REINFORCEMENT LEARNING-BASED PID CONTROLLER DESIGN FOR MASS FLOW

Moritz Allmendinger[1]*, Michael Erhard[1]

[1]*Research and Development, Bürkert Fluid Control Systems, Christian-Bürkert-Straße 13-17, 74653 Ingelfingen*

* Corresponding author: Tel.: +49 7940 10 96 269; E-mail address: moritz.allmendinger@burkert.com

ABSTRACT

This paper demonstrates a model-based control synthesis strategy based on artificial neural networks and reinforcement learning. For this purpose, both the determination of the systems' response as well as the training of the neural network are transferred to a virtual environment. The neural network acting independently but interacting with a conventional PI controller, is optimized in order to achieve the predefined control target. The definition of the control target and the evaluation of the control response are compared with each other in the time domain enabling a flexible integration and adaptation to a wide range of possible requirements. The results illustrate the feasibility of control synthesis based on a virtual trained neural network. Considering variations and uncertainties for the control target and the environment, the neural network should become more robust and suitable for real systems with inherent deviations between each other.

Keywords: Reinforcement Learning, Simulink, Mass Flow Control, PID

1. INTRODUCTION

Fermentation is the conversion of organic substances with the help of bacteria, fungi or cell cultures in admixture with enzymes. The process is used in a variety of industries, from food and beverage to pharmaceutical and chemical, as well as in research laboratories or pilot plants. To achieve optimum fermentation results, precise control of all process parameters is essential. The type and concentration of nutrients, the temperature, the oxygen content, and the pH-value in the fermenter are crucial parameters for this. Up to 4 fermentation gases are used inside the fermenter to control the fermentation process: oxygen (O_2), nitrogen (N_2), carbon dioxide (CO_2) and air (21% oxygen and 79% nitrogen). Reproducible processes and repeatable product quality require precise control of these gases. For this reason, mass flow controllers (MFC's) are used to control the gas supply and exhaust.

Proportional-integral-derivative (PID) controllers are the most widely used control algorithms in the industry. The control strategy of MFC's for gaseous media used in fermentation processes is also based on the proven PID approach. Although classical PID controllers require only a small set of parameters to set up the control action, successfully tuning them can be a challenging task, especially in the presence of dominant and state-dependent nonlinearities. This results in a time-intensive optimization of the PID parameters to achieve an overall stable and robust control performance.

This paper examines whether neural networks and reinforcement learning (RL) can simplify standard PID design and enhance PID control performance. Against this background, the control action of a conservatively parameterized standard PID controller is extended by a further component that originates from a neural network. For this purpose, both the determination of the systems' dynamic response as well as the training of the neural network are moved to a virtual environment. This

approach needs a dynamic simulation model of the system modeling all significant physical effects of the MFC: nonlinear characteristic curves including actuator hysteresis and event-driven changes in system dynamics with respect to system state.

The paper is structured as follows: section 2 provides an overview about reinforcement learning, the used agent, and the training methods. The description of the simulated application is presented in section 3. Section 4 explains the training setup for the agent used in reinforcement learning. The results obtained are shown in section 5 and finally summarized in section 6.

2. REINFORCEMENT LEARNING – BASICS AND FRAMEWORK

2.1. Reinforcement Learning

Deep Learning requires labeled data to train a neural network. However, it is important to note that a neural network will never perform better on the test data than on the training data. This problem can be avoided with RL, which is based on trial and error.

The basic elements of RL are an agent and an environment. The agent observes a continuous-in-value state $s(t_k)$ of the environment at a discrete time t_k. Based on the recent policy $\pi(t_k)$, the agent performs a continuous action $a(t_k)$ out of space A of possible actions. This leads to a new state $s(t_{k+1})$ of the environment. The reward $r(t_{k+1})$ received for the new state informs the agent about the quality of its action so that it can adjust its policy accordingly. The agent's goal is to maximize the cumulative reward.

The breakthrough of RL came in 2013 when the Deep Q learning algorithm developed by Deepmind outperformed the best players of the Atari game [1]. Since then, the RL approach has made further progress. A variety of new techniques and algorithms still exist today. Research is also increasingly focusing on using RL in an industrial control context. Two relevant contributions are described below, with overlapping content in some areas.

As a nonholonomic system with measurement noise and external disturbances, controlling a robotic system is challenging. In their work [2], Gheisarnejad & Khooban demonstrate a possibility of how a RL agent (here the Deep Deterministic Policy Gradient (DDPG)) can assist a conventional controller in trajectory control. Therefore, the control action of the PID controller is manipulated by a superimposed action selected by the DDPG. During training sequences, this approach can compensate for uncertainties and disturbances in the trajectory control to ensure more robust control. The advantages of this approach are highlighted by a comparison between the RL-based PID controller and the previously used controller. Siraskar's article [3] investigates the use of the DDPG algorithm as a flow control algorithm with nonlinear valves. The trained DDPG is compared to a PID controller. The data indicates that the DDPG controller more accurately tracks the command signal compared to the PID controller. In contrast, the PID controller shows superior performance in suppressing disturbance variables.

A literature review suggests that combining PID control with RL-based control enhances the performance of this hybrid control system. This approach has the potential to improve the control performance for various problems, such as regulating gas mass flow. An RL-based PID controller could combine the positive results of the agent's control performance with the suppression of disturbances by the PID controller, thus also improving the results obtained by Siraskar. With reference to both publications and the DDPG used therein, this RL algorithm is also considered suitable for the problem dealt with here.

2.2. DDPG

Lillicrap et al. [4] developed the Deep Deterministic Policy Gradient algorithm for solving problems in environments with continuous-in-value states and action spaces. This previously mentioned algorithm combines the Deterministic Policy Gradient and Deep Q-Learning agents. The DDPG is based on the Deterministic Policy Gradient, as it can be used for continuous action spaces, and is extended by the techniques of the replay buffer and the updating of networks from Deep Q-Learning.

2.3. Training

There are two ways to train an RL algorithm. One way is to train it against a real environment. Therefore, the RL algorithm has to be executed on the intended real-time hardware and must be connected to the environment physically. The advantage of this approach is the existence of a realistic environment, ideally the environment into which the agent will later be integrated. The training with a real environment should cover all effects that may occur during the subsequent operation. In doing so, the agent is optimally trained for the later field of application. Computing power is often the main problem with this training method. Compared to a high-performance computer, embedded systems have significantly less memory and computing power on the target hardware. Another disadvantage is the possible damage of the hardware used during training. It is unavoidable that hardware used in the system is subjected to high loads due to the large number of training sessions or operating points within limit ranges, which lead to increased wear and tear.

For this reason, the model-based training approach is becoming increasingly popular. Compared to the previously described approach, the model-based training strategy relies on a simulated environment. This solves the problems that occurred during training with the real system – limited computing power together with increased wear and tear and sometimes even potentially dangerous conditions depending on the application. The performance of the computing hardware used for training can be easily adapted to the requirements of the training algorithm reducing overall training time. In addition, there is no hardware wear in a simulation model. Both effects jointly reduce the costs of training. In the model, the environment can always be parameterized in the same way making a reproducibly testing and comparison of the several trained networks easy. In this way, parameters and states in the model can be flexibly isolated and evaluated. This is only partially possible when training on the real system.

MATLAB and Simulink in Release R2023a are selected as simulation environment. The Reinforcement Learning Toolbox of MathWorks makes several implemented RL agents available, including the DDPG algorithm. The architecture of the model used for simulation of controller, agent and environment is shown in **Figure 1**. The plant model is described in section 3, RL setup, agent and controller parameterization are discussed in section 4.

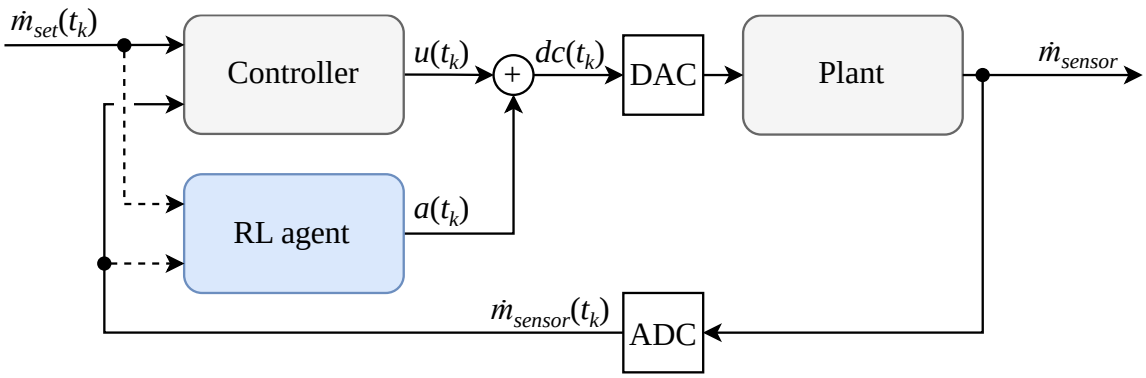


Figure 1: Structure and main components of training model

3. PLANT MODEL

This section deals with the plant model visualized in Figure 1 as counterpart to the controller and RL agent. For reasons of simplicity, the entire fermentation process is reduced to boundary conditions acting at the inlet and outlet of the MFC. Thus, the plant model only includes the physical effects inside the MFC, interacting with the application through (constant) boundary conditions. As shown in **Figure 2**, a MFC consists of 3 main components – an electronic control unit, a proportional valve, and a flow rate sensor. Because the controller is part of the control unit and modeled together with the RL agent in a discrete manner, the physical plant model reduces further. In the end, the modeling task will only include the valve and the sensor.

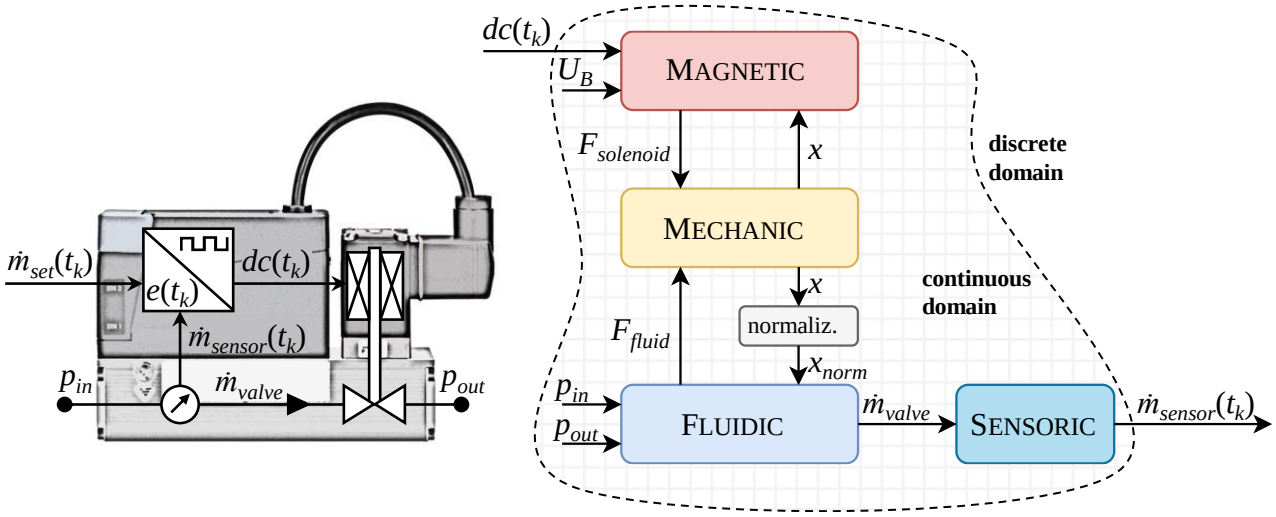


Figure 2: Mass flow controller layout (left), detailed signal flow interface chart for continuous plant model (right)

Taking a closer look at the time-continuous plant model, the governing equations from first principles must describe several physical domains and their interactions as indicated in Figure 2. Hereby, the modeling of the dynamic response of the proportional valve is the largest subtask, which will be discussed in more detail in the sections below.

The electro-magnetic actuator is made from soft-magnetic material excited by a current-driven coil. All relevant data for this solenoid is calculated with the help of finite element simulation yielding the nonlinear current inductance $\partial\psi/\partial i_{coil}|_{x,i}$ and the moving inductance $\partial\psi/\partial x|_{x,i}$ as well as the corresponding reluctance force map $F_{solenoid}$. The underlying equations (1) and (2) describe the current and force behavior for the electro-magnetic actuator.

$$\frac{\partial i_{coil}}{\partial t} = \frac{1}{\frac{\partial \psi}{\partial i_{coil}}} \cdot \left(dc \cdot U_B - R_{coil} \cdot \left(i_{coil} + i_{eddy} \right) - \frac{\partial \psi}{\partial x} \frac{\partial x}{\partial t} \right) \tag{1}$$

$$F_{\text{solenoid}} = f\left(i_{\text{coil}}, x \right) \tag{2}$$

Considering Figure 2 again, there is a nonlinear time delay between the applied duty cycle dc of the PWM excitation and the coil current or solenoid force, respectively. The supply voltage U_B for the PWM-controlled coil acts only as a constant boundary condition and is not changed during the different training episodes. An additional eddy current model fitted with transient finite element results completes the underlying dynamic simulation model of the actuator.

On the fluidic side, the modeling is limited to the description of the mass flow $\dot{m}_{valve}$ as a function of valve stroke x and input or output pressures p_{in}, p_{out} applied. It is known from ISO 6358 that the upstream pressure p_{in} together with the pressure ratio p_{out}/p_{in} is sufficient to calculate the mass flow through a restriction. The dependence from valve opening is implemented through the scaling of the conductance $C(x_{max})$ for the fully opened valve with the normalized valve stroke x_{norm}. Above the critical pressure ratio b, mass flow changes with outlet pressure as equation (3) indicates. Below the critical pressure ratio, the flow is saturated and mass flow depends only on upstream pressure, see equation (4). The missing parameters such as the conductance, the critical pressure ratio or the valve stroke normalization are determined from flow measurements at different valve openings.

$$\dot{m}_{valve} = x_{\text{norm}} \cdot p_{\text{in}} \cdot C(x_{\text{max}}) \cdot \rho_0 \cdot \sqrt{\frac{T_0}{T_{\text{in}}}} \sqrt{1 - \left(\frac{\frac{p_{\text{out}}}{p_{\text{in}}} - b}{1 - b} \right)^2} \tag{3}$$

$$\dot{m}_{valve} = x_{\text{norm}} \cdot p_{\text{in}} \cdot C(x_{\text{max}}) \cdot \rho_0 \cdot \sqrt{\frac{T_0}{T_{\text{in}}}} \tag{4}$$

In addition to these descriptions resulting from the state-of-the-art, the flow forces play an important role. On the one hand, an increase in mass flow results in an additional upstream pressure drop reducing the nominal lift force. On the other hand, the pressure profile interacting with the valve spool is highly velocity-dependent, resulting in an additional deviation form nominal lift force assumptions. Without significantly increasing the measurement effort, the same procedure as for the actuator is used, and the missing force characteristic map F_{fluid} is calculated via computational fluid dynamics.

$$F_{fluid} = f\left(\frac{p_{out}}{p_{in}}, x\right) \tag{5}$$

The translational movement of the valve spool connects the electro-magnetic with the fluidic domain. From balance of forces as shown in equation (6), the position x and velocity $\dot{x}$ are determined. These interface variables complete the calculations in the adjacent domains.

$$\frac{d^2x}{dt^2} = \frac{1}{m} \cdot \left(F_{\text{solenoid}} - F_{\text{fluid}} - F_{\text{friction}} - F_{\text{stop}} \right) \tag{6}$$

Due to the thermal measurement principle for the mass flow, a first order time delay is used as a simplified sensor model, see equation (7).

$$\frac{\partial \dot{m}_{sensor}}{\partial t} = \frac{1}{T} \cdot (\dot{m}_{valve} - \dot{m}_{sensor}) \tag{7}$$

With all model equations and parameters in hand, a complete model verification is performed for each domain as well as for the entire system. Without going too much into details, a final verification of the fluidic performance dependent on electrical excitation is highlighted in **Figure 3**. The measured mass flow is compared to the calculated mass flow. The two values are in good agreement.

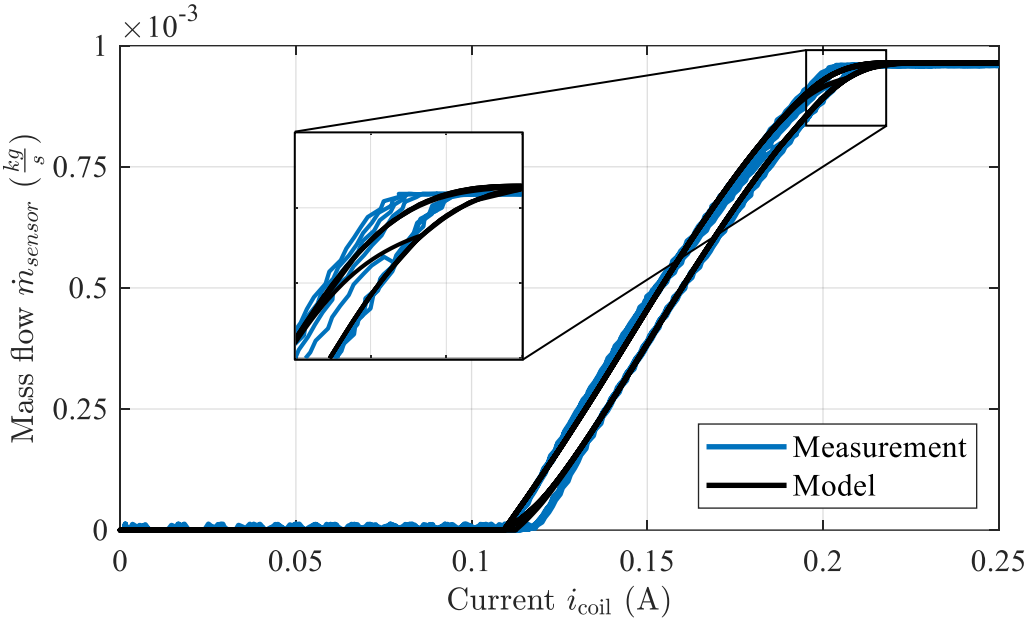


Figure 3: Verification of measured mass flow behavior $\dot{m}_{sensor} = f(i_{coil})$

The verification results suggest that the dynamic simulation model is suitable for the reinforcement learning approach as a counterpart to the real world.

4. REINFORCEMENT LEARNING AGENT AND CONTROLLER

This section describes the setup of the RL agent and the PID controller. As shown in Figure 1, the DDPG adjusts the control signal of the controller. Recalculation of the valve's duty cycle dc takes place at fixed points t_k in time according to equation (8).

$$dc(t_k) = a(t_k) + u(t_k) \tag{8}$$

Accordingly, the parameters of the PI controller remain constant and so does the closed-loop control behavior. The agent interacts with the control-loop at the same point as a feedforward control, but its action is generated in a different manner. In general, the control action of the agent accelerates or delays the opening of the valve depending on the current system status.

4.1. Observation Vector

The state observed by the agent consists of the following elements listed in equation (9). The control error $e(t_k)$ and the actual mass flow $\dot{m}_{sensor}(t_k)$ provide information for the control. The previous control error $e(t_{k-1})$ contains information about the dynamics of the system. Observing the action $a(t_k)$ informs the agent about its own behavior.

$$s(t_k) = \left[e(t_k), e(t_{k-1}), \dot{m}_{sensor,norm}(t_k), a(t_k)\right]^T \tag{9}$$

It is not necessary to include a history of the control error. The current duty cycle can also be omitted. Both approaches were examined in advance and showed no positive effects. Therefore, only the current action remains as part of the state vector. For training purposes, the values for the setpoint $\dot{m}_{set}(t_k)$ and actual value $\dot{m}_{sensor}(t_k)$ are scaled to a range between $[0,1]$. This ensures that all values of the observation vector are in a similar range of values.

4.2. DDPG Agent Networks

The used configuration of the actor and critic networks of the DDPG agent is shown in **Figure 4**. Both networks use fully connected (FC) layers as hidden layer and a ReLu layer as activation function. The concatenation layer inside the critic network combines the current state $s(t_k)$ with the action $a(t_k)$. The actor network uses a tanh-layer as the last layer to limit the value range of the action $a(t_k)$ between $[-1,1]$.

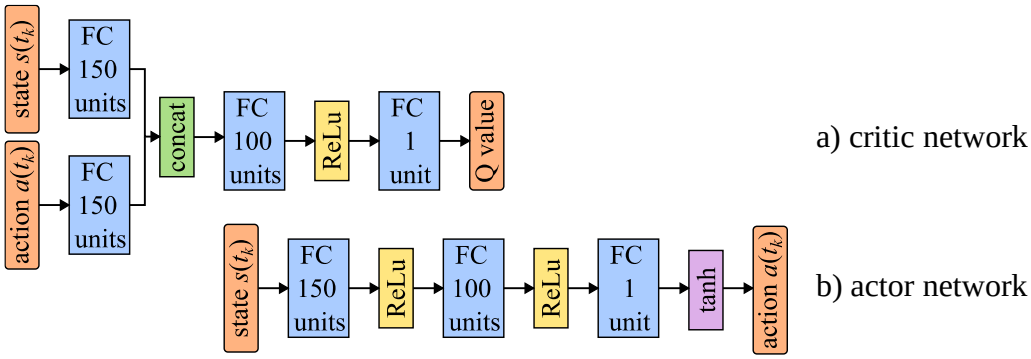


Figure 4: Network configuration of DDPG agent: a) critic network and b) actor network

The action $a(t_k) = 1$ opens the valve completely, irrespective of the PID control action. Conversely, this applies also to an action value $a(t_k) = 0$, the valve is then always closed.

4.3. DDPG Agent Hyperparameters

The hyperparameters of the agent and its training configuration are listed in **Table 1**. The DDPG uses the Ornstein-Uhlenbeck (OU) process as a noise model to solve the problem between exploitation and exploration.

Table 1: Hyperparameter of DDPG agent and training

Hyperparameter	**Unit**	**Setting**
Sample time	ms	10
Steps per episode	-	150
Discount factor	-	0.99
Replay buffer size	-	$1 \cdot 10^6$
Batch size	-	64
Optimizer	-	Adam
Smoothing factor	-	$1 \cdot 10^{-3}$
Learning rate - critic	-	$1 \cdot 10^{-3}$
Learning rate - actor	-	$1 \cdot 10^{-4}$
OU standard deviation	-	0.2
OU standard deviation decay rate	-	$5 \cdot 10^{-5}$

4.4. Reward

The reward function used is described in equation (10). The reward function consists of two parts: the first part rewards the control error, and the other part rewards the reduction in the change in the action signal. The first part aims to teach the agent to minimize the control deviation.

$$r(t_k) = -\sqrt{|e(t_k)|} - 0.5 \cdot \sqrt{|a(t_k) - a(t_{k-1})|} \tag{10}$$

First training results only including the control error showed that the agent's action tends to oscillate. For this reason, further investigations were necessary leading to the integration of the second part shown above. The agent should minimize the control error by making as few changes as possible to

the action signal $a(t_k)$. Weighting factors ensure that the agent's focus remains on minimizing the control error, which is why the second part is only weighted by half.

4.5. Episode Management

The scaled mass flow setpoint of each training episode is chosen randomly between $[0.05,1]$. For the dynamic simulation model of the plant, the initial condition of the valve opening is always fully closed meaning that there is no mass flow. Other environmental conditions such as inlet pressure and heating are not initially varied. The boundary conditions for the MFC model are fixed with an inlet pressure of 6 bar and with a solenoid temperature of 20°C. These fixed environmental conditions simplify the complexity of the training.

5. RESULTS

In this section, the performance of the trained DDPG-PI controller is evaluated using the simulation model of the MFC. **Figure 5** shows the control result of the DDPG-PI controller (solid lines) compared to the result of the conventional PI controller (dotted lines) for different setpoints (dashed lines). The PI controller is usually optimized for a specific setpoint using an autotune algorithm. In this case, the PI controller was optimized for the middle of the normalized operating range. Due to the nonlinear plant characteristics (overcoming the spring preload), settling times and overshoots in other operating points should differ from the tuned configuration.

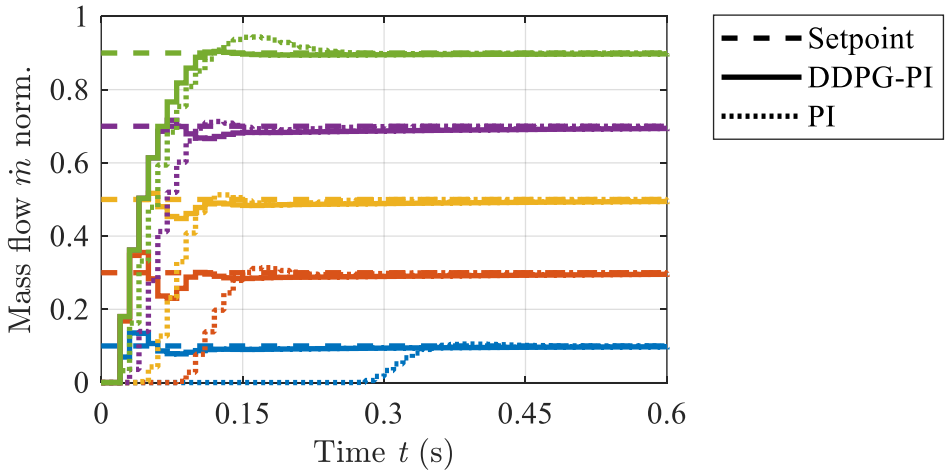
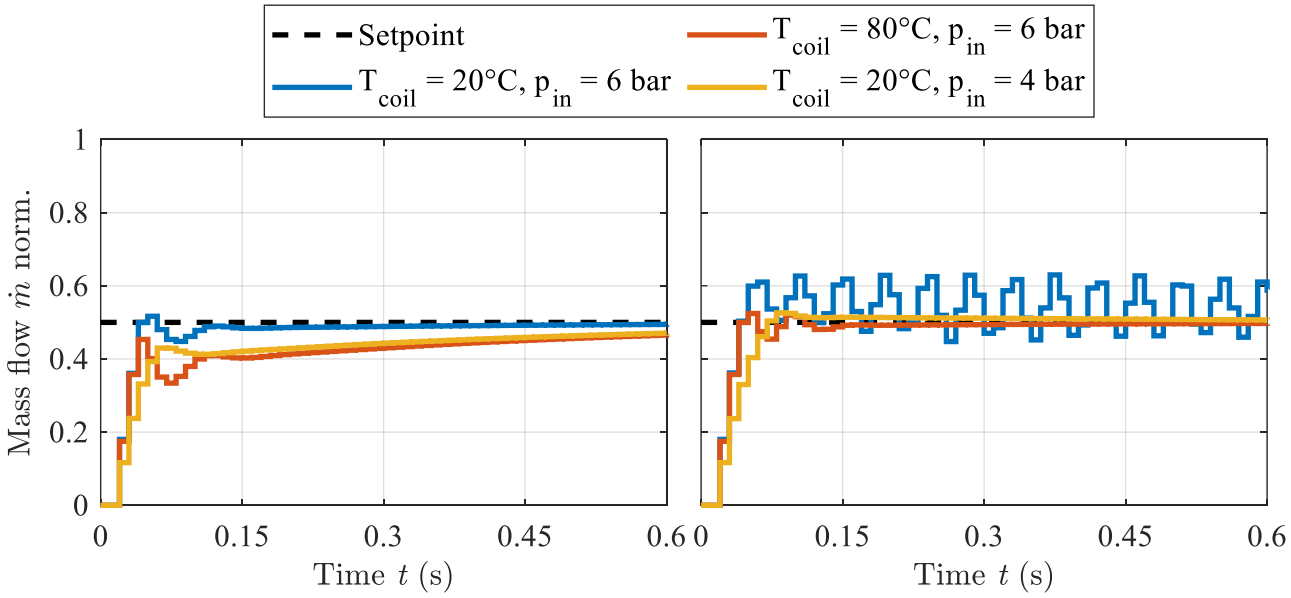


Figure 5: Comparison of the control performance between DDPG-PI and conventional controller

Compared to the conventional PI controller, the control performance of the DDPG-PI controller performance is significantly better in terms of settling times. This is not unexpected, as the agent's interaction with the control loop can be like the behavior of a static feedforward action. With the agent, all settling times are shorter regardless of the setpoint value. Particularly with low set values, the reduced settling time is linked to larger overshoots. However, these are acceptable as they do not damage the system. Because all mass flow transients start in the same way, it can be assumed that the agent has learned the nonlinearity of the plant and has thus implicitly linearized it.

Environmental conditions may vary during the use of MFCs. The capability of the DDPG to handle untrained system states is therefore being investigated. **Figure 6** a) shows the control behavior that results when the DDPG agent has to deal with untrained changes in the environmental conditions. In the fixed environment, the boundary conditions are 6 bar for the inlet pressure and 20°C for the coil temperature. This control result is displayed as blue line. The red line corresponds to the case, where the coil temperature inside the plant model has changed from 20°C to 80°C. The orange line

corresponds to the case in which the inlet pressure of the model has been reduced to 4 bar. With the help of the PI controller, the agent always reaches the setpoint regardless of the changes made. The integral component of the controller is mainly responsible for this because it minimizes the control deviation continuously. However, the settling time for reaching steady state has increased for both variations of the environmental conditions. To achieve the same mass flow at reduced inlet pressure, the valve must be opened further. A rise in coil temperature also delays the valve opening. The integral part requires more time to overcome this.



a) MiL inference with untrained environmental variations

b) MiL inference with trained environmental variations

Figure 6: Control result of DDPG agent with a) untrained and b) trained environmental variations

Different variations of the environmental conditions are part of the further training sequences to improve the control results. The inlet pressure and coil temperature are randomly selected from the following range of values at the start of each training episode: inlet pressure 3…6 bar, coil temperature 20…80°C. Figure 6 b) shows the control results for the same environmental variations as in Figure 6 a), with the difference that the environmental variations were now part of the training. The settling time of the mass flow, especially the last few percent before reaching steady state, has improved significantly. For all variations, the setpoint is reached within 100 ms. However, the robustness of the RL-PI controller has decreased. Although the RL-PI controller can robustly control the mass flow in the first inference at 20°C coil temperature and 6 bar inlet pressure, it is no longer capable in this inference. The mass flow oscillates around the setpoint after it has been reached.

6. CONCLUSION

As part of a feasibility study on RL agents for control engineering problems, this paper demonstrates the possibility of control synthesis based on a virtually trained neural network with RL for mass flow control. The DDPG agent is used as RL algorithm and extends the conventional PI controller by manipulating its control action. The training of the DDPG-PI controller is shifted to a simulation environment while retaining the advantage of high computing power and avoiding the disadvantage of system wear and tear. Therefore, a dynamic simulation model of a MFC is created using MATLAB, Simulink and the RL Toolbox. The DDPG-PI controller is validated using the simulated system. The findings reveal that a DDPG-PI controller improves the control result for different setpoints compared

to a conventional PI controller with fixed parameters. In particular, the settling time is reduced for all setpoint values. Of course, more advanced, e.g. model-based control approaches are also suitable for achieving such improvements.

Tolerable variations in the ambient conditions are also adequately handled by the RL-based controller due to the continuous action of the (fixed) PI controller. Extending the training sequences to include these variations improves the settling time, but also leads the DDPG-based controller losing stability. Further considerations are required to maintain robustness under variable environmental conditions, especially with regard to the signals to be included into the observation vector.

NOMENCLATURE

a	Action of agent	-
e	Control deviation normalized	-
$\dot{m}$	Mass flow	kg/s
p	Pressure (absolute)	Pa
r	Reward	-
s	State of environment	-
T	Temperature	°C
DDPG	Deep Deterministic Policy Gradient	
MFC	Mass flow controller	
PID	Proportional-integral-derivative	
RL	Reinforcement Learning	

REFERENCES

[1] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M A (2013) Playing Atari with Deep Reinforcement Learning. Computing Research Repository (CoRR): abs/1312.5602

[2] Gheisarnejad M, Khooban M H (2021) An Intelligent Non-Integer PID Controller-Based Deep Reinforcement Learning: Implementation and Experimental Results. IEEE TRANSACTIONS IE 68(4):3609-3618

[3] Siraskar R (2021) Reinforcement Learning for Control of Valves. Machine Learning with Applications 121–134 68(4):3609-3618

[4] Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D., Wierstra D (2019) Continuous control with deep reinforcement learning.