Should we use model-free or model-based control? A case study of battery control

Mohamad Fares El Hajj Chehade*, Young-ho Cho*, Sandeep Chinchali, and Hao Zhu
Chandra Family Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX, USA
{chehade, jacobcho, sandeepc, haozhu}@utexas.edu

2024 56th North American Power Symposium (NAPS) | 979-8-3315-2103-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/NAPS61145.2024.10741791

***Abstract*—Reinforcement learning (RL) and model predictive control (MPC) each offer distinct advantages and limitations when applied to control problems in power and energy systems. Despite various studies on these methods, benchmarks remain lacking and the preference for RL over traditional controls is not well understood. In this work, we put forth a comparative analysis using RL- and MPC-based controllers for optimizing a battery management system (BMS). The BMS problem aims to minimize costs while adhering to operational limits, by adjusting the battery (dis)charging in response to fluctuating electricity prices over a time horizon. The MPC controller uses a learning-based forecast of future demand and price changes to formulate a multi-period linear program, that can be solved using off-the-shelf solvers. Meanwhile, the RL controller requires no time-series modeling but instead is trained from the sample trajectories using the proximal policy optimization (PPO) algorithm. Numerical tests compare these controllers across optimality, training time, testing time, and robustness, providing a comprehensive evaluation of their efficacy. RL not only yields optimal solutions quickly but also ensures robustness to shifts in customer behavior, such as changes in demand distribution. However, as expected, training the RL agent is more time-consuming than MPC.**

***Index Terms*—Battery management system, reinforcement learning, model predictive control, time-series forecast.**

I. Introduction

In the domain of control problems, reinforcement learning (RL) and model-predictive control (MPC) have their merits and drawbacks. RL deals with agents in uncertain settings and optimizes actions over infinite horizons without modeling the dynamics of the environment. It excels in environments with large modeling errors but suffers from data inefficiency and safety due to possibly undesirable actions. On the other hand, MPC optimizes actions over a finite horizon, with heavy reliance on accurate environmental models. It is less concerned with safety and data efficiency, due to the deterministic nature of the optimization problem and its use of hard constraints. However, it could perform poorly due to modeling inaccuracies and the limitations of a finite horizon [1]–[3].

In power and energy systems, both RL and MPC have been explored for the battery control problem, because of its sequential decision-making nature. In the problem of battery management system (BMS), the charging and discharging schedule of an energy storage system is controlled to minimize the cost of electricity purchase over a certain period. For example, one work based on MPC uses a long short-term memory (LSTM) forecast to model the system; then, the multi-period optimization problem is solved [4]. Despite showing its superiority over conventional methods like dynamic programming and fuzzy logic, this work does not check the impact of forecasting errors on the optimality of MPC and other methods. On the other hand, RL in [5], particularly deep-Q networks (DQN), has improved the battery's operation and minimized its degradation. However, there is a lack of benchmark in general for justifying the choice of RL over currently used control methods.

In this work, a comparative analysis is conducted between the model-based MPC and the model-free RL approaches to the battery control problem. The closest to our work is [6], which compares the performance of DQN with one-step MPC. Nonetheless, the comparisons therein are limited in three aspects. First, DQNs have been outperformed by several policy-based algorithms, such as proximal policy optimization (PPO) and deep deterministic policy gradient (DDPG), which are currently the popular RL algorithms in power and energy systems [7], [8]. Second, one-step MPC is essentially a greedy algorithm that does not benefit from planning over multiple time steps, thereby compromising the optimality of the MPC results. Third, the comparison is solely based on the control cost, and does not include some important criteria, such as robustness and computational efficiency. To the best of our knowledge, our work is the first to address these three aspects and conduct a full comparative analysis between RL and MPC for BMS. The main contributions of this work are summarized below:

1) We study the optimality of the decisions made by RL and MPC for the battery control problem. We adopt the well-known, widely used algorithms for each approach.
2) We extend the basis of the comparison from control cost to include multiple other criteria: data efficiency, online computation time, and robustness.
3) Our results show that RL performs better in terms of optimality and online computation time. It is also more robust to shifts in the demand distribution. However, it is much more data-inefficient than MPC, whose optimality

*Authors have equal contributions. (Corresponding author: Mohamad Fares El Hajj Chehade.)

This work has been partially supported by NSF Grant 2130706 and ARO Grant W911NF2310266.

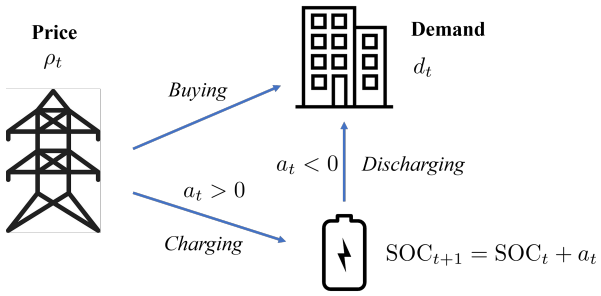

Fig. 1: The components of the system. At a given time t, the load has a demand d_t and the utility sells electricity at a price ρ_t. To minimize the total cost of purchasing electricity over a certain period, the battery controller chooses between charging from the grid ($a_t > 0$) and discharging to the load ($a_t < 0$).

is drastically affected by forecasting errors.

4) Our methodology and analysis go beyond the BMS problem and could be applicable to general control problems in power and energy systems.

The rest of the paper is organized as follows: Section II defines the problem, Sections III and IV respectively introduce the MPC and RL approaches, Section V discusses the main results, and Section VI concludes the paper.

II. Preliminaries

The battery control problem is formulated for a system of three components: a load with a time-varying demand for electricity, denoted by d_t (in kW) for a given time t, an electric utility that sells electricity at a time-varying price ρ_t (in \$/kWh), and a battery storage system of a capacity E (in kWh) and state of charge SOC_t. The latter is capable of charging electricity from the utility and discharging to the load at a rate $a_t = a_{\max}$ (in kW), as long as SOC_t remains within operational limits $\text{SOC}_{\min}$ and $\text{SOC}_{\max}$. The electricity tariff g_t at a given time t is then linear in a_t: $g_t = \rho_t(d_t + a_t)$. The interaction between the three components of the system is best illustrated in Fig. 1.

To minimize the cost of purchasing electricity, the battery must be discharged during periods of high tariffs, and charged otherwise. However, due to the battery's limited capacity, the controller must plan over a certain time horizon, to see when charging and discharging are most economical. This results in a multi-step optimization problem with cost function f, where f is the sum of individual tariffs at the considered timesteps: $f = \sum_t g_t$. Despite the linearity of the cost function in terms of the decision variables a_t, the problem is non-trivial due to the uncertainties in the demand d_t and prices ρ_t. As a result, two approaches are considered for solving the battery control problem: model-based and model-free.

III. Model-Predictive Control (MPC)

The MPC problem aims at minimizing the operating cost of the energy system over a finite time horizon $t \in \mathcal{T} = \{1, 2, \ldots, T\}$, while maintaining the battery within its operational limits. At each time step $t \in \mathcal{T}$, the first element $b_{t|t}$ in the vector of solutions obtained from each optimization problem is stored. Subsequently, another optimization problem is solved starting at the time step $t+1$. The demand and price in the problem are sampled from two time-series distributions: $\rho_{t+k|t} \sim f_\rho(t+k, \theta)$ and $d_{t+k|t} \sim f_d(t+k, \mu)$ for all $k \in \{1, 2, 3, \ldots, T\}$, where θ and μ are the parameters of the price and demand distributions, respectively. We obtain estimated values for price and demand from forecasters, i.e., $\hat{\rho}_{t+k}|t \sim f_{\hat{\rho}}(t+k, \hat{\theta})$ and $\hat{d}_{t+k}|t \sim f_{\hat{d}}(t+k, \hat{\mu})$ for all $t \in \mathcal{T}$, where $\mathcal{T}$ is the full horizon of the problem.

To tackle the inherent unpredictability of future demand and electricity prices, we employ a predictive framework relying on an LSTM network, well-suited for sophisticated time series forecasting. This neural network acts as a forecaster, projecting demand and cost factors from the current time step $t+1$ to a specified horizon $\mathcal{T}$. The MPC framework leverages the forecasted data to make informed decisions about battery charging and discharging, aiming to minimize electricity expenses and stabilize battery energy levels. To enhance the reliability of these forecasts, we engage in iterative refinement of the predictive models. This critical step involves continuously aligning the models with observed data, thereby reducing forecasting errors and improving decision-making accuracy in the MPC framework.

Given the predicted electricity cost $\hat{\rho}_{t+k|t}$ at time $t+k$, which is predicted at time t, the BMS optimization problem seeks to minimize the total cost while satisfying the operational limits, as given by:

$$\min_{a_{t|t}, \ldots, a_{t+T|t}} \quad \sum_{k=0}^{T} \hat{\rho}_{t+k|t}(E \cdot a_{t+k|t} + \hat{d}_{t+k|t}) \tag{1a}$$

$$\text{subject to} \quad \text{SOC}_{t+k+1|t} = \text{SOC}_{t+k|t} + a_{t+k|t}, \tag{1b}$$

$$\text{SOC}_{\min} \leq \text{SOC}_{t+k+1|t} \leq \text{SOC}_{\max}, \tag{1c}$$

$$\|a_{t+k|t}\| \leq a_{\max}, \tag{1d}$$

$$\forall k \in \{0, \ldots, T\}, \ \forall t \in \mathcal{T}. \tag{1e}$$

The constraints in (1b) represent the remaining battery level, $\text{SOC}_{t+k+1|t}$, at $t+k+1$ after making a charging or discharging decision at $t+k$. The battery's operational limits in (1c) range from $\text{SOC}_{\min}$ to $\text{SOC}_{\max}$. Moreover, the maximum allowable charge/discharge rate is limited by the constant $a_{\max}$ (in kW), as denoted in (1d). The problem (1) is a multi-step linear program that can be efficiently solved by off-the-shelf solvers.

IV. Reinforcement Learning (RL)

Due to its sequential decision-making nature and Markovian state space, the problem can be modeled as a Markov decision process (MDP) [9]. An MDP is defined as the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of states and actions, respectively. For an action $a \in \mathcal{A}$ that an agent takes in a given state $s \in \mathcal{S}$, $p(.|s, a)$ models the distribution over the next states. For the transition $s \xrightarrow{a} s'$, the agent receives a scalar reward $r(s, a, s')$. Finally, the discount factor $\gamma \in [0, 1]$ can be used to give smaller weights for future rewards. The elements of the MDP tuple for the BMS problem are defined below:

- **State space** ($\mathcal{S}$): The state is characterized by the tuple $(\text{SOC}_t, \rho_t, d_t, h_t, D_t)$, where:
  - SOC_t represents the state of charge of the battery at time t.
  - ρ_t is the electricity price at time t.
  - d_t indicates the energy demand at time t.
  - h_t is the hour t of the day.
  - D_t distinguishes between weekdays and weekends.
- **Action space** ($\mathcal{A}$): The action space is one-dimensional and represents the control variable a_t, defined as the amount of charge/discharge at a given time t. Since the optimal solution is always one of three values: charge with the maximum amount (a_t = $a_{\max}$ = 0.1E), discharge with the maximum amount (a_t = - $a_{\max}$ = - 0.1E), or idle (a_t = 0), the action space is discrete $A = \{-0.1E, 0, 0.1E\}$.
- **Transition function** (p): Time elements in the state space incur deterministic transitions. The state of charge progression is described below:

$$\text{SOC}_{t+1} = \begin{cases} \text{SOC}_{\min} & \text{if } \text{SOC}_t + a_t < \text{SOC}_{\min} \\ \text{SOC}_{\max} & \text{if } \text{SOC}_t + a_t > \text{SOC}_{\max} \\ \text{SOC}_t + a_t & \text{otherwise} \end{cases} \tag{2}$$

  This ensures that SOC stays within the operational limits $\text{SOC}_{\min}$ and $\text{SOC}_{\max}$. On the other hand, the electricity price and energy demand are assumed to be drawn from two time series distributions: $\rho_t \sim f_\rho$ and $d_t \sim f_d$. Modeling f_ρ and f_d is challenging and may incur significant errors. Fortunately, model-free RL allows for solving the problem without any knowledge of the time-series distributions.
- **Reward function** (r): the reward is defined as the negative of the instantaneous electricity cost of the energy consumed. The power consumed is used to satisfy the load demand and to charge the battery: $r_t = -\rho_t(d_t + a_t E)$.
- **Discount factor** (γ): Since electricity costs at future time steps matter as much as costs in the present, no discounting is applied, i.e. $\gamma = 1$.

In an MDP framework, an RL agent takes its actions based on a policy distribution π, i.e. $a_t \sim \pi(\cdot|s_t)$, where $\pi(a|s)$ is the probability of choosing action a from state s under policy π. Given that the agent is at state s_t at a given time t, the return G_t is defined as the total reward the agent receives starting at time t and until the end of the trajectory, when taking actions based on policy π. The expected value of the return is known as the value function $V^\pi(s_t)$. The goal of the RL problem is to find the optimal policy π^*, which maximizes the expected total reward the agent receives over its trajectory, i.e. $\pi^* = \max_\pi V^\pi(s)\ \forall s \in \mathcal{S}$.

In modern RL, policy-based deep RL methods have enjoyed more success than classical value-based methods [10]–[12]. In the discrete action-space setting, the policy distribution is represented by a deep neural network with parameters θ, as shown in Fig. 2. The input vector to the network is the state

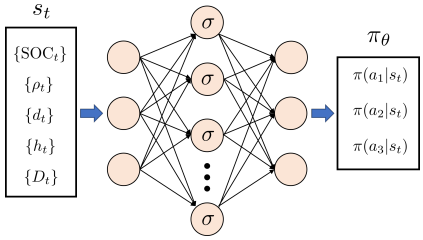

Fig. 2: The policy network π_θ. The state vector s_t propagates through the neural network layers to output the probability masses $\pi_\theta(a|s_t)\ \forall a \in \mathcal{A}$. σ is a non-linear activation function to allow the network to represent non-linear input-output relationships. A softmax function is applied at the output to convert the raw numbers (logits) to probabilities.

s, and the output vector consists of the probability masses $\pi_\theta(a|s)$. For a given state s_t, the action is sampled from the output vector distribution: $a_t \sim \pi_\theta(\cdot|s_t)$. The optimal policy $\pi^* = \pi_{\theta^*}$ is obtained by maximizing the expected return received along its experienced trajectories. In other words, if the agent experiences a trajectory τ, which produces a large return G_0^τ, the actions that yield such a trajectory are enforced by the policy network, i.e. their probabilities increase. The objective function is formally represented below:

$$\max_\theta J(\theta) = \max_\theta \mathbb{E}_\tau[G_0 \mid s_0] \tag{3}$$

$$= \max_\theta \sum_\tau P(\tau) G_0 \tag{4}$$

where $P(\tau)$ is the probability of trajectory τ, and is a function of the actions chosen by the agent, i.e. π_θ and the environment transitions p.

This can be achieved with gradient ascent in the θ space:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t \tag{5}$$

where α is the step size, and T is the number of steps taken in the trajectory.

To reduce the variance in the return during learning, an action-independent baseline is subtracted from the return in (5). In most settings, this baseline is the value function $V(s_t)$, and the objective function becomes:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(G_t - V(s_t)\right) \tag{6}$$

The value function itself can be represented as a separate neural network with parameter ϕ, and is optimized by minimizing the temporal difference (TD) error:

$$\min_\phi \left(r + \gamma V_\phi(s') - V_\phi(s)\right)^2 \tag{7}$$

In some situations, large parameter updates result in unstable learning and potential divergence. As a remedy, these updates can be ensured to be within a trust region, by enforcing the KL-divergence between the parameters before and after the update, $D_{KL}(\theta_{new}||\theta_{old})$, to be within a threshold δ. Therefore, the parameters take the largest step (towards the steepest ascent) that keeps them inside trust region. This translates to the constrained optimization problem below:

$$\begin{aligned} \theta_{\text{new}} = \arg\max_{\theta} \quad & J(\theta) \\ \text{s.t.} \quad & D_{\text{KL}}(\theta \parallel \theta_{\text{old}}) \leq \delta \end{aligned} \tag{8}$$

An approximate form of this problem is solved by the proximal policy optimization (PPO) algorithm [13], whose pseudocode is shown in Algorithm 1.

Algorithm 1 Proximal Policy Optimization (PPO)

Initialize parameters θ for policy π_θ and ϕ for value function V_ϕ
repeat
  Collect trajectories $D_k = \{s_t, a_t, r_t, s'_t\}$ using π_θ
  Compute G_t, $A_t = G_t - V_\phi(s_t)$
  for each epoch over D_k **do**
    Update θ by (approximately) solving (8)
    Update ϕ minimizing $\left(r_t + \gamma V_\phi(s'_t) - V_\phi(s_t)\right)^2$
  end for
until convergence

V. Numerical Comparisons

In this section, we compare the performance of reinforcement learning (RL) and model-predictive control (MPC) on a residential building. We consider two stages in the evaluation: training and testing. In the training phase, the RL agent learns a policy network, as outlined in Algorithm 1, while a forecaster for price and demand is trained for MPC. In the testing phase, RL and MPC are run on a new data set. The performance of each method is judged based on four criteria:

1) **control performance (optimality):** the total cost incurred on the test dataset.
2) **data efficiency:** the data needed to train the RL agent or forecaster
3) **testing time:** the time needed to make control decisions for the test dataset
4) **robustness:** the degree of optimality when the distribution of the model changes from the training dataset to the testing dataset

For this purpose, two datasets (each with training and testing sets) are considered. In the first dataset, the model of the system remains the same over the training and testing sets, and the first three criteria are tested. In the second dataset, the distribution of the demand changes from the training to the testing set, therefore allowing us to examine the robustness of each controller.

As a baseline, we consider a pre-defined policy that discharges the battery when the price is above average (computed based on the training dataset) and charges the battery otherwise - as long as the state of charge is within operational limits.

TABLE I: The performance metrics for the different controllers. The ground truth is computed by considering the actual values of demand and prices and optimizing over the entire test set. RL performs best in terms of optimality and testing time, while MPC is more data-efficient. The optimality of MPC (with true values of demand and price) indicates that errors in the regular MPC model are due to the forecaster.

(a) Optimality (control cost)

Controller	Cost ($)	Optimality Gap
RL	**85,100±200**	**7.36%**
MPC	$89,650 \pm 665$	13.10%
MPC (exact model)	79,268	0%
Baseline	90,989	14.77%
Ground truth	79,268	-
No BMS	114,065	43.89%

(b) Data Used and Testing Time

Controller	Data Used	Testing Time (s)
RL	3×10^6	5
MPC	3×10^3	30

In both experiments, the data is hourly, and the horizon for MPC is 24 hours. An LSTM network is trained in Tensorflow as the forecaster, while StableBaselines3 [14] with default parameters is used to train the RL agent. Five runs are performed for each method, for which a 95% confidence interval of the cost is computed. The simulations took place on a machine equipped with an Intel® CPU @ 2.10 GHz and 32 GB RAM.

A. Dataset 1: Consistent Model through the Training and Testing

The load corresponds to a residential building, and the real-time (RT) prices are from the Pennsylvania, New Jersey, and Maryland (PJM) region. The training and testing sets are for July 2017 and 2018 respectively.

Table I shows the performance of each controller based on the first three criteria. While the table shows that any control method that employs a battery storage system is economic (as compared with no BMS), RL achieves the highest degree of optimality, as compared with MPC and the baseline. This reinforces the capabilities of trust-region policy-based methods in reaching the optimal solution without any model of the environment [10], [13]. On the other hand, MPC suffers from modeling errors incurred by the forecaster and performs just better than the baseline predefined policy. The errors in the forecast are verified by the optimality of the MPC solution when considering the actual values for demand and price, rather than the forecasted ones.

Meanwhile, RL requires about 1000 times more data than MPC, as the RL agent requires a lot of interactions with the environment during training [2]. Nonetheless, training is performed only once, after which RL is 6 times faster during

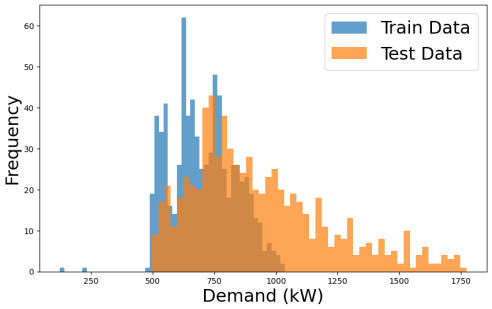

Fig. 3: The shift in the distribution of the demand from the training to the testing datasets.

TABLE II: The optimality of each controller when considering distributional shifts in the demand from training to testing. RL is robust to such shifts, as it maintains a high level of optimality due to its generalization capabilities. The performance of MPC is again drastically affected by forecasting errors.

Controller	Cost ($)	Optimality Gap
RL	**157,091±104**	**1.36%**
MPC	$161,806 \pm 740$	4.40%
MPC (exact model)	154,981	0%
Baseline	166,490	7.43%
Ground Truth	154,981	-
No BMS	179,958	16.12%

the testing phase. This is because decision-making for RL involves a forward propagation over the policy network (Fig. 2), while MPC has to solve a multi-step optimization problem every time a decision is made.

B. Dataset 2: Distributional Shifts in the Demand from the Training to the Testing Sets

Training and testing data are for July and August 2017, respectively. The price data are again retrieved from PJM, while the demand is for a commercial building in South Korea. Fig. 3 illustrates the differences in the distributions of demand between training and testing.

Table II displays the total cost incurred by each controller, where RL is shown to maintain its high level of optimality. This not only emphasizes the advantages of model-free learning utilized by RL, but also RL's ability to learn general rules that transcend changes in the model. On the other hand, the errors in the forecaster are magnified by this distributional shift, which resulted in a relatively high control cost for MPC.

VI. Conclusion

To sum up, we have developed and compared the model-based MPC and model-free RL controllers for the battery management system. Although RL requires a longer training time compared to MPC, it can achieve optimal control and offers a faster testing time than MPC. We summarize the general comparison of RL and MPC for optimal control in Table III. Our future research directions include considering more generalized energy management systems with photovoltaic panels and multiple loads, as well as implementing different types of forecasters to handle uncertainties arising from those components.

TABLE III: Comparison of RL and MPC for the optimal control of the battery storage system. RL yields more optimal solutions in less time. Furthermore, its solutions are robust to changes in customer behavior, represented by shifts in the demand distribution. On the other hand, training the RL agent requires much more time than MPC.

Comparison criteria	RL	MPC
Optimality	✓	✗
Data Efficiency	✗	✓
Online Computation	✓	✗
Robustness	✓	✗

References

[1] H. Zhang, S. Seal, D. Wu, F. Bouffard, and B. Boulet, "Building energy management with reinforcement learning and model predictive control: A survey," *IEEE Access*, vol. 10, pp. 27 853–27 862, 2022.

[2] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.

[3] Y. Lin, J. McPhee, and N. L. Azad, "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, 2020.

[4] H. Chen, R. Xiong, C. Lin, and W. Shen, "Model predictive control based real-time energy management for hybrid energy storage system," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 4, pp. 862–874, 2020.

[5] K.-b. Kwon and H. Zhu, "Reinforcement learning-based optimal battery control under cycle-based degradation cost," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4909–4917, 2022.

[6] A. S. Zamzam, B. Yang, and N. D. Sidiropoulos, "Energy storage management via deep Q-networks," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2019, pp. 1–5.

[7] J. Feng, Y. Shi, G. Qu, S. H. Low, A. Anandkumar, and A. Wierman, "Stability constrained reinforcement learning for decentralized real-time voltage control," *IEEE Transactions on Control of Network Systems*, 2023.

[8] Y. Chen, Y. Shi, D. Arnold, and S. Peisert, "SAVER: Safe learning-based controller for real-time voltage regulation," in *2022 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2022, pp. 1–5.

[9] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] Stable-Baselines3, "Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 2.2.1 documentation," Online, Accessed 2023. [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/