Supplementary Primary Frequency Control through Deep Reinforcement Learning Algorithms

Francisco Zelaya-Arrazabal, Timothy Thacker, Héctor Pulgar-Painemal, Zhenping Guo
Department of Electrical Engineering and Computer Science.
University of Tennessee, Knoxville.
{fzelayaa, tthacke3, zguo19}@vols.utk.edu, hpulgar@utk.edu,

***Abstract*—This work presents the implementation of deep reinforcement learning (DRL) agents as supplementary primary frequency controllers. To achieve this, the primary frequency regulation problem is formulated in a DRL framework, where an actor-critic algorithm, for continuous actions space, is used to change the frequency reference of traditional governors. By modifying this reference, the DRL agent effectively reduces the magnitude of the frequency nadir and rate of change of frequency, thereby enhancing the power grid frequency response. Two DRL algorithms including Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3) are employed for the frequency regulation. The supplementary control using these two algorithms is tested on a 14-bus, 5-machine test system. The results show that the frequency stability of the grid can be improved by using DRL algorithms as supplementary controllers in the primary frequency regulation.**

***Index Terms*—Primary frequency control, frequency stability, deep reinforcement learning, TD3, DDPG.**

2023 North American Power Symposium (NAPS) | 979-8-3503-1509-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/NAPS58826.2023.10318681

I. Introduction

Changes in the load-generation balance lead to frequency deviations due to the electromechanical coupling among all synchronous generators (SGs) in the grid. To address deviations, each SG governor regulates the valve/gate position to adjust the injection of the working fluid—steam, water, or gas—and regulates the mechanical input of the generator. As a consequence, the injection of electrical power into the grid is also regulated. Governors are provided with droop control, which is a characteristic given by the ratio of frequency deviations to changes in valve/gate position [1]. This droop characteristic is defined for each SG, such that for the same frequency deviation, each SG provides injects power according to its own regulating capabilities. In general, this is called primary frequency control (PFC), which is in charge of dealing with the frequency nadir, the rate of change of frequency (ROCOF), and the post-disturbance frequency prior to the automatic generation control (AGC) actuation. In the context of conventional power grids with reduced participation of variable renewable energy resources (VRE), the droop characteristic is enough to provide frequency regulation. However, in the case of new power grids with the increased participation of VRE, distributed energy resources (DERs), and smart loads, dealing with frequency regulation is a bigger challenge, primarily because of the decommissioning of SGs, the inertia reduction, and the increased variability of the grid [2].

Currently, PFC faces several specific challenges. First, the droop characteristics become sub-optimal because the SGs participation and inertia condition in the grid is more variable. Second, there is a lack of coordination between SGs, VRE, and DERs. Furthermore, the droop mechanism introduces a delay in actuation, which can be further exacerbated by inverter-based resources with limited power injection support. Consequently, the need arises for new adaptable solutions capable of adjusting the response according to the system condition, disturbance, and VRE/DERs participation.

In this work, DRL algorithms are used to provide more intelligence and adaptability to the SGs governors. Previous works have explored secondary frequency regulation [4] and primary frequency regulation [5]. The former was implemented using ANDES [6] and OpenAI Gym [7], which are Python libraries for power systems analysis and RL projects, respectively. The latter was implemented in Matlab/Simulink using a basic single-machine finite-bus system. In this work, the IEEE 14-bus system is used as an environment together with AndesGym [4] to implement an agent for continuous action space. The contributions of this work are 1) DRL primary frequency controller (DR-PFC), the primary frequency regulation problem is put in a detailed DRL framework, then actor-critic methods are used as the engine motor of DRL-agent implementation. 2) Comprehensive analysis of DRL algorithms for PFC. Sensitivity analysis is performed using DDPG to understand how hyperparameters could affect DRL agents. Moreover, a final comparison with TD3 is performed, showing its superior capability to deal with continuous action spaces. The rest of this paper is structured as follows: Section II presents the basic concepts of DRL and the DRL-PFC idea; Section III describes its implementation; Section IV presents the study case including results, and finally, the conclusions are presented in Section V.

II. Primary frequency response using DRL

A. Deep reinforcement learning basics

Reinforcement learning is a sub-field of machine learning where an agent uses trial and error to solve problems modeled as a Markov Decision Process (MDP). An MDP, as shown in Figure 1, describes the typical RL problem where an agent interacts with an environment and receives feedback in the form of the environment state s_{t+1} and reward r_{t+1}

This work was supported in part by the National Science Foundation (NSF) under Grant No. 2033910, and in part by CURENT, which is an NSF Engineering Research Center funded by NSF and the Department of Energy under NSF Award EEC-1041877.

corresponding to the action $a_t \in A$, where A is the set of all possible actions. The goal of the agent is to learn an optimal policy π^*, a set of actions corresponding to observed states, that maximizes the expected future rewards. Thus, this policy is obtained as,

$$\pi^* \in \arg\max_{\pi} \left\{ J(\pi) = \mathbb{E}_\pi \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \right\} \tag{1}$$

were π^* is an optimal policy that maximizes the infinite horizon discounted reward or expected return $J(\pi)$ given by the expectation of the sum of all future rewards r weighted by the discount factor γ. The discounted factor allows the convergence of the summation, otherwise for non-episodic problems this result will not be finite [9], [10].

DRL algorithms combine artificial neural networks and traditional reinforcement learning approaches to solve MDPs with high-dimensional states and action spaces. The agent is able to learn complex actions and emulate any nonlinear behavior through deep neural networks while interacting with the environment inside the MDPs.

Prior to presenting DRL on primary frequency control, it is important to define the concept of action space, which represents the set of all possible effective actions that an agent can take. According to this, reinforcement learning algorithms can be divided into three different types: discrete action space, continuous action space, and hybrid action space. The first one refers to those problems in which the action set is countable, the second considers an uncountable number of possible actions, and the third is a combination of both [11]. This paper will focus on algorithms for continuous action space problems since a power grid is a large environment that is constantly changing, making it impossible to quantify the number of possible actions and states.

B. Deep reinforcement learning primary frequency control

To explain the DRL supplementary primary frequency control, consider a IEESGO governor model [8]. By setting $K_2 = K_3 = 0$, the reduced set of differential-algebraic equations associated with the model becomes,

$$T_1 \dot{y_1} = -y_1 + K_1(\omega - 1) \tag{2}$$

$$T_3 \dot{y_3} = -y_3 + y_1 \tag{3}$$

$$T_4 \dot{T_m} = -T_m + P_V \tag{4}$$

$$y_2 = \left(1 - \frac{T_2}{T_3}\right) y_3 + \frac{T_2}{T_3} y_1 \tag{5}$$

$$P_V = \begin{cases} P_C - y_2, P_{min} \leq P_C - y_2 \leq P_{max} \\ P_{max}, P_C - y_2 \geq P_{max} \\ P_{min}, P_C - y_2 \leq P_{min} \end{cases} \tag{6}$$

where ω is the SG speed in p.u., K_1 is the inverse of the droop, P_V is an intermediate variable used to implement P_{max} and P_{min} limits, T_m is the mechanical torque, y_1, y_2, and y_3 are internal variables of the model, and T_1, T_2, T_3, and T_4 are time constants [12]. If an external signal μ_c is added to Eq. 2, this differential equation is modified to:

$$T_1 \dot{y_1} = -y_1 + K_1 \left[\omega - (1 + \mu_c)\right] \tag{7}$$

such that the governor speed reference is modified from ω_s to $\omega_s + \mu_c$. By adding this new signal, the governor's actuation can be manipulated according to a DRL agent. The block diagram of this implementation is presented in Fig. 2. The new signal will modify the frequency reference so the governor can observe a bigger or smaller frequency deviation according to the training conditions. As will be shown in the next sections, this will result in an improvement in the governor's response.

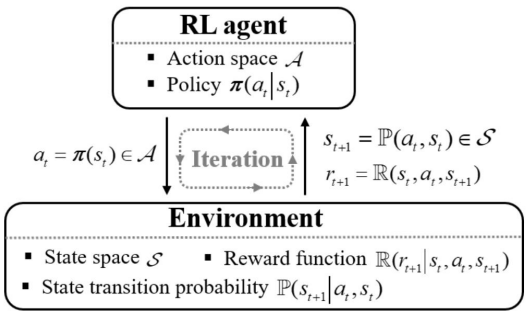

Fig. 1: MDP process: agent-environment interaction. [9]

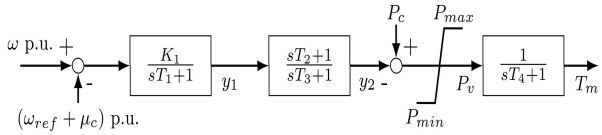

Fig. 2: Simplified IEESGO governor model with coordinating signal μ_c.

III. DRL-PFC Implementation

This section addresses the implementation of the deep reinforcement learning supplementary primary frequency control (DRL-PFC). The power grid is put into a DRL framework, including the action space, observations, and reward function.

A. Primary frequency control in a DRL framework

1) State vector and observations: Ideally in a DRL framework, all the states can be mapped. For example, consider the well-known Atari game "Pong", which consists of two paddles hitting one ball, trying to throw the ball outside the opponent side. The state space is given by the entire set of combinations formed by the position of the ball, its direction, and the position of the two players paddles. These three features are enough to observe the environment and provide the necessary feedback to the agent so it can learn the necessary policy to optimize the agent actuation [13]. However, this is an example of a discrete problem, in which the state space is countable. In a continuous problem, it will be impossible to map all the states of the system. As a result, the most relevant

observations must be selected to capture the different dynamics of the environment. In the frequency regulation problem, the observation must capture those features associated with the primary frequency regulation problem, i.e., the frequency nadir, the ROCOF, and the post-disturbance frequency. In this work, the frequency ω and ROCOF $\dot{\omega}$ at SGs buses are used as observations. Then, consider a system with N SGs, the observations vector at time t is defined by:

$$\Omega_t = [\omega_{t,1}, \omega_{t,2}, ..., \omega_{t,N}] \tag{8}$$

$$\dot{\Omega}_t = [\dot{\omega}_{t,1}, \dot{\omega}_{t,2}, ..., \dot{\omega}_{t,N}] \tag{9}$$

$$s_t = \Omega_t \cup \dot{\Omega}_t \tag{10}$$

2) Action space: The agent action is formed by all the supplementary signals among all the implicated governors in the supplementary control, such that the a_t vector is given by Eq. 11,

$$a_t = [\mu_{t,1}, \mu_{t,2}, \mu_{t,c}, ..., \mu_{t,M}] \tag{11}$$

where $\mu_{t,c}$ is the governor frequency modification at time t for generator c, and M is the total number of SGs participating in the supplementary control. By placing hard limits on the action space, the agent can quickly learn to take actions that will improve the system without risking instabilities or collapse. In this work, the action is limited to the range [-0.1, 0.3] as the system remained stable under consistent application of the extremes, and appropriate values within this range resulted in considerable improvements to the primary frequency response.

3) Reward function: The reward function is an engineered function that, given certain observations (not necessarily the same from section III-A1), will provide the agent with the feedback of taking a_t at state s_t. This feedback quantifies the action taken such that the sum of the discounted rewards can be used to obtain the optimal policy π^*. This can be seen as an incentive or a penalizing mechanism, that leads the agent to take wiser decisions as the training progresses. In terms of primary frequency control, the reward function should consider the actual frequency, the ROCOF, the frequency nadir, and the post-disturbance frequency. To make a practical implementation, the post-disturbance frequency is assumed to be known and is computed through simulations considering a stable scenario. Future research will explore the estimation of the post-disturbance frequency to consider it in the training process.

For this problem, the reward function is given by,

$$r_t = \begin{cases} -\beta_1|f_t - f_s| - 1000(f_s - f_t)(f_t < f_s) \\ -1000(f_t - f^*)(f_t > f^*) \\ -\beta_2|ROCOF| & \text{if } t = T_f \\ \\ -\beta_3|f_t - f_s| - 1000(f_s - f_t)(f_t < f_s) \\ -1000(f_t - f^*)(f_t > f^*) \\ -\beta_4|ROCOF| & \text{if } t \neq T_f \end{cases} \tag{12}$$

where f_t is the frequency at time t, f_s is the post-disturbance, which is our target frequency for the primary frequency control, f^* is the nominal frequency, T_f is the final instant, and all β_i values are scalar weights. The reward is combined with four terms: the first one represents the penalty induced by the absolute difference between the current frequency and the post-disturbance frequency; the second one represents the penalty induced by the excessively low frequency; the third one represents the penalty induced by the over-high frequency; finally, the fourth term is a penalty given by the magnitude of the ROCOF. Weights β_1 and β_2, which are respectively associated with the magnitudes of the frequency difference and ROCOF, are chosen to be bigger negative coefficients than β_3 and β_4. This ensures that the final frequency will be closer to the target frequency.

B. Continuous action space DRL algorithms

Once the RL framework has been defined, the next step is to select an adequate algorithm to solve this problem. This work uses Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3). The former has been the most popular algorithm for learning in continuous action space, while the latter is its improved version. These two algorithms are policy-gradient-based methods with an actor-critic structure, in which two main models participate: the actor and the critic; the former is the policy network, which takes the optimal action given a certain state; while the latter is the optimal state-action value function (Q-function), which evaluates the quality of the action taken by the policy/actor-network. This structure allows the policy network to update its parameters under the guidance of the critic network, while the critic network updates its parameters by minimizing a loss function based on Bellman's optimality equation. Both models are formed by two DNNs, the online and target networks. The target networks facilitate the training process by making it more stable and avoiding overfitting through the use of a replay buffer. TD3 uses three traits that improve its performance with respect to DDPG:

- Clipped Double Q-learning: the algorithm uses two Q-functions instead of one. Then uses the smallest of the two to form the targets in the Bellman error loss function.
- Delayed Policy Updates: the algorithm updates the actor-network less frequently than the Q-function.
- Noise regularization: the algorithm adds noise to the target action. This reduces the high variance on the target values when updating the critic.

For the sake of brevity, the pseudocodes, their implementation, and more details can be found in references [9], [14] and [15].

C. Andes Gym

The final step is to implement the framework in a place where the agent can be trained by interacting with the power grid. RL algorithms are trained in simulation platforms, this allows agents to interact with the environment without being subjected to physical constraints or catastrophic events, e.g. a wide area damping controller trained as an agent in a real power grid could provoke instabilities leading the system to a collapse. Thus, RL algorithms are not trained in real environments. In this work, $andes\_gym$ python library has

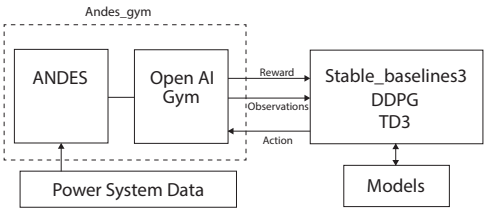

Fig. 3: Andes_gym architecture [4]

been used to train an agent as a centralized primary frequency controller for the grid.

Figure 3 shows the architecture of $andes\_gym$, which will be used to explain how the agent is trained. Three main Python libraries are needed: $andes$ which is a power system simulator, $stable\_baselines3$ which contains the RL algorithms, and $andes\_gym$ which is the API that interconnects the first two libraries —$andes\_gym$ is built on top of gym library from $OpenAI$—. Moreover, $stable\_baselines3$ can be substituted for any RL algorithm, including other libraries or self-implemented new algorithms.

$andes\_gym$ is used to set the environment, the RL algorithm, and its hyperparameters, as well as the condition of the training process. The environment includes the power system model, the selection of observation signals, the definition of the action signal, the reward function, and the parameters to be used in the nonlinear simulation by $andes$. Once this is set, $andes\_gym$ leads the nonlinear simulations performed by $andes$ at the same time that breaks it into action instances. In each action instance, $andes\_gym$ collects the observation from $andes$ and feeds them, including the reward, to the DRL algorithm in $stable\_baselines3$. Then the agent sends the action information to $andes\_gym$ such that the control signal can be modified in $andes$. Once this has been done, $andes\_gym$ enables the next action instance, replicating the same process. This process is performed during several episodes until reaching convergence.

The next section presents the results of implementing the DRL-PFC in a test system, wrapping up all the elements of this section.

IV. Case Study

The DRL-PFC is tested in a 14-bus, 5-machine system [6]. The SGs are described by a GENROU model, each one with an IEESGO turbine-governor model and an IEEE-type EXST1 exciter. Also, all loads have been converted to constant power model and all the IEESGO models have been altered to include the agent action $\mu_{c,t}$. Furthermore, the power imbalance disturbance is achieved by a load change event of 0.1 p.u., applied on bus 4 at 0.1 seconds. The weights in the reward function are set to $\beta_{1-4} = [300, 30000, 50, 1000]$. For practical purposes, the post-disturbance frequency is assumed to be known and is fixed at 59.68 Hz. Finally, considering that generator 3 is the closest one to the disturbance and by consequence the most sensitive to it, its frequency response is used for the performance analysis shown in the results.

Three analyses are done in this section: the first compares the behavior of the DDPG-based controller when different observations are used in the reward function, the second shows a sensitivity analysis of the DDPG-based controller while modifying the learning start hyperparameter, and the third one makes a comparison between DDPG and TD3 algorithms for the primary frequency controller.

A. Frequency and ROCOF as observations in the reward

The frequency behavior of the system is evaluated in the first few seconds after a disturbance. Three different cases are compared: without DRL-PFC, with DRL-PFC using frequency observations in the reward, and with DRL-PFC using frequency and ROCOF observations in the reward. The first case is the actual response of the system without the supplementary controller. The second case uses β_2=β_4=0, in Eq.12, while the third case uses β_i as defined in the introduction of this section.

Figure 4 shows the frequency response of the system after a disturbance. Note the larger nadir in the system when the supplementary controller is not considered, while in the other two cases, the magnitude of the nadir is reduced. This shows that the agent successfully improves the frequency response of the system by controlling the SGs participation in the primary frequency response.

In addition, Fig. 4 shows that the frequency nadir is better when only the frequency observation is included in the reward. Fig. 5 presents the absolute value of the ROCOF, in which the agent that considers both, frequency and ROCOF, succeeds in reducing the ROCOF magnitude during the entire simulation. When deciding which one has a better performance, it is important to consider that even if the nadir is greatly improved, subsequent swing in frequency can cause erroneous activation of protections due to large changes in the frequency and voltage drops. Accordingly, it might be better to have a moderate improvement in the nadir while reducing the ROCOF

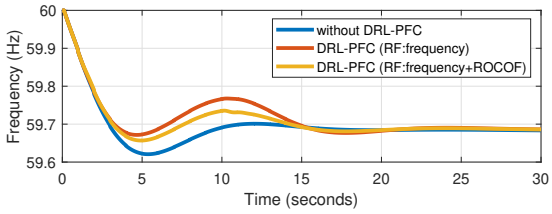

Fig. 4: Frequency with different control signals

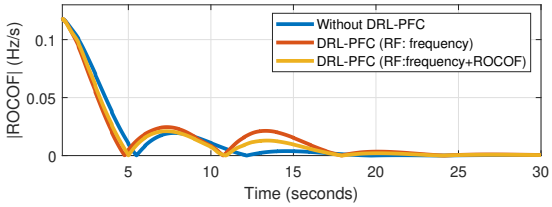

Fig. 5: Absolute ROCOF with different control signals

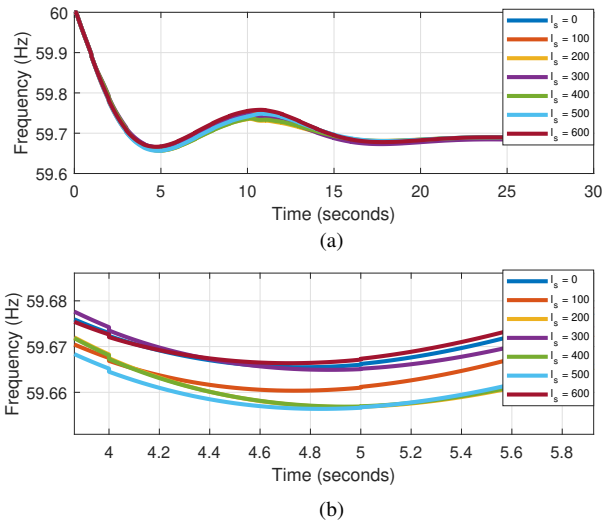

Fig. 6: Frequency with different l_s: a) full plot and b) partial enlargement

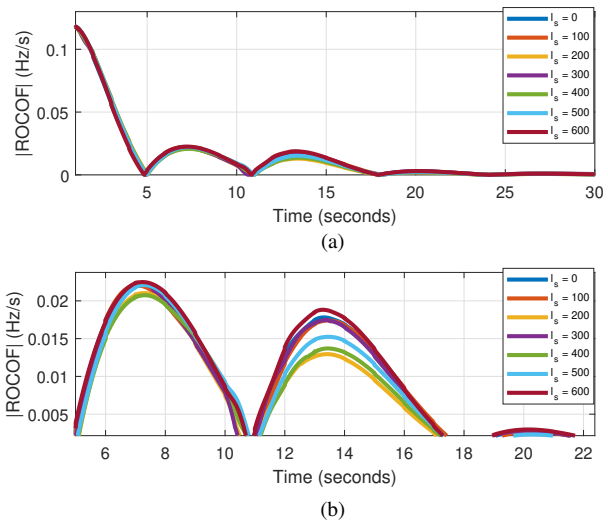

Fig. 7: Absolute ROCOF with different l_s: a) full plot and b) partial enlargement

at the same time. Therefore, the agent that includes both observations can be regarded as a trade-off between the performance of the frequency nadir and ROCOF. This demonstrates that having a better knowledge of the environment, through more observations, provides more intelligence to the agent allowing it to take better decisions.

B. *Effects of learning starts*

The parameter "learning starts" l_s, is used together with the replay buffer to avoid overfitting and convergence problems in the training process due to the strong correlation of sequential states. This parameter refers to how many experiences are necessary to take in the replay buffer before the model starts to learn. In this subsection, the agent that includes both frequency and ROCOF magnitudes as observations is evaluated under different values of l_s. The frequency and absolute ROCOF with different l_s are shown in Fig. 6 and Fig. 7, respectively.

As shown in Fig. 6 and Fig. 7, $l_s = 600$, provides the best performance in terms of frequency nadir, however, the best performance in terms of ROCOF is obtained with $l_s = 200$. It is necessary to find a good trade-off to improve the performance of both parameters. This is a typical issue with machine learning and deep reinforcement learning algorithms. Although in general, these algorithms are able to enhance any application, it is necessary to perform a trial-and-error process to come up with the best set of hyperparameters. Observing once again the results, one can arguably say that the best performance is reached by setting $l_s = 600$, however, as in the case of different reward functions, the overall best setting is $l_s = 200$, as it shows the best trade-off between frequency nadir and ROCOF. From the RL point of view and for continuous action spaces, a vast experience is necessary to avoid any erroneous direction in the initial stages of the learning process, i.e. if the model starts learning after the first step, it might be possible that this is a big outlier which would derail the initial training direction. To avoid this, the model samples from the replay buffer after a certain number of steps; this number is the value of l_s.

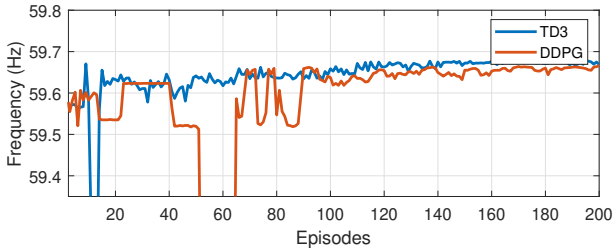

Fig. 8: DDPG vs TD3 frequency nadir for 200 episodes.

C. *DDPG vs TD3*

In this section, the DRL-PFC is compared using DDPG and TD3 algorithms. To make a fair comparison, both models are set with the same hyperparameters.

The first comparison consists in running the training process for 200 episodes of 30 action instances per episode. The idea is to compare which frequency nadir stabilizes faster and to which value. It is important to recall that a smaller frequency nadir represents an smaller drop in the frequency of the power grid, because of this, the agent will try to decrease the value of the nadir. Fig. 8 shows the results of this test. TD3 is able to reach an smaller frequency nadir with fewer training episodes in comparison to DDPG. Moreover, the results are more stable for TD3 than DDPG. As presented in TD3 documentation on [7], this could be due to a common failure on DDPG where the Q-function begins to dramatically overestimate Q-values, leading to policy breaking because it exploits the errors in the Q-function. This issue is addressed by TD3, by using the double clipping trick. The second comparison consists in observing the frequency behavior of the grid right after

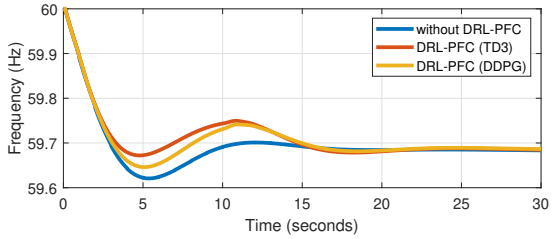

Fig. 9: Frequency with different control signals.

the disturbance. This can be seen in Fig.9. It is evident that the best response is obtained using the TD3 agent. This not only provides a better frequency nadir but also decreases the ROCOF, avoiding larger overshoots. The softer response between the three scenarios is provided by TD3 algorithm, which makes it a better solution for these types of continuous action problems. Next comparison is done by evaluating the performance of each algorithm with different learning starts values. Fig. 10.a shows the frequency nadir performance over 200 episodes by DDPG agent. When using l_s under 300 steps, the results randomly become unstable. Moreover, when observing the performance close to the 200 episodes, the frequency nadir decays and oscillates even for l_s values of 400 and 500 action instances. The most stable response is obtained when using 600 action instances. Contrary to this, as shown in Fig. 10.b, TD3 is more stable for any l_s value. These results are obvious since TD3 is a newer version of DDPG enhancing the capacity of actor-critic methods for solving continuous action problems. As a result, TD3 provides a better solution for the coordinated action of all SGs in the power grid primary frequency response.

V. Conclusion

This paper propose a supplementary frequency controller using DRL for primary frequency regulation. The regulation is done by implementing a supplementary signal in the governor's frequency reference. This modified reference is led by an actor-critic agent, which is trained to adjust the signal according to the states of the system. The results show a reduction in the frequency nadir and ROCOF magnitudes, improving the frequency response of the system after a disturbance. The paper highlights some of the challenges that DRL algorithms face in the training process. The trial-and-error characteristic can lead to suboptimal results that require expert intervention for fine-tuning. This is demonstrated with the modifications in the learning start hyperparameter in which a quality comparison is needed to select its best value. However, an appropriate selection of observation and reward functions could lead to better results. Finally, this paper demonstrates that the application of DRL in stability problems is an open and rising field of study. Future research could explore the generalization of the DRL-PFC to bigger systems and different types of disturbances that can lead to the appearance of new dynamics and the necessity of multi-agent deep reinforcement learning controllers.

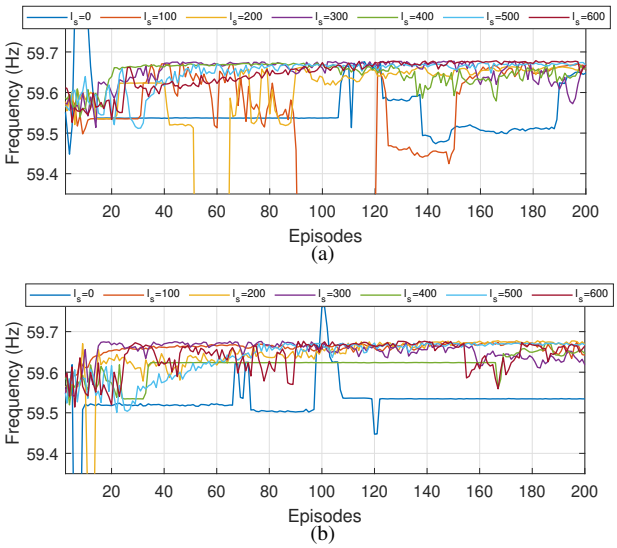

Fig. 10: Frequency nadir with different l_s. a) DDPG, b) TD3

References

[1] Sauer, Peter W., Mangalore A. Pai, and Joe H. Chow. Power system dynamics and stability: with synchrophasor measurement and power system toolbox. John Wiley and Sons, 2017.

[2] F. Milano, F. Dörfler, G. Hug, D. J. Hill and G. Verbič, "Foundations and Challenges of Low-Inertia Systems (Invited Paper)," 2018 Power Systems Computation Conference (PSCC), Dublin, Ireland, 2018,

[3] Chow, J. H., and Sanchez-Gasca, J. J. (2020). Power system modeling, computation, and control. John Wiley and Sons.

[4] Cui, H., and Zhang, Y. (2022). Andes_gym: A Versatile Environment for Deep Reinforcement Learning in Power Systems. arXiv preprint arXiv:2203.01292.

[5] Thacker, T., and Pulgar-Painemal, H. (2022, October). Improved primary frequency response through deep reinforcement learning. In 2022 North American Power Symposium (NAPS) (pp. 1-6). IEEE.

[6] H. Cui, F. Li and K. Tomsovic, "Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis," in IEEE Transactions on Power Systems, vol. 36, no. 2, pp. 1373-1384, March 2021,

[7] "OpenAI; Spinning up Documentation." Spinning up Documentation, spinningup.openai.com/en/latest/index.html. Accessed 11 Dec. 2022.

[8] Kou, G., Hadley, S., and Liu, Y. (2014). Dynamic model validation with governor deadband on the eastern interconnection. Oak Ridge Nat. Lab., Power and Energy Syst. Group, Oak Ridge, TN, USA,

[9] B. She, F. Li, H. Cui, J. Zhang and R. Bo, "Fusion of Microgrid Control With Model-Free Reinforcement Learning: Review and Vision," in IEEE Transactions on Smart Grid, vol. 14, no. 4, pp. 3232-3245, July 2023,

[10] Sutton, R. S., and Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[11] Zhu, J., Wu, F., and Zhao, J. (2021, December). An Overview of the Action Space for Deep Reinforcement Learning. In 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence.

[12] IEESGO model. Governor model: IEESGO and IEESGOD. (n.d.). https://www.powerworld.com/WebHelp/Content/TransientModel

[13] Hausknecht, M., and Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In 2015 aaai fall symposium series.

[14] Qiu, C., Hu, Y., Chen, Y., and Zeng, B. (2019). Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. IEEE Internet of Things Journal, 6(5), 8577-8588.

[15] Fujimoto, S., Hoof, H., and Meger, D. (2018, July). Addressing function approximation error in actor-critic methods. In International conference on machine learning (pp. 1587-1596). PMLR.