Reinforcement Learning-Based Tap-Operations of a Regulated Distribution Transformer for Autonomous Voltage Control

Leschek Kopczynski [1], Oliver Wolf [1], Michael Schallenburger [1], Roland Zeise [1], Holger Hirsch [2]

[1] *Faculty of Electrical Engineering &Information Technology,*
University of Applied Sciences Düsseldorf, Düsseldorf, Germany.
leschek.kopczynski@hs-duesseldorf.de

[2] *Department of Electrical Engineering,* University of Duisburg-Essen, Duisburg, Germany.
holger.hirsch@ets.uni-due.de

***Abstract* – Our research reported in this paper addresses the voltage control problem in low-voltage distribution networks. The objective is to find an optimal policy to determine the tap position for a regulated distribution transformer (RDT) using a deep reinforcement learning (DRL) approach. A deep deterministic policy gradient (DDPG) agent with a continuous action space is used. During the offline training process, the DDPG algorithm learns the best control actions of the RDT for different system states. The main focus of the described approach is to stay in between given voltage boundaries while reducing the number of tap- operations. This is important in a practical manner, since tap-operations can shorten the interval until the next revision and therefore increase maintenance costs. To address this, an additional slope over a window of past observations is calculated and considered within the state space and the reward function. In a detailed simulation environment that explicitly takes into account voltage dependencies, it is shown that this approach results in a reduced number of tap-operations.**

Keywords - power system modeling; reinforcement learning; transformer tap-operation; voltage control

I. INTRODUCTION

The paradigm shift from a centralized electrical power system to a decentralized one takes place all over the world. Especially the massive integration of distributed power generation (DG) and electric vehicles (EV) results in a growing complexity of the low-voltage grid and requires new strategies to solve control problems in order to increase the overall flexibility of the system.

There are several control tasks like e.g., conservation voltage reduction (CVR), which describes a reduction of customer voltage in the specified service voltage range to save energy. Or volt/var optimization (VVO), which focuses on optimizing the power-factor of controllable reactive sources. These control tasks are based on strategies and approaches to manage existing controllable assets such as DG and regulated distribution transformers (RDT). Classical optimization-based approaches like optimal power flow (OPF) are well-known but can suffer from scalability and robustness [1].

More recent model-free strategies are based on machine learning and artificial intelligence approaches such as reinforcement learning (RL). These data-driven methods have sparked interest of power and energy community. In particular, the breakthroughs in [2] and [3] led to an increased volume of scientific work in the field of power grid control tasks. In [4] e.g., deep RL (DRL) is used to handle voltage violations based on growing dynamics and uncertainties in medium-voltage distribution networks. To address this voltage control problem, a DRL-based agent learns control actions for DG and transformer tap position.

This paper addresses the voltage control problem in low-voltage distribution networks by finding an optimal policy to determine the tap position for a RDT. A deep deterministic policy gradient algorithm (DDPG) with a continuous action is used to perform (optimal) tap-operations in the course of the day. Compared to other studies like e.g., in [5], the main focus of the described approach is to stay in between given voltage boundaries while reducing the number of tap-operations. To address this, a linear function and a window of past observations is used to calculate a slope at specific nodes. The performance is compared to an agent without taking this additional slope within the state space and reward function into account. A proven RL framework for electric power system decision and control tasks is presented. Additional scenarios are defined within the framework to take (endogenous and exogenous) uncertainties into account.

II. MARKOV DECISION PROCESS AND (DEEP) REINFORCEMENT LEARNING

A. Markov Decision Process and Reinforcement Learning

For solving a control task with RL the problem has to be formulated as a Markov Decision Process (MDP). In general, the controller, which is called agent in the field of RL [6], interacts with an environment $\mathcal{E}$ over a discrete number of

time steps $t \in \mathcal{T}$ and uses different signals to gain knowledge about the control task to provide a control strategy. The state signal $s_t \in \mathcal{S}$, where $\mathcal{S}$ is a set of finite states and called the state space, action $a_t \in \mathcal{A}$ with the action space $\mathcal{A}$ and $a_t = \pi(a_t|s_t)$ according to agents policy $\pi(a_t|s_t)$ for taking action a_t in state s_t as well as a scalar reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$, where $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ denote the reward function [5]. We refer to tuple (s_t, a_t, s_{t+1}) as transition, where $s_{t+1} \in \mathcal{S}$ is the state following s_t after taking action a_t and receiving reward r_t with a conditional probability:

$$\begin{aligned} \mathcal{P}_t(s_{t+1}|s_t, a_t) &= P\{s_{t+1}|s_t, s_{t-1}, \ldots, s_0, a_t\} \\ &= P\{s_{t+1}|s_t, a_t\} \end{aligned} \quad (1)$$

as a state-transition model.

Finally, the MDP can be described as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\gamma \in [0, 1]$ is a discount factor [6].

The agent learns a policy $\pi : \mathcal{S} \to P(\mathcal{A})$ and uses π to explore the environment $\mathcal{E}$. The overall goal of the agent is to find an optimal policy π^* by maximizing the expected total return:

$$G_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i) \quad (2)$$

from time step t onwards, by interacting with $\mathcal{E}$ [6].

There are several methods and algorithms which are proven to converge against π^* using the action-value function $Q_i \to Q^*$ as $i \to \infty$, like the *SARSAmax* algorithm, also known as Q-learning [7]. Q-learning is a tabular solution method based on *temporal-difference* learning. An action-value function (Q function) is used with a *deterministic* policy $\pi : \mathcal{S} \to \mathcal{A}$, which describes a direct mapping from $\mathcal{S}$ to $\mathcal{A}$ with $a = \pi(s), s \in \mathcal{S}, a \in \mathcal{A}$. To evaluate the quality of the chosen action a_t in state s_t at time step t the action-value function $Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_t \sim \mathcal{E}, a_t \sim \pi}(G_t|s_t, a_t)$ is used, which is the expected total return G_t when taking action a_t in state s_t and following the current policy π afterwards.

Once the optimal action-values (Q values) $Q^*(s, a)$ are obtained, the optimal policy π^* can be directly constructed by taking action:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^*(s_t, a) \quad (3)$$

that maximizes $Q^*(s_t, a_t)$ at every time step t. In this working mode an agent is called *greedy* with respect to G_t.

Since Q-learning is a *temporal-difference* based approach, Bellman's optimality principle leads to a recursive expression:

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q^*\left(s_{t+1}, \pi^*(s_{t+1})\right)\right]. \quad (4)$$

Finding the optimal Q function $Q^*(s_t, a_t)$, and correspondingly, the optimal policy π^*, will solve the MDP [5].

B. Deep Reinforcement Learning

A possible approach to approximate the Q function is the use of neural networks (NN) for nonlinear function approximation with weights θ^Q, so that $Q(s_t, a_t|\theta^Q) \approx Q(s_t, a_t)$ [2]. To update parameter θ^Q, the mean-squared Bellman error (MSBE):

$$\mathcal{L}(\theta^Q) = \mathbb{E}\left[\left(y_t - Q(s_t, a_t|\theta^Q)\right)^2\right] \quad (5)$$

with $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1})|\theta^Q)$ is minimized.

In the past, the use of NN for nonlinear function approximators in RL has often been avoided, since there are no convergence guarantees [3]. This gap was addressed by the authors in [2] with a novel agent (namely deep Q-Network (DQN)) by applying significant improvements. With these improvements it is possible to develop a single, model-free RL algorithm that is able to solve tasks with a varied range of complexity considering a continuous state space $\mathcal{S}$ and a discrete action space $\mathcal{A}$.

However, many electric power system decision and control tasks need a continuous action space to address voltage control strategies. Since, DQN is based on the idea behind the Q-learning algorithm in [7], a main step is to find an action that maximizes the Q function. This operation, formulated in (3), would require a global optimization in the case of a continuous action space $\mathcal{A}$. Even though, to discretize the action space with methods like tile-coding e.g., would make applying algorithms like DQN manageable for continuous domains, but only for small action spaces in a reasonable computational time [3].

An algorithm that considers a continuous state space $\mathcal{S}$ and action space $\mathcal{A}$ is the so-called deep deterministic policy gradient (DDPG) [3]. DDPG is based on the deterministic policy gradient (DPG) in [8] combined with DQNs improvements and even significantly outperforms DQN and other algorithms [3]. NNs are used for function approximation to learn policies in a high-dimensional, continuous action space. The algorithm is based on a simple structure described by an actor-critic (AC) architecture, illustrated in Fig. 1, next page.

The AC architecture is based on the policy gradient theorem in [6] and consists of two components. One component is called critic, which is learning a Q function $Q(s, a|\theta^Q)$ by minimizing MSBE in (5). For performance reasons, the optimization is carried out by a stochastic gradient descent (SGD) algorithm [8].

The other component is the so-called actor which is usually described by a *stochastic* policy gradient (PG) algorithm. PG algorithms are a popular class of RL methods with continuous action space. Instead of learning a value function like the Q function to estimate the policy, PG algorithms directly approximate the stochastic policy $\pi_\theta(s)$ by adjusting parameters θ [8].

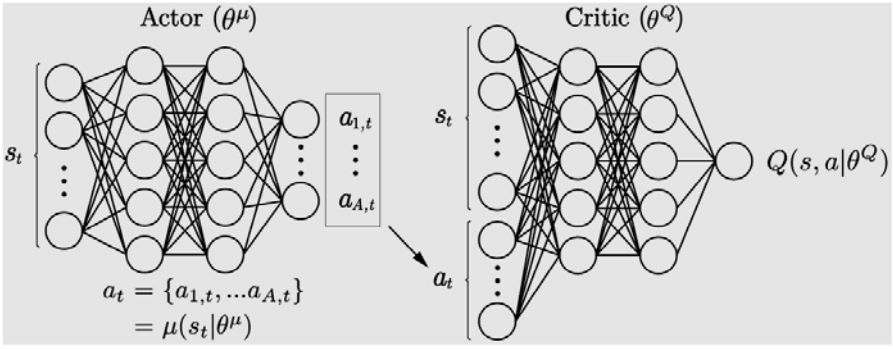

Figure 1. Representation of an actor-critic architecture, which illustrates the working principle of the DDPG algorithm.

In the case of DDPG the policy $\mu(s_t|\theta^\mu)$ is *deterministic* (in [8] it is proven that a *deterministic* PG exists) and the actor network is updated in parallel to find an action $a_t = \mu(s_t|\theta^\mu)$ that maximizes the expected return represented by critic's output $Q(s_t, a_t|\theta^Q)$:

$$\mathcal{L}(\theta^\mu) = \nabla_{\theta^\mu} J \approx \mathbb{E}[\nabla_{\theta^\mu} Q(s_t, \mu(s_t|\theta^\mu)|\theta^Q] \qquad (6)$$

with stochastic gradient ascent (SGA).

In this paper, DDPG is used to solve the RL task. The agent is written in python using the *TensorFlow* [9] library.

III. SIMULATION ENVIRONMENT

A. Reinforcement Learning Framework for Electric Power System Decision and Control Tasks

RL for electric power system decision and control isn't a new topic. In the past twenty years, different decision making and control tasks on different time scales were solved with RL in the field of electric power systems. To address these tasks the framework illustrated in Fig. 2 has proven to be suitable [10]. An agent, two different environments $\mathcal{E}$, the real system $\mathcal{E}_{\text{real}}$ and simulation model $\mathcal{E}_{\text{sim}}$ for the *execution* respectively *learning module* as well as the signals state s_t, action a_t and reward r_t are illustrated.

The *execution* and *learning module* define different stages and therefore different working modes for the agent. This division into separate tasks is necessary to avoid unsafe and unreliable system situations caused by agent's exploration in the search for an optimal policy [10].

1) Learning Module: The *learning module* is a typical RL implementation and addresses the *exploration-exploitation* dilemma [6]. In the beginning the agent must be enabled to explore random system states over a period of time. As the agent progresses, the gained knowledge is continuously exploited for decision making. This ensures that the agent converges against an (optimal) policy to solve the decision and control tasks.

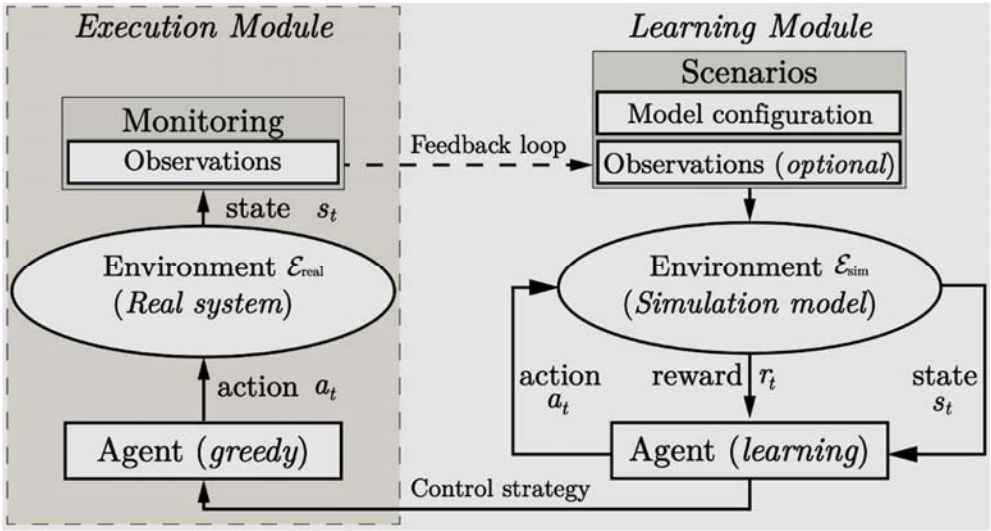

Figure 2. A Proven reinforcement learning framework for electric power system decision and control tasks [10].

To ensure exploration over the entire state space to learn a (optimal) control strategy, the following scenarios are defined and used within $\mathcal{E}_{\text{sim}}$:

- *Worst-case scenario* with high load and no power injection by PV systems.
- *Worst-case scenario* with base load (defined by 11% [11]) and high power injection by PV systems.
- Load consumption and power injection by PV systems are randomly generated in the interval between both *worst-case scenarios* to take exogenous uncertainties into account.
- Up to ten randomly positioned voltage changes $\pm\Delta V_{rel}$ per episode to simulate substation influence at RDTs secondary substation level at voltage bus 0 (slack bus).

Knowledge transfer in the form of observations in a real system can be used to enrich these scenarios. This feedback loop is used to update agent's weights and improve the control strategy since modeling the real system with a simulation model does not always capture all possible system states. Hence, a feedback loop is considered in this paper to enrich *learning module* scenarios by events observed in a real system based on [12].

2) Execution Module: Within the *execution module*, the agent uses the gained knowledge during learning in the *learning module*. In every system state s_t at time step t the agent choses the action a_t which maximizes the expected sum of total discounted rewards over future time steps with reference to the learned control strategy [10].

B. Simulation Model and Voltage Dependencies

Training the agent within the *learning module*, a simulation model is needed. For this purpose, it is necessary to create an environment $\mathcal{E}_{\text{sim}}$ that mimics the real system with a high degree of quality. In this paper, a stochastic power flow simulation is used to consider DG and voltage dependencies within a stochastic model [11].

To mimic a real environment representing an active low-voltage grid in a rural area, two groups of participants are modelled, namely households and PV systems. To model voltage dependencies on active voltage changes, *short-* and *long-term* effects are considered within the simulation model. At first, a *short-term* effect is defined which describes an instantaneous variation of active and reactive power if the supply voltage V changes. There are a variety of known load models to describe this voltage-dependency like e.g.:

$$P(V) = P_0 \cdot \left[Z_{\text{P}} \cdot \left(\frac{V}{V_0}\right)^2 + I_{\text{P}} \cdot \frac{V}{V_0} + P_{\text{P}} \right] \qquad (7)$$

$$Q(V) = Q_0 \cdot \left[Z_{\text{Q}} \cdot \left(\frac{V}{V_0}\right)^2 + I_{\text{Q}} \cdot \frac{V}{V_0} + P_{\text{Q}} \right] \qquad (8)$$

known as the ZIP model, where Z_{P}, Z_{Q}, I_{P}, I_{Q} and P_{P}, P_{Q} represent coefficients for the impedance constant (Z), current constant (I) and power constant (P) portion of active power P and reactive power Q. Active and reactive power, P_0 and Q_0 respectively, are the rated values at nominal voltage V_0.

The second effect extends the ZIP-Model to a ZIP(E)-Model. This leads to consequences on loads power consumption based of a change in supply voltage V in a *long-term*. With the ZIP(E)-Model appliances like e.g., electric water kettle or electric heater with a constant energy ($E = \text{const.}$) are explicitly taken into account. As a direct consequence, loads interval time alters with a reduction or rise in power consumption. Possible resulting scenarios are: due to the increased $\Delta t_{\text{ZIP(E)}} > \Delta t$ or decreased $\Delta t_{\text{ZIP(E)}} < \Delta t$ duty cycle $\Delta t_{\text{ZIP(E)}}$ to hold $E = \text{constant}$, appliances run in parallel or sequential at certain time intervals. At substation level, this behavior results in an effect known as duty cycle rebound (DCR) [13]. In the *long term*, the system state is affected in the opposite direction to the *short-term* effect. As a direct result, this rebound (partially) compensates the described *short-term* effect. More details are provided in [11] and [13].

To take the process of power injection by PV systems within the simulation model into account, a temperature-dependent functional chain:

$$P(t) = E_g(t) \cdot A_{\text{PV}} \cdot \eta_{\text{panel}}(\vartheta) \cdot \eta_{\text{inv}} \cdot f_{\text{cloud}} \qquad (9)$$

is provided, with the global radiation on tiled surfaces $E_g(t)$, panel and inverter efficiency, $\eta_{\text{panel}}(\vartheta)$ and η_{inv} respectively, as well as the total area of the PV panels A_{PV}. An additional factor $f_{\text{cloud}} \in (0, 1]$ is defined to simulate (partly) cloudy sky conditions and therefore reduce power injection by PV systems. With a corresponding $\cos\varphi$ the apparent power is calculated [11]. The process of energy conversion through a PV system is shown schematically in Fig. 3.

C. Tap-Operations of a Regulated Distribution Transformer

To handle the growing complexity, existing controllable assets such as RDT are considered to participate in new strategies for solving control tasks in low-voltage grids. RDT can change the voltage by $\pm\Delta V_{rel}$ per tap-operation $\pm\Delta tap_t$. The operating point Tap_t is defined by:

$$Tap_{\text{limit},-} \leq \Delta tap_t + Tap_t \leq Tap_{\text{limit},+} \qquad (10)$$

where $Tap_{\text{limit},\pm}$ are the physical limits of the RDT and $\{Tap_{\text{limit},-}, \ldots, 0, \ldots, Tap_{\text{limit},+}\} \in \mathbb{Z}$ the possible tap positions. The total number of tap positions results in $Tap_{\text{total}} = 1 + Tap_{\text{limit},-} + Tap_{\text{limit},+}$. A tap position change is modelled as an influence at voltage bus 0.

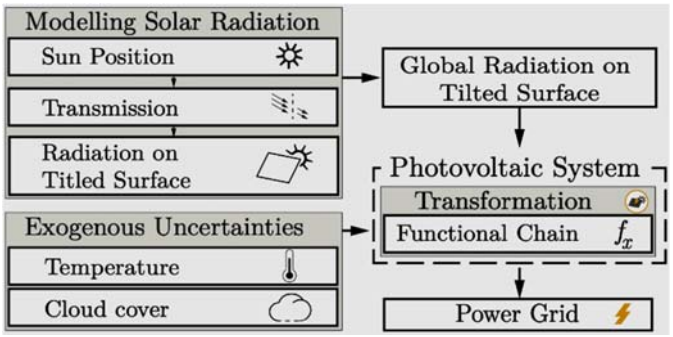

Figure 3. Energy conversion through a PV system with modeling steps of solar radiation on tilted surfaces and exogenous uncertainties

D. Signal Definition

To derive a control strategy and learn best control actions $a_t \in \mathbb{R}^A$ over all actions A the DDPG has to interact with the simulation model $\mathcal{E}_{\text{sim}}$ within the *learning module*. For this use case, the simulation model is extended to a MDP with tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ to fit into the RL framework. To interact with common python RL libraries $\mathcal{E}_{\text{sim}}$ is wrapped with a custom *OpenAI Gym* [14]. The definition of the signals state, action and reward as well as an episode is provided below:

1) State Space: The state space $s \in \mathcal{S}$ of the RL agent is defined as a vector containing tuples $(V, f_m)_{n,t}$ at specific nodes $n \in \mathcal{N}$ at time step t, where V is the measured voltage supplemented by a with factor h scaled slope $f_m = h \cdot m_{n,t}$ calculated as a linear function at node n over a window $\mathcal{K} = \{k_t, k_{t-1}, \dots, k_{t-K}\}$ with a size of $K \subset \mathcal{T}$ at time step t. Furthermore, the current operating point Tap_t of the RDT is provided. Therefore, the state space at time step $t \in \mathcal{T}$ is defined as:

$$s_{\text{t}} = \left((V, f_m)_{1,t}, \dots, (V, f_m)_{N,t}, Tap_t\right). \tag{11}$$

2) Action: A continuous action space $\mathcal{A}$ is provided to adjust RDTs operating point Tap_t between the physical limits $Tap_{limit,\pm}$ in the $\mathcal{E}_{sim}$. The action is RDTs tap-operation $\pm\Delta tap_t$ to change the voltage by $\pm\Delta V_{rel}$ per tap-operation. The control action $a_t \in \mathbb{R}^A$ with $A = 1$ is provided as a real value $a_t \in [-1,1]$ and discretized in Tap_{total} possible tap-operation by $\mathcal{E}_{sim}$.

3) Reward: When applying the RL framework to a control problem, the objective of decision making has to be carefully designed in the reward function. In this paper, the objective is defined as a voltage control problem with the focus on eliminating voltage issues at a minimal cost. To meet this goal, the reward function consists of two parts: staying in between given voltage boundaries and reducing the number of tap-operations.

The first part defines voltage zones, analogous to the grid operation limits. At time step t at each node $n \in \mathcal{N}$ a reward:

$$r_{n,t}(V, b) = \frac{100b}{\left(\frac{1}{b}|V_0 - V|^3 + 50b\right)} - 1 \tag{12}$$

with $b = 2$ is calculated. By design, agents reward results in $r_{n,t}(V \approx V_0, b = 2) = 1.0$ for node voltage $V_{n,t} \approx V_0$ close to nominal voltage V_0. Node voltages with approx. $\pm 3\%$ to nominal voltage V_0 are reinforced negatively.

The second part is described by:

$$f_m = \left|h \cdot m_{n,t} \cdot \Delta tap_t\right| \tag{13}$$

which is a scaled slope at node n for time step t times RDTs tap-operations Δtap_t. The slope $m_{n,t}$ is calculated over a window of voltage measurements $V_{n,t}$ of $K = 20$ steps.

In total, a reward:

$$r_t = \sum_{n=0}^{N} r_{n,t}(V, b) - f_{m_{n,t}} \tag{14}$$

over the sum of all measured nodes $n \in \mathcal{N}$ for time step t is provided.

4) Episode: The steady state is provided in any defined operation scenario with time $\mathcal{T} = [1{,}1440]$ covers the minutes of a day.

IV. NUMERICAL STUDY

To investigate agent's performance, a real rural low-voltage grid topology with high penetration of PV systems is used. In Fig. 4, next page, the real topology from the Lower Rhine region is shown [12]. A $S_{\text{r,RDT}} = 400\text{kVA}$ RDT supplies 114 households and 24 PV systems over two feeders with a total length of $l = 3515\text{m}$. In total, DG by PV systems accumulate to $P_{\text{DG}} = 398\text{kW}_{\text{P}}$.

A. Learning Control Strategy

Since the *learning module* is a typical RL implementation, the *exploration-exploitation* dilemma has to be addressed [6]. In continuous domains, the *exploration-exploitation* dilemma is addressed by adding noise to action a_t [3]. For the DDPG,

this results in $\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \epsilon_t$ with $\epsilon_t \sim \mathcal{W}$ sampled from a noise process $\mathcal{W}$, like Gaussian white noise which is used in this paper.

Learning the control strategy by interacting with $\mathcal{E}_{\text{sim}}$ the steps in Fig. 5 are repeated until the termination criteria is met. After initializing $\mathcal{E}_{\text{sim}}$ and loading scenario and topology information, a power flow (PF) calculation is made to provide state space information. Afterwards, the agent calculates suggested actions based on the current policy $a_t = \mu(s_t|\theta^\mu)$ and stores the transition $e_t = (s_t, a_t, r_t, s_{t+1})$ to the so-called experience replay $\mathcal{D} = \{e_1, \dots, e_t\}$ corresponding to step 3 in Fig. 5. $\mathcal{E}_{\text{sim}}$ takes this action a_t and calculates the new state s_{t+1} and the reward r_t marked as step 1 in Fig. 5. When enough transitions are collected, a uniformly drawn minibatch X of transitions $(s_i, a_i, r_i, s_{i+1}) \sim \mathcal{U}(\mathcal{D})$ is used to update agent's weights by minimizing the loss in (5) and (6). Afterwards, all three steps are executed, as shown in Fig. 5.

In Fig. 6 the evolution of DDPGs reward per episode is shown. The DDPG agent is able to quickly learn a stable and efficient policy.

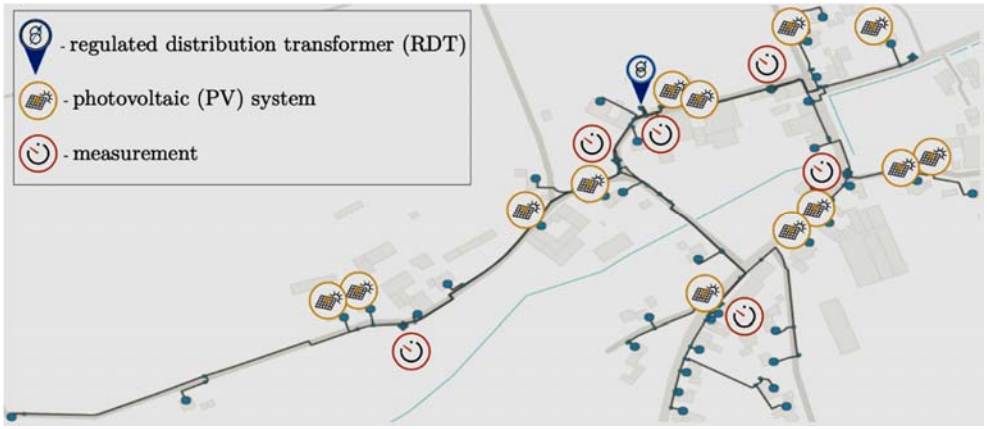

Figure 4. Real rural low-voltage grid topology from the Lower Rhine region with high penetration of photovoltaic generation [12]. Position of RDT, PV systems and measurements are marked.

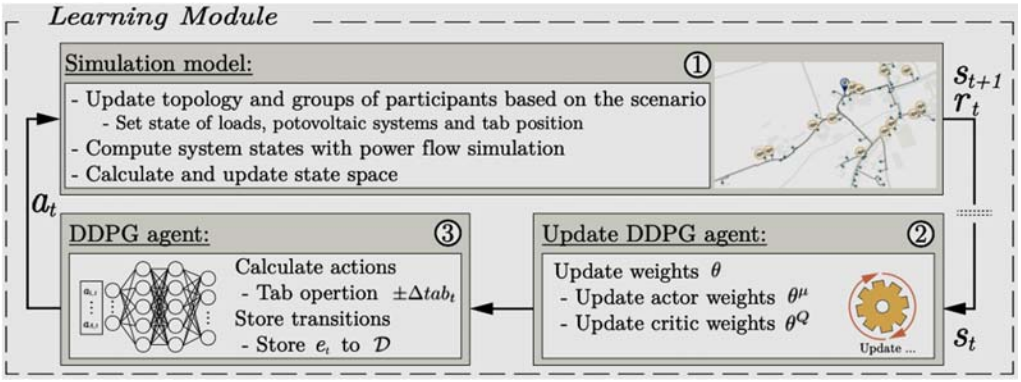

Figure 5. Steps within the learning module.

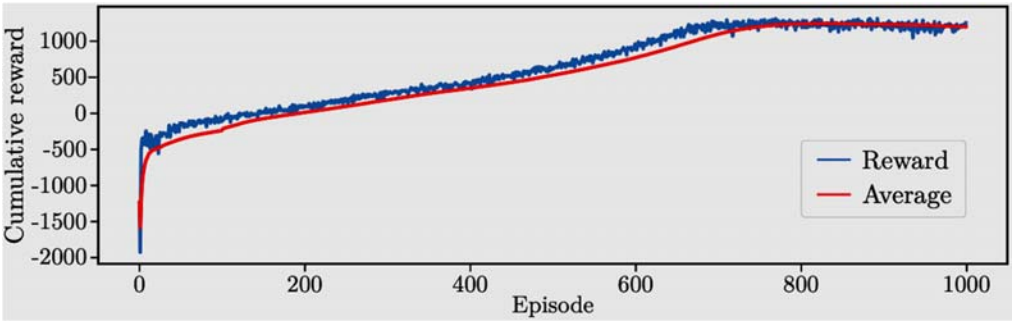

Figure 6. Evolution of agents reward per episode (blue) and a mean reward over a window of 100 episodes (red).

Based on the *exploration-exploitation* dilemma the agent selects random actions in the beginning to explore the state space. In the course of the learning process agent's decision making is continuously shifting from *exploration to exploitation*. With that been said, agent's actions are experience-based at the end of the learning process and therefore finally converges against a (optimal) policy and control strategy.

B. Execution Module

In this paper, the *execution module* is used to validate and compare agent's performance. For this purpose, the agent is tested using 41 days actual voltage measurements based on the research project in [12]. The agent is set to a *greedy* working mode. Therefore, agent's actions will maximize the expected sum of total discounted rewards over future time steps with reference to the learned control strategy at every time step t.

C. Results and Discussion

To compare the behavior of an agent with and without taking into account the described slope within the state space and reward function, a second agent (A2) is defined. In comparison to the in advanced described configuration (A1), for A2 no slope is provided and equation (13) is replaced by $f_m = |h \cdot \Delta tap_t|$.

The results in Fig. 7 indicate the behavior of the agent with (green, A1) and without (blue, A2) taking into account the described slope. Both agents are able to hold the voltage within the defined voltage boundaries, close to the nominal voltage (middle red marker), whereas less tap-operations are clearly visible for the agent A1.

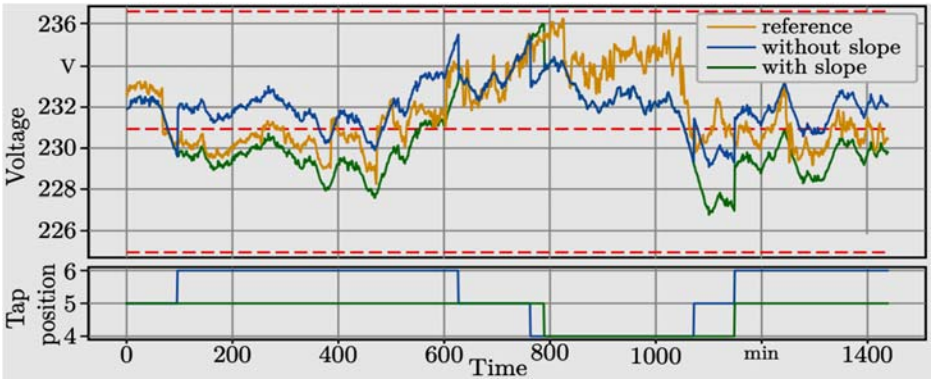

Figure 7. Plot of nominal voltage and tap position to indicate the behavior of the agent with (green) and without (blue) taking into account slope information within the state space and the reward function.

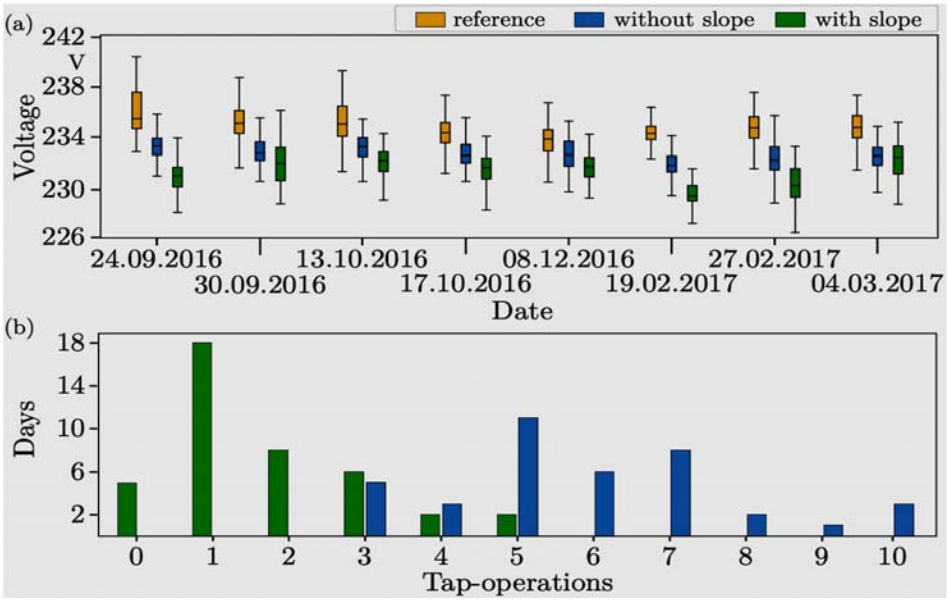

Figure 8. (a) Boxplot for three datasets over eight days. A reference voltage (yellow), the voltage after the agent A1 (green) as well as agent A2 (blue) have performed actions are illustrated. (b) Comparison of A1 (green) and A2 (blue) tap-operations over 41 validation dates.

This behavior reflects the definition of the reward function in section III-D. The first part ensures that the agent stays in between given voltage boundaries, close to a reference voltage $V \approx V_0$. To validate this behavior, a boxplot in Fig. 8 (a) is provided. Shown are three datasets over eight days. In yellow the measured voltage at RDTs position as a reference as well as the voltage after the agent A1 (green) and A2 (blue) have performed actions are illustrated. For each day, the median of the box plot is closer to the nominal voltage for A1 and A2 compared to the reference (yellow).

The second part is defined to reduce the number of tap-operations. For validation, two histograms are compared in Fig. 8 (b). Illustrated are the number of tap-operations A1 (green) and A2 (blue) has executed within the *execution module*. It is shown that in general A1 performs fewer tap-operations than A2.

V. CONCLUSION

In this paper, a DDPG based algorithm is proposed in order to find optimal tap-operations of a RDT for low-voltage distribution networks. The use of additional scenarios, a feedback loop and a detailed simulation environment within the *learning module* is presented. To reduce the number of tap-operations, additional states in conjunction with a measurement are defined. Based on node voltages, an additional slope over a window of past observations is calculated and considered within the state space and the reward function. It is shown that considering this additional information reduces the number of tap-operations.

The main advantage of the proposed RL framework with a continuous action space is to quickly adapt more participants, like e.g., PV systems. Taking a holistic approach to medium- and low-voltage distribution networks is part of our ongoing investigation regarding the intelligent use of flexibilities.

REFERENCES

[1] J. F. Franco, L. F. Ochoa and R. Romero, ”AC OPF for Smart Distribution Networks: An Efficient and Robust Quadratic Approach,” in *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4613–4623, Sept. 2018. doi: 10.1109/TSG.2017.2665559

[2] V. Mnih et al., Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://doi.org/10.1038/nature14236.

[3] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[4] J.-F. Toubeau et al., ”Deep Reinforcement Learning-Based Voltage Control to Deal with Model Uncertainties in Distribution Networks,” *Energies*, vol. 13, no. 15: 3928, 2020. https://doi.org/10.3390/en13153928

[5] H. Xu, A. D. Domínguez-García and P. W. Sauer, ”Optimal Tap Setting of Voltage Regulation Transformers Using Batch Reinforcement Learning,” in *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 1990–2001, May 2020, doi: 10.1109/TPWRS.2019.2948132.

[6] R. S. Sutton and A. G. Barto (1998), Reinforcement Learning: An Introduction, MIT Press.

[7] C. J. C. H. Watkins and P. Dayan, Q-learning. *Mach Learn* **8**, 279–292 (1992). https://doi.org/10.1007/BF00992698.

[8] D. Silver et al., ”Deterministic policy gradient algorithms,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML’14)*, vol. 32, 387–395, 2014.

[9] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.

[10] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement learning for electric power system decision and control: past considerations and perspectives,” in *IFAC-Papers OnLine*, vol. 50, no. 1, pp. 6918–6927, Jul. 2017.

[11] L. Kopczynski, P. Huppertz, M. Schallenburger and R. Zeise, ”Optimal Tap-Operations of a Regulated Distribution Transformer Considering Conservation Voltage Reduction Effects in Active Low-Voltage Grids,” in *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, 2018, doi: 10.23919/PSCC.2018.8442803.

[12] M. Schallenburger, L. Kopczynski, P. Huppertz and R. Zeise, ”Acquisition of Low-Voltage Grid States in Real-Time,” in *2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–6, 2018.

[13] H. Willis, *Power distribution planning reference book*, New York: Marcel Dekker, 2004.

[14] G. Brockman et al., “OpenAI Gym,” *arXiv preprint arXiv:1606.01540*, 2016.