Improving a Proportional Integral Controller with Reinforcement Learning on a Throttle Valve Benchmark

Paul Daoudi[1], Bojan Mavkov[2], Bogdan Robu[3], Christophe Prieur[3], Emmanuel Witrant[3], Merwan Barlier[1] and Ludovic Dos Santos[4]

***Abstract*— This paper presents a control strategy for a non-linear throttle valve with an asymmetric hysteresis, leading to a near-optimal controller without requiring a priori knowledge about the environment. We start with a carefully tuned Proportional Integrator (PI) and leverage the recent advances in Reinforcement Learning to improve it thanks to additional interactions with the valve. We test the proposed controllers in a variety of scenarios, all highlighting the benefits of combining both frameworks for improving control performance in non-linear stochastic systems. The resulting agent has a better sample efficiency than traditional Reinforcement Learning agents and outperforms the PI controller in all the tested use cases.**

I. Introduction

Throttle valves are essential components in various industrial processes such as chemical plants, oil refineries, and power generation. Accurate control of these valves is crucial for maintaining the optimal flow rate of fluids and thus ensuring the efficient functioning of the entire system. However, controlling throttle valves is complicated due to their non-linear behavior, stochasticity, and asymmetric hysteresis. These factors make it challenging to design an optimal controller that can handle the complexities of the valve system.

In this work, we focus on the control of a butterfly valve for vehicles, which regulates the air supply in the combustion chamber by adjusting the rotation of a disk. In order to deal with the static friction effects and the non-linearities of the model, different advanced control strategies have been proposed by the control community. In [1], an adaptive pulse control strategy is developed using a non-linear dynamic model including friction and aerodynamic torque. Discrete-time sliding mode control designed to realize a robust tracking control is presented in [2]. A non-linear control strategy consisting of a Proportional-Integral-Derivative (PID) controller and a feedback compensator is proposed in [3]. Adaptive control strategies are presented in [4], [5], and Linear Parameter-Varying (LPV) modeling and mixed constrained H_2/H_∞ control in [6]. Several works for the control of electric throttle valves already exploit deep learning techniques in their implementation. For example, in [7] a neural network-based sliding mode controller for electronic throttle valves is presented. A control strategy using Reinforcement Learning (RL) algorithms is already presented in [8] where the valve is controlled using the Deep Deterministic Policy Gradient (DDPG) [9] algorithm. However, the algorithm is applied only in a simulator and they do not go beyond the classical DDPG agent.

We propose to take one step further and study the combination of classical control strategies with Reinforcement Learning for the throttle valve. This framework is popular due to its ability to solve complex non-linear control problems without requiring an explicit model of the system [10], [11]. However, despite some exceptions such as balloon navigation [12] and plasma control in Tokamaks [13], RL has only been applied in simulated systems [14], [15], including the aforementioned work [8]. One major obstacle in directly applying RL to real-world systems is the need for a large amount of data and repetitive experiments to learn a good policy, as RL agents start in an unknown environment with no external information available [16].

Only a few recent works started to combine Optimal Control (OC) and RL [17] to reduce the number of data required by RL agents to learn a good policy. For instance, [18] and [19] combined them by proposing a switching mechanism between an LQR controller, and these approaches were only tested in simulated systems such as Pendulum and Acrobot, not in real environments. Similar to these previous methods, we aim to combine the strengths of OC and RL to build a controller for the throttle valve. For this, we adapted the recent area of Reinforcement Learning with Guides [20], [21], [22] to our setting, resulting in an algorithm that leverages a Proportional Integral (PI) controller to guide the RL agent in navigating the control space efficiently. By reducing the search space for learning, the PI guide minimizes the data requirements and accelerates the learning process.

We conducted empirical validation of our approach on the throttle valve by first testing the PI controller, a state-of-the-art RL agent, and their combination under a variety of industry-relevant use cases. Our results demonstrate that the combination of PI and RL produced the most interesting agent, providing a near-optimal controller with fewer data samples than the conventional RL agent.

The paper is structured as follows. We begin by detailing

[1]Paul Daoudi and Merwan Barlier are with the Noah's Ark Laboratory of Huawei Technologies, Paris, France. Email: paul.daoudi1@huawei.com and merwan.barlier@huawei.com

[2]Bojan Mavkov is with Université Côte d'Azur, CNRS, I3S, Nice, France. Email: bojan.mavkov@univ-cotedazur.fr

[3]Bogdan Robu, Christophe Prieur and Emmanuel Witrant are with the GIPSA-Lab, CNRS, Université Grenoble Alpes, Grenoble, France. Email: bogdan.robu@gipsa-lab.grenoble-inp.fr, christophe.prieur@gipsa-lab.fr and emmanuel.witrant@univ-grenoble-alpes.fr

[4]Ludovic Dos Santos is with Criteo AI Lab, Paris, France. Email: l.dossantos@criteo.com

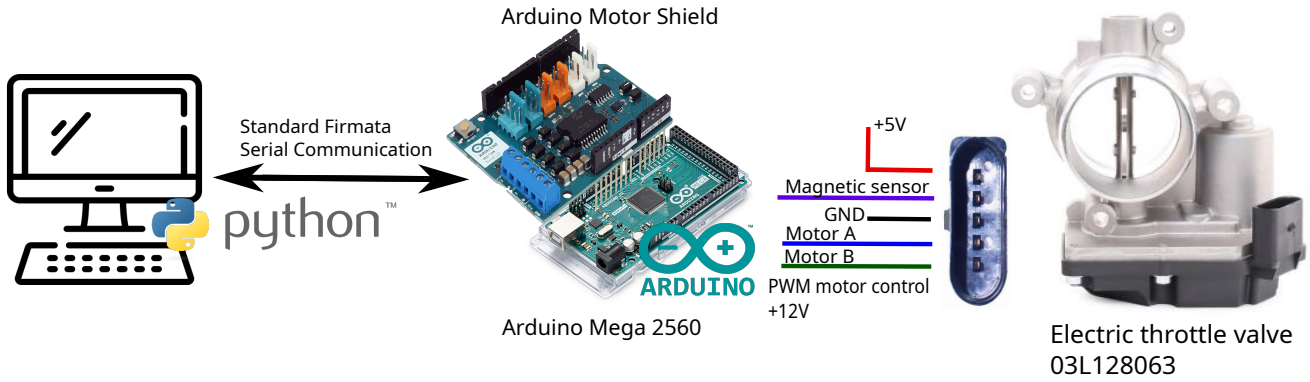

Fig. 1: Experimental test bench: hardware configuration and wiring of the electric throttle valve.

our experimental setup and outlining the physical properties of the throttle valve in Section II. Next, in Section III, we formulate our objective and introduce two traditional control methods that are commonly used for similar systems. Then, we propose a hybrid control algorithm that combines the strengths of the two classical methods in Section IV. Finally, in Section V, we present the results of our experiments, which demonstrate the effectiveness of our hybrid controller in a variety of scenarios.

II. The Throttle Valve System

This section presents a detailed description of our experimental setup for controlling the throttle valve. We then investigate its physical properties and formalize the objective.

A. Experimental setup

The valve used in this study has the commercial reference 03L128063, the same one as in [23]. It possesses one rotational spring, set on the shaft of the valve plate that exerts a torque that counteracts the motor's torque, resulting in control of the plate's angular position. To regulate the opening angle, a piece-wise modulation (PWM) ranging from 0 to 100% is generated and modulates the input voltage from 0 to 255 Volts. The valve is equipped with a magnetic angle sensor. This setup is connected to an Arduino Mega 2560® that controls the input and allows monitoring and analysis of the system performance. The motor control is enabled by the dual full-bridge driver Arduino Motor Shield attached to the Arduino board. In addition, the Arduino is connected to a Python interface that is necessary to build and train the RL agent. Each control input is sent every 50 milli-seconds. The scheme of the experimental test bench is given in Figure 1.

B. A non-linear stochastic system

This valve is particularly known for its non-linear stochastic dynamics with asymmetric hysteresis, which we confirmed in the following experiment. We selected a range of control inputs from 0% to 40% and then decreased it to 0% in increments of 2%, and applied it to the valve. This experiment was repeated 20 times. The resulting data is presented in Figure 2, which clearly demonstrates asymmetric hysteresis as well as the stochastic nature of the dynamics. Notably, the resulting angles vary significantly between each trial.

These results emphasize the necessity of utilizing advanced control strategies to effectively manage the intricacies of the dynamics of the valve.

C. Formalization

Given our experimental set-up and the stochastic nature of the valve's dynamics, the problem is modeled as a discounted Markov Decision Process (MDP) $(\mathscr{X}, \mathscr{U}, c, P, \gamma)$. With $\Delta(\cdot)$ be the simplex on any space $(\cdot)$, let $c : \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ the cost function, $P : \mathscr{X} \times \mathscr{U} \to \Delta(\mathscr{X})$ the transition probabilities, and $\gamma \in [0,1)$ the discount factor. At each time step t, the agent receives an observation $x_t \in \mathscr{X}$ and chooses a control $u_t \in \mathscr{U}$. The system is then updated with $x_{t+1} \sim P(\cdot|x_t, u_t)$.

In this formalism, the objective is to find the optimal controller $\pi^* : \mathscr{X} \to \Delta(\mathscr{U})$ which minimizes the expected discounted cumulative cost $V^\pi(x) = \mathbb{E}_P\left[\sum_{t=0}^{\infty} \gamma^t c(x_t, u_t) | x_0 = x, u_t \sim \pi(\cdot|y_t)\right]$ for each observation $x \in \mathscr{X}$ with as few data samples as possible. V^π is known as the Value function of policy π, and we define the associated Q-value function as $Q^\pi(x,u) = \mathbb{E}_P\left[\sum_{t=0}^{\infty} \gamma^t c(x_t, u_t) | x_0 = x, u_0 = u, u_t \sim \pi(\cdot|y_t)\right]$.

The goal of this work is to develop a throttle valve controller $\pi : \mathscr{X} \to \Delta(\mathscr{U})$ that sets the angle of the valve's plate to a chosen angle from any initial position. The controller must not only perform optimally in nominal operating conditions by minimizing cumulative costs but also demonstrates robustness and stability under various real-world scenarios. Specifically, the controller must be able to reject external disturbances that may affect both the input and output signals, while exhibiting minimal overshoot. Bearing this information in mind, we associate the different components of the MDP with the valve. Let α_t be the angle of the valve's plate at time-step t, and α_{ref_t} be the chosen angle objective at time-step t.

- The control u is the PWM (%) input. To avoid saturation effects, we set it to $\mathscr{U} = [0, 0.6]$.
- The objective is to find a controller $\pi : \mathscr{X} \to \Delta(\mathscr{U})$ that sets the angle of the valve's plate to a chosen angle from

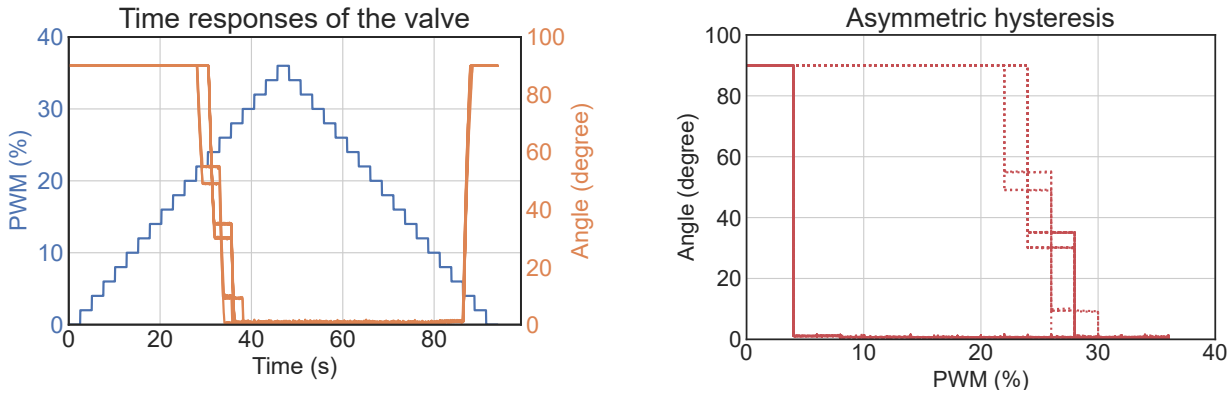

Fig. 2: Time responses of the valve from a range of control inputs.

any initial position. Hence, the cost function c is set to $c(x_t, u_t) = \|\alpha_t - \alpha_{\text{ref}_t}\|_2$.

- The controller must be optimal in all states, so the discount factor γ is set to the high value of 0.99.
- The transition probabilities P are unknown.
- To enhance the convergence of RL algorithms, the system is made episodic, with each episode lasting for 100 time-steps, which represents 5 seconds. In each episode, a single random reference angle is generated, and the primary objective is to set the throttle valve's position to this specific angle.

III. Traditional techniques

In this section, we detail two traditional techniques that will be used to achieve our objective: obtaining the most efficient and robust controller for the throttle valve.

A. Time Discrete PI Controller

The time discrete PI controller is commonly used in control systems design due to its simplicity and effectiveness in a wide range of systems. It can be expressed as:

$$\pi_{\text{PI}}(x_t) = u_{t-1} - r_0\ \alpha_t - r_1\ \alpha_{t-1} + (r_0 + r_1)\ \alpha_{\text{ref}_t}. \quad (1)$$

In order to properly tune the gains for this system, we followed the steps proposed by the previous work[23], detailed below. This work found that the following auto-regressive model

$$\alpha_{t+1} = a\ \alpha_t + b_0\ u_t + b_1\ u_{t-1} \quad (2)$$

correctly represents the dynamics of the valve. The model parameters a and b were tuned to fit the first order model thanks to data describing the response of the valve from a rich signal, generated by a 1022-length Pseudo Random Binary Sequence (PRBS) centered at 16%. Then, r_0 and r_1 were chosen such that the damping ratio ζ is set to 1 and the rising time t_R to 0.8 seconds.

This experiment lead to the values of $a = 0.83$, $b_0 = -0.11$, $b_2 = -0.23$, $r_0 = -2.33$ and $r_1 = 1.96$.

B. Approximate Policy Iteration

In order to have a reliable effective controller for the valve, we also propose to use a traditional Reinforcement Learning agent that uses the Approximate Policy Iteration scheme [24].

Most RL algorithms rely on the Bellman operator, defined as $\mathscr{B}^\pi[Q](x,u) = c(x,u) + \gamma \mathbb{E}_{x' \sim P(\cdot|x,u), u' \sim \pi(\cdot|x')}[Q(x',u')]$. This operator is a γ-contraction, so iteratively applying it to any function Q converges to its unique fixed point Q^π. However, since the transition probabilities P are unknown, the empirical Bellman operator $\hat{\mathscr{B}}$ that uses interactions with the environment to estimate this expectation is used instead.

In the Approximate Policy Iteration framework, both the policy π_{RL} and the Q functions are parametrized with the respective weights $\theta \in \Theta$ and $\omega \in \Omega$ that will respectively be denoted as π^θ_{RL} and Q^ω. Both estimates will work together until the policy converges to a near-optimal one. More exactly, at each epoch k, the agent collects data with the current policy $\pi^{\theta_k}_{\text{RL}}$, evaluates its associated Q-function via Approximate Policy Evaluation (APE) and improves its policy through Approximate Policy Improvement (API). Given a dataset $\mathscr{D} = \{(x_i, u_i, c_i, x_{i+1})_{i=1}^N\}$, $\hat{\mathbb{E}}$ the empirical expectation of the observation-control pair (x,u) induced by the data set $\mathscr{D}$, the scheme is formalized as:

$$Q^{\omega_{k+1}} \leftarrow \arg\min_{\omega \in \Omega} \hat{\mathbb{E}}\left[\left(Q^\omega - \hat{\mathscr{B}}^{\pi^{\theta_k}_{\text{RL}}}[Q^{\omega_k}]\right)^2\right], \quad \text{(APE)}$$

$$\pi^{\theta_{k+1}}_{\text{RL}} \leftarrow \arg\min_{\theta \in \Theta} \hat{\mathbb{E}}_{x \sim \mathscr{D}, u \sim \pi^\theta_{RL}}[Q^{\omega_{k+1}}(x,u)]. \quad \text{(API)}$$

This cycle of evaluation and improvement is commonly solved approximately with gradient descent and continues until the policy converges to an optimal one. Note that this framework does not require knowledge from the environment, apart from the data coming from the interactions with the environment. Therefore, it can be applied as such to any system as long as it possesses the Markov property. However, this lack of initial information makes it difficult to collect meaningful data at the early stages of learning, leading to poor estimates of the Q function and the policy. This problem is known as *bad exploration*.

One of the current best algorithms using this scheme is $TD3$ [25], an improvement over DDPG [9]. It is the one used in this work.

IV. Combining both methods

On the one hand, the PI controller is easy-to-built and requires a small amount of data, but suffers from sub-optimality on non-linear systems because of their linear dependencies. On the other hand, RL agents often reach near-optimality on complex systems but require a huge amount of data to do so. Two of the main reasons for this requirement are the bad initialization of the policy and the search for the control u in the whole control space $\mathscr{U}$ at each time step.

To combine the advantages of both methods, we draw inspiration from the recent framework Reinforcement Learning with Guides [20] and adapt one of the most recent and efficient techniques to our setting: Perturbation Action Guided (PAG) [22]. For clarity purposes, we denote the adaptation of PAG to our setting by PI-RL.

In this framework, to avoid the search on the entire control space $\mathscr{U}$ to find the optimal control, the PI controller is incorporated to guide the training process. Not only does this reduce the search space, but it also leads the RL agent toward good parts of the environment, directly improving the exploration and the quality of the gathered data. In order to do so, the PI-RL policy is centered around the PI controller and learns a perturbation ξ^{ϕ}_{RL}, with $\phi \in \Xi$ to guide it towards better actions. Formally, the new policy is written

$$\pi^{\phi}_{\text{PI-RL}}(\cdot|x) = \pi_{\text{PI}}(x) + \xi^{\phi}_{\text{RL}}(\cdot|x). \tag{3}$$

The perturbation ξ^{ϕ}_{RL} differs from the original RL policy π^{θ}_{RL}. Instead of providing a control input u, it just aims to improve the PI controller. Hence, as opposed to the traditional RL policy π^{θ}_{RL} that is defined in the whole control space $\mathscr{U}$, the perturbation ξ^{ϕ}_{RL} is constrained to a small subset $S(\mathscr{U})$ of $\mathscr{U}$. Hence, instead of searching for the optimal control in the whole control space, the search is constrained to stay within a small neighborhood surrounding the PI controller, where the optimal control is likely to lie. This improves the sample efficiency of the algorithm.

The perturbation is learned similarly to that in traditional RL, that is by minimizing its associated Q^{ω} function. The process becomes the PI-Approximate Policy Iteration, where the APE-API steps are modified to be PI-APE and PI-API, defined as:

$$Q^{\omega_{k+1}} \leftarrow \arg\min_{\omega\in\Omega} \hat{\mathbb{E}}\left[\left(Q^{\omega} - \hat{\mathscr{B}}^{\pi^{\phi_k}_{\text{PI-RL}}}[Q^{\omega_k}]\right)^2\right], \quad \text{(PI-APE)}$$

$$\xi^{\phi_{k+1}}_{\text{RL}} \leftarrow \arg\min_{\phi\in\Xi} \hat{\mathbb{E}}_{x\sim\mathscr{D},u\sim\pi^{\phi}_{\text{PI-RL}}}\left[Q^{\omega_{k+1}}(x,u)\right]. \quad \text{(PI-API)}$$

Similarly to the traditional RL agent, both the Q-function and the perturbation ξ^{ϕ}_{RL} are parametrized with Neural Networks that are learned with gradient descent. The subspace of $\mathscr{U}$ is chosen as $S(\mathscr{U}) = [0,3]$ reducing by two the search space.

The whole algorithm can be found in Pseudo-Code 1.

Algorithm 1 PI-RL

```
Select damping ratio ζ, rising time t_R and Φ
Create PI controller
Initialize Q^{ω_0} and ξ^{φ_0}_RL
for k ∈ (0, ..., K) do
    Gather data with π^{φ_k}_PI-RL from Eq. (3)
    Add data to the replay buffer D
    Sample a batch from D
    Update Q^{ω_{k+1}} with gradient descent on Eq. (PI-APE)
    Update ξ^{k+1}_φ with gradient descent on Eq. (PI-API)
end for
```

V. Comparison of the different controllers

We conducted a series of experiments on various setups. Initially, we performed a nominal comparison between the controllers, where we tested their performance on a simple task: tracking a reference slotted signal with sufficient time to adjust. We then tested the controllers on more challenging tasks, including adapting to quick reference changes and robustness with respect to noise in the control input or in the output observation.

To assess the effectiveness of the different controllers, we use the Mean Square Error (MSE) between the current angle and the objective. It is formally defined with sequences $\alpha = (\alpha_0, \alpha_1, \dots)$ and $\alpha_{\text{ref}} = (\alpha_{\text{ref}_0}, \alpha_{\text{ref}_1}, \dots)$ as:

$$MSE(\alpha, \alpha_{\text{ref}}) = \|\alpha - \alpha_{\text{ref}}\|_2 . \tag{4}$$

This metric seemed appropriate as it encompasses all of the costs during one scenario. The next subsections detail the experiments. All the results can be found in Table I.

In all our experiments, both RL-based algorithms used TD3. The Q-functions and the policies π^{θ}_{RL} and ξ^{θ}_{RL} were parametrized with Neural Networks with 2 hidden layers of size 64 and used the ReLU activation function.

A. Nominal comparison and reaction to quick reference changes

We compared the performance of all agents for tracking a reference signal through a series of experiments. First, we performed a simple comparison on angles close to 45 degrees with ample time to adjust. Next, we assessed their reaction to quick reference changes at extreme values. All results are summarized in Figure 3.

a) PI controller: In the first simple scenario, we observed that while the PI controller was efficient, it struggled to precisely track the reference signal. The controller required time to accumulate errors before adjusting to the reference, and the system's difficulty to control necessitated a sufficiently strong input control, which could lead the valve's plate to move beyond the objective. The PI controller would then need additional time to adjust and could again select too-high control input. This resulted in oscillations around the reference angle, as seen for the two last reference angles in the first scenario's Figure 3. In the subsequent, more

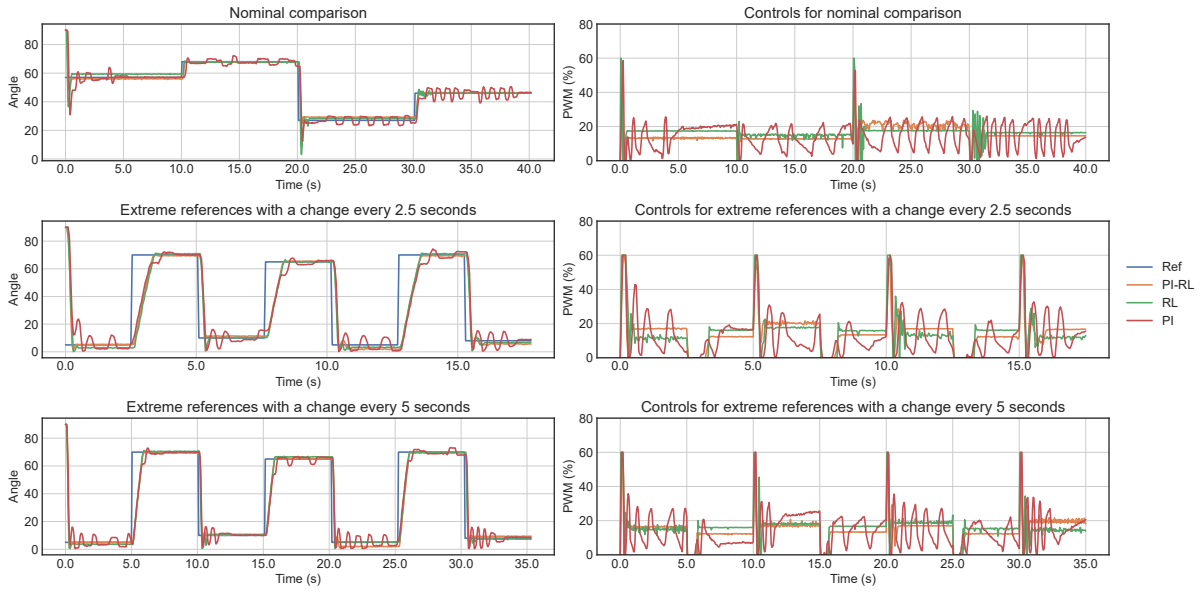

Fig. 3: Comparison of the different agents under different scenarios when all agents have access to the real outputs. The figures on the left display the evolution of the angles over time for the different agents, while the figures on the right indicate the controls provided by the different agents to reach the references.

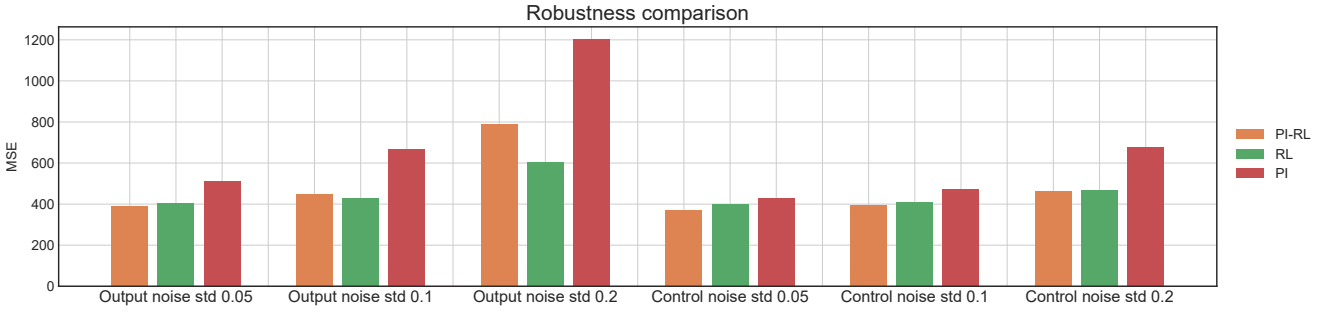

Fig. 4: Comparison of the different agents under noisy outputs or noisy controls with different standard deviations. Each bar represents the Mean-Squared-Error (MSE) between the reference and the angle of the valve.

challenging scenarios, the PI controller had even less time to adjust to its errors, highlighting its difficulty in controlling the valve. Especially, the PI controller struggles with low-angle references where the valve seems to be difficult to control. This reveals the need for more sophisticated controllers and not solely relying on a linear policy.

b) RL-based agents: On the other hand, both RL-based agents demonstrated remarkable success in precisely tracking the reference signal in all scenarios, including the most challenging ones. For example, they were able to track all extreme reference values with a change every 2.5 seconds, which is twice faster as the rate they were trained for. This level of performance was also observed for low-angle references that proved to be challenging for the PI controller. Notably, the controls provided by the RL-based agents were stable in all settings, with no instances of unstable behavior observed. In some cases, a slight tracking bias was observed where the agents reached a point close to but not precisely at the reference. This can be attributed to the RL agents' goal of minimizing the Q function, which is an expectation over cumulative future costs. To address this, a more appropriate RL formulation may be needed, which considers higher-order moments of the cumulative future costs distribution or steady-state errors in a refined way. We leave this for future work. Despite these minor biases, both RL-based agents demonstrated excellent performance in all tasks.

B. Robustness to external noise

We also conducted experiments to assess the robustness of the controllers in the presence of noise, both in the output and in the control. To this end, we tested the controllers under varying levels of noise and analyzed their performance.

To better visualize the effects of noise on the controllers' performance, we plotted in Figure 4 the Mean Squared Error (MSE) rather than the signals and the reference as in the previous section.

TABLE I: MSE of all the different scenarios (lower the better).

SCENARIO	PI	RL	PI-RL
NOMINAL	144	115	**109**
REFERENCE CHANGES 2.5S	385	**372**	**371**
REFERENCE CHANGES 5S	388	**369**	373
NOISE OUTPUT STD 0.05	512	402	**390**
NOISE OUTPUT STD 0.1	666	**427**	450
NOISE OUTPUT STD 0.2	1222	**603**	788
NOISE CONTROL STD 0.05	429	399	**371**
NOISE CONTROL STD 0.1	475	411	**393**
NOISE CONTROL STD 0.2	679	467	**461**

In comparison to the PI controller, both RL-based agents demonstrated superior performance in all scenarios, including in the presence of noise. However, it's essential to note that under a specific condition of output noise with a standard deviation of 0.2, representing 20% of noise, the PI-RL agent exhibited a higher loss than the classical RL one. This result may be attributed to the fact that the PI-RL agent is centered around the PI controller, which is particularly inadequate in this scenario. Consequently, the reduced search space was not sufficient to yield an optimal controller in this case, and a more sophisticated approach may be needed to address the challenges posed by this specific noise condition. Nevertheless, we note that the perturbation ξ_{RL}^{ϕ} was able to improve the PI controller up to a certain point.

In summary, our robustness analysis revealed that both RL-based agents perform well in noisy environments while the PI controller's performance declined more rapidly. This declination eventually affects the PI-RL agent as the search space is too narrow to recover a near-optimal policy. These results suggest that the PI-RL agent offers promising results in low-to-moderate noise environments, where the PI controller can still provide a relevant control signal. However, in more challenging noise conditions where the PI controller's performance is limited, traditional RL may be better suited to provide a more robust and effective controller.

C. Sample efficiency

As previously shown, both the PI-RL and classical RL agents achieved similar performances in controlling the throttle valve. One important advantage of the combined PI and RL approach over the classical RL approach is that it requires much less data to learn. This can be seen in Figure 5, where we plotted the cumulative costs over the training process for both controllers.

As shown in Figure 5, the PI-RL approach achieves a low cumulative cost faster than the RL approach, indicating that it was able to learn the control task more efficiently. This is because PI-RL leverages the prior knowledge of the PI controller to reduce the search space and guide the RL exploration, which helps to reduce the exploration time and improve the sample efficiency. After a sufficient amount of data, that is after 100 episodes, the traditional agent finally catches the performances with the combined agent.

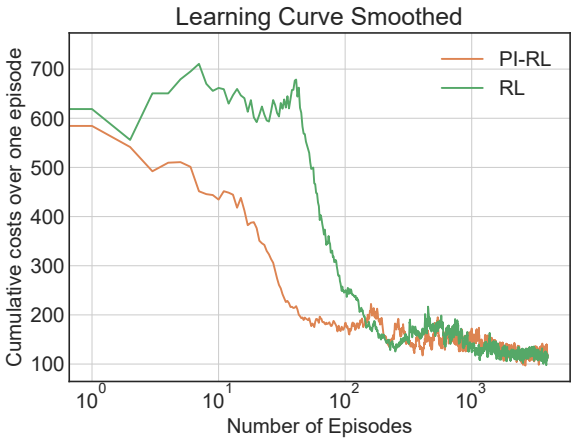

Fig. 5: Learning curves of the different RL-based agents where the x-axis is in log-scale. We trained both agents during 4000 episodes, representing 5 hours with our computer without the need of using a GPU.

The combination of PI and RL, therefore, benefits from the advantages of both worlds. The PI controller provides a good indication of where the optimal control is located, while RL can refine it toward better controls.

VI. CONCLUSION

In this work, we showed that the combination of Reinforcement Learning and Optimal Control (PI) can be a powerful approach to designing near-optimal controllers for throttle valve systems. By adapting the recent area of Reinforcement Learning with Guides to the throttle valve system, we were able to build a near-optimal controller that reduces the data requirement of traditional Reinforcement Learning agents and overcomes the limitations of the PI controller.

This provides evidence that the integration of OC with RL can lead to significant improvements in data efficiency, making it a promising avenue for future research in other complex systems.

REFERENCES

[1] C. Canudas de Wit, I. Kolmanovsky, and J. Sun, "Adaptive pulse control of electronic throttle," in *Proceedings of the 2001 American Control Conference*, vol. 4. IEEE, 2001, pp. 2872–2877.

[2] U. Ozguner, S. Hong, and Y. Pan, "Discrete-time sliding mode control of electronic throttle valve," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 2. IEEE, 2001, pp. 1819–1824.

[3] J. Deur, D. Pavkovic, N. Peric, M. Jansz, and D. Hrovat, "An electronic throttle control strategy including compensation of friction and limp-home effects," *IEEE Transactions on Industry Applications*, vol. 40, no. 3, pp. 821–834, 2004.

[4] D. Pavković, J. Deur, M. Jansz, and N. Perić, "Adaptive control of automotive electronic throttle," *Control Engineering Practice*, vol. 14, no. 2, pp. 121–136, 2006.

[5] X. Jiao, J. Zhang, and T. Shen, "An adaptive servo control strategy for automotive electronic throttle and experimental validation," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6275–6284, 2014.

[6] S. Zhang, J. J. Yang, and G. G. Zhu, "LPV modeling and mixed constrained H_2/H_∞ control of an electronic throttle," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2120–2132, 2014.

[7] M. Barić, I. Petrović, and N. Perić, "Neural network-based sliding mode control of electronic throttle," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 8, pp. 951–961, 2005.

[8] R. Siraskar, "Reinforcement learning for control of valves," *Machine Learning with Applications*, vol. 4, p. 100030, 2021.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *ICLR (Poster)*, 2016.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[11] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[12] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[13] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[14] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[15] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020.

[16] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, 2021.

[17] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[18] S. Gillen, M. Molnar, and K. Byl, "Combining deep reinforcement learning and local control for the acrobot swing-up and balance task," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 4129–4134.

[19] S. Zoboli, V. Andrieu, D. Astolfi, G. Casadei, J. S. Dibangoye, and M. Nadri, "Reinforcement learning policies with local lqr guarantees for nonlinear discrete-time systems," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 2258–2263.

[20] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: a reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[21] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Reincarnating reinforcement learning: Reusing prior computation to accelerate progress," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 955–28 971, 2022.

[22] P. Daoudi, B. Robu, C. Prieur, L. Dos Santos, and M. Barlier, "Enhancing reinforcement learning agents with local guides," in *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, 2023.

[23] E. Witrant, I. D. Landau, and M.-P. Vaillant, "Teaching data-driven control," *arXiv preprint*, 2023.

[24] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 310–335, 2011.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.