SAND2023-07664C

Soft Actor-Critic Based Voltage Support for Microgrid Using Energy Storage Systems

Niranjan Bhujel[†1], Astha Rai[1], Ujjwol Tamrakar[3], Yifeng Zhu[1], Timothy M. Hansen[2], Donald Hummels[1], and Reinaldo Tonkoski[1]
[1]*University of Maine* - Orono, Maine, USA
[2]*South Dakota State University* - Brookings, South Dakota, USA
[3]*Sandia National Laboratories* - Albuquerque, New Mexico, USA
Email: [†]niranjan.bhujel@maine.edu

***Abstract*—A microgrid is characterized by a high R/X ratio, making the voltage more sensitive to active power changes unlike in bulk power systems where voltage is mostly regulated by reactive power. Because of its sensitivity to active power, control approach should incorporate active power as well. Thus, the voltage control approach for microgrids is very different from conventional power systems. The energy costs associated with these power are different. Furthermore, because of diverse generation sources and different components such as distributed energy resources, energy storage systems, etc, model-based control approaches might not perform very well. This paper proposes a reinforcement learning-based voltage support framework for a microgrid where an agent learns control policy by interacting with the microgrid without requiring a mathematical model of the system. A MATLAB/Simulink simulation study on a test system from Cordova, Alaska shows that there is a large reduction in voltage deviation (about 2.5-4.5 times). This reduction in voltage deviation can improve the power quality of the microgrid.**

***Index Terms*—Microgrids, reinforcement learning, soft actor-critic, voltage dynamics, voltage support**

I. Introduction

The rapid integration of renewable energy sources (RES) into the power system is revolutionizing the energy landscape, offering significant advantages such as reduced greenhouse gas emissions and improved energy sustainability. However, this integration also poses voltage stability challenges, including fluctuations, harmonics, flickering, imbalanced loads, and power oscillation. Microgrids face increased vulnerability to voltage instability due to the presence of diverse generation sources, reactive power limitations, and load dynamics [1]. In microgrids, voltage instability occurs in a major event like a sudden change in power demand or output from RES or when a generator stops working. Even small changes in demand can cause voltage instability, especially in systems already operating near their limits. In conventional power systems, the R/X ratio of the transmission/distribution lines is lower, and the voltage is sensitive to reactive power only; hence, reactive power management techniques ensure a reliable power supply by reducing fluctuations. In contrast, microgrids typically operate at low to medium voltage ranges and have a high R/X ratio [2], resulting in high voltage sensitivity to both active and reactive power. Depending on the ratio, the voltage could be more sensitive to active power. Thus, the traditional voltage control approach used in conventional power systems is not applicable to microgrids. The control approach should consider both active and reactive power as control signals. This paper analyzes the potential of utilizing energy storage systems (ESSs), which are systems of energy storage (usually batteries) with an inverter that can absorb or inject the desired amount of active/reactive power, to provide voltage support in microgrids.

Various voltage control methods have been proposed for microgrids. Droop-based control [3], [4] is commonly used but can lead to oscillatory behavior and sub-optimal performance. H-infinity (H_∞) [5] based control offers robust stability but requires a precise system model and does not handle non-linear constraints well. Linear quadratic regulators [6] and model predictive control (MPC) [7] require system models and face challenges in accurately modeling converter-based resources, especially in microgrids with diverse generation mix. Additionally, MPC requires complete state and parameter information and incurs higher computational costs, which may not be suitable for short control time steps involved with microgrid voltage dynamics. Different model-free approaches like Extremum-seeking (ES) control [8] utilize output measurements to optimize power injections from distributed energy resources (DERs) for voltage regulation. However, a significant drawback of existing ES algorithms is the insufficient treatment of constraints. Many ES methods either overlook constraints for simplicity or indirectly penalize constraint violations in the objective function. A reinforcement learning (RL) based approach can provide an ideal framework in microgrids for

This work was supported by the US Department of Energy, Office of Electricity, Energy Storage Program, and National Science Foundation (NSF-2218063). The authors acknowledge the supercomputing resources made available by the Advanced Computing Group of the University of Maine System (https://acg.maine.edu) for conducting the research in this paper.

voltage support. Data-oriented approaches learn directly from historical data without assuming a specific model, enhancing their robustness in noisy, complex environments [9]. However, prior voltage control works using RL techniques such as deep deterministic policy gradient (DDPG) focused on steady-state voltage profiles, neglecting system dynamics [9]. In [10], a time delayed deep deterministic policy gradient (TD3) approach was proposed; however, it is sensitive to hyperparameters. To address this, [11] introduced soft actor-critic (SAC) as an improved alternative to DDPG and TD3. SAC has number of advantages over other RL algorithm such as entropy regularization ,which encourages policy to have higher entropy thus maintaining exploration and preventing premature convergence to sub-optimal solutions, and better learning speed.

In this paper, we propose a SAC-based voltage support technique for a microgrid. The paper is organized as follows: Section II presents the method used for voltage support. The simulation setup is presented in Section III, with results and findings summarized in Section IV. Section V concludes the paper.

II. Soft Actor-Critic (SAC)-based Voltage Support

A. Basics of Reinforcement Learning

Reinforcement learning is the training of a model to make a sequence of decisions. The agent interacts with the environment to learn the control policy. The policy maps states of the environment to the control action. A reward is a scalar value that represents how the agent is performing. A reward is collected every time the agent interacts with the environment. Q-value, represented mathematically as $Q^{\pi}(s_t, a_t)$ represents the predicted value of an accumulated reward an agent would collect over the future when the agent takes action a_t at the current timestep and policy π thereafter. The agent learns to approximate the Q-value, which is often represented by a neural network. In RL, the objective is to maximize accumulated reward. Thus, the optimal control action is:

$$a_t^* = \underset{a_t}{\text{argmax}}\ Q^{\pi}(s_t, a_t). \tag{1}$$

The above optimization problem might not be feasible to solve for continuous action space since Q function is generally nonlinear. Thus, the actor-critic method is used, which has a separate neural network for mapping states to optimal action (called actor-network) that is trained alongside the Q-network (called critic-network).

B. Soft Actor-Critic Method

SAC is one of the types of actor-critic methods where the objective is to maximize $Q-$value along with entropy. In SAC, one NN for the actor and two NNs for a critic are used. Two critics are used since critics are found to overestimate the Q-value or expected reward. Thus, using two critics and the smallest predicted Q-value of two networks for the target Q-value mitigates the problem of overestimation. In SAC, the policy is assumed to be stochastic, i.e., the actor-network gives parameters of probability distribution (generally Gaussian is used). Let $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$ represent action and states of the environment at discrete time t. Let $\pi_{\boldsymbol{\phi}}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ represent the policy with $\boldsymbol{\phi}$ being trainable parameters of the actor-network. The policy gives the probability density of taking action $\boldsymbol{a}_t$ given the system is in state $\boldsymbol{s}_t$ i.e., $\boldsymbol{a}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{s}_t), \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{s}_t))$ where $\mathcal{N}$ represents normal distribution. The sampled action is passed to `tanh` function and scaling is performed to obtain the final action. Let $Q_{\boldsymbol{\theta}_1}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and $Q_{\boldsymbol{\theta}_2}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ represents two Q-networks with $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ being trainable parameters of critic networks. Further, two target networks (one for each critic network) are used. The trainable parameters of the target networks are represented by $\bar{\boldsymbol{\theta}}_1$ and $\bar{\boldsymbol{\theta}}_2$. Let $\mathcal{D}$ represent the replay buffer that stores the experience $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ tuple. Let us define $Q_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) := \min(Q_{\boldsymbol{\theta}_1}(\boldsymbol{s}_t, \boldsymbol{a}_t), Q_{\boldsymbol{\theta}_2}(\boldsymbol{s}_t, \boldsymbol{a}_t))$ and $V(\boldsymbol{s}_t)$ as:

$$V_{\boldsymbol{\theta}}(\boldsymbol{s}) := \mathbb{E}_{\boldsymbol{a}_t \sim \pi}\left[Q_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) - \alpha \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right] \tag{2}$$

where α represents the entropy coefficient and action $\boldsymbol{a}_t$ is sampled from the current policy. The critic loss is defined as:

$$J_Q(\boldsymbol{\theta}_i) = \underset{\substack{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}) \\ \sim \mathcal{D}}}{\mathbb{E}}\left[\Big(Q_{\boldsymbol{\theta}_i}(\boldsymbol{s}_t, \boldsymbol{a}_t) - (r_t + \gamma V_{\bar{\boldsymbol{\theta}}}(\boldsymbol{s}_{t+1}))\Big)^2\right]$$
$$\forall\ i \in \{1, 2\} \tag{3}$$

with γ being a discount factor and i being the index for critic networks. Now, the objective of the actor-network is to find the action that maximizes the Q-value. In SAC, entropy should also be maximized. Thus, the actor loss can be written as:

$$J_{\pi}(\boldsymbol{\phi}) = \underset{\boldsymbol{s}_t \sim \mathcal{D}, \boldsymbol{a}_t \sim \pi_{\phi}}{\mathbb{E}}\left[\alpha \log(\pi_{\boldsymbol{\phi}}(\boldsymbol{a}_t|\boldsymbol{s}_t)) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]. \tag{4}$$

Here, the action is sampled from a current policy with a re-parameterization trick, which is a technique to back-propagate the gradient through sampling. For example, sampling from a normal distribution with mean μ and standard deviation σ is equivalent to $\mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. In the latter case, it is possible to differentiate sampled output with respect to μ and σ. Additionally, the entropy coefficient is also adjusted so that the entropy of policy is equal to the specified target entropy. The loss for entropy tuning can be written as:

$$J(\alpha) = \underset{s_t \sim \mathcal{D}, \boldsymbol{a}_t \sim \pi}{\mathbb{E}}\left[-\log(\alpha)\left(\log \pi(\boldsymbol{a}_t, \boldsymbol{s}_t) + \bar{\mathcal{H}}\right)\right] \tag{5}$$

where $\bar{\mathcal{H}}$ is the target entropy. Finally, the soft update of the target network is done as:

$$\bar{\boldsymbol{\theta}}_i \leftarrow \tau\boldsymbol{\theta}_i + (1-\tau)\bar{\boldsymbol{\theta}}_i \quad \forall\ i \in \{1, 2\} \tag{6}$$

where τ is the update factor for a soft update of the target network. The SAC algorithm is summarized in Algorithm 1.

C. Voltage Support using SAC Method

To provide voltage support, an ESS is connected to one of the buses in a microgrid. Instantaneous values of local quantities (voltage v_c, currents i_g as shown in Fig. 1) are measured and converted to dq-frame using voltage v_c as reference. Let $v_{c,dq}$ and $i_{g,dq}$ represent the voltage and current in dq-frame respectively (with $v_{c,dq}$ being a vector of d and q components of v_c. Same for other variables). The dq-transformed variables

Algorithm 1: SAC Algorithm

```
Input: $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\phi}$
       $\bar{\boldsymbol{\theta}}_1 \leftarrow \boldsymbol{\theta_1}, \bar{\boldsymbol{\theta}}_2 \leftarrow \boldsymbol{\theta_2}$
       $\mathcal{D} \leftarrow \emptyset$
for each iteration do
    $\boldsymbol{a}_t \sim \pi_{\boldsymbol{\phi}}(\boldsymbol{s}_t)$
    Apply action $\boldsymbol{a}_t$ to the environment at state $\boldsymbol{s}_t$ to get $\boldsymbol{s}_{t+1}$
    $\mathcal{D} \leftarrow \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})\}$
    for each gradient step do
        $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \lambda_Q \nabla_{\boldsymbol{\theta}_i} J_Q(\boldsymbol{\theta}_i)$ for $i \in \{1, 2\}$
        $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \lambda_\pi \nabla_{\boldsymbol{\phi}} J_\pi(\boldsymbol{\phi})$
        $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J_\alpha(\alpha)$
        $\bar{\boldsymbol{\theta}}_i \leftarrow \tau \boldsymbol{\theta}_i + (1-\tau)\bar{\boldsymbol{\theta}}_i$ for $i \in \{1, 2\}$
    end
end
```

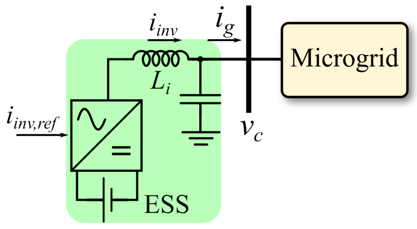

Fig. 1: Schematic showing ESS connection to a microgrid.

(total of 6 variables) should be used as a state. The d-component of v_c, i.e., v_{cd}, represents the instantaneous amplitude of the voltage. Our objective is to keep this quantity as close to some reference value (v_{ref}) as possible. In other words, we want to minimize $(v_{cd} - v_{ref})^2$ or maximize $-(v_{cd} - v_{ref})^2$. Similarly, we want to achieve this goal with the minimum amount of inverter current to reduce the energy cost as well as the size of ESS required. Thus, we want to minimize i_{invd}^2 and i_{invq}^2. Hence, we use the following reward function:

$$r_t = -\left((v_{\text{cd}} - v_{\text{ref}})^2 + R_1 i_{\text{invd}}^2 + R_2 i_{\text{invq}}^2\right). \tag{7}$$

The value of the reward increases when v_{cd} approaches v_{ref}. The values R_1 and R_2 set the relative priority of minimizing the d and q components of inverter currents. Furthermore, its value relative to 1 sets the relative priority of minimizing voltage deviation and ESS utilization. The value of v_{ref} is set externally (e.g., 1 p.u.), thus it is necessary to train the SAC-agent for different values of v_{ref}. Hence, v_{ref} is also passed to the SAC-agent along with other observations, making the dimension of observations to be 7. Additionally, the above reward should be normalized to ease the training. Normalization requires mean and standard deviation, which are unknown before interacting with the environment. Thus, running mean and standard deviation are used. They are calculated using the algorithm outlined in [12].

D. Recurrent Neural Network-based SAC

For the voltage support, the measurements from other buses are not available making the voltage support problem partially observable. Hence, the state s_t that is observed from the environment does not contain full information. Thus, it is necessary to incorporate past data to have more information about the environment. To work efficiently with time-series data, a recurrent neural network (RNN) can be used. In this work, long short-term memory (LSTM) networks have been used. Instead of passing just s_t at any time t, a sequence of the most recent s_t of specified length $\mathcal{L}$ is passed. The LSTM processes this data to give a compressed low-dimensional representation of states, which is passed to a fully-connected layer.

During inference, RNN states from previous timesteps and the value of s_t at the current timestep can be used to compute action. This allows us to avoid processing data of sequence length $\mathcal{L}$, and use just the s_t from current timesteps. This reduces the size of data to be processed at any given timestep, thereby making the inference computationally efficient.

III. Simulation Setup

A. System Description

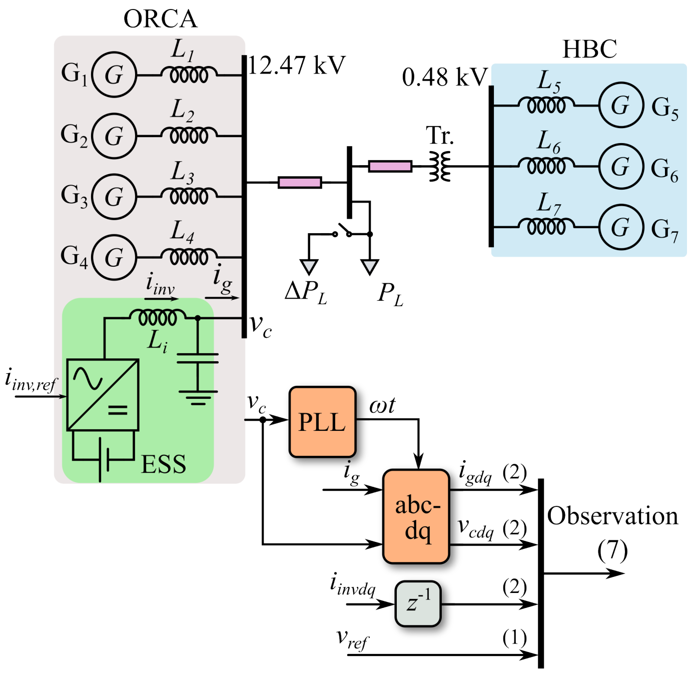

Fig. 2: Modified microgrid benchmark from Cordova, Alaska.

A simulation study was carried out to validate the approach. The setup is illustrated in Fig. 2 and was modeled in MATLAB/Simulink. The microgrid benchmark considered is modified from Cordova, Alaska where two substations ORCA and HBC are considered. An ESS with power capacity of 30% of the total microgrid size is connected to the ORCA substation. In this study, ideal DC source has been used thus making ESS storage capacity infinite. A standard phase-locked loop from Simulink was used to extract the instantaneous angle of v_c to perform dq-transformations. Gaussian noise of zero mean and standard deviation of $(10^{-2}, 10^{-2}, 10^{-2}, 10^{-2})$ pu for $(i_{gd}, i_{gq}, v_{cd}, v_{cq})$ is added to the available measurements [13]. The step-time of the agent is taken to be 100μs based on the approximate time constant of the system. The SAC-agent was implemented using the `keras` library, which is based on

Python. To make the Simulink model interact with the Python environment, the C-code for the Simulink model is generated using Simulink Coder. To make the API simple for Python, interface codes were written. The generated and interface codes are compiled to a shared library using the `GCC` compiler, which is then imported to Python using the `ctypes` library. For training, a base load of random value (denoted by P_L) from the range $[0.2, 0.8]$ pu is connected with random load change from the range $[-0.2, 0.2]$ pu. The random value of v_ref from a range [0.88, 1.1] is used based on the continuous operating range provided by the IEEE 1547-2018 standard. The sampling of these random values is done at the start of each episode. Furthermore, $i_\mathrm{inv,dq}$ is used as a control action and all variables are converted to per-unit before passing them to the agent.

To check computational feasibility, the equivalent C-code for the trained actor network was written. The linear algebra required for implementing the network was implemented using *Basic Linear Algebra Subroutines for Embedded Optimization* (`BLASFEO`) library [14]. The compilation was done using `GCC` compiler with level 3 optimizations.

TABLE I: Summary of parameters used to train the agent.

Parameters	Value	Parameters	Value
Buffer capacity (T)	10^6	γ	0.99
Batch size (M)	256	τ	0.005
Target entropy	-2.0	Initial entropy coefficient	0.1
Optimizer	Adam	Learning rate	3×10^{-4}
$\mathcal{L}$	50	R_1, R_2	0.05, 0.005
Number of parameters (actor-network)	5714	Number of parameters (critic-network)	5842

B. *SAC Training*

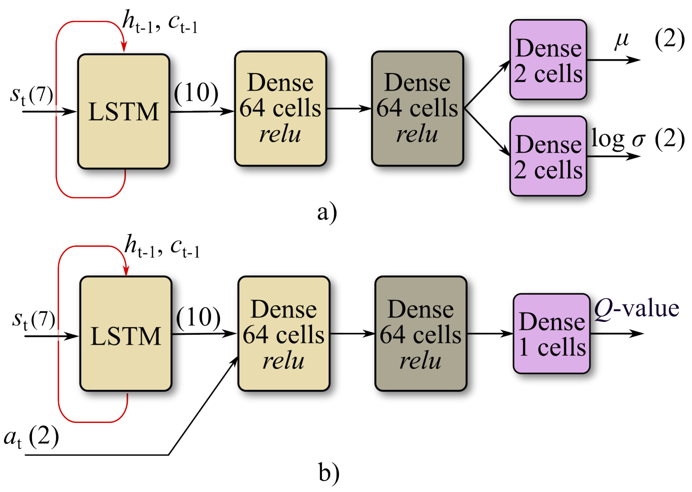

Fig. 3: Structure of neural networks: (a) actor-network (b) critic-network.

For both actor and critic networks, the LSTM layer with an output dimension of 10 is used. The next two layers used are the dense with an output dimension of 64 and an activation function of *relu*. For the actor-network, two output layers are used each with the dimension of 2 (for d and q components of inverter current) and a linear activation function. The first output gives a mean of the Gaussian distribution and the second output gives the logarithm of the standard deviation for the policy. For the critic network, one output layer of dimension 1 (Q-value is scalar) and a linear activation function is used. The structure of the neural network is shown in Fig. 3.

At every agent timestep, a control action is sampled from π_ϕ, which is applied to the environment. The transition i.e., (s_t, a_t, r_t, s_{t+1}) is stored in the buffer $\mathcal{D}$. Furthermore, 256 (batch size) transitions are randomly sampled, and one step update of the neural networks is performed. On the next agent timestep, this process is repeated.

IV. Results and Analysis

A. *Training Metrics*

The variation of training metrics as training proceeds is given in Fig. 4(a-c). The training was performed multiple times (starting with randomly initialized network parameters) to check variance in training metrics (represented by shaded region). Fig. 4(a) shows the variation of mean episodic reward. Initially, the agent takes random action and hence, the reward is very low. As training proceeds, the reward increases. Fig. 4(b) shows the standard deviation of the policy for i_{invd} and i_{invq}. The initial value for both was set to 1.0. As the training proceeds, their values decrease. The σ for q-component is higher than d-component because voltage is more sensitive to i_{invd} and hence, the agent can easily learn the effect of i_{invd}. Fig. 4(c) shows the entropy of stochastic policy. Th entropy coefficient loss (equation (5)) sets the value of α such that actual entropy is close to the target value.

B. *Performance of Voltage Support*

To evaluate the performance of voltage support, the base load is set to 0.3 p.u. and the system is subjected to a load change of 0.10 p.u. at 1.5 s. The performance of the trained agent is shown in Figs. 4(d-f) where the variation of voltage and current when no control (i.e., $i_{inv,ref}$ is set to 0 with no change in other components of ESS) is used is compared against the case when the trained agent is used. Fig. 4(d) shows that when the trained agent is used, the oscillation in current i_{gd} is reduced. Fig. 4(e) shows that the oscillation in voltage v_{cd} is also reduced. From Fig. 4(f), we can also see that a large magnitude of the q-component of inverter current is utilized as compared to the d-component. It is because we set the coefficient of the d-component higher. Thus, the agent tries to provide voltage support with a smaller magnitude of i_{invd}.

Further, the variation of maximum voltage deviation when load change is varied is shown in Fig. 5(a). For any given load change, the value of maximum voltage deviation is smaller when the proposed approach is used than when no ESS is used. This shows that the proposed approach can provide voltage support for all the values of load change that the agent has been trained on. The trained agent was able to reduce the voltage deviation by 2.5 to 4.5 times depending on the value of load change. Fig. 5(b) shows the maximum value of power provided by ESS to provide voltage support for different values of load change. A larger load change requires larger ESS power.

C. *Computational Time*

The equivalent C-code for the trained actor network was compiled and run on a Windows 10 machine with an Intel® Core (TM) i9-11900 @ 2.50 GHz. The average and maximum computational time was found to be 1.72 μs and 9 μs, respectively. This time is very small compared to the agent

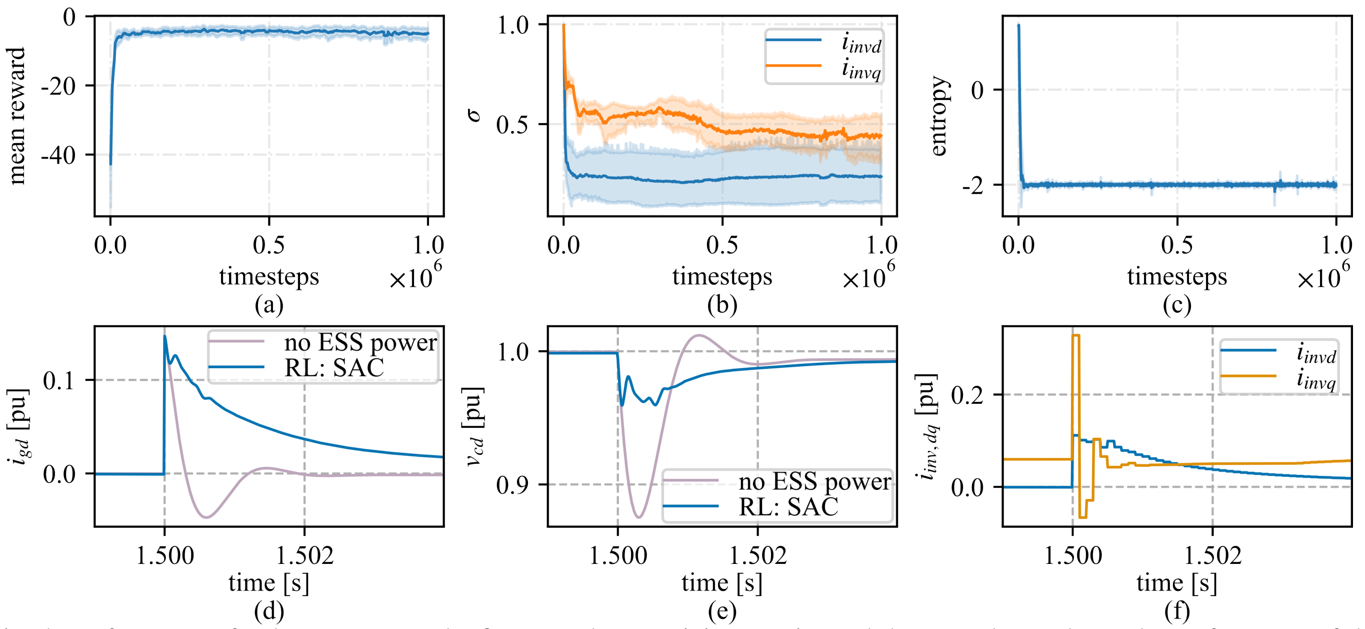

Fig. 4: Performance of voltage support. The first row shows training metrics and the second row shows the performance of the trained agent for load change of 0.1 pu.

step-time (100 μs). This shows that the proposed approach is computationally feasible to operate in real-time.

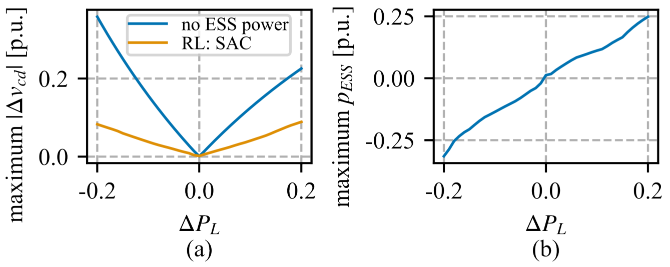

Fig. 5: Variation of (a) maximum voltage deviation and (b) maximum ESS power, with the value of load change.

V. Conclusions

In this paper, we used a SAC method-based approach for dynamic voltage support of microgrids. The trained agent was able to reduce the voltage deviation by about 2.5 to 4.5 times. Furthermore, the oscillation in instantaneous voltage amplitude was also reduced. This helps to mitigate power quality issues and improve the reliability of the grid. The average computational time was found to be 1.72 μs showing real-time applicability.

VI. Acknowledgments

The authors would like to thank Dr. Hyungjin Choi from Sandia National Laboratories for the technical review of this paper.

References

[1] Y. Al Jabri, N. Hosseinzadeh, R. Al Abri, and A. Al Hinai, "Voltage stability assessment of a microgrid," in *2015 IEEE 8th GCC Conference & Exhibition*, Feb 2015, pp. 1–6.

[2] M. Farrokhabadi, C. A. Cañizares, J. W. Simpson-Porco, E. Nasr, L. Fan, P. A. Mendoza-Araya, R. Tonkoski, U. Tamrakar, N. Hatziargyriou, D. Lagos *et al.*, "Microgrid stability definitions, analysis, and examples," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 13–29, 2019.

[3] J. C. Vasquez, R. A. Mastromauro, J. M. Guerrero, and M. Liserre, "Voltage support provided by a droop-controlled multifunctional inverter," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 11, pp. 4510–4519, 2009.

[4] J. W. Simpson-Porco, F. Dörfler, and F. Bullo, "Voltage stabilization in microgrids via quadratic droop control," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1239–1253, 2017.

[5] J. Mongkoltanatas, D. Riu, and X. LePivert, "H infinity controller design for primary frequency control of energy storage in islanding microgrid," in *2013 15th European Conference on Power Electronics and Applications (EPE)*, 2013, pp. 1–11.

[6] T. Vandoorn, B. Renders, L. Degroote, B. Meersman, and L. Vandevelde, "Voltage control in islanded microgrids by means of a linear-quadratic regulator," in *Young Researchers Symposium, Proceedings*. IEEE Benelux Chapter, 2010, p. 5.

[7] N. Bhujel, A. Rai, T. M. Hansen, R. Tonkoski, and U. Tamrakar, "A model predictive approach for voltage support in microgrids using energy storage systems," in *2021 IEEE Power & Energy Society General Meeting (PESGM)*, 2021, pp. 1–5.

[8] D. B. Arnold, M. Negrete-Pincetic, M. D. Sankur, D. M. Auslander, and D. S. Callaway, "Model-free optimal control of var resources in distribution systems: An extremum seeking approach," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3583–3593, 2016.

[9] J.-F. Toubeau, B. Bakhshideh Zad, M. Hupez, Z. De Greve, and F. Vallee, "Deep reinforcement learning-based voltage control to deal with model uncertainties in distribution networks," *Energies*, vol. 13, no. 15, p. 3928, 2020.

[10] D. Cao, J. Zhao, W. Hu, F. Ding, Q. Huang, and Z. Chen, "Distributed voltage regulation of active distribution system based on enhanced multi-agent deep reinforcement learning," 2020. [Online]. Available: https://arxiv.org/abs/2006.00546

[11] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018. [Online]. Available: https://arxiv.org/abs/1812.05905

[12] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," in *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, H. Caussinus, P. Ettinger, and R. Tomassone, Eds. Heidelberg: Physica-Verlag HD, 1982, pp. 30–41.

[13] N. Bhujel, T. M. Hansen, R. Tonkoski, U. Tamrakar, and R. H. Byrne, "Optimization-based estimation of microgrid equivalent parameters for voltage and frequency dynamics," in *2021 IEEE Madrid PowerTech*, 2021, pp. 1–6.

[14] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO," *ACM Transactions on Mathematical Software*, vol. 44, no. 4, pp. 1–30, jul 2018. [Online]. Available: https://doi.org/10.1145%2F3210754