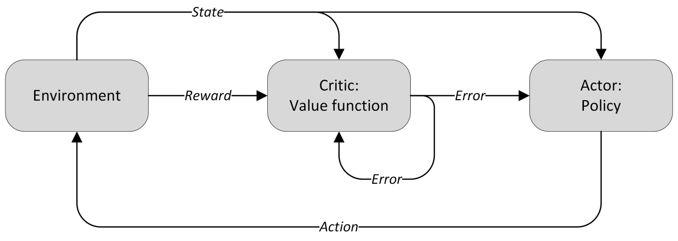


Figure 4: Schematic overview of the Actor-Critic architecture.

This architecture consists of both an actor and a critic for which the former represents the actual policy $\mu(s_t, W_a)$ and the latter represents the action-value function $Q(s_t, a_t, W_a)$. For each state, the actor will deterministically propose an action and thus an actuator signal $u(t)$ to use. The critic will, in its turn, predict the Q-value for the given state-action pair. The goal of the agent then becomes to train the actor as such that it finds the optimal policy guided by the critic who will evaluate the current policy as determined by the actor.

On top of the actor critic architecture, we will require the use of an experience replay memory (RM). As stated in [14], the temporal correlation between state transitions could lead to an increase in variance between the network updates. Therefore, we will, after each state transition, add the tuple (s_t, a_t, r_t, s_{t+1}) to the RM queue, so that a random sample of tuples can be drown from the RM queue to train both the actor and critic during each network update.

Furthermore, both the actor and critic will include an additional target network with respective weights W'_a and W'_c. These target networks will provide the main networks with a more consistent target value as used in the Bellman equation and will, by doing so, make the training of the main networks more stable. We use a specified target update rate τ to update the target networks in accordance with formula (6) and as proposed by [13].

$$\begin{aligned} W'_a &\leftarrow \tau W_a + (1-\tau) W'_a \\ W'_c &\leftarrow \tau W_c + (1-\tau) W'_c \end{aligned} \tag{6}$$

Because the actuators in both our models expect a continuous input signal which is limited between certain given boundaries (e.g. the valve opening is limited between 0 and 100 percent), our agent will have to make use of an interval action space $A = [a_L, a_H]$ for which $a_L < a_H$. To enforce these constraints, we use a clipping gradient method as proposed in [7] which clips the gradients used by the actor in accordance with the formula (7).

$$\nabla_a Q^u(s,a,w) \leftarrow \nabla_a Q^u(s,a,w) \times \begin{cases} (a_H - a)/(a_H - a_L) & \text{if } \nabla_a Q^u(s,a,w) \text{ increases a} \\ (a - a_L)/(a_H - a_L) & \text{otherwise.} \end{cases} \tag{7}$$

Using this clipping method will down-scale the gradients whenever they proceed towards the set boundaries. It will even invert the gradient if the limits would be exceeded.

Finally, we need to ensure that our agent is able to sufficiently explore the given environment. To ensure this, the agent will use a decaying Gaussian noise generator, which will generate a random value during each action selection step. This random value is then subsequently, superimposed on the action value as proposed by the actor network.

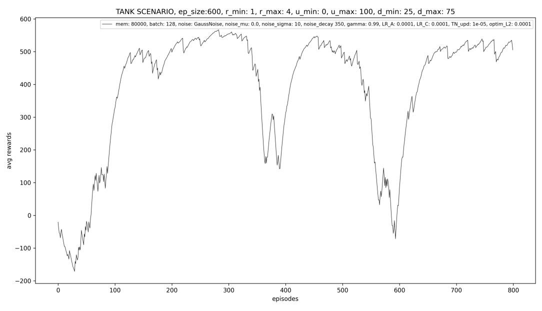
actor network arch.	3x40x30x1	critic network arch.	3x40x30x1
actor learning rate (α_a)	0.0001	critic learning rate (α_c)	0.0001
target betw. update rate (τ)	0.00001	reward discount factor (γ)	0.99
replay mem. size (K)	80000	replay batch size	128
noise μ	0.	noise σ	10

Table 1: List of agent specific hyperparameters.

4 Experimental results

In order to compare the performance of our proposed RL agent with a standard PID controller, we first need to train the agent on each of both process models. For each of the two models, we used the same proposed agent architecture with identical hyperparameters as summarized in 1. Each agent was trained during 800 episodes with an episode size of 300 timesteps for the air heater model and an episode size of 600 timesteps for the tank level model. At the beginning of each episode, the setpoint signal $r(t_0)$, the disturbance signal $d(t_0)$ and the initial process value $y(t_0)$ are set by selecting a random value from a continuous uniform distribution for each value. From that point on, both the setpoint and disturbance signal are kept constant for the complete duration of the episode. During each timestep, the agent will select an action based on the current process state and receive the new state and reward for the action taken. Both will then subequently be used by the agent to update the network weights of the actor and critic networks[3].

By performing the training as described above, we obtained an average cumulative reward plot for each of the process models as shown in Figure 5.



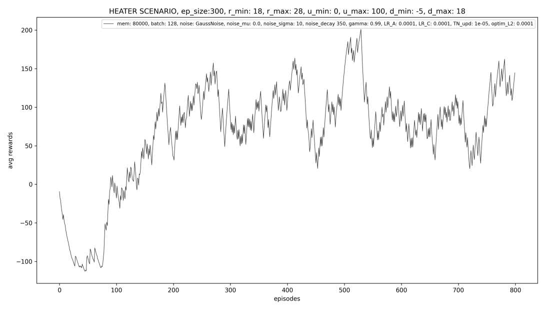


Figure 5: Cumulative average reward during the training evolution for the tank level controlling process (left) and the air heater controlling process (right).

Once trained, the agents, with their corresponding environments, were exposed to three evaluation scenarios in which the setpoint and/or disturbance signal was altered during one evaluation episode[4]. In the two first scenarios, the agent was consecutively confronted with a change in both signals by means of a step and ramp function. In the last scenario, we expose the agents to both a setpoint and a disturbance signal, generated by a non-decaying Ornstein–Uhlenbeck stochastic generator.

The results for these tests are presented on the following pages. To allow for any comparison between the agent and a PID controller, the same evaluation scenarios were conducted using a

[3]The training of both agents can be executed by running the files train_tank_scenario.py and train_heater_scenario.py.

[4]The evaluation of both agents can be executed by running the files eval_tank_scenario.py and eval_heater_scenario.py.

PID controller.

The plots clearly show how the trained agents are capable of tracking the setpoint after an initial stabilisation phase. Furthermore, the RL agents seem to outperform the classical PID controller in three of the four criteria as preconceived in 2. The RL agent is perfectly capable of quickly responding to any changes in the setpoint or disturbance signal without any significant overshoot, settling time or steady state error even when confronted with noisy setpoint or disturbance signals. The use of the RL agent does, however, come with a pressing downside. As the plots clearly show, the agent responds very aggressively to any sudden changes like noisy signals or step wise changes, as is shown in Figures 6, 8, 9 and 11. This clearly collides with our final criterium that states that a controller should avoid excessive/aggressive alterations in order to prevent any excessive wear to the controlled equipment.

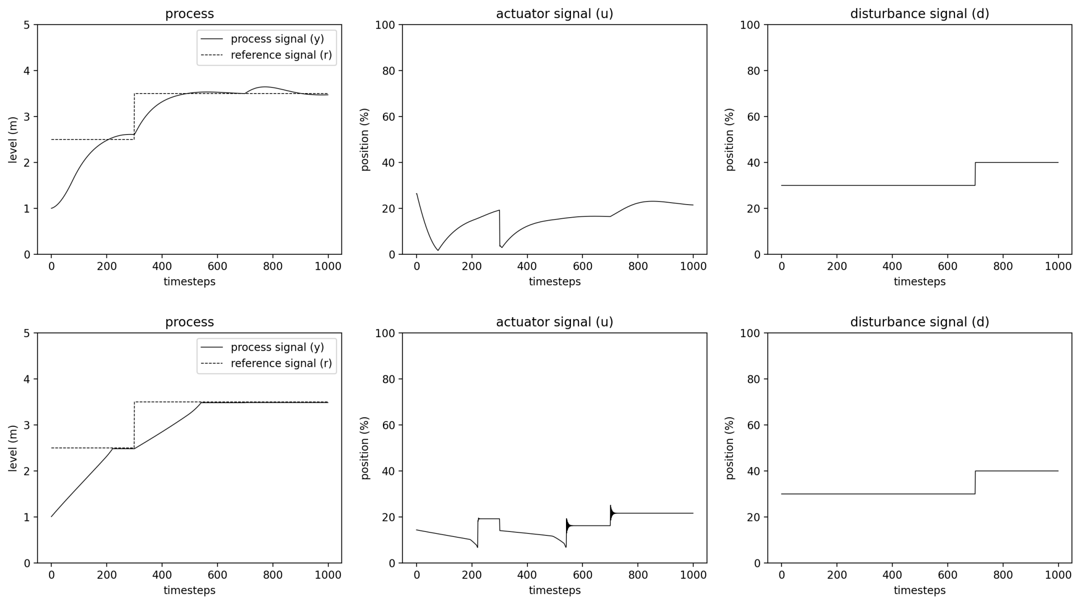


Figure 6: Results using a PID controller (top) and a trained agent (bottom) controlling the tank level controlling process with a stepwise change in both setpoint and disturbance.

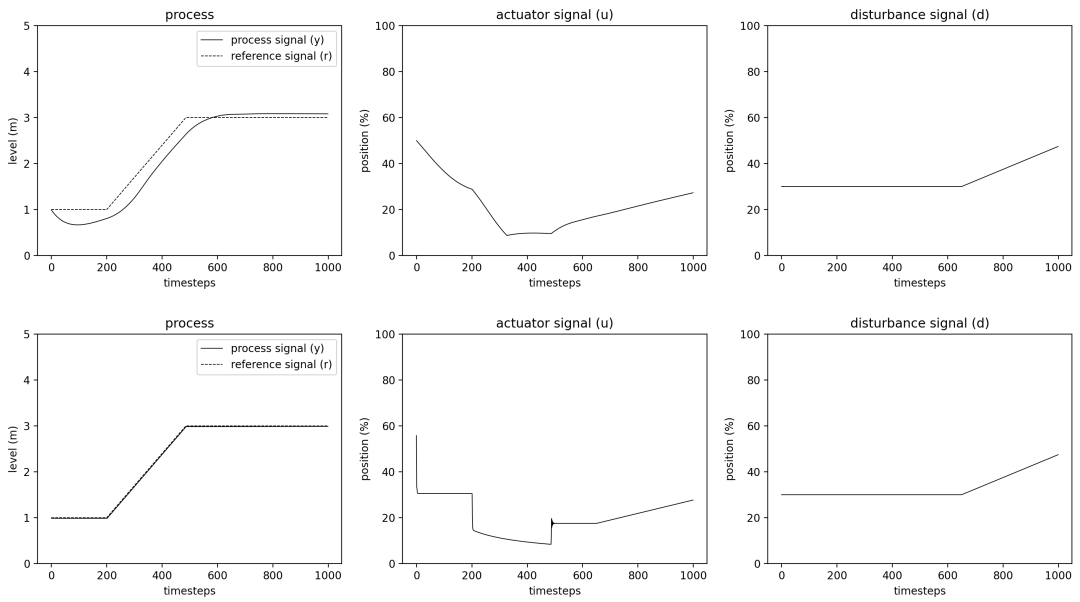


Figure 7: Results using a PID controller (top) and a trained agent (bottom) controlling the tank level controlling process with a ramp-up change in both setpoint and disturbance.

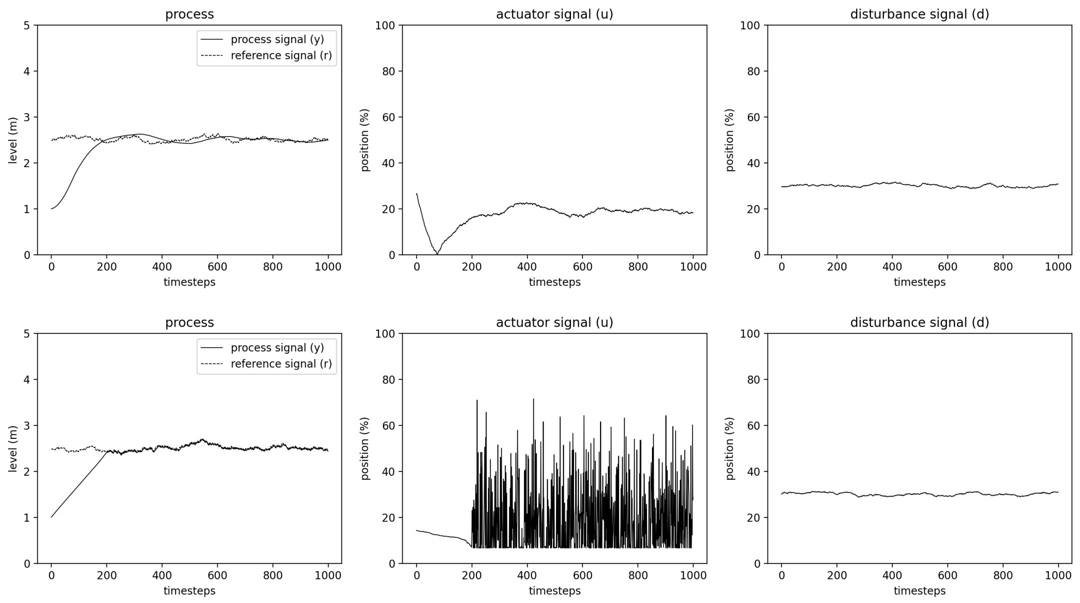


Figure 8: Results using a PID controller (top) and a trained agent (bottom) controlling the tank level controlling process with Ornstein–Uhlenbeck generated signals for both the setpoint and disturbance.

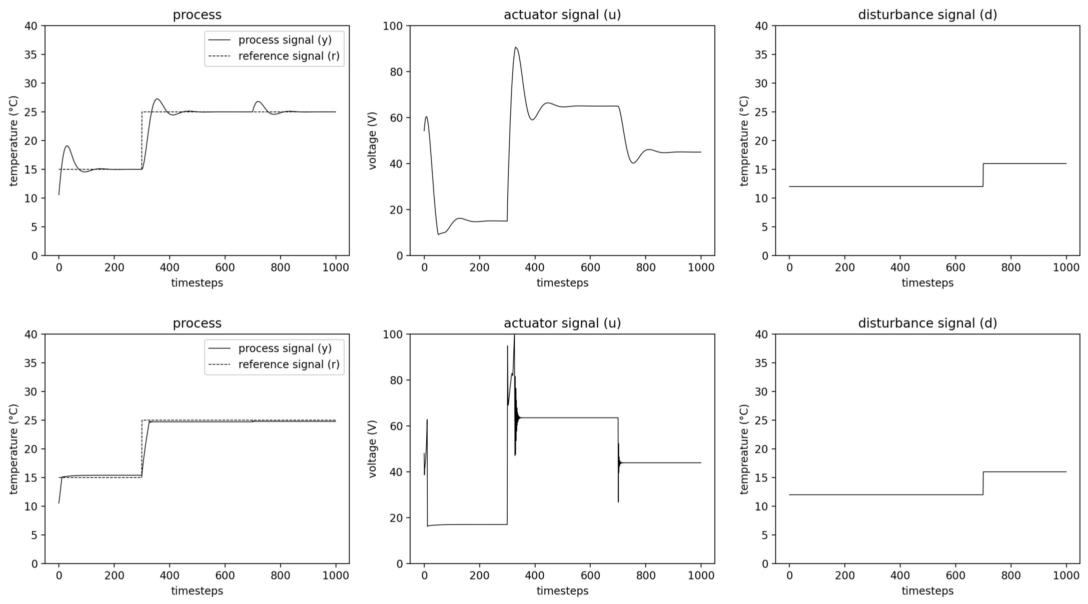


Figure 9: Results using a PID controller (top) and a trained agent (bottom) controlling the air heater controlling process with a stepwise change in both setpoint and disturbance.

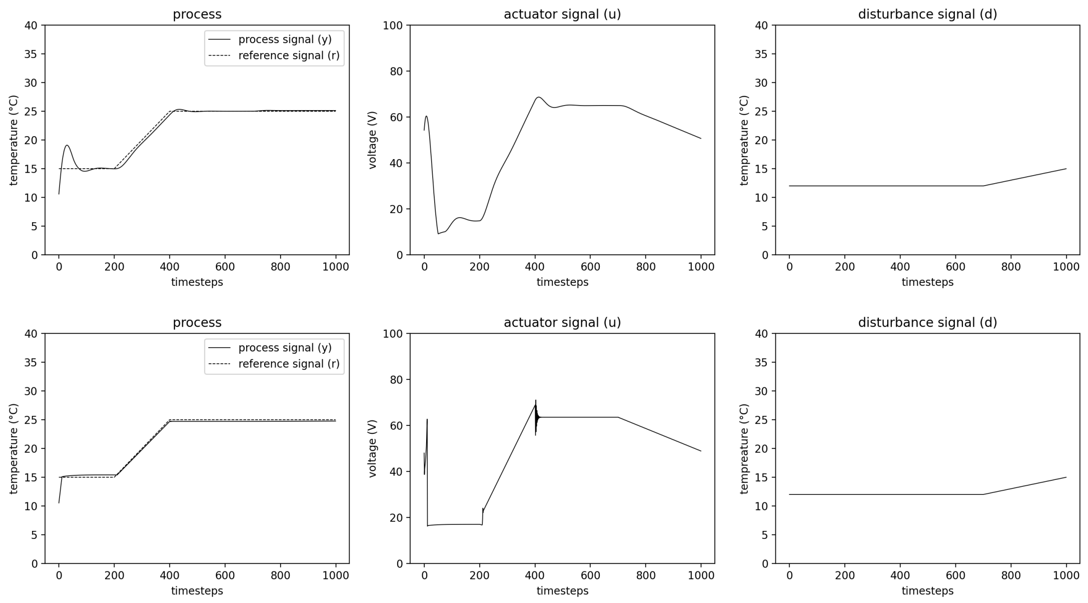


Figure 10: Results using a PID controller (top) and a trained agent (bottom) controlling the air heater controlling process with a ramp-up change in both setpoint and disturbance.

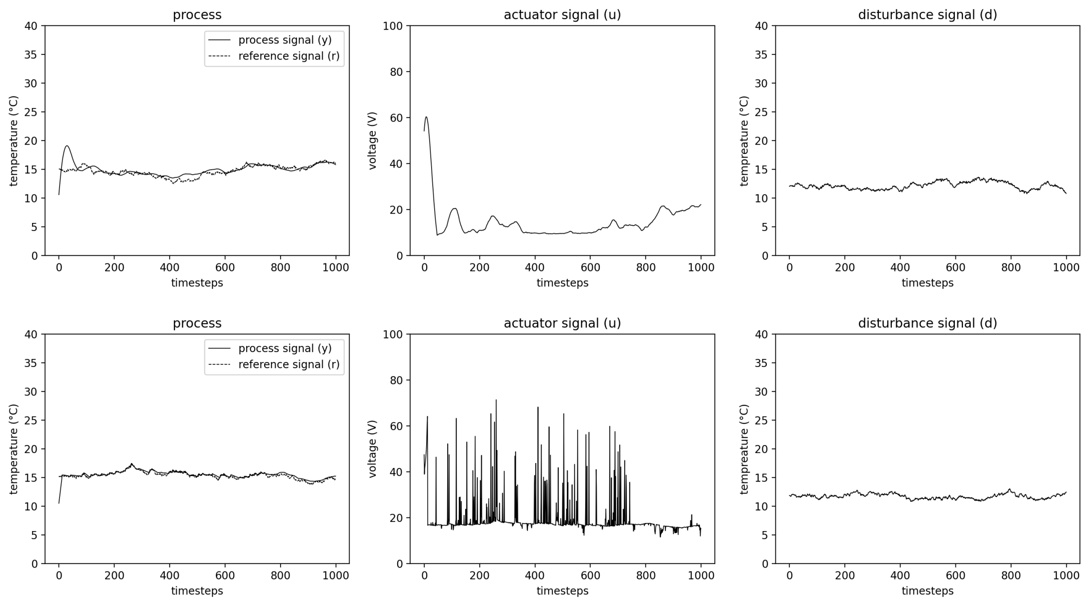


Figure 11: Results using a PID controller (top) and a trained agent (bottom) controlling the air heater controlling process with Ornstein–Uhlenbeck generated signals for both the setpoint and disturbance.

5 Conclusion

In this report we made a successful attempt at implementing a RL agent that can act as an alternative for a classic PID process controller. The results clearly show that the agent is capable of tracking a moving setpoint and can handle any disturbances in the process. Moreover, the results indicate that the RL agent is capable of outperforming a classic PID controller on most of our predefined essential criteria.

Furthermore, the training and thus tuning of the agent, did not require any mathematical model of the process or environment as is sometimes required with some PID controller tuning techniques.

The results did, however, also reveal that the agent has a tendency to react aggressively to any sudden changes in the setpoint or disturbance signal. This might become a serious impediment when the agent is required to manipulate actuators with mechanical components. Intensive and frequent movements of the actuator will after all result in an increase in mechanical wear and thus have a detrimental effect on the overall total cost of ownership.

More generally speaking, the conceptual idea of reinforcement learning, in which an agent is able to map states to specific actions in order to maximize a predefined goal, seems to resonate with many practical applications in the field of process control. The latter is however a large and diverse field with a profound track record throughout the past decades. This report does therefore not claim to be an extensive study of the subject and should be considered as merely a first attempt out of personal interest. This being said, a number of further experiments come to mind that can extend the first experiments described in this report. First of all, the processes used were chosen primarily because of their didactic qualities. The use of reinforcement learning in more comprehensive processes with multiple inputs and outputs and/or more complex objectives could be an interesting extension to these first tests. Furthermore, finding a way to mitigate the aggressive control behavior of the current agent while maintaining the beneficial qualities of the agent, would mean a definite enhancement to the current agent architecture. Finally, a major drawback for implementing reinforcement learning in real life processes, is the fact that training a reinforcement learning controller on an actual physical industrial process while assuring safety and the operational integrity of the installation may be all but a non-trivial challenge.

References

[1] R. Lucian, R. Flesch, and J. Normey-Rico. "Analysis of Anti-windup Techniques in PID Control of Processes with Measurement Noise". In: *IFAC-PapersOnLine* 51 (2018). URL: `https://www.sciencedirect.com/science/article/pii/S2405896318303975`.

[2] A. Visioli. *Practical PID Control*. 1st ed. Springer, London, 2006.

[3] D. Seborg et al. *Process Dynamics and Control*. 4th ed. John Wiley and Sons, Inc., Hoboken, NJ, 2017.

[4] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 14236 (Feb. 2015). URL: `https://storage.googleapis.com/deepmind-data/assets/papers/DeepMindNature14236Paper.pdf`.

[5] Z. Guan and T. Yamamoto. "Design of a Reinforcement Learning PID controller". In: *IEEJ Transactions on Electrical and Electronic Engineering* n/a (2021). URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/tee.23430`.

[6] D. McClement et al. "A Meta-Reinforcement Learning Approach to Process Control". In: *arXiv.org* (Mar. 2021). URL: `https://arxiv.org/abs/2103.14060`.

[7] S. Spielberg et al. "Toward self-driving processes: A deep reinforcement learning approach to control". In: *American Institute of Chemical Engineers Journals* 65 (June 2019). URL: `https://arxiv.org/abs/2004.05490`.

[8] K. Astrom and R. Murray. *Feedback Systems, An introduction for Scientists and Engineers*. 1st ed. Princeton University Press, 2008.

[9] W. Altmann. *Practical Process Control for Engineers and Technicians*. 1st ed. Newnes, Elsevier, 2005.

[10] M. King. *Process Control A Practical Approach*. 2nd ed. John Wiley and Sons, Inc., Hoboken, NJ, 2016.

[11] H. Halvorsen. *Python for Control Engineering*. 1st ed. Hans-Petter Halvorsen, 2020.

[12] W. Luyben. *Process Modeling, Simulation, and Control for Chemical Engineers*. 2nd ed. McGraw-Hill Publishing Company, 1996.

[13] T. Lillicrap et al. "Continuous control with deep reinforcement learning". In: (2019). arXiv: `1509.02971 [cs.LG]`.

[14] V. Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: (2013). arXiv: `1312.5602`.