Proceedings of the ASME 2024
International Conference on Energy Sustainability
ES 2024
July 15–17, 2024, Anaheim, CA

ES2024-122475

IMPROVING CONTROL OF ENERGY SYSTEMS WITH REINFORCEMENT LEARNING: APPLICATION TO A REVERSIBLE PUMP TURBINE

Carlotta Tubeuf[1,*], Jakob aus der Schmitten[2], René Hofmann[1], Clemens Heitzinger[2], Felix Birkelbach[1]

[1]TU Wien, Institute of Energy Systems and Thermodynamics,
Getreidemarkt 9 / BA, 1060 Wien, Austria
[2]TU Wien, Institute of Information Systems Engineering,
Erzherzog-Johann-Platz 1, 1040 Wien, Austria

ABSTRACT

Pumped hydro storage power systems are crucial to account for grid instabilities by providing flexibility services. To further increase flexibility, the acceleration of switching between operating modes is necessary. This can be achieved through precise and automated process control with reinforcement learning (RL). Besides the benefits of RL, safety concerns inhibit industrial-scale applications for process control with RL.

We present measures to increase the reliability and stability of RL algorithms to enable applications for the control of energy systems. We demonstrate the viability of our approach by applying it to the control of the pump start-up process of a reversible pump turbine. To train the RL algorithm, we use a simulation model that accurately represents the test rig of a pump turbine located at the laboratory of TU Wien. Our results show that RL is suitable for finding optimal control strategies that can compete with traditional approaches. However, finding the optimal policy still requires a lot of computational effort. Future research will focus on optimizing the RL framework and then transferring the results to the real machine unit at the test facility.

Keywords: reinforcement learning, pump turbine, process control, reliable machine learning

1. INTRODUCTION

The substantial increase of volatile renewable energy sources in the electricity mix is challenging the stability of the grid. Pumped hydro storage systems are well suited to account for the volatility and maintain the required grid frequency through their fast reaction times and reactive power support. With the transition from fossil-fueled power plants, which are currently still providing a large part of grid stability services, to a fully renewable energy supply, pumped hydro storage power systems will need to provide even more flexibility in the future [1]. Currently, operating mode changes have fixed timings with significant safety margins. To increase flexibility, faster reaction times for the switches between operating modes become necessary, which can only be achieved with a precise and automated process control [2]. While traditional control strategies such as proportional-integral-derivate (PID) controllers or model-based predictive control (MPC) methods are well-established control technologies, reinforcement learning (RL), a type of machine learning, regularly outperforms traditional controllers because it learns an optimal control strategy through direct interaction with its environment and can hence adjust to changing conditions [3]. Therefore, RL has the potential to surpass human-level performance and reveal optimization opportunities that were previously unknown [4]. But despite the great potential of RL, its application for process control at industrial-scale is still very rare. In our work, we identify the necessary requirements to apply RL to energy systems using a reversible pump turbine as a use case. We train two state-of-the-art RL algorithms to control a model pump turbine to operate as a pump within a simulation model and compare their performance to traditional control methods. Thereby, we build upon our preliminary work published in [5] and expand it to use a sophisticated simulation model, accurately representing the pump-turbine test rig at the laboratories of the Institute of Energy Systems and Thermodynamics (IET) at TU Wien [6]. Further, we extended the use case to span the whole operation as a pump and train and compare different RL algorithms.

2. REINFORCEMENT LEARNING FOR PROCESS CONTROL

2.1 Reinforcement Learning

Reinforcement learning (RL) is one of the three core paradigms of machine learning, together with supervised and unsupervised learning. In supervised learning, the algorithm learns to map input to output data from labeled training data, and unsupervised learning algorithms explore unlabeled data sets to find

*Corresponding author: carlotta.tubeuf@tuwien.ac.at

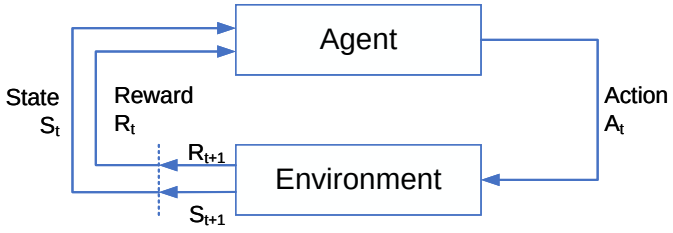


FIGURE 1: Schematic diagram of a Markov decision process (MDP) as used for formulating reinforcement learning (RL) problems, after [7].

inherent structures. RL algorithms don't learn from static data sets but find the optimal strategy for selecting actions in a given situation through direct interaction with the learning environment, guided by a scalar reward signal. Without prior knowledge of which actions to take, RL algorithms explore the state space through trial and error, aiming to maximize the reward. [7]

RL problems are formulated as Markov decision processes (MDPs), as shown in Fig. 1. The characteristic of the Markov property, meaning that the future state depends only on the current state and action, not on the sequence of events that preceded it, is a prerequisite for the modeling and learning process. However, it is important to note that not all real-world problems comply with the Markov property, and violations are handled by introducing additional information that describes the future state. This results in a partially observable MDP, where it is significantly harder to find the optimal policy. [4]

RL problems are stochastic sequential decision-making problems. When presented with a state s_t, a so-called agent decides for an action a_t based on its control strategy or policy π. This action leads to a transition in the environment, yielding a new state s_{t+1}, evaluated with a corresponding reward signal r_{t+1}. The agent aims to learn an optimal policy π^* that maximizes the cumulative reward, called the return G_t, over time. [7]

To estimate the value of a state with the state-value function $v_\pi(s)$, the Bellman Equation

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{1}$$

calculates the expected return when following the policy π, starting from the state s. The action-value function

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{2}$$

estimates the expected return, given a state s and an action a, and then following the policy π.

To ensure that the RL agent explores the whole state-space of the environment, it sometimes has to randomly decide on actions, even though another action would have a higher action value. Finding a true optimal policy could not be guaranteed if the agent only acts deterministically. Conversely, the agent needs to act greedy, i.e., take actions that it knows to yield high rewards, to maximize the episode return [8]. To balance between exploration and exploitation is a key challenge in RL, and tuning of the exploration hyper-parameters of the learning algorithm is a fundamental part of every RL problem.

Generally, RL algorithms can be classified into four main groups [3]:

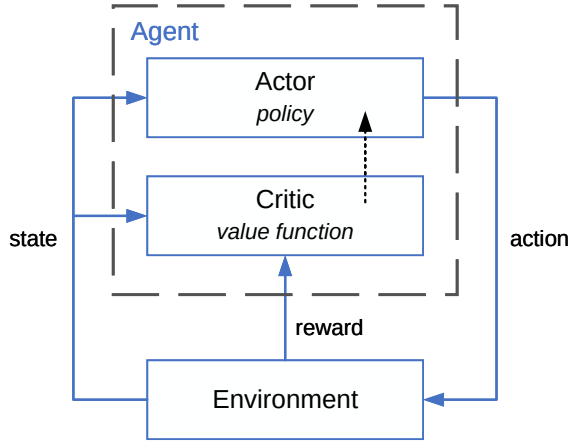


FIGURE 2: Structure of an actor-critic RL class problem formulation, after [3].

- **Model-based** algorithms estimate the transition model $p(s', r|s, a)$, i.e. the probability that the environment will transition to state s' with reward r, given the current state s and action a, and use it directly for control.
- **Value-based** algorithms use a so-called critic to estimate the Q-function, which calculates the value of a state with the state-value function (Eq. 2), and an agent's policy on the Q-value.
- **Policy-gradient** methods directly optimize an agent's policy using gradient ascent to maximize the expected return, using a so-called actor.
- **Actor-critic** algorithms combine value-based and policy-gradient methods, as shown in Fig. 2, where the calculation of the value function is used to evaluate the current policy and the actor then implements the policy and selects the following action.

Besides the distinction between model-based and model-free RL algorithms (value-based, policy-gradient, and actor-critic), a key differentiation is made between on- and off-policy RL methods. In on-policy RL, the same policy that gets updated during learning is used to select actions. Off-policy algorithms use two different policies (one behavior and one target policy) during training. The behavior policy selects the next action to explore the environment's state space, while the target policy is continuously being evaluated and improved, ultimately yielding the optimal policy [7]. Off-policy RL can be particularly beneficial for controlling processes in real machine units, as described in Sect. 2.2.

2.2 Reinforcement Learning for Process Control

Applications of RL for process control have increased significantly over the last few years. Currently, the best-performing traditional control methods rely on complex models, and decisions are based on open-control-loop simulations [9]. With RL, once the algorithm has found the optimal control policy, it can

quickly adapt and control processes with low computational effort. The main advantage of RL is the potential to surpass human level performance and to reveal optimization opportunities that were previously unknown [4].

The publication of several reviews covering recent progress in RL methods for process control underlines the actuality and research interest in the topic [3, 4, 9, 10]. However, all studies conclude that even though the potential of RL is assessed to be high, real-world industrial applications of RL for process control are still rare. The main implementation barrier for using RL for process control certainly is safety concerns [11].

Especially in the domain of energy systems, applications are far from being industry-ready. Most use cases for RL in energy systems deal with building energy management or dispatching problems [12]. Cao et al. [13] list a few studies that propose and test RL for operational control in power systems to optimize stability controllers. Apart from that, successful implementations where an RL agent controls the operation of an industrial-scale energy system are unknown to the authors.

To account for safety requirements, constraints within the environment must be met by the RL policy to ensure a level of control stability [9]. Therefore, RL agents must ensure constraint satisfaction with high probabilities [11].

We propose the following measures to improve and ensure the reliability of RL for process control: First, offline pre-training of the RL agent on a simulation model before transferring it to control processes online ensures safe operation. We suggest the concept of policy transfer, whereby the agent obtains the optimal policy in a simulated environment. Then, the policy is applied to the real process, where it adapts to actual conditions [9]. This way, training gets sped up considerably, as the learning process in the simulated environment is much faster than when operating the real process. Second, using off-policy RL algorithms may increase trustworthiness, as the performance of the RL target policy can be estimated without actually executing it. The environment is being controlled with the reliable behavior policy, which prevents entering unwanted machine conditions. To assure that a newly found policy yields good results without direct execution, the computation of confidence bounds as a performance measure is recommended [14]. Third, we propose to use a digital twin (DT) platform that incorporates the virtual replication of a machine unit and allows for real-time communication between the physical entity and its simulation model. The connection between the virtual and the physical entity of the DT makes it possible to develop an optimal control service using RL [5]. When the agent finished training on the virtual entity of the DT, it gets deployed to control the physical unit. As a result, unforeseen physics previously overlooked in the simulation model could then be eliminated as the RL agent adapts to the actual conditions and physics in the machine unit.

This work presents the first step for applying RL for process control – training the agent within a simulated environment. Future research will deal with assessing the other concepts to improve the reliability of RL and the transfer to the real machine unit.

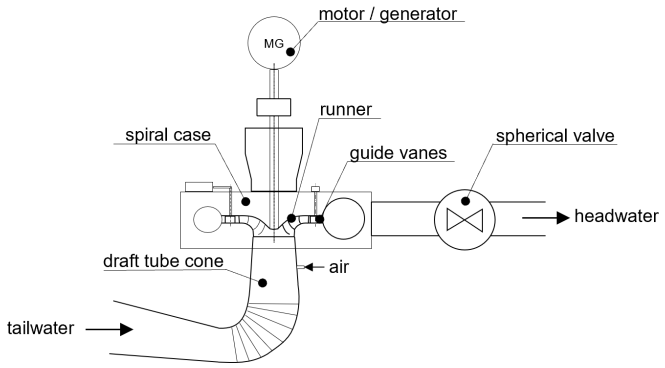


FIGURE 3: Cross-sectional depiction of a reversible pump turbine with relevant components for pump start-up.

3. USE CASE

To demonstrate our concept for optimal control with reinforcement learning (RL) for energy systems, we apply an RL algorithm to control processes for the operation of a reversible pump turbine. Controlling the operation as a pump within a simulated environment thereby serves as a proof-of-concept.

The documentation of the simulation model characteristics can be found in [15]. The model represents the pump turbine test rig at the laboratories of the Institute of Energy Systems and Thermodynamics (IET) at TU Wien [6].

3.1 Pump Operation

To operate a pump turbine in pump mode, the water within the turbine runner in stand-still must first be displaced by air to minimize the start-up torque. Therefore, the headwater-sided spherical valve and guide vanes must be closed so that no water enters the runner from the headwater side. Then, pressurized air gets blown into the runner, lowering the water surface level to the draft tube cone. Only then does the motor-generator unit get started up until the rotational speed of the runner reaches the desired frequency to synchronize with the electricity grid. Subsequently, the guide vanes and spherical valve are opened again, and the water level rises again as the air in the runner dissipates. The machine unit thus operates as a pump [16]. Figure 3 shows all relevant components for the start-up as a pump.

For the proof-of-concept for using RL for precise process control, we consider the pump operation to consist of two consecutive control problems. First, one RL agent controls the blow-out and start-up sequence, and then a different RL agent is trained to operate the pump turbine to maintain a constant flow rate. This allows for using two independent RL algorithms with different action and state spaces. As the two processes have distinct control objectives, training different types of RL algorithms tailored to the specific requirements is beneficial. The flow chart in Fig. 4 illustrates the control problem formulated for the RL use case.

The control problem is designed to start with the machine unit at stand-still, with the headwater-sided spherical valve and the guide vanes already closed, as this step is necessary and without any potential for improvement through applying RL. For Part A, the start-up control process, the RL agent is set to control the blow-out valve (i.e., to decide whether to blow air into the turbine

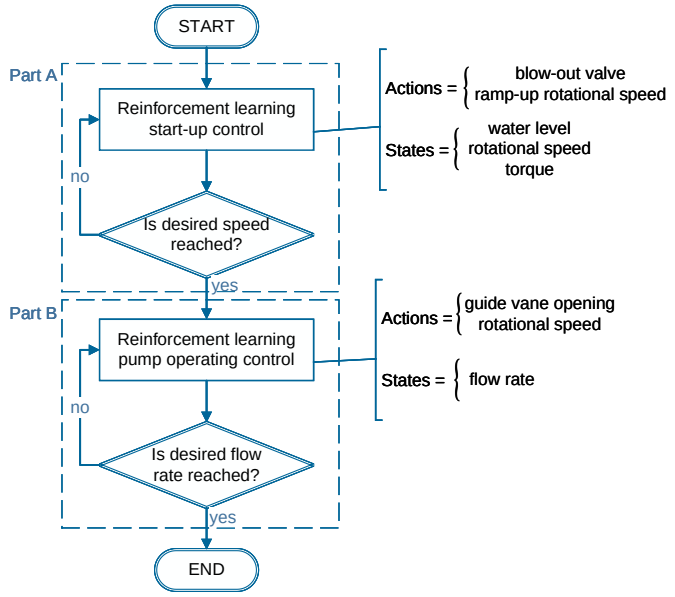


FIGURE 4: Flowchart concept for controlling the operation as a pump with RL.

runner or not) and the signal that starts ramping up the rotational speed. Hereby, the gradient of the acceleration to the desired frequency is fixed, and the agent can only decide at each time step whether to start/continue ramping-up the runner's rotation or stay at the previous rotational speed. Part A of the pump operation aims to minimize the start-up torque while reaching the desired speed as fast as possible.

As soon as Part A is completed, the spherical valve is automatically opened again, leading to the refilling of the turbine runner with water. This process is not part of the process controlled with RL, as it has no optimization potential. The control of Part B starts with the spherical valve completely open and the runner rotating in water. The RL agent aims to find the optimal combination of the runner guide vanes' opening angle and the turbine runner's rotational speed. It can access both actions simultaneously to reach the desired pumping flow rate as fast as possible. When the desired flow rate was held for a certain amount of time, the pump operation process is considered done, and the control use case is considered completed.

3.2 Reinforcement Learning Implementation

For the control of the pump operation, we use RL algorithms from the *Stable-Baselines3* framework for Python [17] using PyTorch [18]. The connection between the agent, implemented in Python [19], and the environment, modeled in Matlab/Simulink using the Simscape language [20], was realized via *Simulink Gym* [21], a gym interface wrapper for Simulink models. The wrapper is based on adding TCP/IP communication between a Simulink model running in a background instance of MATLAB/ Simulink and a Python wrapper class implementing the Gym interface.

Part A. We used a deep Q-network (DQN) RL algorithm to control Part A, the pump start-up process. DQN is a model-free, off-policy RL method that uses deep neural networks (DNNs) to approximate the optimal action-value function (Eq. 2) [22].

TABLE 1: Hyperparametersettings for the DQN Agent

Parameter	Value
Learning rate	0.0001
Replay buffer size	122880
Steps before learning starts	0
Minibatch size	128
Soft update coefficient	0.8
Discount factor	1
Training frequency steps	4
Gradient steps	1
Target update interval	30000
Exploration fraction	0.4
Initial exploration rate	0.5
Final exploration rate	0.0001
Maximum gradient clipping value	10

DQN is a state-of-the-art RL off-policy algorithm recommended for tasks with discrete actions, as it is well-established and very sample-efficient. The DQN agent has a discrete action space with two binary actions

$$A_A = \{A_{\text{blow\_out\_valve}}, A_{\text{speed\_switch}}\} \tag{3}$$

and a continuous state space

$$S_A = \{S_{\text{waterlevel}}, S_{\text{rot\_speed}}, S_{\text{torque}}\}. \tag{4}$$

The reward at each training step is defined as

$$R_A = w_{A1}R_{\text{waterlevel}} + w_{A2}R_{\text{air}} + w_{A3}R_{\text{switching}} + w_{A4}R_{\text{torque}} + w_{A5}R_{\text{time}}. \tag{5}$$

The reward function has to be very carefully defined since it controls the trade-off between fast operation and load on the runner, which affects energy efficiency and wear. In the reward function that we decided on, based on our expertise and many test trainings, the agent gets a penalty $R_{\text{waterlevel}}$ if the water level in the draft tube is below a critical threshold, as air could leak into the tailwater vessel. R_{air} accounts for the penalty the agent receives when blowing air into the runner to minimize the air used during the blow-out process. A penalty for switching the blow-out valve through $R_{\text{switching}}$ encourages more stable operation. The agent learns to minimize the start-up torque through the penalty R_{torque}. Finally, a penalty for each time step R_{time} makes the agent reach the desired rotational speed as fast as possible. A training episode is finished as soon as the desired rotational speed of the turbine runner is reached. The hyperparameters for the training of the DQN agent are listed in Table 1.

Part B. We trained a proximal policy optimization (PPO) algorithm to find the optimal pump operating control strategy. The model-free, on-policy, actor-critic RL method uses multiple epochs of stochastic gradient ascent to perform policy updates [23]. For continuous actions, PPO is currently the prevalent algorithm to use.

The PPO agent has a continuous action space

$$A_B = \{A_{\text{rot\_speed}}, A_{\text{guide\_vane\_opening}}\} \tag{6}$$

TABLE 2: Hyperparametersettings for the PPO Agent

Parameter	Value
Learning rate	0.00001
Steps to run per update	40960
Minibatch size	128
Number of epochs	10
Discount factor	0.99
Clipping parameter	0.2
Clipping parameter for value function	none
Entropy coefficient	0.3
Value function coefficient	0.5
Maximum value for gradient clipping	0.5

and a continuous state space

$$S_B = \{S_{\text{flowrate}}\}. \tag{7}$$

The reward is calculated at each training step as

$$R_B = w_B R_{\text{time}}, \tag{8}$$

with

$$R_{\text{time}} = \begin{cases} 0, & \text{if } S_{\text{flowrate}} = \dot{Q}_{\text{target}} \pm \dot{Q}_{\text{range}} \\ -1, & \text{otherwise.} \end{cases} \tag{9}$$

The agent thus receives a penalty for each time step when the flow rate is not the target flow rate $\dot{Q}_{\text{target}}$. As soon as the agent achieves to hold the target flow rate for ten consecutive seconds, the training episode is finished. The hyperparameters for the training of the PPO agent are listed in Table 2.

4. RESULTS AND DISCUSSION

The DQN and the PPO agents were trained to control the pump start-up and the pump operation processes, respectively.

4.1 Start-Up Control

The DQN agent was trained for 2217 episodes, as can be seen in Fig. 5. Figure 5 shows the cumulative reward received per episode over the episode number. After 2217 episodes, the training results did not enhance anymore, and the training of the DQN agent was terminated.

The trained agent's control strategy is shown in Fig. 6a. The top graph shows the blow-out action $A_{\text{blow\_out\_valve}}$ together with the state describing the water level $S_{\text{waterlevel}}$ in the turbine runner over the simulation time. The bottom graph shows the action for starting to ramp up the rotational speed $A_{\text{speed\_switch}}$ together with the runner speed state $S_{\text{rot\_speed}}$ over the simulation time. We compare the agent's strategy with how the start-up process would be executed manually (i.e., how it is usually done at the lab-scale pump turbine on which the simulation model that the RL agent trains with is based on). Fig. 6b shows the manual control strategy. The control policy states that air is continuously blown into the turbine runner until the water level in the draft tube is below a certain threshold. As soon as the water level reaches the level where the pump turbine is considered blown-out, the frequency of the runner gets ramped up until the desired

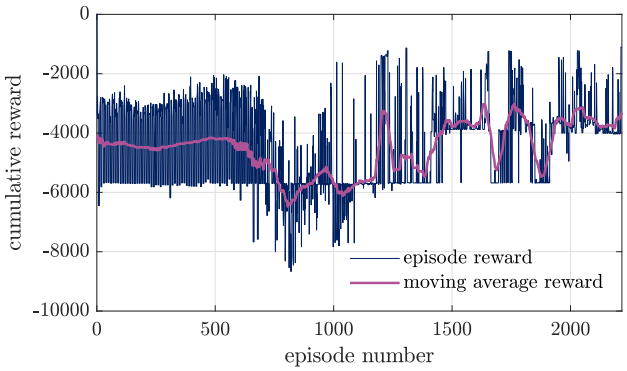
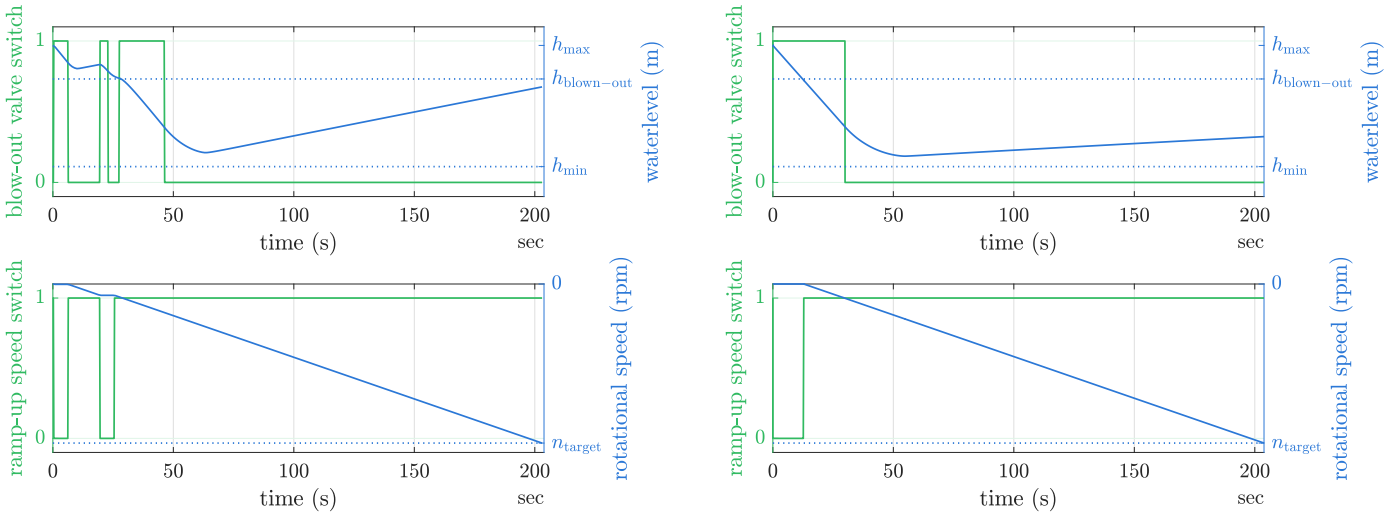


FIGURE 5: Training curve for Part A of the control problem for the DQN agent. The purple line represents the average return, averaged over a moving window of 50 episodes.

frequency with which the machine unit gets synchronized to the grid is reached. Note that as the sense of rotation for pumping in pump turbines is negative, the gradient for the acceleration of the turbine runner is negative for pumping. When comparing the graphs in Fig. 6, it becomes apparent that the DQN agent takes some illogical actions at the beginning of the episode by switching many times between the two binary actions. This indicates that the agent is not fully trained yet.

Table 3 lists criteria relevant for a further assessment of the RL agent's performance. In total, the return achieved with the RL agent's policy is higher than when operated with the manual control scheme. Compared to the episode duration of the manual control, the DQN agent reaches the termination criterion of rotating the turbine runner at the desired speed a bit sooner, which is visible in the episode being terminated earlier. Furthermore, the DQN agent uses less air during the episode. However, the manual blow-out policy reduces the start-up torque by over one percentage point further than the RL agent, with 100% torque rate being the rate that would appear when the runner is not blown-out at all. This indicates that the chosen reward function (Eq. 5) accounts well for minimizing the amount of air used for blowing out the runner and for shortening the time to start up as a pump. Still, it may not account well enough for reaching the goal of minimizing the start-up torque.

A limitation of our study is that the training of the DQN agent to control the start-up sequence has not fully converged yet. Nevertheless, the agent successfully learned to maximize the reward signal and find a near-optimal solution. Finding the optimal hyperparameter setting for training RL algorithms is a key challenge in RL research that we are still working on. Further training with the same hyperparameters did not lead the agent to converge to an optimum. Therefore, we are currently doing extensive hyperparameter studies that take up a large amount of computational resources. Future research will also deal with assessing the reward function and whether it may be necessary to adapt it to achieve better and more comparable rewards per episode. We are confident that once the reward function is evaluated and the training process extended, the RL agent will find the optimal policy for controlling the start-up sequence of the pumping operation.



(a) DQN agent

(b) Manual control

FIGURE 6: Pump start-up control policies

TABLE 3: Comparison of simulation results for the trained DQN agent and a manual control

	DQN	manual
total torque rate (% of max. torque)	2.61%	1.52%
amount of air used (% of max. air mass)	15.59%	25.05%
episode duration (% of max. duration)	99.6%	100%
total return	-2197.7	-2241.2

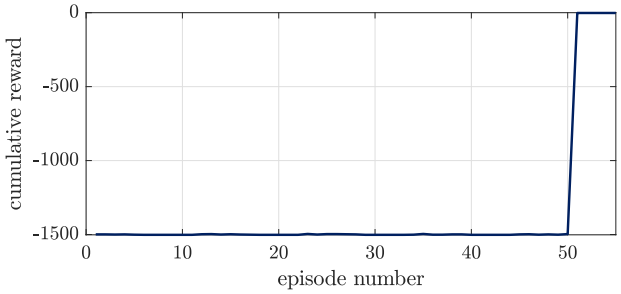


FIGURE 7: Training curve for Part B of the control problem for the PPO agent.

4.2 Flow rate Control

In contrast to Part A of the control problem, the RL algorithm trained for Part B finds the optimal policy very quickly. Figure 7 shows the training progress of the PPO agent used to control the flow rate during the pumping operation of the reversible pump turbine. After 50 episodes of low rewards, the agent's training curve converges to an optimal return (i.e., the cumulative reward) of -2.

As shown in Fig. 8, the agent's control strategy is to stay at a constant rotational speed and open the guide vanes until the state describing the flow rate S_{flowrate} is within the limits set for the target flow rate $\dot{Q}_{\text{target}}$.

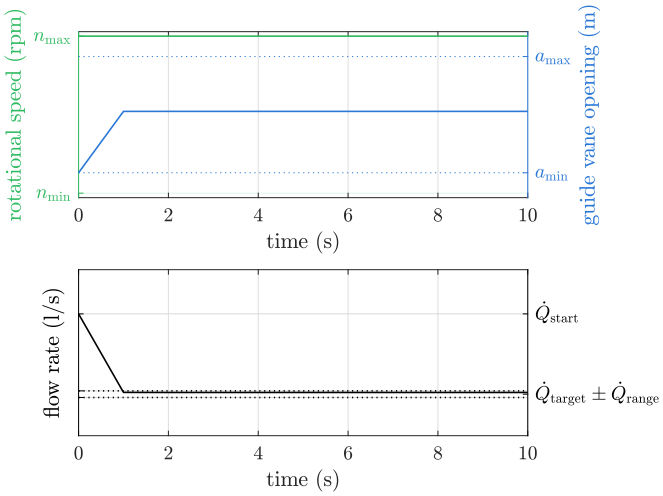


FIGURE 8: Flow rate control policy.

Though the PPO agent sets two continuous actions, increasing the complexity of an RL problem drastically compared to discrete actions, it finds the optimal policy fast and reliable. The optimality of the policy found can be confirmed without comparing it with a manual control strategy. Therefore, we decided not to implement a simple controller but instead focus on adapting the problem so that the RL agent may find more interesting results. Further research will, therefore, focus on the reproducibility of this result and investigate if the PPO agent manages to find the optimal policy as quickly as possible when input parameters are changed.

5. CONCLUSION

We showed our results for training two state-of-the-art RL algorithms, DQN and PPO, to control the operation as a pump of a reversible pump turbine within a simulation model. We divided the use case into two parts: the control of the pump start-up, including the blow-out of the turbine runner, and the control of the flow rate during the pumping operation. Both sequences are relevant regarding optimizing the control process to increase flexibility in pumped hydropower systems. Results indicate that RL is well suited to find control strategies that outperform traditional approaches. While training of the PPO agent, which learned to control the flow rate during operation as a pump, converged very quickly, we did not yet find the suitable hyperparameter settings so that training of the DQN agent converges to the optimum. Finding the optimal parameters requires a very high computational effort because the training has to be started again with each new parameter setting. Depending on the computation resource and the minimum number of training episodes until training is considered to not improve anymore and is therefore terminated, one training run may take up to several days of calculating. To alleviate this issue, hyperparameter analysis and tuning, as well as applying more high-performant algorithms such as TRPO (Trust Region Policy Optimization) or TD3 (Twin Delayed DDPG), will be research objects.

Admittedly, for the current problem formulation, a conventional control method would still be favorable to applying RL algorithms. However, we argue that the true optimization potential will only be apparent once the agent interacts with the actual machine unit, where currently unknown phenomena could occur. Once the optimal RL policies are found, we expect the agent to quickly adapt to changes in the environment. Further, as an RL controller can adjust to changes in drift and other influences, observation may be facilitated and the need for maintenance is reduced. Consequently, our future research will now focus on assessing the reliability of the RL algorithms' policies and then transferring the trained learning algorithms to operate the lab-scale pump turbine at the laboratories of TU Wien.

ACKNOWLEDGMENTS

This work was supported by the project "RELY – Reliable Reinforcement Learning for Sustainable Energy Systems", funded by the Austrian Climate and Energy Fund [FFG, No. FO999899921].

REFERENCES

[1] Kougias, Ioannis, Aggidis, George, Avellan, François, Deniz, Sabri, Lundin, Urban, Moro, Alberto, Muntean, Sebastian, Novara, Daniele, Pérez-Díaz, Juan Ignacio, Quaranta, Emanuele, Schild, Philippe and Theodossiou, Nicolaos. "Analysis of emerging technologies in the hydropower sector." *Renewable and Sustainable Energy Reviews* Vol. 113 (2019): p. 109257. DOI 10.1016/j.rser.2019.109257. URL https://www.sciencedirect.com/science/article/pii/S1364032119304575.

[2] Hirth, Lion. "The benefits of flexibility: The value of wind energy with hydropower." *Applied Energy* Vol. 181 (2016): pp. 210–223. DOI 10.1016/j.apenergy.2016.07.039. Accessed 2024-01-15, URL https://www.sciencedirect.com/science/article/pii/S0306261916309801.

[3] Shin, Joohyun, Badgwell, Thomas A., Liu, Kuang-Hung and Lee, Jay H. "Reinforcement Learning – Overview of recent progress and implications for process control." *Computers & Chemical Engineering* Vol. 127 (2019): pp. 282–294. DOI 10.1016/j.compchemeng.2019.05.029. Accessed 2022-09-19, URL https://www.sciencedirect.com/science/article/pii/S0098135419300754.

[4] Nian, Rui, Liu, Jinfeng and Huang, Biao. "A review On reinforcement learning: Introduction and applications in industrial process control." *Computers & Chemical Engineering* Vol. 139 (2020): p. 106886. DOI 10.1016/j.compchemeng.2020.106886. URL https://www.sciencedirect.com/science/article/pii/S0098135420300557.

[5] Tubeuf, Carlotta, Birkelbach, Felix, Maly, Anton and Hofmann, René. "Increasing the Flexibility of Hydropower with Reinforcement Learning on a Digital Twin Platform." *Energies* Vol. 16 No. 4 (2023): p. 1796. DOI 10.3390/en16041796. Accessed 2024-01-14, URL https://www.mdpi.com/1996-1073/16/4/1796. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[6] Maly, Anton, Käfer, Klaus and Bauer, Christian. "Flow phenomena in a model pump turbine at synchronous condenser mode - description of test rig." *WASSERWIRTSCHAFT* Vol. 109 No. S1 (2019): pp. 19–24. DOI 10.1007/s35147-019-0213-5. Accessed 2024-01-15, URL https://repositum.tuwien.at/handle/20.500.12708/143788. Accepted: 2023-01-30T15:34:32Z.

[7] Sutton, Richard S. and Barto, Andrew. *Reinforcement learning: An introduction*, second edition ed. Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts; London, England (2018).

[8] Langford, John. "Efficient Exploration in Reinforcement Learning." Sammut, Claude and Webb, Geoffrey I. (eds.). *Encyclopedia of Machine Learning and Data Mining*. Springer US, Boston, MA (2017): pp. 389–392. DOI 10.1007/978-1-4899-7687-1_244. Accessed 2024-01-15, URL https://doi.org/10.1007/978-1-4899-7687-1_244.

[9] Faria, Ruan de Rezende, Capron, Bruno Didier Olivier, Secchi, Argimiro Resende and de Souza, Maurício B. "Where Reinforcement Learning Meets Process Control: Review and Guidelines." *Processes* Vol. 10 No. 11 (2022): p. 2311.

DOI 10.3390/pr10112311. Accessed 2024-01-14, URL https://www.mdpi.com/2227-9717/10/11/2311. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[10] Buşoniu, Lucian, de Bruin, Tim, Tolić, Domagoj, Kober, Jens and Palunko, Ivana. "Reinforcement learning for control: Performance, stability, and deep approximators." *Annual Reviews in Control* Vol. 46 (2018): pp. 8–28. DOI 10.1016/j.arcontrol.2018.09.005. Accessed 2022-09-19, URL https://www.sciencedirect.com/science/article/pii/S1367578818301184.

[11] Pan, Elton, Petsagkourakis, Panagiotis, Mowbray, Max, Zhang, Dongda and Rio-Chanona, Ehecatl Antonio del. "Constrained model-free reinforcement learning for process optimization." *Computers & Chemical Engineering* Vol. 154 (2021): p. 107462. DOI 10.1016/j.compchemeng.2021.107462. Accessed 2024-01-14, URL https://www.sciencedirect.com/science/article/pii/S0098135421002404.

[12] Perera, A. T. D. and Kamalaruban, Parameswaran. "Applications of reinforcement learning in energy systems." *Renewable and Sustainable Energy Reviews* Vol. 137 (2021): p. 110618. DOI 10.1016/j.rser.2020.110618. Accessed 2024-01-15, URL https://www.sciencedirect.com/science/article/pii/S1364032120309023.

[13] Cao, Di, Hu, Weihao, Zhao, Junbo, Zhang, Guozhou, Zhang, Bin, Liu, Zhou, Chen, Zhe and Blaabjerg, Frede. "Reinforcement Learning and Its Applications in Modern Power and Energy Systems: A Review." *Journal of Modern Power Systems and Clean Energy* Vol. 8 No. 6 (2020): pp. 1029–1042. DOI 10.35833/MPCE.2020.000552. Accessed 2024-01-15, URL https://ieeexplore.ieee.org/document/9275593/.

[14] Thomas, Philip, Theocharous, Georgios and Ghavamzadeh, Mohammad. "High-Confidence Off-Policy Evaluation." *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 29 No. 1 (2015). DOI 10.1609/aaai.v29i1.9541. Accessed 2024-01-15, URL https://ojs.aaai.org/index.php/AAAI/article/view/9541. Number: 1.

[15] Krause, Maximilian Wilhelm. "Modellierung einer Pumpturbine für flexiblen Betrieb." Thesis, Technische Universität Wien. 2023. DOI 10.34726/hss.2023.96365. Accessed 2024-01-15, URL https://repositum.tuwien.at/handle/20.500.12708/175671.

[16] Giesecke, Jürgen and Heimerl, Stephan. "Funktionsweise von hydraulischen Maschinen." Giesecke, Jürgen and Heimerl, Stephan (eds.). *Wasserkraftanlagen: Planung, Bau und Betrieb*. Springer, Berlin, Heidelberg (2014): pp. 531–590. DOI 10.1007/978-3-642-53871-1_14. Accessed 2024-01-16, URL https://doi.org/10.1007/978-3-642-53871-1_14.

[17] Raffin, Antonin, Hill, Ashley, Gleave, Adam, Kanervisto, Anssi, Ernestus, Maximilian and Dormann, Noah. "Stable-Baselines3: Reliable Reinforcement Learning Implementations." *Journal of Machine Learning Research* Vol. 22 No. 268 (2021): pp. 1–8. URL http://jmlr.org/papers/v22/20-1364.html.

[18] Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Kopf, Andreas, Yang, Edward, DeVito, Zachary, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie and Chintala, Soumith. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. (2019): pp. 8024–8035. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[19] Python Software Foundation. "Python Language Reference." (2023). URL https://www.python.org/.

[20] The MathWorks Inc. "MATLAB version: 9.13 (R2022b)." (2023). URL https://www.mathworks.com.

[21] Brust, Johannes. "Simulink Gym." URL https://github.com/johbrust/simulink_gym.

[22] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane and Hassabis, Demis. "Human-level control through deep reinforcement learning." *Nature* Vol. 518 No. 7540 (2015): pp. 529–533. DOI 10.1038/nature14236. Accessed 2024-01-16, URL https://www.nature.com/articles/nature14236. Number: 7540 Publisher: Nature Publishing Group.

[23] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec and Klimov, Oleg. "Proximal Policy Optimization Algorithms." (2017). Accessed 2024-01-16, URL http://arxiv.org/abs/1707.06347. ArXiv:1707.06347 [cs].