ASME Letters in Dynamic Systems and Control
Online journal at:
https://asmedigitalcollection.asme.org/lettersdynsys

Reinforcement Learning-Based Event-Triggered Model Predictive Control for Electric Vehicle Active Battery Cell Balancing

David Flessner
Department of Electrical and Computer Engineering,
Oakland University,
Rochester, MI 48309
e-mail: dflessner@oakland.edu

Jun Chen[1]
Department of Electrical and Computer Engineering,
Oakland University,
Rochester, MI 48309
e-mail: junchen@oakland.edu

To extend the operation window of batteries, active cell balancing has been studied in the literature. However, such an advancement presents significant computational challenges on real-time optimal control, especially when the number of cells in a battery increases. This article investigates the use of reinforcement learning (RL) and model predictive control (MPC) to effectively balance battery cells while at the same time keeping the computational load at a minimum. Specifically, event-triggered MPC is introduced as a way to reduce real-time computation. Different from the existing literature where rule-based or threshold-based event-trigger policies are used to determine the event instances, deep RL is explored to learn and optimize the event-trigger policy. Simulation results demonstrate that the proposed framework can keep the cell state-of-charge variation under 1% while using less than 1% computational resources compared to conventional MPC. [DOI: 10.1115/1.4067656]

Keywords: electric vehicles, active cell balancing, model predictive control, event-triggered control, reinforcement learning

1 Introduction

Batteries and electric vehicles are key enablers of green technologies that have the potential to address climate challenges [1,2]. Among many limitations, cell imbalance is a critical issue that limits the range of electrical vehicles [3,4]. In battery packs with multiple cells, the individual cells have state-of-charge (SOC) and terminal voltage variation among them that arise from several factors such as manufacturing variation and uneven aging [4,5]. Discharging an individual cell below a minimum voltage results in an accelerated degradation of the pack capacity and poses safety risks. Therefore, the total usable energy of a battery pack is limited by the lowest capacity cell to avoid cell damage and safety risks. Similarly, the total charging capacity of a battery pack is also limited to whichever cell has the lowest capacity and charges to its full capacity first. See Refs. [3,6] for more details.

Cell balancing control is a technology that has arisen to address these issues [7]. Cell balancing methods can be classified into passive and active methods, where the major difference is whether the balancing method relies solely on resistive elements to dissipate energy or relies on charge transfer that preserves energy [8]. This article focuses on active balancing control, where the goal is to reduce the amount of capacity loss at the battery pack level by moving electricity from cells to cells to maximize the usable energy stored in a battery pack. One of the challenges of active balancing control is the real-time computation of the required balancing currents for optimal balancing performance. In this regard, model predictive control (MPC) has then been widely used in the literature [3,9,10] due to its capability of handling both input and state constraints. MPC is an advanced control technique that has been widely studied and successfully applied in many applications [11]. MPC calculates the optimal control actions by minimizing a cost function, together with a model-based prediction of system evolution, over a prediction horizon.

Despite significant research efforts in the literature, the large computational load of MPC prevents its successful demonstration of large battery packs. To address this, event-triggered MPC has been studied in the literature [12,13], where an event triggers the MPC controller to calculate a new optimal control sequence when certain event-trigger conditions are met. Moreover, this optimal control sequence will then be stored in memory. During the absence of an event, the optimal control sequence stored in memory will be used to determine the control action for the current step. However, the design and calibration of such an event-trigger policy is not trivial. Therefore, to realize the largest benefit of event-triggered MPC, a reinforcement learning (RL) agent is investigated in this article for the purpose of finding the optimal event-trigger policy. Specifically, a deep neural network (DNN) is used to approximate the Q-function to capture the value of triggering versus not triggering. Compared to existing work that utilized RL to find the optimal event-trigger policy for MPC, such as Ref. [14], the present work investigates a completely new setting of the reward function for the RL agent. Specifically, a time-based reward is used in Ref. [14], while SOC variation is used in the present work as the reward function, which relates closer to the goal of balancing control. In addition, eligibility is used to evenly distribute the triggers to encourage more periodic trigger behavior. The proposed RL-based event-triggered MPC, termed as RLeMPC, is demonstrated to be effective in reducing more than 99% real-time computation while keeping cell SOC variation at an acceptable level, i.e., under 1%. Finally, the proposed approach significantly simplifies the design of event-trigger policy using a model-free policy network, without knowing the dynamics of the MPC closed-loop system.

The remainder of the article is organized as follows. Section 2 presents preliminaries on RL and MPC-based active battery balancing control. The proposed RLeMPC active balancing control is presented in Sec. 3, together with simulation results discussed in Sec. 4. The article is concluded in Sec. 5.

2 Problem Formulation and Preliminaries

2.1 Model Predictive Cell Balancing Control. The configuration considered in this article is illustrated in Fig. 1, where N

[1]Corresponding author.

Manuscript received September 6, 2024; final manuscript received December 31, 2024; published online February 5, 2025. Assoc. Editor: Javad Mohammadpour Velni.

cells are connected in series. A second-order equivalent circuit model can be used to model SOC, relaxation voltage, and terminal voltage of each cell as follows [9,15]:

$$s_{k+1}^n = s_k^n - \eta^n \frac{T_s}{3600C^n} i_k^n \tag{1a}$$

$$V_{p,k+1}^n = V_{p,k}^n - \frac{T_s}{R_p^n C_p^n} V_{p,k}^n + \frac{T_s}{C_p^n} i_k^n \tag{1b}$$

$$y_k^n = V_{oc,k}^n - V_{p,k}^n - i_k^n R_o^n \tag{1c}$$

where the superscript n denotes the nth cell, s^n is the cell SOC, η^n is the cell coulombic efficiency, C^n is the cell capacity in amp hours, V_p^n is the relaxation voltage over R_p^n, V_{oc}^n is the open circuit voltage, y^n is the terminal voltage, i^n is the cell current, and T_s is the sampling time. Positive i^n indicates discharging the battery while negative i^n indicates charging. The variables V_{oc}^n, R_o^n, R_p^n, and C_p^n are all dependent on s^n resulting in a nonlinear cell model. Moreover, due to manufacturing and operating variations, such dependency varies among cells, making (1) heterogeneous for each cell. For pack-level modeling, (1) can be stacked together to model the dynamics of the entire battery pack.

As shown in Fig. 1, a power converter is used to actively move electricity from any cell to any cell to balance cell SOC. The following optimal control problem (OCP) is utilized to solve for the optimal balancing currents for each time-step, where k is the time index and p is the prediction horizon.

$$\min_{U_k} \quad J = \sum_{j=1}^{p}\sum_{n=1}^{N}(y_{k+j}^n - y_{k+j}^0)^2 + \sum_{j=0}^{p-1}\sum_{n=1}^{N} r\left(u_{k+j}^T\right)^2 \tag{2a}$$

$$\text{s.t.} \quad \text{system dynamics (1)} \tag{2b}$$

$$u_{\min} \le u_k^n \le u_{\max}, \qquad 1 \le n \le N \tag{2c}$$

$$y_{\min} \le y_{k+j}^n, \quad 1 \le j \le p, \ \ 1 \le n \le N \tag{2d}$$

$$0 = \sum_{i=1}^{N} u_k^n \tag{2e}$$

where the cost function (2a) is based on minimizing the tracking error between each individual cell to a nominal cell without any imbalance, which was studied in Ref. [9]. Weight r is positive. The first term penalizes cell voltages that deviate from the nominal cell while the second term penalizes the magnitude of balancing current to reduce resistant heating losses. Note that constraint (2c) limits the magnitude of each balancing current, while constraint (2d) requires the MPC to avoid cell voltages dropping below a discharge voltage limit $y_{\min}$, if possible. Lastly, constraint (2e) is a simplification of a power balancing equation considering that the variation of cell voltage is minimum. Note that more details about the model predictive cell balancing control can be found in Ref. [9].

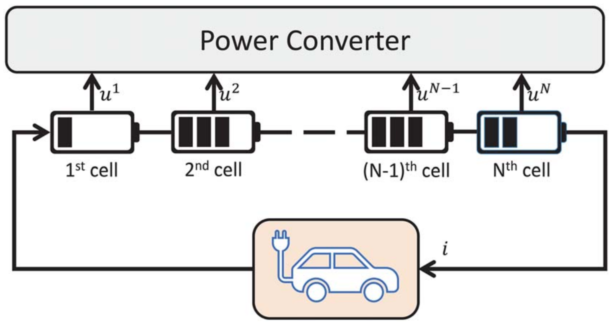

Fig. 1 Battery pack configuration for active cell balancing, where the power converter circuit is capable of moving electricity from any cell to any cell

2.2 Preliminaries on Reinforcement Learning.

The RL paradigm [16] operates on a state–action framework where at each time-step t, the state of the environment, s_t, is observed by an agent which then selects an action a_t to take. Note that in the context of this article, average and minimum cell voltages, average cell SOC, and pack current demand are used as state and the action is binary with 1 triggering a new optimization instance and 0 reusing the previous control sequence.

After the action is taken, a scalar reward $r(s_t, a_t)$ is given to the agent. The goal of the agent is to learn an optimal policy $\pi^*: s \rightarrow a$ that maximizes the expected cumulative future rewards:

$$G = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \tag{3}$$

where $r_t = r(s_t, a_t)$ and the scalar $\gamma \in [0, 1]$ is the discount factor that reduces the value of future expected rewards. Note that here $E_\pi(\cdot)$ denotes the expectation under policy π.

To measure the value of the agent being in state s and taking action a under policy π, a state–action value function $Q_\pi(s, a)$, termed as a Q-function or Q-value, can be defined as the expected return starting from s followed by action a and then policy π:

$$Q^\pi(s, a) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k} \mid s_t = s, a_t = a\right] \tag{4}$$

Through the agent's interactions with the environment, the Q-function can be learned. Then the optimal policy π^* can be defined as

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{5}$$

In other words, the optimal policy π^* is to select the action that results in the highest Q-value for a given state. To model Q-function $Q(s, a)$ over a large number of state–action pairs, a DNN can be used with the environment state variables as inputs, and the Q-value for each action as the output. This implementation of RL is also known as deep Q-learning [17,18] and the DNN used here can be referred to as a deep Q-network (DQN). The goal of training the network is to adjust the DQN parameters while the RL agent interacts with the environment to closely approximate the actual Q^* function. Then, this Q^* function, as the optimal policy, can be used to select the action with the highest expected reward as described in (5).

3 Reinforcement Learning-Based Event-Triggered Model Predictive Control for Active Cell Balancing

Conventional MPC periodically solves OCP (2) for an optimal control sequence U_k, where the first element is implemented by the power converter and the remaining elements of U_k are abandoned. The whole process repeats for the next time-step where MPC solves another instance of OCP (2). However, solving OCP (2) requires significant computation, and hence the conventional MPC approach is intractable when the number of cells, N, is large. To address this challenge, event-triggered MPC is introduced, where an event γ_{ctrl} is used to trigger a new instance of OCP. In other words, when $\gamma_{\text{ctrl}} = 1$, a new instance of (2) is solved; otherwise when $\gamma_{\text{ctrl}} = 0$, the optimal control sequence obtained at the previous instance is reused for balancing control.

In Ref. [14], RL is used to learn the optimal event-trigger policy, where a time-based reward function is used. This article continues

Table 1 Cell imbalance parameters at 100% SOC

Param.	Unit	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
C	A h	62.87	60.00	66.61	56.73	61.66
R_p	mΩ	6.40	5.66	5.47	6.68	6.36
C_p	kF	153.7	177.8	175.9	168.9	150.4
R_o	mΩ	1.49	1.27	1.41	1.51	1.53

Table 2 Key training hyperparameters

Parameter	Value
α (learning rate)	10^{-5}
γ (discount factor)	0.95
λ (eligibility discount rate)	0.95
ε_0 (initial value for ε)	0.05
ε-decay rate	5×10^{-7}
Max episodes	500
M (mini-batch size)	256
$\lvert\mathcal{D}\rvert$ (experience buffer length)	10,000

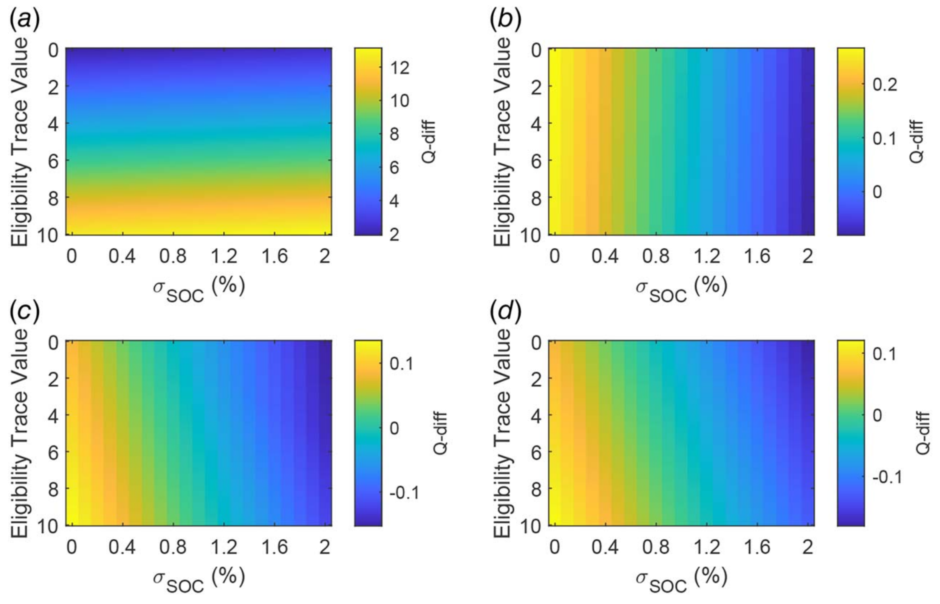

Fig. 2 Q-diff for $\lambda = 0.9$ and various ρ: (a) $\rho = 2$, (b) $\rho = 0.2$, (c) $\rho = 0.02$, and (d) $\rho = 0.002$

to investigate this framework using a more direct reward function based on SOC variation at each time-step. During training, the RL agent will select an action between $\gamma_{\text{ctrl}} = 0$ and $\gamma_{\text{ctrl}} = 1$ using ε-greedy. In other words, with a probability of ε, the RL agent will choose randomly between $\gamma_{\text{ctrl}} = 0$ and $\gamma_{\text{ctrl}} = 1$, while with a probability of $1 - \varepsilon$, the action with higher Q-value will be chosen. At each time-step t, the RL agent receives a reward, defined as follows:

$$r_t = -\sigma_{\text{SOC}} - e_t * \rho \tag{6}$$

where σ_{SOC} is the standard deviation of the cell SOC at time-step t and ρ is a hyperparameter that balances the event-trigger frequency and balancing performance. The first term of (6) encourages lower SOC variation, which is the primary objective of cell balancing control, while the second term penalizes frequent event to reduce computation load.

Moreover, e_t in (6) implements an eligibility trace as follows. At each time-step, if $\gamma_{\text{ctrl}} = 1$, the eligibility trace e_t is incremented by 1; otherwise if $\gamma_{\text{ctrl}} = 0$, e_t decays by a factor of λ. This can be described as

$$e_t = \begin{cases} \lambda e_{t-1} & \text{if } \gamma_{\text{ctrl}} = 0 \\ \lambda e_{t-1} + 1 & \text{if } \gamma_{\text{ctrl}} = 1 \end{cases} \tag{7}$$

With e_t, the penalty of triggering an event $\gamma_{\text{ctrl}} = 1$ is distributed for subsequent time-steps. The goal of using e_t is to encourage RL to distribute the triggers more evenly through time to approach periodic trigger behavior but with more freedom to trigger when σ_{SOC} can be improved.

4 Results and Discussions

To simulate imbalance, cell parameters, such as C^n, R_o^n, and C_p^n, are randomly generated as listed in Table 1. In addition, the power converter used for balancing is assumed to have a limit of ±2 A for each cell. Repeated FTP-72 drive cycles[2] are used to generate battery discharge and charge currents, assuming the same vehicle configuration as in Ref. [9]. To speed up the simulation, the discharge and charge currents are scaled so the overall simulation time is manageable. Note that the results reported in this article require around 10 hours to train the RL agent on a standard laptop computer with an Intel® Core® i5-6600K processor and 8 GB of RAM.

To tune hyperparameter ρ, simulations are run with various values for ρ, with results shown in Fig. 2, where the z-axis is the difference between the Q-value estimates (termed as Q-diff) for the no trigger action and the trigger action, i.e., $Q(s, 0) - Q(s, 1)$. Therefore, a positive Q-diff value will indicate $\gamma_{\text{ctrl}} = 0$ and a negative Q-diff value will indicate $\gamma_{\text{ctrl}} = 1$. As can be seen, when $\rho = 2$, which places a heavy penalty on MPC computation, the RL agent learns not to trigger MPC throughout the whole δ_{SOC} spectrum. When $\rho = 0.2$, the RL agent learns to trigger MPC when δ_{SOC} is significantly large. However, the optimal policy in this case is very reactive to δ_{SOC} only and does not relate to the eligibility trace. When $\rho = 0.02$ and $\rho = 0.002$, the optimal policy learns to trigger MPC even when the SOC variation is small, which aligns with the objective of keeping SOC consistent throughout the entire battery pack. Therefore, for the rest of this study, we will

[2]https://en.wikipedia.org/wiki/FTP-75

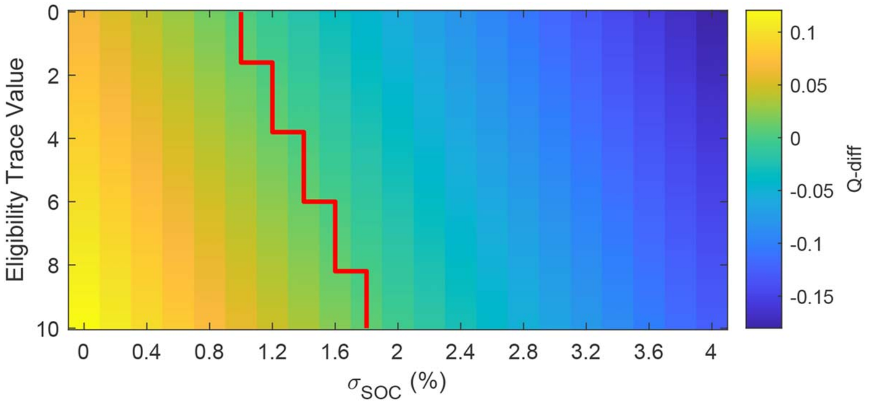

Fig. 3 ***Q*-diff for $\rho = 0.002$ and $\lambda = 0.95$, where the region to the left of the dividing line corresponds to not to trigger MPC and the region to the right corresponds to trigger, if a greedy policy is used**

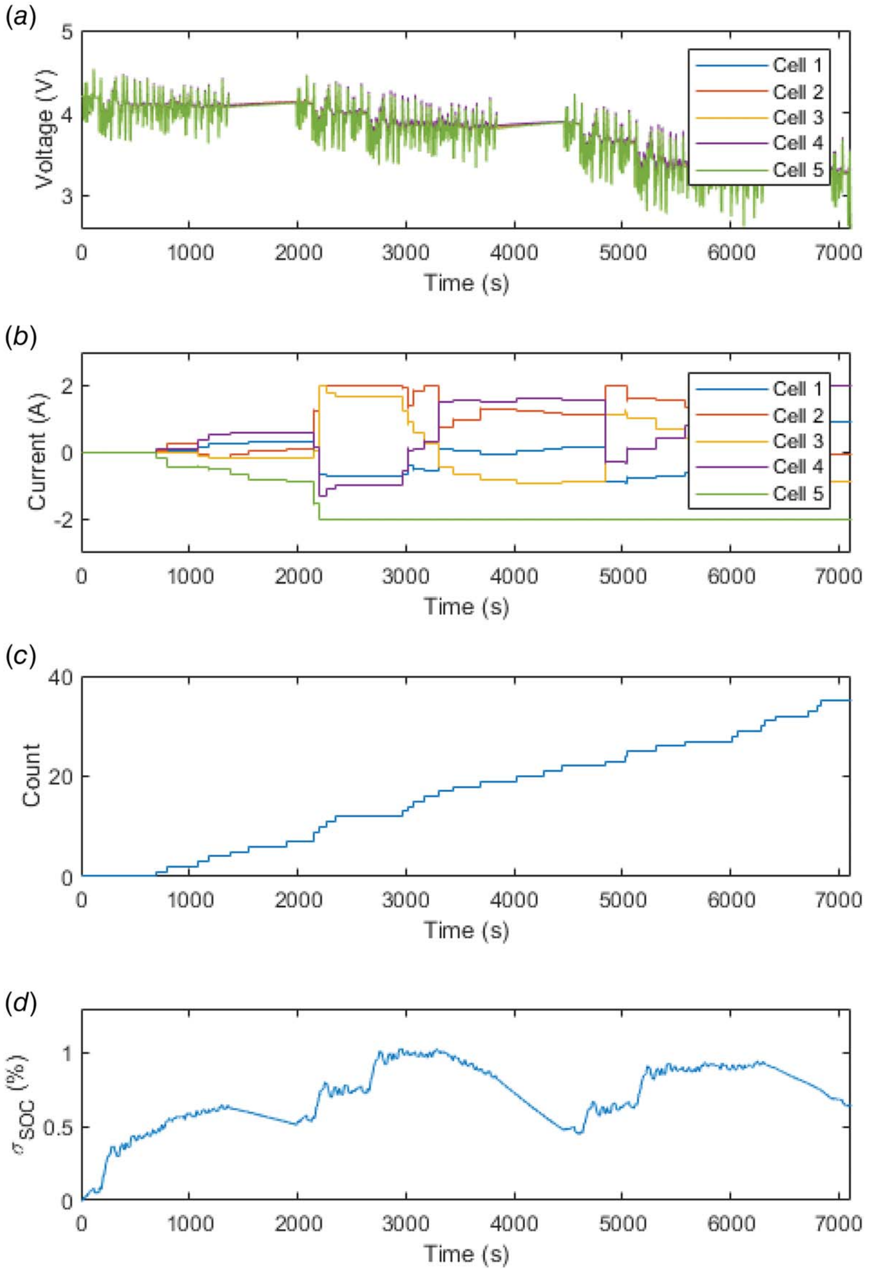

Fig. 4 **Final results for $\rho = 0.002$ and $\lambda = 0.95$: (*a*) Cell Voltage, (*b*) Balancing Currents, (*c*) Trigger counter, and (*d*) σSOC**

keep $\rho = 0.002$. Other hyperparameters, such as discounted factors γ and λ, are similarly tuned. Table 2 lists all the hyperparameters used in the numerical simulation. In addition, the DQN architecture includes two layers, a fully connected layer with four nodes followed by a ReLU (rectified linear unit) layer and another fully connected layer with two nodes.

Figures 3 and 4 show the final results with $\rho = 0.002$ and $\lambda = 0.95$, where Fig. 4 plots the individual cell voltage, balancing currents, event-trigger counts, as well as the time profile of σ_{SOC}. Note that the optimal policy in Fig. 3 is similar to that of Fig. 2(d), but the core differences become clear when comparing tests using the final learned ϕ on-policy. In general, the policy in Fig. 2(d) can achieve better balancing results with higher event-trigger frequency. Due to the space limitation, we only focus on the policy in Fig. 3. It can be seen clearly that the proposed RL-based event-triggered MPC can keep the SOC variation under 1% throughout the whole simulation, while requiring an average of 1 MPC computation every 175 s. Note that for balancing methods based on conventional MPC, such as Refs. [3,9], sampling time T_s is usually 1 s, meaning that MPC solves a new optimization problem for every 1 s. Therefore, the RLeMPC framework only requires less than 1% online computation for MPC.

As discussed in Ref. [14], the proposed RLeMPC can generally achieve comparable control performance as threshold-based event-triggered MPC. However, the proposed approach significantly simplifies the design of event-trigger policy using a model-free policy network, without knowing the dynamics of the MPC closed-loop system.

5 Conclusion

This article studies the active balancing control problem for electric vehicle battery packs. While existing work on model predictive control (MPC) demonstrates the applicability of real-time balancing control, the computational requirement of MPC prevents its usage on large battery packs. In this regard, this article proposes a framework that integrates reinforcement learning, event-triggered control, and MPC, termed as RLeMPC. More specifically, a deep neural network is used to learn the optimal event-trigger policy without knowing the closed-loop system dynamics. The proposed framework is illustrated through simulation, where results suggest that the proposed RLeMPC framework can reduce the computational load by up to 99% while keeping the cell SOC variation well under 1%. Compared to conventional MPC, the computation reduction can in turn reduce the cost as lower-end microcontrollers can now be used. Compared to threshold-based event-triggered MPC, the proposed method reduces the need to manually tune the threshold, which generally requires the knowledge of the dynamics of the MPC closed-loop system, which can be difficult to obtain. The primary future work stream would be to use a hyperparameter tuning tool to find the hyperparameters that could maximize the capability of the RLeMPC strategy. Another future work is to scale the simulation to include more cells.

Acknowledgment

This work is supported in part by National Science Foundation through Award #2237317.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Chen, J., Liang, M., and Ma, X., 2021, "Probabilistic Analysis of Electric Vehicle Energy Consumption Using MPC Speed Control and Nonlinear Battery Model," 2021 IEEE Green Technologies Conference, Denver, CO, Apr. 7–9, pp. 181–186.
[2] Singh, A., Rustagi, R., and Hegde, R. M., 2024, "Lifetime Improvement in Rechargeable Mobile IoT Networks Using Deep Reinforcement Learning," IEEE Trans. Circuits Syst. II: Express Briefs, **71**(7), pp. 4005–4009.
[3] Hoekstra, F. S. J., Ribelles, L. A. W., Bergveld, H. J., and Donkers, M. C. F., 2020, "Real-Time Range Maximisation of Electric Vehicles Through Active Cell Balancing Using Model-Predictive Control," 2020 American Control Conference, July 1–3, Denver, CO, pp. 2219–2224.
[4] Chen, J., Zhou, Z., Zhou, Z., Wang, X., and Liaw, B., 2022, "Impact of Battery Cell Imbalance on Electric Vehicle Range," Green Energy Intell. Transp., **1**(3), pp. 1–8.
[5] Dubarry, M., Vuillaume, N., and Liaw, B. Y., 2010, "Origins and Accommodation of Cell Variations in Li-Ion Battery Pack Modeling," Int. J. Energy Res., **34**(2), pp. 216–231.
[6] Plett, G. L., 2015, *Battery Management Systems, Volume II: Equivalent-Circuit Methods*, Artech House, Norwood, MA.
[7] Omariba, Z. B., Zhang, L., and Sun, D., 2019, "Review of Battery Cell Balancing Methodologies for Optimizing Battery Pack Performance in Electric Vehicles," IEEE Access, **7**(1), pp. 129335–129352.
[8] Einhorn, M., Roessler, W., and Fleig, J., 2011, "Improved Performance of Serially Connected Li-Ion Batteries With Active Cell Balancing in Electric Vehicles," IEEE Trans. Veh. Technol., **60**(6), pp. 2448–2457.
[9] Chen, J., Behal, A., and Li, C., 2024, "Active Battery Cell Balancing by Real Time Model Predictive Control for Extending Electric Vehicle Driving Range," IEEE Trans. Autom. Sci. Eng., **21**(3), pp. 4003–4015.
[10] Gong, Z., van de Ven, B. A. C., Gupta, K. M., da Silva, C., Amon, C. H., Bergveld, H. J., Donkers, M. C. F. T., and Trescases, O., 2019, "Distributed Control of Active Cell Balancing and Low-Voltage Bus Regulation in Electric Vehicles Using Hierarchical Model-Predictive Control," IEEE Trans. Ind. Electron., **67**(12), pp. 10464–10473.
[11] Alessio, A., and Bemporad, A., 2009, *A Survey on Explicit Model Predictive Control* (Nonlinear Model Predictive Control), Springer, New York, pp. 345–369.
[12] Li, H., and Shi, Y., 2014, "Event-Triggered Robust Model Predictive Control of Continuous-Time Nonlinear Systems," Automatica, **50**(5), pp. 1507–1513.
[13] Luo, Y., Xia, Y., and Sun, Z., 2019, "Robust Eventtriggered Model Predictive Control for Constrained Linear Continuous System," Int. J. Robust Nonlinear Control, **29**(5), pp. 1216–1229.
[14] Flessner, D., Chen, J., and Xiong, G., 2024, "Reinforcement Learning-Based Event-Triggered Active Battery Cell Balancing Control for Electric Vehicle Range Extension," Electronics, **13**(5), pp. 1–22.
[15] He, H., Xiong, R., Zhang, X., Sun, F., and Fan, J., 2011, "State-of-Charge Estimation of the Lithiumion Battery Using an Adaptive Extended Kalman Filter Based on an Improved Thevenin Model," IEEE Trans. Veh. Technol., **60**(4), pp. 1461–1469.
[16] Sutton, R. S., and Barto, A. G., 2018, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.
[17] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., 2013, "Playing Atari With Deep Reinforcement Learning," URL 1312.5602.
[18] Van Hasselt, H., Guez, A., and Silver, D., 2016, "Deep Reinforcement Learning With Double Q-Learning," Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30.1, Phoenix, AZ, Feb. 12–17, pp. 2094–2100.