46765 Machine Learning for Energy Systems

Assignment 2: Real-time control of a battery
Code: `https://github.com/albertkjoller/ml-energy-systems`

Albert Kjøller Jacobsen (s194253)
Ilias Manolakis (s222834)
Tais Abrahamsson (s193900)
Denisa Enache (s222964)
Andrei Lavric (s222965)

10th of November, 2023

Contents

Contribution table

	Model development	**Coding**	**Results analysis**	**Report writing**
Albert	20%	35%	40%	30%
Ilias	20%	10%	15%	10%
Tais	20%	15%	15%	15%
Denisa	20%	25%	15%	20%
Andrei	20%	15%	15%	20%

1 Introduction

This project report examines the use of reinforcement learning for learning how to control an electrical battery to execute storage arbitrage by acting on a real-time price signal (the spot price of electricity in DKK). As stated in the assignment description, the battery owner (or *agent*) has access to the price for the coming hour as well as the current state-of-charge (SOC) of the battery from which the goal is to learn what action to take. The battery has a maximum storage capacity of 500 MWh and a maximum charging capacity of ±100 MW per hour. We assume familiarity with the data set containing electricity prices and its' processing as presented in Assignment 1.

For clarity, this assignment considers a battery located in the DK1 bidding zone. Thus the price signal - denoted by λ - is restricted to the spot price time series for this area. The time series is visualized in Figure 1.1.

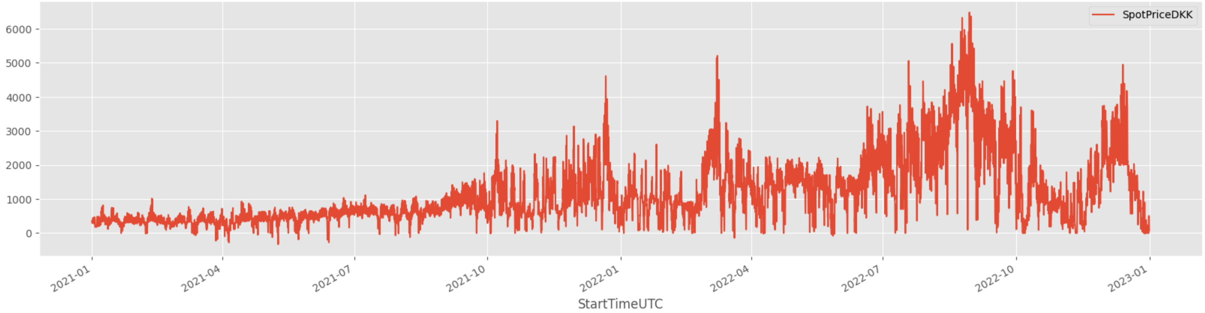


Figure 1.1: The spot price in DKK for the time period ranging from the 1st of January 2021 to the 1st of January 2023. It is evident that the spot price is a non-stationary signal for which reason care needs to be taken in the learning process, since the underlying model should be able to capture the dynamic changes of the system.

1.1 Formulating the problem as a Markov Decision Process

A Markov Decision Process (MDP) is a control process defined in discrete time for decision making in environments that are partly stochastic and partly controlled by actions taken by the decision maker. A MDP is defined by a state space, $\mathcal{S}$, an action space, $\mathcal{A}$, a reward function, $R(s,a)$ and state transition functions, $P_a(s,s')$, determining the probability of going from state s to s' when taking action a. Solving the MDP then comes down to finding the optimal policy, $\pi(s)$, that maps from states to actions.

1.1.1 Defining the state space

By nature both the SOC of the battery and the spot price, λ, are discrete-time signals as electricity is traded hourly. Thus, a natural way of formulating a state is;

$$\boldsymbol{s}_t = \begin{bmatrix}\text{SOC}_t & \lambda_t\end{bmatrix}^T$$

where t denotes the hour from the initially considered point in time. Since the SOC is bounded by $[0, 500]$ MWh and the spot price (i.e. λ) is unbounded the current formulation introduces an infinite and continous state space as the SOC and price can be combined in infinitely many ways. To simplify the problem at hand, a discretization of the state space is carried out. The underlying approach is to discretize the SOC into levels of 100 while binarizing the price signal into a high/low price indication signal. Thus, the state space considered can be expressed as:

$$\begin{aligned}\text{SOC} &= \{0, 100, 200, 300, 400, 500\}\\ I^\lambda &= \{H, L\}\end{aligned}$$

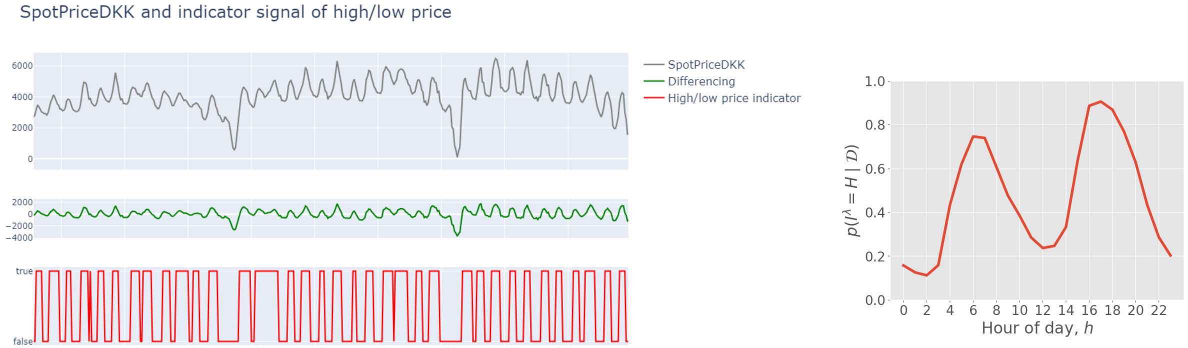
$$\begin{aligned}\mathcal{S} = \text{SOC} \times \lambda &= \left\{\begin{bmatrix}c & p\end{bmatrix}^T \mid c \in \text{SOC}, p \in I^\lambda\right\}\\ &= \left\{\begin{bmatrix}0\\H\end{bmatrix}, \begin{bmatrix}0\\L\end{bmatrix}, \begin{bmatrix}100\\H\end{bmatrix}, \begin{bmatrix}100\\L\end{bmatrix}, \dots, \begin{bmatrix}500\\H\end{bmatrix}, \begin{bmatrix}500\\L\end{bmatrix}\right\}\end{aligned}$$

where × denotes the Cartesian product of the sets. Thus, $|\mathcal{S}| = |\text{SOC}| \cdot |I^\lambda| = 6 \cdot 2 = 12$ possible states.

As presented in Figure 1.1, the electricity price signal is non-stationary and as such, the concept of high and low price should also evolve over time. Thus, discretizing the price requires a bit more thought than what was needed for the SOC. In general the spot price seems to increase across time while a daily periodic pattern that is higher at peak hours and lower at night can also be observed. As such the high/low price indicator signal will be defined by differencing the signal with the average price over a 48-hour window placed with the coming hour at the center. Choosing a 48-hour window is due to the fact that a high/low price should both be defined from past and future prices while assuming that we are restricted to obtaining price information from the day-ahead market only. The indicator signal can thus be defined as:

$$I_t^\lambda = \begin{cases} H & \text{if } \left(\lambda_{t+1} - \frac{1}{48}\sum_{i=1}^{24}\left[\lambda_{t-i+1} + \lambda_{t+i+1}\right]\right) \geq 0 \\ L & \text{otherwise} \end{cases} \tag{1.1}$$

Figure 1.2a illustrates a limited sequence of the price signal, it's 48-hour averagely differenced version and the discretized price signal. In addition, Figure 1.2b reveals that this discretization procedure seem to capture a daily pattern where the spot price is labelled as *high* during peak load hours (i.e. where the electricity consumption is also high) and *low* during the night.



(a) Discretization procedure for the price signal. *Top:* a sequence of the spot price signal for DK1. *Mid:* the 48-hour average differenced version of the price signal. *Bottom:* price level indicator signal where high means that the current spot price (top) is higher than the differenced price signal (mid).

(b) Probability of the price being high as a function of the hour of a day. It appears that prices tend to be higher around peak load hours and lower during the night.

Figure 1.2: Spot price discretization and analysis.

1.1.2 Defining the action space

For simplicity a discrete action space is considered. This is constituted by a 100 MW *charge* action, C, a 100 MW *disharge* action, D, and a *no-op* action, W, that essentially is to wait and do nothing. The action space and action functions are chosen as:

$$\mathcal{A} = \{C(\cdot), D(\cdot), W(\cdot)\}$$

$$C(\boldsymbol{s}_t) = \begin{bmatrix} \text{SOC}_t + 100 \\ I_{t+1}^\lambda \end{bmatrix}, \quad D(\boldsymbol{s}_t) = \begin{bmatrix} \text{SOC}_t - 100 \\ I_{t+1}^\lambda \end{bmatrix}, \quad W(\boldsymbol{s}_t) = \begin{bmatrix} \text{SOC}_t \\ I_{t+1}^\lambda \end{bmatrix}$$

What is captured by this formulation is the partial stochasticity of the environment, where the agent has control of the SOC, but is incapable of controlling the price level at the coming hour. Here, the assumption that the spot price is independent of how the battery operates in the electricity system is used, though this is not actually how the electricity market dynamics work.

1.1.3 Defining the reward signal

With the previous definitions, the reward signal is somewhat straightforward to define. As immediate reward the amount of energy traded (in MW per hour) times the spot price for the associated price level (given in DKK per MWh) is considered. Additionally, the reward for charging a fully loaded battery or discharging an empty battery is manually set to $-\infty$ as this action should be avoided - in practice, a clipping function is defined and exploited for modeling the part-wise reward function:

$$\text{clip}\left(x, \mathit{capacity}^{\min}, \mathit{capacity}^{\max}\right) := \min\left(\mathit{capacity}^{\max}, \max\left(a^{\text{SOC}}\left(x\right), \mathit{capacity}^{\min}\right)\right)$$

Letting $a^{\mathrm{SOC}}(\boldsymbol{s}_t) = \mathrm{SOC}_{t+1} - \mathrm{SOC}_t$ denote how an action, $a \in \mathcal{A}$, changes the SOC, the reward signal can be defined as:

$$R(\boldsymbol{s}_t, a) = \begin{cases} -\infty & \text{if } \operatorname{clip}\big(a^{\mathrm{SOC}}(\boldsymbol{s}_t)\big) = 0 \\ a^{\mathrm{SOC}}(\boldsymbol{s}_t) \cdot \tilde{\lambda}(I_t^{\lambda}) & \text{otherwise} \end{cases} \tag{1.2}$$

In practice the immediate choice for the price associated with the price level of the state, i.e. $\tilde{\lambda}(I_t^{\lambda})$, is to use the average of price values for either *high* or *low* price estimated from data, resulting in respectively $\tilde{\lambda}(I_t^{\lambda} = H) = 1436.35$ DKK and $\tilde{\lambda}(I_t^{\lambda} = L) = 887.46$ DKK.

1.2 Estimating transition probabilities

For completely describing the problem as a MDP definitions of the state-action transition functions are required. These functions are important concept as they constitute the building block for the transition matrices that are needed for solving the MDP and obtaining the optimal policy.

Assumptions: Before estimating the transition probabilities, we highlight that *1)* independence between the SOC and the price level, I^{λ}, is assumed while additionally *2)* assuming that the spot price is independent from actions taken. This assumption is in practice invalid as e.g. the price would be impacted by how much electricity is brought into the market by the battery owner through the action of discharging the battery. In addition, we *3)* assume a fully deterministic battery where the dynamics behave as expected - i.e. we know exactly what happens to the SOC when taking an action while the only stochastic changes to the state originates from the price level. Lastly, we rely on the Markov property by *4)* assuming that future states are independent of all past states and action given that we know the present state-action pair.

Thus, the state-action transition functions can be expressed by:

$$P_a\left(s = \begin{bmatrix} \mathrm{SOC}_t \\ I_t^{\lambda} \end{bmatrix}, s' = \begin{bmatrix} \mathrm{SOC}_{t+1} \\ I_{t+1}^{\lambda} \end{bmatrix}\right) = P_a\left(\mathrm{SOC}_{t+1} \mid \mathrm{SOC}_t, a\right) \cdot P\left(I_{t+1}^{\lambda} \mid I_t^{\lambda}\right)$$

Thus, the task of defining transition probabilities between states has now been simplified to defining transition probabilities for all combinations of price transitions as well as all transitions for the SOC. Due to assumption *3)*, the SOC transitions are non-stochastic for which reason the state-action transition functions can be defined as:

$$P_a\left(\mathrm{SOC}_{t+1} = c' \mid \mathrm{SOC}_t = c\right) = \begin{cases} 1 & \text{if } c' - c = a^{\mathrm{SOC}}\left(s_t = \begin{bmatrix} \mathrm{SOC}_t \\ I_t^{\lambda} \end{bmatrix}\right), \qquad a \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$$

As an example, consider the case of charging, i.e. $a = C(\cdot)$ in the state $s_t = \begin{bmatrix} 100 \\ I_t^{\lambda} \end{bmatrix}$, yielding $s_{t+1} = \begin{bmatrix} 200 \\ I_{t+1}^{\lambda} \end{bmatrix}$. Here, $a^{SOC}(s_t) = 200 - 100 = 100$ which gives a 100% chance of transitioning if $c' = 200$ and $c = 100$, and 0% otherwise. Considering all possible SOC combinations, the SOC transition matrices can be expressed as in Eq. 1.3. Here, the rows represent SOC levels at time t, i.e. $\mathrm{SOC}_t = 0, \mathrm{SOC}_t = 100$, etc. while columns represent SOC levels at time $t+1$. This way of expressing transitions clearly reveal that the matrices express *probabilities* since the sum of transition scores from SOC_t to SOC_{t+1} equals 1 (each row). We remark that the row corresponding to a fully charge battery when charging and the row corresponding to an empty battery when discharging have both been included with probabilities of 0 for illustrative purposes, however, in practice these rows are omitted as the actions are not considered valid given those states. Additionally, it is clear that *charging* or *discharging* from a respectively full and empty battery does not change the SOC as is the case for all states when choosing the *noop* action.

$$T_{a=C}^{\mathrm{SOC}} = \begin{bmatrix} 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \end{bmatrix}, T_{a=D}^{\mathrm{SOC}} = \begin{bmatrix} 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \end{bmatrix}, T_{a=W}^{\mathrm{SOC}} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix} \tag{1.3}$$

Now, following from assumption *2)* the price level is independent of the action taken. Additionally, it is possible to estimate the price level transition probabilities from the given data set. Thee procedure and resulting transition matrix is presented in Eq. 1.4-1.5 along with an illustration in Figure 1.3.

$$P\left(I_{t+1}^{\lambda}=p' \mid I_t^{\lambda}=p\right)=\frac{\#\text{ of transitions from } I_t^{\lambda}=p \text{ to } I_{t+1}^{\lambda}=p'}{\#\text{ of transitions from } I_t^{\lambda}=p \text{ to any } p'} \tag{1.4}$$

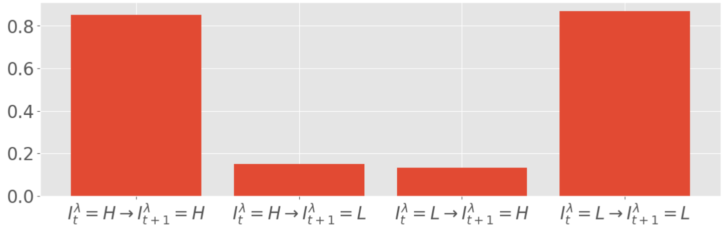


Figure 1.3: Visualization of the price level transition probabilities. As is seen, there is a higher tendency for the price to stay in the state it already is than changing to the other price level.

$$T^{I_\lambda}=\begin{array}{cc} I_{t+1}^{\lambda}=H & I_{t+1}^{\lambda}=L \\ \left[\begin{array}{c} 0.8513 \\ 0.1315 \end{array}\right. & \left.\begin{array}{c} 0.1487 \\ 0.8685 \end{array}\right] \end{array}\begin{array}{c} \\ I_t^{\lambda}=H \\ I_t^{\lambda}=L \end{array} \tag{1.5}$$

Lastly, what needs to be done is to combine the SOC- and price transition matrices. This has been done in Appendix A by expanding and multiplying the SOC transition matrices to fit the binary outcome of the possible price states. A visual representation is given in Figure 1.4 and it should be remarked that the construction does not allow for taking the *charge* action when the battery is fully charge nor taking the *discharge* action when the battery is empty.

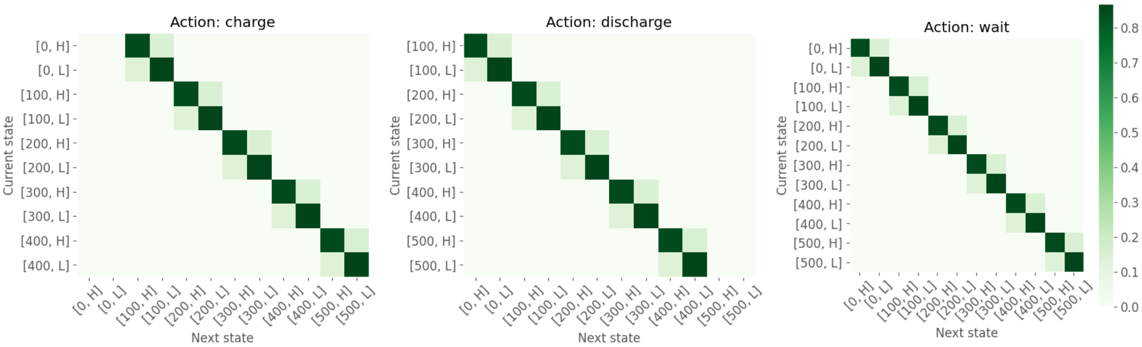


Figure 1.4: Transition matrices for each action of the combined SOC-price states, i.e. T_C, T_D and T_W, denoting the probability of going from current state, s_t, to the next state, s_{t+1}. The partly stochastic nature of the system - where the action impacts the SOC while the stochasticity origines from switches in the price level - is reflected within all matrices. Matrix representations are found in Appendix A.

With all elements of the MDP defined, Figure 1.5 presents a graphical representation of the battery problem. Note that the actual transition probabilities have not been reported due to notational clutter.

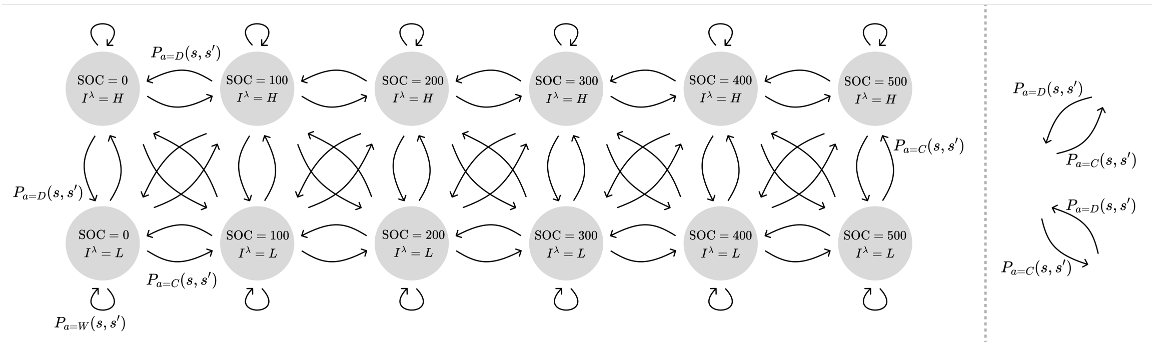

SOC = 0 $I^\lambda = H$
SOC = 100 $I^\lambda = H$
SOC = 200 $I^\lambda = H$
SOC = 300 $I^\lambda = H$
SOC = 400 $I^\lambda = H$
SOC = 500 $I^\lambda = H$
SOC = 0 $I^\lambda = L$
SOC = 100 $I^\lambda = L$
SOC = 200 $I^\lambda = L$
SOC = 300 $I^\lambda = L$
SOC = 400 $I^\lambda = L$
SOC = 500 $I^\lambda = L$
$P_{a=D}(s,s')$
$P_{a=C}(s,s')$
$P_{a=W}(s,s')$

Figure 1.5: Graphical representation of the battery MDP formulation. As is seen the "rows" has constant price level and varying SOC levels, whereas the opposite is the case for "columns". For all transition probabilities s' is the state pointed at by the arrow head and s is the state from .

2 Solving the Markov Decision Process

For solving the MDP, it is important to understand its dynamics: starting in a state s_0, an agent selects an action a_0 according to the transition probabilities P. When performing the action, the state of the environment changes to s_1 and the agent receives a numerical reward $R(s, a)$. A new action a_1 is chosen and the process proceeds. In the end, the agent will have performed a finite series of actions resulting in a sequence of state-action pairs:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3...$$

The goal of reinforcement learning is to maximize the expected cumulative reward - or the sum of all rewards received by the agent along the action trajectory - while possibly discounting the future rewards, i.e.:

$$E\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...\right] = E\left[\sum_{k=0} \gamma^k R(s_k)\right] \tag{2.1}$$

where γ is the discount factor $\in$[0, 1] that determines the prioritization of immediate rewards and future rewards. A small value for γ will mean a higher prioritization of immediate rewards, while the importance of future rewards is reduced, becoming non-existent for $\gamma = 0$ and reversely for increasing values of γ, eventually weighting the immediate and future rewards equally if $\gamma = 1$.

Generalizing this idea, the value function serves as an assessment mechanism for evaluating the performance of the agent when taking specific actions in specific states. This can be seen as the expected sum of the discounted immediate reward and future rewards that the agent can obtain by applying an action. Closely related to the value function is the optimal policy. This defines the optimal mapping from states to actions, i.e. $\pi : s \to a$ - in other words, the policy is a protocol describing the best way the agent can act when observing a specific state, s. Thus, finding the optimal policy gives an optimal solution to the Markov Decision Process.

Extending Equation 2.1, one can condition the policy as well as on the state of interest, s, which eventually gives an expression of the value function.

$$V^\pi(s) = E\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...|s_0 = s, \pi\right] \tag{2.2}$$

From this formulation it is straight-forward to derive the corresponding Bellman equation (Equation 2.3) by realizing that $R(s_0)$ is deterministic and given by $s_0 = s$ and by weighting value for "neighbouring" states according to the transition probabilities associated with the policy, π. The Bellman equation is a fundamental concept in dynamic programming and reinforcement learning.

$$V(s) = E\left[R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')\right] \tag{2.3}$$

2.1 Value Function Approximation and Optimal Policy

The optimal value function is the one giving the maximum value compared to all other values for each state. Mathematically this can be expressed as:

$$V^*(s) = \max V_\pi(s) = R(s) + \max_a \left(\gamma \cdot \sum_{s' \in \mathcal{S}} P_{sa}(s') \cdot V(s')\right)$$

For solving finite-state MDPs, two common approaches exist; 1) value iteration and 2) policy iteration. While both approaches originate from the Bellman equations, value iteration directly approximates the value function after which the optimal policy can easily be found using by maximizing wrt. the decision variables. On the other hand policy iteration considers continuously evaluating the approximated policy for obtaining an approximate value function from which a policy improvement step can be taken. Thus, value iteration finds the optimal policy after that value function has converged and thus result in the optimal policy - contrarily policy iteration depends on the initially presented policy. For the remaining part of this assignment, we focus on finding the optimal solution using *value iteration.*

2.2 Value Iteration Algorithm

The Value Iteration algorithm (Algorithm 1) involves updating the value function of each state based on the Bellman Equations until it converges to the optimal values. Initially values for all states are set to 0 after which the updated value for a state is computed by weighting the current values according to the transition probabilities. The algorithm runs until a convergence criteria is met at which point the value function represents the optimal expected cumulative rewards for each state.

After convergence of the value iteration algorithm, the optimal policy can be obtained by selecting actions that maximize the expected cumulative reward in each state, i.e. by using the approximated value function. This can be done using the policy improvement step, where for each state, the algorithm selects the action that leads to the highest expected cumulative reward. It should however be emphasized that extracting the policy is done once for value iteration while iteratively carried out for policy iteration.

Algorithm 1 Value Iteration Algorithm

```
Initialize $V(s) = 0$, for all $s \in \mathcal{S}$
repeat
    $\Delta = 0$
    for each $s \in S$ do
        $v = V(s)$
        $V(s) = \max_a \left(R(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s', a) \cdot V(s')\right)$        ▷ Bellman Equation
        $\Delta = \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$        ▷ Convergence number

$\pi^*(s) = \arg\max_a \left(R(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s', a) \cdot V(s')\right)$        ▷ Extract optimal policy
```

The key insight behind value iteration is based on the Bellman Optimal Equation. The optimal value function satisfies this equation, which relates the value of a state to the maximum expected cumulative reward achievable from that state. By iteratively applying the Bellman Optimal Equation, the algorithm converges to the optimal values. The reason why the algorithm guarantees to converge to the optimal policy is because it assumes a finite MDP, meaning there is a finite number of states and actions. However, for infinite MDPs, modifications or approximations will be necessary.

3 Evaluation approach

For evaluating possible solutions to the MDP a two-part evaluation protocol is considered, that relies on *1)* qualitatively analyzing the optimal policy, π^*, and *2)* quantitatively applying the optimal policy on the true price signal to empirically determine the expected revenue of the battery controlling agent. For both approaches, a variety of discount factors, γ, are considered and compared in terms of convergence time and expected revenue. A similar experiment is conducted for the reward function where the ratio between high- and low-price state rewards is varied.

3.1 Empirically computing revenues

As for the quantitative evaluation approach, the goal is to find the expected accumulated revenue after letting the battery owner interact with the environment for a certain number of time steps, T.

The resulting revenue expression is expressed in Equation 3.1, where h denotes the considered evaluation horizon and k denotes the horizon index, i.e. which parts of the time series that are exploited. Similarly to previous definitions related to actions, $\pi^*\left(\boldsymbol{s}_t\right)^{\text{SOC}}$ denotes the change in SOC when taking the optimal action for state $\boldsymbol{s}$ at time t and the price associated with the action is λ_{t+1}. Thus, the revenue gained from interacting with the market exploits the price for the coming hour and is oppositely related to the SOC update (in terms of being positive and negative).

$$G(k, h, T) = -\sum_{t=k\cdot h}^{T+k\cdot h} \pi^*\left(\boldsymbol{s}_t\right)^{\text{SOC}} \cdot \lambda_{t+1} \tag{3.1}$$

With an expression of the revenue at hand, the expected revenue and its standard error is computed using Monte Carlo sampling by considering a variety of "folds", i.e.:

$$\tilde{G}\left(h,T\right) = \mathbb{E}\big[G(k,h)\big] = \frac{1}{K}\sum_{k=0}^{K} G(k,h,T) \tag{3.2}$$

4 Results & Discussion

Figure 4.1 and Table 1 (Appendix B) shows the optimal policies determined from the approximated value functions for a sweep of γ values selected through pilot runs. One trend that is general across discount values is that *waiting* is optimal if the battery is empty and the price level is high. In addition, the *discharge* action seems to dominate high and low price states as the discount value decreases. This suggests that increasing the discount factor introduces a lower level of sensitivity wrt. the price level within the model. Luckily, this is in line with the underlying intuition of the discount factor, where lower values should give a value function that is more prone to forget values associated with previous update iterations. That the discount factor works as expected is furthermore visible from the fact that the *waiting* action is shifted towards higher SOC states when forgetting less - e.g. considering a relatively "good" memory of e.g. $\gamma = 0.99$, the optimal policy seems in line with what one could consider a simple but efficient trading strategy, namely charging at low price states and discharging at high price states.

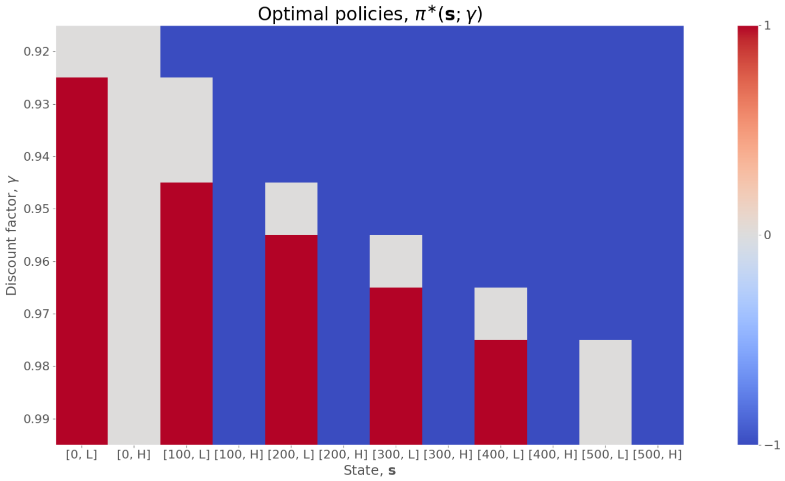


Figure 4.1: A visualization of the optimal policies found for a sweep of discount factor values, γ. The coloring of the policy matrix follows that of the colorbar where 1 indicates *charging*, -1 indicates *discharging* and 0 indicates the *waiting* action.

In Figure 4.2 the outcome of following the quantitative evaluation protocol presented in Section 3 is reported. As is seen on the left-most figure, the convergence rates increase with the discount factor value - remark, that the axes are given in log-scale for better comparison of the individual model realizations.

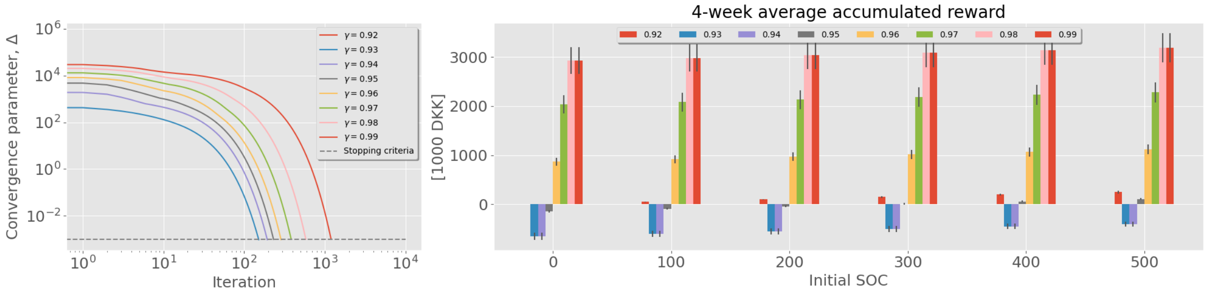


Figure 4.2: *Left:* Convergence plot of Δ against iterations of the Value Iteration algorithm. Both the x- and y-scale are logarithmic for easier comparison between the convergence rates across discount factor values. *Right:* Effects of discount factor, γ, and the initial SOC on the expected accumulated revenue within a four week period. The associated data is presented in Table 2 (Appendix B).

On the right-most of Figure 4.2, the expected revenue evaluated using $K = 26$ "folds" and a horizon of 4 weeks, i.e. $h = 24 \cdot 7 \cdot 4$ hours, is presented along with the standard error of the mean (SEM). It is clearly seen that some parameter setting yield high expected revenues - such as $\gamma = 0.99$ with expected revenue of approx. 3.000.000 DKK - while other parameter values results in expected revenues of incomparable scales. Considering the associated policies (Figure 4.1), the occurring behaviours are mostly reasonable as having a better memory (higher γ) would result in the policy being able to disregard focusing on immediate revenue as e.g. is the case for the policy with $\gamma = 0.92$. For that case, the *discharge* action is always chosen when possible while *waiting* is otherwise chosen, thus gaining revenue reduces to the cases where the initial SOC is non-zero, where the agent will then instantaneously bid into the market. Contrarily for $\gamma = 0.99$ the policy captures the fact that repeated high- and low-price levels can occur in sequence for which reason the price level ends up having an impact on what the optimal action is.

Taking a closer look at the cases of $\gamma \in \{0.93, 0.94, 0.95\}$ evaluating the optimal policies almost surely yield negative expected revenues, independently of the battery's initial state-of-charge. Considering the associated optimal policies this seems rather off since the battery has learned to charge and discharge at respectively low and high price states for which reason one would expect a positive revenue outcome. Since this is not the case, it could be the case that the price level indicator signal is not properly capturing high- and low price level which impacts the fast dynamics and that this indicator signal should eventually be rephrased.

In Figure 4.3 an example of the battery-price system and its reaction to executing the policy obtained with $\gamma = 0.99$ and an initial SOC of 0 on a 4-week period is visualized. From the 24 hour averaged accumulated revenue, an approximately constant increasing revenue trend is seen.

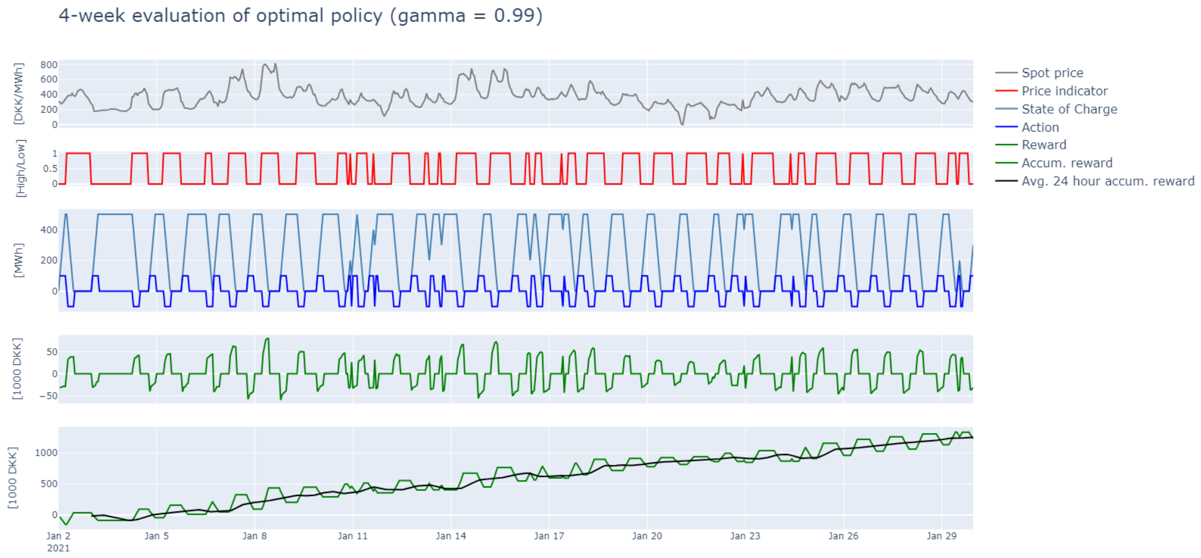


Figure 4.3: Dynamics of the battery-price system when executing the best found policy for a random time period of 4 weeks in January 2021. *Row 1:* Raw spot price (in DKK). *Row 2:* Price level indicator signal. *Row 3:* State-of-Charge and the actions taken. *Row 4:* The revenue obtained according to Equation 3.1 with $h = T = 1$ and $k = 0$. *Row 5:* Accumulated and 24-hour averagely accumulated revenue.

4.1 Experimenting with the reward signal

As a brief note, an experiment considering the ratio between the reward associated with high- and low price states is conducted. As is seen in Figure 4.4, difference between the price level of the two states has a clear impact on the expected revenue. What is clearly visible that if the reward for high- and low price states are identical, i.e. price factor 1, the expected revenue is zero for all of the examined discount values. Contrarily, high differences between high- and low price rewards (i.e. price factor 1000) implies that the discount factor can decrease by quite a margin while giving similar empirical results (from $\gamma = 0.99 \rightarrow \gamma = 0.5$). This highlights that the definition of the reward function has a great impact on the policy that the agent is able to learn using the simple scheme of value iteration.

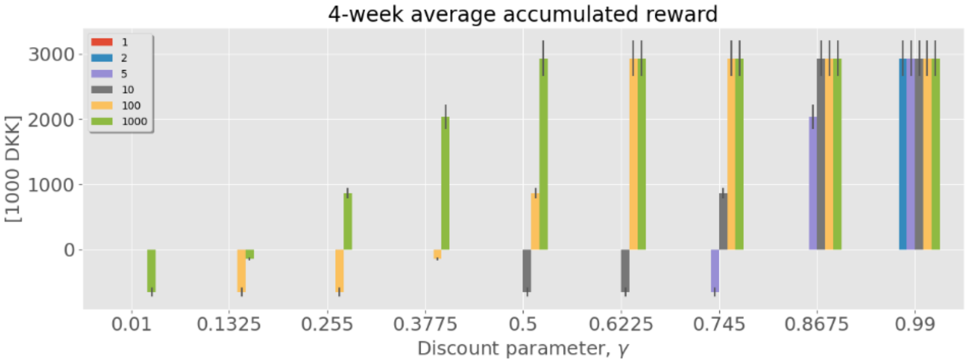


Figure 4.4: Label indicating the price difference factor

Limitations

First of all the current evaluation approach includes evaluation of the price signal that was also used for empirically estimating the transition probabilities used for fitting the model. This introduces a bias in the estimated revenues that one should be aware of. In practice, this could simply be defined by evaluating the policies on separate test sets that were not used for estimating transition probabilities, however, by the time of realization, the deadline was too close.

Secondly, though reducing the complexity of the modeling task, the initial assumption stating that the actions taken - i.e. charging or discharging - does not influence the price is a rather strong assumption that in practice does not hold as the price is strongly influenced by the ratio of demand and supply. Simplifying the electricity market, if we choose to charge the battery, the general demand for electricity will increase and implicitly the price will increase. On the other hand, if we choose to discharge the battery and inject a certain amount of energy into the network, the supply is greater and the price will tend to decrease.

Lastly, the discrete modelling approach applied limits how well we can actually solve the problem at hand - in practice attributes such as the price level are in nature defined on the entire real axis (i.e. continuous) for which reason one could approach the modeling task by solving a continuous MDP instead. If doing so, one could additionally consider including continuous time as a feature in the state such that the policy associated with an otherwise discrete state (e.g. high SOC, high price) could vary over time which would be essential for modeling dynamic systems properly.

5 Conclusion

Throughout this project the problem of learning how to optimally control a large battery in real-time was examined. This was done by *1)* phrasing the problem as a Markov Decision Process, *2)* discretizing the state and action spaces in order to *3)* determine the transition probabilities. Using those *4)* the Value Iteration algorithm was applied to approximate the value function and determine the optimal policy after which *5)* an investigation of the importance of the discount factor was carried out. For meeting the real-time part of the assignment description, we *6)* created an evaluation setup based on continuously applying the optimal policies to a time-varying battery-price system before lastly *7)* discussing limitations related to the modelling approach and proposing approach to problem from a continuous MDP point-of-view if we had had more time. As such, the presented work gives an overall idea of how to approach the major parts of reinforcement learning in energy systems while focusing on modeling discrete Markov Decision Processes.

Appendix

A - Transition matrices

$$T_C \approx \begin{array}{cccccccccccc|c}
\begin{bmatrix}0\\H\end{bmatrix} & \begin{bmatrix}0\\L\end{bmatrix} & \begin{bmatrix}100\\H\end{bmatrix} & \begin{bmatrix}100\\L\end{bmatrix} & \begin{bmatrix}200\\H\end{bmatrix} & \begin{bmatrix}200\\L\end{bmatrix} & \begin{bmatrix}300\\H\end{bmatrix} & \begin{bmatrix}300\\L\end{bmatrix} & \begin{bmatrix}400\\H\end{bmatrix} & \begin{bmatrix}400\\L\end{bmatrix} & \begin{bmatrix}500\\H\end{bmatrix} & \begin{bmatrix}500\\L\end{bmatrix} & \\
\hline
0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}0\\H\end{bmatrix} \\
0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}0\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}100\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}100\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & \begin{bmatrix}200\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & \begin{bmatrix}200\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & \begin{bmatrix}300\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & \begin{bmatrix}300\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & \begin{bmatrix}400\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & \begin{bmatrix}400\\L\end{bmatrix}
\end{array}$$

$$T_D \approx \begin{array}{cccccccccccc|c}
\begin{bmatrix}0\\H\end{bmatrix} & \begin{bmatrix}0\\L\end{bmatrix} & \begin{bmatrix}100\\H\end{bmatrix} & \begin{bmatrix}100\\L\end{bmatrix} & \begin{bmatrix}200\\H\end{bmatrix} & \begin{bmatrix}200\\L\end{bmatrix} & \begin{bmatrix}300\\H\end{bmatrix} & \begin{bmatrix}300\\L\end{bmatrix} & \begin{bmatrix}400\\H\end{bmatrix} & \begin{bmatrix}400\\L\end{bmatrix} & \begin{bmatrix}500\\H\end{bmatrix} & \begin{bmatrix}500\\L\end{bmatrix} & \\
\hline
0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}100\\H\end{bmatrix} \\
0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}100\\L\end{bmatrix} \\
0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}200\\H\end{bmatrix} \\
0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}200\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}300\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & \begin{bmatrix}300\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & \begin{bmatrix}400\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & \begin{bmatrix}400\\L\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & \begin{bmatrix}500\\H\end{bmatrix} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & \begin{bmatrix}500\\L\end{bmatrix}
\end{array}$$

$$T_W \approx \begin{array}{cc} \begin{array}{cccccccccccc} \begin{bmatrix}0\\H\end{bmatrix} & \begin{bmatrix}0\\L\end{bmatrix} & \begin{bmatrix}100\\H\end{bmatrix} & \begin{bmatrix}100\\L\end{bmatrix} & \begin{bmatrix}200\\H\end{bmatrix} & \begin{bmatrix}200\\L\end{bmatrix} & \begin{bmatrix}300\\H\end{bmatrix} & \begin{bmatrix}300\\L\end{bmatrix} & \begin{bmatrix}400\\H\end{bmatrix} & \begin{bmatrix}400\\L\end{bmatrix} & \begin{bmatrix}500\\H\end{bmatrix} & \begin{bmatrix}500\\L\end{bmatrix} \end{array} & \\ \left[\begin{array}{cccccccccccc} 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0.15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0.87 \end{array}\right] & \begin{array}{c} \begin{bmatrix}0\\H\end{bmatrix} \\ \begin{bmatrix}0\\L\end{bmatrix} \\ \begin{bmatrix}100\\H\end{bmatrix} \\ \begin{bmatrix}100\\L\end{bmatrix} \\ \begin{bmatrix}200\\H\end{bmatrix} \\ \begin{bmatrix}200\\L\end{bmatrix} \\ \begin{bmatrix}300\\H\end{bmatrix} \\ \begin{bmatrix}300\\L\end{bmatrix} \\ \begin{bmatrix}400\\H\end{bmatrix} \\ \begin{bmatrix}400\\L\end{bmatrix} \\ \begin{bmatrix}500\\H\end{bmatrix} \\ \begin{bmatrix}500\\L\end{bmatrix} \end{array} \end{array}$$

B Additional results

Table 1 presents the optimal policies for the gamma sweep experiment which was also reported visually in Figure 4.1.

Table 1: Matrix version of the optimal policies found from Value Iteration with a variety of discount factors.

State, s	$\begin{bmatrix}0\\L\end{bmatrix}$	$\begin{bmatrix}0\\H\end{bmatrix}$	$\begin{bmatrix}100\\L\end{bmatrix}$	$\begin{bmatrix}100\\H\end{bmatrix}$	$\begin{bmatrix}200\\L\end{bmatrix}$	$\begin{bmatrix}200\\H\end{bmatrix}$	$\begin{bmatrix}300\\L\end{bmatrix}$	$\begin{bmatrix}300\\H\end{bmatrix}$	$\begin{bmatrix}400\\L\end{bmatrix}$	$\begin{bmatrix}400\\H\end{bmatrix}$	$\begin{bmatrix}500\\L\end{bmatrix}$	$\begin{bmatrix}500\\H\end{bmatrix}$
0.92	W	W	D	D	D	D	D	D	D	D	D	D
0.93	C	W	W	D	D	D	D	D	D	D	D	D
0.94	C	W	W	D	D	D	D	D	D	D	D	D
0.95	C	W	C	D	W	D	D	D	D	D	D	D
0.96	C	W	C	D	C	D	W	D	D	D	D	D
0.97	C	W	C	D	C	D	C	D	W	D	D	D
0.98	C	W	C	D	C	D	C	D	C	D	W	D
0.99	C	W	C	D	C	D	C	D	C	D	W	D

Table 2: The data associated with 4.2.

SOC	0	100	200	300	400	500
0.92	0.00 ± 0.00	48.84 ± 5.31	98.73 ± 11.06	149.59 ± 17.12	198.44 ± 22.51	250.03 ± 28.47
0.93	-650.35 ± 69.98	-601.51 ± 66.39	-552.67 ± 63.03	-502.39 ± 59.09	-451.61 ± 55.22	-400.87 ± 51.68
0.94	-650.35 ± 69.98	-601.51 ± 66.39	-552.67 ± 63.03	-502.39 ± 59.09	-451.61 ± 55.22	-400.87 ± 51.68
0.95	-138.57 ± 23.99	-88.32 ± 19.78	-39.48 ± 17.95	9.36 ± 17.58	59.14 ± 17.78	110.02 ± 19.83
0.96	867.02 ± 80.49	917.84 ± 84.20	967.41 ± 87.97	1016.25 ± 91.57	1065.09 ± 95.34	1114.98 ± 99.57
0.97	2031.58 ± 185.29	2082.93 ± 189.77	2133.17 ± 194.43	2183.34 ± 199.05	2232.18 ± 202.96	2281.02 ± 206.94
0.98	2930.70 ± 273.81	2983.92 ± 278.76	3034.60 ± 283.65	3085.34 ± 288.76	3136.14 ± 293.80	3184.97 ± 297.82
0.99	2930.70 ± 273.81	2983.92 ± 278.76	3034.60 ± 283.65	3085.34 ± 288.76	3136.14 ± 293.80	3184.97 ± 297.82