Training A Deep Reinforcement Learning Agent for Microgrid Control using PSCAD Environment

Arash Farokhi Soofi
Electrical & Computer Engineering
University of California, San Diego
San Diego, USA
afarokhi@ucsd.edu

Reza Bayani
Electrical & Computer Engineering
University of California, San Diego
San Diego, USA
rbayani@ucsd.edu

Mehrdad Yazdanibiouki
Electrical & Computer Engineering
San Diego State University
San Diego, USA
myazdanibiouki6352@sdsu.edu

Saeed D. Manshadi
Electrical & Computer Engineering
San Diego State University
San Diego, USA
smanshadi@sdsu.edu

***Abstract*—The accessibility of real-time operational data along with breakthroughs in processing power have promoted the use of Machine Learning (ML) applications in current power systems. Prediction of device failures, meteorological data, system outages, and demand are among the applications of ML in the electricity grid. In this paper, a Reinforcement Learning (RL) method is utilized to design an efficient energy management system for grid-tied Energy Storage Systems (ESS). We implement a Deep Q-Learning (DQL) approach using Artificial Neural Networks (ANN) to design a microgrid controller system simulated in the PSCAD environment. The proposed on-grid controller coordinates the main grid, aggregated loads, renewable generations, and Advanced Energy Storage (AES). To reduce the cost of operating AESs, the designed controller takes the hourly energy market price into account in addition to physical system characteristics.**

***Index Terms*—Microgrid energy management system, Distributed energy resources, Reinforcement learning, Deep Q-learning, Artificial neural network**

I. Introduction

Distributed Energy Resources (DERs) have been experiencing a rapidly growing presence in the power systems due to government policies, technological advancements, and environmental concerns to meet the increasing electricity demand. Residential rooftop photovoltaics (PV) and residential Energy Storage Systems (ESSs) have increased by 51% and 18%, respectively, since 2011 [1]. Microgrids are small-scale electricity networks that can be operated independent of the main grid and are composed of DERs, ESSs, and consumers. In addition to providing electricity to remote places outside the reach of the conventional grid, these structures are introduced to improve the reliability and resilience of modern power systems [2]. Even though the integration of DERs improves grid operations, it also presents operators with a number of challenges, such as variations in the grid's voltage and frequency [3], [4]. The need for Microgrid Energy Management Systems (MEMS), which maximize the flow of energy by making the most of the available renewable resources, has increased due to the growing DER penetration.

DERs are one of the most efficient demand responsive assets of the grid. In [5], authors designed a MEMS for both residential and commercial DERs to perform peak shaving and load smoothing using a low complexity fuzzy logic control. An adaptive optimal model predictive control is introduced in [6] to establish balance between generation and demand in low-inertia power systems. Historical data is utilized in [5] to forecast and optimize the power exchange between DERs and the grid. An energy management solution for a multi-resource microgrid using fuzzy logic while including uncertainty by using an ANN to forecast demand and DER output is proposed in [7]. Many researchers have used fuzzy logic, interval programming, and stochastic programming models in the design of MGEMSs. Despite the many benefits provided by these approaches, several factors contribute to reducing their accuracy and effectiveness. For instance, lack of requirement for particular set of inputs compromises the solution accuracy of fuzzy logic [8]. Also, most of these approaches implement simplifying assumptions to reduce problem complexity to relieve the computation burden.

The smart grid infrastructure development has produced a wealth of historical operational datasets, which help applying Machine Learning (ML) algorithms to power system problems. An ML model for efficient event detection based on datasets of field-recorded PMUs is presented in [9]. The authors in [10] have demonstrated the capability of ANN in forecasting solar insolation. A weather prediction module in the energy management system by use of a feed-forward ANN is presented in [11]. Authors in [12] utilized a supervised machine learning algorithm in designing a power flow management platform for microgrid resources given the voltage, load, and weather data. The study in [13] proposes an unsupervised learning algorithm to predict next-hour energy demand and determine a discrete set of actions for AES to meet the predicted load. As it is suggested by the authors [13], the unsupervised approach lacked accuracy compared to the results obtained by a supervised ANN model. To increase

the accuracy of these methods, Reinforcement Learning (RL) models are proposed to develop rules (policy) using dynamic data obtained through interactions with the existing system. RL algorithms are a great tool to tackle uncertain network parameters and their effectiveness has been demonstrated in applications such as electric vehicle management [14] and electricity markets [15]. An RL approach to design a multi microgrid energy management where the electric systems states are estimated using multi-layer ANN is proposed in [16].

In this paper, a Deep Q-Network (DQN) agent is utilized to operate DERs under specific modes of operation which are voltage regulation, minimizing operation cost, and meeting the load demand. We use a dynamic modeling simulation software (PSCAD) to simulate smart grid features by providing measured data points to the agent for internal analysis. The DQN agent controls the AES while mimicking other characteristics of the electric system grid, including PV generation, weather forecasting, energy market price, power losses, and time. This approach helps to better understand realistic results compared to previous projects, where mathematical models of the electric system or supervised learning techniques were used to represent the behavior of the grid.

II. Problem Formulation

Unlike supervised and unsupervised ML methods, in RL, the agent policies are learned through interactions with the environment. At each time step (index t), specific characteristics of the environment are represented by the *state* (denoted by s_t). Every time the agent takes a specific *action* (denoted by a_t), a certain *reward* (denoted by r_t) results, and the environment's state is updated. The optimal solution to the problem is determined based on Bellman's equation stated in (1). Here, Q denotes the value of taking action a_t in state s_t. Actions are selected based on the *policy*, which in the DQN context, is the ANN which approximates the Q-value.

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{1}$$

A. Reward Definition

In this part, we first introduce the desired operational constraints which are included in the problem and then, three different reward functions which are used to train our model are described. Equation (2a) enforces the State-of-Charge (SOC) of the AES to stay between the nominal minimum and maximum values. The AES real power dispatch (denoted by P) is enforced within its nameplates by (2b). The real power set-points are considered discrete steps to improve the DQN agent performance by minimizing the action's array dimension. The primary objective of the DQN agent is to dispatch the AES so that the bus voltage (denoted by V) does not violate the normal bounds given in (2c). The voltage rewards are designed in a way that this constraint is satisfied.

$$SOC_t^{AES-Min} \leq SOC_t^{AES} \leq SOC_t^{AES-Max} \tag{2a}$$

$$P_k^{AES-Min} \leq P_k^{AES} \leq P_k^{AES-Max} \tag{2b}$$

$$V_k^{Bus-Min} \leq V_k \leq V_k^{Bus-Max} \tag{2c}$$

1) Voltage Regulation: The DQN's objective is to maximize the reward gained from the environment based on defined reward functions. In the Discrete Voltage Reward Function (DVRF), the agent is penalized when the voltage falls outside the 5% and 10% thresholds as resented in (3a). However, in the Continuous Voltage Reward Function (CVRF) the agent is penalized based on the difference between the voltage of bus i (V_i) and Nominal voltage (V_N) as presented in (3b).

$$DVRF = \begin{cases} -5 & If \quad : |V_i - V_N| > 5\% \\ -10 & If \quad : |V_i - V_N| > 10\% \end{cases} \tag{3a}$$

$$CVRF = \begin{cases} \frac{V_i - V_N}{V_N} & If \quad : V_i \neq V_N \\ 0 & If \quad : V_i = V_N \end{cases} \tag{3b}$$

2) Minimizing Operation Cost: The next objective of the DQN agent is to dispatch power to minimize the cost of charging the battery. Given historical hourly time-of-use data [17], the AES should become charged when the energy market price falls below the average market price during a particular day, and the energy storage must discharge when the hourly price of the energy market is above the average market price during a particular day. This behavior is applied by defining a reward function in (4). Here, C_t denotes the time-of-use price at time t where $\bar{C}_d$ denotes the daily average energy market price.

$$Market\_Reward = \begin{cases} C_t - \bar{C}_d > 0 & Discharge \\ C_t - \bar{C}_d = 0 & Idle \\ C_t - \bar{C}_d < 0 & Charge \end{cases} \tag{4}$$

3) Meeting the Demand: Finally, the demand response (DR) is defined as a reward function in (5), where DR reward takes priority over the market reward. This step requires finding the net generation and price difference defined respectively in (5a) and (5b) before calculating the conditional DR reward given in (5c).

$$Net - Generation = \sum_{k \epsilon m} P_k^{PV} + \sum_{k \epsilon m} P_k^{AES} - P_k^{Load} \tag{5a}$$

$$Price - difference = C_t - \bar{C}_d \tag{5b}$$

$$DR\_R = \begin{cases} if\ Net - Gen. > 0: & \begin{cases} if\ Price - df \geq 0: & Dis \\ if\ Price - df < 0: & Ch. \end{cases} \\ if\ Net - Gen. \leq 0: & Dis. \end{cases} \tag{5c}$$

III. Solution Methodology

Algorithm 1 describes the architecture of the implemented algorithm. There are two nested *for* loops that iterate through the episode counts (up to maximum episode i_{max}), where each episode is iterated for t_{max} time steps. At the beginning of each episode, the environment is reset. In the very first episode, the system is initialized with the nominal values. In the next

episodes, the continuous values such as SOC, bus voltages, and commanded set-points are inherited from the last time-step of the previous episode. Resetting the environment with nominal values periodically helped eliminate the consecutive state dependencies.

At each time-step, an action is selected based on policy π. We select the ϵ-greedy policy for our training, where actions are selected randomly with a probability of ϵ. Otherwise (with a probability of $1-\epsilon$), the action which yields the best Q-value is chosen. Then, the selected action is applied to the environment which brings about reward r_t and updates the environment's state s_{t+1}. At this point, the experience tuple is added to the replay buffer. If the number of stored experience tuples in the replay buffer is more than the determined buffer size, a batch of experience tuples is selected randomly. Then, the Q-value (y_t) is determined based on line 8, which is used to calculated the loss function defined in line 9 to update ANN weights.

Algorithm 1: Deep Q-Learning Using Experience Replay

Initiate: Q-value approximator ANN with random weights

1 **for** $i = 1 : i_{max}$ **do**
2 Reset the environment
3 **for** $t = 1 : t_{max}$ **do**
4 $a_t \leftarrow \pi(s_t)$
5 $s_{t+1}, r_t \leftarrow env(s_t, a_t)$
6 Store the experience tuple (s_t, a_t, r_t, s_{t+1}) in memory $\mathcal{D}$
7 select a sample batch from $\mathcal{D}$
8 $y_t \leftarrow r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$
9 Update ANN weights with $Loss = (Q(s_{t+1}, a_{t+1}) - y_t)^2$
10 **end**
11 **end**

In our simulation, the ANN Q-approximator represents tge agent, where policies are determined by the value of each action. The environment includes the PSCAD dynamic simulation, reward function, and the stored data. The complexity of the environment development is primarily due to the power system dynamic modeling being used as the environment. Fig. 1 illustrates the primary components for defining the DQN framework.

IV. Simulation Setup

PSCAD is a dynamic modeling software and has a robust parallel computation capability, which enables running the simulations close to real-time by designing proper pipelines. In addition, PSCAD provides a strong scripting functionality which was used for the automation process in this project. Article [18] describes different methodologies to simulate the time domain instantaneous responses using Electromagnetic Transients including DC (EMTDC). EMTDC is designed to

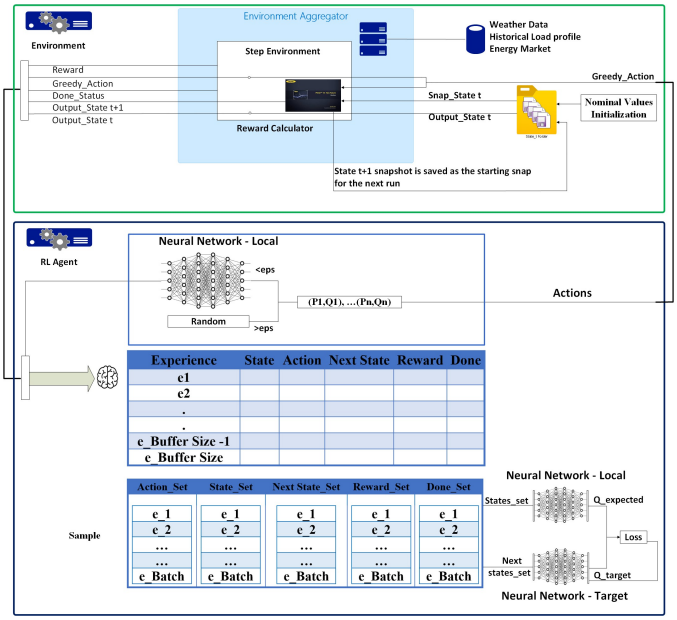

Fig. 1. Detailed Main Loop

capture the transient responses in an electric system and PSCAD is the graphical user interface to work on EMTDC.

The elements of the experience tuple are defined as follows. States are 2-D numerical arrays that are formatted using Python's Pandas data frames. The columns of the data frame include the system measurements, and the rows represent measurements in different time steps throughout a specific period. We take multiple measures after applying a particular action was to capture system transients, command delays, and other none ideal criteria in real-world operations. State includes the measurements from the PSCAD model and other external inputs at time t, which is a Numpy array as follows: $\{t,\ V^{BUS}_{Phase_{A,B,C}},\ C_t,\ \bar{C}_d,\ P^{Load},\ P^{PV},\ P^{AES},\ SOC^{AES}\}$. The index of the taken action at time t which is equal to [0,1,2] which respectively represents [Charge, Idle, Discharge].

The PSCAD automation file contains the python scripts responsible for synchronizing the PSCAD simulations with the DQN agent. At each time step, the DQN agent suggests an action based on the ϵ-greedy policy. The PSCAD automation file contentiously listens to the DQN agent, forms the pipeline between the DQN agent, and applies the suggested actions to the PSCAD software. In addition to creating this pipeline, the PSCAD automation file is essential for recording the measurements from PSCAD that generate the environment states.

V. Results and Discussions

In this section, the designed DQN agent is demonstrated in different cases. The base case creates the realistic condition by assuming historical values, and normal trends for loads and renewable generation. The episode is defined as one entire day (twenty-four hours) and the time steps in each episode are one hour. The structure of neural network affects the loss function of DQN. Utilizing the same structure for neural network as the fixed rate definition increases the convergence

rate of the learning algorithm [19]. In Fig. 2, it is shown that utilizing neural network with 2 hidden layers leads to less loss overshoot and earlier converge compared to utilizing neural network with 3 hidden layers.

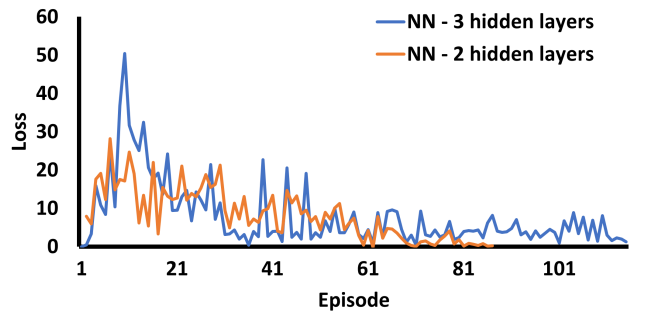

Fig. 2. Convergence rate of different neural network structures

Other than number of hidden layers, the activation function of the NN affects the convergence rate of the algorithm. Table I shows how utilizing ReLU function can increase the convergence rate. It is demonstrated that the best choice is to utilized ReLU function in a NN with 2 hidden layers.

TABLE I
NEURAL NETWORK STRUCTURE COMPARISON

Structure #	Activation Function	Layers	Convergence
1	Sigmoid	3	115
2	**ReLU**	**2**	**80**
3	ReLU	3	100

A. Voltage Reward Case

Based on the power quality requirements, the voltage magnitude of the buses should remain within 95% and 105% of the voltage nominal value.

Fig. 3 shows the calculated loss function assuming the agent is only considering the voltage stability which is presented in (3). Since the baseline scenario was simplified, the agent was expected to explore the environment and minimize the calculated loss between Q_expected and Q_target. Consequently, the problem reached acceptable loss function in approximately 80 episodes.

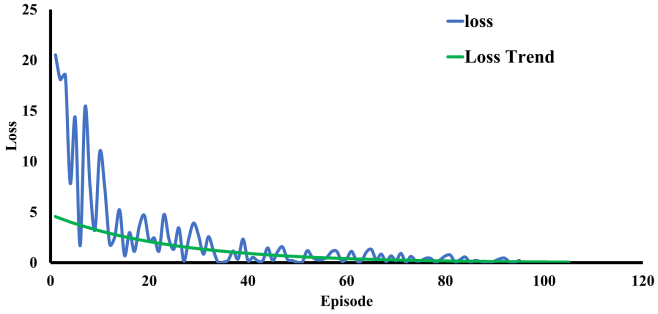

Fig. 3. Convergence of the proposed DQN agent with voltage reward function

B. Market Price Reward Case

In this scenario, the reward function was targeted to optimize the Battery Energy Storage System (BESS) dispatch based on hourly market price compared to the average market price of the same day, obtained from historical data as presented in (4). The main objective is to charge the AES (i.e. BESS) with the lowest cost to achieve certain SOC threshold at the end of each episode. Fig. 4 depicts a single-day operation of the AES based on dispatched set-points received from the DQN agent. The SOC, market hourly price, PV generation, load, and AES actions are shown in Fig. 4. The agent considers the global objective while deciding the best action for each hour. As demonstrated, the AES is dispatched to charge when the hourly market price falls below the average market price, and ASE's SOC is achieved and maintained above 60% after t=21 (9 PM). From time steps 1 to 5, the hourly energy market price is below the average price; thus, the agent should send either charge or idle set points to the AES. If the AES SOC is below 50%, the agent is expected to only send the charge set point.

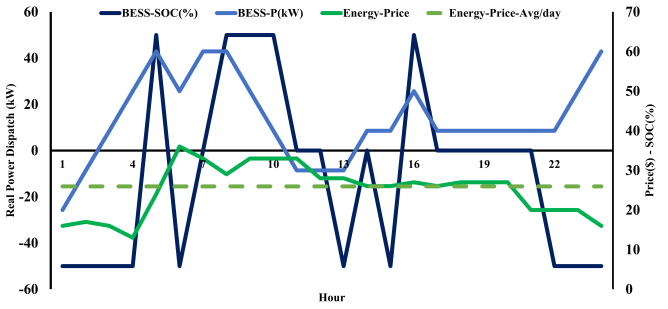

Fig. 4. Power dispatch of the BESS procured by the agent based on market reward function

C. Demand Response Reward Case

In this scenario, the agent was designed to respond to the demand fluctuations as a demand-responsive asset. The designed DQN agent must consider the market price and the excess PV generation to charge the battery. The demand-response reward function only considers the net generation and the net demand to evaluate the proper times to charge the AES. Therefore, the objective is to charge the AES when there is excess generation and discharge the AES when demand exceeds the generation. Although this reward function meets the Demand Response objectives, it eliminates the rest of the constraints. The cumulative reward function was defined to address how to optimize AES dispatch based on the system constraints, voltage reward, market price reward, and the cumulative reward shown in (5c). Fig. 5 represents the loss function considering the cumulative reward. A linear regression model was used to demonstrate the decreasing trend of the loss function upon training the model with more sample data. Comparing Fig. 5 to the results of the base model presented in Fig. 3, we can see how the increase in model

complexity requires more sample data to train the defined DQN agent based on the defined objective functions.

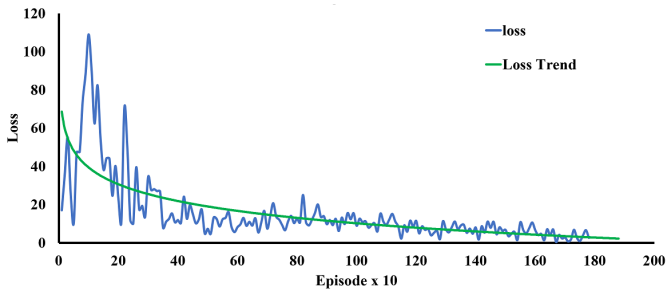

Fig. 5. Cumulative reward function: demand response, market price, voltage

VI. Conclusion

This article addressed the application of reinforcement learning in designing a microgrid controller for a grid following operation modes. The primary purpose of this article was to improve the existing procedures for designing microgrid energy management solutions. The previous papers' approaches were mainly based on optimization techniques such as fuzzy logic and linear programming or conventional supervised and unsupervised machine learning algorithms. The designed procedure aimed to optimize the energy flow between the AES and the primary grid. The Deep Q-learning algorithm helped the system to dispatch the optimized set points using the feedback from the environment.

References

[1] C. L. Kwan, "Influence of local environmental, social, economic and political variables on the spatial distribution of residential solar pv arrays across the united states," *Energy Policy*, vol. 47, pp. 332–344, 2012.

[2] R. Bayani, A. F. Soofi, and S. D. Manshadi, "Coordinated scheduling of electric vehicles within zero carbon emission hybrid ac/dc microgrids," in *2021 IEEE Transportation Electrification Conference & Expo (ITEC)*. IEEE, 2021, pp. 1–6.

[3] A. F. Soofi, R. Bayani, and S. D. Manshadi, "Analyzing power quality implications of high level charging rates of electric vehicle within distribution networks," in *2021 IEEE Transportation Electrification Conference & Expo (ITEC)*. IEEE, 2021, pp. 684–689.

[4] A. Farokhi Soofi, R. Bayani, and S. D. Manshadi, "Investigating the impact of electric vehicles on the voltage profile of distribution networks," *arXiv e-prints*, pp. arXiv–2012, 2020.

[5] D. Arcos-Aviles, J. Pascual, F. Guinjoan, L. Marroyo, P. Sanchis, and M. P. Marietta, "Low complexity energy management strategy for grid profile smoothing of a residential grid-connected microgrid using generation and demand forecasting," *Applied energy*, vol. 205, pp. 69–84, 2017.

[6] A. Abazari, M. M. Soleymani, M. Babaei, M. Ghafouri, H. Monsef, and M. T. Beheshti, "High penetrated renewable energy sources-based aompc for microgrid's frequency regulation during weather changes, time-varying parameters and generation unit collapse," *IET Generation, Transmission & Distribution*, vol. 14, no. 22, pp. 5164–5182, 2020.

[7] A. Chaouachi, R. M. Kamel, R. Andoulsi, and K. Nagasaka, "Multiobjective intelligent energy management for a microgrid," *IEEE transactions on Industrial Electronics*, vol. 60, no. 4, pp. 1688–1699, 2012.

[8] M. F. Zia, E. Elbouchikhi, and M. Benbouzid, "Microgrids energy management systems: A critical review on methods, solutions, and prospects," *Applied energy*, vol. 222, pp. 1033–1055, 2018.

[9] T. Dokic, R. Baembitov, A. A. Hai, Z. Cheng, Y. Hu, M. Kezunovic, and Z. Obradovic, "Machine learning using a simple feature for detecting multiple types of events from pmu data," in *2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*. IEEE, 2022, pp. 1–6.

[10] R. Nematirad and A. Pahwa, "Solar radiation forecasting using artificial neural networks considering feature selection," in *2022 IEEE Kansas Power and Energy Conference (KPEC)*. IEEE, 2022, pp. 1–4.

[11] J. Faraji, A. Abazari, M. Babaei, S. Muyeen, and M. Benbouzid, "Day-ahead optimization of prosumer considering battery depreciation and weather prediction for renewable energy sources," *Applied Sciences*, vol. 10, no. 8, p. 2774, 2020.

[12] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 9, pp. 1395–1408, 2010.

[13] H. Musbah, H. H. Aly, and T. A. Little, "Energy management of hybrid energy system sources based on machine learning classification algorithms," *Electric Power Systems Research*, vol. 199, p. 107436, 2021.

[14] R. Bayani, S. D. Manshadi, G. Liu, Y. Wang, and R. Dai, "Autonomous charging of electric vehicle fleets to enhance renewable generation dispatchability," *CSEE Journal of Power and Energy Systems*, 2021.

[15] M. Jabbari Zideh and S. S. Mohtavipour, "Two-sided tacit collusion: Another step towards the role of demand-side," *Energies*, vol. 10, no. 12, p. 2045, 2017.

[16] Y. Du and F. Li, "Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1066–1076, 2019.

[17] X. Yan, D. Wright, S. Kumar, G. Lee, and Y. Ozturk, "Enabling consumer behavior modification through real time energy pricing," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 2015, pp. 311–316.

[18] O. Anaya-Lara and E. Acha, "Modeling and analysis of custom power systems by pscad/emtdc," *IEEE Transactions on Power Delivery*, vol. 17, no. 1, pp. 266–272, 2002.

[19] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.