# Reinforcement Learning for Energy Storage Arbitrage in the Day-Ahead and Real-Time Markets with Accurate Li-Ion Battery Dynamics Model

by

Dheekshita Kumar

S.B., Engineering as Recommended by the Department of Mechanical Engineering and Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2020)

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Robert C. Armstrong
Professor, Director of MIT Energy Initiative
Thesis Supervisor

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Apurba Sakti
Research Scientist, MIT Energy Initiative
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

# Reinforcement Learning for Energy Storage Arbitrage in the Day-Ahead and Real-Time Markets with Accurate Li-Ion Battery Dynamics Model

by

Dheekshita Kumar

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Decarbonizing power systems will require introducing renewable sources to the energy supply mix. Intermittent sources in the supply mix, however, make balancing energy supply and demand more challenging. Energy storage systems can be used to balance supply and demand by storing energy when renewable sources generate more energy than needed, and providing energy when generation is insufficient. Failing to account for degradation, however, when operating a battery can dramatically reduce the battery's life span and increase degradation-related costs. Existing optimization techniques that account for degradation when determining the optimal battery operation policies are both computationally intensive and time-consuming. Machine Learning techniques like reinforcement learning, can develop models that calculate action-policies in milliseconds and account for complicated system dynamics. In this thesis, we consider the problem of battery operation for energy arbitrage. We explore the use of reinforcement learning to determine arbitrage policies that account for degradation. We compare policies learned by reinforcement learning to the optimal policy, as determined by an advanced mixed-integer linear programming (MILP) model, on NYISO 2013 day-ahead electricity price data. We show that accounting for reinforcement learning results in learned policies that are comparable to the behavior of MILP-determined policies with degradation. We then present a case study that uses reinforcement learning to determine arbitrage policies on PJM 2019 real-time electricity price data, and we find that the use of reinforcement learning for real-time battery operations in the case of energy arbitrage, has promise.

Thesis Supervisor: Robert C. Armstrong
Title: Professor, Director of MIT Energy Initiative

Thesis Supervisor: Apurba Sakti
Title: Research Scientist, MIT Energy Initiative

## Acknowledgments

The writing of this thesis would not have been possible without the support of several individuals. I would like to thank Dr.Apurba Sakti for asking questions that allowed me to reflect on my work and for always reminding me of the bigger picture. I'd like to thank Prof.Robert Armstrong for agreeing to co-supervise this thesis and Prof.Jing Kong for her guidance throughout the semester as my academic advisor.

I would also like to extend my gratitude to Dr.Mehdi Jafari for answering all my technical questions, meeting with me for several hours throughout the process to clarify and refine my research questions, encouraging me to try new ideas, and helping me develop the frameworks to implement those ideas. This project would not have reached this stage without you.

I'd also like to thank Dr.Audun Botterud for asking technical questions in our group meetings that encouraged me to fill in and flesh out my understanding of more subtle points in this research project.

Last, but not least, I would never have been able to finish my last semester at MIT in the middle of a pandemic without support from my friends and family. Thank you all for encouraging me to write during our working sessions, letting me bounce my research ideas off you, and for supporting me throughout this process.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 The Role of Energy Storage in Decarbonization

Climate change poses risks to both human and natural systems around the world. Mitigating climate change will require substantial decarbonization particularly in the power generation sector which accounts for 25% [1] of the world's emissions.

Significant decarbonization of power systems will require replacing carbon-intensive generation sources with low- or zero-carbon generation sources like wind and solar. The challenge, however, of integrating renewable sources like wind and solar with the power system grid is controlling their generated electrical output. The power system, like any system, must balance the generated power (supply) with the load (demand) at all times. An imbalance in these two quantities can lead to significant losses and/or power outages. It is difficult to balance the load with a renewable supply because both wind and solar generation are weather-dependent and not controllable. Grid scale energy storage can help reinstate balance in the system by shifting energy from times of low demand to times of high demand [2].

Therefore, as renewable penetration increases in the electricity supply mix, the need for integrating grid-scale energy storage to provide load shifting, frequency regulation, grid stabilization, and energy management services will increase. Energy storage, however, is expensive [3]. Traditional lithium-ion (Li-ion) batteries, which are an increasingly popular choice of energy storage due to their lack of geo-spatial constraints (i.e. they can be installed anywhere) boast charging and discharging efficiencies of up to 90% [4], but are priced at \$500-650/kW and \$175-200 per kWh [3] [5]. Energy storage becomes a more compelling investment when it can generate revenue and offset its costs. One promising avenue for revenue generation is real-time energy arbitrage. In energy arbitrage, storage asset owners can take advantage of fluctuations in the electricity market to buy electricity (charge battery) when prices are low and sell (discharge battery) when prices are high, therefore turning a profit.

## 1.2 Energy Arbitrage Background

There are two main markets that are explored in energy arbitrage literature, the day-ahead market (DAM) and the real-time market (RTM). To participate in the day ahead market, the day before trading day, storage asset owners submit bids to buy energy and offers to sell energy at particular price points for each hour of the day. After the market operator has received bids and offers from all potential market participants, the operator balances the supply with the forecasted demand in a process known as "clearing the market". After this occurs, the asset owner is told which of their bids and offers are accepted. During the trading day, however, there is often a mismatch between the forecasted and actual supply and demand. This discrepancy is exacerbated in markets with high renewable penetration (since renewables generation is highly weather dependent, and therefore, difficult to forecast). The imbalance in the market is rectified via intra-day trading in the real-time market, which often operates on a smaller interval such as 15 minutes or 5 minutes. The authors of [6] suggest that a time resolution of one-hour can increase costs associated with balancing and propose the day ahead market switches to a time resolution of 15 minutes. At a higher granularity, flexible resources like renewable sources are more valued and better utilized due to their ability to quickly ramp up/down [7, 8] and the variability in the market is better captured. The trade-off, however, is that high granularity requires market participants to have more computational power and the ability to quickly and effectively make arbitrage decisions.

Thus there is a need for methods that can determine good energy arbitrage policies quickly.

## 1.3 Related Works

### 1.3.1 Methods for Determining Battery Operation Policies

Many studies have been done on optimizing the control of a battery for energy arbitrage. In [9–14], mixed integer linear programming (MILP) is used to determine an optimal energy arbitrage policy under specific electricity market conditions in Australia and the United States at a one-hour time interval. The policy is then used to assess the value of energy storage participation in those markets. [15] uses MILP to optimize over PJM's 5 minute interval real-time 2014 market prices and determine upper and lower bounds on profit from potential market participants. In [16], the authors propose a stochastic bidding approach for energy arbitrage in the day-ahead market. Their approach uses an uncertain day-ahead market price forecast and adjusts bids in the real-time market for feasibility. In [17], a look-ahead technique is used to optimize for both clearing the market and maximizing profits from energy arbitrage in the day ahead market. The research in [9–17], however does not explore explore more realistic battery models.

### 1.3.2 Accounting for Battery Degradation in Arbitrage Policy

The studies estimate the most profitable charging and discharging schedule for energy storage based on electricity market prices, but most of these characterize the efficiency of the battery as a fixed percentage and do not account for degradation at all [9–13], or consider the lifetime of the battery [14], but do not degrade the capacity accordingly. Assumptions about battery lifetime, efficiency, and degradation, however, are critical for obtaining realistic estimates of profitability [18].

Degradation and efficiency are hard to account for in energy arbitrage models because most arbitrage models are formulated as linear optimization problems and the degradation and efficiency characteristics of a battery are non-linear.

Attempts have been made, however, to capture some of the non-linear behavior through linearizing behavior over shorter time frames (piece-wise linear segments). Sakti et al (2017) [18], propose a model that linearizes the non-linear efficiency curves on smaller time intervals. With this model, the power-limits and efficiency of a battery are more accurately represented, and the profits generated by the arbitrage model decreased by as much as 10%, as compared to the profits when the characteristics of the battery were held constant. Maheshwari et al. (2020) [19] used similar piece-wise linearzation techniques to develop a degradation model that is compatible with MILP, but still preserves some non-linearity. They found that accounting for degradation in the charging and discharging schedule yielded approximately 80% of the maximum revenue possible (without degradation), but the new scheduling reduced battery capacity degradation by 81.6%.

While such methods can capture some non-linearities, they require special linearization techniques that dramatically increase computation time and resources. The increased computation time, however, makes it difficult to use such methods in real-time operation. Use in real-time operation is also made difficult since such methods cannot easily adapt to changes in market prices and battery state. The model has to recalculate the optimal battery cycling policy, undergoing a computationally intensive process, each time a new set of prices or battery parameters is introduced.

The challenge, therefore, is calculating a good arbitrage policy that accounts for the complicated dynamics of the battery environment, easily handles changes in prices and battery parameters, and has a small computation time. Reinforcement Learning has been proposed as method to develop machine learning models that can output close-to-optimal arbitrage policies while addressing the aforementioned challenges.

### 1.3.3 Reinforcement Learning (RL) for Energy Arbitrage

Machine Learning models can capture complex non-linear functions and after proper training, can generalize to changes in price signal data and changes in battery parameters. After training, machine learning models take milliseconds to generate close-to-optimal policies. This would enable real-time operational decision-making that uses the most recent and informed forecast.

In [20–22], the authors used reinforcement learning to learn arbitrage policies for controlling energy storage units for buildings. Authors in [23–26] used reinforcement learning to control a grid-scale battery for energy arbitrage in the day-ahead hourly electricity market. In order to simplify the training process, these studies do not account for non-linear battery degradation or efficiency. In [27], the authors use reinforcement learning to determine an optimal arbitrage policy with a battery that does account for non-linear battery degradation. In this study, however, the efficiency calculation is simplified and the effect of degradation on the arbitrage policy is not discussed in depth. Furthermore, the authors consider arbitrage on an hourly time interval in one-market and do not explore how such a model could be applied to real-time operational decision-making.

## 1.4 Contributions

This thesis explores how accounting for degradation in reinforcement learning changes the behavior of the learned policy through operation of a 1MWh battery in NYISO's 2013 Day Ahead Market. We also compare the RL learned policies to the policies determined by the advanced MILP model in [28], both with and without degradation. Finally, we share a case study simulating the use of reinforcement learning to operate a 1 MWh battery in PJM's 2019 real-time and day-ahead markets.

# Chapter 2

# Methodology

Reinforcement Learning is a type of machine learning that allows one to learn action policies through trial-and-error. An agent chooses an action to take knowing only the current state of the environment and its past actions. Depending on how the action affects the environment, the agent receives a reward (or penalty) (Fig 2-1). Over time, the agent learns to optimize its cumulative reward. The interactions between the agent and the environment are modeled after Markov Decision Processes (MDP). Thus, the agent has a a set of actions it can take in the environment, which are defined by the action space. It takes these actions based on the state of the environment, defined by the state space. And based on the state and action, a reward is calculated using the reward function.

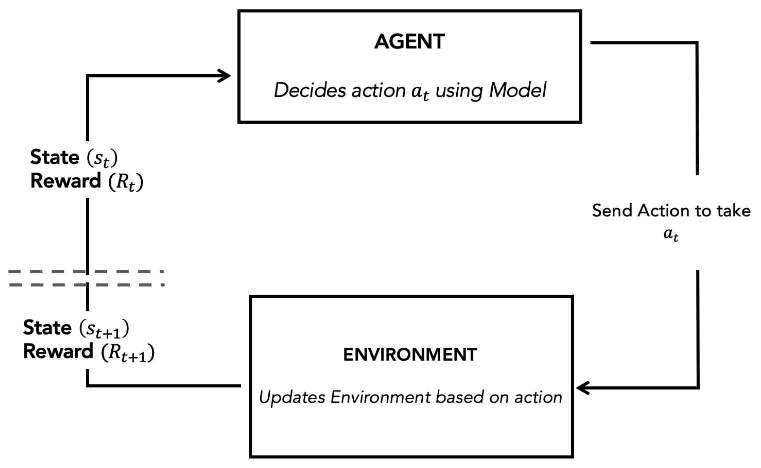
AGENT
Decides action $a_t$ using Model
State ($s_t$)
Reward ($R_t$)
Send Action to take $a_t$
State ($s_{t+1}$)
Reward ($R_{t+1}$)
ENVIRONMENT
Updates Environment based on action

Figure 2-1: Reinforcement Learning Basic Algorithm

## 2.1 Battery Environment

### 2.1.1 State space

The state space is defined using the state of charge of the battery (SoC), the current price, and the forecasted prices up to a specified time horizon. Thus, the state $s_t$ at time $t$ with a future forecast horizon of $h$ is defined as $s_t = (SoC_t, p_t, p_{t+1}, \dots p_{t+h})$, where $p_t$ is the price of energy at time $t$.

The SoC is expressed as a percentage between 0 to 1. In our experiments, we assumed we were using a 1 MWh Li-ion battery. This allowed us to treat $SoC$ as having units MWh. Changes to the SoC are defined as follows:

$$SoC_t = SoC_{t-1} - \frac{1}{SoC_{max,0}} P_t \Delta t \tag{2.1}$$

where $SoC_t$(MWh) is the state of charge at time $t$, $SoC_{max,0}$ (MWh) is the initial maximum state of charge capacity of the battery (Since our battery is a 1MWh (1MW) battery, $SoC_{max,0} = 1$), $\Delta t$ is the time interval (in hours) over which the power is charged or discharged, and $P_t$ (MW) is the power from the power source to the battery at time $t$ after considering losses due to imperfect efficiency. When the battery is charging, $P_t < 0$, and when the battery is discharging $P_t > 0$.

### 2.1.2 Action Space

It was assumed that the battery would not be able to both charge and discharge at the same time. There are five possible actions the battery can take over a given time interval $\Delta T$ [hours]:

$$a \in \{-P_{max}, -0.5P_{max}, 0, 0.5P_{max}, P_{max}\}$$

The action is set to be the intended power $P_t$, which is used to calculate the power before losses, $P_{grid}$. $P_{grid}$ is used in financial transactions. Because the battery has capacity limits, the power $P_t$ is limited as follows:

$$SoC_{min,t} \leq SoC_{t-1} - \frac{1}{SoC_{max,0}} P_t \Delta t \leq SoC_{max,t} \tag{2.2}$$

where $SoC_{max,t}$ and $SoC_{min,t}$ are the maximum and minimum SoC capacities, respectively, at time $t$.

**Efficiency** To calculate pre-losses power, $P_{grid}$, of the battery, we implemented a round-trip efficiency of $\eta$. For a particular charging/discharging power $P_t$:

$$P_{grid} = \begin{cases} P_t * \eta & \text{if } P_t > 0 \text{ (discharging)} \\ \frac{P_t}{\eta} & \text{if } P_t \leq 0 \text{ (charging)} \end{cases} \tag{2.3}$$

**Degradation** Upon taking an action $P_t$, we also decrease $SoC_{max}$ according to our degradation model.

Accounting for battery degradation is crucial in the process of energy arbitrage, since the battery's operating costs mostly stem from degradation. There are two kinds of degradation: calendar-based degradation and cycling degradation. Calendar degradation is the battery's inherent degradation over time due to environmental factors like temperature. Cycling degradation is the amount of capacity lost each time a battery undergoes one charge and discharge cycle. While calendar degradation is mostly fixed, cycling degradation can vary depending on the depth of charge/discharge and the power.

We implemented the degradation model as follows:

First we calculate a depth of discharge, which characterizes the change in SoC, a particular action causes.

$$DOD_t = \frac{|SoC_t - SoC_{t-1}| * 100}{SoC_{max,0}} \tag{2.4}$$

We then use the depth of discharge to calculate the cycle life $N_{cyc}$

$$N_{cyc} = 0.0035 * DOD_t^3 + 0.2215 * DOD_t^2 - 132.29 * DOD_t + 10555 \tag{2.5}$$

This relationship between cycle life and DOD was calculated by fitting a third order polynomial to empirical data of cycle life at different DODs as introduced in [28]. The cycle life is then used to calculate the new, degraded $SoC_{max,t}$ at time step $t$. If the battery is at rest (i.e. $P_t = 0$), then the battery undergoes calendar degradation. We assume that the battery capacity due to calendar aging, declines linearly over time as defined by 2.6.

$$SoC_{max,t} = SoC_{max,t-1} - \frac{\Delta T * EoL * (1-p) * SoC_{max,0}}{\text{Battery Life}} \tag{2.6}$$

where $EoL[\%]$ (End of Life) is defined as the maximum percentage (expressed as a decimal between 0 and 1) of the battery capacity lost before the battery needs to be replaced. For our battery environment, we assumed $EoL = 0.3$. Thus, the 1MWh battery needs to be replaced when it loses $0.3MWh$ of its capacity. $p$ is the ratio between cyclical degradation and calendar degradation. In our case, we assumed an even split between cyclical degradation and calendar degradation, so $p = 0.5$. And both Battery Life and $\Delta T$ are in the same unit, hours. We assumed a battery life of 10 years, so Battery Life $= 365 * 24 * 10$.

If the battery is not at rest (i.e. $P_t \neq 0$), then the battery undergoes cyclical degradation. In which case:

$$SoC_{max,t} = SoC_{max,t-1} - \frac{\Delta T * EoL * (1-p) * |P_t|}{2 * N_{cyc}} \tag{2.7}$$

.

### 2.1.3 Reward Function

The energy arbitrage reward [$] has three components: a revenue component, overshooting penalty, and a degradation penalty.

$$R_t = p_t(\frac{P_t^{int}}{P_{max}}) - BatteryLife * \alpha_{deg} * \frac{l_t}{EoL} - \alpha_{over} * f(P_t^{int}) \tag{2.8}$$

where $f(z)$ is a flag that indicates whether the intended action would cause the battery to exceed its limits.

$$f(z) = \begin{cases} 0 & \text{if } SoC_{min,t} \leq SoC_{t-1} - \frac{1}{SoC_{max,0}} P_t \Delta t \leq SoC_{max,t} \\ 1 & otherwise \end{cases} \tag{2.9}$$

The revenue component, $p_t(\frac{P_t^{int}}{P_{max}})$, captures the revenue from arbitrage. The overshooting penalty is a product of the cost of overshooting, $\alpha_{over} = 10$, and the boolean flag $f(z)$, that indicates whether the given action at time $t$, would cause the battery to exceed its limits. The degradation penalty is expressed as the product between the degradation cost, $\alpha_{deg}$ (\$/MWh-year), $BatteryLife$ (years), and the fraction of end-of-life capacity degradation expended, $\frac{l_t}{EoL} = \frac{SoC_{max,t-1} - SoC_{max,t}}{EoL}$ (MWh), at time step $t$.

## 2.2 Training with Double Dueling Q Network (DDQN)

The reinforcement learning algorithm is one that trains an agent to select actions that maximize its cumulative future reward. Specifically, the agent learns the action-value function, $Q(s, a)$, of taking an action $a$ in state $s$. $Q(s, a)$ is defined as:

$$Q(s, a) = E\left[\sum_{h=0}^{h=H} \gamma^h R_{t+h} | s_t = s, a_t = a\right] \tag{2.10}$$

where $\gamma$ is a discount factor for future rewards received.

The action-value function is learned and continually estimated from experiences the agent has in the environment. At each time step $t$, the agent decides on an action $a_t$ to take. With probability $\epsilon$ the agent decides to take a random action, and all other times it decides to take the optimal action (as dictated by $a_t = \max_a Q(s_t, a)$) for the current state, $s_t$. When the action $a_t$ is taken, the state of the environment changes to a new state, $s_{t+1}$ and a reward $r_t$ is received. This information is used to update the action-value function, $Q(s, a)$, using the Bellman Equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_t - Q(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right] \tag{2.11}$$

where $\alpha$ is the learning rate.

$Q(s, a)$ is iteratively updated until it converges to the best action-value function $Q^*(s, a)$. For small action and state-spaces, the action-value function $Q$, can be approximated with a look-up table (with dimensions, $\#$ *of states* $\times$ $\#$ *of actions*). As the dimension of states and actions increases, however, the Q-learning task becomes more difficult.

To address this Google DeepMind [29] proposed approximating the optimal action-value function $Q^*(s_t, a_t)$ with a deep neural network with weights $\theta$ (i.e. $Q(s_t, a_t|\theta)$). The objective when training the neural network is to minimize the mean squared error between $Q(s_t, a_t|\theta)$ and the target value $[R_t - Q(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a|\theta)]$, as shown in (2.12). We used $\gamma = 0.9999$ to reduce the penalty of waiting for future rewards.

$$min(\left[R_t + \gamma \max_a Q(s_{t+1}, a|\theta)\right]) - Q(s_t, a_t|\theta))^2 \tag{2.12}$$

### 2.2.1 Network Architecture

For training, we used a noisy double dueling Q network as proposed by [27]. In this, we decouple the network used to select the action from the network used to generate the target Q values, as proposed by [30]. The network used to generate the target Q values is called $DQN_{target}$ and the network used to select actions is $DQN_{current}$. The parameters of $DQN_{target}$ are periodically replaced with the parameters from $DQN_{current}$. In other words, $DQN_{target}$ serves as a temporally older version of the current model. Both networks, therefore, have the architecture defined in Figure 2-2, with parameter count and output sizes defined in Table 2.1. The final output size is $(batchsize, 5)$ because the size of our action space is 5.

We decouple the estimation of the action-independent value function from the advantage function, as proposed by [31] by using two paths in parallel within our

$$\begin{tikzcd}
 & \textbf{InputLayer} \arrow[d] & \\
 & \textbf{Linear}_{\textbf{noisy},\textbf{1}} \arrow[d] & \\
 & \textbf{ReLu} \arrow[dl] \arrow[dr] & \\
\textbf{Linear}_{\textbf{noisy},\textbf{value}} \arrow[d] & & \textbf{Linear}_{\textbf{noisy},\textbf{advantage}} \arrow[d] \\
\textbf{ReLu} \arrow[d] & & \textbf{ReLu} \arrow[d] \\
\textbf{Linear}_{\textbf{noisy},\textbf{2}} \arrow[dr] & & \textbf{Linear}_{\textbf{noisy},\textbf{3}} \arrow[dl] \\
 & \textbf{Combination} &
\end{tikzcd}$$

Figure 2-2: Architecture of the networks used to estimate the Q action-value function.

network architecture (one path for the value function $Val(s)$ and another for the advantage function $Adv(s, a, \theta)$). The goal of the advantage function is to estimate the value of taking an action in a particular state, whereas the goal of the value function is to estimate the value of being in a particular state. The outputs of these two parallel paths are then combined to approximate the Q-function as follows:

$$Q(s, a) = Val(s) + Adv(s, a, \theta) \tag{2.13}$$

Table 2.1: Output Size and Parameter Count for each layer in network

| Layer Name | Output Size | Parameters |
|---|---|---|
| **InputLayer** | (batchsize, 25) | 0 |
| **Linear**$_{\textbf{noisy},\textbf{1}}$ | (batchsize, 16) | 400 |
| **Linear**$_{\textbf{noisy},\textbf{value}}$ | (batchsize, 16) | 256 |
| **Linear**$_{\textbf{noisy},\textbf{advantage}}$ | (batchsize, 16) | 256 |
| **Linear**$_{\textbf{noisy},\textbf{2}}$ | (batchsize, 1) | 16 |
| **Linear**$_{\textbf{noisy},\textbf{3}}$ | (batchsize, 5) | 80 |
| **Combination** | (batchsize, 5) | 0 |

The standard fully-connected linear layers are also replaced with noisy linear layers as proposed by [32]. A noisy linear layer is defined as:

$$y = (\mu^w + W^w \odot \sigma^w)x + (\mu^b + W^b \odot \sigma^b) \tag{2.14}$$

where $\mu^w, W^w, \mu^b, W^b$ are learned parameters of the networks and both $\sigma^w$ and $\sigma^b$ are randomly sampled, zero mean noise matrices with fixed statistics (in our case, sampled from a normal distribution with a standard deviation of 0.017).

## 2.2.2 Training algorithm

We modified the training algorithm to use a linearly decreasing epsilon-greedy function for choosing an action, as proposed by [27] to improve convergence time to a solution. We also changed the initialization of the battery to randomly select a starting SoC of either 0, 0.5, or 1. The details of the algorithm implementation are shown in Algorithm 1, and all training processes were implemented using open-source machine learning framework, PyTorch [33] and the environment was designed using OpenAI Gym [34]. For all models trained, we used Pytorch's implementation of the Adam optimizer with a learning rate of 0.00025, where the learning rate is $\alpha$ from Eqn.2.11. For the epsilon-greedy action selection, we chose an initial $\epsilon = 0.8$ and linearly scaled it down to $\epsilon_{min} = 0.001$ using

$$\epsilon = min(\epsilon_{min}, \epsilon - \frac{c}{num\_eps}\epsilon) \tag{2.15}$$

where c is a constant used to scale the speed of linear decline. We tried $c \in [0, 10]$ and empirically found that $c = 3$ gave the best results with the most consistency. The training settings used are summarized in Table 2.2.

## 2.2.3 Utilizing the trained model

To use the trained model, we use the training algorithm (Algorithm 1), but remove steps that update the model. The process is explained in detail in Algorithm 2.

**Algorithm 1:** Training NN-DDQN for Energy Arbitrage

```
1  Initialize the noise matrices $\sigma^b$ and $\sigma^w$;
2  Initialize the current network ($DQN_{current}$) and target ($DQN_{target}$) network
     parameters;
3  Initialize Memory, $M$ and mini-batch size;
4  Initialize $\epsilon$ for $\epsilon$-greedy policy;
5  for Episode $e = 1$ to $num\_eps$ do
6      Reset battery and Initialize battery's SoC as either 0,0.5, or 1 (randomly);
7      Observe the state of the environment $s_t = (SoC_t, p_t, p_{t+1}, \dots p_{t+h})$ ;
8      for $t = 1$ to $T$ do
9          Resample zero mean noise matrices $\sigma^b$ and $\sigma^w$;
10         With probability $\epsilon$, select a random action $a_t$ else select action
             $a_t = argmax_a(DQN_{current}(s_t, a))$ ;
11         Execute action $a_t$ in the environment to receive reward $r_t$ and next
             state $s_{t+1}$ ;
12         Store the transition $(s_t, a_t, r_t, s_{t+1})$ as an experience in $M$ ;
13         Sample a random mini-batch of experiences from $M$ of the form
             $(s_k, a_k, r_k, s_{k+1})$;
14         Resample zero mean noise matrices $\sigma^b$ and $\sigma^w$;
15         Estimate the target
             $y_k = [R_k + \gamma * DQN_{current}(s_{k+1}, \max_a DQN_{target}(s_{k+1}, a))]$
16         Do gradient descent with loss $(y_k - Q(s_k, a_k))^2$
17         Every $C$ steps update $\theta_{target}$ parameters $= \theta_{current}$ parameters
18     end
19     Update $\epsilon = min(\epsilon_{min}, \epsilon - \frac{3}{num\_eps}\epsilon)$
20 end
```

Table 2.2: Training Setting Hyperparameters

| Name | Value |
|---|---|
| Number of nodes in each layer | 16 |
| Learning rate | 0.00025 |
| Optimizer | Adam optimizer |
| batchsize | 32 |
| initial $\epsilon$ | 0.8 |
| $\epsilon_{min}$ | 0.001 |
| Target Model update (C) | 1000 |
| Number of Epochs | 5000 |

**Algorithm 2:** Using NN-DDQN for Energy Arbitrage

```
1 Load the network ($DQN$);
2 Reset battery and Initialize battery's SoC to 0 (or SoC required for
    comparison to MILP model). ;
3 Observe the state of the environment $s_t = (SoC_t, p_t, p_{t+1}, \dots p_{t+h})$ ;
4 for $t$ = 1 to $T$ do
5     Select action $a_t = argmax_a(DQN(s_t, a))$ ;
6     Execute action $a_t$ in the environment and receive next state $s_{t+1}$ (no
        reward, $r_t$ needed during testing) ;
7 end
```

# Chapter 3

# Evaluating Reinforcement Learning Model on Day Ahead Market

We trained reinforcement learning (RL) models both with and without degradation to operate on the NYISO 2013 wholesale day-ahead electricity prices at a one-hour resolution. The energy storage asset is a 1MWh, 1MW lithium ion battery with a roundtrip efficiency of 90% and an assumed annualized degradation penalty, ($\alpha_{deg}$), of $20,000[1]. The models were trained on one week of data and tested on the following week of data. Specifically, we trained a model on the first week of January and the first week in July to explore the model's performance in both the winter and summer. Thus, we ultimately present results of four scenarios.

1. Winter Scenario with $20k degradation
2. Winter Scenario with $0k degradation (no degradation penalty)
3. Summer Scenario with $20k degradation
4. Summer Scenario with $0k degradation (no degradation penalty)

The policies were evaluated against mixed-integer-linear programming (MILP) models with a round-trip efficiency of 90% and degradation penalties of $20,000 and $0.

[1]The value $20000 was calculated as an estimated annualized cost of replacing the battery after the battery's lifespan of 10 years. The total replacement cost of the energy component of the battery was calculated to be approximately $1MWh \times \$200/kWh = \$200,000$ (For cost basis see [3])

## 3.1 Method of Comparison between MILP and RL policies

The key differences between the MILP model and RL model are:

1. MILP models have a continuous action space, where the power charged or discharged can be any number between $[0, 1]$. As discussed in section 2.1.2, the reinforcement learning model only has 5 discrete actions it can take.

2. The reward function of the reinforcement learning algorithm includes and overshooting penalty which is unnecessary to include in the MILP objective function since the MILP model includes a constraint that prevents overshooting. The MILP objective function also includes an operations and maintenance cost on the battery which is negligible when compared to the cost of degradation. In all other aspects, the reward function of the reinforcement learning algorithm, which is being maximized, and the objective function of the MILP model (also being maximized), are comparable since both are mainly comprised of a revenue reward and a degradation penalty.

3. The MILP model uses a linearized form of the non-linear degradation model used in the Reinforcement learning model.

To minimize any potential effects of these differences, we used the reinforcement learning battery environment to calculate the effects of taking the actions dictated by the MILP policies rather than using the revenues and calculated battery degradation output by the MILP model.

## 3.2 MILP Policy vs Reinforcement Learning Policy

We compare the policies generated by MILP and Reinforcement Learning in three aspects: battery degradation, revenue, and the qualitative characteristics of cycling.

### 3.2.1 Degradation

We characterize the degradation of the battery through capacity fade (i.e. the slow decrease in the maximum energy capacity of the battery over time). In Fig 3-1, we present the capacity fade that results from using Reinforcement Learning policies and MILP policies over the first two weeks of January (winter) and the first two weeks of July (summer). The first week of price data in both the winter and summer scenarios was used to train the agent. In the winter, the difference in capacity fade between the RL degradation scenario and the no degradation scenario is 0.01%. This corresponds to a degradation penalty of \$66.67[2]. Thus, accounting for degradation allows us to save \$66 in degradation costs. The comparison for winter 0-degradation penalty scenario is the only comparison with a clear discrepancy between MILP and RL of 0.02%. Even in this case, however, reinforcement learning learns to degrade less than MILP. These discrepancies may reduce when provided with more training data to allow for more robustness to diverse price signals.

### 3.2.2 Revenue

The reinforcement learning algorithm is able to learn the general charging and discharging strategy of buying energy when the price is low and selling when the price is high (Fig 3-2, Fig 3-3), as is expected and comparable to MILP's general strategy. In both MILP and reinforcement learning we see the battery cycling about once a day (if it cycles at all).

The reinforcement learning model is able to generate comparable revenues. In the worst case, the RL model generates 86% of the MILP's revenue (summer scenario no degradation). In the best scenario (winter scenario degradation), the RL model generates 102% of the MILP's revenue, although this comes at a higher degradation cost (as seen in Fig 3-1). In the other two scenarios the RL model is able to generate on average, 96% of the revenue made by the MILP model.

[2] $BatteryLife * \alpha_{deg} * \frac{l_t}{EoL}$ =10 years* 20000 \$/MWh-year * 0.0001 MWh/0.3 MWh = \$66.67

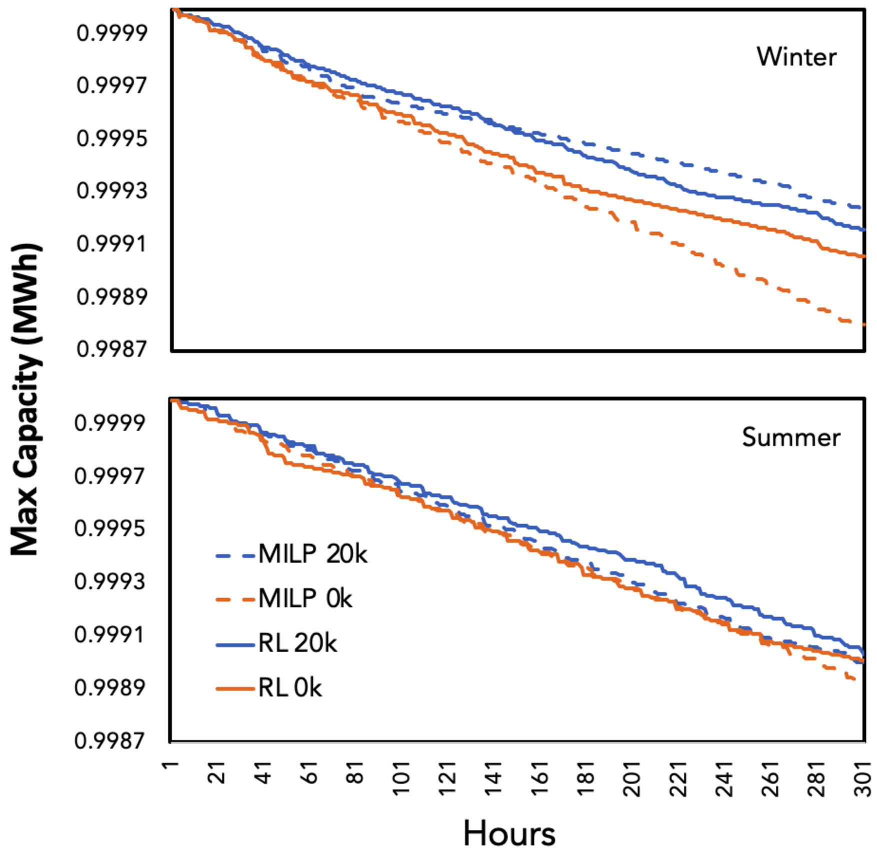

Figure 3-1: Battery Capacity Fade using MILP and Reinforcement Learning

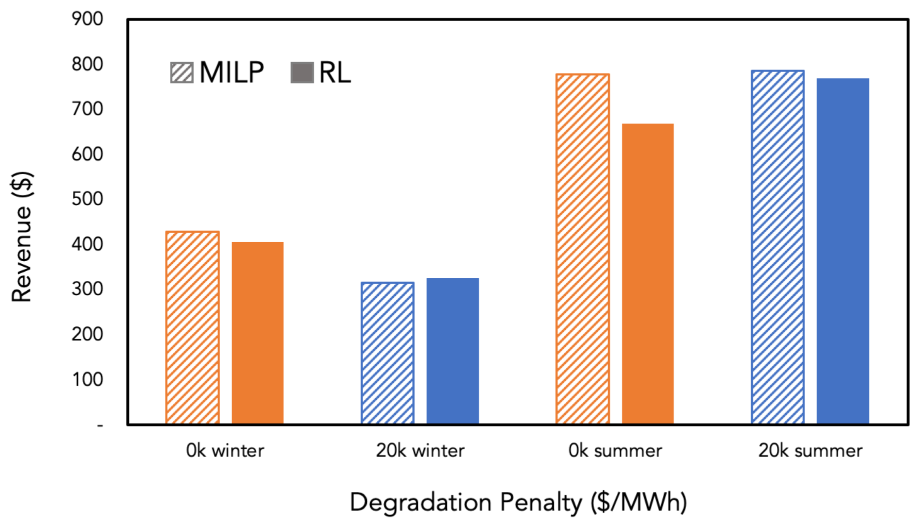

Figure 3-4: Comparison of MILP Revenues with Reinforcement Learning Revenues. Initial condition of summer degradation scenario differs from initial condition of other scenarios.

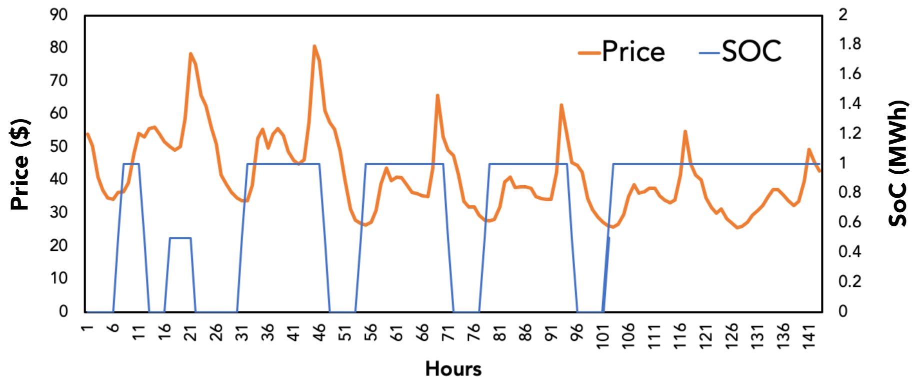

Figure 3-2: One week of cycling in the winter scenario with degradation. SOC equivalent to battery capacity in MWh when using 1 MWh battery.

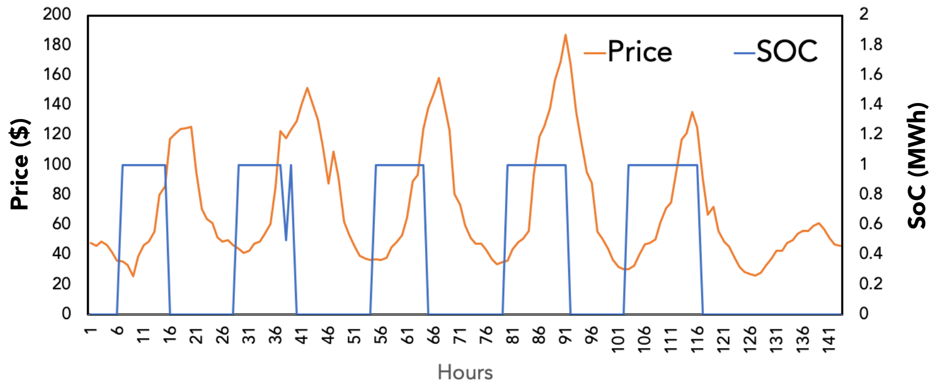

Figure 3-3: One week of cycling in the summer scenario with no degradation. SOC equivalent to battery capacity in MWh when using 1 MWh battery.

### 3.2.3 Motivation and Justification for Case Study in Real-Time Market

MILP still outperforms reinforcement learning in all cases. However, reinforcement learning shows promise. By accounting for degradation, we see that reinforcement learning is able to change policy to reduce capacity fade. We also see that reinforcement learning in most cases, is able to generate most of the revenue possible by MILP. Given more computational power and time to better tune the hyperparameters of the model and training process (as listed in 2.2), the discrepancy between MILP and reinforcement learning can be decreased. In particular, by training the model on a longer and more diverse price signal (e.g. an entire year of price data), we would expect the performance of the model to dramatically improve. Having used the day-ahead market as a test-bed to assess whether reinforcement learning is capable of changing its policies to account for degradation in the expected manner, we were motivated to use similar approaches on the Real-Time Market. This motivation stems from the fact that the real-time market requires quick operational decision making, on the order of 5 minutes. Thus, traditional approaches to this problem such as using MILP models, are too computationally intensive to run when also accounting for degradation.

# Chapter 4

# Case Study: Using Reinforcement Learning to Operate a battery in Real Time

## 4.1 Methodology Changes for Real-Time Market

Once we showed that reinforcement learning was able to generate comparable results with MILP with similar degradation trends (i.e. accounting for degradation reduces capacity fade and can make most of the revenue) with the 2013 NYISO day ahead market, we explored the use of reinforcement learning on a 5-minute interval real-time market. For the case study, we used prices from PJM's 2019 dataset and chose to train and test our reinforcement learning model on 8 days in the winter and 8 days in the summer. Specifically, we train on the first 4 days in a given set of days, and evaluate the model on the last 4 days.

To modify the existing model training framework from day-ahead market use to real-time market use, we made the following changes.

1. Real Time prices are very uncertain. We assumed that the forecast of prices over the next hour is always close to true prices. Thus, we used a perfect forecast of 12 real-time prices (i.e. prices over the course of the next hour) in

our environment state as opposed to the 24 hour future prices provided in the day ahead market. This modifies our state vector to be a vector of length 13, $s_t = [SoC, p_t, \ldots, p_{t+12}]$.

2. Batteries often will participate in the day-ahead market and use the remaining capacity for the real-time market. To account for this, we used the MILP model to generate day ahead optimal bids and offers with PJM's 2019 day-ahead market prices. At every time-step, we calculate the state of charge available for use in the real-time market, and provide this adjusted state of charge to the agent when decision making.

3. We continue to assume that our battery is a 1MWh battery, but we assume that since it is operating on a smaller time-scale, it has more robust power control infrastructure, and thus we set the maximum power, $P_{max}$ to 2MW as opposed to 1MW, as was the case in the day-ahead comparison with MILP.

4. Since the revenues generated on a single time step in the real-time market are about 12 times smaller than those generated in the day-ahead market, we modified the overshooting penalty cost from $\alpha_{over} = 10$ to $\alpha_{over} = 1$. This is done to prevent the overshooting penalty from overwhelming other components of the reward function.

5. Because there is more uncertainty in the prices of the real-time market, we incentivize present rewards more than future rewards (since future rewards have high uncertainty depending on the prices). We do this by decreasing $\gamma$ (the discount factor as mentioned in Algorithm 2) to 0.95.

All other aspects of the training process and infrastructure remained identical to the process previously discussed in Section 2.2.

## 4.2 Degradation, Revenues, and Cycling Behavior

In Fig 4-1, we show the isolated effect of real-time market arbitrage on the capacity of the battery over 8 winter days (i.e. the figure does not include any degradation that stems from taking actions in the day-ahead market). As expected, the model that accounts for degradation with a $20,000 annualized penalty (correlates to a total battery replacement cost of approximately $200,000 in 10 years), degrades 0.02% less than the model that does not account for degradation. This 0.02% capacity fade results in a degradation cost difference of $133 according to our model's degradation penalty calculation as discussed in section 2.1.3. When we include degradation from the day-ahead market as well, we see a capacity fade difference of 0.06%, which is equivalent to a degradation cost difference of $400.

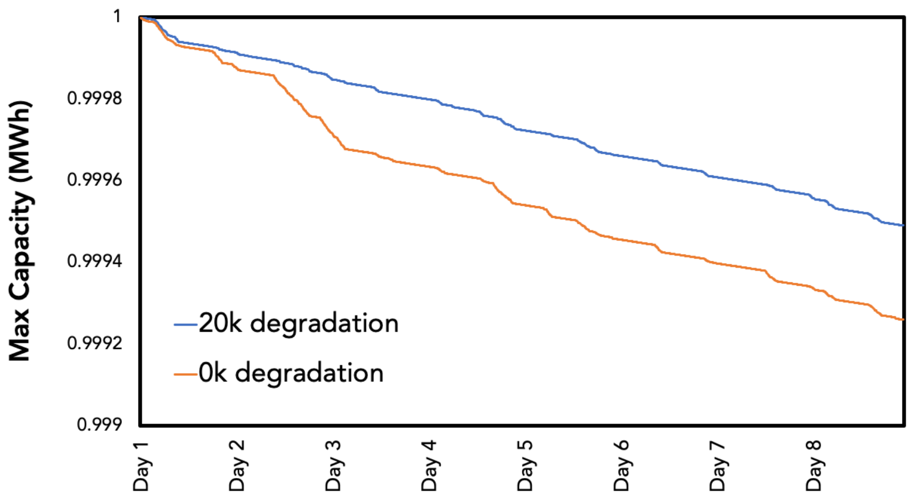

Figure 4-1: Degradation from Real-Time Market Arbitrage, Winter days

The difference in revenue, however, between the real-time market profits two scenarios is smaller than the difference in their degradation costs. In the case where we account for degradation, the optimal policy is to not cycle in the day-ahead market at all. The total revenue from both the real-time and day-ahead markets when accounting for degradation is $75, all from real-time market arbitrage. When we do not account for degradation, the model acquires a revenue of approximately $150, where $103 comes from real-time market arbitrage and the remaining amount ($47) comes from the day-ahead market arbitrage. Although the model accounting for degradation makes \$75 less in revenue, it also saves a degradation cost of \$133, or if including degradation from day-ahead arbitrage, a cost of \$400, as compared to the other model. It is therefore, more profitable than not-accounting for degradation, in which, the additional costs of degradation significantly outweigh the gain in revenue from the additional cycling. In both cases, however, regardless of whether degradation is accounted for the net profit is negative after accounting for degradation costs and less money is lost if the battery rests. The battery at rest loses, in total, about 0.028% of its max capacity, and makes no money in revenue. The total profit is just the degradation cost of -\$190. Whereas, in the scenario with degradation, the model causes the battery to lose 0.05% of the max capacity, which translates to a total profit of -\$258 (degradation cost of -\$333 offset by \$75 in revenue from real time market). In the scenario without degradation, the total profit is -\$363 ( degradation cost of -\$466 offset by \$103 in revenue from real-time market) when ignoring degradation and revenue from day-ahead arbitrage actions and -\$583 (degradation cost of -\$733 offset by \$150 revenue from real-time and day-ahead markets) when including degradation costs associated with day-ahead arbitrage actions.

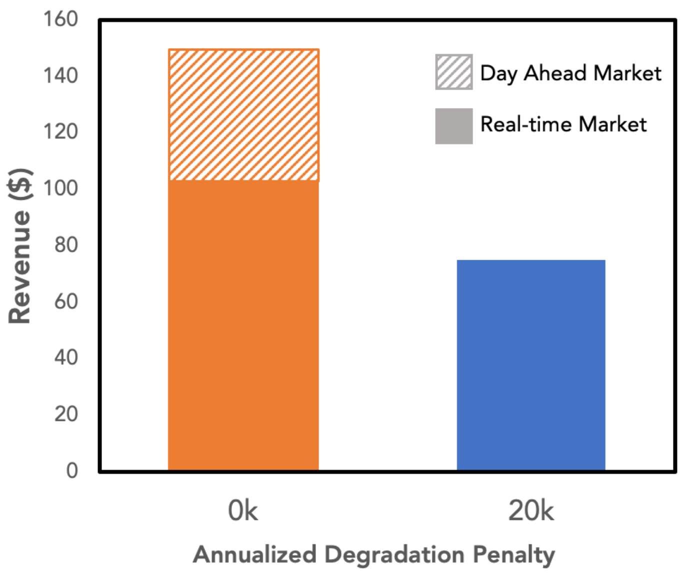

Figure 4-2: Revenue from Reinforcement Learning Model Policy in Real Time and Day Ahead Markets, over 8 winter days.

Thus, we see that accounting for degradation is significant when using reinforcement learning for energy arbitrage. Even though in all cases, the profit is negative, when accounting for degradation, the battery makes a smaller loss. The model is still not optimal, since, as we can see, resting would have lost significantly less money.

When we performed a similar study over 8 summer days, our results were different. We saw that the results from the scenario with degradation and without degradation were comparable. Although we cannot make a direct comparison between the performance of the MILP model in the day-ahead market of 2013 NYISO summer pricing and the performance of reinforcement learning in the real-time market of 2019 PJM summer pricing, it is interesting to note that in both cases, the summer scenarios did not differ significantly when accounting for degradation. The discrepancy in revenue for the real time market PJM summer scenario is around \$5 (No degradation - \$705 revenue, degradation - \$699 revenue), and the difference in degradation is 0.005% where the scenario that accounts for degradation degrades slightly more. Based on the MILP model and RL model comparison in the day-ahead market, we expect models that account for degradation to degrade less by at least around 0.01%. The reason for this result which does not match expectation, we hypothesize, is due to the lower $\gamma = 0.95$. By incentivizing present rewards more, but also placing a penalty on degradation (including calendar degradation), the model seeks to cut losses from calendar degradation by generating revenue. However, because the $\gamma$ is lower and future rewards are highly discounted, the model is not incentivized to rest. The model is even more incentivized to avoid resting because when there is a degradation penalty, resting receives a negative reward. When there is no degradation penalty, however, the model does not receive negative reward for resting (i.e. it does not account for losses from calendar degradation), and thus is less incentivized to cycle unnecessarily and can rest, which results in lower capacity fade overall. This same logic does not seem to apply to the winter prices because with the winter price signal we used, the opportunity for arbitrage did not justify the degradation cost because differences in prices were minimal over time. In the winter the penalty for resting is clearly more advantageous than the penalty and revenue received through cycling. Thus, accounting for degradation has the expected effect on cycling, in that the winter-trained model with degradation clearly cycles less often than the winter, no-degradation penalty model (Fig 4-3). In the summer price signal, however, the opportunity for arbitrage was larger, and the penalty for resting is sometimes worse than the penalty for cycling (offset with revenue), so the model has more incentive to cycle.

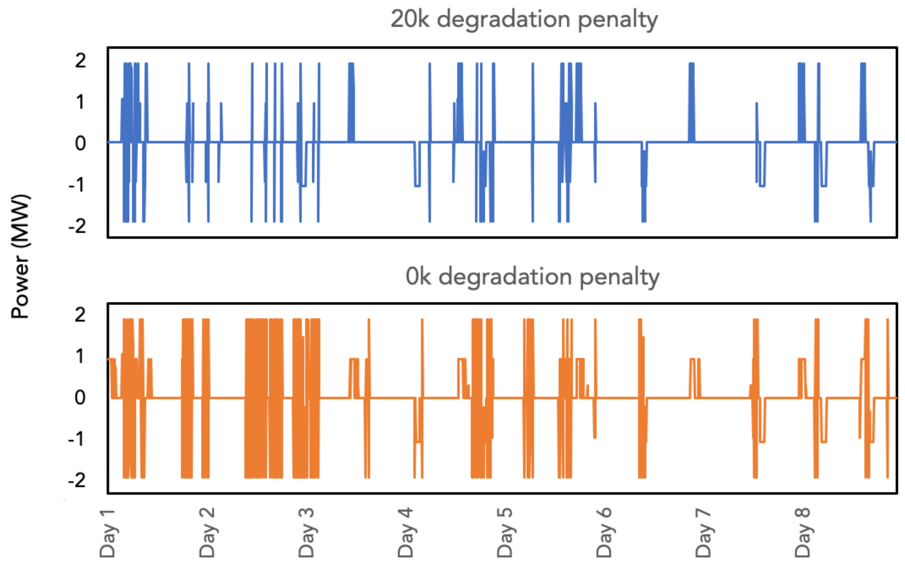

Figure 4-3: Difference in Cycling over 8 winters days with and without degradation

# Chapter 5

# Conclusions

Accounting for battery degradation when controlling a battery can increase profits by cycling wisely and reducing degradation costs. However, accounting for degradation with traditional methods as part of a mixed integer linear programming model, leads to large models that have long computation times. Reinforcement Learning can be used to develop models that can determine close-to-optimal policies in a matter of milliseconds. This thesis explores the use of reinforcement learning (RL) to train models that can control a battery for energy arbitrage while accounting for degradation costs. As is always the case with machine learning, a key challenge is training the model to achieve solutions that are close to the optimal solution. In this study, we evaluate the RL model's performance against the performance of an MILP model with equivalent degradation and power dynamics. We find that, although RL is unable to find the optimal solution, the changes in the policy that stem from different degradation penalties are comparable to the changes we see between MILP models that account for degradation and do not account for degradation. Accounting for degradation in the day-ahead market and real-time market, in most cases, resulted in less capacity fade by about 0.01%. RL is prone to exploiting loopholes in the environment to achieve high rewards, but in our scenarios, we found that the reduction in capacity fade was a result of less cycling, which is the correct learned policy. We therefore show that using RL to build a model that can operate a battery while accounting for degradation, has potential. There are many ways the performance of Rl models that account for degradation can be improved through future work. In this study we trained, in the longest instance, on 1 week of price data due to computing power limitations. Increasing the amount of price data the RL model has access to and training the model on a full year or multiple years of data will allow the agent to learn more generalizable policies that are robust to different kinds of price signals. Performance can also be improved by differentiating between the forecasted price and the true price. In our study we always assumed a perfect forecast, but in reality, forecasts are far from perfect.To account for this, a longer study may involve calculating the optimal action policy using MILP and RL with forecasted data (as one would do in real life), and then apply the action policy to real price data. In this case, we can train the reinforcement learning algorithm to be robust to differences b etween the forecasted price and the real price. An alternative approach to using RL that could be explored is allowing the reinforcement learning agent to use a pre-calculated MILP action policy result. The agent would be able to determine when to follow the MILP action and when to do something different. F uture w ork c ould a lso g o b eyond the scope of energy arbitrage. Because it is easy to implement complicated environments with RL, one could easily explore other battery operation problems by allowing the environment to participate in different m arkets, b e r ewarded w ith c omplicated tariff structures, account for different battery chemistries, and interact with energy sources. This thesis illustrates that using of reinforcement learning to build models that can operate a battery has promise. Reinforcement learning models do not always find the optimal solution, but they can compute good policies quickly, can appropriately account for complicated environmental dynamics like degradation, and with more training time and improvements to model robustness, they show potential as models that can be used in real-life applications to efficiently and effectively operate batteries.

# Bibliography

[1] Inventory of u.s. greenhouse gas emissions and sinks, Apr 2021.

[2] Mehdi Jafari, Magnus Korpås, and Audun Botterud. Power system decarbonization: Impacts of energy storage duration and interannual renewables variability. *Renewable Energy*, 156:1171–1185, 2020.

[3] A. Will Frazier Wesley Cole. Cost projections for utility-scale battery storage, 2019. NREL/TP-6A20-73222.

[4] Muhammad Sufyan, Nasrudin Abd Rahim, Muhammad Aman, Chia Tan, and Siti Raihan. Sizing and applications of battery energy storage technologies in smart grid system: A review. *Journal of Renewable and Sustainable Energy*, 11:014105, 02 2019.

[5] P Gardner. E-storage: Shifting from cost to value wind and solar applications.

[6] J. Hu, R. Harmsen, Wina Crijns-Graus, E. Worrell, and M. Broek. Identifying barriers to large-scale integration of variable renewable electricity into the electricity market : A literature review of market design. *Renewable & Sustainable Energy Reviews*, 81:2181–2195, 2018.

[7] Michael Milligan, Bethany A. Frew, Aaron Bloom, Erik Ela, Audun Botterud, Aaron Townsend, and Todd Levin. Wholesale electricity market design with increasing levels of renewable generation: Revenue sufficiency and long-term reliability. *The Electricity Journal*, 29(2):26–38, 2016.

[8] Increasing space granularity in electricity markets.

[9] Monica Giulietti, Luigi Grossi, Elisa Trujillo-Baute, and Michael Waterson. Analyzing the potential economic value of energy storage. *The Energy Journal*, 39, 09 2018.

[10] Felix Keck, Manfred Lenzen, Anthony Vassallo, and Mengyu LI. The impact of battery energy storage for renewable energy power grids in australia. *Energy*, 173, 02 2019.

[11] Sheikh Jakir Hossain, Biswajit Dipan Biswas, Rojan Bhattarai, Muhammad Ahmed, Sherif Abdelrazek, and Sukumar Kamalasadan. Operational value based

energy storage management for photo-voltaic(pv) integrated active power distribution systems. *IEEE Transactions on Industry Applications*, PP:1–1, 05 2019.

[12] Jonathan Ogland-Hand, Jeffrey Bielicki, Yaoping Wang, Benjamin Adams, Thomas Buscheck, and Martin Saar. The value of bulk energy storage for reducing co2 emissions and water requirements from regional electricity systems. *Energy Conversion and Management*, 181:674–685, 02 2019.

[13] Micah Ziegler, Joshua Mueller, Gonçalo Pereira, Juhyun Song, Marco Ferrara, Yet-Ming Chiang, and Jessika Trancik. Storage requirements and costs of shaping renewable energy toward grid decarbonization. *Joule*, 3, 08 2019.

[14] Fernando J. de Sisternes, Jesse D. Jenkins, and Audun Botterud. The value of energy storage in decarbonizing the electricity sector. *Applied Energy*, 175:368 – 379, 2016.

[15] M. B. C. Salles, M. J. Aziz, and W. W. Hogan. Potential arbitrage revenue of energy storage systems in pjm during 2014. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, 2016.

[16] D. Krishnamurthy, C. Uckun, Z. Zhou, P. R. Thimmapuram, and A. Botterud. Energy storage arbitrage under day-ahead and real-time price uncertainty. *IEEE Transactions on Power Systems*, 33(1):84–93, 2018.

[17] Y. Wang, Y. Dvorkin, R. Fernández-Blanco, B. Xu, T. Qiu, and D. S. Kirschen. Look-ahead bidding strategy for energy storage. *IEEE Transactions on Sustainable Energy*, 8(3):1106–1117, 2017.

[18] Apurba Sakti, Kevin G. Gallagher, Nestor Sepulveda, Canan Uckun, Claudio Vergara, Fernando J. de Sisternes, Dennis W. Dees, and Audun Botterud. Enhanced representations of lithium-ion batteries in power systems models and their effect on the valuation of energy arbitrage applications. *Journal of Power Sources*, 342:279 – 291, 2017.

[19] Arpit Maheshwari, Nikolaos G. Paterakis, Massimo Santarelli, and Madeleine Gibescu. Optimizing the operation of energy storage using a non-linear lithium-ion battery degradation model. *Applied Energy*, 261:114360, 2020.

[20] Philip Odonkor and Kemper Lewis. Designing optimal arbitrage policies for distributed energy systems in building clusters using reinforcement learning. 08 2019.

[21] Philip Odonkor and Kemper Lewis. Automated design of energy efficient control strategies for building clusters using reinforcement learning. *Journal of Mechanical Design*, 141, 10 2018.

[22] Elena Mocanu, Decebal Mocanu, Phuong Nguyen, Antonio Liotta, Michael Webber, Madeleine Gibescu, and J. Slootweg. On-line building energy optimization

using deep reinforcement learning. *IEEE Transactions on Smart Grid*, PP, 07 2017.

[23] Philip Odonkor and Kemper Lewis. Control of shared energy storage assets within building clusters using reinforcement learning. 09 2018.

[24] Hao Wang and Baosen Zhang. Energy storage arbitrage in real-time markets via reinforcement learning. pages 1–5, 08 2018.

[25] Fiodar Kazhamiaka, Srinivasan Keshav, and Catherine Rosenberg. Adaptive battery control with neural networks. pages 536–543, 06 2019.

[26] Ramachandra Kolluri and Julian de Hoog. Adaptive control using machine learning for distributed storage in microgrids. pages 509–515, 06 2020.

[27] Jun Cao, Dan Harrold, Zhong Fan, Thomas Morstyn, David Healey, and Kang Li. Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model. *IEEE Transactions on Smart Grid*, PP:1–1, 04 2020.

[28] Mehdi Jafari, Audun Botterud, and Apurba Sakti. Estimating revenues from offshore wind-storage systems: The importance of advanced battery models. *Applied Energy*, 276:115417, 2020.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[30] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. 2015. cite arxiv:1509.06461Comment: AAAI 2016.

[31] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

[32] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. *CoRR*, abs/1706.10295, 2017.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc,

E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.