Complementary reward function based learning enhancement for Deep Reinforcement learning

George Claudiu ANDREI, Mayank Shekhar JHA, Didier THEILLOL

Abstract Several complex sequential decision-making problems have been successfully implemented using reinforcement learning (RL) for continuous optimal control. However, the sample efficiency of data collection process during the learning phase is still not well addressed. The convergence rate to the optimal policy and the time of the learning process are strongly influenced by the efficiency of the data collected during the learning phase by the agent. This paper proposes a method to generate efficient sample data which allows the agent to collect high reward trajectories more frequently decreasing the learning phase time. The proposed method consists of Complementary reward (CR) function augmented to the traditional reward function. The CR tends to infinity when the control input leads to the system performance that meets the given requirements very accurately. Consequently, the control policy that maximizes the reward function can render the system to optimal performance. The main contributions of this study include the following aspects: (1) a new proposed Complementary reward that is augmented to the reward function which improves performance of the reinforcement learning based controller in terms of system requirements; (2) speed-up of training phase via generation of more efficient data resulting in a better learned policy.

Centre de Recherche en Automatique de Nancy (CRAN), UMR 7039, CNRS
Université de Lorraine, 54506 Vandoeuvre-lès-Nancy Cedex, France
George Claudiu ANDREI, e-mail: george.andrei@etu.unilim.fr
Mayank Shekhar JHA, e-mail: mayank-shekhar.jha@univ-lorraine.fr
Didier THEILLIOL, e-mail: mayank-shekhar.jha@univ-lorraine.fr

1 Introduction

Reinforcement Learning (RL) has become one of the most important and useful approach in control engineering. RL uses a trial-and-error learning process to maximize a decision-making agent's total reward observed from the environment. Compared to optimal control theory, this maximization problem can be viewed as minimizing the cost function since the reward function is designed based on the control problem and system requirements. Recently, significant progress has been made by combining advances in deep learning (DL) with RL methods for solving optimal control problems (Krizhevsky et al.,2012) resulting in the "Deep Q Network" (DQN) algorithm (Mnih et al., 2015). DQN solves RL problems with high-dimensional observation spaces but it can deal only with discrete action spaces. Deep Deterministic Policy Gradient (DDPG) method was introduced in 2015 [1] in order to overcome the curse of dimensionality problem as well as to deal with continuous spaces. DDPG requires an actor-critic framework making the algorithm easy to implement and applicable to optimal control problems characterized by continuous spaces. A key mechanism used in the DDPG algorithm is the use of replay buffers to store trajectories of experience in order to extract a batch of samples during training phase as well as break the correlation between data. A prioritized experience replay buffer was introduced as improvement of the previous technique in order to make the samples characterized by high rewards be selected more frequently [1]. The efficiency of the stored trajectories in terms of reward defines training phase duration and convergence of the policy: more efficient collected trajectories in terms of high reward are and the faster the agent will learn the optimal policy. This process could make the training phase very slow and increases the agent's ability to learn a rather sub-optimal policy. This paper proposes an approach to generate efficient sampled data during the learning phase which consists of a Complementary reward (CR) function augmented to the traditional reward function in order to guide control learning for speeding it up and favouring the convergence to the optimal policy. Similar to the Barrier function concept [5], the proposed CR tends to infinity when the control input leads to the system performance meet its requirements very accurately. The main contributions of this study include the following aspects: (1) a new proposed Complementary reward that is augmented to the reward function which improves performance of the reinforcement learning based controller in terms of system requirements; (2) speed-up of training phase via generation of more efficient data resulting in a better learned policy.

The rest of the paper is organized as follows. Section 2 provides a general state-of-the-art of reinforcement learning including DDPG algorithm. Section 3 provides more details on RL-based control for balancing a rotary inverted pendulum formulation problem. The proposed method using Complementary function-based reinforcement learning will be discussed in Section 4. Finally, the results of simulation and its analysis in comparison with traditional reward function method is discussed in Section 5. Conclusions are outlined in Section 6.

2 Preliminaries

RL involves an agent exploring an unknown environment to achieve a goal: the agent builds up its knowledge of the environment by gaining experience. Beyond the agent and the environment, it is required to identify four main sub-elements that need to be defined in a RL problem : a *policy*, a *reward signal*, a *value function* [4]. A *policy* π defines the learning agent's way of behaving at a given time t. A *reward signal* $r_{t+1} = r(s_t, a_t)$ defines the goal in a RL setting and it is used for measuring the performance of the agent based on the same concept as cost function in control theory. The objective of the agent is to learn an optimal policy by maximizing the accumulated discounted reward $\sum_{i=t}^{T} \gamma^{i-t} r(s_t, u_t)$ from current time step t to a future time step T, where $\gamma \in (0\ 1)$ is the discount factor which discounts the value of future rewards. A *value* of a state usually denoted as $V_\pi(s_t)$ is the expected accumulated reward over the future following the policy π and starting from a particular system state s_t: $V_\pi(s_t) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t]$, where E_π is the expected value under policy π. By considering how good it is to be in a state by taking into account the action taken by the Agent, it is necessary to refer to the *action-value function*, also known as *Q-function* or *Q-value function* as $Q_\pi(s_t, a_t)$.
$Q_\pi(s_t, a_t)$ gives the expected return for performing action a_t in state s_t at time step t, under the policy π: $Q_\pi(s_t, a_t) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t, a_t]$.
Generally, RL algorithms are based on Markov decision process (MDP) framework for modeling the environment, the policy and the agent. It is defined by the tuple (S, A, r, P):

1. A set of states, S, which contains all possible states of the environment. In other terms, all the possible discrete or continuous measurable outputs.
2. A set of possible actions, $A(s)$, which contains all possible actions or control inputs which can be applied to the system in each state.
3. A transition model, $P(s_{t+1}|s_t, a_t)$, which denotes the probability of reaching state s_{t+1} when performing action a_t in state s_t in case of stochastic environments. It can be assumed equal to one in deterministic environments.
4. A reward function $r_{t+1} = r(x_t, u_t)$, which is the immediate reward perceived after transition from state s_t to s_{t+1} as consequences of taking an action a_t.

2.1 Deep Deterministic Policy Gradient Method

DDPG is a *model-free*, *off-policy* and *actor-critic* framework based algorithm with continuous action spaces, proposed by Dr. Lillicrap et al. in 2015. This method is an extension of two other algorithms, Deep Q-Networks (DNQ) and Deterministic Policy Gradient (DPG) which uses the experience replay and target network as main techniques. DDPG uses two neural networks in the actor-critic and it employs the use of off-policy data and the Bellman equation to learn the Q-function $Q(s_t, a_t)$ which is then used to derive and learn the policy π:

$Q_\pi(s_t, a_t) = E_{s_{t+1}\sim E}[r_{t+1} + \gamma E_{a_{t+1}\sim\pi}[Q_\pi(s_{t+1}, a_{t+1})]$, where r_{t+1} is the reward observed from the environment after the action a_t at time step t, $s_{t+1} \sim E$ means that the transition is sampled from the environment defined as E, and $a_{t+1} \sim \pi$ means that the action is sampled from policy π. If the policy is deterministic, it is usually denoted as μ, and the inner expectation of the Bellman equation can be avoided: $Q_\mu(s_t, a_t) = E_{s_{t+1}\sim E}[r_{t+1} + \gamma Q_\phi(s_{t+1}, \mu(s_{t+1})]$. Q_μ can be learned off-policy, by using transitions generated by a different policy β since the expectation only depends on the environment. The function approximator is represented by the critic neural network parameterized by θ_ϕ and it is used to approximate the Q-function. The mean-squared error can be used as loss function to be minimized in the optimization process: $L(\theta_Q) = E_{s_t\rho^\beta, a_t\sim\beta, r_{t+1}\sim E}[(y_t - Q(s_t, a_t)|\theta_\phi)^2]$. y_t is the *target value*: $y_t = r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta_\phi)$, where ρ^β is the discounted state transition for the policy β, and $\mu(s_t) = \underset{a}{\text{argmax}} Q(s_t, a_t)$ is the policy defined by acting greedily. The deterministic policy $\mu(s_t)$ is represented by the actor neural network parameterized by θ_μ. The Bellman equation is then used as in a conventional RL method to update the critic network. The actor neural network is updated using the following sampled policy gradient to maximize the expected discounted reward: $\nabla_{\theta_\mu} J \approx E[\nabla_{\theta_\mu} Q(s_t, \mu(s_t|\theta_\mu)|\theta_\phi)]$. The off-policy data defined as experience tuple $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ is generated during the training phase and then is stored in a replay buffer in order to break the inter-correlations between experiences which are sampled from the memory during the updating phase of the networks. This procedure makes the training phase easier to converge by leading the target value as well as Q-function prediction be independent from each other resulting in a more stable learning. The idea is that the weights of the target networks are initialized as copies of the actor and critic networks weights, but updated more slowly to keep them fixed for some time steps. Finally, the parameters of the target network are updated using moving averages for both actors and critics:

$$\theta_{\phi'} \leftarrow \tau\theta_\phi + (1 - \tau)\theta_{\phi'} \tag{1}$$

$$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1 - \tau)\theta_{\mu'} \tag{2}$$

where $\tau << 1$ is the soft target update rate which makes the target networks change slowly improving learning stability.

An important problem in RL is the trade-off between *exploration* and *exploitation*. To achieve high rewards, the agent has to choose actions by applying a control input it has tried before and knows to be a good one. *Exploration* refers to the RL agent exploring the environment to collect unseen trajectories with high rewards. *Exploitation* means simply following the current better policy from the experiences collected in the past to gain as much reward as possible. Thus, DDPG uses a noise sampled from a noise process N is added to the actor policy when selecting an action during training as exploration law: $\pi(s_t) = \mu(s_t|\theta_\mu) + N$.

This is called action noise. Different noise processes can be used, the original DDPG paper [19] suggest an *Ornstein-Uglenbeck process* explained in [3], which is time correlated.

3 Problem formulation

In this paper, the balance control of a rotary inverted pendulum system based on the DDPG algorithm has been implemented. The control of a pendulum has been one of extensively applied problems in control field and its choice is justified by the fact that balancing an inverted pendulum in up-right position from the its stable position is not a trivial task. The rotary inverted pendulum consists of a motor arm, which is actuated by a DC servo motor, with a swinging pendulum arm attached to its end. The measured output coming from the system is defined by the motor angle $\theta \in [-\frac{\pi}{2} \ \ \frac{\pi}{2}]$ and by the inverted pendulum angle $\alpha \in [-2\pi \ \ 2\pi]$. The set of all possible values of the two angles among with their rate of change defines the continuous observation space: $S \in [\theta \ \ \alpha \ \ \dot{\theta} \ \ \dot{\alpha}]$. The unstable equilibrium point defines the reference point positioned at the origin of the reference system: $\alpha_0 = 0$ and $\theta_0 = 0$. The control input u_t is the applied voltage by the DC Motor and the set of all possible values defines the continuous action space: $A \in [-10 \ \ 10]$ V. The reward function indicates whether the task is successful or not during the training phase and it is one of the most critical part of the RL design.
The reward function is then designed: $r_{t+1} = -\left(q_{11}\theta_t^2 + q_{22}\alpha_t^2 + q_{33}\dot{\theta}_t^2 + q_{44}\,\dot{\alpha}_t^2\right)$. The quadratic reward function indicates the agent how far or close his performance is from the reference position. It has been designed in such a way as to explicitly emphasizes the importance of minimizing the control error with respect to the up-right position of the inverted pendulum ($\alpha_{ref} = 0$ and $\theta_{ref} = 0$) positioned at reference system origin. Consequently, the learned policy has to make the control error e_t converge to zero as close as possible at each time step t:

$$e_t = |e_\alpha| \simeq 0 \ \text{ with } \ e_\alpha = \ |\alpha_{ref} - \alpha| = \ |0 - \alpha|. \tag{3}$$

The absolute value is used to not differentiate between positive (counter clockwise) or negative rotations (clockwise) of the inverted pendulum.
Additionally, the motor angle's position needs to be controlled as addition system requirement: $\theta \in$ [-30 30] [deg] when balancing the inverted pendulum.

4 Complementary reward function method

In RL problems, the agent learns the optimal control policy which maximizes the reward function by exploring the range of control strategies and selecting the optimal control strategy which maximizes the reward function. In this context, a new method is proposed to speed-up the control learning process and favouring the convergence to the optimal policy by generating more trajectories with high rewards. Such trajectories are generated by a CR included in the reward function which tends to infinity when the system response leads to optimal performance making the agent collect a higher reward during the learning phase. In this sense, in the case of balancing control problem defined in section 3, the following CR is proposed:

$$\mathrm{C}(s_t, e_\alpha, \dot{\alpha}) : \begin{cases} e^{\left(\frac{1}{\gamma|\alpha ref - \alpha|}\right)} + e^{\left(\frac{1}{\gamma|\dot{\alpha}|}\right)} & \text{if } \theta < \pm \mathrm{A} \text{ and } \alpha < \pm \mathrm{B} \text{ and } \dot{\alpha} < \pm \mathrm{C} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where γ is the damping coefficient defining the speed divergence of the function with respect to the control error e_t, q_{55} is a weight parameter, A and B and C are CR conditions to be fixed based on the control problem or system requirements.

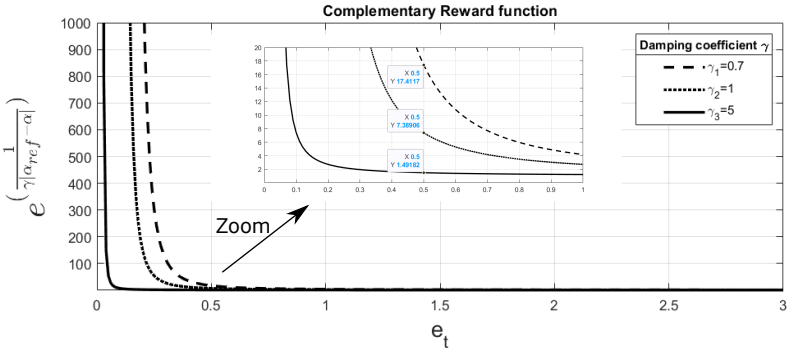

Fig. 1: Complementary function with different damping coefficient

As shown in Fig. 1, the proposed CR is inversely proportional control error (see Eq. 3) tending to infinity when e_α is very close to zero. The rate of convergence of the CR is dictated by the damping coefficient as shown in the zoom sub-figure. The choice of γ is very important and must be chosen according to system requirements and control problem.
The CR is included in the DDPG agent reward function defined in section (3:

$$R_{t+1} = r_{t+1} + C(s_t, e_\alpha, \dot{\alpha}) \tag{5}$$

where r_{t+1} is the reward function defined in (3), $C(s_t, \alpha, \dot{\alpha})$ in (4).
The traditional reward function becomes negligible when at least e_α is close to zero: $R_{t+1} = r_{t+1} + C(s_t, e_\alpha, \dot{\alpha}) \simeq C(s_t, e_\alpha, \dot{\alpha})$ if at least $e_\alpha \simeq 0$ and $\dot{\alpha} \simeq 0$.

The corresponding pseudo-algorithm concerning the proposed method implemented with DDPG algorithm is given below 1 with M, T and n being the total number of episodes, episode duration and number of samples respectively.

Algorithm 1 DDPG using Complementary reward function

Step 1. Initialization
Randomly initialize actor network $\mu(s_t|\theta_\mu)$ and critic network $Q(s_t, a_t|\theta_\phi)$
with weights θ_μ and θ_ϕ respectively
Copy the weights to target network $\mu^{'}$ and $Q^{'}$, $\theta_{\mu'} \leftarrow \theta_\mu$, $\theta_{\phi'} \leftarrow \theta_\phi$
Initialize replay buffer
Step 2. Exploration
for episode = [1:M]
Initialize the noise process N ~ OU for exploration
for t = [1:T]
Obtain the current system state s_t from the environment
Generate control input action $a_t = \mu(s_t|\theta_\mu) + N$ based on the current actor policy
exploration action noise
Execute action a_t, then **if**: e_θ < A and e_α < B, receive reward $R_{t+1}(5)$ otherwise r_{t+1}
Obtain the resulting system state s_{t+1}
Store experience (s_t, a_t, R_{t+1} *or* r_{t+1}, s_{t+1}) into the replay buffer
Step 3. Update
Extract a mini-batch of n experiences (s_t, a_t, R_{t+1} or r_{t+1}, s_{t+1}) from replay buffer
Compute the target value: $y_t = r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta_\phi)$
Update critic network by minimizing the loss $L = \frac{1}{n}\sum_i (y_i - Q(s_{i+1}, a_{i+1}|\theta_\phi))^2$
Update the actor policy by using the policy gradient
$\nabla_{\theta_\mu} J \approx \frac{1}{n}\nabla_{\theta_{\mu i}} Q(s_i, \mu(s_i|\theta_{\mu i})|\theta_\phi)$
Update the target networks
$\theta_{\phi'} \leftarrow \tau\theta_\phi + (1-\tau)\theta_{\phi'}$
$\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'}$
end
end

To demonstrate the effectiveness of the proposed approach while using DDPG algorithm for optimal control, a comparison between traditional method and the proposed one has been carried out. The training settings, hyper and weighted parameters, neural networks configuration remain the same. The training phase duration is set to 600 episodes whose duration is defined by 2500 number of steps each resulting in 25 seconds since the sample time is set to 0.01. The learned policy by the agent is then tested on the same simulink model that was used to train the agent for both methods with simulation time of 10 [sec]. The position of the pendulum and motor angles are always initialized in the same way at the beginning of each training episode: $\alpha_0 = \pm\pi$ [rad] and $\theta_0 = 0$ which consist on vertically down position for the pendulum and motor angle positioned at the origin of the reference frame respectively, During each episode, the agent will generate online trajectories by observing the measured output and reward coming from the environment and will subsequently use them to improve and update their policy.

4.1 Traditional reward function approach

The trained agent performance without using CR function is first analyzed. Fig. 2 shows the output response of the inverted pendulum angle α with respect to the unstable equilibrium point $\alpha_{ref} = 0$. As it can be noticed, the trained agent is not able to balance the pendulum in up-right position but leads to oscillations around the stable equilibrium point which is positioned at α = +180 [deg] as well as α = -180 [deg] depending on the rotation sense. On the other hand, Fig. 3 shows that θ output response is respecting the system requirements of keeping the motor angle in between ± 30 [deg]. These performances demonstrate that the agent learnt a sub-optimal policy and it would need to be trained for more than 600 episodes for converging to a better policy. The sub-optimal convergence can be observed in the episode reward function as shown in Fig. 6 where the steady state response has an offset of -500 with respect to zero episode reward which is the maximized one.

4.2 Complementary function approach

The implementation of the proposed method via CR is carried out with the same training and structure of the neural networks and a damping coefficient $\gamma = 0.7$. Fig. 4 shows the inverted pendulum angle α with respect to the reference position $\alpha_{ref} = 0$. As it can be noticed, the trained agent is able to keep the inverted pendulum in up-right position in less than 1 [sec]: $\alpha \pm 5$ [deg]. However, the trained agent is not able to balance the pendulum for more than one 1.5 [sec] since the learned policy is not optimal. Fig. **??** shows that θ output response is respecting the system requirements of keeping the motor angle in between ± 30 [deg]. This improvement in terms of performance and convergence to a better policy in the same amount of episodes took place thanks to the introduction of the CR which allowed the agent to collect higher rewards with respect to the traditional approach resulting in a better performance. From episode reward shown in Fig. 7 is possible to notice that at the beginning of the training phase the reward function response is similar to the traditional approach until the agent is within the CR conditions. As soon as the agent collects high trajectories, he focuses on generating others with higher reward by updating the policy. Consequently, CR speed-up the convergence rate to a better policy with respect to the traditional approach.

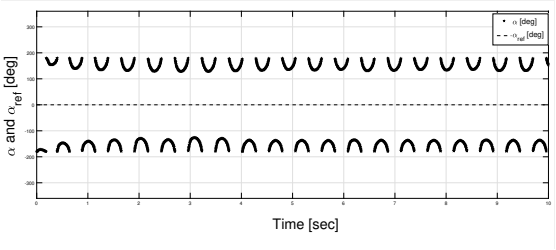

Fig. 2 Inverted Pendulum angle response with no CR

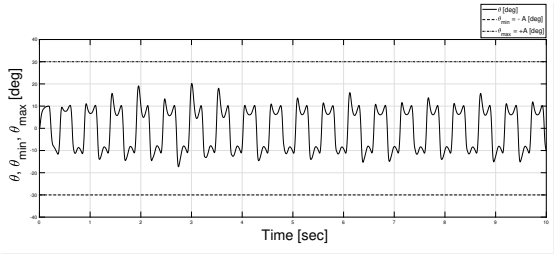

Fig. 3 Motor angle response with no CR

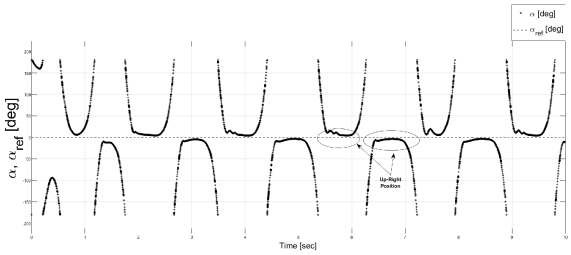

Fig. 4 Inverted Pendulum angle response with CR and $\gamma = 0.7$

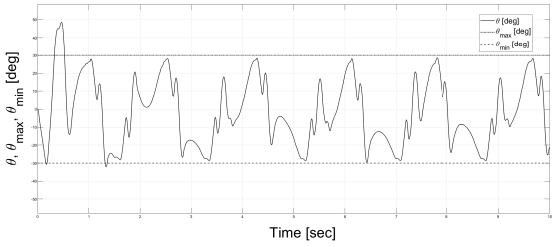

Fig. 5 Motor angle response with CR and $\gamma = 0.7$

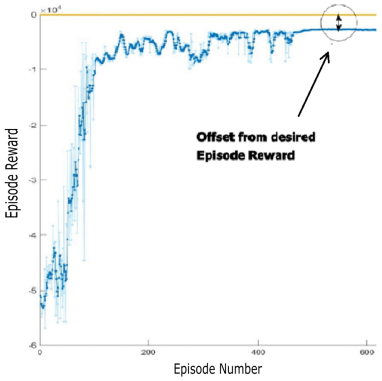

Fig. 6: Episode reward function with no CR

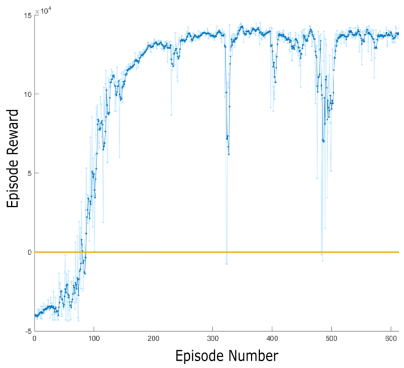

Fig. 7: Episode reward function with CR and $\gamma = 0.7$

5 Conclusions and Future work

The proposed approach improves the policy learning by speeding up its convergence rate via Complementary reward function making the agent able to learn a better policy in less amount episodes thanks to its capability of rewarding the agent with high values when the control task achieves the required performance levels. The proposed approach can be implemented in any kind of RL problems since it acts at the level of reward function design which is a step inherent to all the RL algorithms. Thus, the proposed CR function concept can be viewed as a general contribution to RL based approach. The comparison between the traditional method and the proposed one demonstrates that Complementary function based approach can provide a better performance under similar conditions. The next step will be to tune the hyper parameters of the CR function, in particular, the damping coefficient in order to speed-up the convergence of the optimal policy in as few episodes as possible.

References

[1] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2015. DOI: 10.48550/ARXIV.1509.02971. URL: https://arxiv.org/abs/1509.02971.

[2] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. arXiv:1509.02971 [cs, stat]. July 2019. DOI: 10.48550/arXiv.1509.02971. URL: http://arxiv.org/abs/1509.02971 (visited on 08/19/2022).

[3] Matthias Plappert et al. *Parameter Space Noise for Exploration*. arXiv:1706.01905 [cs, stat]. Jan. 2018. DOI: 10.48550/arXiv.1706.01905. URL: http://arxiv.org/abs/1706.01905 (visited on 08/19/2022).

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. ISBN: 9780262039246.

[5] Thanh Long Vu et al. *Barrier Function-based Safe Reinforcement Learning for Emergency Control of Power Systems*. 2021. DOI: 10.48550/ARXIV.2103.14186. URL: https://arxiv.org/abs/2103.14186.