Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master's Thesis Computer Science

Stochastic Optimization for Market-Integrated Scheduling of Energy Storage Systems

Gereon Recht

20.05.2023

Reviewers

Dr. Nicole Ludwig

Prof. Dr. Volker Franz

Recht, Gereon:
Stochastic Optimization for Market-Integrated Scheduling of Energy Storage Systems
Master's Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: 21.11.2022 - 20.05.2023

Abstract

We take the role of an energy storage system operator bidding into a short-term energy market. The goal is to obtain a decision rule which places bids such that the expected revenue is maximized. We profit through *arbitrage* and thereby stabilize the power grid as we take some of the excess supply when *buying low* and some of the excess demand when *selling high*. Moreover, since excess supply often occurs due to high renewable penetration, we shift renewable energy sources in time. In order to characterize the optimal decision rule we first describe the problem as a Markov decision process. The complexity of the problem entails that it is infeasible to determine this optimal decision rule, so we formulate a computationally efficient approximation method which leads to a near-optimal decision rule. Our approximation method BADP-LATTICE uses backward approximate dynamic programming with a scenario lattice. The approach is evaluated in a stylized setting where an optimal solution can be computed. There, it achieves $\sim 92 - 97\%$ of the optimal revenue while leading to a reduction of $\sim 91\%$ in computation time. Furthermore, it is evaluated in a realistic setting modelling the trading on the real-time market of the New York Independent System Operator. The market prices are modelled using a Poisson Spike Process which is calibrated on empirical data. As an optimal solution cannot be computed in this setting, the approach is evaluated against a perfect foresight solution, where all prices are known in advance. It achieves $\sim 82\%$ of the perfect foresight revenue. Furthermore, the optimized implementation of BADP-LATTICE leads to reduction in computation time of more than 99% when compared to a benchmark method.

Zusammenfassung

In dieser Arbeit nehmen wir die Rolle des Betreibers eines Energiespeichers ein, der mit seiner Kapazität auf einem kurzfristigen Energiemarkt handelt. Das Ziel ist die Bestimmung einer Entscheidungsregel, welche die Gebote so setzt, dass der erwartete Ertrag maximiert wird. Wir machen Gewinn durch *Arbitrage* und stabilisieren dadurch das Energienetz, denn wir konsumieren einen Teil des Überangebots wenn wir *billig kaufen* und reduzieren die Knappheit wenn wir *teuer verkaufen*. Weil ein Überangebot häufig durch eine hohe Produktion von erneuerbaren Energieträgern ausgelöst wird, verschieben wir damit auch erneuerbare Energieproduktion auf einen späteren Zeitpunkt. Um eine optimale Entscheidungsregel zu beschreiben, formulieren wir das Problem als Markow-Entscheidungsprozess. Die Komplexität des Problems erfordert eine effiziente Approximationsmethode, welche statt der optimalen eine nahezu optimale Entscheidungsregel bestimmt. Unsere Approximationsmethode BADP-LATTICE nutzt backward approximate dynamic programming mit einem scenario lattice. Wir evaluieren den Ansatz in einem vereinfachten Experiment, wo die optimale Entscheidungsregel berechenbar ist. Dort erreicht BADP-LATTICE $\sim 92 - 97\%$ des optimalen Ertrags. In einem weiteren Experiment werten wir den Ansatz in einer realistischeren Umgebung aus, welche den Handel auf dem Echtzeit-Markt des New York Independent System Operators simuliert. Dazu modellieren wir die Marktpreise mit einem Poisson Spike Process, den wir auf empirischen Daten kalibrieren. Da die optimale Lösung in dieser Umgebung nicht berechnet werden kann, evaluieren wir BADP-LATTICE gegen die Lösung des deterministischen Problems, wo die Preise im Voraus bekannt sind. Im Vergleich zur deterministischen Lösung erreicht BADP-LATTICE $\sim 82\%$ des Ertrags. Des Weiteren führt die optimierte Implementierung von BADP-LATTICE zu einer Reduktion der Rechenzeit von mehr als 99% im Vergleich zu einer Benchmark-Methode.

Acknowledgements

Thanks to my advicor Dr. Nicole Ludwig for giving me the freedom and trust to try out my ideas. Thanks to Prof. Dr. Volker Franz and his group for good questions and remarks. Thanks to Prof. Dr. Bismark Singh for the advice and sharing code from his PhD thesis [1], which served as a starting point for my implementation. Thanks to my friends Robin Link and Frederik Unger who proofread the thesis. Thanks to my family who supported me throughout my studies. And finally, thanks to my girlfriend Helen for supporting me and participating in long library sessions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this work we take the role of an energy storage operator who uses their capacity to trade on a short-term energy market. We profit from *arbitrage*, i.e. buying low and selling high, and want to maximize our expected revenue. Each trading decision is made under uncertainty, as we must commit to buying or selling energy ahead of the delivery period where the exchange takes place. Our behavior happens to benefit the power grid: Prices reflect the balance of supply and demand, which must match at all times within a tolerance such that the grid remains stable [3]. Times of excess supply coincide with low or even negative prices, while times of excess demand show high prices. Thus, when we do arbitrage we take some of the excess supply and some of the excess demand, and in turn help to stabilize the grid. Moreover, low prices coincide with high renewable penetration, whereas high prices indicate high conventional generation. This is due to the merit-order curve, which activates generators in an order that increases in their marginal costs. Hence, we often shift renewable energy generation in time, leading to reduced CO_2 emissions.

The goal of this work is formulating an approximation method which determines a decision rule for bidding into the real-time market of the New York Independent System Operator (NYISO). We build on a publication which models the same problem [2]. Therein, the authors formulate the bidding problem as a Markov decision process and use an *approximate dynamic programming* (ADP) approach called MONOTONE-ADP to obtain an approximate solution. This is necessary due to the complexity of the problem, which does not allow determining an optimal solution using *backward dynamic programming* (BDP), the canonical algorithm for finite Markov decision processes. Following the credo of "try the simplest model first", we use a simpler approach called *backward approximate dynamic programming* (BADP) which is structurally very similar to BDP. We aim to answer the question whether our BADP approach achieves a similar solution quality as MONOTONE-ADP.

We introduce the foundations of Markov decision processes, backward (ap-

proximate) dynamic programming, and stochastic price simulations in Chapter 2. Note that the methodological fundamentals of stochastic price simulations are often neglected in the literature. Following this, we present the formulation of the bidding problem as a Markov decision process and our approximation method BADP-LATTICE in Chapter 3. In Chapter 4, we evaluate the decision rules determined by BADP-LATTICE in two different experiments. The first evaluates the approximation quality of our approach in a stylized setting and allows a comparison to MONOTONE-ADP. The second uses a more realistic setting, which simulates the NYISO real-time market. For this second experiment we conduct a sub-experiment in order to determine the effect of some hyper-parameters of BADP-LATTICE on the solution quality, allowing us make an informed choice.

Consider [4] for a thorough review of publications on modeling energy storage problems. Recent publications successfully demonstrate the use of BADP on energy storage problems [3, 5–7]. In [3], the authors model the problem of bidding into both the German day-ahead market and the opening auction of the German intraday-market. They apply BADP with a competitive price forecast and evaluate their approach on out-of-sample empirical price data. Their decision problem differs from ours as it only has two decision stages in each day, while the short-term market we consider has 24. Moreover, we do not use a price forecast but a stochastic price model, which is only intended to reproduce the characteristics of energy prices. Hence we also evaluate our approach on synthetic data from this price model, which is common practice. Finally, they find that for their specific problem a BADP-LATTICE approach does not perform well and instead handle the uncertainty with an approach that linearly combines sample paths. In [5], a hybrid setting for the bidding problem is considered where there is not only an energy storage system, but also a wind power plant. A decision rule is determined using BADP. The authors of [6] consider a similar hybrid setting and especially find that their BADP approach outperforms established ADP approaches. Moreover, in [7] the authors successfully obtain near-optimal solutions for an energy storage problem using BADP. They use low-rank approximations to extrapolate their approximate decision rule. Other approaches use e.g. ADP [2, 8], reinforcement learning [9, 10], a multi-stage stochastic program [11], prediction bands of a novel probabilistic forecast to determine trading recommendations [12], or even an analytical approach to determine optimal bidding functions [13].

Chapter 2

Foundations

In this chapter, we introduce Markov decision processes as the framework we use for the formulation of our bidding problem. After a brief description of energy markets, we then introduce models and methods for the stochastic simulation of energy prices.

2.1 Markov Decision Processes

Markov decision processes (MDPs) can be used to describe a setting where we observe the *state* of some dynamic system at time t and must choose an *action*, which we choose according to some decision rule called a *policy*. We receive a *contribution* for this action and our system *transitions* to the subsequent state at time $t+1$, depending on our action and the arrival of new *exogenous information*. We observe the subsequent state at $t+1$ and choose our next action. This procedure is repeated, either with a finite or an infinite optimization horizon. In this work, we focus on a finite horizon problem. We aim to determine an optimal policy such that our expected revenue is maximized [14].

Formally, we model an MDP following the general framework for sequential decision problems presented in [15, 16]. We define the problem over the time indices $t \in \mathcal{T}$, where the subscript t on a variable indicates when it becomes known[1]. We describe an MDP using the following objects:

- The *state* variable $S_t \in \mathcal{S}$, where $\mathcal{S}$ denotes the space of possible system states, which we assume to be finite.
- The *action* variable $a_t \in \mathcal{A}$, where $\mathcal{A}$ denotes the space of possible actions, which we assume to be finite as well. The action a_t is chosen at

[1] This is an important distinction especially for the actions: We must often commit to an action before the reward is known, i.e. before it changes the system state. Thus, it could be tempting to set the index of the action variable to the time where it is applied, but this is not the notational convention we use here.

time t after state S_t becomes known.

- The *exogenous information* W_{t+1} which we observe during $t+1$ before taking action a_{t+1}. It may be comprised of several variables which are uncertain at time t. We write its possible outcomes as $\Omega = \{\omega_1, \omega_2, \dots\}$.

- The *contribution function* $C_t : \mathcal{S} \times \mathcal{A} \times \Omega \to \mathbb{R}$. It represents the reward we receive for an action given a certain state at time t. The contribution can be random if it depends on the realization of the exogenous information.

- The *transition function* $S_{t+1} = S^M(S_t, a_t, W_{t+1})$. It determines the subsequent state S_{t+1} based on S_t, a_t, and the realization of the exogenous information W_{t+1}.

- The *policy* (function) $A_t^\pi : \mathcal{T} \times \mathcal{S} \to \mathcal{A}$ which maps each state to an action. Any decision rule can form a policy, so the set of possible policies Π is used for notational purposes in the following, but we do not explicitly define it. We aim to determine a policy such that our expected cumulative reward is maximized.

- The *objective function*

$$\max_{\pi \in \Pi} \mathbb{E}\Big[\sum_{t=0}^{T-1} C_t(S_t, A_t^\pi(S_t), W_{t+1}) \,|\, S_0 \Big],$$

  where $S_{t+1} = S^M(S_t, A_t^\pi(S_t), W_{t+1})$, i.e. the state transitions are given by the transition function. It defines the optimal policy given some initial state $S_0 \in \mathcal{S}$.

- The recursively defined *value function*

$$V_t(S_t) = \max_{a_t \in \mathcal{A}_t} \mathbb{E}\Big[C_t(S_t, a_t, W_{t+1}) + V_{t+1}\big(S^M(S_t, a_t, W_{t+1})\big) \Big], \tag{2.1}$$

  with some terminal reward V_T (often set to zero), which is also called "Bellman's equation" after its author [17]. It represents the value of being in a certain state at time t and, like the objective function, defines the optimal policy, which is given by the actions that form the value function. Intuitively, the value function interleaves the sequential decisions over the individual stages: In each state, it considers both the expected contribution of taking the action a_t and the expected value of being in the subsequent state if we were to take action a_t. Note that the expectation is formed over the exogenous information W_{t+1}.

It is called a *Markov* decision process because for a fixed policy, the sequence of states forms a Markov chain. That is, given the current state S_t and action a_t, the transition function is conditionally independent of prior states and actions [14]. In principle, every dynamic system can be modeled to fulfill the Markov property by including history-dependent information in the state variable [16].

2.1.1 Backward Dynamic Programming

For Markov decision processes with a finite horizon and finite state, action and outcome spaces, the optimal policy can be computed using *backward dynamic programming*[2] (BDP). This algorithm steps backwards in time and uses Bellman's equation (Eq. 2.1) as an update rule to determine the optimal action $a_t \in \mathcal{A}$ for each $t \in \mathcal{T}$ [16]. A suitable terminal reward beyond the final stage must be specified, which is often simply set to zero. The procedure is shown in Algorithm 6. After obtaining the policy A_t^π, it can be evaluated given an initial state $S_0 \in \mathcal{S}$ as follows: Starting at stage $t = 0$, for each stage t we take action $a_t = A_t^\pi(S_t)$ and then transition to the following state $S_{t+1} = S^M(S_t, a_t, W_{t+1})$ using our transition function. The revenue is given as the cumulative reward $\sum_{t=0}^{T} C_t(S_t, a_t)$ over all stages.

The complexity of BDP is $\Theta(|\mathcal{T}| \cdot |\mathcal{S}| \cdot |\mathcal{A}| \cdot |\Omega|)$[3], where Ω represents the outcome space of the exogenous information. Thus, it suffers from three curses of dimensionality caused by the state space, the action space, and the outcome space [16].

Algorithm 1: BDP

$V_T(S) = 0 \quad \forall S \in \mathcal{S}$ `// Terminal reward`
for t = *T - 1, . . . , 0* **do**
  for $S_t \in \mathcal{S}$ **do**
    Let $f(a_t) = \mathbb{E}\Big[C(S_t, a_t, W_{t+1}) + V_{t+1}\big(S^M(S_t, a_t, W_{t+1})\big)\Big]$.
    $V_t(S_t) := \max_{a_t \in \mathcal{A}_t} f(a_t)$
    $A_t^\pi(S_t) := \arg\max_{a_t \in \mathcal{A}_t} f(a_t)$

2.1.2 Backward Approximate Dynamic Programming

Due to the three curses of dimensionality or continuous state and action spaces it is often not feasible to use BDP. Thus, several approximation techniques have

[2] It is also known as the *method of successive approximations*, *backward induction*, or *value iteration*.

[3] Recall the definition of a tight asymptotic bound: $\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 \geq 0 \text{ such that } 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \, \forall n \geq n_0\}$ [18].

emerged. Backward Approximate Dynamic Programming (BADP) stays closest to BDP by sampling[4] the spaces involved, i.e. by using sampled state and action spaces $\hat{\mathcal{S}} \subseteq \mathcal{S}$, $\hat{\mathcal{A}} \subseteq \mathcal{A}$. Moreover, the exogenous information is described by a finite set of scenarios $\hat{\Omega} = \{\omega_1, \ldots, \omega_N\} \subseteq \Omega$. The expectations in the MDP formulation can then be approximated for example using sample average approximation over this large number of scenarios, or by reducing the number of scenarios into a *scenario lattice*, where each scenario has an associated probability [3, 19]. These approximations allow determining a *lookup table* value function approximation $\bar{V}$ over the sampled spaces in the same manner as in BDP. If an action or a realization of the exogenous information causes the transition to a state which is not part of the sampled state space, it is common practice to linearly interpolate the value function [3].

2.1.3 Forward Approximate Dynamic Programming

Another commonly used technique is called *(forward) approximate dynamic programming* (ADP), which uses Bellman's equation but steps forward in time. Like BADP, it uses a finite set of scenarios $\hat{\Omega} \subseteq \Omega$, but it uses them sequentially: In each forward pass over the optimization horizon, the problem is simulated for a single scenario $\omega \in \hat{\Omega}$. Along the way, the value function approximation is updated, which can e.g. have the form of a lookup table. As ADP only visits states where the scenarios lead it to, it avoids the curse of the state space size. However, it needs to iterate over a potentially very large number of scenarios in order to converge to a near-optimal solution. For further information consider [16]. The MONOTONE-ADP approach of the publication we build on uses this technique, but speeds up convergence by exploiting the monotonicity of the value function [2].

2.2 Energy Markets

For a thorough description of energy markets, see [20]. Electrical energy is traded in the unit of megawatt-hours (MWh), where x MWh is equivalent to the output of x MW of power over the duration of one hour. Bids to buy or sell energy are submitted in the form of (p, x), where p denotes the price below (above) which the market participant wants to buy (sell) the energy generated by an output of x MW of power over the fixed duration of the delivery period. The market operator determines a market clearing price by matching bids to buy and sell, a process which is also called *settling*.

Trading takes place on sequential markets which serve different purposes as

[4] This is also called *discretizing* in the literature, although this technique is not limited to the approximation of continuous problems.

they differ in the length of their trading periods and in the difference between the closing time and the time of delivery. The largest amount of energy is traded on forward markets which close many hours before the time of delivery. Short-term imbalances between realized supply and demand are traded on the spot market, which allows settlements much closer to the delivery period and in shorter time intervals. In this work, we consider the real-time market of the New York Independent System Operator (NYISO), which is actually comprised of two markets: The NYISO real-time commitment market, where bids are submitted ahead of the delivery period, and the NYISO real-time dispatch market, where the bids submitted to the commitment market are settled [21]. Moreover, the dispatch market operates on smaller time increments than the commitment market, which entails that a single bid is settled multiple times.

2.3 Stochastic Simulation of Energy Prices

Electricity price simulations are most notably used for pricing derivatives [22, 23], but also in experimental setups by the operations research (OR) community [5, 8, 11, 24–30]. In the context of sequential decision problems involving energy prices, these simulations are used to create a number of *price paths* for the numerical computation of the expected values contained in the problem formulations. Moreover, they are used to evaluate the decision rules determined by the models. Despite the ubiquity and non-triviality of these simulation methods, the OR publications mostly do not give a detailed account of the theory or the practical usage. In particular, the steps involved for the estimation and simulation of stochastic processes in discrete time are often omitted. Hence, we give a brief introduction into stochastic processes and present all the necessary steps for their simulation as well as the approximations involved.

In general, after stabilizing the variance with a suitable transformation f, the energy price P_t is modeled as a sum of a stochastic part X_t and a deterministic part S_t. Thus, the foundation of many price models is formed by

$$f(P_t) = X_t + S_t.$$

The parameters of the components are estimated as follows: First, the deterministic part is estimated, which is usually given by seasonal patterns of different lengths. Then, the parameters of the stochastic part X_t are estimated on the difference $f(P_t) - S_t$. In the following, after briefly describing electricity price characteristics, we introduce the variance-stabilizing transformation f and our models for the deterministic and stochastic parts, respectively.

2.3.1 Characteristics of Electricity Prices

Electricity prices have a tendency to revert to a mean which may be periodic, i.e. undergoes seasonal variations. Smaller movement around this mean exists due to imbalances in supply and demand. Furthermore, price or demand shocks lead to spikes, that is steep jumps which quickly return to the previous level. This happens due to several reasons that are unique to electricity markets: Supply and demand have to be matched at each point in time within a tolerance to ensure grid stability. Demand in short-term markets is price-inelastic, i.e. higher prices do not induce a lower demand. Furthermore, electricity is not storable in economic terms [31]. In recent years with rising renewable penetration, negative electricity prices can occur regularly: Sudden, hard to predict output from wind or solar generators combined with the inflexibility of conventional generators which cannot reduce their output quickly enough leads to an oversupply and thus negative prices [32]. We aim to capture these characteristics in our price model.

2.3.2 Variance-Stabilizing Transformations

To improve the parameter estimation of stochastic price models, it is common to apply an invertible function which reduces the volatility by stabilizing the variance [33]. The estimation and simulation then occur in the transformed space, after which the inverse transformation is applied to the simulation to obtain values from the original space.

Historically, the logarithmic transformation was used. In fact, the stochastic processes for electricity price simulations are often explicitly defined for logarithmic prices [22, 31, 34]. This was a suitable transformation during the time of release of these publications, which was justified by the fact that "negative electricity prices have rarely been observed" [31]. However, it cannot be used for today's frequently negative or near-zero electricity prices without modifications as the logarithm quickly approaches negative infinity below one and is undefined at zero and below. In [33], various alternative transformations are evaluated which are specifically designed for parameter estimation in the presence of negative electricity prices. One such transformation is the area hyperbolic sine transformation (called Asinh-transform in the following), which was originally proposed in [35]. It is defined as

$$Y_t = f(P_t) = \sinh^{-1}\left(\frac{P_t - \xi}{\lambda}\right),$$

where ξ, λ are fixed offset and scale parameters and Y_t denotes the transformed

price. Its inverse is given by

$$P_t = f^{-1}(Y_t) = \lambda \cdot \sinh(Y_t) + \xi.$$

In [35], the author recommends to set the offset ξ such that the data is centered around zero, while the scale parameter λ is determined experimentally by evaluating the visual appearance of the resulting transformed data. The Asinh-transform behaves asymptotically logarithmic, as the positive and negative parts are approximately logarithmic with an approximately linear part in between, close to zero. Moreover, the Asinh-transform shares a useful property with the log-transform: The stochastic differential equation defining an Ornstein-Uhlenbeck process (see Section 2.3.4) in the transformed space still has an explicit solution. For the log-transform, this solution can be expressed with the log-normal distribution. For the Asinh-transform, it is given by the Johnson SU distribution.

2.3.3 Deterministic Part: Seasonality in Electricity Prices

Electricity prices contain seasonality, i.e. repeating patterns of different lengths. Such patterns most notably occur on a daily, a weekly, and annual basis. A simple technique of modeling seasonality occurring within a length T time interval consists of determining a representation of this interval in averages, i.e. taking the mean or median values for each time step within this interval. Formally, the *median profile* of a length $L \in \mathbb{N}_+$ seasonal pattern is given by

$$S_t^L = \text{median}\left(\left\{P_{t'} : t' \in \mathcal{T}, (t' \mod L) = (t \mod L)\right\}\right). \tag{2.2}$$

Given a time series with five minute increments, the daily seasonality can be modeled using Eq. (2.2) with $L = 288$, the number of five minute increments within a day. Similarly, the weekly seasonality is modeled with $L = 2016$. Annual seasonality is often modeled with a sum of trigonometric functions (motivated by the Fourier decomposition) [36]. This *trigonometric function* technique is used for example in [31] where the annual seasonality is defined as

$$S_t^a = \alpha + \beta t + \gamma_1 \sin(2\pi t) + \gamma_2 \cos(2\pi t) + \delta_1 \sin(4\pi t) + \delta_2 \cos(4\pi t). \tag{2.3}$$

Since this model is linear in the parameters $\{\alpha, \beta, \gamma_1, \gamma_2, \delta_1, \delta_2\}$, they can be obtained using the least-squares method.

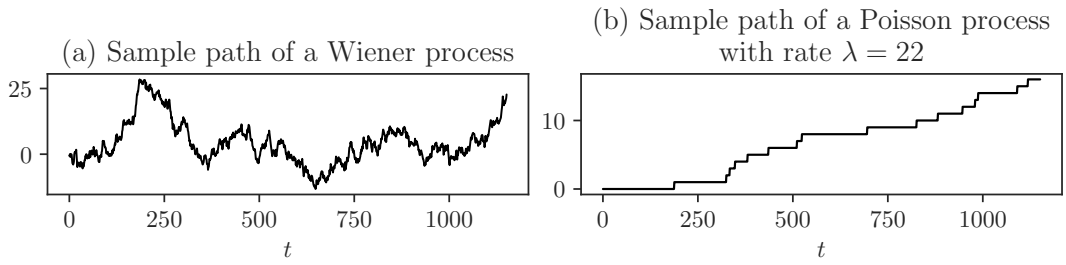


Figure 2.1: Sample paths of (a) a Wiener process and (b) a Poisson process.

2.3.4 Stochastic Part: Modeling Jump Diffusions

A stochastic process is a collection of random variables $\{X_t\}_{t\in\mathcal{T}}$ where the index set $\mathcal{T}$ represents time. It can also be viewed as a random function $X(\omega) : t \to X_t(\omega)$ or short X_t^ω, which defines a function of time for each realization ω of the uncertainty. We call ω a *scenario* and X_t^ω a *sample path*. For discrete-time processes $\mathcal{T}$ is finite or countably infinite, for continuous-time processes it is uncountably infinite. Stochastic processes can be used to model the evolution of some uncertain quantity over time [14, 37].

Two of the most important stochastic processes are the Wiener and the Poisson process, which are both part of our price model. A *Wiener process* or *Brownian motion* $\{W_t\}_{t\in\mathcal{T}}$ has independent, normally distributed increments. That is,

(i) for all $t > 0$ and $u > 0$ the increments $W_{t+u} - W_t$ do not depend on past increments and

(ii) $W_{t+u} - W_t \sim \mathcal{N}(0, u)$.

We can use W_t to represent the smaller movement around the mean electricity price. A *Poisson process* $\{J_t\}_{t\in\mathcal{T}}$ is a counting process, i.e. J_t counts the number of some events which occur by time t. These events occur independently of each other with a mean *rate* of $\lambda > 0$. It gets its name from the property that for an interval of length t, the number of events $J_{t+u} - J_u$ is Poisson distributed with mean λt. We can use a Poisson process to model the occurrence of price jumps. See Figure 2.1 for sample paths of a Wiener and a Poisson process.

Continuous-time stochastic processes are defined with stochastic differential equations (SDEs), which can be viewed as ordinary differential equations whose forcing function is a stochastic process. The solution of an SDE is a stochastic process [38]. We will now define the Ornstein-Uhlenbeck process, which will be part of our price model, in the form of an SDE. The Ornstein-Uhlenbeck process has the property that it evolves around a long-term mean to which it returns at a certain rate of *mean-reversion*. The SDE defining the

Ornstein-Uhlenbeck process is given as

$$dX_t = \kappa(\mu - X_t)dt + \sigma dW_t, \tag{2.4}$$

where $\kappa > 0$ represents the speed of mean reversion, μ is the long-term mean, σ is the short-term volatility parameter, and W_t is a Wiener process [23]. Analytical solutions to SDEs can be obtained using stochastic calculus if they exist. However, they can also be approximated in discrete time using the Euler-Maruyama (EM) method [39], which e.g. allows writing the continuous-time Ornstein-Uhlenbeck process (Eq. 2.4) as

$$X_{t+1} = X_t + \kappa(\mu - X_t)\Delta t + \sigma \Delta W_t,$$

where Δt is the discrete time increment. The discrete increment of the Wiener process ΔW_t can be represented as a random variable $\xi_t \sim \mathcal{N}(0, 1)$ with a multiplicative factor of $\sqrt{\Delta t}$ to account for the standard deviation of ΔW_t. Thus, we can write

$$X_{t+1} = X_t + \kappa(\mu - X_t)\Delta t + \sigma\sqrt{\Delta t}\xi_t, \tag{2.5}$$

which often forms the basis for parameter estimation and simulation of continuous-time processes[5] building on Eq. (2.4). The Ornstein-Uhlenbeck process itself can be calibrated on historic data using the least-squares method: By substituting $\kappa = 1 - \Phi$ and $\mu = \frac{\alpha}{1-\Phi}$ we can write Eq. (2.5) as

$$X_{t+1} = \alpha + \Phi X_t \Delta t + \sigma\sqrt{\Delta t}\xi_t$$

such that it attains the form of a linear regression model with intercept α, slope Φ, and normally distributed error term ξ_t. The short-term volatility σ is given by the standard deviation of the residuals.

The Ornstein-Uhlenbeck process sufficiently models how prices evolve around a long term mean. However, we also want to model the occurrence of price spikes, which we do in the following two sections.

Model 1: A Poisson Jump Model (PJM)

One possibility of including jumps is adding a Poisson process to the Ornstein-Uhlenbeck process in order to model the jump occurrences [34]. We define such a Poisson Jump Model (PJM) in continuous time as

$$\begin{aligned} f(P_t) &= X_t + S_t, \\ dX_t &= \kappa(\mu - X_t)dt + \sigma dW_t + \chi_t J_t, \end{aligned} \tag{2.6}$$

[5]This is because, naturally, the empirical data used for the estimation is defined over discrete time increments.

where f is a variance-stabilizing transformation, X_t is the stochastic component, and S_t is the seasonal component. In the stochastic component the parameters κ, μ, σ, W_t are from the Ornstein-Uhlenbeck process as in Eq. (2.4). Furthermore, the jumps are represented by the Poisson process J_t with constant intensity λ and normally distributed jump sizes $\chi_t \sim \mathcal{N}(\mu_J, \sigma_J)$. For small time increments Δt, the Poisson jump process can be approximated with a Bernoulli process, i.e. a sequence of Bernoulli trials with a consistent parameter $p \in (0, 1)$ [40]. This allows expressing the likelihood function of the PJM and thus its calibration using maximum likelihood estimation. For a detailed derivation and estimation results see Section A.2.

The PJM is not well-suited to describe the highly volatile and spiky prices of spot markets such as the NYISO real-time dispatch market: Since the jump component is included in the mean-reverting process X_t, extreme price spikes force the speed of mean-reversion κ to be very large such that prices can quickly return to the pre-spike level [23]. Moreover, due to the normally-distributed jump sizes it cannot model a difference in the probability of positive and negative jumps. However, it seems to be a suitable model for the less spiky prices of day-ahead markets, forward markets which often close the day prior to the day of delivery and define hourly prices. Calibration results and an exemplary sample path of the PJM given empirical NYISO day-ahead prices are displayed in Table A.1 and Figure A.1.

Model 2: A Poisson Spike Model (PSM)

Poisson Spike Models (PSMs) are better suited for modeling real-time markets as they separate the jump component from the mean-reverting component [23]. We adopt the model, but modify the calibration procedure from [29]. Formally, the price is modeled as

$$\begin{aligned} P_t &= \hat{P}_t + Z_t, \\ \sinh^{-1}\left(\frac{\hat{P}_t}{\lambda}\right) &= X_t + S_t, \\ dX_t &= \kappa(\mu - X_t)dt + \sigma dW_t, \\ dZ_t &= \chi_t J_t, \end{aligned} \tag{2.7}$$

where $\hat{P}_t$ is the mean-reverting component and Z_t is the jump component. The mean-reverting component is decomposed into a seasonal component S_t and a stochastic component X_t. Note that the stochastic component is modeled in the variance-stabilized Asinh-space (Section 2.3.2) with an Ornstein-Uhlenbeck process as defined in Section 2.3.4. The jump component Z_t is governed by a Poisson process J_t with jump sizes χ_t. Again, this Poisson process can be approximated with a sequence of Bernoulli trials [40]. Moreover, the jump sizes

χ_t follow an empirical distribution based on historical data. Calibrating this price model first requires detecting and removing spikes, which can be achieved with a variety of methods [41]. This also gives us an empirical spike probability $p \in (0, 1)$ for the Bernoulli trials. After the spikes are removed, we apply the Asinh-transform on the despiked prices $\hat{P}_t$ and then determine the seasonality S_t. Following this, we can estimate the parameters of the Ornstein-Uhlenbeck process on the difference $\hat{P}_t - S_t$, i.e. the stochastic part X_t, as described in Section 2.3.4.

Simulation of Model 2

Once the model parameters are estimated, its simulation is straightforward: The mean-reverting component $\hat{P}_t$ and the jump component Z_t are simulated separately. First, we simulate the Ornstein-Uhlenbeck process in Asinh-space by (a) sampling from the normal distribution to represent the short-term volatility and then (b) iteratively applying its discrete definition (Eq. 2.5). Then, we transform the simulated prices back to the original space by adding back the seasonality and applying the inverse Asinh-transform, thereby obtaining the mean-reverting component $\hat{P}_t$. Following this, we model the jump component by (a) obtaining samples from the Bernoulli distribution to model the occurrences and (b) obtaining samples from the empirical spike size distribution. Finally, we sum the mean-reverting and the spike component to obtain the simulated prices. See Algorithm 2 for the detailed procedure.

Algorithm 2: Simulation of Poisson Spike Model

Data: Parameters $\theta = \{\kappa, \mu, \sigma, p\}$, collection of spike sizes C^{spike}, time increment Δt, initial value X_0, length T, seasonality S.

Result: A length T price simulation.

$X_0 := \sinh^{-1}(P_0/30) - S_0$

$X := \begin{bmatrix} X_0, & 0, & \cdots & 0 \end{bmatrix}$

Obtain T samples $\xi_1, \ldots, \xi_T \sim \mathcal{N}(0, 1)$ from standard normal distribution.

`// Simulate Ornstein-Uhlenbeck process with Euler-Maruyama method.`

for $t = 0, \ldots, T-1$ **do**

  $X_{t+1} := X_t + \kappa(\mu - X_t)\Delta t + \sigma\sqrt{\Delta t}\xi_{t+1}$

$X := \begin{bmatrix} X_1, & \cdots, & X_T \end{bmatrix}$ `// Remove` X_0`.`

$\hat{P} := 30 \cdot \sinh(X + S)$ `// Add seasonality, invert Asinh-transform.`

Obtain T samples $b_1, \ldots, b_T \sim \text{Ber}(p)$ from Bernoulli distribution with parameter p.

Obtain T samples $u_1, \ldots, u_T \sim \mathcal{U}(1, |C_{\text{spike}}|)$ from uniform distribution.

$Z := \begin{bmatrix} b_1 \cdot C^{\text{spike}}_{u_1}, & \cdots & b_T \cdot C^{\text{spike}}_{u_T} \end{bmatrix}$ `// Define the spike component.`

$P := X + Z$

return P

Chapter 3

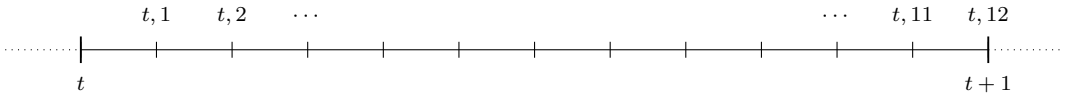
The Bidding Problem

We model the problem of bidding into the NYISO real-time market as an energy storage operator. As discussed in Section 2.2, bids and settlements on this market happen on different time scales, i.e. a single bid is subject to multiple settlements. We follow the simplifications detailed in [2], who model the same problem: Bids are placed one hour ahead of the hour-long delivery period. Within the delivery period, the bids for this period are settled in five period increments, so $M = 12$ times. Instead of submitting multiple bids with different price-volume combinations, we simplify the problem by choosing one sell and one buy bid, both with a fixed volume or capacity of 1 MW. Fixing either the price or the volume is common as it avoids non-linear and non-concave decision problems [13]. Furthermore, we assume that we are a price taker, i.e. that our actions do not have an effect on the market price. Note that we ignore degradation of the storage unit, which can e.g. occur with battery storage after many charge-discharge cycles [2]. Moreover, we do not include ramping times, the time needed to change the charging or discharging level, a constraint of pumped-hydroelectric storage systems [3].

We index the hourly periods with $t \in \mathcal{T}$ where $\mathcal{T} = \{0, \dots, T+1\}$. The first of the T decision stages is $t = 0$, while the last bidding decision is made at $t = T-1$. Due to the bidding rules explained above, this last bid is applied in period $T+1$; hence the unusual index set. The intra-hour settlements are indexed with $m \in \mathcal{M}$ where $\mathcal{M} = \{1, \dots, M\}$. We use this as follows for indexing variables: We write $x_{t,m}$ to denote the value of some variable x in stage t and after settlement m. Bids are placed after observing the last settlement of the previous delivery period, so we allow the notational trick that $t+1$ equals (t, M). This is better expressed visually in Figure 3.1.

Before formulating the problem as an MDP, we first introduce some notation and auxiliary functions. We build on [2], but improve notation where we deem it necessary: We describe the storage level of our energy storage system as a *resource state* $R_{t,m} \in \mathcal{R} = [R_{\min}, R_{\max}]$, where $R_{\min}$ and $R_{\max}$ are lower

Figure 3.1: Visualization of the indexing notation.



and upper limits. The system has constant charging and discharging efficiencies $\eta_c, \eta_d \in (0, 1)$. This does not reflect reality, but is a common simplification [4]. We adopt the notation

$$P_{(t,t+1]} = \begin{bmatrix} P_{t,1}, & P_{t,2}, & \dots & P_{t,M-1}, & P_{t,M} \end{bmatrix} \in \mathcal{P}^M$$

of [2] for the delivery period prices, where $\mathcal{P}$ is the set of possible prices. The possible prices at which we can bid lie in the interval $B := [0, B_{\max}]$ where $B_{\max} \in \mathbb{R}_+$. The lower limit is set to zero as it is always beneficial to buy at non-positive prices. The possible bidding decisions are then

$$\mathcal{B} = \{(b^-, b^+) \in B^2 : b^- \leq b^+\},$$

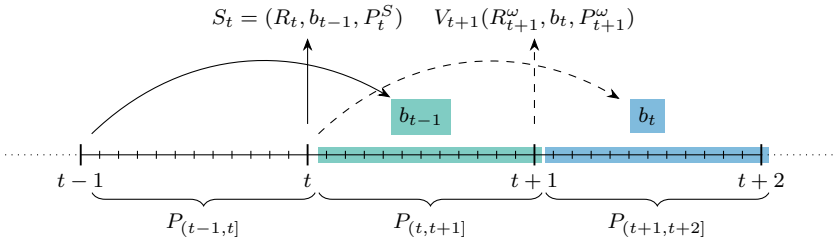
where b^- denotes the price of the buy bid, b^+ the price of the sell bid. Selling and buying at the same time is avoided by requiring the buy bid to be smaller than the sell bid. We use the notation $b \in \mathcal{B}$ to refer to both the buy and sell bid decision in the following. Moreover, we always implicitly bid a capacity of 1 MW. Since energy is traded in the unit [\$/MWh] but our bids are settled in time increments smaller than an hour, we actually sell or buy $\frac{1}{M}$MWh of energy in a single settlement. To describe the change in resource state in an individual settlement m within a delivery period t, we define the function

$$g(P_{t,m}, b, R_{t,m}) = \begin{cases} -\frac{1}{M}, & \text{if } P_{t,m} > b^+ \text{ and } R_{t,m} - 1/M \geq R_{\min}, \\ \frac{1}{M}, & \text{if } P_{t,m} < b^- \text{ and } R_{t,m} + 1/M \leq R_{\max}, \\ 0, & \text{otherwise.} \end{cases}$$

If we win a sell bid and our resource level is sufficient, we discharge. If we win a buy bid and the resource is not full, we charge. Similarly, to define the reward in an individual settlement m within a delivery period t we define the function

$$h(P_{t,m}, b, R_{t,m}) = \begin{cases} P_{t,m} \cdot \frac{1}{M} \cdot \eta_d, & \text{if } P_{t,m} > b^+ \text{ and } R_{t,m} - 1/M \geq R_{\min}, \\ -P_{t,m} \cdot \frac{1}{M}, & \text{if } P_{t,m} > b^+ \text{ and } R_{t,m} - 1/M < R_{\min}, \\ -P_{t,m} \cdot \frac{1}{M} \cdot \frac{1}{\eta_c}, & \text{if } P_{t,m} < b^- \text{ and } R_{t,m} + 1/M \leq R_{\max}, \\ 0, & \text{otherwise.} \end{cases}$$

Figure 3.2: A visualization of the Markov decision process.



If we win a sell bid we get paid for $\eta_d \cdot \frac{1}{M}$ MWh of energy, which takes the discharging efficiency of our storage unit into account. If we win a sell bid but cannot discharge, we must pay a penalty equivalent to buying back the energy from the market (following [2]). If we win a buy bid, we pay for $\frac{1}{\eta_c} \cdot \frac{1}{M}$ MWh of energy, where we take the charging efficiency into account. In other words, we trade the amount of energy that is or will be in our storage unit.

3.1 Markov Decision Process

Now we have all necessary tools to formulate our problem as an MDP. Again, the formulation is based on [2] with notational changes. See Figure 3.2 for a visualization of parts of this formulation.

- The *state* variable $S_t = (R_t, b_{t-1}, P_t^S) \in \mathcal{S}$, where R_t is the resource level of the energy storage system, $b_{t-1} \in \mathcal{B}$ is the hour-ahead bid settled in the current stage t, $P_t^S \in \mathcal{P}$ is the price which we observe at the beginning of the current stage.

- The *decision* $b_t \in \mathcal{B}$ which is determined at time t but settled in the interval $(t+1, t+2]$.

- The *exogenous information* P_t^ω given by some stochastic process modeling the energy prices where ω is a scenario of the outcome space Ω.

- The *transition function*

$$S_{t+1} = S^M(S_t, b_t, P_{(t,t+1]}) = (R_{t+1}, b_t, P_{t+1}^S)$$

  where the bid b_t is determined in the decision process, P_{t+1}^S is simply observed, and R_{t+1} is defined as follows: The resource levels within the

delivery period are given as

$$\begin{aligned} R_{t,1} &= R_t, \\ R_{t,m+1} &= R_{t,m} + g(P_{t,m}, b_{t-1}, R_{t,m}) \quad \text{for } m = 1, \dots, M-1, \end{aligned}$$

where the resource level of the subsequent stage R_{t+1} is equal to $R_{t,M}$ as per our notational convention. This recursive definition of the resource level reflects the sequential settlements. Note that the bid b_{t-1}, which is settled in stage t, is contained in S_t.

- The *contribution function*

$$C_{t,t+2}(S_t, b_t, P^{\omega}_{(t,t+2]}) = \mathbb{E}\big[\sum_{m=1}^{M} h(P^{\omega}_{t+1,m}, b_t, R^{\omega}_{t+1,m})\big]$$

where R^{ω}_{t+1} is determined over $P^{\omega}_{(t,t+1]}$ as for the transition function; hence the additional dependence on the interval $(t, t+1]$. The contribution function represents the expected reward for taking action b_t in the subsequent stage $t+1$.

- The *policy* $B^{\pi}_t : \mathcal{S} \to \mathcal{B}$, which maps each state to a bidding decision.

- The *objective function*

$$\max_{\pi \in \Pi} \mathbb{E}\Big[\sum_{t=0}^{T-1} C_{t,t+2}(S_t, B^{\pi}_t(S_t), P^{\omega}_{(t,t+2]}) \mid S_0\Big],$$

which maximizes the expected cumulative reward. The optimal policy can also be characterized with a value function using Bellman's equation:

$$V_t(S_t) = \max_{b_t \in \mathcal{B}} \Big\{ \mathbb{E}\Big[C_{t,t+2}(S_t, b_t, P^{\omega}_{(t,t+2]}) + V_{t+1}\big(S^M(S_t, b_t, P^{\omega}_{(t,t+1]})\big)\Big] \Big\} \qquad \forall t = 0, \dots, T-1,$$

$$V_T(S) = 0 \quad \forall S \in \mathcal{S}.$$

As the authors of [2] remark, the contribution and the value function in stage t are both expected values defined over the uncertain prices in the interval $(t+1, t+2]$. The uncertain prices in interval $(t, t+1]$ do not directly affect the reward, but only the subsequent resource state R_{t+1}.

The price and bid spaces can be considered finite, but very high-dimensional since market trading rules often dictate lower and upper price limits and a fixed smallest increment for (bid) prices. Thus, we cannot apply BDP but must formulate an approximate solution method.

3.2 Approximation with BADP-lattice

We lift the three curses of dimensionality with BADP-LATTICE, a BADP approach which uses a *scenario lattice* to represent the uncertainty given by the price process. We choose BADP as multiple publications have successfully applied it on energy storage problems [3, 5–7, 42]. This approach requires choosing suitable state and action space samples $\hat{\mathcal{S}} \subseteq \mathcal{S}$ and $\hat{\mathcal{B}} \subseteq \mathcal{B}$, such that the first two curses of dimensionality are lifted. Moreover, the exogenous information in the form of a stochastic price process is approximated by first obtaining a sample $\hat{\Omega} \subseteq \Omega$ of the outcome space, which gives us a finite number of sample paths P_t^ω for $\omega \in \hat{\Omega}$. Following [3, 43], we then create a *scenario lattice* of 50 paths by clustering the sample paths using k-means with the k-means++ initialization procedure [44] and determine their probability according to the cluster sizes. These *centroid paths* are then used to approximate the expectations contained in the MDP formulation, which is our approach of lifting the curse of the outcome space dimensionality.

BADP-LATTICE is given in Algorithm 3. We determine $\bar{V}$ and B^π by iterating backwards in time and iteratively determining the value function approximation and policy for each sampled state, considering only actions from the sampled action space. As mentioned in Section 2.1.2, it is practice to linearly interpolate the value function if an action or realization of the exogenous information leads to a state outside of the sampled state space. Due to the fixed bid volume of 1 MW, our problem is set up to only result in consistent resource states. Thus, only the realization of the sample price paths can result in a state outside of $\hat{\mathcal{S}}$.

3.3 Perfect Foresight Model

A theoretical upper bound for the reward of the MDP formulation can be obtained by removing the stochasticity and then determining the optimal policy. This setting, where all prices are known in advance, is called the *perfect foresight* setting. We define this model as a deterministic dynamic program using the same framework as before:

- The *state variable* $S_t = (R_t, P_{(t,t+1]}) \in \mathcal{S}$.
- The *decision* $b_t \in \mathcal{B}_t$ which is settled in the interval $(t, t+1]$.
- The *transition function*

$$S^M(S_t, b_t, P_{(t,t+1]}) = (R_{t+1}, P_{(t+1,t+2]}) = S_{t+1}$$

where R_{t+1} is defined as follows: The resource levels within the delivery period are given as

$$\begin{aligned} R_{t,1} &= R_t, \\ R_{t,m+1} &= R_{t,m} + g(P_{t,m}, b_t, R_{t,m}) \quad \text{for } m = 1, \ldots, M-1. \end{aligned}$$

The resource level of the subsequent stage R_{t+1} is equal to $R_{t,M}$.

- The *contribution function*

$$C_t(S_t, b_t) = \sum_{m=1}^{M} h(P_{t,m}, b_t, R_{t,m}).$$

- The *objective function*

$$\max_{\pi \in \Pi} \sum_{t=1}^{T} C_t(S_t, B_t^\pi(S_t)),$$

given an initial state $S_0 \in \mathcal{S}$. The optimal policy can also be characterized with a value function using Bellman's equation:

$$V_t(S_t) = \max_{b_t \in \mathcal{B}} C_t(S_t, b_t) + V_{t+1}\big(S^M(S_t, b_t, P_{(t,t+1]})\big), \quad \text{for } t = 0, \ldots, T-1,$$

$$V_T(S_T) = 0.$$

Algorithm 3: BADP-LATTICE

Initialize the terminal reward of the value function with
$\bar{V}_T(S) := 0 \quad \forall S \in \hat{\mathcal{S}}$.

for $t = T-1, \dots, 0$ **do**

  for $S_t = (R_t, b_{t-1}, P_t^S) \in \hat{\mathcal{S}}$ **do**

    Obtain sample paths $P^{\omega_1}_{(t,t+2]}, \dots, P^{\omega_{1000}}_{(t,t+2]}$ conditioned on P_t^S.

    Determine scenario lattice of *centroid paths* $P^{(1)}_{(t,t+2]}, \dots, P^{(50)}_{(t,t+2]}$ with k-means and their probabilities $\Pr(k)$, which are defined by the cluster sizes.

    Obtain resource states $R^{(1)}_{t+1}, \dots, R^{(50)}_{t+1}$ by applying the transition function (Eq. 3.1) for the interval $(t, t+1]$ with b_{t-1} and the respective centroid path out of $P^{(1)}_{(t,t+2]}, \dots, P^{(50)}_{(t,t+2]}$.

    Let

$$f(b_t) = \sum_{k=1}^{50} \Pr(k) \cdot \Big[\sum_{m=1}^{M} h(P^{(k)}_{t+1,m}, b_t, R^{(k)}_{t+1,m}) + \bar{V}_{t+1}(R^{(k)}_{t+1}, b_t, P^{(k)}_{t+1}) \Big]$$

    The transition function can lead to a state outside of the sampled state space only if $P^{(k)}_{t+1} \notin \hat{\mathcal{P}}$, i.e. $\hat{P}_{(n)} < P^{(k)}_{t+1} < \hat{P}_{(n+1)}$ for $\hat{P}_{(n)}, \hat{P}_{(n+1)} \in \hat{\mathcal{P}}$. In this case we linearly interpolate the value function by solving

$$\begin{aligned} &\frac{y - \bar{V}_{t+1}(R^{(k)}_{t+1}, b_t, \hat{P}_{(n)})}{P^{(k)}_{t+1} - \hat{P}_{(n)}} \\ =&\frac{\bar{V}_{t+1}(R^{(k)}_{t+1}, b_t, \hat{P}_{(n+1)}) - \bar{V}_{t+1}(R^{(k)}_{t+1}, b_t, \hat{P}_{(n)})}{\hat{P}_{(n+1)} - \hat{P}_{(n)}} \end{aligned}$$

    for y.

    Approximate the expected value of being in state S_t with the centroid paths and set the policy accordingly:

$$\begin{aligned} \bar{V}_t(S_t) &= \max_{b_t \in \hat{\mathcal{B}}} \{f(b_t)\}, \\ B^\pi_t(S_t) &= \arg\max_{b_t \in \hat{\mathcal{B}}} \{f(b_t)\}. \end{aligned}$$

return $\bar{V}$, B^π

Chapter 4

Numerical Study

To evaluate BADP-LATTICE, we conduct two experiments: The first experiment sets up a price process with finite support such that the optimal policy can be computed by BDP. Since we aim to approximate this solution, this allows assessing the solution quality of BADP-LATTICE. In the second experiment we model the bidding into the NYISO real-time market by using our more realistic Poisson Spike Process (Eq. 2.3.4). In this setting, the optimal solution cannot be explicitly computed. Hence, we compare BADP-LATTICE to the perfect foresight solution, which is an unattainable maximum.

In both experiments we use $T = 24$ decision stages representing the hourly decisions, i.e. our optimization horizon is a single day. Moreover, we start with an empty storage level. Given a price process, we define the *value* of a policy as the mean revenue over $|\Omega| = 1000$ price paths. This setup allows for a direct comparison with [2]. Note that this especially means that we do not evaluate our model with empirical price data, which is common for publications with a focus on modeling [5, 8, 11, 24–30]. Evaluation with out-of-sample empirical data would require using a competitive price forecast, which is for example done in a more recent publication [3]. Stochastic price models such as our Poisson spike model are not suited for forecasting, as they only intend to reproduce the characteristics of empirical prices [45].

4.1 Experiment 1: Approximation Quality

We reproduce the experiment of [2] with a finite support price process and $M = 1$, i.e. with no intra-hour settlements. In this setting there exists a closed-form expression for the value function as the expectations are defined over discrete random variables. Thus, we can use BDP to determine the optimal policy. We then use the optimal policy as a measure for the optimality of BADP-LATTICE, which can only sample from the price process. The price process with

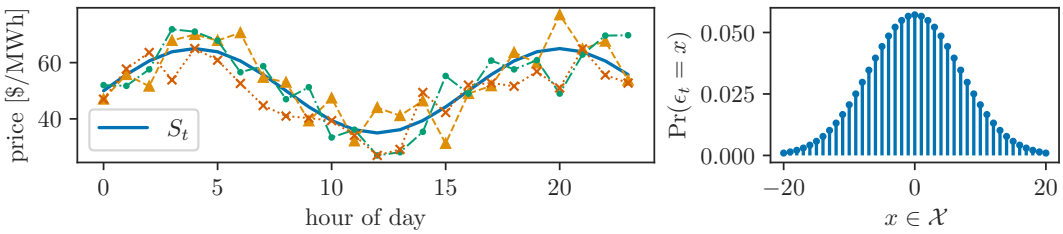


Figure 4.1: Visualization of finite support price process with pseudonormally distributed noise (see Equation 4.1).

finite support is defined as

$$\begin{aligned} P_t &= S_t + \epsilon_t, \\ S_t &= 15\sin(3\pi t/24) + 50, \end{aligned} \tag{4.1}$$

with a sinusoidal deterministic component S_t and a sequence of i.i.d. random variables ϵ_t from a discrete distribution representing the noise. The deterministic component emulates the typical characteristics of electricity prices in forward markets throughout a day, which often show higher prices in the morning and evening, while prices in the middle of the day are lower. Following [2], we use two variants for representing the noise while fixing the realizations of ϵ_t to $\mathcal{X} = \{-20, -19, \ldots, 19, 20\}$:

(a) The noise ϵ_t follows the discrete pseudonormal distribution with $\mu = 0$ and $\sigma^2 = 49$ whose probability mass function is defined as

$$\Pr(x) = \Pr(\epsilon_t = x) = \frac{f(x \,|\, \mu = 0, \sigma^2 = 49)}{\sum_{x' \in \mathcal{X}} f(x' \,|\, \mu = 0, \sigma^2 = 49)},$$

where f is the density function of the normal distribution.

(b) The noise ϵ_t follows the discrete uniform distribution where $\Pr(x) = \frac{1}{|\mathcal{X}|}$.

See Figure 4.1 for a visualization of the price process in Experiment 1(a). A state in this benchmark problem is given by $S_t = (R_t, b_{t-1})$ since the price process does not depend on its prior realizations but only on $t \in \mathcal{T}$. Let P_t^ϵ be the random variable defining the price at time t which depends on the noise ϵ_t. The closed-form expression for the value function is then given as

$$\begin{aligned} V_t(S_t) &= \max_{b_t \in \mathcal{B}} \left\{ \mathbb{E}\Big[C_{t,t+2}(S_t, b_t, \big[P_{t+1}^\epsilon, \quad P_{t+2}^\epsilon\big]) + V_{t+1}\big(S^M(S_t, b_t, P_{t+1}^\epsilon)\big) \Big] \right\} \\ &= \max_{b_t \in \mathcal{B}} \left\{ \sum_{x_1 \in \mathcal{X}} \Pr(x_1) \cdot \Big[\sum_{x_2 \in \mathcal{X}} \Pr(x_2) \cdot h(P_{t+2}^{x_2}, b_t, R_{t+1}^{x_1}) + V_{t+1}(R_{t+1}^{x_1}, b_t) \Big] \right\} \end{aligned}$$

where $R_{t+1}^{x_1} = g(P_{t+1}^{x_1}, b_{t-1}, R_t)$ and the terminal reward V_T is set to zero. Recall

Table 4.1: Experiment 1: Comparison of BADP-LATTICE to BDP regarding policy value and time elapsed for the computation. Percentage of optimality and time elapsed shown for BADP-LATTICE.

	value of policy [$] ($\epsilon_t$ pseudonormal)	value of policy [$] ($\epsilon_t$ uniform)	elapsed time [mm:ss]
BDP	86.46	137.95	06:55
BADP-LATTICE	79.96 [∼92%]	133.73 [∼97%]	00:36 [∼9%]

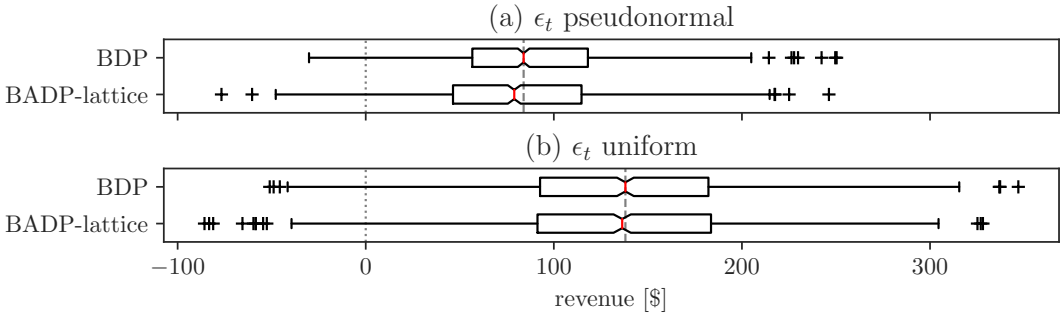


Figure 4.2: Experiment 1: Revenue of BADP-LATTICE compared to BDP over 1000 price paths. The box spans from the first quartile to the third quartile and shows the median in red. The whiskers indicate the minimum and maximum revenue lying within 1.5 times the interquartile range from the respective box boundary.

from Chapter 3 that the functions g and h represent the change in resource state and the reward within a settlement, respectively.

We follow [2] in setting $B_{\min} = 15, B_{\max} = 85$ and $|\hat{B}| = 30$ and moreover in evaluating the problem over $T = 24$ decision stages, i.e. one day. We set the resource space with $R_{\min} = 0, R_{\max} = 18$. Our state space thus has a cardinality of $|\hat{\mathcal{S}}| = 8854$. Again following [2], we ignore charging and discharging efficiency by setting $\eta_c = \eta_d = 1$.

4.1.1 Discussion of Results

The results of the experiment are shown in Table 4.1 and Figure 4.2. The table shows that in the two variants with pseudonormal and uniform noise BADP-LATTICE achieves ∼92% and ∼97% of the optimal policy value respectively while leading to a ∼91% reduction in time elapsed for the computation. This is on par with the MONOTONE-ADP approach in [2] and indicates that BADP-LATTICE comes sufficiently close to the optimal policy value.

Figure 4.2 shows the revenues resulting from applying the policies of BDP

and BADP-LATTICE on 1000 price paths. The sub-figures (a) and (b) match the two variants of the experiment. Note that the revenue spread in variant (b) with uniform noise is larger due to the higher probability of extreme price realizations compared to variant (a) with pseudonormal noise. In both (a) and (b) it is visible that BDP performs better than BADP-LATTICE, as it determines the optimal policy. In sub-figure (a), the BDP policy leads to a higher median revenue, and the first and third quartiles of revenues are higher as well. Moreover, while the lower whiskers of the BDP policy revenues are higher, the upper whiskers are lower than those of the BADP-LATTICE policy revenues. This is because the BDP policy is only optimal in expectation. Hence, it can still occur that the BDP policy is outperformed by the BADP-LATTICE policy for a specific price path or that it leads to a negative revenue. In sub-figure (b), the BDP policy again leads to a higher median revenue. However, the BADP-LATTICE policy performs better in this variant, as it comes closer to the median revenue as well as the first and third quartiles of revenues.

4.2 Experiment 2: NYISO real-time market

Now we evaluate BADP-LATTICE in a more realistic experiment, which simulates trading on the NYISO real-time market. First, we estimate the parameters of the Poisson Spike Model on historical NYISO real-time market prices in the bidding zone "North" over the years of 2019 to 2022, which are shown in Figure 4.3. We test how well the sample paths of the price process reproduce some statistical properties of the empirical data. In order to choose suitable values for sampling the state space, we evaluate the effect of sampling on the value of the policy in a separate experiment. Then, we can make an informed decision about the sample sizes and finally compare the value of the policy determined by BADP-LATTICE to that of perfect foresight, where all prices are known in advance.

We model a 5 MWh energy storage system with a capacity of 1 MW over $T = 24$ decision stages. Following [42], the storage system has constant charging and discharging efficiencies of $\eta_c = \eta_d = 0.9$ leading to a round-trip efficiency of 0.81. Due to the simplification that we always bid 1 MW for a five minute settlement, the resource is fully described by $|\mathcal{R}| = 61$ states, matching 60 increments of $\frac{1}{12}$ MWh and an additional empty state.

4.2.1 Price Process

We model the NYISO real-time prices with the Poisson Spike Model (Eq. 2.7). To do this, we estimate the parameters of the price model as follows: First, we detect and remove spikes in order to obtain the empirical spike distribution and probability as well as the despiked prices $\hat{P}_t$. We use the *fixed price thresholds*

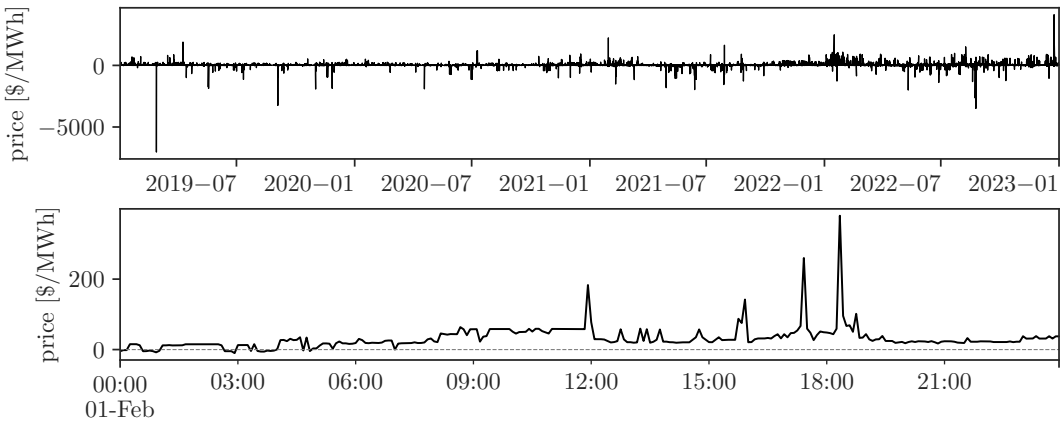


Figure 4.3: NYISO real-time dispatch prices in bidding zone "North" between 2019 and 2023 [46]. February 1, 2022 is shown as an exemplary day.

technique and classify the lower 1% and upper 4% of all prices as spikes, which are all prices below −$13.98 and above $84.46 in the used data set [41]. These bounds are chosen subjectively with the intention of defining a *normal* regime where prices evolve stochastically without (significant) jumps. The spikes are then removed by replacing them with annual seasonality S_t^a as defined in Eq. (2.3), hence the size of a spike at time t is given by $P_t - S_t^a$. The detected spikes form our empirical spike size distribution. Furthermore, the empirical spike probability naturally is 5% due to the chosen threshold. The result is shown in Figure 4.4. Then, we remove the seasonality of the despiked prices sequentially on the daily, weekly and annual scale as described in Section 2.3.3: For the daily and weekly scale we use the *median profile* technique, i.e. S_t^{288} and S_t^{2016}. The annual seasonality S_t^a is given by the *trigonometric function* technique. After removing the seasonality from the despiked prices, we can estimate the stochastic part X_t, which is modeled as an Ornstein-Uhlenbeck process (see Section 2.3.4). The resulting estimates are shown in Table 4.2. To evaluate our calibrated price model we compare statistical

Table 4.2: Parameters estimated for Poisson spike model on NYISO real-time dispatch prices between 2019 and 2023. The empirical spike size distribution is shown in Figure 4.4.

$\hat{\kappa}$	$\hat{\mu}$	$\hat{\sigma}$	p
0.1220	0.0242	0.1420	0.05

properties of its sample paths to those of the empirical data by evaluating the first four moments and the minimum and maximum values. This is similar to the approach in [23, 31], but instead of analyzing price returns[1] we evaluate

[1] A price return is the change in price over a period of time.

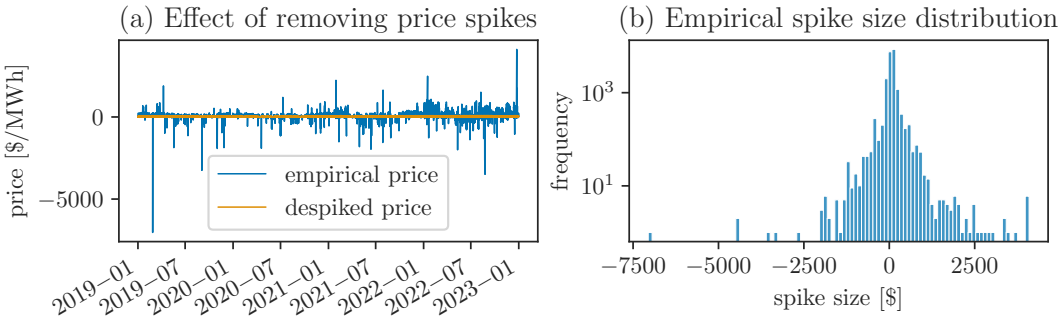


Figure 4.4: The result of applying the *fixed price thresholds* despiking procedure with lower and upper bounds −\$13.98 and \$84.46 on NYISO real-time dispatch prices. The empirical spike probability is $p \approx 0.05$.

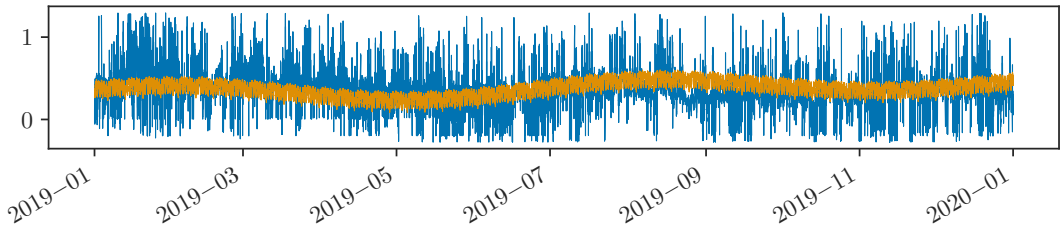


Figure 4.5: Despiked, variance-stabilized NYISO real-time dispatch prices $\sinh^{-1}(\hat{P}/30)$ and their seasonality S_t in the year of 2019.

the properties of the prices directly as we do not use our model for derivatives pricing. To do this, we simulate $|\Omega| = 250$ sample paths whose length matches those of the empirical prices, i.e. four years in five minute increments. We determine the first four moments and minimum, maximum values separately for each path, then take the mean as an estimate. The results are shown in Table 4.3. With the exception of the mean minimum value, the properties of our synthetically generated sample paths almost perfectly match those of the empirical data. The higher mean minimum can be explained by the low empirical probability of this particular spike in the beginning of the year 2019. In conjunction with the similar trajectorial properties shown in Figure 4.6 we conclude that the calibrated price model suffices for our purposes.

Table 4.3: Statistical properties of NYISO real-time dispatch price simulations in comparison to empirical data.

	mean	std	skewness	kurtosis	max	min
empirical	25.91	57.13	3.75	1575.37	4116.90	-7033.77
simulation	25.80	56.83	3.72	1585.70	4116.80	-6107.50

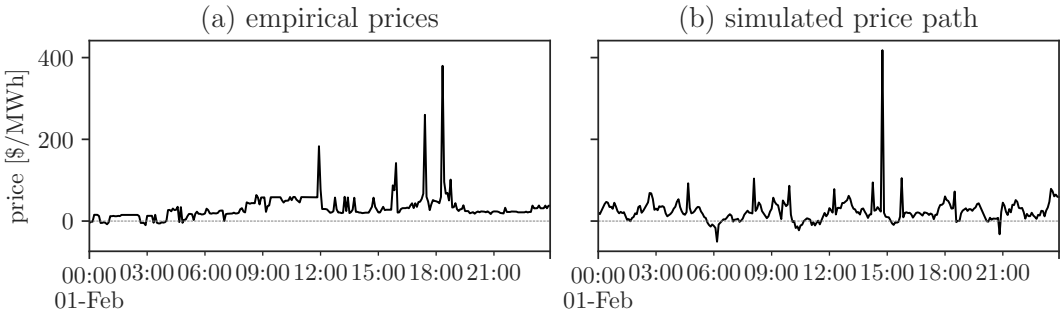


Figure 4.6: Empirical NYISO real-time dispatch prices on February 1, 2022 in comparison to a sample path from the calibrated Poisson spike model on the same day. The sample path covers the same time frame as the empirical path and was conditioned on the first empirical price in 2019.

4.2.2 Sampling State and Action Space

In order to apply BADP-LATTICE, we sample the price space $\mathcal{P}$ and the set of possible bid prices B from which the bid space $\mathcal{B}$ is created. We do so by first specifying lower and upper limits, then defining the sampled spaces via equidistant steps between those limits. This requires making informed decisions about (a) the lower and upper limits of the sampled space and (b) the size of the sampled space.

For (a), we follow [2] by choosing the lower and upper limits depending on empirical price data: 95% of prices lie within the interval $[-\$9.99, \$103.44]$. Thus, we set $\hat{\mathcal{P}}_{\min} = -\$9.99$, $\hat{\mathcal{P}}_{\max} = \$103.44$ and $\hat{B}_{\min} = \$0$, $\hat{B}_{\max} = \$103.44$. Furthermore, $\mathcal{B}$ always contains the bid $(0, \infty)$, i.e. "buy below zero, but never sell".

For (b), we run a numerical experiment. Note that the size of B has a particularly high impact on the computational complexity of BADP-LATTICE: The size of $\mathcal{B}$ is quadratic in the size of B. Furthermore, $\mathcal{B}$ forms both the action space and a subspace of the state space and hence results in $\mathcal{O}(|\mathcal{B}|^2) = \mathcal{O}(|B|^4)$ iterations in BADP-LATTICE.

To the best of the author's knowledge, it is practice in the literature to choose some sampling rate for the state and action space which is computationally feasible without considering the potential optimality gap[2]. Of course, the very reason why the state space is sampled makes such an experimental analysis difficult: It would require obtaining solutions of finely sampled and thus much larger problem instances. Hence, we choose to analyze the effects of sampling B and $\mathcal{P}$ on the policy value separately as follows:

1) Evaluate the effect of sampling B on the value of the perfect foresight

[2]One publication however proposes the use of their analytical solution of a bidding problem to investigate the effect of sampling on this gap [13].

Table 4.4: Step sizes resulting from equidistant sampling of B and $\mathcal{P}$.

	size of sample						
	5	10	15	20	30	45	60
$\hat{B}_{\text{step}}$ [\$]	25.86	11.49	7.39	5.44	3.57	2.35	1.75
$\hat{\mathcal{P}}_{\text{step}}$ [\$]	28.36	12.60	8.10	5.97	3.91	2.58	1.92

policy. We do this to ease the computation, since the complexity of BADP-LATTICE grows asymptotically in the size of $\hat{B}$ like a quartic polynomial. As the perfect foresight policy best exploits the difference between low and high prices, the results represent the minimum loss of arbitrage caused by sampling B. Nevertheless, it makes the choice of $|\hat{B}|$ for the next step less arbitrary.

2) Now we can make an informed decision about $|\hat{B}|$ and evaluate the effect of $|\hat{\mathcal{P}}|$ on the value of the BADP-LATTICE policy.

3) This, in turn, allows making a decision about $|\hat{\mathcal{P}}|$ and evaluating the effect of sampling B on the value of the BADP-LATTICE policy.

The sampling of $\mathcal{R}$ is not considered as it fully describes all possible resource states within our problem formulation. The results are shown in Figure 4.7, while the step sizes corresponding to the equidistant sampling are shown in Table 4.4. The figure displays the relation of the sample size to the percentage of the largest policy value over Ω within the experiment.

4.2.3 Discussion of Results

First, we discuss the results of the sub-experiment concerning the effect of the sampling from the previous subsection. Then, we make an informed choice about the parameters of BADP-LATTICE for the NYISO real-time market experiment, run this experiment, and discuss its results.

The Effect of Sampling on the Policy Value

Consider sub-figure 1) of Figure 4.7 showing the effect of sampling B on the policy value of perfect foresight. The value of the policy increases with the size of $|\hat{B}|$, converging quickly towards the maximum. The coarsest sample with $|\hat{B}| = 2$ already achieves ∼78% of the maximum policy value, the third coarsest achieves ∼98% with $|\hat{B}| = 10$. This is because most revenue is made from exploiting the difference between negative prices and positive price spikes, which is evident from the comparatively high revenue for $|\hat{B}| = 2$: Recall that

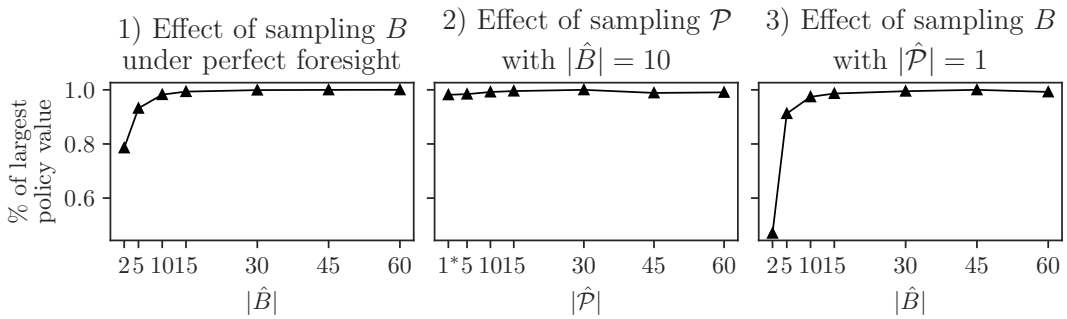


Figure 4.7: Effects of sampling B and $\mathcal{P}$ on policy value.

for $\hat{B}$ we set a lower limit of $\hat{B}_{\min} = \$0$ and an upper limit of $\hat{B}_{\max} = \$103.44$. Then, we create $\hat{B}$ by taking equidistant steps from $\hat{B}_{\min}$ to $\hat{B}_{\max}$. From $\hat{B}$, the set of possible bids $\hat{\mathcal{B}} \subset \hat{B}^2$ is created, which always contains the bid $(0, \infty)$. Thus, the coarsest sampling size $|\hat{B}| = 2$ only allows buying or selling below or above the limits respectively, as well as buying below \$0 and not selling. It still achieves 78% of the maximum policy value, showing that there is less value in doing arbitrage within the limits.

Now consider sub-figure 2) of Figure 4.7 showing the effect of sampling $\mathcal{P}$ on the policy value of BADP-LATTICE while fixing $|\hat{B}| = 10$. The size of $\hat{B}$ was chosen as it allows for a good compromise between policy value and problem size in sub-figure 1). The smallest run with $|\hat{\mathcal{P}}| = 1$ essentially removes the price dimension from the problem, as it allows only one price state $P_t^S = \$26$, which is equal to the mean of our empirical price data (see Table 4.3). The other runs sample $\mathcal{P}$ with fixed bounds and equidistant steps as before. The result might be surprising at first: There seems to be no connection between $|\hat{\mathcal{P}}|$ and the policy value, which is within more than 98% of the maximum for every chosen sample size. The variation seems to purely stem from noise, i.e. the price paths within BADP-LATTICE describing the uncertainty. The reason for this lies in both the structure of the bidding problem and the price process we use: The settlement prices are not restricted to $\hat{\mathcal{P}}$. When we choose a bid b_t at time t, we consider the uncertain prices of the interval $(t, t+2]$ with $2M$ settlements. Within this interval, only the price P_{t+1}^S must be restricted to $\hat{\mathcal{P}}$ in order to obtain the expected future reward from the value function at $t+1$. The uncertain resource state R_{t+1} and the uncertain contribution of our bid b_t both depend on prices which are not restricted to $\hat{\mathcal{P}}$. Recall from Section 3.1 that the future reward also depends on the uncertain resource state R_{t+1} and the bidding decision b_t. These two variables have a much more significant effect on the future reward than the price P_{t+1}^S: First, we already established that we reach more than 98% of the maximum policy value with $|\hat{\mathcal{P}}| = 1$, i.e. while ignoring the price space. Second, consider Figure 4.8, which displays the value function at a fixed time in relation to other parameters of

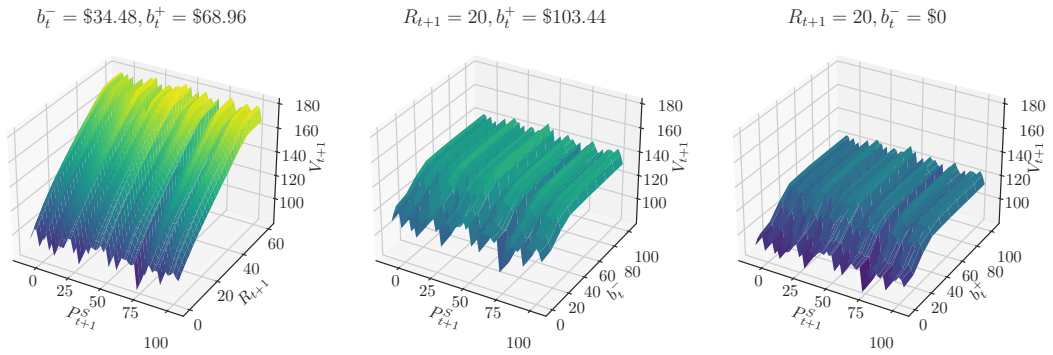


Figure 4.8: The effect of P_t^S on the value function at $t+1 = 12$ in relation to that of R_t, b_{t-1}^- and b_{t-1}^+. Other parameters are fixed as indicated in the sub-figure titles.

the state variable. Especially in the leftmost sub-figure, showing the effect of P_{t+1}^S in relation to that of R_{t+1}, it is evident that the value of being in the subsequent state depends much more strongly on the other parameters of the state variable. The variation due to P_{t+1}^S again seems to purely stem from noise. Third, we know from the effect of $|\hat{B}|$ that most revenue is made from price spikes. Apparently, BADP-LATTICE learns to exploit this regardless of the price space size. Most likely, this is specific to short-term market prices or rather the Poisson spike model we use, as it is defined to revert to a long-term mean regardless of the initial value (which is restricted to the price space).

Finally, consider sub-figure 3) of Figure 4.7 showing the effect of sampling B on the policy value of BADP-LATTICE while fixing $|\hat{\mathcal{P}}| = 1$ (as we know from sub-figure 2), the effect of $|\hat{\mathcal{P}}|$ on the policy value is insignificant). As expected, the effect of the size of $\hat{B}$ is stronger than in the perfect-foresight case of sub-figure 1): The lowest sampling size only achieves ∼45% of the maximum policy value. However, it still converges quickly, reaching ∼91% at $|\hat{B}| = 5$ and ∼97% at $|\hat{B}| = 10$.

Results of the NYISO Real-Time Market Experiment

Now, we can make an informed choice about the sampling parameters, which we set to $|\hat{\mathcal{P}}| = 1$ and $|\hat{\mathcal{B}}| = 30$ in order to obtain a feasible problem size. We conduct the experiment for the duration of a single day as in [2]. Following [3], we compare the value of the BADP-LATTICE policy to the value of the perfect foresight policy, which serves as an unattainable upper bound. The results are shown in Figure 4.9. A simulation of the policy determined by BADP-LATTICE for a single price path is shown in Figure 4.10. BADP-LATTICE achieves ∼82% of the perfect foresight policy value.

However, perfect foresight is an *artificial* benchmark since it naturally cannot be used to solve the stochastic bidding problem. To make conclusions

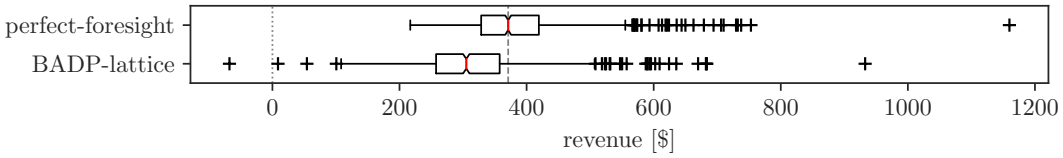


Figure 4.9: Experiment 2: Revenue of BADP-LATTICE compared to perfect foresight over 1000 price paths. The box spans from the first quartile to the third quartile and shows the median in red. The whiskers indicate the minimum and maximum revenue lying within 1.5 times the interquartile range from the respective box boundary.

about the performance of BADP-LATTICE in this setting, a state of the art approach should be used as a benchmark.

4.3 Evaluation of Performance

The main motivation for the MONOTONE-ADP approach in [2] is the computational difficulty of backward dynamic programming techniques such as our BADP-LATTICE. The authors justify this by comparing the CPU times[3] of their approach to those of a backward dynamic programming approach. However, they fail to provide details on the computer hardware and programming language used and do not publish their implementation. This is reason for concern, as running time can be drastically improved e.g. by using a compiled language, parallelizing code, and optimizing memory access, which could make backward approximate dynamic programming approaches viable. Moreover, this nondisclosure of information makes a direct comparison of algorithm performance impossible. Finally, their MONOTONE-ADP approach is computationally intensive as well, as it needs thousands of iterations to converge.

4.3.1 Discussion of Results

While we cannot say with absolute certainty that the elapsed times of BADP-LATTICE are comparable to those in the publication, we justify the comparison as follows: First, we solve the same MDP with the same action space and similarly-sized state spaces, which were reported in the publication. Secondly, as it happens, the author's rather outdated personal computer contains a CPU released three years before the publication of [2], which can thus be considered an "average" workstation CPU at that time. We report details on the computer hardware and implementation in the following subsection.

[3] "CPU time" denotes the actual amount of time a CPU has spend on a task which e.g. does not include time spent on other tasks due to context-switching.

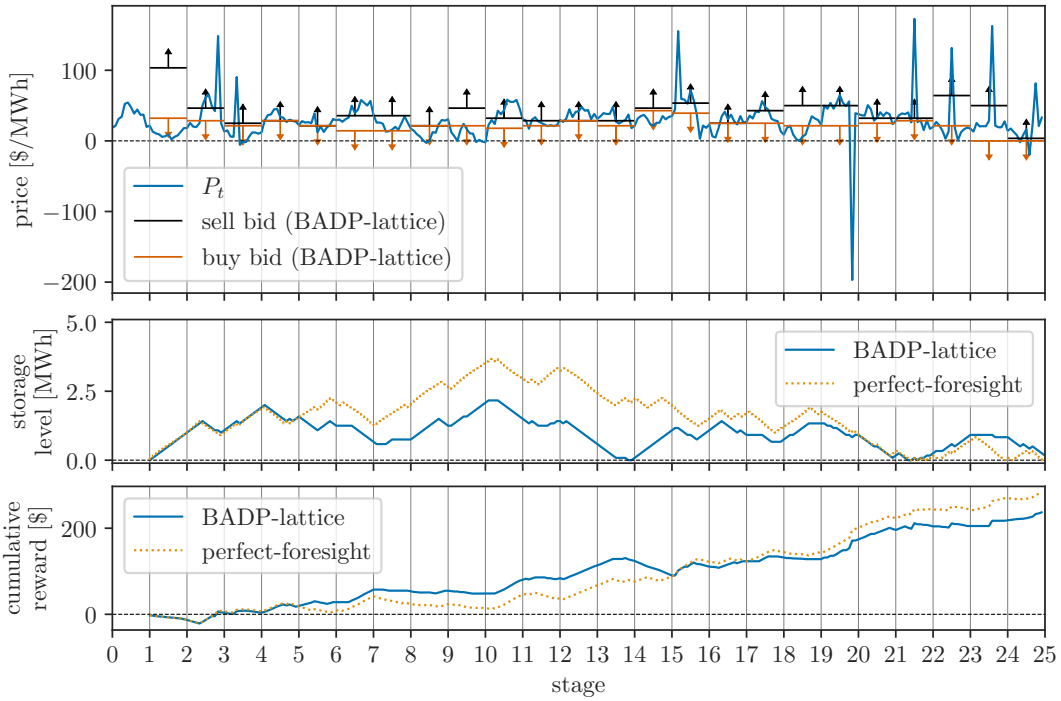


Figure 4.10: Experiment 2: Simulation of BADP-LATTICE and perfect foresight policies for a single day.

The comparison is shown in Figure 4.11. The figure shows the computation time for Experiment 1 (Section 4.1) with different state space sizes: The smallest run corresponds to the exact configuration of Experiment 1. A run with similar state space size is not contained in [2] since they have an additional dimension in their state space to take battery degradation into account. We thus enlarge the size of the resource space $\mathcal{R}$ to $|\mathcal{R}| = 47$ and $|\mathcal{R}| = 140$ in order to obtain similarly sized state spaces in the two larger runs. As the action spaces are also equivalent, the resulting computation times are comparable. Note that for our BDP and BADP-LATTICE we measure the elapsed time instead of the CPU time since parts of the implementation are parallelized. Because of this, the measured CPU time would be larger than the elapsed time as the workload is spread over the CPU cores, each of which would add to the CPU time.

Our BADP-LATTICE leads to a reduction of the computation time compared to MONOTONE-ADP in the two larger experiments of more than 99%. In absolute terms, this is an improvement from 16 hours and 42 minutes to 54 seconds and from 21 hours and 4 minutes to 2 minutes and 33 seconds. Even our BDP implementation is more performant than their approximation. However, we obtain a similar proportional reduction in computation time between our BDP implementation and BADP-LATTICE of $91 - 95\%$.

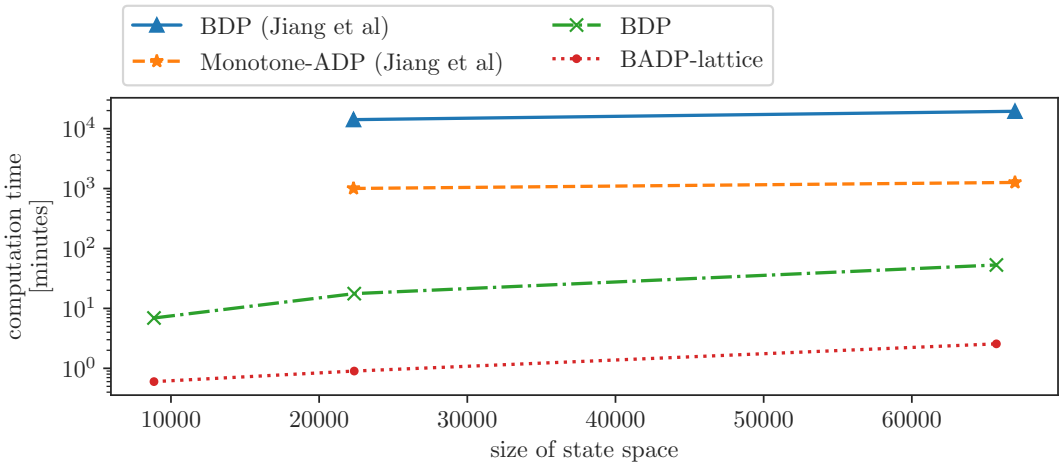


Figure 4.11: Computation times of BDP and BADP-LATTICE for variants of the benchmark problem in Experiment 1 (Section 4.1) compared to those of BDP and MONOTONE-ADP from [2].

4.3.2 Details on Implementation and Hardware

All experiments were run on a personal computer with a 3.3GHz Intel Xeon 1230v2 (released in 2012) and 16 GB of main memory. The implementation is available on GitHub [47]. The NYISO market data could not be included in the repository as its license does not permit redistribution, but it is openly available [46]. We use the programming language Python, but compile critical parts to performant machine code[4] using the package "Numba" [49]. The following modules were implemented:

- `markov_decision_process.py`: BADP-LATTICE (Numba), perfect foresight, procedures for simulating policies from both for sample paths.
- `jump_diffusion_process.py`: Estimation and simulation procedures for Poisson Jump and Poisson Spike Model (Numba).
- `kmeans.py`: k-means (Numba), k-means++ intialization procedure (Numba).

Parts of the implementation are parallelized, but BADP-LATTICE is not, although it would be possible to apply data parallelism by partitioning the state space: For a fixed $t \in \mathcal{T}$, the computation of the value function $\bar{V}_t$ for each individual state is independent from all other states. Executing the computation in parallel for the state space partitions would further decrease the CPU time.

[4] Usually, scientific software written in Python makes heavy use of "Numpy" [48], which also speeds up computation significantly. However, our specific problem was hard to formulate purely in Numpy's syntax.

4.4 Summary of Results

In this chapter we evaluate our approximation method BADP-LATTICE. We find that despite its relative simplicity compared to approaches like (forward) approximate dynamic programming, it performs well: In an experiment where the optimal solution is computable, it achieves $\sim 92 - 97\%$ of the optimal policy value while leading to a reduction of $\sim 91\%$ in computation time, which is comparable to [2]. In the subsequent experiment, which models bidding into the NYISO real-time market by calibrating a Poisson spike model on empirical prices, it achieves $\sim 82\%$ of the perfect-foresight policy value. Moreover, we evaluate the effect of parameter choices which are necessary to execute BADP-LATTICE on the policy value. The results show that the policy value quickly converges with the size of the sampled bid space $\hat{B}$, while the effect of the price space size $|\hat{\mathcal{S}}|$ can be neglected. The reason for the latter likely lies in the structure of the problem and the price process which is used. Furthermore, BADP-LATTICE beats MONOTONE-ADP from [2] in terms of performance by a large margin, as it requires less than 1% of the computation time.

Chapter 5

Conclusion

In this work we apply a backward approximate dynamic programming approach on the problem of an energy storage operator who bids into a real-time market. We build on a publication which models the same problem, but uses a more intricate (forward) approximate dynamic programming approach [2]. Despite the relative simplicity of our approach, we find that it performs just as well in terms of approximation quality. Furthermore, it leads to a similar reduction in computation time relative to the canonical backward dynamic programming approach, which computes the optimal solution in a stylized setting. Relative to the approach in [2], the optimized implementation of our approach shows a reduction in computation time of more than 99%. This shows that if computation time is a key motivator for developing an approximation method, it is a worthwhile endeavor to carefully implement a simpler approach first.

We also evaluate our approach in a more realistic setting which models the real-time market of the New York Independent System Operator. There, we compare it with an idealistic perfect foresight solution, where the stochasticity is removed. Our approach compares favorably, but it is difficult to draw conclusions from this experiment as the perfect foresight solution naturally is not applicable to the stochastic problem. Moreover, we neglect further constraints such as ramping times (pumped-hydro) or degradation of the storage unit (battery) in our problem formulation. Thus, some schedules could possibly be technically infeasible.

Future work should evaluate the approach against a state of the art approach for real-time bidding. Moreover, variants of the problem formulation which include battery degradation or ramping times should be considered to obtain more realistic decision rules. In the same spirit, one could also employ a competitive price forecast to represent the uncertainty, which would allow evaluating the approach using out-of-sample empirical data.

Appendix A

Appendix

A.1 Normalized Variance-Stabilized Transformation

In [33], the Asinh-transform is evaluated next to other variance-stabilizing transformations designed specifically for parameter estimation in the presence of negative electricity prices. Before applying the Asinh-transform, the authors of this publication normalize the prices using a robust normalization scheme which centers the prices around the median and scales them with the median absolute deviation. The median absolute deviation is adjusted by a factor of $\frac{1}{z_{0.75}} \approx 1.4826$, where $z_{0.75}$ is the 75% quantile of the standard normal distribution. This normalization is more robust to outliers than the more common normalization with the mean and standard deviation, and therefore is better suited for spiky electricity price data. After the normalization, the authors apply the Asinh-transform without offset or scale parameters.

Our experiments showed that the parameter estimation of our jump diffusion model is very sensitive to changes in the scale parameter λ, even after applying the robust normalization. However, the offset parameter was not needed, as the data is already centered by the robust normalization. Hence, our variance-stabilizing transformation is given by

$$P'_t = f(P_t) = \sinh^{-1}[\frac{1}{\lambda} \cdot \frac{z_{0.75}}{\text{median}(\sum_{t=1}^{T} |P_t - \text{median}(P)|)} \cdot (P_t - \text{median}(P))], \tag{A.1}$$

where λ is the scale parameter which can be adjusted depending on the data. The inverse transformation is given by

$$P_t = f^{-1}(P'_t) = \lambda \cdot \frac{\text{median}(\sum_{t=1}^{T} |P_t - \text{median}(P)|)}{z_{0.75}} \cdot \sinh(P'_t) + \text{median}(P). \tag{A.2}$$

A.2 Calibration of the Poisson Jump Model (PJM)

The PJM as defined in Eq. 2.6 can be calibrated as follows: Using the Ornstein-Uhlenbeck process discretization (see Eq. 2.5), this allows us to write the stochastic part in (2.6) as

$$X_{t+1} = \begin{cases} X_t + \kappa(\mu - X_t)\Delta t + \sigma\sqrt{\Delta t}\xi_t, & \text{with probability } (1-p), \\ X_t + \kappa(\mu - X_t)\Delta t + \sigma\sqrt{\Delta t}\xi_t + \chi_t, & \text{with probability } p. \end{cases} \tag{A.3}$$

Furthermore, we can define the likelihood function as a Bernoulli mixture of normal distributions:

$$\begin{aligned} &f(X_{t+1}|X_t, \theta) \\ &= (1-p)(2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(\frac{-(X_{t+1} - (X_t + \kappa(\mu - X_t)\Delta t))^2}{2\sigma^2}\right) \\ &\quad + p\,(2\pi(\sigma^2 + \sigma_J^2))^{-\frac{1}{2}} \exp\left(\frac{-(X_{t+1} - (X_t + \kappa(\mu - X_t)\Delta t + \mu_J))^2}{2(\sigma^2 + \sigma_J^2)}\right). \end{aligned} \tag{A.4}$$

The likelihood can then be used to calibrate the PJM on historic data using maximum likelihood estimation (MLE), i.e. by minimizing the negative log-likelihood function

$$\hat{\theta} = \text{argmin}_\theta - \sum_{t=1}^{T-1} \log f(X_{t+1}|X_t, \theta) \tag{A.5}$$

with the constraints $\kappa > 0, \sigma > 0, \sigma_J > 0, 0 \leq p \leq 1$.

Simulating the process now simply involves iteratively applying its discrete definition for T time steps with the estimated parameters θ, a certain starting value and samples from the distributions of the random variables. As per the definition of the PJM, we simulate in Asinh-space and use the inverse transformation to obtain prices in the original space. This is described in Algorithm 4.

Algorithm 4: Simulation of Poisson Jump Model

$P'_0 := f(P_0) - S_0$
$P' := \langle P'_0, \perp, \ldots, \perp \rangle$
$\xi_1, \ldots, \xi_T \sim \mathcal{N}(0,1)$ `// Obtain` T `samples from standard normal distribution`
$\xi^J_1, \ldots, \xi^J_T \sim \mathcal{N}(0,1)$ `// Obtain` T `samples from standard normal distribution`
$b_1, \ldots, b_T \sim \mathcal{B}(p)$ `// Obtain` T `samples from Bernoulli distribution with parameter` p
for $t = 1, \ldots, T-1$ **do**
  $P'_{t+1} := P'_t + \kappa(\mu - P'_t)\Delta t + \sigma\sqrt{\Delta t}\xi_{t+1} + b_t(\mu_J + \sigma_J \xi^J_{t+1})$
$P := f^{-1}(P' + S)$
return P

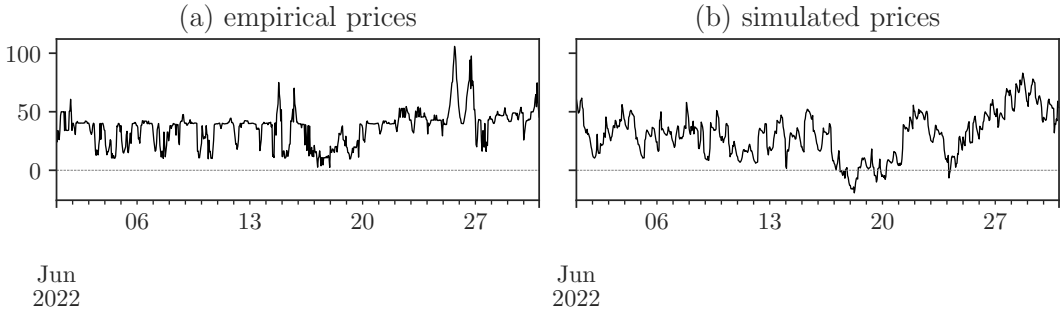


Figure A.1: Empirical NYISO day-ahead prices in June 2022 compared to a sample path from the calibrated Poisson Jump Model.

Table A.1: Parameters estimated for Model 1 on NYISO day-ahead prices in the years 2021 and 2022.

κ	μ	σ	p	μ_J	σ_J
376.8586	-0.0687	0.0421	0.2044	0.0072	0.1713

Bibliography

[1] Bismark Singh. *Optimal spatiotemporal resource allocation in public health and renewable energy*. PhD thesis, The University of Texas at Austin, 2016.

[2] Daniel R. Jiang and Warren B. Powell. Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS Journal on Computing*, 27(3):525–543, 2015. doi: 10.1287/ijoc.2015.0640. URL https://doi.org/10.1287/ijoc.2015.0640.

[3] Benedikt Finnah, Jochen Gönsch, and Florian Ziel. Integrated day-ahead and intraday self-schedule bidding for energy storage systems using approximate dynamic programming. *European Journal of Operational Research*, 301(2):726–746, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.11.010. URL https://www.sciencedirect.com/science/article/pii/S0377221721009565.

[4] Timm Weitzel and Christoph H. Glock. Energy management for stationary electric energy storage systems: A systematic literature review. *European Journal of Operational Research*, 264(2):582–606, 2018. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.06.052. URL https://www.sciencedirect.com/science/article/pii/S0377221717305933.

[5] Benedikt Finnah and Jochen Gönsch. Optimizing trading decisions of wind power plants with hybrid energy storage systems using backwards approximate dynamic programming. *International Journal of Production Economics*, 238:108155, 2021. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2021.108155. URL https://www.sciencedirect.com/science/article/pii/S0925527321001316.

[6] Joseph L. Durante, Juliana Nascimento, and Warren B. Powell. Backward approximate dynamic programming with hidden semi-markov stochastic models in energy storage optimization, 2020. URL https://doi.org/10.48550/arXiv.1710.03914.

[7] Bolong Cheng, Tsvetan Asamov, and Warren B. Powell. Low-rank value function approximation for co-optimization of battery storage. *IEEE Transactions on Smart Grid*, 9(6):6590–6598, 2018. doi: 10.1109/TSG.2017.2716382.

[8] Jochen Gönsch and Michael Hassler. Sell or store? an adp approach to marketing renewable energy. *OR Spectrum*, 38:633–660, 07 2016. doi: 10.1007/s00291-016-0439-x.

[9] Gilles Bertrand and Anthony Papavasiliou. Adaptive trading in continuous intraday electricity markets for a storage unit. *IEEE Transactions on Power Systems*, 35(3):2339–2350, 2020. doi: 10.1109/TPWRS.2019.2957246.

[10] Ioannis Boukas, Damien Ernst, Thibaut Théate, Adrien Bolland, Alexandre Huynen, Martin Buchwald, Christelle Wynants, and Bertrand Cornélusse. A deep reinforcement learning framework for continuous intraday market bidding. *Machine Learning*, 110, 09 2021. doi: 10.1007/s10994-021-06020-8.

[11] Goran Vojvodic, Ahmad I. Jarrah, and David P. Morton. Forward thresholds for operation of pumped-storage stations in the real-time energy market. *European Journal of Operational Research*, 254(1):253–268, 2016. ISSN 0377-2217. doi: 10.1016/j.ejor.2016.03.020. URL https://www.sciencedirect.com/science/article/pii/S0377221716301485.

[12] Tomasz Serafin, Grzegorz Marcjasz, and Rafał Weron. Trading on short-term path forecasts of intraday electricity prices. *Energy Economics*, 112:106125, 2022. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2022.106125. URL https://www.sciencedirect.com/science/article/pii/S014098832200281X.

[13] Benedikt Finnah. Optimal bidding functions for renewable energies in sequential electricity markets. *OR Spectrum*, 44, 03 2022. doi: 10.1007/s00291-021-00646-9.

[14] S.M. Ross. *Applied Probability Models with Optimization Applications*. Dover Books on Mathematics. Dover Publications, 2013. ISBN 9780486318646.

[15] Warren B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.07.014. URL https://www.sciencedirect.com/science/article/pii/S0377221718306192.

[16] Warren B. Powell. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. John Wiley & Sons, Inc., 2022.

[17] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

[19] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002. doi: 10.1137/S1052623499363220. URL https://doi.org/10.1137/S1052623499363220.

[20] Daniel S. Kirschen and Goran Strbac. *Markets for Electrical Energy*, chapter 3, pages 51–88. John Wiley & Sons, Ltd, 2 edition, 2019. ISBN 9781119213246.

[21] New York Independent System Operator. Transmission and dispatch operations manual, 2023. URL https://www.nyiso.com/documents/20142/2923301/trans_disp.pdf.

[22] Alfred Müller, Markus Burger, Bernhard Klar, Alfred Muller, and Gero Schindlmayr. A spot market model for pricing derivatives in electricity markets. *Quantitative Finance*, 4:109–122, 02 2004. doi: 10.1088/1469-7688/4/1/010.

[23] Jan Seifert and Marliese Uhrig-Homburg. Modelling jumps in electricity prices: Theory and empirical evidence. *Review of Derivatives Research*, 10:59–85, 02 2007. doi: 10.2139/ssrn.903804.

[24] Jae Ho Kim and Warren B. Powell. Optimal energy commitments with storage and intermittent supply. *Operations Research*, 59(6):1347–1360, 2011. doi: 10.1287/opre.1110.0971. URL https://doi.org/10.1287/opre.1110.0971.

[25] Daniel F. Salas and Warren B. Powell. Benchmarking a scalable approximate dynamic programming algorithm for stochastic control of grid-level energy storage. *INFORMS Journal on Computing*, 30(1):106–123, 2018. doi: 10.1287/ijoc.2017.0768. URL https://doi.org/10.1287/ijoc.2017.0768.

[26] Matt Thompson, Matt Davison, and Henning Rasmussen. Valuation and optimal operation of electric power plants in competitive markets. *Operations Research*, 52:546–562, 08 2004. doi: 10.1287/opre.1040.0117.

[27] Warren R Scott, Warren B Powell, and Somayeh Moazehi. Least squares policy iteration with instrumental variables vs. direct policy search: Comparison against optimal benchmarks using energy storage. *arXiv preprint arXiv:1401.0843*, 2014.

[28] Yangfang (Helen) Zhou, Alan Scheller-Wolf, Nicola Secomandi, and Stephen Smith. Electricity trading and negative prices: Storage vs. disposal. *Management Science*, 62(3):880–898, 2016. doi: 10.1287/mnsc.2015.2161. URL https://doi.org/10.1287/mnsc.2015.2161.

[29] Yangfang (Helen) Zhou, Alan Scheller-Wolf, Nicola Secomandi, and Stephen Smith. Managing wind-based electricity generation in the presence of storage and transmission capacity. *Production and Operations Management*, 28(4):970–989, 2019. doi: https://doi.org/10.1111/poms.12946. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/poms.12946.

[30] Somayeh Moazeni, Warren B. Powell, and Amir H. Hajimiragha. Mean-conditional value-at-risk optimal energy storage operation in the presence of transaction costs. *IEEE Transactions on Power Systems*, 30(3):1222–1232, 2015. doi: 10.1109/TPWRS.2014.2341642.

[31] Hélyette Geman and Andrea Roncoroni. Understanding the fine structure of electricity prices. *The Journal of Business*, 79, 05 2006. doi: 10.1086/500675.

[32] Joachim Seel, Dev Millstein, Andrew Mills, Mark Bolinger, and Ryan Wiser. Plentiful electricity turns wholesale prices negative. *Advances in Applied Energy*, 4:100073, 2021. ISSN 2666-7924. doi: https://doi.org/10.1016/j.adapen.2021.100073. URL https://www.sciencedirect.com/science/article/pii/S2666792421000652.

[33] Bartosz Uniejewski, Rafał Weron, and Florian Ziel. Variance stabilizing transformations for electricity spot price forecasting. *IEEE Transactions on Power Systems*, 33(2):2219–2229, 2018. doi: 10.1109/TPWRS.2017.2734563.

[34] Alvaro Escribano, J. Ignacio Peña, and Pablo Villaplana. Modelling electricity prices: International evidence. Working Paper Economic Series 08, Universidad Carlos III de Madrid, Departamento de Economia, 2002.

[35] Stefan Schneider. Power spot price models with negative prices. *Journal of Energy Markets*, 4(4):77–102, 2011. doi: 10.21314/JEM.2011.079.

[36] Rafał Weron. *Stylized Facts of Electricity Loads and Prices*, chapter 2, pages 25–65. John Wiley & Sons, Ltd, 2006. ISBN 9781118673362. doi: https://doi.org/10.1002/9781118673362.ch2. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118673362.ch2.

[37] Peter Tankov. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, 2004. doi: 10.1201/9780203485217.

[38] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019. doi: 10.1017/9781108186735.

[39] Mustafa Bayram, Tugcem Partal, and Gulsen Buyukoz. Numerical methods for simulation of stochastic differential equations. *Advances in Difference Equations*, 2018:17, 01 2018. doi: 10.1186/s13662-018-1466-5.

[40] Clifford A. Ball and Walter N. Torous. A simplified jump process for common stock returns. *Journal of Financial and Quantitative Analysis*, 18(1):53–65, 1983. doi: 10.2307/2330804.

[41] Joanna Janczura, Stefan Trück, Rafał Weron, and Rodney C. Wolff. Identifying spikes and seasonal components in electricity spot price data: A guide to robust modeling. *Energy Economics*, 38:96–110, 2013. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2013.03.013. URL https://www.sciencedirect.com/science/article/pii/S0140988313000625.

[42] Bolong Cheng and Warren B. Powell. Co-optimizing battery storage for the frequency regulation and energy arbitrage using multi-scale dynamic programming. *IEEE Transactions on Smart Grid*, 9(3):1997–2005, 2018. doi: 10.1109/TSG.2016.2605141.

[43] Nils Löhndorf, David Wozabal, and Stefan Minner. Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, 61(4):810–823, 2013. doi: 10.1287/opre.2013.1182. URL https://doi.org/10.1287/opre.2013.1182.

[44] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

[45] Rafał Weron. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, 30(4):1030–1081, 2014. ISSN 0169-2070. doi: https://doi.org/10.

1016/j.ijforecast.2014.08.008. URL https://www.sciencedirect.com/science/article/pii/S0169207014001083.

[46] New York Independent System Operator. Energy market & operational data, 2023. URL https://www.nyiso.com/energy-market-operational-data.

[47] Gereon Recht. Energy storage bidding, 2023. URL https://github.com/grecht/energy-storage-bidding.

[48] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[49] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift