A Real-time Digital Simulator Accelerated Reinforcement Learning Training Environment for Power System Frequency Dynamics

Pooja Aslami[†], Tara Aryal, Niranjan Bhujel, Hossein Moradi Rekabdarkolaee, Kaiqun Fu, Zongjie Wang, and Timothy M. Hansen

***Abstract*—The growing integration of distributed energy resources (DERs), power electronic devices, and flexible loads in power systems is increasing the complexity and uncertainty in the modern power grid, bringing new challenges in its operation and control. However, with the recent advancement in artificial intelligence techniques, reinforcement learning (RL) has received increasing research interest. It has shown its potential to ensure reliable operation and improve the resilience of the power grid. Effective RL training requires significant interaction with an environment which can be difficult in the case of large-scale power system models due to long simulation duration. To address this issue, this paper proposes a co-simulation framework involving real-time digital environment simulation for accelerated training of RL agents. The proposed co-simulation framework is tested for providing fast frequency response (FFR) to the microgrid model and analyzed based on its training speed compared to another RL training approach based on an equivalent C-code environment model interface. Results show that the proposed co-simulation framework can speed up the RL training process by approximately 17 times while generating optimal RL agents.**

***Index Terms*—Reinforcement Learning, Power System, Opal-RT, Fast Frequency Response, Soft Actor-Critic .**

I. Introduction

The critical electricity infrastructure in recent years has started undergoing an energy transition, with the gradual decrease of fossil fuels and increasing penetration of renewable energy resources (RES) like solar energy, wind power, increasing integration of power electronic devices, and flexible loads such as electric vehicles and distributed energy storage system (ESS) [1], [2]. Although this transformation has made a significant contribution to reliable, safer, and cleaner energy, it also brings new challenges such as less predictable power generation, system complexities, and uncertainties.

In recent years, the deployment of advanced information and communication technologies within the power system, such as advanced metering infrastructures (AMIs), phasor measurement units (PMUs), and wide area monitoring systems (WAMS), has significantly enhanced access to a large volume of mutually correlated data in complex structures [3]. The information contained in this data is valuable and can be used to supplement the shortcomings of physical model-based methodologies for the proper operation and planning of the modern power grid. Using these data, machine learning (ML) can help to address the mentioned challenges in the modern power grid [4], [5]. ML can extract useful information from historical data, can deal with extremely uncertain system dynamics, and can generate adaptive models.

Among different ML-based methods, Reinforcement Learning (RL) is one of the most significant branches for power system controls and other decision making processes. RL is particularly effective at self-learning through interactive trial and error with dynamic environments. Due to its benefits, RL is now widely used in industries including robotic control, industrial manufacturing operation, and scheduling [6], [7]. By analyzing reward feedback from its experiences, RL learns and becomes more resilient against uncertain or highly unpredictable dynamics. This allows RL to create optimal policies without the need for the knowledge of the system dynamics. This makes RL a suitable approach, especially for control and optimization in power systems [8].

Much research has been conducted to showcase the implementation of RL in operational control, optimizing smart grids, electricity markets, and demand-side management [9]–[12]. To obtain optimal agents from RL training, it is necessary for RL agents to have significant interaction with an environment or at least with high-fidelity environment simulation [13]. However, the computational burden of simulation, especially high-fidelity simulation, grows significantly with the size and complexity of the power system environment. As a result, the simulation duration significantly increases impacting the training process of the RL agent.

One method to manage the complexity of a large-scale power system model is to implement a hierarchical framework to divide the major tasks into more manageable sub-

P. Aslami, T. Aryal, K. Fu and T. M. Hansen are with the Department of Electrical Engineering and Computer Science, and H. M. Rekabdarkolaee is with the Department of Math&Stats at South Dakota State University, Brookings, South Dakota, USA 57007 ([†]corresponding email: Pooja.Aslami@jacks.sdstate.edu).

N. Bhujel is with the Department of Electrical and Computer Engineering, University of Maine, Orono, Maine, 04469 USA.

Z. Wang is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, Connecticut, 06269 USA.

This work is supported by the South Dakota State University Richard McComish Fellowship in Energy Infrastructure and by the U.S. Department of Energy Office of Science, Office of Electricity Microgrid R&D Program, and Office of Energy Efficiency and Renewable Energy Solar Energy Technology Office under the EPSCoR grant number DE-SC0020281. This research made use of the Opal-RT real-time simulator purchased under NSF grant number MRI-1726964 and the "Roaring Thunder" cluster at South Dakota State University funded under National Science Foundation (NSF) grant number CNS-01726946.

tasks [14]. However, due to its design complexity, lack of dynamic adaptation capability, and lack of general hierarchical framework, studies of RL application in power systems with hierarchical framework are rare [15]. Another approach to manage high-fidelity simulation is to implement multi-agent deep RL (MADRL) based on centralized training and decentralized execution [16], [17]. However, when dealing with large-scale power systems, the current MADRL algorithms face issues with convergence, scalability, coordination, and training duration [15].

This study proposes a co-simulation framework with online training of RL agents to address the aforementioned issues and speed up the training process of RL agents for large power systems. The proposed co-simulation framework uses a real-time digital simulator (namely an OPAL-RT) to achieve high computational speed while performing high-fidelity simulation for RL training and evaluation. To evaluate the effectiveness of the proposed co-simulation framework, the computational time to conduct one episode of training is compared with the standard approach of RL training involving an equivalent C-code environment. The real-time simulation training environment is shown to significantly enhance the feasibility in training RL agents for complex power system operations.

The rest of the paper is organized as follows: section II introduces the proposed co-simulation framework with online training of the RL agent for the power system. Section III explains the implementation of the proposed co-simulation framework for fast frequency response (FFR) in microgrid. The simulation setup for the implementation of the proposed framework is explained in Section IV. Results and Conclusions are presented in Sections V and VI.

II. Proposed Co-simulation Framework with Real-Time Digital Environment

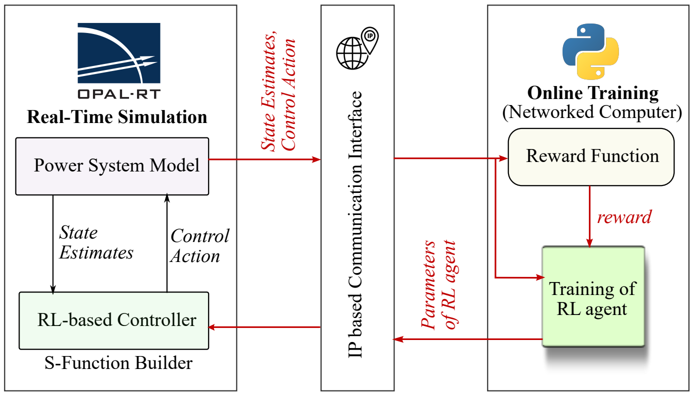


Fig. 1. Proposed co-simulation framework

This section outlines the architecture of the proposed framework for the online training of the RL agents for the power system dynamics including the simulation with modeling and control in real-time digital environment and communication interface. Fig. 1 displays the schematic of the proposed co-simulation framework that can accelerate the training process of RL agents for power system dynamics. The red arrow lines represent the flow of data involved in the training of the RL agent, while the black arrow lines represent the data flow in a real-time simulation environment.

The first block in the proposed framework contains a real-time simulation environment which is performed using a combination of OPAL-RT and RT-Lab, integrated with the Artemis toolbox for advanced real-time electromagnetic simulation. OPAL-RT is a real-time simulation platform fully integrated through RT-Lab with MATLAB/Simulink. The proposed framework is further improved by the accelerated real-time simulation provided by the integration of the Artemis toolbox. This is especially crucial when simulating scaled power grid models or micro-grid models to ensure reliable, accurate, and fixed-time step computations.

Here, the Simulink model is grouped into three subsystems master, slave (the term from the Opal-RT library, further referred to as 'secondary'), and scope subsystem (terminology derived from RT-lab environment). The master subsystem contains the power system model, which generates states and receives control signals. The SimPowerSystems (SPS) Toolbox from Simulink is used to model the power system model since the models built in SPS are compatible with OPAL-RT and can be executed in real time. The secondary subsystem contains a copy of the RL-based controller to ensure the flow of the control signal for the power system model in real-time. The scope subsystem contains scopes to monitor the status and control action during the training process. This partition enables the implementation of the RL-based controller and the power system model in the distinct cores, as well as the evaluation of the computing time of the control signals from the RL-based controller. This process helps to simulate the the power system model at a significantly faster time step to accurately capture all of the system dynamics which accelerates the training process of the RL agent. To import the RL-based controller into the Simulink model, the following steps are performed:

1) Write C code with actor-network prediction logic, and corresponding header files with necessary declarations for the actor-network functions and constants.
2) Use the S-function builder in MATLAB/Simulink to implement the written c code. The intel compiler is used to build the S-function builder inside the Simulink model.
3) Execute the model in MATLAB/Simulink.

In this block, the state estimates obtained from the power system model in the master subsystem are sent to the RL-based controller in the secondary subsystem. Then RL-based controller feedbacks the control action signal to the power system model.

The second block in the proposed framework contains online training of RL agents implemented in Python. The PyTorch package from Python is used to model and optimize the neural networks involved in the RL algorithm. RL is a machine learning technique that focuses on training agents that are capable of producing policies that result in maximum cumulative reward over time [18]. In RL, the decision-making problem is modeled as a Markov Decision Process (MDP). Its key concepts are a) State (s), b) Action (a), and c) Reward (r). Reward is the signal reflecting the desirability of an action for a given state. For the RL agent to generate an optimal policy, it is very important to define the reward function to properly align with the overall goals of the agent/controller. In this block, the state estimates and control action from the Simulink model are sent to be used as input for the designed reward function. Then the state estimates, control action, and the calculated reward are used for the training of the RL agent as shown in Figure 1.

The third block in the proposed framework contains a communication interface based on the Internet Protocol (IP), which routes and addresses the data packets so that they can travel across the computing devices in a network [19]. The most popular transport layer protocols operating over IP are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), which are both available in the I/O interface from RT-lab [20] and also can be implemented in Python via sockets [21]. For an RL agent training based on Python, the socket package from Python is used to build TCP/UDP sockets to receive state estimates and control action from the Simulink model and send the parameters of the RL agent to the S-function of RL-based controller in Simulink. For the Simulink model operating in Opal-RT, the RT-LAB Opinput and Opoutput blocks are used to receive parameters of the RL agent and send state estimates and control actions in the training environment of the RL agent. The details regarding the TCP/UDP setup in the I/O Interfaces in RT-LAB are given in [22].

Before compilation, the .h, .c, and .tlc files, which are the header and code files for the RL-based controller in Simulink model and S-function generated wrapper file, should be uploaded in the file properties of RT-LAB. After completing the configuration, the following steps are followed to perform the proposed co-simulation framework.

Step 1: load and execute the simulink model in RT-LAB
Step 2: wait for simulink model to stabilize to avoid intial transient.
Step 3: implement the python code with TCP/UDP socket to establish the connection with simulink model
Step 4: save optimized RL Agent neural networks

The UDP protocol-based timing diagram following the above steps to perform the proposed co-simulation framework is shown in Fig. 2.

III. Implementation of the proposed Co-simulation Framework for Fast Frequency Response in Microgrid Model

With increasing penetrations of inverter-based resources causing faster grid dynamics and system inertia reduction, the need for FFR is growing for power system frequency stability [23]. Among various FFR solutions, using trained RL control signals through an Energy Storage System (ESS) has been shown to provide effective FFR [24]. However, better learning of RL-based FFR agents requires a higher frequency of interaction with the environment. This can be difficult in the case of a power system or microgrid model because of their long simulation period that comes with their complexity, size, and the need for small time steps. Therefore, the proposed co-simulation framework is tested for proving FFR in the microgrid model as shown in Fig. 3

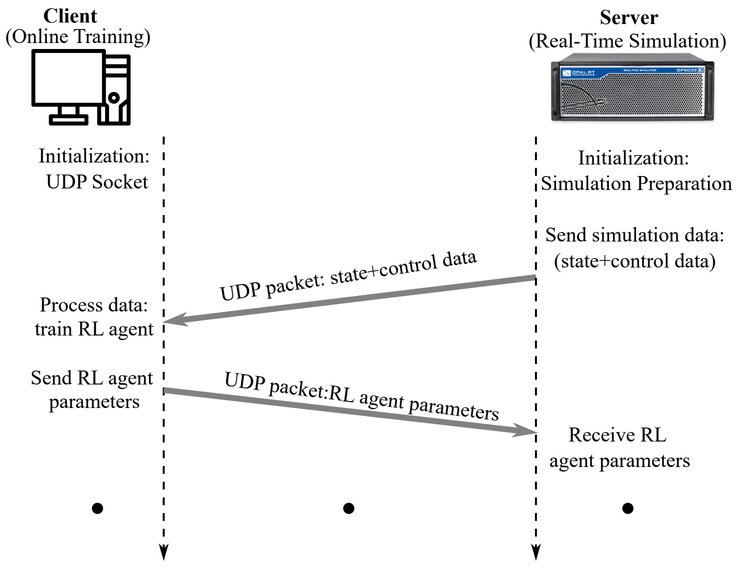


Fig. 2. Network timing diagram

In Fig. 3, the power system model considered is a modified microgrid model from Cordova, Alaska. There are two substations called ORCA and Humpback Creek (HBC) with voltage levels of 12.47 kV and 0.48 kV respectively in the microgrid model. The microgrid model also consists of an Energy Storage System (ESS), which is modeled by a DC source and an inverter. The model also comprises a phase-locked loop (PLL) that measures the change in frequency ($\Delta\omega$).

A recursive Bayesian filter called the Kalman filter (KF) uses measurements and estimates of states from the previous timestep to determine optimal current states [25]. While the process of trial and error is used to find the weights for noise-free systems, a previous work outlines a methodology for weight selection for noisy systems [26]. In this study, the measurement $\Delta\omega$ from the microgrid model is utilized to estimate the rotor angle $\Delta\delta$, change in frequency ($\Delta\omega$), rate of change of frequency ROCOF ($\Delta\dot{\omega}$), and disturbance in the system P_d using the KF. These state estimates are then used by the RL-based controller to generate a reference control signal P_{ref} for the ESS unit. The ESS unit then generates corresponding control signal ΔP_{inv}, which is controlled power from the inverter.

Among different RL algorithms, the FFR is implemented using an off-policy algorithm called Soft Actor-Critic (SAC), because it offers a faster convergence rate and higher sampling efficiency. In SAC, there are three major components. First is a replay buffer $\mathcal{D}$ to store the agent's experiences to allow off-policy training. Second is actor-critic architecture with deep neural networks (DNN) for policy π and state-

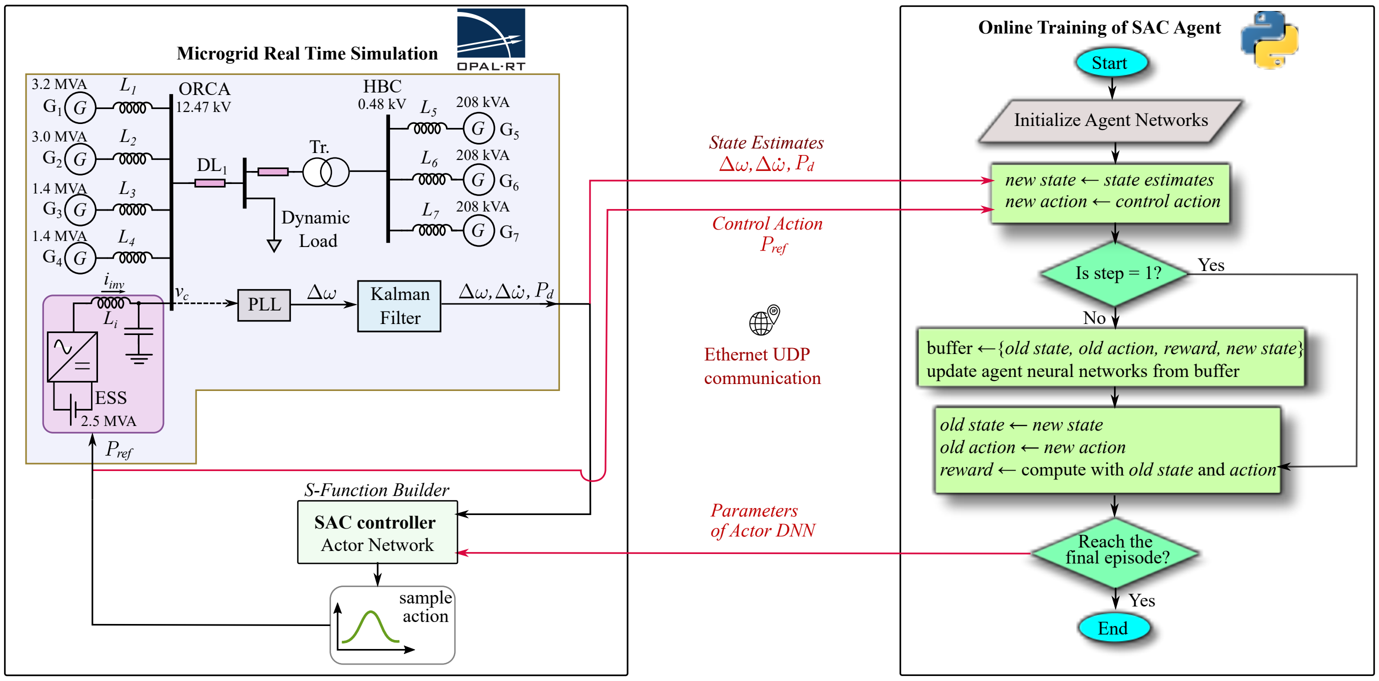


Fig. 3. Implementation of the proposed co-simulation framework for FFR in Cordova Benchmark

action value Q respectively. And third is an entropy $\mathcal{H}$ to promote exploration while preserving learning stability. The loss functions associated with the components Q, π, and $\mathcal{H}$ are given in [18]. In SAC, the objective is to determine the optimal stochastic policy π^* that maximizes its expected sum of rewards as well as entropy. To obtain this objective, the optimal parameters of the SAC components are learned with the SAC algorithm as proposed [24]. To align this objective with FFR, it is important to model MDP accordingly. So for the environment model of Cordova Benchmark, the MDP is modeled as follows:

1) State (s_t): $[\Delta\omega, \Delta\dot{\omega}, P_d\ ]$
2) Action (a_t): P_{ref}
3) Reward (r_t): For the given objective, the r_t is defined as

$$r = -\left[\mathcal{Q}_{\Delta\omega}(\Delta\omega_{obs} - \Delta\omega_{ref})^2 + \mathcal{Q}_{\Delta\dot{\omega}}(\Delta\dot{\omega}_{obs} - \Delta\dot{\omega}_{ref})^2 + \mathcal{R}P_{ref}^2\right],$$

where $[\mathcal{Q}_{\Delta\omega}, \mathcal{Q}_{\Delta\dot{\omega}}, \mathcal{R}]$ are the penalizing weights. The learning process of the SAC agent is further improved by normalizing the r_t.

In Fig. 3, the data transfer across two platforms takes place with the implementation of the UDP communication interface represented by the red arrow lines. Here, the state estimates, control action, and parameters of actor DNN are transferred.

IV. Simulation Setup

The Cordova benchmark model is operated with a step time of 0.01 ms while the Kalman Filter and SAC RL-based controller is operated with a step time of 20 ms. The difference in the step time is explained in [27]. To support this multi-rate configuration setup within the proposed co-simulation framework, the decimation factor in the RT-lab is set at 400, which is a sample acquiring factor for UDP communication with respect to the sample time of the micro-grid model. For KF, the values used for measurement noise covariance and process noise covariance were based on [24].

To implement SAC, two critic DNNs (Q_1 and Q_2), two target critic DNNs ($\overline{Q}_1$ and $\overline{Q}_2$), and one actor DNN (π) are utilized. The details regarding the use of four critic networks are explained in [18]. The dimensions of the layers in these DNNs corresponding to the modeled MDP in Section III are presented in Table I. The values for hyper-parameters of the SAC algorithm were obtained from [18]. The DNNs in the SAC-based FFR are optimized using Adam optimizer, available in the PyTorch package.

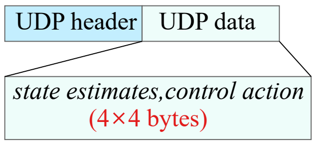
TABLE I
Summary of Network Architecture

Networks	Input Size	Hidden Size (1)	Hidden Size (2)	Output Size
Q_θ	4	256	256	1
π_ϕ	3	256	256	2

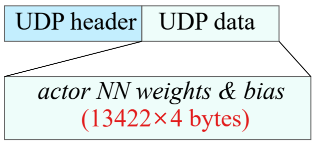
The design of the data packets for this study is shown in Fig. 4. To deal with the DNNs with a high number of layers and neurons, the weights and bias of the actor-network are transferred in chunks, within a limit imposed by the underlying IPv4 protocol (65,535 bytes for UDP datagram) [28].

To generate a robust SAC agent that can provide effective FFR, it is important to train the agent while subjecting the microgrid model to various load disturbance cases. The train-

Note: The code for real-time digital environment simulation with proposed co-simulation framework will be available as open-source at https://github.com/PAslami/Co-simulation-framework-for-RL-accelerated-training upon the acceptance of the paper.



(a)



(b)

Fig. 4. byte structure of the data packets (a) from server to client (b) from client to server

ing of the SAC agent was performed on Intel(R) Core(TM) i7-10700T CPU @ 2.00GHz with a 64-bit operating system. The training was started at the base load of 0.5 p.u. Once the system went to a steady state, the load was increased by applying a square signal with a period of 200 sec with step load amplitude varying between [-0.2, 0.2] p.u. The control action from the SAC agent was limited between [-0.5, 0.5] p.u., which is determined by the ESS unit. The episode duration was set to 100 sec [24].

To evaluate the performance of the proposed co-simulation framework in accelerating the training of RL agents, another training approach involving the Python-based RL controller interfaced with the C-code of the Simulink-based environment is also conducted [24]. In this approach, the C-code from a Simulink model is created from Simulink Coder which is then compiled into shared library using `gcc` compiler. Using the `ctypes` library, the built shared library is then loaded into Python.

Along with the speedup of the training process, the proposed co-simulation framework must support the generation of effective and optimized RL agents. To test this, the trained RL agent is tested with the micro-grid model under the step load change of 20%. For this, the simulation is started with the base load of 0.5 p.u. and at 170 sec load is increased to 0.7 p.u. The testing simulation run time was set to 325 sec.

V. Result and Analysis

To analyze the performance of the proposed co-simulation framework, the simulation was conducted based on the setup described in Section IV. In this evaluation, training duration per episode and the RL agent's capability to provide FFR to the microgrid model were used as key metrics.

TABLE II
Training Duration per Episode with different approaches

	proposed co-simulation framework	C-code based interface model
training duration per episode	209 sec	1.01 hrs

Table II shows the computational time it took for the two approaches to train the RL agent for the given microgrid model. It can be observed that the approach with the C-code based interface model takes a prohibitively long time to train one episode, which suggests that it will take an extremely long time to train RL agents for a higher number of iterations. This poses an increased risk of interruptions and conflicts to the training process and delays in the timely implementation of RL agents. Also with this approach, the training duration increases with increase in the complexity of the power system model used, making it very difficult to conduct proper RL training for larger power system model.

On the other hand, it can be observed that with the proposed co-simulation framework, it takes a relatively short (and computationally feasible) time to train one episode. While comparing, the proposed co-simulation framework outperforms the approach with the C-code based interface model with 17.4 times faster training duration. Because the simulation of the microgrid model takes place separately within the Opal-RT digital simulator, the environment simulation does not impact the computational time associated with RL training; the training duration remains constant even with the increased complexity of the power system model. As the Opal-RT has special hardware and software features for large-scale power system simulation, it becomes easier to handle complex power system models. Therefore, the training duration in the proposed co-simulation framework is expected to remain relatively constant, even with a larger power system model, leading to the ability to train RL agents across a variety of complex power system tasks. Moreover, it is also expected that the proposed co-simulation framework will be adaptable to the changing power system topology as the training process in this framework utilizes real-time data. Therefore, it can be concluded that the proposed co-simulation framework involving real-time digital environment simulation can accelerate the training process of RL agents.

To evaluate the proposed co-simulation framework in terms of effective training of RL agents, the training was continued and the corresponding actor DNN was saved for every 50 episodes. The optimized actor DNN was determined at the 1200th episode. Fig. 5 shows the plot of $\Delta\omega, \Delta\dot{\omega}, and \Delta P_{ref}$ for the step load change mentioned in Section IV. It can be observed that the SAC RL-based controller begins to provide control signals immediately after the load change event at 170 sec. While comparing the performance of the SAC RL-based controller with the system's response without the controller, it can be observed that the proposed SAC RL-based FFR can reduce the nadir of the frequency and ROCOF significantly. The controlled reference signal from SAC RL-based controller settled at around 270 sec. It can be concluded that the proposed co-simulation framework can accelerate the training process of RL agents while maintaining the effective training of RL agents.

VI. Conclusion

This research work developed a co-simulation framework to accelerate the training process of the RL agent for power system applications. The proposed co-simulation framework involved an IP-based interface between Opal-RT and Python. To test the effectiveness of the proposed co-simulation framework, it was implemented to train an SAC-based RL agent to provide FFR to the Cordova microgrid model. The co-simulation performance was analyzed based on training duration per episode and the ability of the trained RL agent to provide FFR. Compared to the standard training approach,

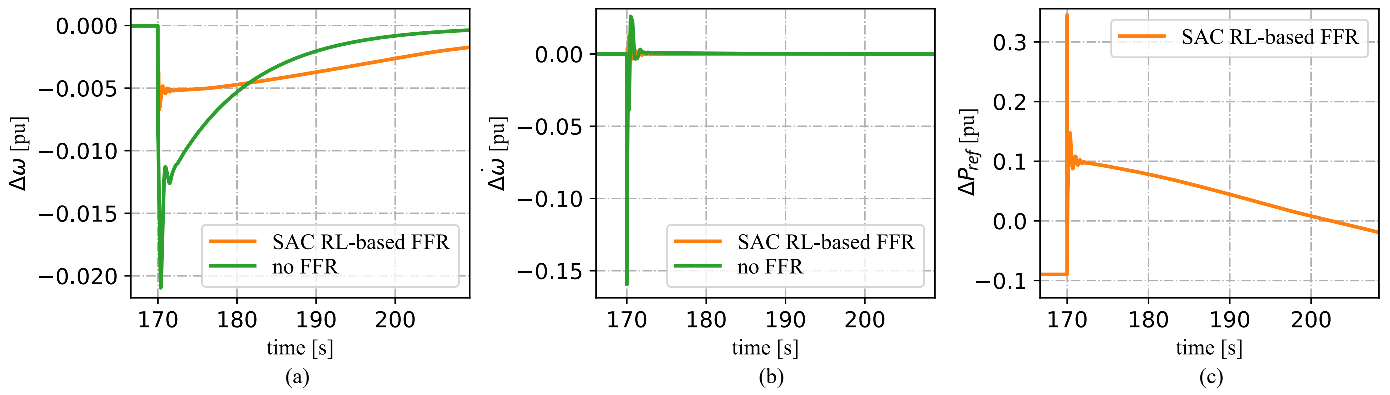


Fig. 5. Performance of the trained RL agent in terms of reduction in (a) change in frequency, (b) ROCOF, and (c) controlled reference signal from RL agent, for a step load change of 20% to Cordova Benchmark

the proposed co-simulation framework demonstrated better performance in terms of accelerating the training process of the RL agent. The results also showed that the trained SAC-based RL agent from the proposed co-simulation framework can effectively provide FFR to the microgrid model. In conclusion, the studies in this paper validate the feasibility of the proposed co-simulation framework to accelerate the training process of the RL agents for power system dynamics. Future studies will focus on further feasibility testing of this approach by applying it to a broader range of RL algorithms.

References

[1] K. S. Ratnam, K. Palanisamy, and G. Yang, "Future low-inertia power systems: Requirements, issues, and solutions-a review," *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109773, 2020.

[2] S. Peyghami, P. Palensky, and F. Blaabjerg, "An overview on the reliability of modern power electronic based power systems," *IEEE Open Journal of Power Electronics*, vol. 1, pp. 34–50, 2020.

[3] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[4] J. Xie, I. Alvarez-Fernandez, and W. Sun, "A review of machine learning applications in power system resilience," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, 2020, pp. 1–5.

[5] O. A. Alimi, K. Ouahada, and A. M. Abu-Mahfouz, "A review of machine learning approaches to power system security and stability," *IEEE Access*, vol. 8, pp. 113 512–113 531, 2020.

[6] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.

[7] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[8] R. Rocchetta, L. Bellani, M. Compare, E. Zio, and E. Patelli, "A reinforcement learning framework for optimal operation and maintenance of power grids," *Applied Energy*, vol. 241, pp. 291–301, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261919304222

[9] A. O. Erick and K. A. Folly, "Reinforcement learning approaches to power management in grid-tied microgrids: A review," in *2020 Clemson University Power Systems Conference (PSC)*. IEEE, 2020, pp. 1–6.

[10] K. Zhou, K. Zhou, and S. Yang, "Reinforcement learning-based scheduling strategy for energy storage in microgrid," *Journal of Energy Storage*, vol. 51, p. 104379, 2022.

[11] L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang, and X. Guan, "A review of deep reinforcement learning for smart building energy management," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 046–12 063, 2021.

[12] E. O. Arwa and K. A. Folly, "Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review," *Ieee Access*, vol. 8, pp. 208 992–209 007, 2020.

[13] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement learning–overview of recent progress and implications for process control," *Computers & Chemical Engineering*, vol. 127, pp. 282–294, 2019.

[14] Y. Y. Haimes and D. Li, "Hierarchical multiobjective analysis for large-scale systems: Review and current status," *Automatica*, vol. 24, no. 1, pp. 53–69, 1988.

[15] D. Cao, W. Hu, J. Zhao, G. Zhang, B. Zhang, Z. Liu, Z. Chen, and F. Blaabjerg, "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, pp. 1029–1042, 2020.

[16] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[17] S. Wang, J. Duan, D. Shi, C. Xu, H. Li, R. Diao, and Z. Wang, "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4644–4654, 2020.

[18] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[19] A. Shiranzaei and R. Z. Khan, "Internet protocol versions—a review," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2015, pp. 397–401.

[20] Apr 2023. [Online]. Available: https://www.opal-rt.com/software-communication-protocols/tcp-udp/#:~:text=With%20the%20proper%20user%20configuration,port%20will%20begin%20data%20transmission.

[21] G. M. Hassan, N. M. Hussien, and Y. M. Mohialden, "Python tcp/ip libraries: A review," *International Journal Papier Advance and Scientific Review*, vol. 4, no. 2, pp. 10–15, 2023.

[22] Feb 2024. [Online]. Available: https://www.opal-rt.com/support-knowledge-base/?article=AA-01860

[23] N. I.-B. R. P. Task *et al.*, "Fast frequency response concepts and bulk power system reliability needs," *NERC*, p. 1, 2020.

[24] P. Aslami, T. Aryal, N. Bhujel, A. Rai, H. M. Rekabdarkolaee, and T. M. Hansen, "A soft actor-critic approach for power system fast frequency response," in *2023 North American Power Symposium (NAPS)*, 2023, pp. 1–6.

[25] D. Simon, *Optimal State Estimation: Kalman, H infinity and nonlinear approaches*. Wiley-Interscience, 2006.

[26] B. Poudel, P. Aslami, T. Aryal, N. Bhujel, A. Rai, M. Rauniyar, H. M. Rekabdarkolaee, U. Tamrakar, T. M. Hansen, and R. Tonkoski, "Comparative analysis of state and parameter estimation techniques for power system frequency dynamics," in *2022 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*. IEEE, 2022, pp. 754–761.

[27] U. Tamrakar, F. B. dos Reis, A. Luna, D. Shrestha, R. Fourney, and R. Tonkoski, "Virtual inertia emulation using commercial off-the-shelf inverters," in *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2018, pp. 1111–1116.

[28] A. Steed and M. F. Oliveira, "Chapter 3 - overview of the internet," in *Networked Graphics*, A. Steed and M. F. Oliveira, Eds. Boston: Morgan Kaufmann, 2010, pp. 71–123. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123744234000033