DISSERTATION

Electricity markets under the influence of renewables: Modelling, prediction, and stochastic optimization

RIA GRINDEL

Am Fachbereich Mathematik der Rheinland-Pfälzischen Technischen Universität Kaiserslautern Landau genehmigte Dissertation zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.)

1. Gutachter: Prof. Dr. Ralf Korn
2. Gutachterin: Prof. Dr. Simone Göttlich

Datum der Disputation: 07.09.2023

DE-386

Rheinland-Pfälzische Technische Universität
Kaiserslautern Landau
Fachbereich Mathematik
Gottlieb-Daimler-Straße 47
67663 Kaiserslautern

Fraunhofer ITWM
Abteilung Finanzmathematik
Fraunhofer-Platz 1
67663 Kaiserslautern

Zusammenfassung

Im Rahmen des Zubaus erneuerbarer Energieträger in Deutschland verändert sich der deutsche Energiemix, der sich aus den in Deutschland vorhandenen Stromquellen zusammensetzt. Mit dem Wandel hin zu nachhaltigen Stromquellen wie Wind- und Solarenergie verändert sich auch die Situation, der sich der Strommarkt gegenübersieht. Während in der Vergangenheit wenig Unwägbarkeiten in der Stromerzeugung existierten und nur die Nachfrage stochastische Unsicherheiten aufwies, ist mittlerweile aufgrund der Wetterabhängigkeit auch die Erzeugung stochastischen Schwankungen ausgesetzt. Um für diese andersartige Situation einen unterstützenden Rahmen zu bieten, wurden am Strommarkt unter anderem der Intradaymarkt, Produkte mit halb- und viertelstündigen Zeitscheiben und ein verändertes Regelenergiemarktdesign eingeführt. Damit sind sowohl die Themen der Strompreisvorhersage als auch die der Optimierung auf den Strommärkten weiterhin aktuell.

Diese Arbeit beschäftigt sich zunächst mit der Modellierung des Intraday-Markts und der Prognose von Intraday-Indices. Dafür bewegen wir uns auf die Ebene der einzelnen Gebote am Intraday-Markt und modellieren mit diesen die Limitorderbücher der Intraday-Produkte. Basierend auf ausgewählten statistischen Kenngrößen der modellierten Limitorderbücher stellen wir einen neuartigen Schätzer für die Intraday-Indices vor. Gerade für Produkte mit weniger Liquidität enthalten die Orderbuchstatistiken relevante Informationen, die signifikant genauere Vorhersagen als der Vergleichsschätzer ermöglichen.

Da der Day-Ahead-Markt im Gegensatz zum Intraday-Markt als Markt mit täglicher Auktion betrieben wird, bietet er kleineren Unternehmen ohne eigene Handelsabteilung die Möglichkeit, sich am Strommarkt zu beteiligen. Aus deren Perspektive optimieren wir ihr Flexibilitätsangebot am Day-Ahead-Markt und modellieren dabei die Preise mithilfe eines stochastischen Mehrfaktormodells, welches bereits Einsatz in der Industrie findet. Um dieses Modell für die stochastische Optimierung aufzubereiten, wird eine Diskretisierung des Modells in Zeit und Raum mithilfe von Szenariobäumen vorgenommen. Hier stellen wir sowohl vorhandene Algorithmen zur Szenariobaumerzeugung als auch unsere eigenen Erweiterungen und Anpassungen vor. Diese basieren auf der Nested Distance, welche den Abstand zweier Verteilungen stochastischer Prozesse misst. Auf Basis der so entstandenen Szenariobäume wenden wir schließlich die stochastischen Optimierungsmethoden der stochastischen Programmierung, der dynamischen Programmierung und des Reinforcement Learnings an und untersuchen, in welchem Kontext die Methodiken jeweils geeignet sind.

Abstract

The German energy mix, which provides an overview of the sources of electricity available in Germany, is changing as a result of the expansion of renewable energy sources. With this shift towards sustainable energy sources such as wind and solar power, the electricity market situation is also in flux. Whereas in the past there were few uncertainties in electricity generation and only demand was subject to stochastic uncertainties, generation is now subject to stochastic fluctuations as well, especially due to weather dependency. To provide a supportive framework for this different situation, the electricity market has introduced, among other things, the intraday market, products with half-hourly and quarter-hourly time slices, and a modified balancing energy market design. As a result, both electricity price forecasting and optimization issues remain topical.

In this thesis, we first address intraday market modeling and intraday index forecasting. To do so, we move to the level of individual bids in the intraday market and use them to model the limit order books of intraday products. Based on statistics of the modeled limit order books, we present a novel estimator for the intraday indices. Especially for less liquid products, the order book statistics contain relevant information that allows for significantly more accurate predictions in comparison to the benchmark estimator.

Unlike the intraday market, the day ahead market allows smaller companies without their own trading department to participate since it is operated as a market with daily auctions. We optimize the flexibility offer of such a small company in the day ahead market and model the prices with a stochastic multi-factor model already used in the industry. To make this model accessible for stochastic optimization, we discretize it in time and space using scenario trees. Here we present existing algorithms for scenario tree generation as well as our own extensions and adaptations. These are based on the nested distance, which measures the distance between two distributions of stochastic processes. Based on the resulting scenario trees, we apply the stochastic optimization methods of stochastic programming, dynamic programming, and reinforcement learning to illustrate in which context the methods are appropriate.

Danksagung

Ich möchte die Gelegenheit nutzen, mich bei all den Menschen zu bedanken, ohne die diese Arbeit nicht möglich gewesen wäre.

Zunächst gilt mein Dank meinem Doktorvater Ralf Korn und der RPTU Kaiserslautern für das Ermöglichen meiner Doktorarbeit. Ralf hat mich im Laufe dieser Arbeit stets mit hilfreichen Treffen und Diskussionen, mit dem mich immer wieder zum Schreiben zurückbringen, mit Korrekturlesen meiner Texte in unmöglich schneller Zeit und teilweise über Wochenenden hinweg, und natürlich mit vielen guten und wichtigen Anmerkungen sehr unterstützt und diese Arbeit überhaupt erst ermöglicht. Danke dafür! Vielen Dank auch an meinen Zweitbetreuer Jörn Saß für seine interessierten Nachfragen, die mich weiter zum Nachdenken angeregt haben.

Außerdem möchte ich meiner Abteilung, der Finanzmathematik am Fraunhofer ITWM, danken - natürlich einerseits für die finanzielle Unterstützung, aber andererseits auch, dass ich durch mein Team immer Lust hatte, auf die Arbeit zu kommen, auch wenn es bei der Doktorarbeit mal weniger gut lief als gewünscht. Zu wissen, dass ihr auf unserem Flur anzutreffen sein werdet, und stets ein offenes Ohr für mathematische Fragen oder eine leere Kaffeetasse für eine gemeinsame Kaffeepause und ein privates Gespräch haben würdet, finde ich nicht selbstverständlich und danke ich euch allen. Insbesondere auch herzlichen Dank an meinen Abteilungsleiter Jörg Wenzel, der viel zu dieser schönen Abteilungskultur bei uns beiträgt.

Speziell benennen möchte ich außerdem Simon Schnürch und Florian Schirra, die sich mit mir darauf eingelassen haben, wöchentliche Treffen zum Stand der Doktorarbeit zu machen, was für mein Vorankommen essentiell war. Danke euch dafür, und dass ihr die wunderbaren Freunde seid, die ihr eben seid. Letzteres gilt auch für Robert Sicks als besten Büromitbewohner, den man sich vorstellen kann, und für Felix Riexinger und Till Heller, bei denen ich mich für Spaziergänge, wenn frische Luft und eine Pause nötig waren, als auch für Diskussionen, die meine Doktorarbeit vorangebracht haben, bedanken muss. Ihr wart da, wenn ich euch gebraucht habe. Danke!

Schließlich, an letzter und doch wichtigster Stelle, möchte ich mich auch bei meinen weiteren engen FreundInnen (Eva und Christine, ich denke hier sehr an euch), meiner Familie und natürlich meinem Freund Jonas bedanken. Ihr alle wart für mich da, wenn es mir gut, und viel wichtiger, wenn es mir schlecht ging. Eure emotionale Unterstützung kann ich gar nicht in Worte fassen, und will nur sagen: Ich bin unendlich dankbar, euch in meinem Leben zu haben. An Jonas zusätzlich noch ein paar Worte: Du hast mich scheinbar problemlos ausgehalten, wenn ich von der Arbeit gestresst und am nörgeln war; du hast immer an mich geglaubt und mich ermutigt und unterstützt, ohne dass ich dich darum bitten musste; du hast mit mir das Leben neben der Doktorarbeit genossen. Und außerdem hast du die ganze Arbeit Korrektur gelesen! Danke für alles.

Contents

List of Figures

List of Tables

1 Introduction

The 1970s saw the birth of a discourse on energy imports and security of supply in Germany, due to the 1973 oil crisis [68]. In the years that followed, this discourse developed into a call for an "energy turnaround" (Germ. "Energiewende"), which included the use of renewable energy sources. Finally, in 1990, the Electricity Feed Act (Germ. Stromeinspeisungsgesetz) provided the first major support for renewable energy in Germany, even though that was not explicitly expressed [68]. This law made it compulsory for energy supply companies to purchase electricity from renewable sources, and in addition, this electricity had to be remunerated accordingly, which had not been the case before. This led to the first major expansion of renewable electricity generation in Germany and started a trend, which accelerated significantly from 2001 onward, when the Renewable Energy Sources Act came into force. Following the (re)adoption of Germany's nuclear phase-out by 2022 in the wake of the Fukushima nuclear disaster in 2011, the expansion of renewable electricity suppliers continued. As of mid-2022, 146.8 GW out of a total of 225.6 GW of installed net nominal capacity has already been accounted for by renewables [35]. This is largely made up of solar energy (66.5 GW) and wind energy (67.3 GW). All shares are collectively presented in Table 1.1. Based on this installed capacity, renewables generated 244.2 TWh in 2022 from a total of 549.8 TWh of electricity generation in Germany, thus accounting for 44.4%. The majority of this was provided by wind power plants, which accounted for a remarkable 22.5% of the 44.4%. Other major contributors were photovoltaic with 10.6% and biomass with 7.6% [35].[1]

These figures mean increasing independence from conventional energy sources, but there is a catch: unlike electricity generated from fossil fuels, electricity from photovoltaic and wind is weather-dependent and therefore much more difficult to plan. It also cannot be generated at will. Because of these two characteristics of some renewable electricity generation techniques, there is volatility in the supply of electricity. Although unplanned outages also occur in conventional power generation, they are the exception rather than the rule. With photovoltaic (PV) and wind, on the other hand, the exact number of generated kilowatt-hours can usually only be predicted shortly before they are actually fed into the grid. This uncertainty is a serious issue because the grid itself cannot store electricity, which would not be a problem if there were a sufficient amount of electricity storage available. But this is not the case: There are some pumped storage hydro power stations for electricity storage and some batteries, but they are far from

[1] The presented numbers vary depending on the data source; e.g. [6] state that 506.8 TWh of electricity were generated in 2022 in Germany. Furthermore, they announced a renewable share of 48.3%. As these numbers were published in early January 2023, we assumed the values of [35], that stem from the end of February 2023, to contain corrections.

Technology	Installed Capacity (GW)	Produced Electricity (TWh)
Hydro	4.94	15
Pumped Storage	9.78	6.3
Biomass	8.96	41.9
Uranium	4.06	32.8
Brown Coal	18.90	107.9
Hard Coal	19.04	61.9
Mineral Oil	4.72	1.3
Natural gas	32.09	89.2
Wind onshore	58.23	98.7
Wind offshore	8.13	24.8
Solar	66.50	58.3
Other	-	18

Table 1.1: Installed capacity and produced electricity in Germany over several energy sources, presented for the year 2022 [35].

being able to absorb all the imbalances.[2] Consequently, feed-in and off-take must always be in balance, otherwise the grid will be damaged (e.g. by overheating) and any difference will cause the actual grid frequency to deviate from the target frequency. The frequency of the interconnected European grid, now stretching from Portugal to Turkey and Denmark to Western Sahara, is 50Hz [31]. If this grid frequency becomes too high or too low because electricity production and demand do not match, large generators have to be shut down to avoid major damage. This can ultimately lead to a blackout, i.e. a large-scale power failure.

The increased uncertainty on the production side is therefore a very serious issue, which was also recognized as such in 2011. Then, ENTSO-E, the European Network of Transmission System Operators for Electricity, formulated the following [9]:

> ”Increasing amounts of variable renewable generation will reduce the availability of traditional balancing resources which tends to increase the cost of maintaining system security. Higher levels of imbalances will occur at increasing levels of renewable generation which will lead to increased short term balancing costs.”

Let us take a closer look at this statement: Balancing energy is used to compensate for deviations from forecasted production or demand levels. To do this, so-called balancing resources are activated in the amount of the missing difference, i.e. in case of missing production, electricity producers are switched on or consumers are switched off, and in case of missing demand, consumers are switched on or producers are taken off the grid.

[2] In 2022, Germany had access to 9.78 GW pumped storage, 4.03 GW battery storage for power and 6.02 GWh of battery storage capacity [35].

Since the demand side has always been volatile and largely inelastic in its demand, and thus unable to react to price fluctuations in the market, the production side has been the main player in balancing out fluctuations and differences. The fact that this side now also has its own fluctuations has further increased the need for ways to compensate for these fluctuations and is the reason for the ENTSO-E statement.

However, in 2019, [38] noted the following:

> ”Previous studies have noted that, unexpectedly, Germany’s dramatic expansion of wind and solar energy coincided with a reduction of short-term balancing reserves. This observation has been dubbed the ‚German Balancing Paradox‘. [...] Since 2011, wind and solar energy have nearly doubled while both reserve requirements and reserve use have declined by 50%. [...] One reason for reduced balancing needs: increased and improved short-term wholesale electricity trading on the intraday market.”

So, contrary to ENTSO-E’s fears, demand for balancing energy has not increased but actually decreased. [38] attribute this to a change in the design of the electricity market. We return to the statement from ENTSO-E in [9] to shed more light on this: In fact, it did not end by painting a bleak future for energy technology. Instead, it put forward a number of ideas on how markets could be adapted to make it as easy as possible to integrate renewables into the system. The following suggestions were made:

- Make generators that deviate from their forecasts financially responsible for these deviations.
- Introduce cross-border balancing markets.
- Allow negative market prices, as these can occur as a natural consequence of market decisions.
- In the event of negative market prices, link the feed-in of renewable energy to market prices.
- Make the day ahead and intraday markets more flexible regarding gate closing times and time resolution to be able to deal with short-term deviations.
- Encourage the participation of renewable energy providers in the balancing energy market.

These proposals have all been taken on board and have significantly changed the market landscape in Germany and Europe over the last 12 years: Negative prices have been common for a few years now, renewable energy suppliers are more closely linked to the market price in periods of negative prices and differ from their usual remuneration in these situations, virtual power plants have emerged that serve as pools for renewable energy and are active in balancing energy supply, the first cross-border markets for renewable energy are already outdated, and the electricity markets have increased the

number of products offered. These developments form the basis of the German balancing paradox.

Despite this development, it is questionable whether the flexibility gained through the implemented measures will be sufficient to continue to absorb the fluctuations that occur in the grid. Germany plans to go green and become carbon neutral by 2045, with the EU following suit by 2050 [10]. In addition, the share of renewable energies in the electricity sector is to be increased to 80% by 2030. These efforts will result in a further significant increase in the amount of renewable energy being fed into the grid, and thus also in the amount of weather-induced fluctuations that need to be absorbed. In January 2023, ENTSO-E again expressed its views on this issue [10]: The decarbonisation of Europe and the associated sharp increase in the number of renewable energy suppliers in the grid pose major challenges to maintaining system stability, not least because the reactive power in the grid is decreasing due to fewer and fewer rotating masses of synchronous generating units. The latter are relevant when demand and supply do not match as they are able to instantly compensate for small mismatches, and they are usually contained in big conventional power plants. With fewer synchronous units on the grid, more alternative sources of compensation must be available and take effect to maintain grid frequency. The ENTSO-E considers the market to be a very important factor in this respect [10]:

> ”The availability of the necessary technical capabilities of grid users and the consistent improvement of Europe’s electricity market to ensure a reward system for system flexibility solutions and incentives for market participants to act in line with system needs remain key priorities.”

This also applies because high flexibility requirements will be normal due to the expansion of wind and PV. The proposed solution includes ”[...] new roles for thermal plants, RES participation, demand side response and storage” [10]. Consequently, it can be concluded that both electricity markets and demand-side management will play an important role in stabilizing the system in a decarbonised Europe and will continue to evolve with the demands placed on them.

This thesis is situated in the context of this changing energy landscape and contributes its share to support the shift towards more renewable energy in the energy mix. In addition, our goal is to be accurate in our modeling so that our results are easily applicable in practice. In doing so, we take into account that sometimes we can only approximate the solution.

We first introduce the reader to the structure of the day ahead and intraday markets in Chapter 2. Due to their temporal proximity to actual consumption and generation, these two markets are essential for renewables. The most liquid short-term market in Germany is the day ahead market that is introduced in Section 2.1. Due to its auction format, not only companies with their own trading department for electricity can participate in this

market, but also smaller businesses. We describe the structure of the market, explain its importance and go into detail about the pricing mechanisms and the merit order curve.

The intraday market, on the other hand, is used for the very short-term balancing of forecast changes in the feed-in of renewable energy sources or in the consumption forecasts. In Section 2.2, we give a brief overview of the market structure, its evolution and typical price patterns. In addition, we will introduce the topic of intraday indices, with a special focus on the ID_3. The ID_3 is an electricity price index that represents the volume-weighted average price of the last three national trading hours for products traded on the intraday market.

Indices such as this one provide an indicator of the fair price of a product in an environment of highly volatile prices in the intraday market for traders who need to compensate for a change in forecast. Accurate forecasting of this index therefore gives traders the ability to correctly classify and react to prices in the market, thereby reducing overall uncertainty. In addition, the index can serve as a hedging product for power producers. Therefore, in Chapter 3, we present a novel estimator for the ID_3 based on limit order book data, which provides significantly more accurate predictions than the comparison estimator for some of the products. First, in Section 3.1, we detail the data used to model the ID_3. Then, in Section 3.2, we show how to build a model of the limit order book from raw order data. The resulting model provides the basis for our initial estimate of the ID_3 price,

$$
\begin{aligned}
{}_{\text{full}}\hat{\text{ID}}_3^{d,i} \quad = \quad & a_1 \sum_{s_j \in T_{\text{mid}}^{d,i} \cup\, g(d)} m_{s_j} \frac{(s_{j+1} - s_j)}{g(d) - f(d)} + \\
& a_2 \sum_{s_j \in T_{\text{spread}}^{d,i} \cup\, g(d)} \eta_{s_j} \frac{(s_{j+1} - s_j)}{g(d) - f(d)} + \\
& \sum_{v=1}^{6} \left(a_{k+2} \sum_{s_j \in T_{\text{skew}}^{d,i,v} \cup\, g(d)} SK_{s_j}(v) \frac{(s_{j+1} - s_j)}{g(d) - f(d)} \right) + \\
& \sum_{t \in T_{\text{trans}}^{d,i}} \frac{p_t v_t}{\sum_{t \in T_{\text{trans}}^{d,i}} v_t}.
\end{aligned}
$$

It uses many key figures derived from the limit order book, which are explained in detail in Section 3.3. Furthermore, two naive forecasting models plus their regression adaptations are introduced. Finally, Section 3.4 puts all theory from before to practice. First, it is found that regarding all introduced limit order book statistics, only weighted mid prices have a significant correlation with ID_3 prices. In order to keep the calibration from putting weight in places that would reduce the out-of-sample performance of the estimator, it is first reduced to the time-weighted mean price, and then compared in its performance to a benchmark estimator. We find that for hourly products, both estimators perform similarly. For quarter hourly products though, our estimator's accuracy in forecasting the ID_3 prices is significantly higher than that of the benchmark estimators.

Finally, an analysis of the regression adaption of the estimator is done in case of extreme price regimes. The findings from this chapter are concluded in Section 3.5: Our analysis has indeed found a tool to boost the performance of ID_3 estimators, and thus a tool to reduce uncertainty for traders in a volatile market with renewable energy sources.

In Chapter 4, we turn our attention to the day ahead market. This market is used to optimize power portfolios in the short term, to take positions on renewable power generation, and to determine power plant schedules for the next day. It can also be used to optimally position available flexibility in the market. Of course, flexibility is often capitalized on in the intraday market, but smaller companies with flexibility may not have a power trading department to handle this form of trading. This is where the day ahead market with its auction format offers an opportunity to offer existing flexibility in a win-win situation: Companies want to buy power at the lowest possible prices and avoid hours with high electricity prices. In addition, low price hours are often those when there is a high supply of electricity - for example, at midday in summer due to high solar input - and high price hours are those when high demand coincides with a not-yet-large supply of renewable energy. As a result, both companies and renewable energy producers benefit from such optimization. In order to contribute to this market, we are working on optimizing the flexibility of power marketing in the form of a battery with a forecasting horizon of one week.

Section 4.1 discusses the modelling of the day-ahead market, presenting typical features and checking whether they are present in the data set we use. We then introduce a stochastic factor model based on this analysis, which includes these features, and discuss the calibration of the model.

In Section 4.2, we present methods that reduce possible evolution paths of a stochastic model to a discrete set of paths that represent optimal proxies for this evolution. To define "optimal" more precisely, different distance measures are introduced to determine the quality of an approximations. The distance measure we choose to proceed with is the nested distance, which is given by the solution to the optimization problem

$$\begin{aligned} \min_{\pi} \quad & \Big(\int \mathrm{d}(\omega, \tilde{\omega})^r \, \pi(d\omega, d\tilde{\omega}) \Big)^{1/r} \\ \text{s.t.} \quad & \pi(A \times \tilde{\Omega} \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t) = P(A \mid \mathcal{F}_t) \ \text{ for } A \in \mathcal{F}_t, \\ & \pi(\Omega \times B \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t) = \tilde{P}(B \mid \tilde{\mathcal{F}}_t) \ \text{ for } B \in \tilde{\mathcal{F}}_t. \end{aligned}$$

Furthermore, this section presents the theory behind scenario trees, which are our tool of choice to obtain discrete approximations. We show how such trees can be optimally constructed with respect to the nested distance and introduce a new method to speed up the generation of a tree.

Thus, in Section 4.3, a discretized scenario tree based on the factor model is available for which three stochastic optimization methods are described: dynamic programming, stochastic programming, and reinforcement learning.

Finally, Section 4.4 applies the theory from the previous sections to the German market. First, we describe the setting we assume for the optimization. Then, we go into

more detail about the scenario trees that are generated for the market setting with the different methods we presented. We discuss the applicability of dynamic optimization in our setting in more detail and present theoretical conditions under which it can be applied without restriction. We close with an application of the presented stochastic optimization methods to the generated scenario trees and an analysis of the obtained results.

All results from this chapter are gathered and discussed in Section 4.5.

Unless otherwise noted, all figures and tables presented in this thesis were prepared by us. They as well as the analyses behind them were made either with Python [61] and the package seaborn [64], or with R [55] and the package ggplot2 [66].

2 German electricity markets

The German electricity market consists of four main parts: the long-term forward market and the short-term day ahead, intraday and balancing energy markets, with their order being determined by how far away the delivery start of a product is during trading. On the futures market, electricity derivatives are traded that allow electricity transactions for up to six years. Futures, swaps, and options are traded in various forms. They allow electricity suppliers or buyers to manage their future price risks and reduce the volatility of their sales. Short-term markets are used to offset these positions, as more information may be available on actual expected generation or consumption closer to the delivery date. The short-term market furthest away from the delivery time is the day ahead market. It takes place as an auction on the day before the start of delivery and offers the possibility to buy or sell electricity for hourly products of the next day. There are also separate auctions for half and quarter hours, which take place a little later in the day and belong to the intraday market. Unlike the day ahead market though, the intraday market also offers the possibility of continuous trading. During the hours when the limit order books are open, bids are collected and matched against each other. This results in many prices per product rather than one. In the intraday market, products can be traded up to 5 minutes before the start of delivery. Finally, the balancing energy market serves to compensate for the imbalances that occur despite all the efforts made. For this purpose, the balancing energy market is divided into two sub-markets, the capacity reserve market and the operating reserve market. The former serves to build up an energy capacity reserve for the event that balancing energy is actually needed, and also remunerates this reserve accordingly. The latter, on the other hand, only subsidizes and remunerates bids that actually provide balancing energy.

In the following we will look at the exact rules and regulations of the two markets, day ahead and intraday.

2.1 Day ahead market design

The German day ahead market at the European Power Exchange (EPEX SPOT SE) offers producers and consumers the option to trade electricity products one day before the day of their delivery. The day ahead market is the most important electricity market for close-to-date products, as can be seen from volumes traded on this market that are given in Figure 2.2. Nonetheless, it is also visible that the importance of the day ahead market is stagnating or even shrinking, while the intraday market is on the rise. The current importance of the day ahead market stems from the fact that it was the first market to

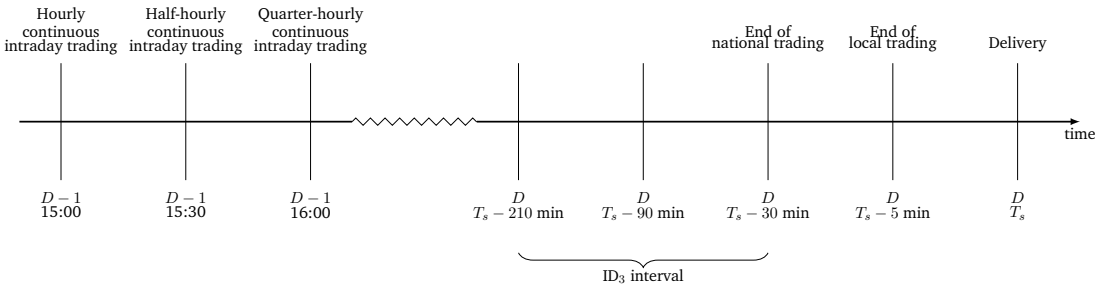

Figure 2.1: A timeline presenting the trading beginnings and ends of the EPEX SPOT SE for the continuous intraday trading.

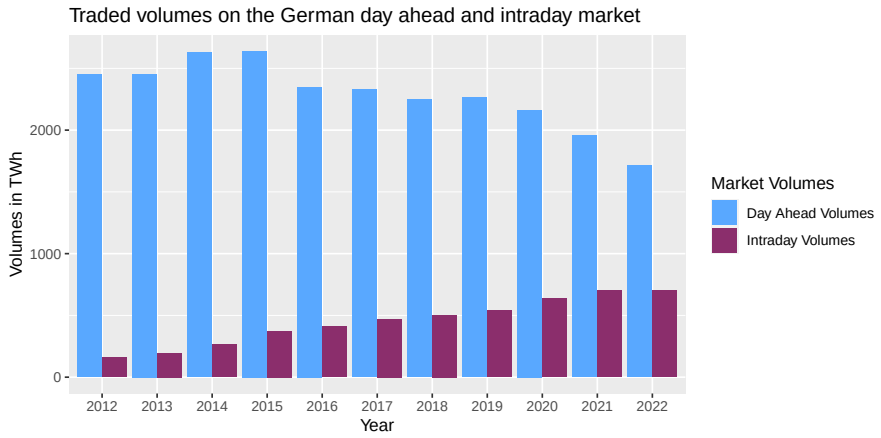

Figure 2.2: Yearly volumina traded on the German electricity day ahead and intraday markets, starting from the year 2012 up to 2022. It is important to notice that in September 2018, the bidding zone Germany / Austria / Luxembourg was split into Germany / Luxembourg and Austria, see [12]. Consequently, the bidding zone at the basis of this data has shrunk in that year.

be introduced for close-to-date trading. Even today, it is the main market for selling or buying produced or needed electricity. One reason for that is its market design, as it yields transparency and gathers liquidity for all market participants [17]. The following paragraphs are based on [16] and [17].

The day ahead market is conducted through a daily blind auction, where each hour of the following day is traded as a single hourly product. The auction is called blind because market participants do not get to know who else participates on the market or which other offers are placed. All orders of the participants are gathered anonymously every day until 12am, when the books finally close. At 12:57 pm every day, the results of the day ahead auction are published. Each order that is placed must have a volume of at least 0.1 MWh, and prices are allowed in the interval $[-500€/\text{MWh}, 4.000€/\text{MWh}]$ with a stepsize of $0.1€$, see [16].[1] The products that are available are hour 1 to 24, where hour 1 starts at 00:00 o'clock and ends at 1:00 o'clock, and hour 24 begins at 23:00 o'clock and finishes at 00:00 o'clock. The auction design of the market allows for complex bids connecting different volumes with different price levels for a product: A single hour order can contain up to 256 combinations of prices and quantities for that one hour. This ensures that a trader can buy or sell more of the respective hourly product depending on the price that is reached in the market. Block orders, as the name suggests, are orders that affect several hourly products simultaneously. The classic block order has a maximum volume of 600 MW in Germany, and can either be entirely executed or entirely rejected, or it is executed when a pre-specified acceptance ratio is surpassed. Other block order types are also offered, such as linked blocks, exclusive blocks, big blocks or loop blocks. For their specifications and the detailed rule set in general, we refer to [16] or [14].

Collections of orders gathered in their respective order book can be used to form demand and supply curves. Supply curves on the selling side strongly depend on the merit order curve that is depicted in Figure 2.3. It represents the marginal electricity generation costs of electricity suppliers. Renewable electricity producers have marginal costs near zero, closely followed by nuclear power plant. Then, most brown or hard coal power plants appear, with their order usually depending on the age of the power plant - the newer it is, the smaller its marginal production costs (usually) are. Some oil based power plants then enter the game and are finally followed by gas powered plants. These tended to be much lower in comparison to oil and were only slightly higher than coal based plants in 2018, see [22]. This changed due to steeply risen gas prices in Germany: In 2018, marginal costs of gas plants were around 40€/MWh to 80€/MWh in 2018, and around 200€/MWh to 450€/MWh in 2022 [22].
The intersections of supply and demand curves determine the market clearing prices, one for each product. When the auction results are made public, allocated volumes are announced and all participants - buyers as well as sellers - pay the corresponding market

[1] Until May 2022, the maximum price allowed was $3000€$ [19].

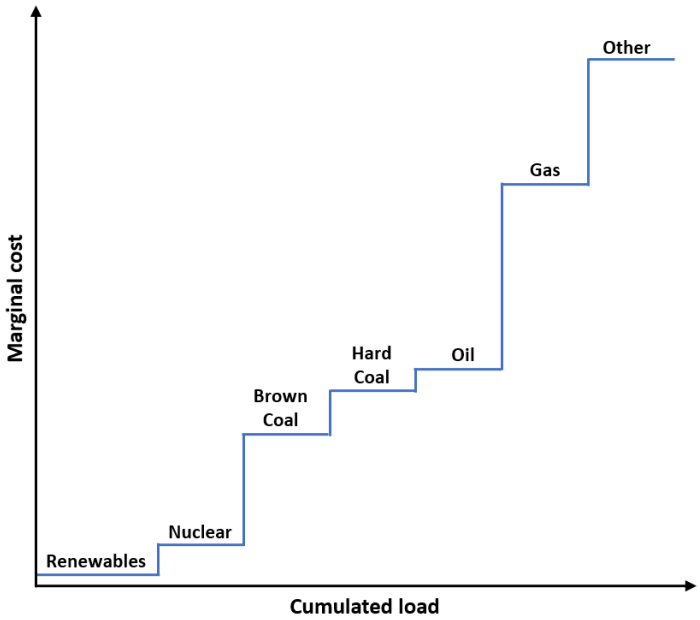

Figure 2.3: Stylized merit order curve in Germany 2022 based on current merit order information in [22].

clearing price for their allocated volume and products. This market clearing price is also called spot price, and we will use this term from now on.

2.2 Intraday market design

The German intraday market that is operated by EPEX SPOT SE is mainly used to adjust existing positions to the latest generation or consumption forecasts [16]. It compensates for forecast errors or unplanned changes, for example due to a power plant outage. In addition, this market allows owners of flexibility in their electricity purchases to monetize this flexibility. As explained in the Introduction 1, the expansion of renewable energy producers in Germany means that this flexibility is in greater demand than ever. It is needed to compensate for weather-related forecast errors by wind and solar power producers. The growing importance of the intraday market is reflected in the volumes traded there: While 59 TWh were traded across Europe in 2015, this volume has already more than doubled to 134.6 TWh by 2022, see again Figure 2.2.

The intraday market on EPEX SPOT SE consists of two components: An auction part, which is designed in the same way as the day ahead auction, and a continuous trading part. Figure 2.1 shows the timeline of the continuous intraday market. In the intraday market, there is an auction for both the half-hourly and the quarter-hourly products.

However, as these play only a minor role in comparison to the continuous market, we will concentrate on the latter from now on.[2] Continuous trading runs seven days a week, 24 hours a day, every day of the year, see [16], and involves products with hourly, half-hourly and quarter-hourly maturities. The so-called limit order books (LOBs) of the next day's hourly products open at 3:00 pm, soon followed by the start of trading in half-hourly (3:30 pm) and quarter-hourly (4:00 pm) products. Bids for all products have a volume tick size of 0.1 MW. Prices must be in the interval $[-9.999€, 9.999€]$, and must have a step size of minimally 0.01 €/MWh [16]. Cross-border trading via Single Intraday Coupling (SIDC), which allows trading with neighbouring countries in Europe, starts at 18:00. It ends one hour before delivery. Each product can then be traded nationally up to 30 minutes before delivery, and within the same Transmission System Operator (TSO) zone up to five minutes before delivery. This period of time during which the market is closed for trading is referred to in the literature as 'lead time'.

Unlike the day ahead market, the intraday market is based on a pay-as-bid system. The orders for the individual products are collected in limit order books, the structure of which will be explained in more detail in a later section. A trade occurs when an incoming order matches an order in the limit order book, i.e. when a buy order meets a sell order or vice versa, and their prices match or exceed the price of the other order. This is known as a match. Because the price paid depends only on the two orders that match, there may be many different prices for the same product during the opening period of the limit order book. As a result, the price paid for a particular product is highly dependent on the current supply and demand for that product, as well as market liquidity. Figure 2.4 presents an exemplary development of the transaction prices of one product over the opening time of the LOB on the left, namely the prices of the hourly product from 12pm to 1am on the 1st of February. The reason that the time axis begins only after 10pm on the 31st of January is that before this time, no transaction for that product is conducted. It is well visible that the liquidity on the market is very low until around six hours before delivery, and then starts to rise. The histogram on the right of Figure 2.4 illustrates that the prices to be paid are notably different over the whole time interval.

In order to give traders signals about the actual price of the various traded products during their trading time interval, several volume-weighted transaction price indices exist and can be bought or sold on the exchange. They smooth the volatile prices of their respective products and guard against price deviations that were not factored in. Each index is limited to a different time interval. The most popular one is called ID_3,and it is used as the underlying for the German intraday cap and floor futures, see e.g. [28]. It is defined as the volume-weighted average transaction price of all transactions during

[2] In 2022, all intraday auctions from EPEX SPOT SE over all trading areas combined yielded a volume of 13.5 TWh. In comparison to that, the continuous intraday trading yielded a total trading volume of 121 TWh in the same year [18].

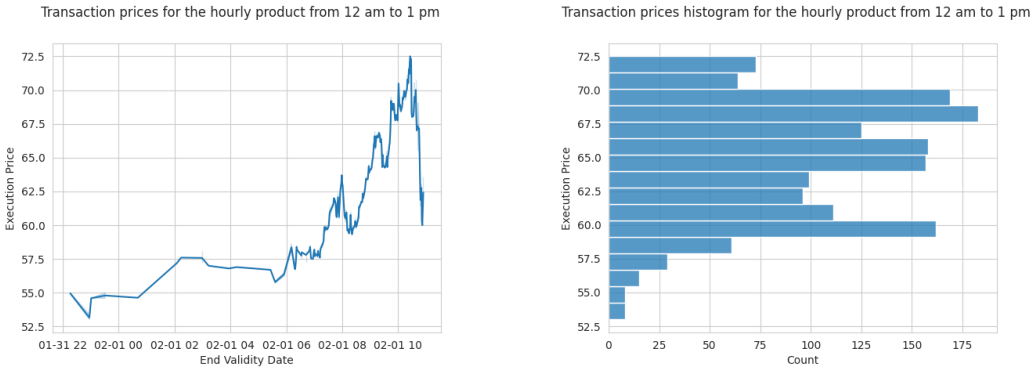

Figure 2.4: Exemplary transaction price development over the LOB opening time of one day for the hourly product from 12 am to 1 pm.

the last three hours of national trading, i.e.

$$\mathrm{ID}_3^{d,i} = \sum_{t \in T^{d,i}} \frac{p_t v_t}{\sum_{t \in T^{d,i}} v_t}. \tag{2.1}$$

Here, d specifies the timestamp of the delivery start, and $i \in \{\mathrm{h}, \mathrm{qh}\}$ denotes the type of the product, i.e. whether it is an hourly or quarter hourly product. The ID_3 also exists for half hourly products, but because the traded volume for half hourly products is much lower than for quarter hourly or hourly products, we will not consider them in this thesis.[3] Furthermore, $T^{d,i}$ contains all transactions of the product with delivery start at d and type i that appeared during the ID_3 interval, which is also visible in Figure 2.1. As this interval depends on the national and SIDC trading only, it is e.g. for the ID_3 of the hourly product delivering from 8 am to 9 am given by the time interval 4:30 am - 7:30 am. Finally, p_t is the price of transaction t and v_t is the corresponding traded volume. As the ID_3 depends on the prices and volumes that were matched on the market in the most active trading period of each product, it can be considered as a volume-weighted mean of prices that buyers are willing to pay and sellers are willing to take close to delivery.

To gain a feeling for the ID_3 prices, Figure 2.5 presents the ID_3 prices for the hourly product from 12 am to 1 pm over 2019. It is visible that no obvious yearly or weekly seasonal component is contained in the data, whereas spikes do indeed occur. This is a result of the fact that most market participants use the day ahead market to clear their position and use the intraday market for short-term adjustments - the day ahead market is the one that captures most of the seasonality, whereas the intraday market captures short-term changes in e.g. weather developments.

An overview over the price distribution of the ID_3 values for hourly and quarter hourly products over the year 2019 is given in Figures 2.6 and 2.7, respectively. We see here that a daily seasonal component is contained that resembles the one found on the day

[3] In 2021, the EPEX SPOT SE data present a traded volume of around 28.9 TWh in Germany for the hourly ID_3, the quarter hourly still has a volume of 6.6 TWh and the half hourly ID_3 a volume of 0.1 TWh.

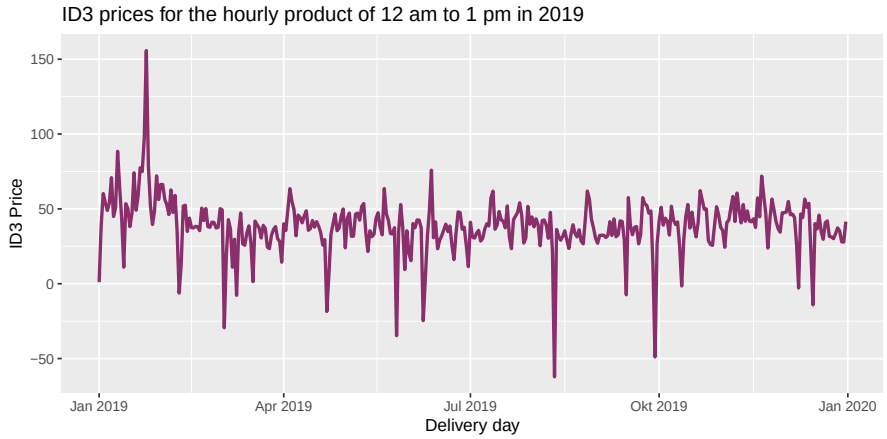

Figure 2.5: Exemplary ID_3 price development over 2019 for the hourly product from 12 am to 1 pm.

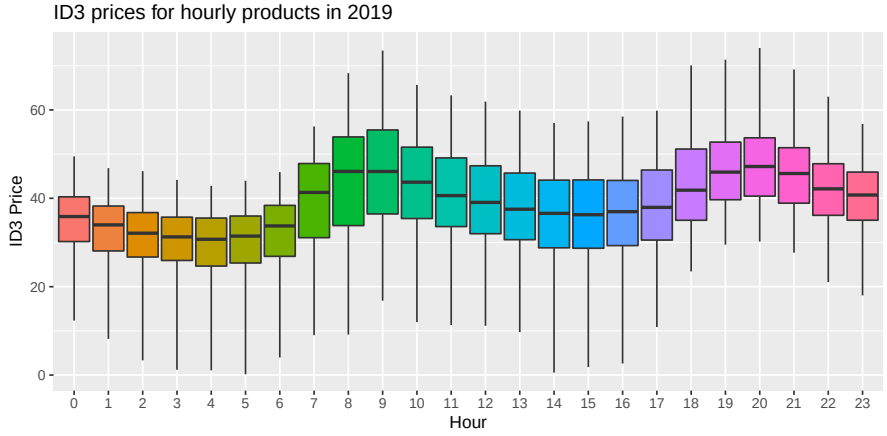

Figure 2.6: The distribution of hourly ID_3 prices over 2019.

ahead market. Furthermore, the typical sawtooth pattern for intraday prices in one hour is visible, see also [40].

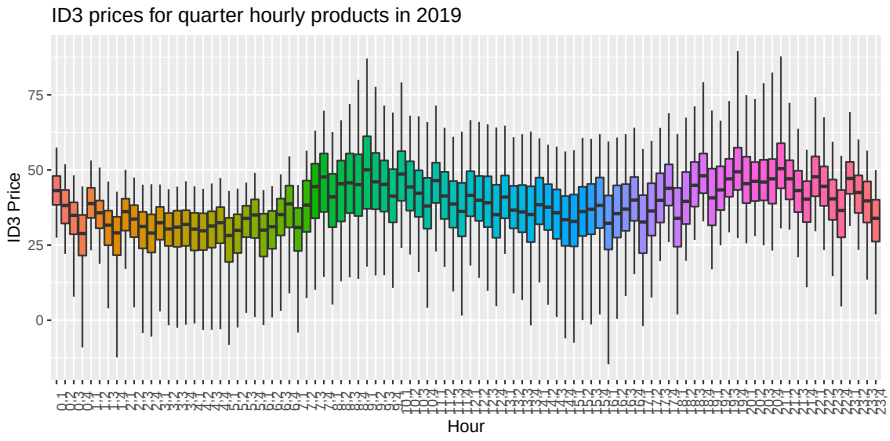

Figure 2.7: The distribution of quarter hourly ID_3 prices over 2019.

3 Modelling and prediction on the intraday market

In 2010, the German government issued plans regarding the electricity supply in Germany for the coming decades [5]. Its goals were to decrease greenhouse gas emissions as well as dependencies from electricity imports while keeping electricity supply stable, sustainable and affordable. These goals all have one solution component in common: An increase of renewable electricity production locally in Germany. The governmental plan thus included to enlarge the share of renewable electricity in respect of gross electricity consumption to 80% in 2050. The release of this goal led to a surge in the expansion of renewable electricity generation capacity in Germany over the last decade. As a consequence, the share of renewable electricity in respect of gross electricity consumption increased from 17% in 2010 to a remarkable 45.4% in 2020 already. This increase has indeed been stronger than before, as in the decade from 2000 to 2010, the share only rose from 6.7% to 17% [59].

The main sources for the rise in the share of renewable electricity in Germany are wind and solar energy, see [59]; and at the same time, from all possible sources of renewable electricity, these two are the ones with the most volatile infeed pattern. Solar electricity heavily depends on whether clouds or other shadows reduce the amount of light that reaches the panels and with that, the amount of electricity produced. Wind on the other hand is hard to predict, as every wind farm location and every wind turbine height has its own physical circumstances. The fact that these two sources are the main drivers for renewable electricity production in Germany thus leads to new challenges for the coupling of electricity production and demand. Demand is usually quite stable in the sense that it can be predicted in a reliable fashion, and does not immediately react to price signals on the market from e.g. a forecast change for renewable infeed. The day ahead market provided by EPEX SPOT SE, an auction where traders can buy or sell electricity for the next day, is not close enough to the start of delivery to react to these changes. This is where the electricity intraday market comes into play: It can be used to run and plan power generation close to delivery, and therefore close to a better forecast of consumption, as well as to adjust for forecast errors or unforeseen events [16]. These balancing features in combination with the increase in renewable production led to the electricity intraday market mirroring the development of renewable electricity production and continuously growing over the last years. In the year 2020 alone, the continuous intraday market operated by EPEX SPOT SE for the market area Germany/Luxembourg grew by 19.6% [15]. This indicates that a functioning and flexible electricity intraday market is

a necessity for a functioning growth in volatile renewable electricity production.

In order to keep up with the increasing need to trade shortly before delivery, the EPEX SPOT SE electricity intraday market has continued to develop over the last years and EPEX SPOT SE expanded its service portfolio [13]. Traders can nowadays buy or sell hourly, half hourly or quarter hourly products on the market. The market principle is pay-as-bid, i.e. orders are collected and matched when possible during trading hours via *limit order books* (LOBs). This market structure leads to many different occurring prices for the same product, which complicates the assessment of the product's "true" price. In order to facilitate an evaluation of this actual price, several price signals in form of indices have been introduced for the traders on the intraday market [11]. All these indices focus on materialized trades for the different products on the market. Two of these indices take into account price evolutions shortly before delivery e.g. due to forecast corrections: the ID_3 and the ID_1. They are limited to the last three hours and last hour of national trading, respectively. In this thesis, we focus on the ID_3, as it is the index that is used as underlying for German Intraday Cap and Floor Futures, and much of the existing literature concentrates on it.

Prediction of electricity intraday and especially index prices in Germany is a relatively new research topic. Nonetheless, a few recent publications address forecasting of intraday prices. [43] focus on a small renewable energy producer that wants to place an optimal amount of electricity on the day ahead and the rest on the intraday market. Thus, they forecast the spread between day ahead prices and the weighted average of all intraday trades for each hourly contract. [60] present a thorough study of intraday electricity forecasting with a focus on ID_3-prices. They apply the least absolute shrinkage and selection operator (LASSO) to an abundance of variables and compare ten of the emerging models with two benchmarks. A major takeaway is that the most recent intraday price as well as the day ahead price for the specific hour seem to have the highest prediction power, with many other variables by far not as relevant. [47] follow a similar path and compare different full-information econometric time series models for all ID_3-prices, that are also reduced through the usage of LASSO and elastic nets. All resulting models are used for forecasting and compared to each other and to benchmark models in terms of their forecast performance. They find that for hourly products, surprisingly none of the full-information models outperforms one of their simple naive benchmarks. [36] concentrate on forecasting the whole price distribution for the ID_3 instead of only the expected value. Their models manage to outperform the naive benchmark from [47] in the tails of the distribution, but also show no significantly better results for the expected value. [44] use principal component analysis to average over a pool of forecasts for ID_3-prices to enhance the overall forecasting skill. Their model of the ID_3 relies on lagged ID_3-prices as well as day ahead prices, if no lagged prices are available. Their findings suggest that no forecast averaging outperforms the forecast based on the longest calibration window. Finally, [45] manage to outperform the naive benchmark from [47] for hourly products by combining it with a least absolute shrinkage and selection model

and thus show that fundamental variables still play a role in forecasting electricity prices.

Our approach to electricity index price forecasting relies on the fact that the transaction data tracked by the indices are always a result of the underlying limit order book movements. We examine how to derive data from the limit order book and then how to use them to enhance these forecasts. Our findings indicate that lagged time-weighted mid prices, i.e. time-weighted mean prices between best ask and best bid prices from a time interval before the index interval, are a valuable source of information for forecasting ID_3 hourly and quarter hourly prices. Regarding quarter hourly ID_3 prices, they actually outperform the naive benchmark significantly, partly because of occasionally arising illiquidity for the corresponding products. Half hourly prices have not been taken into consideration. As forecasting requires usage of information that is adapted to the time of the forecast, we will omit the affix "lagged" from here on.

We continue this chapter with Section 3.1, which gives insight into the structure of limit order book data and discusses available data sets. This is followed by Section 3.2, where we present an algorithm that is used to model the limit order book from order book data, and introduces first statistics that can be elicited from it. Then, Section 3.3 presents a collection of order book statistics as well as forecasting models based on them for the ID_3, that are put into practice for the German market in Section 3.4. Finally, Section 3.5 concludes the topic by summing up the results and presenting further ideas. This chapter is based on our preprint [24][1].

3.1 Data

This section introduces the structure of an order a trader places on the intraday market. Furthermore, we present the structure of order book data from EPEX SPOT SE, and it is explained in detail which information is available during its usage. Finally, the subtleties and pitfalls of the data set are discussed. As limit orders are mostly used on the market, we concentrate on them in the following section and only explain basics regarding market sweep orders in Section 3.2. The section is based on information from [14] and [16].

3.1.1 Order structure

The intraday market operated by EPEX SPOT SE uses a trading system called M7, which, aside from allowing to enter the product of interest, a side (buy or sell) as well as a price and quantity combination, enables the traders to make several specifications for limit orders. Orders can be specified regarding type, execution restriction and validity restriction. In the following, we present the offered alternatives for these three specification

[1] I want to thank Nikolaus Luckner for the provision of his code which was used as first draft for the building of the limit order book code, and for discussions regarding the topic.

types.

The exchange provides different types of orders:

(i) **Regular Order** (REG): Regular orders are limit orders that are placed with a certain limit price and quantity. They have to be executed at their specified limit price or better.

(ii) **Iceberg Order** (ICB): An iceberg order is a limit order which is used to conceal a bigger trade in order to limit the market movement against the trading party. Only a part of the total wanted quantity is visible to the market participants, even though the full quantity is available to them. An order with beforehand-specified quantity is placed at the market, and after matching an existing or incoming order, a next order with the same quantity is placed. This goes on until deletion of the iceberg order or complete fulfillment.

(iii) **Indicative Order** (IND): This order provides an indication that the trader is willing to negotiate around the specified price for the chosen delivery instrument.

(iv) **Over-the-counter Order** (OTC): This is an order with a predesignated counterpart, i.e. a trade which is not available to all market participants.

Furthermore, an execution restriction can be added to the order:

(i) **All or None** (AON): The order is either matched in its entirety or not matched at all. There is no partial execution. This order type is not found in the regular order book, but solely in the order book for user-defined block orders.

(ii) **Fill or Kill** (FOK): The order is either immediately matched in its entirety or immediately deleted after submission.

(iii) **Immediate or Cancel** (IOC): The order must be immediately executed or is deleted. In comparison to ”fill or kill”, it is possible to only partially match other orders.

Finally, the trader has the possibility to choose a validity restriction for the order:

(i) **Good for Session** (GFS): The order is deleted when the trading end time of the contract is reached.

(ii) **Good till Date** (GTD): The order comes with an additional information containing the time and date at which to delete the order.

Another very important choice for traders is in which book they want to place their order: In the book for local orders or in the book for remote orders [14]. Local orders are matched locally in M7, whereas remote orders are transferred to the XBID System which has exclusive access to cross-border capacities. In the latter case, orders can be matched with orders in another country if cross-border capacity allows it. As the working hours for the XBID market are a subset of the M7 working hours, traders can first enter their orders locally, then remove them and insert them remotely, and finally remove them again and insert them locally again.

3.1.2 LOB structure

The listed specifications lead to a basket of information that is attached to every order. Some of this information is contained in the data sets for order book data available from EPEX SPOT SE. Of course though, their structure has seen changes over the last years, as the intraday market itself has. Therefore, we explain in detail what information the latest data format D2 (applied from November 2019 onward) as well as its predecessor D1 (encompassing the period from 2017 to November 2019) contain.

Columns D2	Columns D1
Order ID	Order ID
Initial ID	Initial ID
Parent ID	Parent ID
Entry Time	Start Validity Date
Transaction Time	End Validity Date
Validity Time	Cancelling Date
Action Code	
Delivery Start	
Delivery End	
	Delivery Date
	Delivery Instrument
Product	Instrument Type
Delivery Area	Area
Market Area	Area
Side	Side
Price	Price
Currency	
Quantity	Volume
isOTC	
RevisionNo	
Is User Defined Block	Is Block
Execution Restriction	
	Is Executed
	Execution Price
	Execution Volume

Table 3.1: Columns contained in the data formats D2 (valid from November 2019 onward) and D1 (valid from 2017 to November 2019). Columns found in both data formats are contained in the same row, those without an equivalent stand alone.

The elements contained for both types are listed in Table 3.1. Columns corresponding to each other are placed in the same row. Some columns don't have an exact equivalent, but their information is nonetheless found in the other data type as well, e.g. Delivery

Start and Delivery End for D2 can be translated to Delivery Date and Delivery Instrument and Instrument Type of D1. In comparison, some columns of D2 contain information that is not found in D1, namely Action Code, Currency, isOTC, RevisionNo and Execution Restriction.

An innovation presented in D2 is the introduction of action codes: In the old data format, one data row contained always at least two different events for one order ID, i.e. activation and deactivation of the order, and possibly also a timestamp for a modifying event. The new format has a different approach: Every row accounts for one event only, and the type of event is denoted by an action code. The action code specifies which action is applied to the order. The different possible actions and their typical appearances are listed in the following:

- **A for Activation**: Following an order through its lifetime, its first appearance always starts with its activation. An order with action code A also appears after deletion of the previous order and creation of a (slightly) changed one.

- **X for Automatic Deletion**: This action determines the end of the lifetime of an order and is performed by the system itself. Nonetheless, the order could still be parent order of a new order appearing afterwards.

- **D for Manual Deletion**: This action also determines the end of the lifetime of an order, with the small difference that now the order was manually deleted. Again, the order could still be parent order of a new order appearing afterwards.

- **M for Matched**: If a transaction takes place and an order is completely matched, its quantity is denoted by 0.0 and its action code is M. An M furthermore determines the end of the lifetime of an order.

- **P for Partially Matched**: In the case of a partial matching of the order, the corresponding action code is P and the quantity associated with it is the remaining quantity after the partial match. If the corresponding order is an Iceberg Order, even a complete fulfillment of the stated quantity is commented with action code P. This goes on until complete fulfillment (M) of the Iceberg Order or until its deletion (X or D).

- **I for Insert**: If one shown share of an Iceberg Order is matched (action P), the next equally sized limit order is placed at the market through an insertion I. In this case, the order ID stays the same. This procedure goes on until complete fulfillment (M) of the Iceberg Order or until its deletion (X or D).

- **C for Changed**: This represents technical changes in the M7 trading system and does not account for any real modification. This leads to the counter-intuitive behavior that a modification of the order limit is not denoted with C, but instead the order is deleted and a new order gets activated.

- **H for Hibernated**: Hibernation temporarily deactivates an order until further notice.

The life story of one order is thus presented by filtering for the order ID of interest and then sorting after revision number.

3.1.3 Discussion about data sets

Listing the advantages of the new data format, one might assume it to be the better fit for building LOBs or similar tasks. However, the fragmentation of an order into several timestamps, namely an entry time, a transaction time and sometimes a validity time, complicates this. The entry time is the time used in the price/time ranking in M7, and should therefore also be used for building LOBs. Unfortunately, the data for this timestamp suffers from inconsistencies due to the export from M7 to the database, especially for historical files[2]. This behavior is supposed to have changed in a further M7 release, but prevented us from using the new data format to reliably build LOBs. As a consequence, we suggest to use the new data format for scientific purposes only after checking whether this problem has been solved.

The orders in the old format can be subject to the different order type choices as well as execution and validity restrictions as described above. [34] describe the existence of an execution restriction as a cause of trouble while building LOBs with the old data format from the EPEX SPOT SE, as the chosen restrictions are not communicated and can therefore not be recreated from the data. Nonetheless, we believe this to be easily handled: As AON orders are part of a separate order book, they are not part of the data. FOC and IOC orders, if not filled or partly executed, respectively, have a validity duration of zero and therefore do not enter the LOB. Furthermore, the problem of erroneous order information in columns ”execution price” and ”execution volume” mentioned in [34] was removed in 2017.

3.2 Modelling of the intraday market

Based on order data from the exchange, it is possible to model and reconstruct the true underlying limit order book. The idea for this procedure is found in [34], and is detailed in the following.

During the trading window of every product, orders for it can be placed on the market. The trader chooses between placing a *limit order* or a *market sweep order*, in the literature often called market order, see [14].

[2]This was stated by EPEX SPOT SE in a mail correspondence from November 2021.

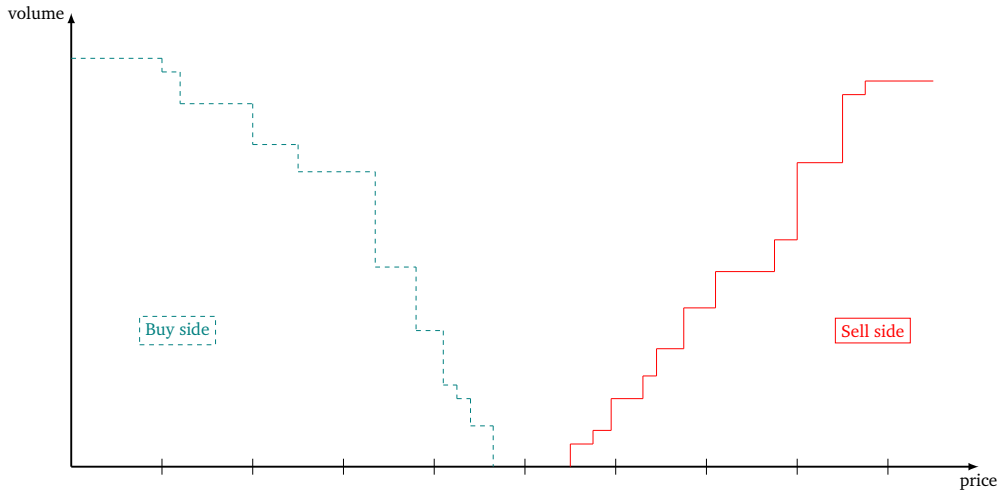

Figure 3.2: This figure represents the state of a limit order book at a fixed moment in time. In dashed turqoise, the accumulated volume of the buy side is presented for every price, and in solid red, the accumulated volume of the sell side is presented for every price. When the highest buy price and the lowest sell price intersect, a match occurs.

A limit order contains a volume and a limit price at which the trader would buy or sell the product, with the option to get a better price. If the incoming limit order does not match an already existing order on the other side, it is placed into the limit order book (LOB). The LOB contains all orders that are neither immediately (and fully) matched nor canceled, see Figure 3.2. When a new limit order matches one (or several) orders inside the book, the latter is (are) removed from the LOB and the initiating one does not enter the LOB at all. It is also possible that an order is only partly matched. Then, the remaining volume is again entered into the LOB.

In contrast to a limit order, a market sweep order contains only a volume and will always be filled completely except for the case that not enough volume is contained in the whole LOB. The price of the market order is composed by the volume-weighted prices from the limit orders that were matched. As a market sweep order is always filled as much as possible and unexecuted quantity is canceled, it never enters the LOB [14]. In everyday business, market orders tend to be replaced by limit orders with their price equaling the best price of the opposing market side.

In conclusion, only those orders enter the LOB that actually provide liquidity to the market; those that instead use the provided liquidity do not enter the book.

Each product has its own LOB where orders for it are collected, and ordering as well as matching in the LOB is based on a price-time ranking: a limit order is sorted into the LOB first by checking its price, and, if another order with the same price is already part of the LOB, its execution priority is ranked lower than that of the existing order. Orders with the same price form a *price level*, and the *LOB depth* is defined as the number of

different price levels. A match of the orders contained in the book with a new incoming order will always happen first for those orders with the best prices. Consequently, the buy order with the highest buy price is called *best bid* and defined as bid_t^* for time t, and the sell order with the lowest sell price is called *best ask* and respectively defined as ask_t^* for time t. These two orders determine the prices a new bidder has to outbid in order to be placed first in the book, and also determine the price range that needs to be crossed by a new order in order to match. The arithmetic mean m_t between best bid and best ask is called *mid price* m_t, and the difference between them is called *bid-ask spread* s_t, see e.g. [41]. Therefore, we have

$$m_t := \frac{\text{ask}_t^* + \text{bid}_t^*}{2}, \tag{3.1}$$

$$s_t := \text{ask}_t^* - \text{bid}_t^*. \tag{3.2}$$

The concept of looking at mid prices and bid-ask spreads is known from stock price analysis: The mid price reflects the current value of the product, while the bid-ask spread yields an indicator for how much liquidity is on the market as well as how high the incurred transaction costs are. Transaction costs in this sense arise from the fact that an order in the book provides liquidity, and an order matching it demands, and consequently removes, liquidity from the market, which is paid via the bid-ask spread as well as through the difference between the best and the realized price.

All order data is commercially available from EPEX SPOT SE, and as described in Section 3.1, contains among others identification numbers, timestamps, prices and volumes. These can be used to recreate the different states that the LOB attains throughout its opening time. Consequently, we present in Algorithm 1 how to recreate the LOB for a certain product.

Moving step by step through Algorithm 1, first, all orders that are entered into the trading system on the exchange are collected in the variable ob_data. Before processing these orders, they are first filtered (line 4) such that the result only contains orders regarding the prespecified product. All orders that are specified as block orders for that product are removed from the data, as they are traded and matched in a different book. Then, all orders with a live span of zero milliseconds that are not matched are removed (line 5), as they do not enter the LOB. This concludes the preselection of the data to be processed. For every remaining order, two events are constructed: its activation event and its deactivation event, both with the corresponding timestamp of beginning and end of the order's lifetime, respectively (line 6). All these events are collected in the variable *all_events*. After sorting *all_events* by start validity time, order ID, and activation to let them appear in correct order of appearance, the events are processed one by one. It is checked whether they enter the LOB (which is the case if they do not match with an existing order in the book) or match and therefore remove an order from the order book. The detailed procedure is illustrated from the perspective of a buy order: The first check verifies if the corresponding event is an activation (line 10). If the answer is affirmative, the second check verifies whether the offered price is equal to or higher

Result: The output is a collection of all order book states for a given delivery product over its lifetime.

begin
- Initialize the lists buy_OB, $sell\_OB$ and the numerics $best\_ask$, $best\_bid$.
- Read order book data as ob_data.
- Filter ob_data such that only rows with fitting delivery product remain.
- Rows with $ob\_data["validity\_duration"] = 0$ & $ob\_data["is\_executed"] = 0$ are removed.
- Split each row in ob_data in two events (activation of the order and deactivation of the order). Define all_events as the data frame with all those events.
- Sort all_events by $start\_validity\_time, order\_id, activation$.
- **for** row *in* all_events **do**
  - **if** $row["side"] = "buy"$ **then**
    - **if** $row["activation"] = 1$*, i.e. is an activation event* **then**
      - **if** $row["price"] \geqslant best\_ask$ & $row["is\_executed"] = 1$ **then**
        - row contains a market order and is not entered in the order book.
        - The corresponding deactivation event with the same $order\_id$ is marked s.th. it is omitted later on.
      - **else**
        - buy_OB gains a new list entry with the name being the timestamp $row["start\_validity\_time"]$ and the content being the list entry before this one plus row.
    - **else**
      - buy_OB gains a new list entry with the name being the timestamp $row["start\_validity\_time"]$ and the content being the list entry before this one minus the entry belonging to row.
    - Update $best\_buy$ if necessary.
  - **else**
    - **if** $row["activation"] = 1$*, i.e. is an activation event* **then**
      - **if** $row["price"] \leqslant best\_bid$ & $row["is\_executed"] = 1$ **then**
        - row contains a market order and is not entered in the order book.
        - The corresponding deactivation event with the same $order\_id$ is marked s.th. it is omitted later on.
      - **else**
        - $sell\_OB$ gains a new list entry with the name being the timestamp $row["start\_validity\_time"]$ and the content being the list entry before this one plus row.
    - **else**
      - $sell\_OB$ gains a new list entry with the name being the timestamp $row["start\_validity\_time"]$ and the content being the list entry before this one minus the entry belonging to row.
    - Update $best\_ask$ if necessary.
- **end**
- **return** buy_ob, $sell\_ob$.

end

Algorithm 1: Order book building for one delivery product

than the momentary best ask, in combination with a check whether the order is marked as executed (line 11). This indicates that the order is a market order, and these do not enter the LOB. Consequently, the corresponding deactivation event is marked such that it is omitted later on (line 13). If one of the conditions in the second check were not fulfilled, the order would be entered into the LOB by following the price-time ranking (line 17). Finally, in the case where the first check was not affirmative, the event is a deactivation of an order, so the corresponding element in the LOB is deleted (line 19). This procedure leads to full information about all states of the LOB at every point in time during its lifetime.

3.3 Prediction of the intraday index ID_3

Whether a company wants to predict its electricity costs on the market more accurately, wants to make a profit by using its temporal flexibility in consumption, or has a structural interest in predicting a market: In any case, a model of the underlying market is required. The same is true for many of the applications in the intraday market, but in the literature, often only a single price is modelled for each product, see e.g. [71] or [39]. This method, which suits well the day ahead market with its auction structure, reaches its limits in the intraday market. The many different prices of a product over the duration of the opening of the associated LOB are an inherent feature of the market, see again exemplarily in Figure 2.4. Simplifying this by mapping it to a single price – which is often one of the intraday indices that is being modelled – can lead to misinterpretation of the price or even of the volume and illiquidity risks that arise. As it is our goal to model exactly, we follow the approach in [34] and instead model the underlying order book based on available order book data. One might now assume that the modelling of the order book on the basis of order book data would be a perfect replication of the LOB. However, this is not the case due to errors in the data and partially invisible data in the order books. Therefore, the simulation of the LOB based on order book data is only an approximation of the true LOB process. In addition, it is of course possible to model how the market would behave in this respect by adding more orders. We explain how this modelling is done below.

The best performing models use a huge load of explanatory variables, amounting from up to 222 in [45] over up to 372 in [60] to up to 26259 in [47]. Model reduction and selection is managed through the usage of the least absolute selection and shrinkage operator (LASSO) method or through elastic nets. Neither in these huge sets of explanatory variables nor in any of the other contributions though, other LOB information aside from transaction data is used. Therefore, we want to investigate whether some of the LOB statistics introduced in Section 3.1 could actually enhance ID_3 forecasting performance. In order to answer that question, we first introduce interval versions of the LOB statistics described above for further usage as well as additional statistics.

Time-weighted LOB statistics From the LOB recreation that is available after the application of Algorithm 1, several derived data like mid prices, bid-ask spreads and order book depths can be calculated for every snapshot during the LOB's life span. Usually, a collection of these values for every change in the LOB amounts to a very high number of entries, most being valid for only seconds. Therefore, we aggregate them over specified time intervals. A reasonable way to reach interval statistics is to introduce a time-weighting. The candidate statistics are aggregated over a time interval $[f(d), g(d)]$, where d is again the timestamp of the delivery start of the chosen product, and f, g are functions of time with $f(d) < g(d)\ \forall d$. Then, we define the following interval statistics for the product with delivery start d and product type i:

- Mid price: The time-weighted mid price $m^{d,i}$ in the interval $[f(d), g(d)]$ is defined as

$$m^{d,i} = \frac{1}{g(d) - f(d)} \int_{[f(d),g(d)]} m_s^{d,i} \, \mathrm{d}s = \sum_{j=0}^{J} m_{s_j}^{d,i} \frac{(s_{j+1} - s_j)}{g(d) - f(d)}, \tag{3.3}$$

  where $m_s^{d,i}$ is the mid price at time s, i.e. $m_s^{d,i} := \frac{\text{ask}_s^{*,d,i} + \text{bid}_s^{*,d,i}}{2}$. As the mid price only changes finitely often in the interval $[f(d), g(d)]$, the integral can be rewritten as a sum. Finally, the ordered set of time points with changes in that interval is denoted by $T_{\text{mid}}^{d,i}$ and contains J elements $s_1, ..., s_J$. We define $s_0 := f(d)$ and $s_{J+1} := g(d)$.

- Spread: The time-weighted bid-ask spread $\eta^{d,i}$ in the interval $[f(d), g(d)]$ is defined as

$$\eta^{d,i} = \frac{1}{g(d) - f(d)} \int_{[f(d),g(d)]} \eta_s^{d,i} \, \mathrm{d}s = \sum_{j=0}^{J} \eta_{s_j}^{d,i} \frac{(s_{j+1} - s_j)}{g(d) - f(d)}, \tag{3.4}$$

  where $\eta_s^{d,i}$ is the spread at time s, i.e. $\eta_s^{d,i} := \text{ask}_s^{*,d,i} - \text{bid}_s^{*,d,i}$. As the spread also only changes finitely often in the interval $[f(d), g(d)]$, the integral can again be rewritten as a sum. The ordered set of time points with changes in that interval is denoted by $T_{\text{spread}}^{d,i}$ and contains J elements $s_1, ..., s_J$. We define $s_0 := f(d)$ and $s_{J+1} := g(d)$.

- Skewness: In order to measure skewness in the order book, we consider volumes v with $v \in \{10, 20, 30, 40, 50, 60\}$ MW. For each v, we compare how much traders demanding v MW would pay, to how much they would get when instead offering v MW on the market. The difference between both is defined as skewness and is supposed to measure the well-balancedness of the LOB. To avoid incorporating the spread into this definition, we deduct the best prices of both sides from all respective following price levels. Consequently, we define skewness at one point in time as

$$S^{d,i}(v) = \sum_{j=1}^{N} v_j^s (p_j^s - p_1^s) \mathbf{1}_{\sum_{k=1}^{j} v_k < v} + v_{j*}^s (p_{j*}^s - p_1^s) - \sum_{j=1}^{M} v_j^b (p_j^b - p_1^b) \mathbf{1}_{\sum_{k=1}^{j} v_k < v} - v_{j*}^b (p_{j*}^b - p_1^b),$$

where N is the number of price levels on the sell side and M the number of price levels on the buy side. Consequently, v_j^s is the volume available on the sell side at price level j and price p_j^s. Similar roles on the buy side are taken by v_j^b and p_j^b. Finally, p_{j*} and v_{j*} are the price on the last reachable level and the rest volume which is still not covered from the price levels before. Defined like this, skewness incorporates the statistics price levels and LOB depth. Now, we can define

$$SK^{d,i}(v) = \frac{1}{g(d) - f(d)} \int_{[f(d),g(d)]} S_s^{d,i}(v)\,\mathrm{d}s = \sum_{j=0}^{J} S_s^{d,i}(v) \frac{(s_{j+1} - s_j)}{g(d) - f(d)}, \tag{3.5}$$

where $S_s^{d,i}(v)$ is the skewness at time s for volume v. Again, the set of time points with changes in that interval can be denoted by $T_{\text{skew}}^{d,i,v}$ and contains J elements $s_1, ..., s_J$. We define $s_0 := f(d)$ and $s_{J+1} := g(d)$.

Note that $|T_{\text{mid}}^{d,i}|$, $|T_{\text{spread}}^{d,i}|$ and $|T_{\text{skew}}^{d,i,v}|$ need not be equal. What is left to define is the subtraction of two timestamps. We follow [33] and specify that subtraction of two timestamps leads to a duration $\text{Dur}(\cdot)$ in minutes. This time unit has been chosen out of convenience and could be replaced by any other time unit. Regarding the defined statistics, the time-weighted mid price is assumed to capture the actual value of the product, whereas the bid-ask spread could be seen as an indicator for how much liquidity is available on the market. Skewness parameters try to capture well-balancedness of the market and therefore indicate whether the market is a buyer's or seller's market, i.e. a market where purchasers have an advantage over sellers due to an abundance of goods for sale or vice versa. Nonetheless, this should not be mixed up with the skewness of a distribution, although the idea is inspired from it.

In order to decide for an adequate time interval to aggregate over, we adopt the idea of [47]: Here, the forecasting results of one of the constructed naive benchmark models for the ID_3 for hourly products could not be significantly outperformed by the best suggested models, and even though [45] showed that combining it with fundamental variables improves forecasting quality, the naive benchmark in itself is already a very good estimator. This specific benchmark model only relies on volume-weighted transaction data from the quarter hour before the ID_3 interval. Based on this result, we decide to restrict ourselves to LOB statistics from that quarter hour as well. Furthermore, we also include the volume-weighted transaction price as an interval statistic for further analysis, even though the weighting method is a different one in comparison to the other

statistics. Consequently, the full model is given by

$$
\begin{aligned}
{}_{\text{full}}\hat{\text{ID}}_3^{d,i} \quad = \quad & a_1 \sum_{s_j \in T_{\text{mid}}^{d,i} \cup\, g(d)} m_{s_j} \frac{(s_{j+1} - s_j)}{g(d) - f(d)} + \\
& a_2 \sum_{s_j \in T_{\text{spread}}^{d,i} \cup\, g(d)} \eta_{s_j} \frac{(s_{j+1} - s_j)}{g(d) - f(d)} + \\
& \sum_{v=1}^{6} \left(a_{k+2} \sum_{s_j \in T_{\text{skew}}^{d,i,v} \cup\, g(d)} SK_{s_j}(v) \frac{(s_{j+1} - s_j)}{g(d) - f(d)} \right) + \\
& \sum_{t \in T_{\text{trans}}^{d,i}} \frac{p_t v_t}{\sum_{t \in T_{\text{trans}}^{d,i}} v_t}.
\end{aligned} \tag{3.6}
$$

All variables are defined as in Equations (3.3) to (3.5). Regarding the last row of Equation (3.6), we define p_t as the price of transaction t and v_t as the corresponding traded volume. $T_{\text{trans}}^{d,i}$ contains all transactions that take place in the interval $[f(d), g(d)]$ for products with type i and delivery start d. Again, d denotes the time stamp of the delivery start, and as before, i is again contained in $\{\text{h}, \text{qh}\}$, representing either hourly or quarter hourly products. In case there are no transactions in the relevant quarter hour, the forecasted value is replaced by the corresponding day-ahead price.

In order to build simpler models with less variables as well, we furthermore define the following models.

The two naive forecasting models We present two naive models, where one is based on mid prices and the other is based on transaction prices. The former is defined by

$$
{}_{\text{mid}}\hat{\text{ID}}_3^{d,i} = \sum_{s_j \in T_{\text{mid}}^{d,i} \cup\, g(d)} m_{s_j} \frac{(s_{j+1} - s_j)}{g(d) - f(d)}, \tag{3.7}
$$

where all variables are defined as in Equation (3.3). The second model that is based on transaction prices is denoted as

$$
{}_{\text{trans}}\hat{\text{ID}}_3^{d,i} = \sum_{t \in T_{\text{trans}}^{d,i}} \frac{p_t v_t}{\sum_{t \in T_{\text{trans}}^{d,i}} v_t}. \tag{3.8}
$$

With this definition, ${}_{\text{trans}}\hat{\text{ID}}_3^{d,i}$ equals the naive benchmark model from [47] mentioned above. Now, what is left to define are f and g to determine start and end time of the interval. As we want to use the quarter hour before the start of the ID_3 interval, we define

$$
f(d) := d - \text{Dur}(225 \text{ MINUTES})
$$

and

$$
g(d) := d - \text{Dur}(210 \text{ MINUTES}).
$$

Their difference is always equal to Dur(15 MINUTES). Again with the example of the ID_3 of the hourly product 8 am - 9 am, the transactions from 4:15 am to 4:30 am are contained in $T^{d,i}_{\text{trans}}$.

It is worth mentioning that, even though ${}_{\text{trans}}\hat{\text{ID}}_3^{d,i}$ performs very well on hourly products as seen in [47], its performance regarding quarter hourly products fell behind the results of the other presented models in the publication. Nonetheless, it serves well as a benchmark to assess whether LOB statistics can contribute to ID_3 forecasting.

Regression forecast A simple way to increase the complexity of both models is to add an intercept as well as a slope to the models, i.e.

$$\tilde{P}^{d,i} = a_0^i + a_1^i \hat{P}^{d,i} \tag{3.9}$$

for $\hat{P}^{d,i} \in \{{}_{\text{mid}}\hat{\text{ID}}_3^{d,i}, {}_{\text{trans}}\hat{\text{ID}}_3^{d,i}\}$ and a_0^i, a_1^i the regression parameters for product type i.

For the resulting new models, the first step is to find adequate sizes for the calibration windows of the regression parameters. [44] present a model for the hourly ID_3 and test which calibration window size fits the model best. Their findings indicate that for ID_3 prices, the longest possible calibration window indeed performed the best with their forecasting model.

Equipped with these prediction models, we now move on to a case study on the German intraday market.

3.4 Case study on the German intraday market

The data we use for the case study are LOB data as well as transaction data from EPEX SPOT SE for the time from 01.01.2019 to 15.7.2019. This data set is split in training and test set, with the split appearing after three-quarters of the data set. Furthermore, all presented plots were produced using Python 3.6 [61] and the package seaborn 0.11.1 [64]. For calculation of the order book statistics, the Cython package 0.29.24 [1] as well as the Numpy package 1.19.5 [25] were used.

Figures 3.3 and 3.4 present the time series for hourly as well as quarter hourly ID_3 values. We find that in the order book data set belonging to that time period, some rare anomalies appear due to technical reasons on the side of the exchange, like the order book emptying and being filled only 20 minutes later again three hours before delivery, which is usually a very busy time in the order books. This is far from the norm and thus, we decided to remove periods where the order book is empty on at least one side for the calculation of mid prices and spreads. The affected time intervals are adjusted accordingly. Furthermore, we see spikes in the test set with values far over 200 that have not appeared in the training set. As we assume these spikes to not be well catchable by any estimator knowing only the presented training data and we cannot be sure whether they are again artificial artifacts in the data, they are removed prior to further analysis.

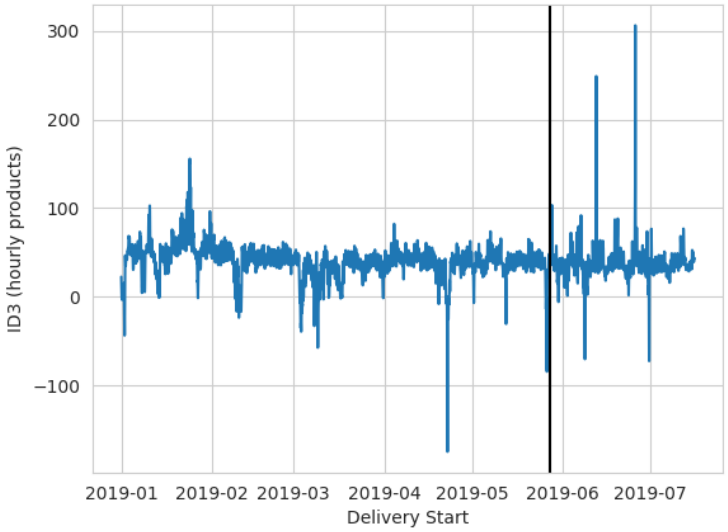

Figure 3.3: ID_3 values over the time horizon from 01.01.2019 to 15.07.2019 for hourly products. The black line indicates the split between training set and test set, the former being on the left and the latter on the right of the black line.

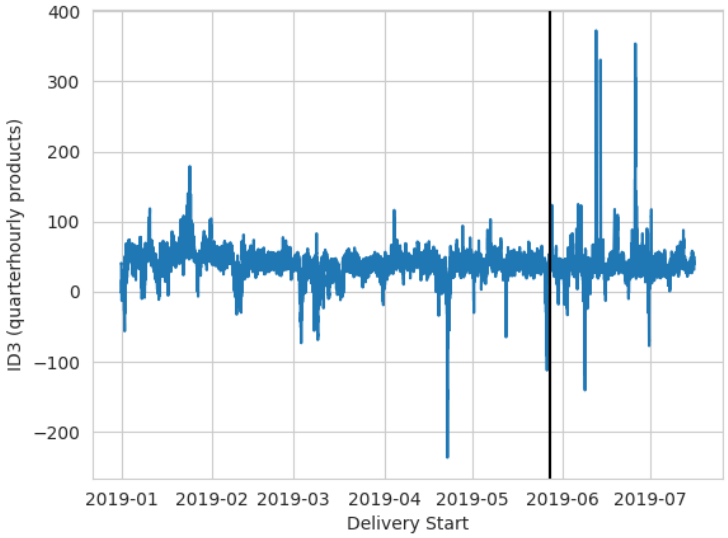

Figure 3.4: ID_3 values over the time horizon from 01.01.2019 to 15.07.2019 for quarter hourly products. The black line indicates the split between training set and test set, the former being on the left and the latter on the right of the black line.

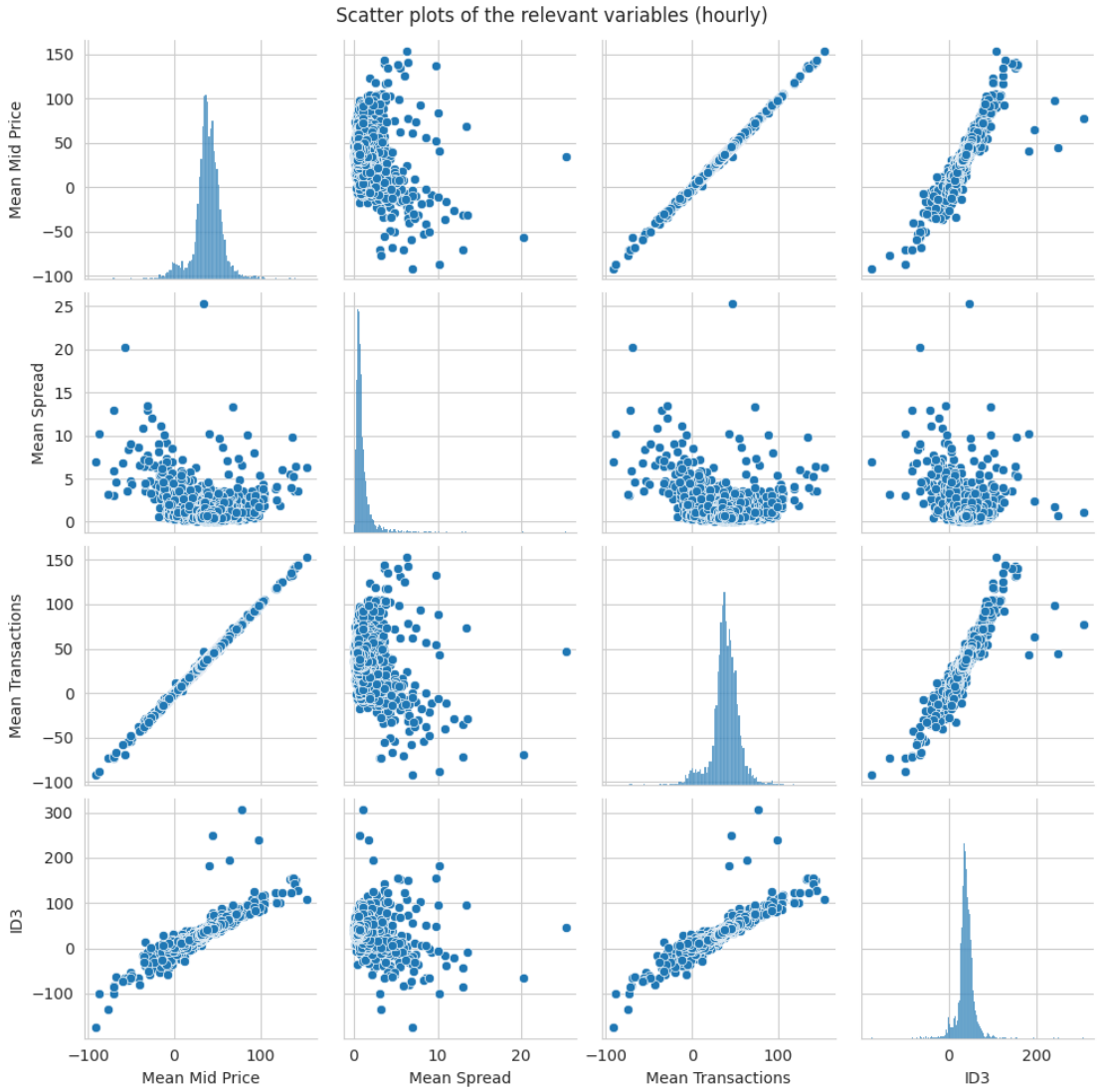

Figure 3.5: Scatter plots for LOB statistics and the ID_3 for hourly products, excluding order book asymmetry.

To assess which, if any, of the introduced LOB interval statistics have forecasting abilities regarding the ID_3 prices, we check in a first step whether we find a (linear) relationship between the LOB statistics and the ID_3 values as well as between the LOB statistics themselves. This is analyzed through the usage of scatter plots as well as through a correlation analysis. Figures 3.5 and 3.6 present scatter plots for hourly and quarter hourly products, respectively.

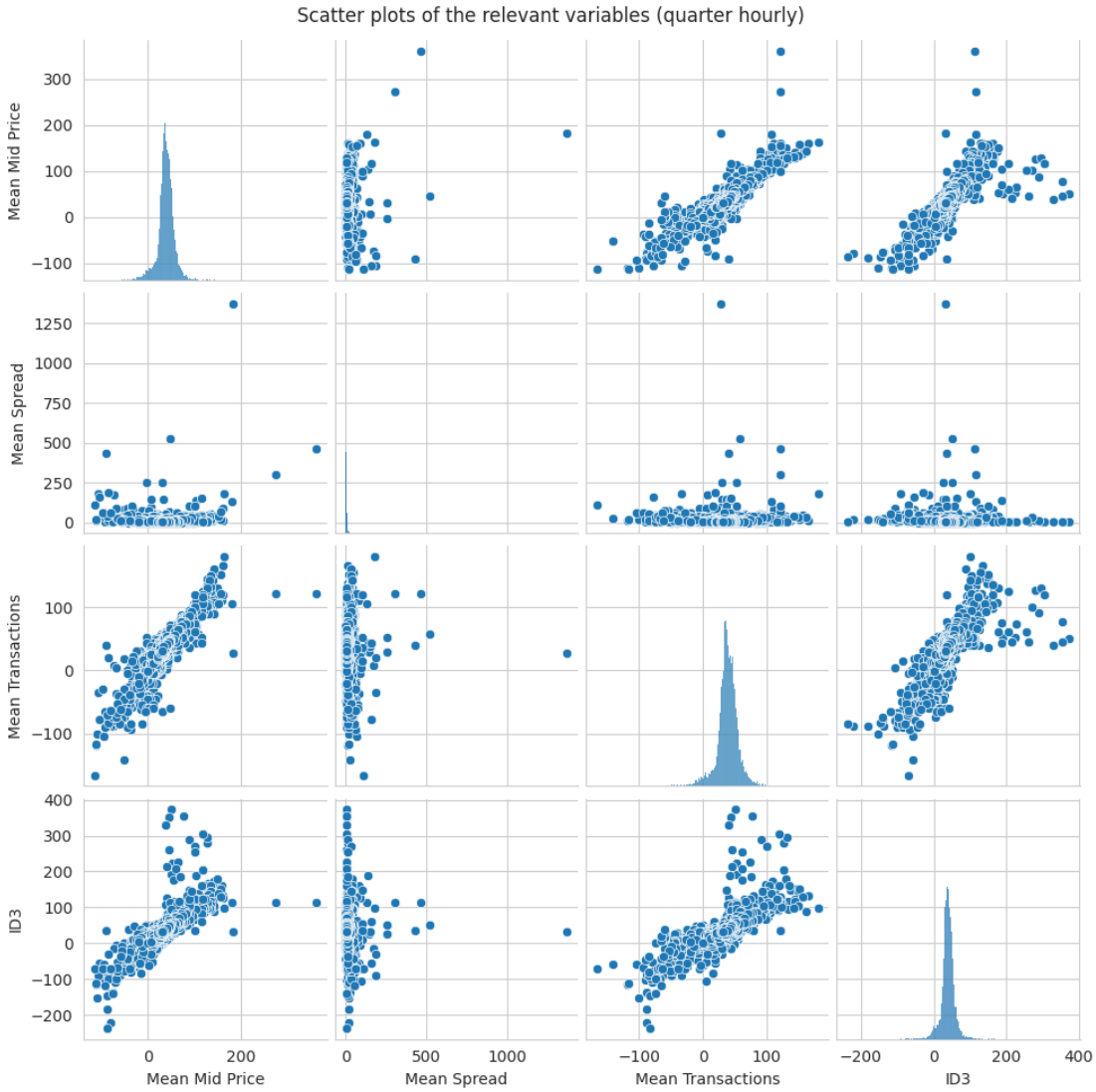

Figure 3.6: Scatter plots for the LOB statistics and the ID_3 for quarter hourly products, excluding order book asymmetry.

Concerning the scatter plots for hourly products, the most prominent feature is the almost perfectly linear relationship between mean transaction prices and mean mid prices. Furthermore, a less pronounced linear relationship between both statistics and ID_3 prices is visible. In comparison to that, the mean spread does not seem to have any sort of linear relationship with the ID_3 prices. A possible nonlinear relationship would require other analysis techniques which are beyond the scope of this thesis.

For quarter hourly products, the scatter plots present a similar picture. Mean transaction prices and mean mid prices still show a strong linear relationship, even though it

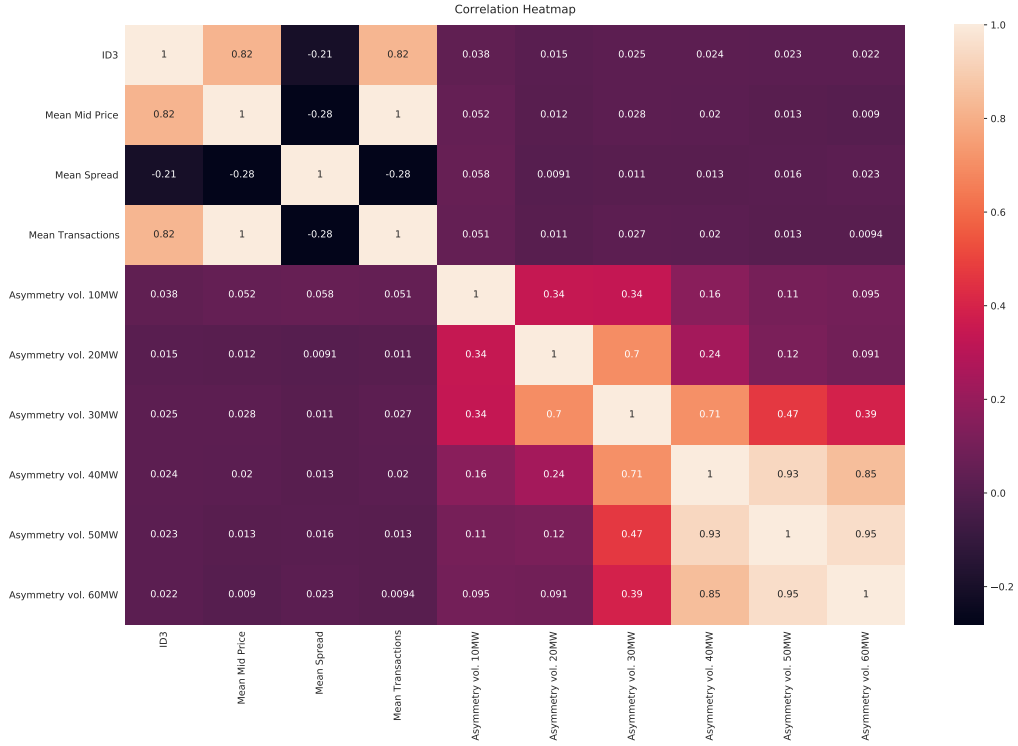

Figure 3.7: Correlation heatmap for LOB statistics and the ID_3 for hourly products.

is much less pronounced in comparison to the hourly products. As before, both have a linear relationship with the ID_3 prices, whereas the spread does not have a visible linear relationship with them.

Figures 3.7 and 3.8 show heat maps for hourly and quarter hourly products, respectively. Again, we notice the correlation of almost 1 for hourly products, and still 0.94 for quarter hourly products when comparing mean mid price and mean transaction price. Furthermore, both statistics show the highest correlation with the ID_3 for hourly as well as quarter hourly products. All asymmetry values show a positive correlation with ID_3 prices, which first grows in numbers with growing asymmetry and then diminishes again. Finally, the spread has a negative correlation with all considered statistics.

Again inspired by [47] and [60], we furthermore conduct an elastic net analysis that was first introduced by [70] for variable and model selection. Elastic nets are a regression method that combines the approaches from LASSO and ridge regression. Both are regression techniques that incorporate some sort of regularization, with LASSO also supporting variable selection. LASSO uses an l_1 regularization, ridge regression an l_2 regularization, and elastic nets combine both by summing them in a convex combination with choosable parameter λ. Furthermore, the decision for the usage of any regularization at all is made by choosing a second parameter α, with very small values for α

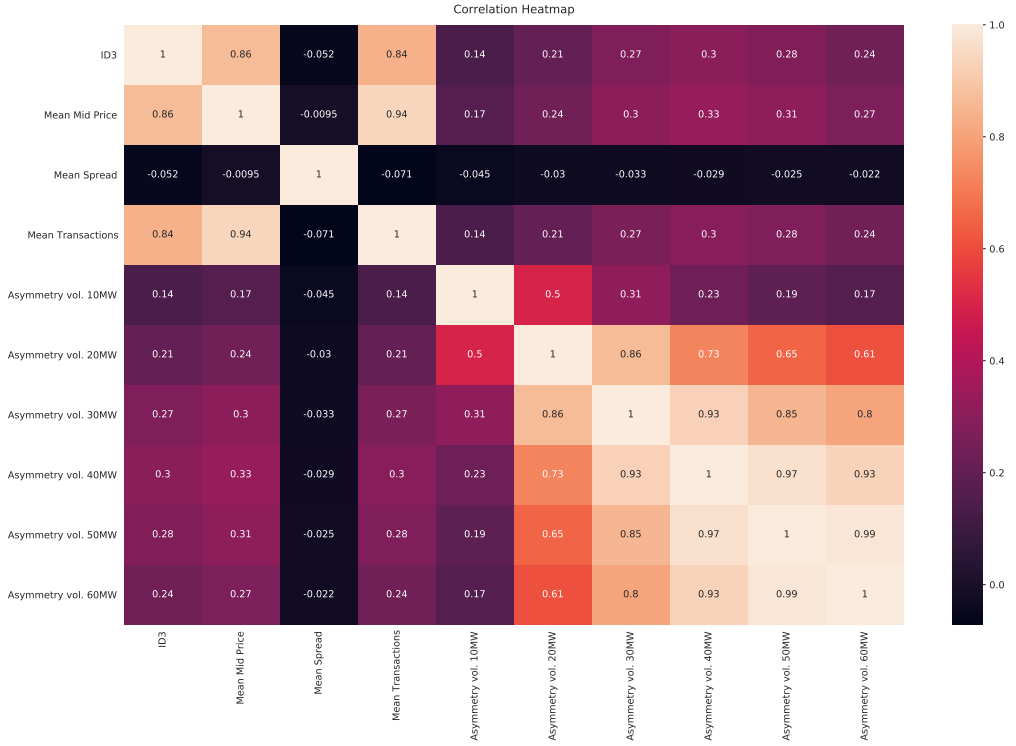

Figure 3.8: Correlation heatmap for LOB statistics and the ID_3 for quarter hourly products.

indicating a negligible influence of the regularization terms. We choose this method due to the strong correlation between transaction prices and mid prices, as ridge regression is known to handle that very well, while LASSO reduces unnecessary variables' coefficients to zero. The needed parameters can both be determined by cross validation techniques that are implemented in the SciKit-Learn Python package presented in [48].

We analyze the chosen coefficients for models trained on different time horizons: First, we look at the model chosen based on the whole training set, then at the models for every month in the training set and finally for every week. This procedure is chosen to capture possible time inhomogenities or dependencies in the data. We present the results for hourly products in Table 3.9, and the results for quarter hourly products can be found in Table 3.10.

Our main findings from the analysis of the LOB statistics are: For hourly products, transaction prices are deemed the most relevant indicator over the whole time horizon, closely followed by mid prices. This pattern repeats when looking at the monthly and weekly values, with a fluctuation between mostly mid prices and transaction prices regarding the biggest coefficient. Overall, they are incorporated in the models most of the

Time	Mid Price	Trans-actions	Spread	As. 10	As. 20	As. 30	As. 40	As. 50	As. 60
none	0.6	0.4	-0	0	0	0	0	0	0
Month 1	0.2	0.77	-0	-0.03	-0	0	0	0.01	0
Month 2	0.11	0.88	-0.06	0	-0	-0	0	0	-0
Month 3	0.78	0.19	0	0.02	0	0.01	0	0	0
Month 4	1.1	0	0	0	0.01	0	0	0	0
Month 5	0.54	0.52	-0.1	-0	-0	-0	-0	-0	-0
Week 1	0	0.95	-0.57	-0.03	0	-0.03	-0	0.02	0
Week 2	0.52	0.5	1.03	-0.27	0.08	0.01	0	0	0
Week 3	0.45	0.44	0	-0	0	0	0	0	0
Week 4	0.28	0.67	0	-0	-0	0	0	0	0
Week 5	0.54	0.49	1.32	0.09	-0.03	-0	-0.02	0	0.01
Week 6	0	0.98	-0	0	0	0	0	0	-0
Week 7	0.84	0	-0	-0	-0	-0	0	0	0
Week 8	0	0.93	0.99	0	0	0	0	0	0
Week 9	1.12	0	-0	0.03	0.02	0	0	0	-0
Week 10	0.42	0.43	-0	0.05	0.01	0.01	0	0	-0
Week 11	0.78	0.12	-0	-0	0	0	0	0	0
Week 12	0.48	0.54	1.76	0.01	-0.06	0.01	0.03	0	-0.01
Week 13	0.88	0	-0	0	0	0	0	0	0
Week 14	1.09	0	1.44	-0	0	0	0	0	0
Week 15	0	0.94	-0	0	0.03	0	-0	-0	-0
Week 16	0.92	0	-0	0	0	0	-0	-0	-0
Week 17	1.05	0.09	0	0	0.03	0	-0	0	0
Week 18	0.66	0.36	-0	-0	-0	-0	0	0	0
Week 19	0.64	0.4	-0.9	0.07	-0.03	-0	0	0	0
Week 20	0	0.89	-0	-0	-0	-0	-0	-0	-0
Week 21	1.16	0	0	0	-0	-0	-0	-0	-0
Week 22	0	0.76	-0.27	0	0	0	0	0	0

Table 3.9: Values assigned by elastic net to the different variables for hourly ID_3 products. The considered time intervals are the whole training set, each month and each week.

Time	Mid Price	Trans-actions	Spread	As. 10	As. 20	As. 30	As. 40	As. 50	As. 60
none	0.65	0.28	-0	0	0	0	0	0	0
Month 1	0.46	0.43	-0.06	-0	-0	-0	-0	0	0
Month 2	0.91	0.01	0.22	-0	-0	0	-0	0	-0
Month 3	0.79	0.09	0	-0	0	0	0	0	0
Month 4	0.61	0.46	0.1	-0	0	0	0	0	0
Month 5	0.87	0.06	0	0	0	0	0	0	0
Week 1	0.63	0.25	-0.18	-0	-0	0	0	-0	0
Week 2	0.71	0.28	0.22	-0	-0	-0	-0	-0	-0
Week 3	0.58	0.2	0	0	0	0	0	0	0
Week 4	0.57	0.41	-0.18	-0	-0	-0	-0	-0	0
Week 5	0.69	0.25	0.06	0	-0	-0	-0	-0	-0
Week 6	0.91	0	0.11	-0	-0	0	0	0	0
Week 7	0.52	0.25	0.13	0	-0	0	0	0	-0
Week 8	0.69	0.12	0	0	-0	-0	-0	-0	-0
Week 9	0.95	0.17	-0.09	-0	0	0	0	-0	0
Week 10	0.54	0.26	-0.05	0.01	-0	0	0	-0	0
Week 11	0.84	0	0.06	-0	-0	-0	0	0	0
Week 12	0.87	0.02	0	0	0	0	0	0	0
Week 13	0.62	0.23	-0	-0	0	0	0	0	0
Week 14	1.06	0.06	0	-0	0	0	0	0	0
Week 15	0.5	0.34	0.02	-0	-0	0	0	0	0
Week 16	0.89	0.02	-0.18	-0	0	0	0	0	0
Week 17	0.71	0.53	0.44	-0	-0	-0	-0	-0	0
Week 18	1.05	0	-0	0	-0	0	0	0	-0
Week 19	0.85	0.21	-0.38	0	-0	-0	0	0	0
Week 20	0.7	0.17	-0.15	-0	0	0	0	0	-0
Week 21	0.92	0	0.13	0	0	0	0	0	0
Week 22	0.37	0.3	-0	-0	-0	-0	-0	-0	-0

Table 3.10: Values assigned by elastic net to the different variables for quarter hourly ID_3 products. The considered time intervals are the whole training set, each month and each week.

time, but – most likely due to the strong linear relationship with transaction prices – we do not see a convergence of coefficients here. The spread is also included in roughly half of the models, but the sign of its coefficient varies. Same holds true for all asymmetry values. Therefore, including them in the model does not seem reasonable. Adding to the technical point of view, the models all decided for $\alpha \leqslant 0.1$, and consequently the resulting elastic net regression is close to an unregularized ordinary least-squares regression.

For quarter hourly products, the algorithm again chooses α very close to zero, we find $\alpha \leqslant 0.05$ for all considered constellations. Therefore, the regression is nearly identical to ordinary-least squares. In comparison to the hourly products, now the main component of nearly all models is the mid price, followed by transaction prices. The spread again is incorporated in more than half of the considered models, but its sign changes and thus, we will again not include it in our further analysis. Asymmetry values now are almost all equal to zero and are also not considered any further.

Based on this analysis, it seems unreasonable to fit the full model given in Equation (3.6), as most of the contained predictors would only add irrelevant or erroneous information. As a consequence, we decide to drop all explanatory variables except mid prices and transaction prices for both hourly and quarter hourly products. Furthermore, we will not combine these two variables in one model due to the strong correlation between them, but instead compare them against each other in their predictive ability. This leads us to analyzing the data based on the naive predictors presented in Equations (3.7) and (3.8).

Naive forecast We conduct our first analysis by using the models ${}_{\text{mid}}\hat{\text{ID}}_3^{d,i}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,i}$ defined in Equations (3.7) and (3.8). The error of the forecasted ID_3 prices in comparison to the actual observed ID_3 prices is calculated via mean absolute error (MAE) as well as via root mean squared error (RMSE), which are defined by

$$MAE(\hat{P}^i) = \frac{1}{|D^i|} \sum_{d \in D^i} |\text{ID}_3^{d,i} - \hat{P}^{d,i}|, \tag{3.10}$$

$$RMSE(\hat{P}^i) = \sqrt{\frac{1}{|D^i|} \sum_{d \in D^i} (\text{ID}_3^{d,i} - \hat{P}^{d,i})^2}, \tag{3.11}$$

where D^i is the set of all timestamps in the test set that a forecast is calculated for in regard to product type i, and $\hat{P}^{d,i} \in \{{}_{\text{mid}}\hat{\text{ID}}_3^{d,i}, {}_{\text{trans}}\hat{\text{ID}}_3^{d,i}\}$. Both are regularly used as error measures in the literature, see e.g. [47] or [36]. These error measures indicate differences in forecasting accuracy between both estimators. We present them for both models in Table 3.11.

The results show that in the naive version of the forecasts, ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$ perform similarly well. This is supported by an analysis using the adjusted Diebold-Mariano test from [26], where we cannot reject the hypothesis that ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$ and ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$ differ in accuracy regarding their forecasts. In contrast to that, though, we find that for quarter hourly

Estimator	MAE_h	$RMSE_h$	MAE_{qh}	$RMSE_{qh}$
${}_{\text{mid}}\hat{ID}_3^{d,i}$	4.02	10.64	6.04	12.52
${}_{\text{trans}}\hat{ID}_3^{d,i}$	4.01	10.62	6.72	13.78

Table 3.11: Forecasting errors for both considered naive models.

products, ${}_{\text{mid}}\hat{ID}_3^{d,\text{qh}}$ shows higher accuracy than ${}_{\text{trans}}\hat{ID}_3^{d,\text{qh}}$, with the p-value returned by the Diebold-Mariano test being 0.03. Thus, in our opinion, the informational gain from using ${}_{\text{mid}}\hat{ID}_3^{d,\text{qh}}$ does justify the somewhat higher computational effort that comes with it.

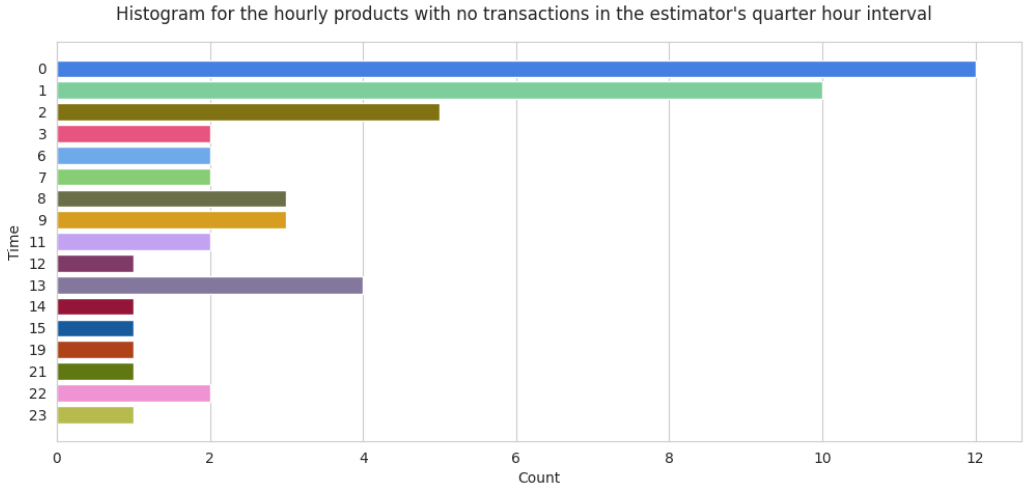

Figure 3.12: Counts of how often in the whole data set the observation period of the ${}_{\text{trans}}\hat{ID}_3^{d,\text{h}}$ did not contain any transactions.

When looking for the reason of this outperformance, the difference in the liquidity of the products comes to mind: In [47] it is shown that the liquidity of quarter-hourly products is about a quarter of the liquidity of hourly products. To test the hypothesis of this being the decisive point, as a first step we check if, and if so how often, there are no transactions in the estimators' quarter hour before the ID_3 time interval. Figure 3.12 and 3.13 present count plots for the hourly and the quarter hourly case, respectively.

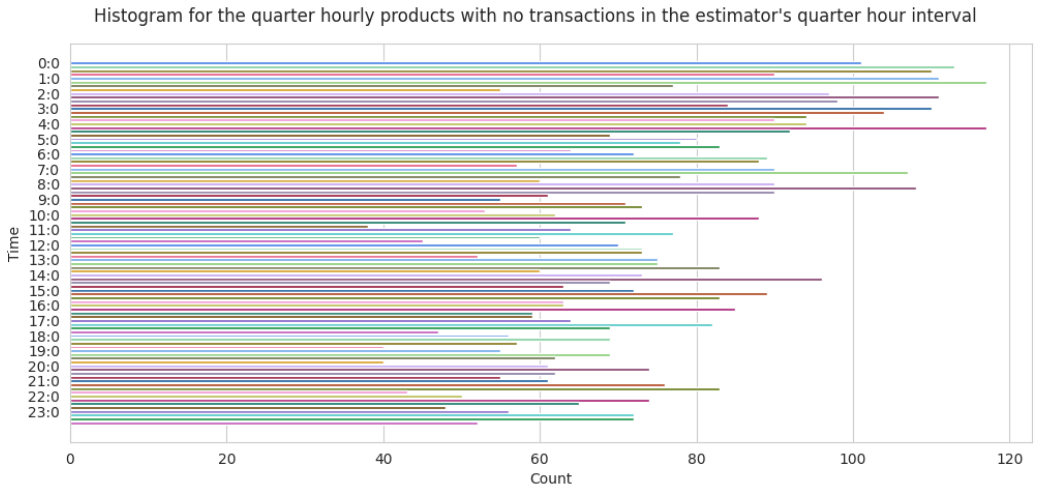

Figure 3.13: Counts of how often in the whole data set the observation period of the ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ did not contain any transactions.

It is visible that while the transaction-based estimators for the hourly products usually have transactions that can be used for forecasting, the situation for the quarter hourly transaction-based estimators is much worse. Here, in about 40% of all observations, no new information is given to the estimators and they refer back to the value of the corresponding day ahead product. Consequently, at least for the quarter hourly products, this difference in information might be the reason for the higher accuracy of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$.

To test the strength of the influence of the quarter hours without transactions information, we decide to remove the corresponding observations from the test data set, and repeat the analysis with the naive estimators on the trimmed test set. Results of this analysis can be found in Table 3.14. For the hourly estimators, only seven observations were removed from the test set, which is less than 1% of the observations. In comparison to that, 1,470 observations are removed for the quarter hourly products, which makes up about 31% of their total number. Both estimators for the hourly products profit visibly

Estimator	MAE_h	RMSE_h	MAE_{qh}	RMSE_{qh}
${}_{\text{mid}}\hat{\text{ID}}_3^{d,i}$	3.91	9.83	6.21	12.75
${}_{\text{trans}}\hat{\text{ID}}_3^{d,i}$	3.90	9.83	6.18	12.75

Table 3.14: Forecasting errors for both considered naive models.

from the removal of the observations where no transactions were recorded. This result seems somewhat astonishing for the hourly products, as only seven observations were removed here. A further analysis shows that the main reason for the errors' decline lies with one of the seven observations, where the ID_3 price was much higher than usually,

which led to a spike in errors for both estimators. We conclude that this does not give hints at structural properties of the estimators.

Turning to the quarter hourly products, the picture is more mixed: The errors of ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ decreased, which shows that there is significant value in the recent transaction prices in comparison to the corresponding day ahead price for this estimator. In contrast to that, the prediction accuracy of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ grew worse, and even though ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ does not significantly outperform ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$, it puts forward a smaller MAE than ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$. A deeper analysis finds that during midday, missing transaction observations occur less often than they should if they were equally distributed over all products. This leads to the question whether the ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ performs better in general for quarter hourly products outside of this area.

Consequently, we repeat our analysis for all individual products. This is also supported by the fact that the ID_3, even if it is usually given as a time series over all products of one type (hourly or quarter hourly), is nevertheless an individual index for each of these products. The results for hourly products are given in Table 3.15. Although for some hours the ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$ performs significantly better than the ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$, in general it is clear that no significant improvement is achieved.

When the ID_3 forecasts for the quarter hourly products are evaluated for each product individually, the picture changes: Now we see a clear outperformance of ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ by ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$. The error values and corresponding Diebold-Mariano p-values are given in Tables 3.16, 3.17 and 3.18. The headers are truncated due to the page layout: "mid" translates to ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$, "trans" translates to ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$, and DM 1 and DM 2 stand for the Diebold-Mariano test based on the l_1-norm and the l_2-norm, respectively. Looking at the products around midday, we do find that the ourperformance here is not as strong as it for most products from other times of the day. Thus, this explains the error growth for ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ after removing the observations with missing transactions.

In a second step, we analyze the introduced regression versions of the naive forecast.

Regression forecast Before analyzing the performance of the regression estimators ${}_{\text{trans}}\widetilde{\text{ID}}_3^{d,i}$ and ${}_{\text{mid}}\widetilde{\text{ID}}_3^{d,i}$, we need to choose calibration window sizes for their fit. For that, we perform a rolling window approach on the whole training data set, repeated with differing window sizes $w \in [1, 3000]$. In order to let all windows roll over the same data set, we start after the first 3000 data points, which are then used for the first calibration. This approach is detailed in the following: At every step, we forecast one point in time only. For each new hourly ID_3 price, new information is available that is included in the calibration. It is important to notice, though, that the most recent information tuple available for calibration are the ID_3 price and its estimate from four hours before, see Figure 3.19.

Hour	MAE ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$	RMSE ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$	MAE ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$	RMSE ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$	DM 1	DM 2
0	3.11	5.05	3.22	5.3	0	0.01
1	2.79	4.09	2.77	4.03	0.64	0.73
2	2.69	4.03	2.69	4.05	0.44	0.24
3	2.93	5.1	2.91	5	0.84	0.91
4	2.64	4.56	2.62	4.59	0.66	0.44
5	2.67	4.05	2.69	3.96	0.24	0.9
6	2.9	4.53	2.98	4.54	0.02	0.33
7	3.41	5.22	3.43	5.14	0.19	0.78
8	4.1	6.15	4.09	6.11	0.53	0.63
9	4.03	6.31	4.1	6.34	0.18	0.35
10	3.81	5.39	3.88	5.55	0.05	0.06
11	4.1	6.02	4.15	6.16	0.15	0.17
12	5.23	16.49	5.22	16.54	0.54	0.1
13	5.04	12.46	4.95	12.21	0.98	0.98
14	4.01	7.82	3.99	7.95	0.71	0.15
15	3.79	7.9	3.72	7.77	0.99	0.99
16	3.5	4.94	3.45	4.95	0.86	0.44
17	3.21	4.5	3.22	4.48	0.26	0.8
18	3.17	4.44	3.12	4.4	0.93	0.94
19	3.53	5.36	3.56	5.38	0.16	0.23
20	4.31	11.35	4.37	11.4	0.06	0.13
21	3.47	5.97	3.39	5.89	0.94	0.93
22	3.63	10.15	3.62	10.2	0.61	0.21

Table 3.15: Table containing MAEs and RMSEs for hourly product forecasts of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{h}}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{h}}$ as well as their significance level.

Now, for each calibration window size w, the calibration set as well as when the forecast takes place is rolled through the entire data set. This is depicted in Figure 3.20. The resulting mean absolute errors for $w \in [1, 3000]$ on the training data set and models ${}_{\text{mid}}\widetilde{\text{ID}}_3^{d,\text{h}}$, ${}_{\text{trans}}\widetilde{\text{ID}}_3^{d,\text{h}}$ are given in Figure 3.21 for hourly and in Figure 3.22 for quarter hourly products. For hourly products, they indicate the same behavior as described by [44], i.e. the bigger the calibration window, the better the forecast. In contrast to that, though, we also notice small errors for smaller window sizes around 260. These two patterns lead us to the decision to choose calibration window sizes of $w \in \{260, 3000\}$ for comparing the models on the test set for hourly products. The decision against even bigger calibration windows is due to the total size of the training data set, as increasing the calibration window is equivalent to decreasing the number of points for forecasting in the training set.

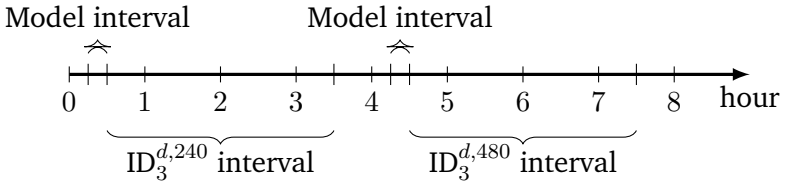

Figure 3.19: Example of available calibration data while forecasting the ID_3 for the hourly product 8 am - 9 am. At the beginning of the model interval, only information from four or more hours earlier is available.

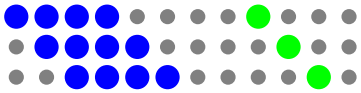

Figure 3.20: Example for rolling calibration window with window size $w = 4$. The blue dots represent the calibration set, the green one is the point where the forecast is taking place and the calibration is used, and the gray dots are data points that are not used in that iteration.

For quarter hourly products, the picture is similar when looking at big calibration window sizes for ${}_{\text{trans}}\widetilde{\text{ID}}_3^{d,\text{qh}}$, where we find the lowest error values. Interesting enough though, we see a strong dip in the MAE for a window size of $w = 3$ for ${}_{\text{mid}}\widetilde{\text{ID}}_3^{d,\text{qh}}$. Consequently, we decide for calibration window sizes of $w \in \{3, 3000\}$ for the test set of quarter hourly products.

Based on the calibration windows found in this first step, we follow the steps from before and compare the forecast performance of both estimators on the test set. We again use MAE as well as RMSE as error measures on the test set, with w defined as above. Results are given in Table 3.23 for hourly and in Table 3.24 for quarter hourly estimators.

We find a pattern similar to the one for the naive forecast for hourly products, with the transaction price model outperforming the mid price model. When comparing the values to the errors reached by the naive models in Table 3.11, we see no improvement regarding MAE. For the RMSE, though, both regression approaches outperform or at least equal their naive counterparts for $w = 260$.

For quarter hourly products, our findings mirror the results of the naive approach: Again, the mid price model outperforms the transaction price model and does so with a p-value of $p = 0.001$ for the l_1-norm and a value of $p = 0.02$ for the l_2-norm.

It holds true for both estimators that they are dominated by their respective naive counterpart regarding both MAE and RMSE. To further analyze this behavior, we study the performance for different price regimes. As the RMSE is more sensitive to extreme

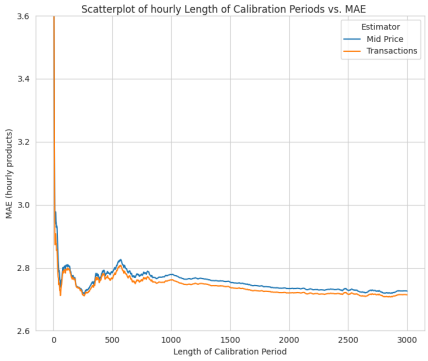

Figure 3.21: Mean absolute errors for both estimators on the training set for growing calibration window size and hourly products.

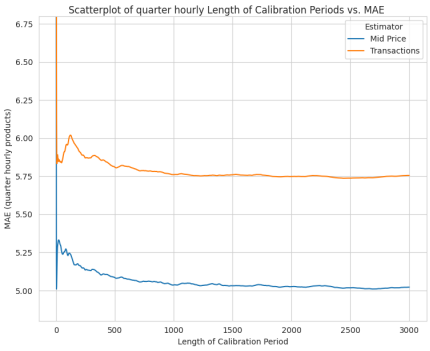

Figure 3.22: Mean absolute errors for both estimators on the training set for growing calibration window size and quarter hourly products.

errors in comparison to the MAE, we base the next step on the the lower RMSE errors for hourly products for the regression estimators: Repeating the error measurements for different price regimes in order to analyze which of the models performs best in different market situations, as we attribute some higher error values to extreme price regimes.

Forecast errors for price regimes Electricity index prices, as seen in Figure 2.5, tend to be stable around a certain value, with sometimes abrupt jumps up- or downwards. In order to analyze which model performs best in these differently behaved market situations, the first step is to split the observations into defined regimes. Here, we follow [60] and introduce three price regimes. [60] define positive, negative, and regular price regimes through

$$\text{pos.: } \mu^i + 3\sigma^i < \text{ID}_3^{d,i}, \quad \text{neg.: } \mu^i - 3\sigma^i > \text{ID}_3^{d,i}, \quad \text{reg.: } \mu^i + 3\sigma^i \geqslant \text{ID}_3^{d,i} \geqslant \mu^i - 3\sigma^i.$$

The data in the test set is split into these groups and errors are calculated separately for each of them. Our analysis includes the naive models as well as the regression models. Tables 3.25 and 3.26 hold the results for hourly and quarter hourly products, respectively.

We find that for hourly products and the regular price regime, the naive models perform best as was to be expected based on all former results. For all error measures regarding positive or negative price regimes though, the regression models outperform their naive counterparts. Here, we notice that in the case of the negative price regime and MAE, the mid price model actually outperforms the transaction price model. Nonetheless, these results have to be handled with caution, as the positive price regime

only contains 8 observations, and the negative one only 10.

In comparison to that, the price regimes for quarter hourly products have a few more observations, namely 46 in the case of positive and 33 for the case of negative spikes. We find that for quarter hourly products and positive price regimes, the regression models again outperform their naive counterparts, which is most visible for both estimators with $w = 3$. In comparison to the hourly products, though, we don't see an improvement for the negative price regime. The regular price regime is again dominated by the naive estimators. A possible reason for these findings is that in market situations based on the positive price regime, we see a stronger autocorrelation in the prices in comparison to negative or regular price regimes.

3.5 Discussion and Conclusion

This chapter treats the question whether the usage of limit order book statistics has the potential to enhance existing models for ID_3 prices. Based on our analysis, LOB derived time-weighted mid prices indeed contain much information about index prices. For hourly products, this information does not seem to enhance the forecasting performance notably, as the mid price based estimator is slightly outperformed by the naive benchmark for most hours. Furthermore, since working with LOB data requires somewhat more computational resources than working with transaction data only, established estimators seem preferable for hourly ID_3 products. For quarter hourly products, however, the time-weighted mid prices carry significantly more information than the transaction prices, which is highlighted by the superior performance of the mid price estimator with respect to MAE and RSME for most quarter hours.A closer analysis yields that the smaller liquidity of quarter hourly ID_3 products in comparison to hourly ID_3 products is the main reason for this outperformance. Less liquidity equals less transactions in total, and in a non-negligible amount of times leads to no transactions in the estimator's time interval. Thus, the transaction-based estimator has a worse data base in comparison to its hourly counterpart. Based on this analysis, we do strongly recommend to include limit order book information – especially time-weighted mid prices – in full information models for the ID_3 when quarter hourly products are concerned.

A simple increase of model complexity through usage of a regression did not yield the anticipated improvement of forecasting accuracy. Nonetheless, we find that for extreme price regimes, regression models tend to outperform their naive counterparts, which was strongly visible with quarter hourly products and the positive price regime. Even though these results have to be handled with caution due to small observation numbers for both regimes, it might be worthwhile to include order book information, and here again mainly the time-weighted mid price, to estimators that target exactly these extreme price regimes.

Our results show that a structurally different approach to ID_3 price forecasting – relying on a time-weighting and neglecting every volume-related information – is on par with existing transaction-based estimators. Moreover, it can even contain more information than the volume-weighted transaction data themselves, which in itself might be worth further research. Furthermore, using limit order book information for the forecast of half hourly ID_3 prices is an obvious next step, as the half hourly ID_3 is the least liquid in comparison to its hourly and quarter hourly counterpart and often has no transactions at all over the course of the whole order book opening time. Therefore, transaction data provide no helpful information for forecasting, whereas the LOB does contain orders and thus mid prices that could enhance an estimator's performance. Finally, an extended analysis of extreme price regimes based on a bigger data source could yield information on whether regression models consistently perform better for extreme price regimes and thus could improve estimators' accuracies further.

Hour	Minute	MAE mid	RMSE mid	MAE trans	RMSE trans	DM 1	DM 2
0	0	5.56	7.33	6.65	8.88	0	0
0	15	5.2	7.41	5.63	7.87	0.03	0.01
0	30	4.79	7.44	5.74	8.36	0	0.01
0	45	6.12	9.25	6.62	9.46	0.06	0.37
1	0	4.32	5.75	5.48	8.1	0	0
1	15	4.56	6.64	4.97	6.98	0.13	0.27
1	30	4.51	6.43	4.83	7.03	0.11	0.08
1	45	4.97	7.23	5.43	8.32	0.03	0.02
2	0	4.24	5.85	5.16	8.43	0.01	0.06
2	15	5.07	12.16	4.72	6.33	0.69	0.84
2	30	4.33	6.4	4.81	6.78	0.05	0.25
2	45	4.93	7.38	5.99	9.78	0	0.02
3	0	5.46	11.61	5.57	8.04	0.44	0.81
3	15	4.87	7.18	5.42	8.34	0.01	0.09
3	30	4.84	7.07	5.12	7.01	0.07	0.61
3	45	4.92	7.27	5.46	8.11	0.05	0.17
4	0	5.06	7.93	5.54	9.12	0.14	0.22
4	15	4.23	5.95	5.15	7.66	0	0.05
4	30	4.54	6.45	5.21	7.67	0.01	0.07
4	45	4.33	6.04	5.68	8.83	0	0
5	0	4.55	6.53	5.36	7.48	0	0
5	15	4.45	6.72	5.21	8.24	0.02	0.07
5	30	4.31	6.36	5.03	8.08	0.03	0.09
5	45	4.33	5.93	5.41	7.68	0	0.01
6	0	5	7.66	6.9	10.02	0	0
6	15	4.91	6.82	5.69	8.58	0.01	0.09
6	30	4.57	6	5.66	8.32	0	0
6	45	5.23	7.44	6.07	9.18	0.02	0.02
7	0	5.11	7.26	6.44	9.99	0	0.05
7	15	5.55	7.7	6.66	10.93	0.01	0.04
7	30	6.05	9.96	7	12.32	0.01	0.02
7	45	5.97	10.06	6.82	11.54	0.01	0.06
8	0	6.18	9.12	8.12	13.13	0	0.02
8	15	6.3	9.62	7.22	11.76	0.01	0.01
8	30	5.84	8.84	6.73	11.54	0.01	0.09
8	45	6.41	10.06	7.31	11.02	0	0.01
9	0	6.01	8.85	6.63	10.08	0	0
9	15	5.92	8.62	6.08	8.97	0.23	0.24
9	30	5.58	8.21	6.22	9.9	0.01	0.01
9	45	6.09	9.32	6.59	9.84	0.01	0.07

Table 3.16: Table containing MAEs and RMSEs for quarter hourly product forecasts of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ as well as their significance level.

Hour	Minute	MAE M	RMSE M	MAE T	RMSE T	DM 1	DM 2
10	0	5.66	7.73	6.57	9.06	0	0
10	15	5.29	7.57	5.56	8.08	0.12	0.14
10	30	6.08	8.21	6.52	9.72	0.09	0.07
10	45	6.49	9.51	7.35	11.57	0.03	0.07
11	0	6.14	9.85	6.64	10.4	0.12	0.35
11	15	6.39	10.08	6.72	10.77	0.15	0.2
11	30	7.35	13.85	7.62	14.3	0.22	0.18
11	45	6.71	9.95	7.03	10.61	0.2	0.28
12	0	6.82	15.51	7.12	15.92	0.13	0.11
12	15	6.42	10.5	6.36	10.15	0.56	0.65
12	30	8.31	19.86	8.62	20.47	0.19	0.22
12	45	8.33	17.31	8.93	17.27	0.09	0.52
13	0	6.57	11.48	7.55	14.74	0	0.11
13	15	6.02	10.07	6.87	13.06	0.01	0.06
13	30	6.09	9.94	6.04	9.36	0.55	0.81
13	45	6.32	10.4	7.14	11.92	0	0.01
14	0	6.03	9.81	7.22	10.99	0.01	0.01
14	15	5.86	9.77	7.01	11.48	0	0
14	30	7.02	14.42	7.42	13.78	0.11	0.84
14	45	6.84	13.93	7.44	14.53	0	0.01
15	0	5.53	8.51	6.53	9.91	0	0.03
15	15	5.62	10.39	6.79	11.67	0	0.01
15	30	5.88	8.72	6.13	8.7	0.2	0.52
15	45	5.74	7.74	6.81	9.66	0	0.01
16	0	5.66	9.22	6.38	9.98	0	0.01
16	15	5.29	7.91	5.97	10.95	0.09	0.14
16	30	5.26	7.6	5.96	10.13	0	0.11
16	45	5.33	7.54	6.76	9.23	0	0
17	0	5.35	7.87	7.34	12.33	0	0.01
17	15	5.27	8.27	5.33	7.79	0.45	0.63
17	30	4.66	6.83	5.74	9.36	0	0.05
17	45	4.57	7.2	5.89	10.43	0	0.06
18	0	4.69	6.54	5.65	7.7	0	0
18	15	4.8	6.38	5.17	6.93	0.01	0.01
18	30	5.99	18.86	5.12	7.39	0.79	0.84
18	45	6.46	14.28	6.53	11.24	0.47	0.72
19	0	5.24	7.9	5.97	8.41	0	0.01
19	15	5.11	7.34	6.06	9.35	0	0
19	30	5.87	10.01	6.61	10.58	0	0.01
19	45	6.97	14.34	7.28	14.78	0.05	0.06

Table 3.17: Continued: Table containing MAEs and RMSEs for quarter hourly product forecasts of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ as well as their significance level.

Hour	Minute	MAE M	RMSE M	MAE T	RMSE T	DM 1	DM 2
20	0	6.83	14.76	7.67	15.39	0	0.01
20	15	5.04	7.41	5.66	8.58	0.03	0.03
20	30	6.32	16.33	7.1	16.85	0	0.1
20	45	5.96	13.65	6.66	14.16	0.01	0.23
21	0	5.93	14.58	6.76	15.13	0	0
21	15	5.99	14.08	6.92	16.51	0.02	0.17
21	30	5.93	12.83	6.67	13.69	0.02	0.06
21	45	5.96	11.26	6.94	11.88	0	0.17
22	0	4.48	6.22	5.46	7.77	0	0
22	15	5.03	10.31	5.85	10.99	0	0.04
22	30	5	6.36	5.72	7.33	0.01	0.02
22	45	5.18	7.23	5.83	8.25	0.03	0.05
23	0	4.49	6.73	6.02	9.09	0	0.03
23	15	4.43	7.14	5.58	9.59	0	0
23	30	5.08	8.64	6.18	11.43	0.01	0.01
23	45	5.65	9.24	6.5	10.08	0.04	0.28

Table 3.18: Continued: Table containing MAEs and RMSEs for quarter hourly product forecasts of ${}_{\text{mid}}\hat{\text{ID}}_3^{d,\text{qh}}$ and ${}_{\text{trans}}\hat{\text{ID}}_3^{d,\text{qh}}$ as well as their significance level.

Regression Model	MAE_h $(w = 260)$	RMSE_h $(w = 260)$	MAE_h $(w = 3000)$	RMSE_h $(w = 3000)$
${}_{\text{mid}}\widetilde{\text{ID}}_3^{d,i}$	4.32	10.71	4.06	10.63
${}_{\text{trans}}\widetilde{\text{ID}}_3^{d,i}$	4.31	10.69	4.05	10.62

Table 3.23: Forecasting errors for all hourly regression models.

Regression Model	MAE_{qh} $(w = 3)$	RMSE_{qh} $(w = 3)$	MAE_{qh} $(w = 3000)$	RMSE_{qh} $(w = 3000)$
${}_{\text{mid}}\widetilde{\text{ID}}_3^{d,i}$	6.38	14.04	6.13	12.56
${}_{\text{trans}}\widetilde{\text{ID}}_3^{d,i}$	7.05	15.03	6.81	13.82

Table 3.24: Forecasting errors for all quarter hourly regression models.

Model	MAE_h pos.	$RMSE_h$ pos.	MAE_h neg.	$RMSE_h$ neg.	MAE_h reg.	$RMSE_h$ reg.
${}_{\text{mid}}\hat{ID}_3^{d,i}$	86.95	111.94	11.19	12.59	3.39	5.13
${}_{\text{trans}}\hat{ID}_3^{d,i}$	85.87	111.57	11.32	12.45	3.38	5.16
${}_{\text{mid}}\widetilde{ID}_3^{d,i}$ with $w = 3000$	86.56	111.64	11.09	12.38	3.43	5.16
${}_{\text{trans}}\widetilde{ID}_3^{d,i}$ with $w = 3000$	85.49	111.27	11.23	12.25	3.42	5.18
${}_{\text{mid}}\widetilde{ID}_3^{d,i}$ with $w = 260$	85.97	111.31	10.74	11.75	3.69	5.38
${}_{\text{trans}}\widetilde{ID}_3^{d,i}$ with $w = 260$	84.85	110.9	10.84	11.53	3.69	5.4

Table 3.25: Forecasting errors for all models and three price regimes for hourly products.

Model	MAE_{qh} pos.	$RMSE_{qh}$ pos.	MAE_{qh} neg.	$RMSE_{qh}$ neg.	MAE_{qh} reg.	$RMSE_{qh}$ reg.
${}_{\text{mid}}\hat{ID}_3^{d,i}$	75.3	95.88	21.64	25.6	5.24	7.92
${}_{\text{trans}}\hat{ID}_3^{d,i}$	82.42	100.42	23.77	28.93	5.84	9.28
${}_{\text{mid}}\widetilde{ID}_3^{d,i}$ with $w = 3000$	74.93	95.6	21.68	25.7	5.32	8.01
${}_{\text{trans}}\widetilde{ID}_3^{d,i}$ with $w = 3000$	82.32	100.38	23.52	28.73	5.93	9.35
${}_{\text{mid}}\widetilde{ID}_3^{d,i}$ with $w = 3$	57.35	76.07	29.25	33.5	5.71	11.6
${}_{\text{trans}}\widetilde{ID}_3^{d,i}$ with $w = 3$	62.8	78.51	30.45	36.75	6.32	12.58

Table 3.26: Forecasting errors for all models and three price regimes for quarter hourly products.

4 Modelling, prediction and optimization on the day ahead market

The day ahead market is, as we saw in Section 2.1, still the most important market for companies to buy and sell short-term electricity. An immediate consequence is that there are many people with a need to model, predict and eventually optimize on this market in order to either make more money with the electricity they have or to spend less for the electricity they need. The latter is mainly then a relevant point when the possibility exists to shift the own demand or supply in order to match the need of the market to sell or buy, respectively. Due to this fact, there exists much literature on how to model and predict the day ahead market and some on how to optimize on it. Most articles concentrate on how to predict the day ahead market of the very next day. Due to this research, the day ahead prices of the next day are fairly well known beforehand. That is not necessarily true for the days from day two onward, even though knowledge about later days is needed if an optimization is concerned with more than the very short-term view. To adress this problem, we model the day ahead price for the time window of a week, condense the modelled paths to a manageable size for optimization purposes, and finally optimize the use of a battery that trades based on these predictions.

The sections are structured as follows: Section 4.1 gives a deeper insight in how to model the day ahead market for the days two to seven as prediction horizon and presents which price model we are going to use. This is followed by Section 4.2, where we concentrate on how to condense the modelled paths in a reasonable fashion in order to prepare them for optimization. We elaborate on how to build scenario trees from modelled paths and how to shrink such a tree from big to small size. Furthermore, we present own approaches and heuristics for programming and using these techniques in practice. Finally, the theoretical part is concluded by Section 4.3, where several methods of how to optimize on scenario trees are presented and their advantages and drawbacks are discussed. The knowledge from these three sections is then applied to data from the German day ahead market in Section 4.4. The chapter is concluded by Section 4.5, which sums up results from the previous section and gives an outlook to further research.

4.1 Modelling of the day ahead market

The survey [65] treats five different classes of electricity price modelling approaches, namely:

- **Multi-agent models**: Models containing agents that interact with each other; the market and price curves are formed through this interaction.
- **Fundamental models**: Usually elaborate models that contain physical information about power plants, grid capacities, weather among others, as well as economic information e.g. about development plans. Prices are modelled as result of dynamics considering these information sources.
- **Reduced-form models**: Black-box models that do not try to explain electricity prices but instead aim to capture inherent properties of the price movements.
- **Statistical models**: Models that follow from an application of statistical techniques, either directly on the electricity price or on its influencing factors.
- **Computational intelligence models**: Models based on machine learning, evolution or fuzziness that try to capture the underlying dynamic system.

All of these modelling approaches have seen a surge in interest after the liberalization of many electricity markets from the early 1990s onward [3]. Models from all these classes often have similar, though not identical, scopes of application. Multi-agent models are suitable for markets that are highly regulated and hardly subject to price fluctuations, whereas the other model classes can also represent more heterogeneous markets [65]. Statistical models as well as computational intelligence models are often located in the short-term forecast range and serve to predict the prices of the next day, whereas fundamental models and reduced-form models operate in the medium forecast range of several days up to a few months and thus are more suitable for risk management. As it is our goal to optimize over the time horizon of one week, we need to consider medium term forecasts. This leads to the usage of either a fundamental or a reduced-form model. Since fundamental models are immanently based on the existence and availability of huge amounts of well-prepared data, we choose the approach of reduced-form models instead. Prominent examples of models from this class are regime-switching and jump-diffusion models, see [65]. We concentrate on the latter in the following.

Expanding the ideas from the classic stochastic model for electricity spot prices in [57], where the price is defined through the exponential of an Ornstein-Uhlenbeck process, [3] introduce the idea of factor models. These are spot price models that contain several factors, where each factor is defined as a non-Gaussian Ornstein-Uhlenbeck process, to capture different traits and behaviors exhibited by spot prices. In order to apply this modelling class to the German day ahead price market, we first need to identify which features are present there.

4.1.1 Analysis of the German day ahead market

When gathering insights about the German day ahead market, it is an obvious step to look on the one hand at existing literature and on the other hand at historical spot price

data. Regarding the literature, we follow [29] who analyzed German day ahead data and its stylized features. They identify seven distinct features and test for them on the historical data from 2011 to 2016:

- **Seasonality**: Because electricity use depends on people's consumption and therefore their habits, patterns in consumption are also transferred to the electricity price. Next to other influencing variables like e.g. the weather, this creates daily, weekly and annual seasonalities that are reflected in the electricity price. Holiday effects or those of major events can also be observed. Usually, this seasonal and cyclical behavior is represented in the literature by means of a seasonality function. This function often takes either a sinusoidal form or contains dummy variables. The former choice allows for a continuous seasonality function with the advantages that result from it, whereas the latter choice is better suited to integrate singular events like holidays.

- **Negative prices**: Since negative prices are allowed on the spot market, they do appear in some hours of the year. The main reason for negative prices is too much electricity supply combined with too little demand. There are several reasons why this situation occurs more often today than in the past. For one, the greater amount of renewable energy in the grid has shifted the merit order curve into regions of generally lower prices because of their zero marginal costs, see Figure 4.1. Secondly, some types of power plants have long start-up times, which means that the costs of shutting down and starting up again are higher than temporarily generating too much supply in minimum operation mode and thus having to pay negative electricity prices.

- **Mean reversion**: As is often the case with commodities, electricity prices show a return to their seasonal mean, see e.g. [23], [29].

- **Spikes**: These are defined by strong upward or downward price movements that occur from time to time, which come quickly and after which the price also quickly settles back at its previous level.

- **Autocorrelation**: According to [46], the autocorrelation of spot prices is well represented by
$$\rho(t) = \rho_1 e^{-t/\alpha_1} + \rho_2 e^{-t/\alpha_2},$$
where the parameters can be fitted to spot price data, and according to [29], this equation is often used to specify mean reversion speeds in spot price models. They furthermore present another method of how to estimate this mean reversion speed.

- **Stationarity**: Various studies find that spot prices exhibit stationary behavior, cf. [46], [29].

- **Non-Gaussian distribution**: Based on analyses of the moments of spot prices e.g. in [46], [23], [29], it is widely accepted that the prices are not Gaussian.

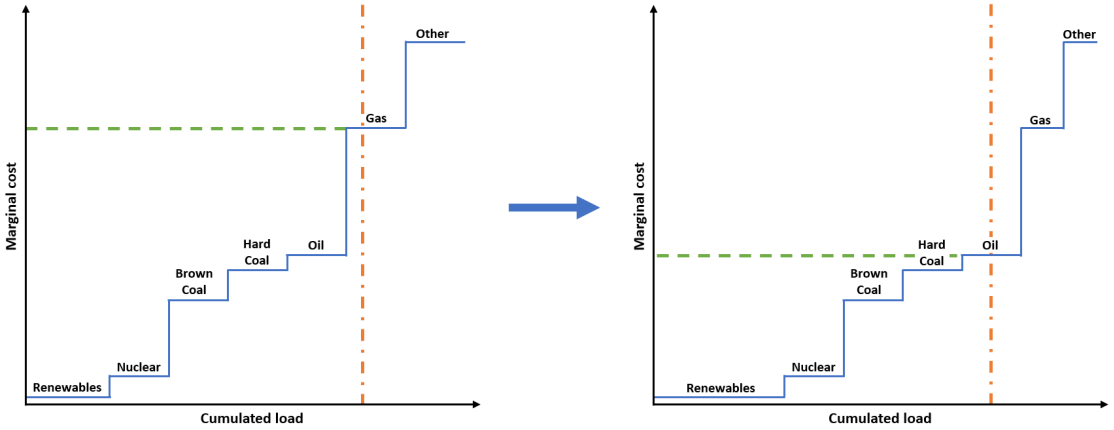

Figure 4.1: Display of changing marginal costs in situations with equal demand, but more renewable electricity. More renewables cause the current load (orange) to cross the merit order curve at lower marginal costs (green).

We will now analyze whether these stylized features still match with more recent spot prices on the German day ahead market: Figure 4.2 shows all spot prices from the year 2015 up to 2020, whereas Figure 4.4 shows two extracted weeks from May 2019. In direct contrast, we also take a look at the time interval from July 2021 to June 2022 in Figure 4.3 and two weeks from May 2022 in Figure 4.5. It is noticeable that prices underwent a strong systematic change beginning roughly in September 2021. This change is visible in much higher price levels as well as in a much higher volatility of prices. It can be traced back in its beginnings to Russia using its market maker power to increase price levels for gas drastically and afterwards on the sanctions against Russia from the EU [31]. As gas is the fossil with the ability to react the fastest to deviations in supply or demand, the diminished infeed from it increased prices and volatility. As important as well as concerning as the new price levels are, they are politically driven and likely do not represent how spot prices behave regularly. Therefore, we concentrate on the former time interval, the years 2015 to 2020, in order to determine typical spot price behavior.

Remark 4.1. Regarding all box plots that follow, the middle line depicted represents the median, the ends of each box are given by the 25% and 75% quantile, respectively, and the length of the vertical bars below and above the boxes is determined by the 5% and the 95% quantile, respectively.

Regarding yearly patterns, we notice a pattern with higher electricity prices in winters and lower prices in summers, also visible in Figure 4.6. This is explained by a higher need for electricity in Germany during the winter due to heating and less day light compared to summer times.

Furthermore, Figure 4.4 shows a strong weekly as well as daily pattern: The weekly pattern consists of higher prices during the week than on the weekend, whereas the daily pattern presents two typical peaks during the day, the first around 8am, and the second

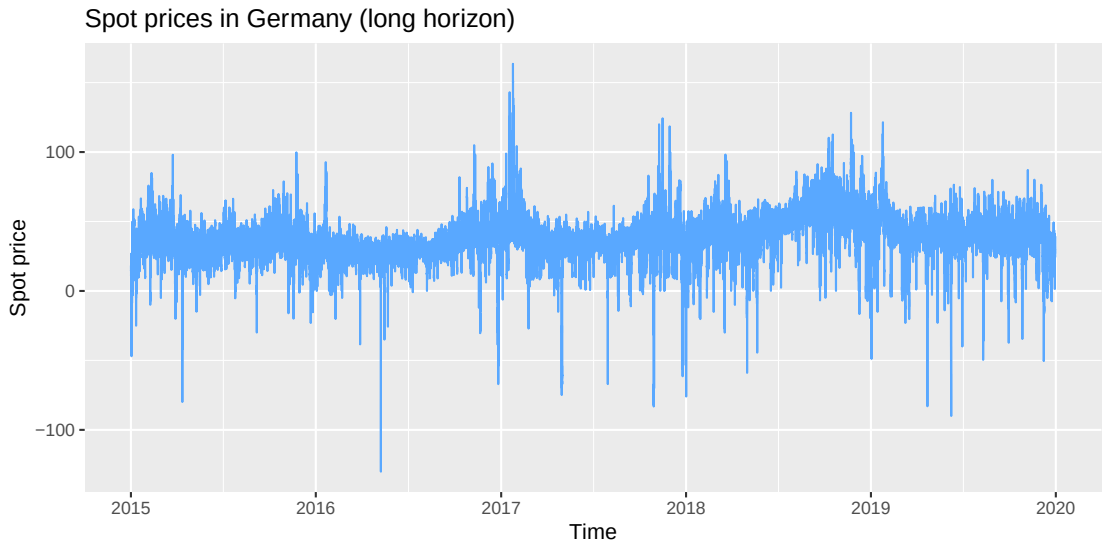

Figure 4.2: Hourly electricity German spot prices from the year 2015 up to 2020.

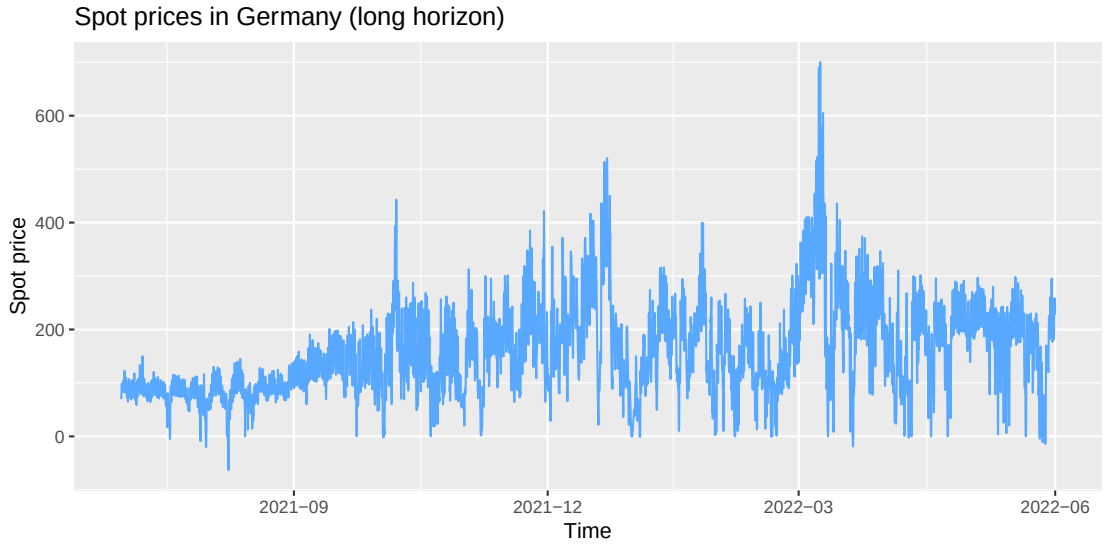

Figure 4.3: Hourly electricity German spot prices from July 2021 up to June 2022.

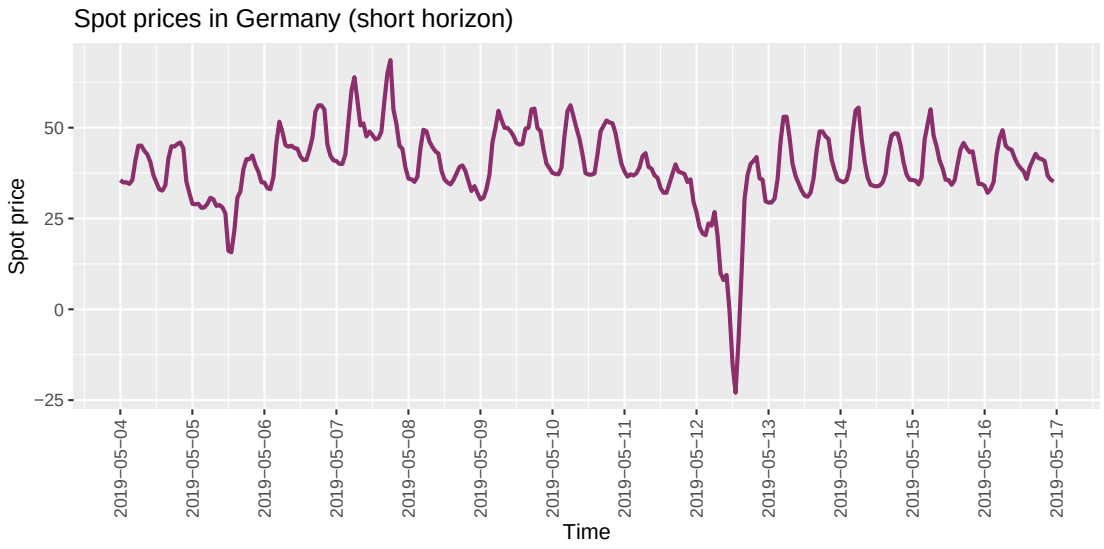

Figure 4.4: Hourly electricity German spot prices of two weeks in May 2019.

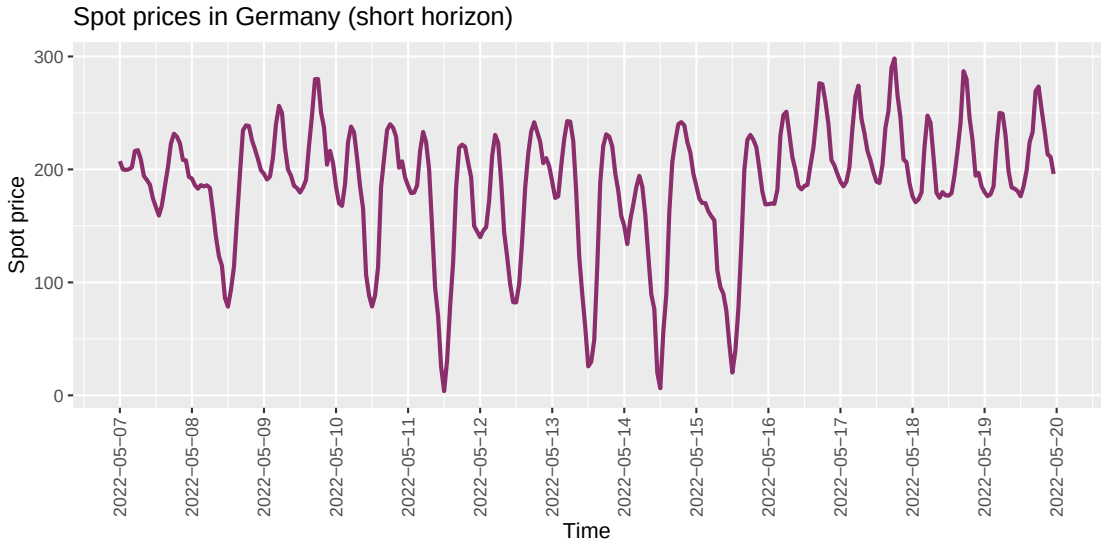

Figure 4.5: Hourly electricity German spot prices of two weeks in May 2022.

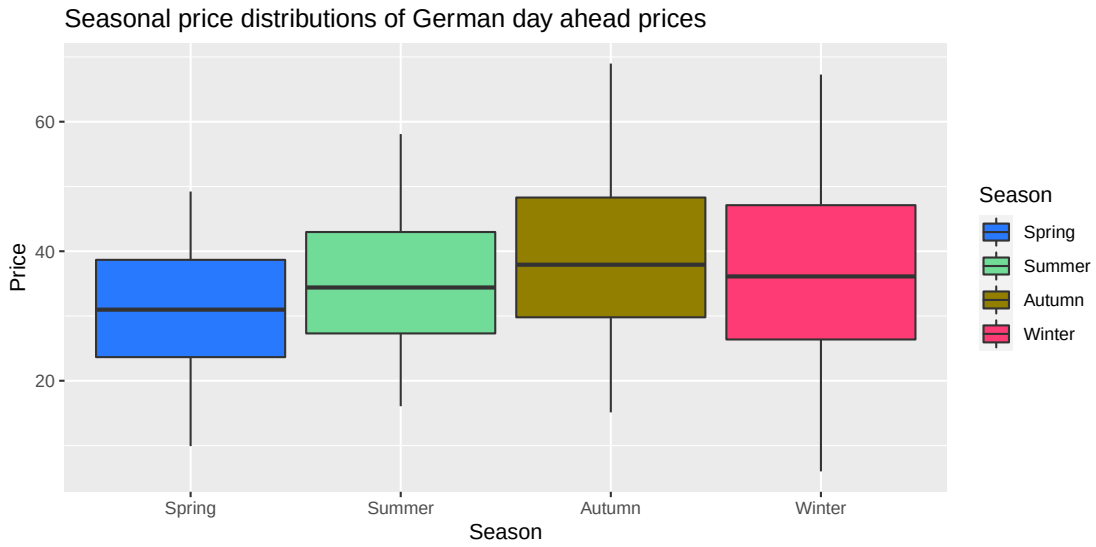

Figure 4.6: Seasonal spot price distributions in Germany expressed through boxplots. Spring covers the months March to May, summer covers June to August, autumn covers September to November and finally winter goes from December to February.

Neg. share	Min	Mean	5%	25%	50%	75%	95%
1.66%	-130.09	-17.10279	-70.064	-19.9575	-7.135	-1.93	-0.05

Table 4.10: Statistics of negative prices on the German spot market from 2015 to 2020

around 7pm. This is confirmed in Figure 4.7, which shows spot prices grouped by their corresponding weekday, and in Figure 4.8, which presents mean hourly spot prices. Both figures span the time interval from 2015 to 2020 in Germany. In comparison to that, we also present the same figure for the time interval from July 2021 to June 2020 in Figure 4.9. We see that, even though price levels have changed drastically, the pattern itself stayed the same. This also hints at one of the main features of electricity demand: A very inflexible demand structure that most often cannot follow given price incentives. Nonetheless, and as stated above, we stick to spot price behavior in the years from 2015 to 2020.

Combining the information from Figures 4.6, 4.7 and 4.8, it seems reasonable to assume that spot prices exhibit an underlying deterministic mean, which could be represented by a seasonality function. Regarding negative prices, Table 4.10 presents their amounts as well as some statistics of interest. With a share of 1.66%, their relevance cannot be neglected and it is obvious that a spot price model must be able to simulate these prices.

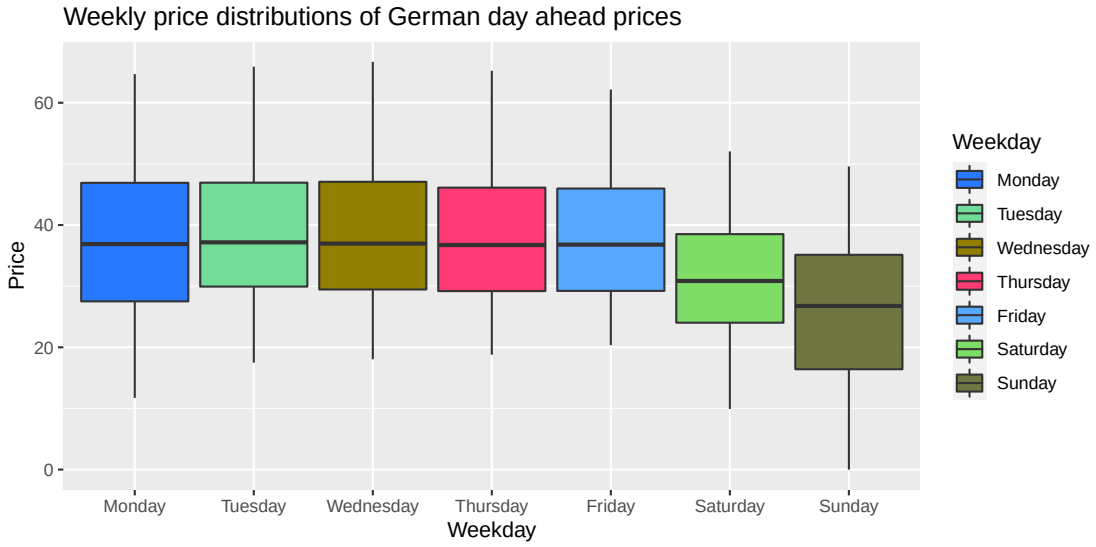

Figure 4.7: Box plot of spot prices grouped by weekdays from the time interval of 2015 to 2020.

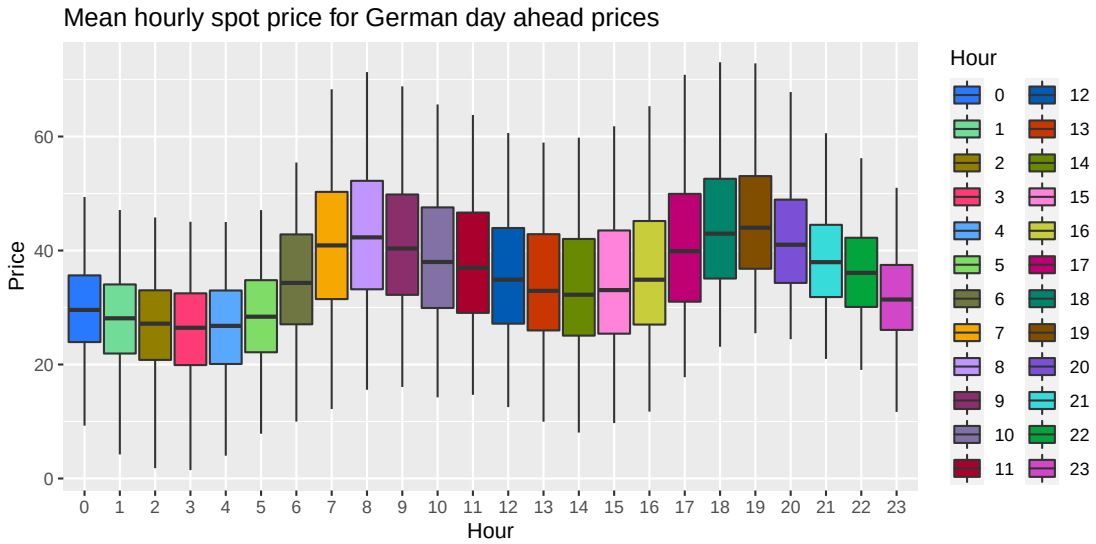

Figure 4.8: Box plot of hourly spot prices from the time interval of 2015 to 2020.

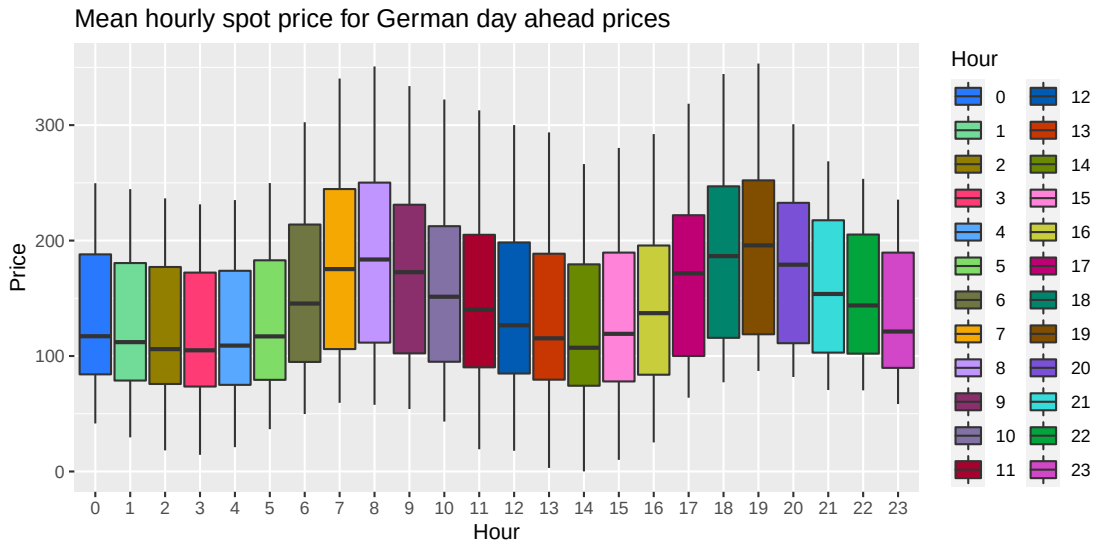

Figure 4.9: Mean hourly spot prices from the time interval of July 2021 to June 2022.

Concerning mean reversion, Figure 4.2 indicates that the prices never deviate long from their mean, but tend back to it. This is most obvious when concentrating on the jumps that lead away from the mean, as prices tend to return quickly instead of determining a new price regime. These jumps in both directions, positive as well as negative, are also an obvious feature that is still contained in the prices. It is visible in Figure 4.6 that during spring and summer, the jumps tend to be slightly smaller, which is due to more solar electricity in the grid, while in autumn and winter, they tend to be of bigger size.

Regarding stationarity, Figure 4.11 indicates that mean prices have moved over the years, even though no clear trend is visible. Consequently, we suggest that a factor covering long-term change should be contained in the model.

Finally, we ask whether the spot price distribution can be modelled via Gaussian distributions. Figure 4.12 contains the density of the spot prices in comparison to a Gaussian density with the same mean and standard deviation. Their difference is well visible mainly around the mean. Consequently, we conclude that this is also a valid feature for the considered time period and should be taken into account.

4.1.2 Factor model for the day ahead market

Following the analysis in Subsection 4.1.1, we deem it necessary to include the described characteristics in the model. This leads us to use arithmetic jump-diffusion models introduced in [3], which are able to capture all these elements in different factors. In [3],

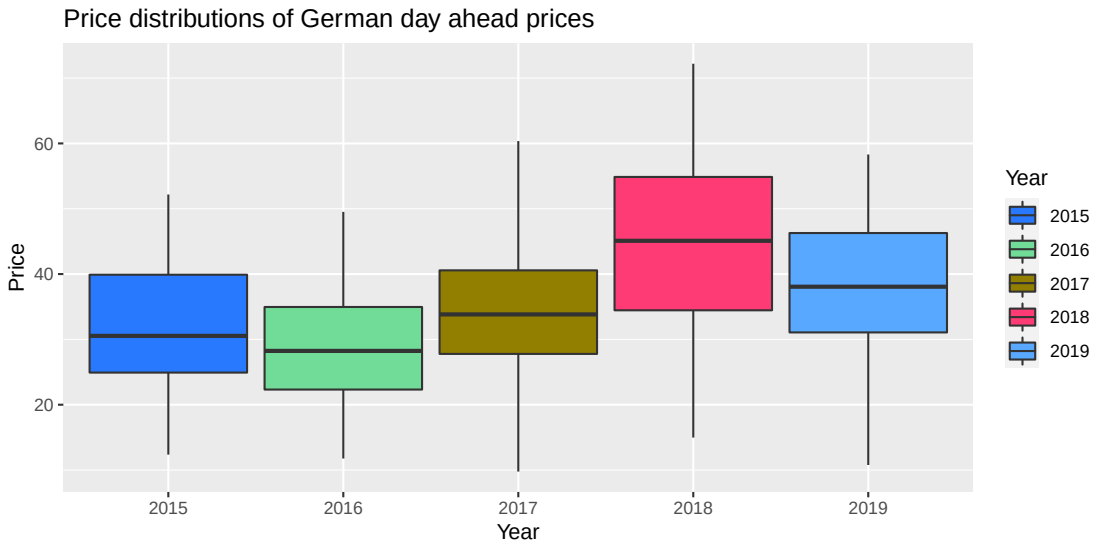

Figure 4.11: Mean yearly spot prices from the years 2015 to 2020.

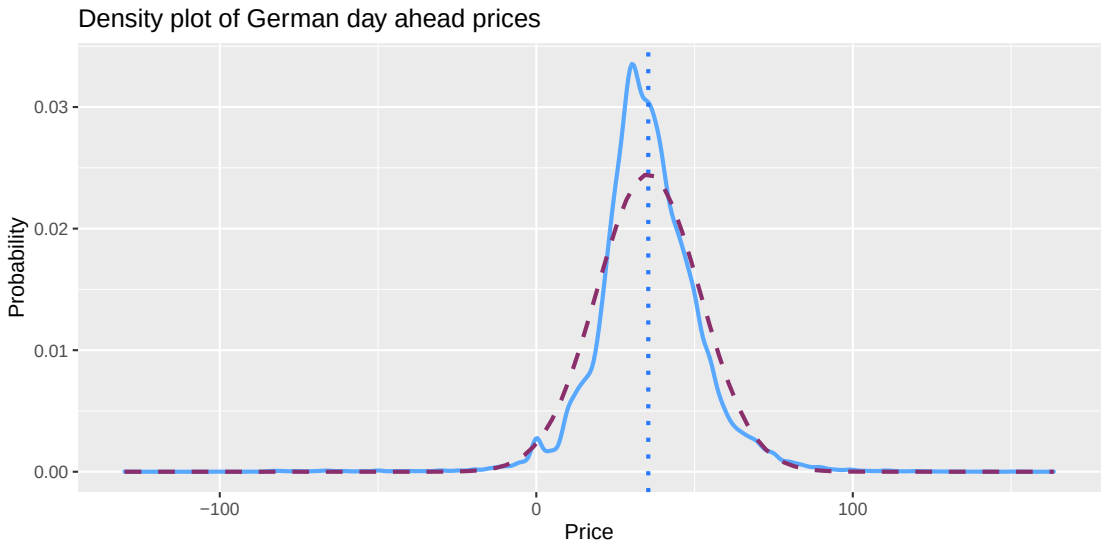

Figure 4.12: Density of German spot prices from the years 2015 to 2020. The price density is given in solid blue, the dotted line indicates the mean price, and dashed purple references a Gaussian distribution with the same mean and standard deviation as of the spot prices.

the general arithmetic factor spot price model is given by

$$S(t) = \Lambda(t) + \sum_{i=1}^{m} X_i(t) + \sum_{j=1}^{n} Y_j(t), \tag{4.1}$$

where $\Lambda(t)$ represents a deterministic seasonality function that is assumed to be continuously differentiable, and

$$dX_i(t) = (\mu_i(t) - \alpha_i(t)\, X_i(t))\, dt + \sum_{k=1}^{p} \sigma_{ik}(t)\, dB_k(t), \quad i = 1, ..., m \tag{4.2}$$

$$dY_j(t) = (\delta_j(t) - \beta_j(t)\, Y_j(t))\, dt + \eta_j(t) dI_j(t), \quad j = 1, ..., n \tag{4.3}$$

where $\mu_i(t), \alpha_i(t), \delta_j(t), \beta_j(t), \sigma_{ik}(t), \eta_j(t)$ are all continuous functions with $\alpha_i(t) > 0$ and $\beta_j(t) > 0$ for all t. Furthermore, $B(t) = \big(B_1(t), ..., B_p(t)\big)$ represents a p-dimensional Brownian motion, and $I_j(t)$ a general independent increments process. In this model, the factors X_i represent the diffusion terms that are identified with long-term and short-term developments of spot prices, whereas Y_j integrate jump components into the model.

To adapt this model setting to the findings in Subsection 4.1.1, we follow [30] and integrate a factor for short-term price movements, a factor to account for jumps, and finally add a geometric Brownian motion multiplied by the seasonality function to represent the long-term movements of the deterministic seasonality. Consequently, our model for the spot market price $S(t)$ has the following form:

$$S(t) = \Lambda(t)X(t) + Y(t) + Z(t), \tag{4.4}$$

where we define the three factors by the following stochastic differential equations:

(i) The first factor is defined as the unique solution to the stochastic differential equation

$$dX(t) = \sigma_1 X(t) dW_1(t), \quad X(0) = 1,$$

a geometric Brownian motion with zero drift. It models long term changes in the electricity prices. This differential equation has an explicit solution, namely

$$X(t) = \exp\,[-\frac{\sigma_1^2}{2}t + \sigma_1 W_1(t)]. \tag{4.5}$$

(ii) The second factor is denoted as

$$dY(t) = -\alpha_2 Y(t)dt + \sigma_2 dW_2(t), \quad Y(0) = 0,$$

an Ornstein-Uhlenbeck process with zero mean and the Brownian motion $W_2(t)$ as driver. This factor captures daily deviations from the mean and has a mean reverting property with $\alpha_2 > 0$. Again, this factor has an explicit solution, i.e.

$$Y(t) = Y(0)\, e^{-\alpha_2 t} + \int_0^t e^{-\alpha_2 (t-s)}\, \mathrm{d}W_2(t)(s), \tag{4.6}$$

where $W_2(t)$ again is a Brownian motion.

(iii) The third factor is defined as

$$dZ(t) = -\alpha_3 Z(t)dt + dN(t), \quad Z(0) = 0,$$

an Ornstein-Uhlenbeck process with the compound Poisson process $N(t)$ as its driver and again a mean reversion $\alpha_3 > 0$. The compound Poisson process is itself defined as

$$N(t) = \sum_{i=1}^{P(t)} D_i,$$

with $P(t)$ being a homogeneous Poisson process with mean arrival rate λ_3 and Gaussian i.i.d. jump sizes $D_i \sim \mathbb{N}(\mu_3, \sigma_3)$. $Z(t)$ represents price jumps through information that suddenly reach the market. Its explicit solution is given by

$$Z(t) = \sum_{i=1}^{P(t)} e^{-\alpha_3(t-\tau_i)} D_i, \tag{4.7}$$

where τ_i represents the arrival time of jump i.

The element of Equation (4.4) that is left to detail is $\Lambda(t)$. So far, the literature suggests the usage of sinusoidal functions or dummy variables in order to capture relevant deterministic features, see [29]. We combine both approaches in the following: We assume a seasonality of the form

$$\tilde{\Lambda}(t) = a + bt + c\sin\left(\frac{2\pi}{365}(t-d)\right), \tag{4.8}$$

i.e. we first follow the sinusoidal approach. In a next step, we want to incorporate the dependency on factors like weekdays or holidays. Therefore, a day-dependent constant $A(t)$ is added to the sinusoidal function. The days are separated into the type-days

- Mondays,
- Tuesdays, Wednesdays, Thursdays,
- Fridays,
- Saturdays, bridge days, non-national holidays,
- Sundays and national holidays.

The final seasonality model is then defined as follows:

$$\Lambda(t) = \tilde{\Lambda}(t) + A(t). \tag{4.9}$$

Of course, hourly price forward curves (HPFCs) corresponding to whatever time interval is chosen are also a valid option. With that, the model is completed. As the factor model defined in Equation (4.4) has already proven itself in the industry, we believe it is well suited for our context.

Remark 4.2. Note that the form of the jump process implicitly assumes a symmetric jump distribution. Furthermore, the jumps are not assumed to be time-dependent. That is a simplification, but facilitates the calibration.

Model calibration Obviously, calibration of the three factors including the seasonality function is essential before the model can be used. The model parameters that need calibration are the following, see [30] and [21]:

- The volatility of the geometric Brownian Motion in the first factor, σ_1,
- The mean reverting rate in the second factor, α_2,
- The volatility of the Brownian Motion in the second factor, σ_2,
- The mean reverting rate in the third factor, α_3,
- The jump arrival rate of the Poisson process in the third factor, λ_3,
- The mean jump size of the Poisson process in the third factor, μ_3,
- The volatility of the jump size of the Poisson process in the third factor, σ_3.

We follow [21] with the explanation of the calibration. The first parameter to be calibrated is σ_1. As the first factor in itself represents long-term movements in the prices, it is based on the futures prices available on the market. These prices have been shown to be approximately log-normally distributed. Therefore, σ_1 can be derived through a volatility estimation based on the log returns of the future prices.

After that, all other parameters are calibrated. As the second and third factor both are representing short-term developments in the prices, they are calibrated on the mean daily spot prices instead of the futures prices. The deterministic seasonality is the first to be calibrated, as it has to be removed from the price time series in order to calibrate the other parameters. Here, a least squares regression regarding $\tilde{\Lambda}(t)$ is the first step, followed by the calibration of the corresponding addendum $A(t)$ for every type-day with least-squares after deduction of $\tilde{\Lambda}(t)$ from the spot prices. This yields the required result, and the seasonality $\Lambda(t)$ is deducted from the time series of the mean daily spot prices. With this, we assume the deseasonalization to be complete and thus the time series to be free of deterministic patterns. Furthermore, we assume the effect of the first factor to be included in the deterministic seasonality and therefore assume that we can ignore it for the calibration of the second and third factor. Based on the deseasonalized prices, we can estimate the mean reverting rates α_2 and α_3 via an exponentially decreasing autocorrelation function. Then, the second and third factor need to be separated from each other. This happens through the following repeated procedure: The whole time series is taken into account and the value deviating the most from the mean is declared to be the first jump, and is removed afterwards. This is repeated until the jump size of the next jump to be removed is equal to or less than the volatility of the time series. This procedure results in the remaining time series, which is assumed to now contain only the values of the second factor, whereas the removed jumps form the time series for the third factor. Now, the second factor's values can be estimated with Equation (4.6) based on maximum likelihood estimation. Finally, we can estimate the parameters of the third

process through Equation (4.7), again with maximum likelihood.

Example paths for the final model are shown in Figure 4.13. As is visible, their shape strongly depends on the underlying deterministic seasonality. This deterministic seasonality as well as the parameter calibration are based on the year 2020 and on all futures prices available in that time period. The mean reverting property of the second factor is well visible, keeping the simulated paths close to its deterministic mean.

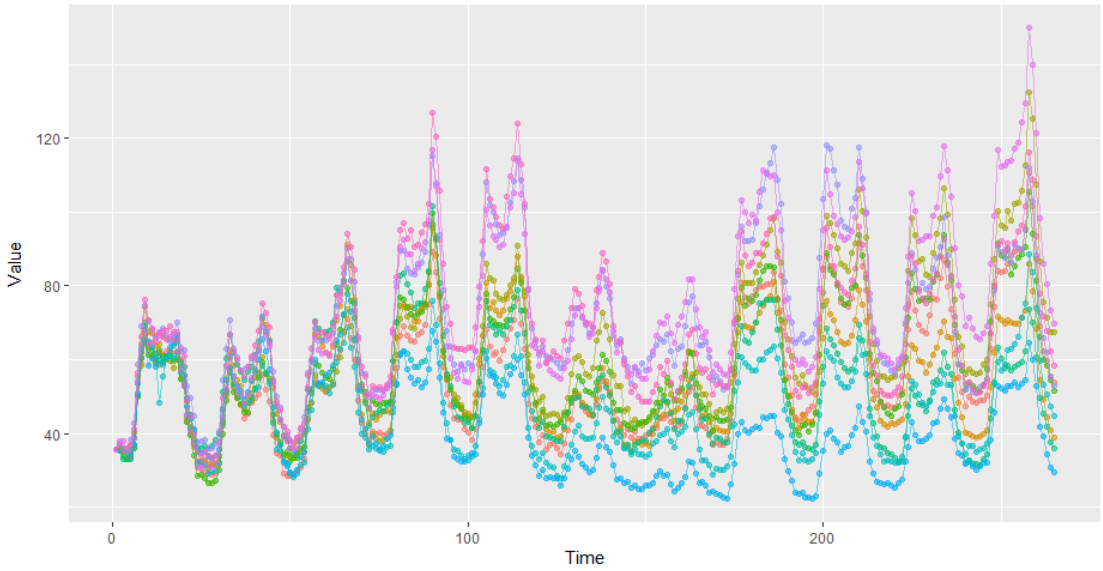

Figure 4.13: Example paths of the model in Equation (4.4), over a time horizon of two weeks.

As described above, models of this kind are not used to forecast exact prices of the next day, but are rather used to give an impression about how spot prices could develop over the following weeks or months and which risks or opportunities are faced in their development. Consequently, for prediction purposes, we assume that the prices of the next day are predicted by a well-performing short-term price model, and apply the factor model from Equation (3.6) for the days two to seven. However, for the procedure from this point onward, it is not necessary to use the specified model from Equation (3.6). All that is needed is a model from which sample paths can be created for the coming week.

4.2 Prediction of the day ahead market

Now that the model for the day ahead market has been identified and set up, we might assume that everything has exactly the form needed to optimize on this market. However, since jump-diffusion processes will have continuous parts due to the diffusion and discontinuous parts due to the jumps in their distribution, the image measure induced

from the model in Equation (3.6) necessarily has these properties as well. This leads to the problem that image measures without closed form that are not discrete measures cannot be optimized algorithmically, since computers need discrete data or closed form solutions to work with. To tackle the optimization problem, it is therefore necessary to reduce the complexity of the model by finding a finite approximation to it. This finite approximation is called a scenario model. Consequently, we follow [50] and define the original model as

$$\min\left\{F(x) = \mathcal{R}_P[Q(x,\xi)] : x \in \mathbb{X}, x \text{ adapted to } \mathcal{F}\right\} \tag{4.10}$$

for a real-valued objective function F defined by a cost functional $\mathcal{R}$, that has a cost function Q as its input. Q itself calculates costs by using as input a decision x from the set of feasible decisions $\mathbb{X}$ as well as a random process ξ with distribution P. To change the model to its corresponding scenario model, the distribution P is replaced by its finite approximation $\tilde{P}$, and the filtration $\mathcal{F}$ is replaced by a finite filtration $\tilde{\mathcal{F}}$, see [50]:

$$\min\left\{F(\tilde{x}) = \mathcal{R}_{\tilde{P}}[Q(\tilde{x},\xi)] : \tilde{x} \in \mathbb{X}, \tilde{x} \text{ adapted to } \tilde{\mathcal{F}}\right\}. \tag{4.11}$$

Since it is essential that the scenario model not only contains only a finite number of realizations of the stochastic process, but that it also approximates the original model reasonably well, a trade-off is made between computational complexity and model accuracy.

The approximation of an arbitrary probability measure P on $\mathbb{R}^m$ by a discrete probability $\tilde{P}$ can be accomplished by various methods. The following three different methods are most commonly used, see [50]:

- **Monte Carlo**: Here, randomized inputs for the model to be estimated are generated with the help of Pseudo random numbers, and on their basis the model is evaluated and mass points of the output distribution are found.
- **Quasi-Monte Carlo**: The modus operandi is similar to Monte Carlo, but the pseudo-random numbers are replaced by a low-discrepancy sequence. This can lead to an improved convergence speed compared to Monte Carlo.
- **Optimal quantization**: An optimal facility location problem is solved and in this way mass points of the output distribution are found.

According to [50], the methods are sorted both by increasing accuracy and by increasing computational complexity. Generating pseudorandom numbers is a relatively easy task, while solving optimal quantization is NP-hard. Nevertheless, the approximated discrete distributions generated by Monte Carlo contain an inherent randomness that is not present in an optimal quantization. Consequently, a choice between these methods is also a choice for a realization of the mentioned tradeoff between model accuracy and computational complexity.

As it is our goal to model exact and solve approximately, we will focus on the theory and application of optimal quantization. Therefore, we first mathematically introduce the concept of how the quality of a process approximation can be measured in the next subsections. Subsequently, the theory behind scenario trees as discrete approximations of stochastic processes is presented. We then discuss how to actually construct scenario trees based on real data or a probability model, and show how to reduce a large scenario tree to a manageable size. For all these tasks, we mainly follow [49] and [50], but also [67], [37] and [32]. Finally, a heuristic is presented that aims to reduce the computation time for large models.

4.2.1 Approximating probability distributions of random variables

In the context of stochastic optimization, as explained above, it is necessary to approximate the probability distribution of the original model from Equation (3.6), namely P, by another finite probability distribution $\tilde{P}$. We could now choose all of our favorite process realizations as the discretization, and use their probability distribution as the desired approximation. Most likely, this would not work very well. However, in order to be able to quantify whether an approximation $\tilde{P}$ is of high quality or not, we rely on a distance measure, which is defined in [50] as follows:

Definition 4.3. Let $\mathcal{P}$ be a set of probability measures on $\mathbb{R}^m$. Then, a *distance* d on $\mathcal{P} \times \mathcal{P}$ has to satisfy the following four conditions:

(i) **Nonnegativity**: For all $P_1, P_2 \in \mathcal{P}$ it holds true that $\mathrm{d}(P_1, P_2) \geqslant 0$.

(ii) **Symmetry**: For all $P_1, P_2 \in \mathcal{P}$ it holds true that $\mathrm{d}(P_1, P_2) = \mathrm{d}(P_2, P_1)$.

(iii) **Triangle inequality**: For all $P_1, P_2, P_3 \in \mathcal{P}$ it holds true that $\mathrm{d}(P_1, P_2) \leqslant \mathrm{d}(P_1, P_3) + \mathrm{d}(P_3, P_2)$.

(iv) **Strictness**: For all $P_1, P_2 \in \mathcal{P}$ it holds true that iff $\mathrm{d}(P_1, P_2) = 0$, then $P_1 = P_2$.

There exist several distances or semi-distances, where the latter do not fulfill the strictness condition in Definition 4.3, that could be considered in order to measure the distance between two probability distributions.

Remark 4.4. A distance concept that is often chosen in practice is the moment-matching semi-distance. It is defined over the set of probabilities $\mathcal{P}_q$ on $\mathbb{R}$ which possess the q-th moment as

$$\mathrm{d}_{M_q}(P_1, P_2) := \sup \Big\{ \mid \int \omega^s \, P_1(\mathrm{d}\omega) - \int \omega^s \, P_2(\mathrm{d}\omega) \mid : \; s \in \{1, ..., q\} \Big\}.$$

This semi-distance tests whether all moments up to the q-th moment are identical or not. The problem with moment-matching is that there exist many distributions that are very different from each other but still have the same moments, see e.g. [50], Example 2.2. This illustrates why moment matching yields only a semi-distance.

As a consequence of Remark 4.4, instead of matching only moments, we want to match the distribution as a whole. Furthermore, following [50], the distance should measure the closeness of distributions without regard to the underlying probability spaces, as well as provide a representation of the weak* topology. The latter is relevant because the weak* topology offers the possibility to approximate measures by discrete measures and to show convergence of sequences of measures to a target measure. These properties are satisfied by the Kantorovich distance, which is defined as follows:

Definition 4.5. The *Kantorovich distance* is defined over the class $\mathcal{H}$ of all Lipschitz functions:

$$\mathrm{d}_1(P_1, P_2) := \sup\Big\{ \int h \,\mathrm{d}P_1 - \int h \,\mathrm{d}P_2 \;:\; h(\omega) - h(\tilde{\omega}) \leqslant \|\omega - \tilde{\omega}\|,\, h \in \mathcal{H} \Big\}.$$

This distance is a metric for weak convergence on the sets of probability measures that uniformly have a first moment, see [50]. The Kantorovich distance is generalized by the Wasserstein distance, which in turn is generalized by [50] as follows:

Definition 4.6. The *Wasserstein distance* of order r of two Borel measures P on Ω and $\tilde{P}$ on $\tilde{\Omega}$ is given by

$$\mathfrak{d}_r(P, \tilde{P}) := \Big(\inf_{\pi} \int\int \mathrm{d}(\omega, \tilde{\omega})^r \,\pi(\mathrm{d}\,\omega, \mathrm{d}\,\tilde{\omega}) \Big)^{1/r} \tag{4.12}$$

for a distance d, where the infimum is taken over all joint probability measures π on the product space $\Omega \times \tilde{\Omega}$, i.e.

$$\pi(A \times \tilde{\Omega}) = P(A),$$
$$\pi(\Omega \times B) = \tilde{P}(B).$$

The optimal measure π is often called *optimal transport plan*.

Thus, if one interprets the distance as a kind of cost function, the Wasserstein distance represents the minimum cost of transferring one distribution to another, while keeping the marginal distributions of the joint distribution equal to the two original probability distributions. Note that computing the Wasserstein distance between two distributions is an optimization problem in itself and depends on the choice of the distance d.

Remark 4.7. The infimum in Equation (4.12) is attained if both measures P and $\tilde{P}$ are tight, see Remark 2.6. in [50]. Furthermore, Equation (4.12) is feasible and well defined whenever $P \in \mathcal{P}_r(\Omega; \mathrm{d})$ and $\tilde{P} \in \mathcal{P}_r(\tilde{\Omega}; \mathrm{d})$, with $\mathcal{P}_r(\Omega; \mathrm{d})$ containing all probability measures P which satisfy

$$\int_{\Omega} \mathrm{d}(\omega, \omega_0)^r \, P(\mathrm{d}\omega) < \infty$$

for any $\omega_0 \in \Omega$.

Remark 4.8. The definition of Wasserstein distance mostly used in the literature is one where the distance function is not defined over two different spaces Ω and $\tilde{\Omega}$, but as $\mathfrak{d}_r(\cdot, \cdot) : \Omega \times \Omega \to \mathbb{R}$, i.e. only one underlying space is considered.

The following lemma collects some properties of the Wasserstein distance:

Lemma 4.9. *For the Wasserstein distance, the following properties hold true:*

(i) ***Monotonicity****: If $r_1 \leqslant r_2$, then we find that $\mathfrak{d}_{r_1}(P, \tilde{P}) \leqslant \mathfrak{d}_{r_2}(P, \tilde{P})$.*

(ii) ***Triangle equality****: $\mathfrak{d}_r(P, \tilde{P}) \leqslant \mathfrak{d}_r(P, \tilde{\tilde{P}}) + \mathfrak{d}_r(\tilde{\tilde{P}}, \tilde{P})$, i.e. the Wasserstein distance is a distance.*

(iii) ***Convexity****: For $0 \leqslant \lambda \leqslant 1$, it holds that*

$$\mathfrak{d}_r\big(P, (1-\lambda)P_0 + \lambda P_1\big)^r \leqslant (1-\lambda)\mathfrak{d}_r(P, P_0)^r + \lambda \mathfrak{d}_r(P, P_1)^r$$

and

$$\mathfrak{d}_r\big(P, (1-\lambda)P_0 + \lambda P_1\big) \leqslant \max\{\lambda, 1-\lambda\}^{1/r-1}\big((1-\lambda)\mathfrak{d}_r(P, P_0) + \lambda \mathfrak{d}_r(P, P_1)\big);$$

consequently, the Wasserstein distance is r-convex in any of its components.

Proof. See proof of Lemma 2.10 in [50]. □

Now, the choice of distance function is left to be discussed. A probability space $(\Omega, \mathcal{F}, P)$ is not necessarily endowed with a distance. However, it is always possible to inherit a distance concept from a random variable mapping from that probability space to $\mathbb{R}^m$ by looking at

$$\mathrm{d}(\omega_1, \omega_2) := \|\xi(\omega_1) - \xi(\omega_2)\|$$

with $\|\cdot\|$ a norm on $\mathbb{R}^m$. As it is our goal to metricize the distance between two different probability spaces, it is necessary to extend the distance function from one to two different underlying spaces. Here, the inherited distance comes into play.

Definition 4.10. Let $\xi : \Omega \to \mathbb{R}^m$ and $\tilde{\xi} : \tilde{\Omega} \to \mathbb{R}^m$ represent two random variables on distinct spaces. Then, the *inherited distance* between elements of Ω and $\tilde{\Omega}$ is defined by

$$\mathrm{d}(\omega, \tilde{\omega}) := \mathrm{d}\big(\xi(\omega), \tilde{\xi}(\tilde{\omega})\big) \tag{4.13}$$

for a distance d in $\mathbb{R}^m$. Usually, d is defined as a norm on $\mathbb{R}^m$.

Consequently, the Wasserstein distance uses the distance inherited from the random variables in play to measure the distance between the underlying probability spaces. The norms we consider in this thesis for the distance are the l_1-norm and the l_2-norm, with

$$\mathrm{d}^p(w, v) := \left(\sum_{i=1}^{m} |w_i - v_i|^p\right)^{1/p}, \tag{4.14}$$

where $p = 1, 2$ correspondingly. Other norms on $\mathbb{R}^m$ would of course also be possible.

Remark 4.11. If the chosen distance on the image space is the inherited distance from ξ, and it is based on a norm in $\mathbb{R}^m$, then it holds true that $P \in \mathcal{P}_r(\Omega; \mathrm{d}) \iff \xi$ has finite r-th moment [50]. In combination with Remark 4.7, this means that if ξ has finite r-th moment, then the problem in Equation (4.12) is well defined and feasible.

Endowed with the inherited distance which again is based on a norm $\|\cdot\|$, it holds true that

$$\|\mathbb{E}_P(\xi) - \mathbb{E}_{\tilde{P}}(\tilde{\xi})\| \leqslant \mathfrak{d}_r(P, \tilde{P}) \tag{4.15}$$

for $r \geqslant 1$, see [50], Lemma 2.13. One interpretation is that the transport distance that a particle has to move on average from one distribution to match the other is at least the distance of the two expected values from each other.

In the following, we present two examples how to calculate the Wasserstein distance in specific situations that are also found in [50].

Example 4.12. Let ξ and $\tilde{\xi}$ be random variables, taking values in $\mathbb{R}^N$ and $\mathbb{R}^M$, respectively, with discrete measures $P = \sum_{i=1}^N p_i \delta_{\xi_i}$ and $\tilde{P} = \sum_{j=1}^M \tilde{p}_j \delta_{\tilde{\xi}_j}$. Then, the computation of the Wasserstein distance of order r is given through the optimal solution of the linear program

$$\begin{aligned} &\min_{\pi} \sum_{i,j} \pi_{i,j} \cdot d_{i,j}^r \\ &\text{s.th.} \sum_{i=1}^N \pi_{i,j} = \tilde{p}_j && \text{for } i = 1, ..., N, \\ &\quad \sum_{j=1}^M \pi_{i,j} = p_i && \text{for } j = 1, ..., M, \\ &\quad \pi_{i,j} \geqslant 0. \end{aligned} \tag{4.16}$$

Here, $d_{i,j} = \mathrm{d}(\xi_i - \tilde{\xi}_j)$ is defined as the distance between ξ_i and $\tilde{\xi}_j$ in the chosen norm.

Example 4.13. Given two real valued random variables $\xi \sim \mathcal{N}(\mu, \sigma^2) =: P$ and $\tilde{\xi} \sim \mathcal{N}(\tilde{\mu}, \tilde{\sigma}^2) =: \tilde{P}$, we calculate the Wasserstein distance of order 2 between their distributions. For measures on the real line, it holds true that

$$\mathfrak{d}_r(P, \tilde{P})^r = \int_0^1 |G_P^{-1}(\alpha) - G_{\tilde{P}}^{-1}(\alpha)|^r \, \mathrm{d}\alpha, \tag{4.17}$$

where $G_P(y) = P((-\infty, y])$ is the cumulative distribution function of P and G_p^{-1} is its quantile function, see [50], Thm. 2.15. Thus, we apply Equation (4.17) to the normal

distribution and find

$$\begin{aligned}\mathfrak{d}_2(P,\tilde{P})^2 &= \int_0^1 \left(\mu-\tilde{\mu}+(\sigma-\tilde{\sigma})\Phi^{-1}(z)\right)^2 \mathrm{d}z \\ &= \int_0^1 \left((\mu-\tilde{\mu})^2+2(\mu-\tilde{\mu})(\sigma-\tilde{\sigma})\Phi^{-1}(z)+(\sigma-\tilde{\sigma})^2(\Phi^{-1}(z))^2\right)\mathrm{d}z \\ &= (\mu-\tilde{\mu})^2+2(\mu-\tilde{\mu})(\sigma-\tilde{\sigma})\int_0^1 \Phi^{-1}(z)\,\mathrm{d}z+(\sigma-\tilde{\sigma})^2\int_0^1(\Phi^{-1}(z))^2\,\mathrm{d}z \\ &= (\mu-\tilde{\mu})^2+(\sigma-\tilde{\sigma})^2. \end{aligned} \tag{4.18}$$

Here, Φ corresponds to the cumulative distribution function of the standard Gaussian distribution. The first equation holds due to $\xi \sim \sigma Z+\mu$ with $Z \sim \mathcal{N}(0,1)$. The forth equation holds due to $\int_0^1 \Phi^{-1}(z)\,\mathrm{d}z = 0$ and $\int_0^1(\Phi^{-1}(z))^2\,\mathrm{d}z = 1$, both reached by the substitution $z=\Phi(x)$.

Now, one might wonder whether properties like mean, variance or covariances of two probability distributions are close when the Wasserstein distance between them is small. The following proposition and remark yield answers to these questions.

Proposition 4.14. *Let $\xi \sim P$ and $\tilde{\xi} \sim \tilde{P}$. Then the following hold true:*

(i) $|\mathbb{E}(\xi)-\mathbb{E}(\tilde{\xi})| \leqslant \mathfrak{d}_1(P,\tilde{P})$.

(ii) $|\mathbb{E}(|\xi|)-\mathbb{E}(|\tilde{\xi}|)| \leqslant \mathfrak{d}_1(P,\tilde{P})$.

(iii) $|\mathbb{E}(\xi-a)_+-\mathbb{E}(\tilde{\xi}-a)_+| \leqslant \mathfrak{d}_1(P,\tilde{P})$.

(iv) $|\mathbb{E}(\xi^q)-\mathbb{E}(\tilde{\xi}^q)| \leqslant q\cdot\mathfrak{d}_1(P,\tilde{P})\cdot\left(\left(\mathbb{E}\left[|\xi|^{r\frac{q-1}{r-1}}\right]\right)^{\frac{r-1}{r}}+\left(\mathbb{E}\left[|\tilde{\xi}|^{r\frac{q-1}{r-1}}\right]\right)^{\frac{r-1}{r}}\right)$ *for* $q\in\mathbb{N}_+$.

(v) $|\mathbb{E}(|\xi|^q)-\mathbb{E}(|\tilde{\xi}|^q)| \leqslant q\cdot\mathfrak{d}_1(P,\tilde{P})\cdot\max\left\{\left(\mathbb{E}\left[|\xi|^{r\frac{q-1}{r-1}}\right]\right)^{\frac{r-1}{r}},\left(\mathbb{E}\left[|\tilde{\xi}|^{r\frac{q-1}{r-1}}\right]\right)^{\frac{r-1}{r}}\right\}$.

Proof. The proof is found in [50], see Props. 2.20 and 2.21. □

Consequently, the distances between all moments are bounded from above by the Wasserstein distance, and thus a small distance implies a small difference of the moments. However, since the Wasserstein distance is usually not equal to 0 when a continuous distribution is approximated by a discrete one, equality of moments is usually not given.

Finally, we take a look at whether a diminishing Wasserstein distance of a sequence of probability measures to a target measure equals a convergence of the probability sequence to the target distribution. This is answered by the uniform tightness condition [50]:

Theorem 4.15. *Let $(P_n)_{n\geqslant 1}$ be a sequence of measures in $\mathcal{P}_r(\Xi)$, where $\mathcal{P}_r(\Xi)$ is defined as in Remark 4.7 above. Furthermore, let P also be a measure contained in $\mathcal{P}_r(\Xi)$. Then, the following two statements are equivalent:*

(i) $\mathfrak{d}_r(P_n, P) \overset{n\to\infty}{\longrightarrow} 0$.

(ii) $P_n \overset{n\to\infty}{\longrightarrow} P$ *in a weak* sense, and P_n satisfies the following uniform tightness condition: For any $\xi_0 \in \Omega$, it holds true that*

$$\limsup_{n\to\infty} \int_{\{d(\xi_0,\xi)\geqslant R\}} d(\xi_0,\xi)^r \, P_n(\mathrm{d}\xi) \overset{R\to\infty}{\longrightarrow} 0.$$

Proof. The proof can be found in [62], Thm. 7.12. □

This in combination with the separability of $\left(\mathcal{P}_r(\Xi), \mathfrak{d}_r\right)$ and it being a Polish space if (Ξ, d) is a Polish space, see Thm. 2.25 and Thm. 2.26 in [50], yields the desired outcome.

Remark 4.16. In [52], a relaxation of the Wasserstein distance called nested Sinkhorn divergence is introduced. It regularizes the Wasserstein distance and thus gives a much faster computation time. The reduction in computational time is paid for by a worse approximation of the true Wasserstein distance.

4.2.2 Approximating probability distributions of stochastic processes

Now that we have found a distance that works well for evaluating approximations of probability distributions of random variables, the remaining step is to transfer it to the case of stochastic processes. We will see that the direct reuse of the Wasserstein distance in its multivariate form does not capture all the information conveyed by stochastic processes. Consequently, an extension to the framework of multi-period models must be applied. This extension is called *nested distance* or *process distance* and is introduced in [49]. It measures the distance between two different stochastic processes, which is needed to evaluate whether an approximation of a continuous stochastic process by a discrete approximation is a good fit. The nested distance adapts to multi-period models by taking into account all filtrations generated by the stochastic process ξ. To define it properly, we first introduce stochastic processes and translate the induced distance to a multi-period setting. Furthermore, we show why a direct application of the Wasserstein distance does not correctly measure distances between stochastic processes. This subsection is again guided by [50].

In this subsection, we consider a measurable stochastic process $\xi = (\xi_t(\omega))_{t\in\mathbb{R}_+}$, that is defined on $(\Omega, \mathfrak{F} = (\mathcal{F}_0, \mathcal{F}_1, ..., \mathcal{F}_T), \mathbb{P})$, which is a filtered probability space, with $\xi_t : \Omega \to (\Xi_t, \mathrm{d}_t)$. We assume that (Ξ_t, d_t) is a Polish space. The index t of the stochastic process is defined on the positive real numbers, but evaluated in discrete time over $T+1$ different stages, $\xi = (\xi_0, ..., \xi_T)$. In this setting, ξ_0 is considered to be known and thus

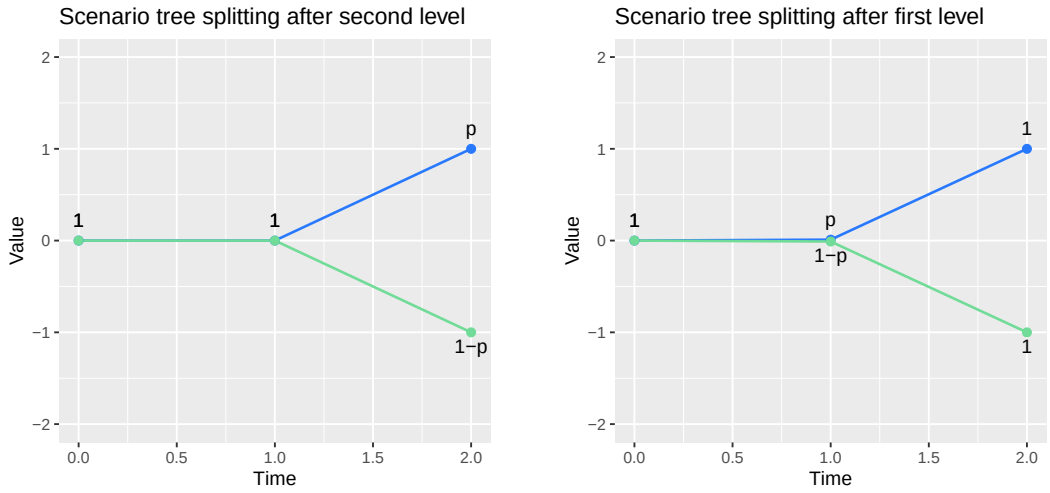

Figure 4.14: Two stochastic processes with identical states and identical final probabilities. The conditional probability to reach the specified value from its predecessor is written above each node.

deterministic. Furthermore, the various spaces $(\Xi_t, \mathfrak{d}_t)$ can differ from each other. With $T < \infty$, we can define the random variable

$$\xi : \Omega \to \Xi_0 \times \Xi_1 \times ...\Xi_T, \qquad \omega \mapsto (\xi_0(\omega), \xi_1(\omega), ..., \xi_T(\omega)). \tag{4.19}$$

Consequently, instead of looking at each time step of the stochastic process as a random variable in itself, we concentrate on the realization of one ω and map it to its path in the space $\Xi := \Xi_0 \times \Xi_1 \times ...\Xi_T$. Based on this definition, the law of the process can be defined.

Definition 4.17. Let ξ be defined as in Equation (4.19). Then, we define the *law of the process* ξ as $P^\xi = P \circ \xi^{-1}$. It defines a probability measure on $\Xi := \Xi_0 \times \Xi_1 \times ...\Xi_T$.

As (Ξ, d) was assumed to be Polish, the Wasserstein distance on $\mathcal{P}_r(\Xi, \mathrm{d})$ is well defined. Here, d could be the l_1 or the l_2 norm. Nonetheless, the following example shows why looking at

$$\mathfrak{d}_r(P^\xi, P^{\tilde{\xi}}), \tag{4.20}$$

which we call *multivariate distance* from now on, falls short in capturing all information provided by the processes ξ and $\tilde{\xi}$. Compare the graphs in Figure 4.14. Both processes have the same states, and both processes have the same final probabilities. The Wasserstein distance of both processes is computed as follows. We have $\Xi = \tilde{\Xi} = \{(0,0,1),(0,0,-1)\}$, and the distance matrix for $r = 1$ is given by $\mathrm{d} = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$. Furthermore, the transport map $\pi = \begin{pmatrix} p & 0 \\ 0 & 1-p \end{pmatrix}$ is feasible and optimal. The combination

of the two gives $\mathfrak{d}_r(P, \tilde{P}) = \sum_{i,j} \mathrm{d}_{i,j}\pi_{i,j} = 0$. Thus, the Wasserstein distance does not distinguish between the two processes. Nevertheless, they are clearly not the same: For the first process at time $t = 1$, we do not know which of the final results will occur. In contrast, for the second process, we have this exact knowledge when we are at time $t = 1$. The Wasserstein distance does not consider conditional probabilities, but only final probabilities, and thus loses the information contained in the conditional probabilities. Therefore, if we want to use this distance concept for our application with a seven-stage multi-period model, it must be adapted. Filtrations and conditional probabilities induced by the processes must be taken into account. For this purpose, the history process and the natural filtration are introduced:

Definition 4.18. The *history process* of a stochastic process ξ is defined by

$$\xi_{0:t} := (\xi_0, ..., \xi_t). \tag{4.21}$$

This naturally induces the *natural filtration* of ξ by

$$\mathfrak{F}^\xi = (\mathcal{F}_t^\xi)_{t=0}^T, \qquad \mathcal{F}_t^\xi := \sigma(\{\xi_{0:t}^{-1}(A_0 \times ... \times A_t) \,:\, A_s \in \mathcal{B}(\Xi_s)\}). \tag{4.22}$$

Here, $\mathcal{B}(\Xi_s)$ represents the Borel sets on the space Ξ_s.

The natural filtration yields the possibility to introduce the concept of information gain over time through the stochastic processes. This idea is addressed by the nested distance, which is defined as follows for two stochastic processes [50]:

Definition 4.19. The *nested distance* of order $r \geqslant 1$ of two filtered probability spaces $\mathbb{P} = (\Omega, (\mathcal{F}_t), P)$ and $\tilde{\mathbb{P}} = (\tilde{\Omega}, (\tilde{\mathcal{F}}_t), \tilde{P})$, for which a distance $\mathrm{d} : \Omega \times \tilde{\Omega} \mapsto \mathbb{R}^m$ exists, is defined as the optimal value of the optimization problem

$$\begin{aligned} \mathrm{D}_r(\mathbb{P}, \tilde{\mathbb{P}}) := \inf_{\pi} \quad & \Big(\int \mathrm{d}(\omega, \tilde{\omega})^r \, \pi(\mathrm{d}\,\omega, \mathrm{d}\,\tilde{\omega}) \Big)^{1/r} \\ \text{s.t.} \quad & \pi(A \times \tilde{\Omega} \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t) = P(A \mid \mathcal{F}_t) \;\text{ for } A \in \mathcal{F}_t, \\ & \pi(\Omega \times B \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t) = \tilde{P}(B \mid \tilde{\mathcal{F}}_t) \;\text{ for } B \in \tilde{\mathcal{F}}_t \end{aligned} \tag{4.23}$$

for all stages $t \in \{1, ..., T\}$. The infimum here is taken over all bivariate probability measures $\pi \in \mathcal{P}(\Omega \times \tilde{\Omega})$.

The nested distance takes into account information gains by conditioning on the corresponding sigma algebra. Of course, just as in the case of random variables, it is possible to apply the nested distance to the filtered probability spaces of two stochastic processes. Thus, we are able to use it when comparing an approximation with its approximated process. We will refer to filtered probability spaces over stochastic processes $(\Omega, \mathfrak{F}, P, \xi)$ as *nested distributions* in accordance with the term "nested distance" [50]. The well-definedness of the nested distance is given by the following theorem:

Theorem 4.20. *Let $(\Omega, \mathfrak{F}, P, \xi) \sim \mathbb{P}$ and $(\tilde{\Omega}, \tilde{\mathfrak{F}}, \tilde{P}, \tilde{\xi}) \sim \tilde{\mathbb{P}}$ be nested distributions. Furthermore, let $\mathcal{F}_0 = \{\varnothing, \Omega\}$ and $\tilde{\mathcal{F}}_0 = \{\varnothing, \tilde{\Omega}\}$. Then, the product measure $\pi := P \otimes \tilde{P}$ is feasible and the nested distance is well defined. Furthermore, it holds true that*

$$\mathfrak{d}_r(P, \tilde{P})^r \leqslant D_r(\mathbb{P}, \tilde{\mathbb{P}})^r \leqslant \mathbb{E}_{P \otimes \tilde{P}}(d^r). \tag{4.24}$$

Proof. See the proof of Lemma 2.37 in [50]. □

Furthermore, the properties of the Wasserstein distance regarding monotonicity, convexity and the triangular equation carry over to the nested distance:

Lemma 4.21. *For the nested distance, the following properties hold true:*

(i) ***Monotonicity****: If $r_1 \leqslant r_2$, we find that $D_{r_1}(\mathbb{P}, \tilde{\mathbb{P}}) \leqslant D_{r_2}(\mathbb{P}, \tilde{\mathbb{P}})$.*

(ii) ***Triangle equality****: $D_r(\mathbb{P}, \tilde{\mathbb{P}}) \leqslant D_r(\mathbb{P}, \tilde{\tilde{\mathbb{P}}}) + D_r(\tilde{\tilde{\mathbb{P}}}, \tilde{\mathbb{P}})$, i.e. the nested distance is a distance.*

(iii) ***Convexity****: For $0 \leqslant \lambda \leqslant 1$, it holds that*

$$D_r\big(\mathbb{P}, \mathcal{C}(\mathbb{P}_0, \mathbb{P}_1, \lambda)\big)^r \leqslant \lambda D_r(\mathbb{P}, \mathbb{P}_0)^r + (1-\lambda) D_r(\mathbb{P}, \mathbb{P}_1)^r$$

and

$$D_r\big(\mathbb{P}, \mathcal{C}(\mathbb{P}_0, \mathbb{P}_1, \lambda)\big) \leqslant \max\{\lambda, 1-\lambda\}^{1/r-1}\big(\lambda D_r(\mathbb{P}, \mathbb{P}_0) + (1-\lambda) D_r(\mathbb{P}, \mathbb{P}_1)\big);$$

consequently, the nested distance is also r-convex in any of its components. Here, $\mathcal{C}(\mathbb{P}_0, \mathbb{P}_1, \lambda)$ is the compound of the two nested distributions $\mathbb{P}_0$ and $\mathbb{P}_1$ and is defined by

$$\mathcal{C}(\mathbb{P}_0, \mathbb{P}_1, \lambda) := \begin{cases} \mathbb{P}_0, & \text{with probability } \lambda, \\ \mathbb{P}_1, & \text{with probability } 1-\lambda. \end{cases}$$

The compound of two nested distributions is again a nested distribution.

Armed with this distance, we have a tool to measure the quality of an approximation. However, in order to determine the nested distance between two processes algorithmically, probability measures with finite support are needed. Therefore, the next subsection deals with how stochastic processes can be represented as discrete scenario trees.

4.2.3 Scenario trees - Theory

Stochastic processes with a discrete state space can be represented by a scenario tree or a scenario lattice. The main concepts on which scenario trees are based are stochastic process theory and graph theory, since trees can be represented as graphs. As stochastic processes have been treated already, we concentrate on introducing the basics of graphs shortly.

A scenario tree is a defined as a polytree, i.e. a circle-free, directed graph $G(N, E)$ with a single root, and all leaves ending at the same tree height. It contains $M+1$ nodes, numbering starting from 0 for the root node to M for the last leaf node. The set of all nodes is defined as $N := \{n_0, ..., n_M\}$, and the set of $\tilde{M}$ edges between the nodes is defined as $E = (e_1, ..., e_{\tilde{M}})$. Here, every edge $e_i := (n_k, n_m)$ determines a route between two nodes $n_k, n_m \in N$, with $t(e_i) = n_k$ forming the tail of edge e_i and $h(e_i) = n_m$

forming its head. Furthermore, a relation between different nodes is established through the introduction of parents, children and predecessors: For a node $n \in N$, we define as $E_+(n) := \{e \in E \mid t(e) = n\}$ the set of outgoing and by $E_-(n) := \{e \in E \mid h(e) = n\}$ the set of incoming edges to node n. Based on this relation, the set of children of a node n is defined by $N_+(n) := \{v \in N \mid v = t(e), e \in E_-(n)\}$, and the set of parents is consequently defined by $N_-(n) := \{v \in N \mid v = h(e), e \in E_-(n)\}$. Moreover, the set of predecessors of node n contains all nodes from the root of the tree to n that are connected through edges and is denoted by $\overline{N_-(n)}$. Based on these definitions, we can allocate each node n to a stage t, $t = 0, ..., T$, and a set $n \in N_t$ that collects all nodes with exactly t predecessors. Finally, we have a total of $T+1$ stages, with $T+1$ also being defined as height of the tree.

Definition 4.22. A *scenario tree* $\mathbb{T}$ is given by the following three concepts:

(i) The *topology* of the tree is determined by its list of predecessors,

$$N_-(n_0), ..., N_-(n_M).$$

$N_-(n_i)$ denotes the predecessor of node n_i, and we set $N_-(n_0) := \varnothing$. The collection of all pairs $\big(N_-(n_i), n_i\big)$ consequently contains all information about the edges in the tree and because of its circle-free property about the whole tree topology.

(ii) The *probability structure* of the tree is given by the list of probabilities

$$p_0, ..., p_M,$$

where p_i denotes the conditional probability to reach node n_i from its predecessor in $N_-(n_i)$. We define $p_0 := 1$ for the root node.

(iii) The *values* of the tree process at the nodes is given by the list of states

$$\bar{\xi}_0, ..., \bar{\xi}_M.$$

Their values correspond to the outcome of the process ξ at the respective nodes $n_0, ..., n_M$. Thus, they may be vectors of any dimension, matching the original process.

Furthermore, for every node $n \in N \backslash N_0$ we have that $|N_-(n)| = 1$, i.e. its parent is uniquely determined. As a second consequence, we obtain $\tilde{M} = M$.

A sample scenario is specified through

$$\big(n_0 \in N_0, n_1 \in N_1 \cap N_+(n_0), ..., n_T \in N_T \cap N_+(n_{T-1})\big),$$

connecting root and leaf n_T by its path through the tree. Here, the numbering of the nodes is not in correspondence with the general numbering of the nodes in the tree but is used to count through the tree stages. The scenario's probability is given by $p(n_0, ..., n_T) := p_{n_0} \cdot ... \cdot p_{n_T}$. Consequently, a scenario tree where $|N_-(n)| = 1$ for all

$n \in N \backslash N_0$, i.e. where all nodes have one unique predecessor except for the root (which has no parents), contains as many scenarios as it has leaves. This is the form of scenario tree that we consider in this thesis.

A second possible setup is that $|N_-(n)| > 1$ is true, which indicates that a node can have more than one parent, and hence the parent is not uniquely determined. This leads to the definition of scenario lattices.

Definition 4.23. Let $\mathbb{T}$ be a tree as given in Definition 4.22, with the difference that $N_-(n_t) = N_{t-1}$ for all nodes n_t on level t and all $t = 1, ..., T$. Consequently, all nodes on level t have all nodes on level $t-1$ as parents. This construction is denoted as *scenario lattice*.

Scenario lattices are the natural way to discretize Markov processes, since their lack of memory removes the dependence on the preceding path except for the current position, which is mirrored by this structure. If neither the posed optimization model nor the depicted process is path dependent, integrating this knowledge into the scenario tree implementation by choosing a scenario lattice over a scenario tree leads to computational advantages. Given the same number of nodes, a scenario lattice contains a much larger number of scenarios than a scenario tree, since its number of scenarios is equal to the number of the product of all stage nodes $\prod_{t=1}^{T} |N_t|$.

Regardless of whether a scenario tree or a scenario lattice is considered appropriate to approximate the original process ξ, one goal in choosing an approximation is of course to minimize the difference between the distribution of the original process and the distribution of the discretized scenario process. Consequently, the nested distance introduced in the previous subsections comes into play. Its construction principle is backwards, i.e. the computation of the nested distance between two trees of the same height starts with the leaves and ends with the roots. The nested distance is then given by the distance at the roots of both trees. For a single period observation, Wasserstein and nested distance are necessarily equal.

Equipped with the notions of scenario trees, lattices and nested distance, we present algorithms for constructing scenario trees that fit the original stochastic process well but are also computationally feasible.

4.2.4 Scenario trees - Construction

This subsection serves as a non-exhaustive introduction to the construction of scenario trees for an underlying stochastic process and is again based on [50]. Obviously, the best way to construct a tree would be to minimize $\mathrm{D}_r(\mathbb{P}, \tilde{\mathbb{P}})$ over all trees with e.g. a given number of nodes or leaves, where $\mathbb{P}$ represents the nested distribution of the original stochastic process and $\tilde{\mathbb{P}}$ denotes its approximation. Unfortunately, such a computation is not feasible on today's computers [50]. Therefore, the general approach to generating a scenario tree in multi-stage stochastic programming follows these three steps:

1) Generate a scenario tree with a large number of scenarios that captures the distribution of the underlying stochastic process very well.

2) Reduce the generated scenario tree to a tree with fewer scenarios and a different tree structure that still represents the large tree well and is nonetheless small enough to be processed by optimization algorithms.

3) Adjust the probabilities and values in the reduced tree so that the resulting tree is an optimal approximation to the original large tree, given its structure.

An underlying assumption for this procedure is that the large tree from the first step approximates the process distribution well enough to represent it from here on. Goodness-of-approximation of the tree from the third step is calculated in comparison to this big tree. For each of the scenario tree generation steps, we introduce algorithms to handle the described tasks.

The data used to construct a scenario tree usually stems either from a model (where an infinite number of trajectories can be generated) or from real-world applications, such as real day ahead prices, which can be collected over comparable time intervals and thus form the required trajectories. In both cases, trajectories can be denoted by

$$\nu_j, \quad j = 1, ..., \mathcal{M}$$

with $\mathcal{M} \in (\mathbb{N} \cup \infty)$. Each trajectory contains the values of the process over time, i.e. $\nu_j := (\nu_{j,0}, ...\nu_{j,T})$, and $\nu_{j,i} \in \mathbb{R}^m \, \forall \, j = 1, ..., \mathcal{M}$, $i = 0, ..., T$. Equipped with these trajectories, we now proceed to the actual construction.

Step 1 - Generation To build a first, large scenario tree, we introduce the nested clustering algorithm in Algorithm 2, and follow [37] hereafter. The nested clustering algorithm generates a tree $\mathbb{T}$ from a finite number of trajectories $\nu_j, j = 1, ..., \mathcal{M}$ with $\mathcal{M} \ll \infty$. The set of these trajectories is also called scenario fan, since the nodes of the trajectories are usually not bundled and thus, their only common point is the deterministic value at time 0. The algorithm starts with the known initial state of the process, i.e. $\nu_{j,0} = \nu_{i,0}$ for all $i, j \in \{1, ..., \mathcal{M}\}$. From the second stage on, the values of the process are unknown and must be approximated from the available trajectories. This is done for each stage subsequently, starting with stage 1. We use the k-means clustering technique to find cluster means on this stage that minimize the average minimal distance d_1 (usually a l_1 or l_2 norm), i.e. we look for values $\bar{\xi}_i^1$, representing the cluster means on stage 1, by

$$\frac{1}{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} \min_{\bar{\xi}_i^1 \in \mathbb{R}^m} \mathrm{d}_1(\nu_{j,1}, \bar{\xi}_i^1).$$

Here, i represents the index of the cluster means. The number of cluster means can be chosen separately for each stage. Moving on to stage 2, what changes now is that

in most cases none of the trajectories have any of the cluster means of stage 1 as their predecessor. In order to preserve the path dependencies in the scenario tree, all paths that were associated with a particular cluster mean $\bar{\xi}_1^i$ are considered for the k-means clustering of the next stage. These paths are again used to find cluster means in stage 2, and the predecessor of these nodes is $\bar{\xi}_i^1$. This procedure is repeated until the last level of the tree. Clustering with this method is called Voronoi tesselation.

Result: The output is a scenario tree process $\bar{\xi}$ with fixed branching structure.

begin
  Input: $\mathcal{M}$ trajectories $\nu_1, ..., \nu_{\mathcal{M}}$; branching structure, i.e. amount of nodes on each stage.
  Set stage $t = 0$ and $\bar{\xi}_1^0 = \nu_{1,0}$.
  Set the closest mean of all trajectories for level $t = 0$ to $\bar{\xi}_1^0$.
  for $t = 0, ..., T-1$ **do**
    Set M_t to the amount of nodes on level t.
    for $i = 1, ..., M_t$ **do**
      Consider all trajectories ν, that were associated with $\bar{\xi}_i^t$ as the closest mean for $\nu_{\cdot,t}$, name their set T.
      Find new cluster means $\bar{\xi}^{t+1}$ for stage $t+1$ by solving

$$\frac{1}{|T|} \sum_{\nu \in T} \min_{\bar{\xi}_l^{t+1} \in \mathbb{R}^m} d_{t+1}(\nu_{j,t+1}, \bar{\xi}_l^{t+1}) \tag{4.25}$$

      for those trajectories.
      Here, d_{t+1} is defined as the chosen distance function on stage $t+1$.
      Define the set of trajectories for which $\bar{\xi}_l^{t+1}$ is the closest mean for $\nu_{\cdot,t+1}$ as $\tilde{T}_l \subset T$.
      Define the probability to reach $\bar{\xi}_l^{t+1}$ as

$$\bar{p}_{\bar{\xi}_l^{t+1}} = \frac{|\tilde{T}_l|}{|T|}. \tag{4.26}$$

    end
  end
  return *The approximating tree $\bar{\xi}$ for all nodes.*
end

Algorithm 2: Nested clustering algorithm

Step 2 - Reduction Now that we have created a large tree, we focus on reducing it to a manageable size. This is done by merging different subtrees inside $\mathbb{T}$ into a single subtree to reduce the number of nodes. Eligible subtrees for this procedure originate from the same parent, are close to each other, and yield a merged tree that represents a good compromise between the two.

As before, the definition of closeness is given by the nested distance, which can be computed between two different trees as well as between subtrees in the same original tree. As described above, it is computed recursively, and the exact procedure is given in Algorithm 3, where we follow [50]. First, the distance between all combinations of leaves is computed based on the paths leading from the root to each leaf. Then, in a backward fashion, the minimization in Equation (4.27) is performed for each step. Its constraints ensure that the marginal transport values are equal to the conditioned probabilities in the system. Finally, the nested distance between the two trees is returned as the distance between the roots, and the optimal transport plan for each leaf is given by a multiplication of all transport values on the paths from the root to the corresponding leaf.

Example 4.24. We build two trees with differing structures to calculate the nested distance between them. Tree $\mathbb{T}$ is based on a generalized Ornstein-Uhlenbeck process with a branching structure of $(1, 3, 4, 2)$ and is depicted in Figure 4.15. The second tree $\mathbb{T}'$ is in comparison based on a generalized Brownian motion and has a branching structure of $(1, 2, 3, 2)$. It is given in Figure 4.16. The distance between both trees based on a calculation with Algorithm 3 is $D_0 = 18.1$.

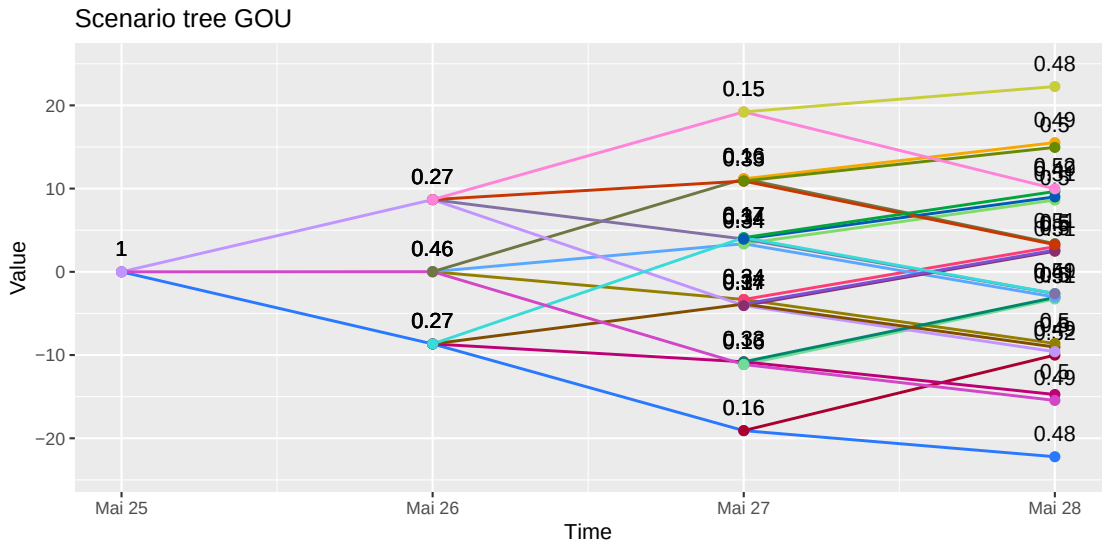

Figure 4.15: A scenario tree based on a generalized Ornstein-Uhlenbeck process calculated with Algorithm 2 and $100,000$ trajectories.

Now that we are able to compute distances between trees, the way is paved to reduce a tree in a way that the resulting smaller tree is still close to the original one. But before we dive into the algorithm, we need to define a subtree:

Result: The output are two lists, one containing the distances between different stages of the trees, the other containing the transport plans that were returned during the optimizations.

begin

Input: Tree $\mathbb{T}$ and tree $\mathbb{T}'$, order r.

Initialize: Set $t = T$. Calculate for all combinations of leaf nodes $n_i \in N_T$ and $n'_j \in N'_T$ and with respective paths $(\xi^0_{n_i}, \xi^1_{n_i}, ..., \xi^T_{n_i})$ and $(\xi^0_{n'_j}, \xi^1_{n'_j}, ..., \xi^T_{n'_j})$ the distance

$$\mathrm{D}^r_T(n_i, n'_j) := \mathrm{d}\big((\xi^0_{n_i}, \xi^1_{n_i}, ..., \xi^T_{n_i}), (\xi^0_{n'_j}, \xi^1_{n'_j}, ..., \xi^T_{n'_j})\big)^r.$$

Backward iteration:

for $t = T-1, ..., 0$ **do**

For all combinations $n_k \in N_t$ and $n'_l \in N'_t$, solve the following optimization problem:

$$\mathrm{D}^r_t(n_k, n'_l) := \min_{\pi} \sum_{n \in N_+(n_k),\, n' \in N_+(n'_l)} \pi(n, n' \mid n_k, n'_l) \cdot \mathrm{D}^r_{t+1}(n, n') \quad (4.27)$$

$$\text{s. th.} \sum_{n \in N_+(n_k)} \pi(n, n' \mid n_k, n'_l) = P(n'|n'_k), \quad n' \in N_+(n'_k),$$

$$\sum_{n' \in N_+(n'_l)} \pi(n, n' \mid n_k, n'_l) = P(n|n_l), \quad n \in N_+(n_k),$$

where $P(\cdot \mid \cdot)$ represents the conditional probability.

Save the resulting distances and the resulting transport plan in the respective lists.

end

Final step: The nested distance of the trees to one another equals the distance at their roots, i.e. at $t = 0$. The optimal transport plan of leave nodes $n_i \in N_T$ and $n'_j \in N'_T$ is then given by

$$\pi(n_i, n'_j) = \pi_1(i_1, j_1 \mid i_0, j_0) \cdot ... \cdot \pi_{T-1}(n_i, n'_j \mid i_{T-1}, j_{T-1}).$$

return *Distance list, transport list.*

end

Algorithm 3: Nested distance calculation for order r

Definition 4.25. A *subtree* $\mathbb{T}_t^k$ is part of the scenario tree $\mathbb{T}$, starts on stage t and consists of the root node n_k, its children $n_1, ..., n_m$ and their corresponding values and probabilities. Furthermore, the children's children plus all corresponding information up to the leaves are contained.
Mergeable subtrees have the same parent and start at the same stage t.

The subtree merging algorithm is defined as follows, where we follow [50] as well as [32]: First, we compute the nested distances of each candidate pair of subtrees. Then we select the pair $(\mathbb{T}_t^k, \mathbb{T}_t^l)$ with the minimum distance. If $\mathbb{T}_t^k$ and $\mathbb{T}_t^l$ are leaves, i.e. $t = T$, both leaves are removed and replaced by a new node with a score equal to the probabilistic mean score of n_k and n_l, and a probability equal to the sum of the probabilities p_k and p_l.

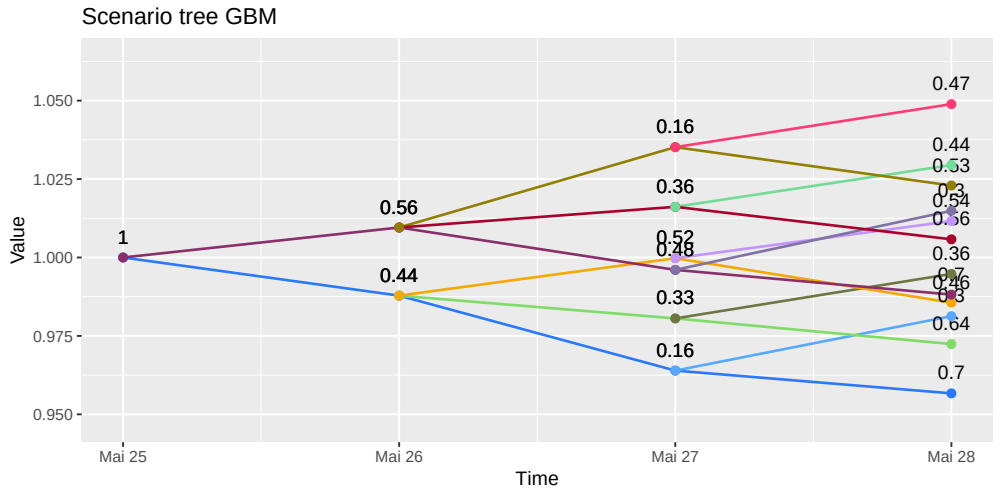

Figure 4.16: A scenario tree based on a generalized Brownian motion calculated with Algorithm 2 and $100,000$ trajectories.

If $\mathbb{T}_t^k$ and $\mathbb{T}_t^l$ are not leaves, only a certain percentage of the calculated transport values are performed, and thus only a part of the distribution of the original subtrees is transferred to the merged subtree. This percentage is called *retention level* p. If p is close to zero, the merged subtree is only marginally determined by the original ones, whereas a retention level close to one will result in a subtree approximating the distribution of the two original subtrees closely. At the same time, it can do so with many nodes, possibly more than the sum of the old nodes. This is due to the fact that if node n_k and node n_l have three children each, $n_{k_1}, n_{k_2}, n_{k_3}$ and $n_{l_1}, n_{l_2}, n_{l_3}$, transport values are built for all combinations of children and determine new nodes. Therefore it is possible that up to nine new nodes $(n_{k_1}, n_{l_1}), (n_{k_1}, n_{l_2}), ..., (n_{k_3}, n_{l_3})$ are created [32]. Therefore, the choice of p should not be taken lightly. If the subtree pair with minimal nested distance is found, the elements of the corresponding transportation map are gathered and arranged in descending order with new indices $\tilde{k}$ and $\tilde{l}$, such that

$$\pi_{\tilde{k_1}\tilde{l_1}} \geqslant ... \geqslant \pi_{\tilde{k_m}\tilde{l_m}}. \tag{4.28}$$

Here, m is chosen as the smallest integer for which $s = \sum_{i=1}^{m} \pi_{\tilde{k_i}\tilde{l_i}} \geqslant p$. These m elements contain the most costly transports and thus determine the elements that differ the most from each other. Consequently, the new 'averaged' distribution will sit on these m points and the remaining elements will be removed. The new roots are calculated, e.g. for $\pi_{\tilde{k_1}\tilde{l_1}}$ again by the mean $(\xi_{\tilde{k_1}} + \xi_{\tilde{l_1}})/2$, and the probability is given by $\pi_{\tilde{i_1}\tilde{j_1}}/s$. The division by s ensures that we get a probability distribution again. This procedure is presented in [50] and is given here by Algorithm 4.

Result: The output is a scenario tree process that does not exceed a pre-specified size but is still close to the input tree regarding the nested distance of order r.

begin

Input: Initial tree $\mathbb{T}$, retention level p, maximum number of vertices $M_{\max}$, order r.

while $\hat{V} > M_{\max}$ **do**

Tree Selection: Compute the nested distance of order r between all subtrees for all stages t. For all t, select the pair with the smallest nested distance and compute their optimal transport plan π.

Tree Merging:

1) Calculate the new root value as the mean between the two roots of the selected trees.

2) The new nodes $(n_{\tilde{k_1}}, n_{\tilde{l_1}}), ..., (n_{\tilde{k_m}}, n_{\tilde{l_m}})$ are found first by following Equation (4.28) and then are calculated as mean of the original node values.

3) Calculate the new probabilities by $\pi_{n_{\tilde{k_1}}, n_{\tilde{l_1}}}/s, ..., \pi_{n_{\tilde{k_m}}, n_{\tilde{j_m}}}/s$.

4) Then, in a recursive step, merge the subtrees starting from the new node pairs in step 2).

Calculate new number of nodes $\hat{V}$.

end

return *The reduced tree* $\bar{\mathbb{T}}$.

end

Algorithm 4: Tree reduction algorithm

The described procedure is repeated in a recursive structure with the subtrees starting from the new root nodes $(\xi_{\tilde{k_i}} + \xi_{\tilde{l_j}})/2$ determined above. As long as the number of nodes $\hat{V}$ exceeds the allowed number of nodes $M_{\max}$, the next pair with the second-lowest nested distance is selected and the procedure continues.

For larger trees, the computation of a single nested distance can become quite time intensive, let alone the computation of many nested distances as described in Algorithm 4.

The original formulation assumes a recalculation of all nested distances within the original tree after each reduction step, which are by far the most computationally intensive steps. We modify the algorithm in two respects in Algorithm 5: First, instead of computing the nested distances for all eligible subtrees in a complicated procedure, we compute the distance of the whole tree to itself. Of course, on level 0 this will be 0, but on all other levels it gives us the distances between the leaves of the nodes as we need them. The advantage of this method is that we save the time of cutting the paths to the length of the specific subtree we are looking at. This procedure is possible because for all subtrees in $\mathbb{T}$ that are considered eligible for the algorithm, the following holds true:

Proposition 4.26. *Let $\mathbb{T}$ be a tree with stages $0, ..., T$, and $\mathbb{T}_1, \mathbb{T}_2$ being subtrees of $\mathbb{T}$, both beginning on the same stage t and with root nodes n_1 and n_2, respectively. Furthermore, we require $\overline{N_-(n_1)} = \overline{N_-(n_2)}$, i.e. that both trees have the same predecessors. Then it holds true that $D_t^r(n_1, n_2)|\mathbb{T} = D_t^r(n_1, n_2)$.*

Note that by $\mathrm{D}_t^r(n_1, n_2)|\mathbb{T}$, we denote the calculation of the nested distance between $\mathbb{T}_1$ and $\mathbb{T}_2$ based on the paths of the whole tree $\mathbb{T}$, i.e. they do not start at level t but at level 0.

Proof. We will prove this by induction. For the initial case, we set $t = T$ and therefore $\mathbb{T}_1 = n_1$ and $\mathbb{T}_2 = n_2$ are degenerated trees, i.e. leaves. Then, we find

$$\begin{aligned}\mathrm{D}_T^r(n_1, n_2)|\mathbb{T} = \mathrm{d}(n_1, n_2)^r|\mathbb{T} &= \sqrt[r]{\sum_{i=0}^{T} |\xi_i^{n_1} - \xi_i^{n_2}|^r} = \sqrt[r]{|\xi_T^{n_1} - \xi_T^{n_2}|^r} \\ &= \mathrm{D}_T^r(n_1, n_2),\end{aligned}$$

where the third equality sign stems from the assumption that both subtrees have the same predecessors. Now, let t be arbitrary but fixed. Then, we do the induction step and look at $t-1$ with $\mathbb{T}_1, \mathbb{T}_2$ not being degenerate:

$$\begin{aligned}\mathrm{D}_{t-1}^r(n_1, n_2)|\mathbb{T} &:= \min_{\pi} \sum_{n \in N_+(n_1),\, n' \in N_+(n_2)} \pi(n, n' \mid n_1, n_2) \cdot \mathrm{D}_t^r(n, n')|\mathbb{T} \\ &= \min_{\pi} \sum_{n \in N_+(n_1),\, n' \in N_+(n_2)} \pi(n, n' \mid n_1, n_2) \cdot \mathrm{D}_t^r(n, n') \\ &= \mathrm{D}_{t-1}^r(n_1, n_2), \qquad (4.29)\end{aligned}$$

where the second equality sign is based on the initial case. □

The second difference between Algorithms 4 and 5 is that the nested distances are not all recalculated after each reduction. Instead, the calculated nested distances as well as the tree itself are split into a part that is affected by the reduction and a part that is unaffected and therefore does not change anyway. After calculating the reduced tree element based on the affected part of the tree, both affected parts are deleted. Then the nested distance between the unaffected part and the new reduced tree is calculated

and merged with the unaffected nested distances. Finally, the reduced tree is merged with the unaffected tree part. This is done recursively. This method prevents us from recalculating nested distances that have not changed and therefore do not need to be recalculated.

Result: The output is a scenario tree process that does not exceed a pre-specified size but is still similar to the input tree.

begin

Input: Initial tree $\mathbb{T}$, retention level p, maximum number of vertices $M_{\max}$, order r.

Compute the nested distance of order r of the tree to itself.

while $\hat{V} > M_{\max}$ **do**

Find the subtrees $\mathbb{T}_1$ and $\mathbb{T}_2$ with the smallest distance between them and calculate their corresponding transport plan.

Separate $\mathbb{T}$ and $\mathbb{T}_1 \cup \mathbb{T}_2$. Also separate nested distances belonging to $\mathbb{T}_1$ or $\mathbb{T}_2$.

Tree Merging:

1) Calculate the new root value as the mean between the two roots of $\mathbb{T}_1$ and $\mathbb{T}_2$.

2) The new nodes $(n_{\tilde{i_1}}, n_{\tilde{j_1}}), ..., (n_{\tilde{i_m}}, n_{\tilde{j_m}})$ are found first by following Equation (4.28) and then are calculated as mean of the original node values.

3) Calculate the new probabilities by $\pi_{\tilde{i_1},\tilde{j_1}}/s, ..., \pi_{\tilde{i_m},\tilde{j_m}}/s$.

4) Then, in a recursive step, merge the subtrees starting from the new node pairs in step 2).

Calculate the changed nested distance between the merged branches and the separated main tree.

Assign the separated distance of the main tree as add-on to the changed nested distance.

Assign the merged branches as add-on to the separated main tree.

end

return *The reduced tree* $\bar{\mathbb{T}}$.

end

Algorithm 5: Altered tree reduction algorithm

Step 3 - Optimization Our final step is to adjust the probabilities and scenario values on the reduced tree while keeping the achieved tree topology intact. The probabilities and values are adjusted to reduce the distance between the reduced tree and the large

tree. The algorithm we present here is based on stochastic approximation and the idea to modify tree values and probabilities, again through using trajectories; we follow [37] and [51] here. The idea behind it is to utilize that we have a model at hand which allows to generate new trajectories arbitrarily. These trajectories are used to refine the existing tree values and probabilities while maintaining the tree structure. For a trajectory $\nu_j := (\nu_{j,0}, ...\nu_{j,T})$ used in the j-th iteration, we find the tree path $(\bar{\xi}_*^0, ..., \bar{\xi}_*^T)$ which is closest to the trajectory. This path is found by

$$\min_{\bar{\xi}} \sum_{t=0}^{T} \mathrm{d}_t\big(\nu_{j,t}, \bar{\xi}_*^t\big) \tag{4.30}$$

for all $t = 0, ..., T$, where the node corresponding to $\bar{\xi}_*^t$ is contained in N_t and must have the nodes corresponding to $\bar{\xi}_*^1, ..., \bar{\xi}_*^{t-1}$ as its predecessors. Furthermore, d_t represents the distance measure used on stage t, and a_k is a step sequence with $a_k > 0$ and $\sum_{k=1}^{\infty} a_k = \infty$ as well as $\sum_{k=1}^{\infty} a_k^2 < \infty$. [37] propose $a_k = 1/(30 + k)$, as this has proven itself in practice. The values of the path $(\bar{\xi}_*^0, ..., \bar{\xi}_*^T)$ are updated in the k-th iteration to

$$\bar{\xi}_*^t(k+1) \leftarrow (1 - a_k)\bar{\xi}_*^t(k) + a_k\, \nu_{k,t} \qquad t = 0, ..., T. \tag{4.31}$$

The values of all other vertices stay the same. Finally, the algorithm terminates after $M_{\max}$ iterations or if the relative change of $\bar{\xi}_*^t(k+1)$ to $\bar{\xi}_*^t(k)$ is small enough. We then compute the final conditional probabilities of each vertex.

Result: The output is a small tree that is fitted as close as possible to a given stochastic process.

begin
  Input: Initial approximating tree $\mathbb{T}$, number of iterations $M_{\max}$, trajectories of stochastic process ν, step sequence a_k, threshold τ.
  Set $k = 1$. Set $d = \tau + 1$.
  while $k \leqslant M_{\max}$ *and* $d > \tau$ **do**
    Use trajectory ν^k. Find the path $(\bar{\xi}_*^0, ..., \bar{\xi}_*^T)$ in $\mathbb{T}$ through Equation (4.30) for all $t = 0, ..., T$.
    Update the path following the update step (4.31).
    Set $d = ||\bar{\xi}_*^T(k+1) - \bar{\xi}_*^T(k)||$.
    Set $k = k + 1$.
  end
  Compute new probabilities $p(n)$ for all nodes by using m further trajectories $p(n) = \frac{1}{m}\#\{\nu : n \text{ is corresponding node to one of } (\bar{\xi}_*^0, ..., \bar{\xi}_*^T)\}$.
  return *The improved tree* $\mathbb{T}$ *and estimation* $D_r(\xi, \mathbb{T})$.
end

Algorithm 6: Stochastic approximation for scenario trees

Example 4.27. As an example, we present a tree $\mathbb{T}$ based on a generalized Ornstein-Uhlenbeck process. Its original form is found in Figure 4.17. After application of Algo-

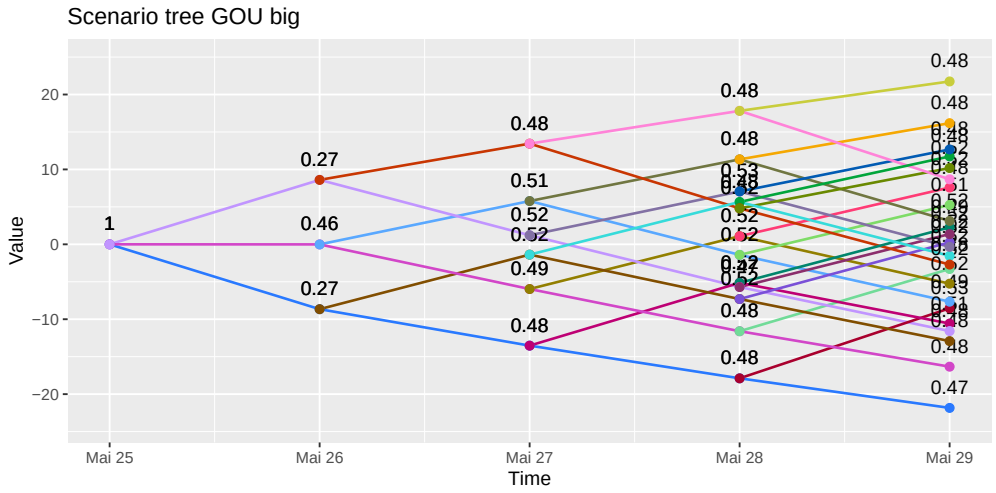

Figure 4.17: A scenario tree based on a generalized Ornstein-Uhlenbeck process calculated with Algorithm 2 and $100,000$ trajectories.

rithm 4, the reduced tree can be found in Figure 4.18. In comparison to 46 nodes, it now is based on 20 nodes only and therefore has reduced the amount of nodes by more than half. The fact that in this example, each branch starting from the root node has the mirrored structure of the other, is due to the generalized Ornstein-Uhlenbeck process being a symmetric process. In other cases, a tree structure exploiting the possibility of nodes on the same stage having a different number of children or different structures can result from this algorithm as well to best match the underlying process structure. Finally, the reduced tree is improved by the usage of Algorithm 6. Here, the tree structure is kept fixed and only values as well as probabilities are allowed to change.

4.2.5 Scenario tree construction - heuristics

Many models consist of several independent factors or stochastic processes to be simulated. If some of these processes take longer to compute, it is worth analyzing the importance of each factor for modeling the whole. For example, consider a model in which one process takes ten times as long to simulate as the other processes, but only has a minor impact on the overall behavior of the model. However, often a large amount of overall paths is required for generating a scenario tree. In this case, instead of simulating the entire model, the idea is to simulate the underlying processes individually and generate more or fewer paths per process depending on their contribution to the model, thus keeping the total simulation time low. To make this possible, we first define a method for adding trees to each other, so that the trees of the individual factors can be merged:

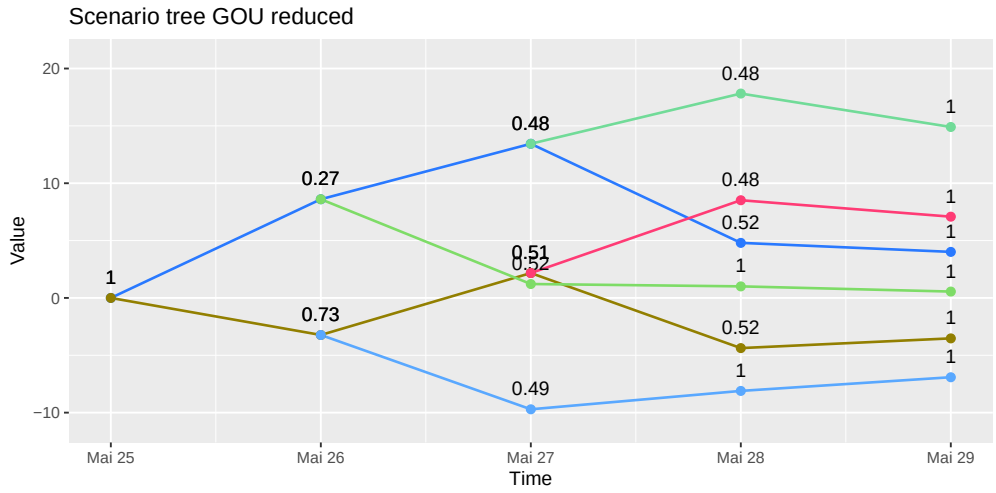

Figure 4.18: The reduced scenario tree based on Figure 4.17. It was calculated with Algorithm 4.

Definition 4.28. For scenario trees $\mathbb{T}_1$ and $\mathbb{T}_2$ with the same height T and their value processes in $\mathbb{R}$, we define *tree addition* as

$$\bigoplus : \mathbb{T}_1 \times \mathbb{T}_2 \to \mathbb{T}_3,$$
$$\bigoplus(n_i^1, n_j^2) \mapsto n_{i,j}^3 = (n_i^1, n_j^2)^T \quad \forall i = 1, ..., M_1,\ j = 1, ..., M_2.$$

Consequently, both trees $\mathbb{T}_1$ and $\mathbb{T}_2$ are merged to one tree $\mathbb{T}_3$, with value process in $\mathbb{R}^2$. With this, all information from the original trees is contained in the new tree.

Remark 4.29. This form of tree addition only captures all relevant information when the underlying stochastic processes are independent from each other and should therefore only be used in such cases.

Now, based on this form of tree addition, we prove in the case of the model from Equation (4.4) that joining single factor trees to one large model tree is distributionally equivalent to building one model tree from the beginning.

Proposition 4.30. *For the model in Equation* (4.4) *and* $t = 0, ..., T$*, it holds true that*

$$\Lambda\mathbb{T}_{X_n}(t) \bigoplus \mathbb{T}_{Y_n}(t) \bigoplus \mathbb{T}_{Z_n}(t) \overset{d}{=} \mathbb{T}_{\Lambda X_n + Y_n + Z_n}(t) \qquad \forall t$$

and

$$\Lambda\mathbb{T}_{X_n}(t) \bigoplus \mathbb{T}_{Y_n}(t) \bigoplus \mathbb{T}_{Z_n}(t) \overset{n\to\infty}{\longrightarrow} \mathbb{T}_{\Lambda X + Y + Z}(t) \qquad \forall t$$

in weak sense.*

Proof. We first consider w.l.o.g. the process X with its distribution at time t given through $P_{X(t)}$. This is approximated by the tree $\mathbb{T}_{X_n}(t)$ with corresponding distribution $\tilde{P}_{X_n(t)}$, with n representing the number of nodes in the tree. As $\mathbb{T}_{X_n}(t)$ by construction minimizes the Wasserstein distance for the corresponding n, it holds true that

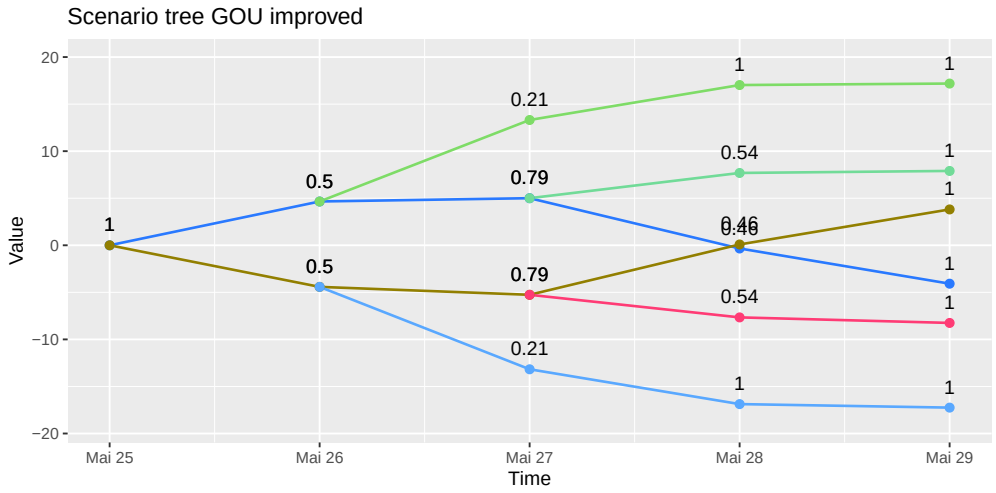

Figure 4.19: The improved scenario tree based on Figure 4.18. It was calculated with Algorithm 6.

$\mathfrak{d}_r(\tilde{P}_{X_n(t)}, P_{X(t)}) \longrightarrow 0$ for $n \to \infty$. Using Theorem 4.15, this is equivalent to a convergence in weak* sense, and therefore equivalent to convergence in distribution. This holds true for all considered trees. Consequently, we now have to prove that this convergence in distribution from the single trees' distributions equals the convergence of their sum.

For that, we know from Lévy's continuity theorem that from convergence in distribution follows pointwise convergence of the corresponding characteristic functions. Using this theorem and that we have a bijection between characteristic functions and probability distributions, we find

$$\begin{aligned}
\varphi_{\Lambda X_n + Y_n + Z_n}(a) &= \mathbb{E}\left(e^{ia(\Lambda X_n + Y_n + Z_n)}\right) \\
&= \mathbb{E}\left(e^{ia(\Lambda X_n)} e^{ia(Y_n)} e^{ia(Z_n)}\right) \\
&= \mathbb{E}\left(e^{ia(\Lambda X_n)}\right)\mathbb{E}\left(e^{ia(Y_n)}\right)\mathbb{E}\left(e^{ia(Z_n)}\right) \\
&= \varphi_{\Lambda X_n}(a)\varphi_{Y_n}(a)\varphi_{Z_n}(a) \\
&\to \varphi_{\Lambda X}(a)\varphi_{Y}(a)\varphi_{Z}(a) \\
&= \varphi_{\Lambda X + Y + Z}(a) \qquad \forall a. \qquad (4.32)
\end{aligned}$$

Be reminded that all stochastic processes in model 4.4 are independent; this is used in the third equality. Consequently, the first assertion holds through the fourth equality sign, and the second follows from the shown convergence. □

Remark 4.31. The proposition is meant to be indicative for the special case in this thesis; nonetheless, it is easily extended to the setting of $\mathbb{M}$ different, independent factors.

We now present a heuristic to suggest which processes should be given which weight at the desired level of discretization. We assume that $\mathbb{M}$ factors of the original model are involved, that they are independent of each other and that the model as such has an additive structure. Then, a number of stages T_s is chosen which is smaller than the actually desired number of stages T. For the former, the whole model is built as a tree with Algorithm 2, resulting in tree $\mathbb{T}_O$, and a chosen fineness, i.e. a structure $(s_1, ..., s_{T_s})$. Then, Algorithm 7 is applied. It computes the factor division with $\mathbb{M}$ components for every $s_t, t = 1, .., T_s$, and collects its permutations in a list for every level. Given these lists, the algorithm computes the corresponding structure for each individual process tree and builds the trees accordingly. They are then added to a large tree $\mathbb{T}_A$ and the nested distance between $\mathbb{T}_A$ and $\mathbb{T}_O$ is calculated. This process is repeated for each possible structure combination and finally the algorithm returns a table with the selected structures and the resulting nested distances.

Result: The output is a mapping of each process to how many paths of it should be simulated.

begin
  Input: $\mathbb{T}_O$ as tree of the original model, with T_s stages and structure $(\int_1, ..., \int_{T_s})$. Also, models 1 to $\mathbb{M}$ for all independent processes.
  For every stages $t = 2, ..., T_s$, build a list $list(t)$ containing the permutations of the prime factor division with $\mathbb{M}$ components of s_t.
  Create a table res to contain the results later on.
  for *entry*$_2 \in list(2)$ **do**
    ⋮
    for *entry*$_{T_s} \in list(T_s)$ **do**
      for $i = 1, ..., \mathbb{M}$ **do**
        struct$_i$ = vector(1, entry$_2$[i],...,entry$_{T_s}[i]$).
        $\mathbb{T}_i$ = Algorithm 2 for model i with structure struct$_i$.
      end
      Add all trees $\mathbb{T}_i, i = 1, ..., \mathbb{M}$ to $\mathbb{T}_A$.
      Check nested distance between $\mathbb{T}_A$ and $\mathbb{T}_O$ with Algorithm 3.
      Append a line with all structures and the final nested distance to res.
    end
  end
  return *A table res containing all tested structures as well as the corresponding nested distance to $\mathbb{T}_O$.*
end

Algorithm 7: Tree process weighting heuristic

Example 4.32. We demonstrate the functionality of Algorithm 7 with an example. Starting from the day ahead market model from Equation (4.4), we move step by step through the algorithm. In this example, the factor in Equation (4.7), i.e. the jump process, is removed from the model. Consequently, we have two sub-processes that we

can represent individually. Let the desired shorter period be $T_s = 3$, so we need to schedule two more periods after step 1. Also, let the desired structure be $(1, 24, 24)$, i.e., we reach a tree with 576 scenarios. The prime factorization for two components yields the elements $(1, 24), (2, 12), (3, 8), (4, 6)$ for both levels. Therefore, for each of these splits, modeling is now started and the nested distance is calculated. The results of this calculation can be found in Table 4.20 and 4.21. We see that for the chosen fineness of the discretization, most weight is placed on the Generalized Ornstein-Uhlenbeck process, which governs nearly all activity of the whole process. Nonetheless, we find that the Geometric Brownian Motion process is also slightly taken into account. In comparison, we find that for a desired structure of $(1, 12, 12)$ with corresponding prime factorization elements $(1, 12), (2, 6), (3, 4)$, this changes a little, as visible in Table 4.22. There, the tree which is closest to the original is the one with a structure of $(1, 12, 12)$ for the Generalized Ornstein-Uhlenbeck process and only $(1, 1, 1)$, i. e. one calculated path, for the Geometric Brownian Motion. Consequently, the chosen fineness of the discretization puts most emphasis on the former process.

Of course, the approximation of the original tree is worse when each factor is simulated individually than when the whole model is simulated and processed. This is due to the simple fact that what we are trying to achieve with the heuristic, namely the correct weighting of the different processes, is done by the whole model itself intrinsically. Also, fewer process values of each factor are simulated. This is due to the fact that the definition of tree addition yields $M_1 \cdot M_2$ nodes for the new tree and for $M_1 \geqslant 2$ and $M_2 \geqslant 2$, it holds true that $M_1 + M_2 \leqslant M_1 \cdot M_2$. Consequently, the resulting nodes of the merged tree $\mathbb{T}_A$ will be less well distributed over the image space than the original tree $\mathbb{T}_O$.

Nevertheless, this method has the following three advantages: The first is clearly the smaller number of paths needed to compute the same number of nodes. The second is the fact that for processes with important, but rarely appearing paths, the initial quantizers can be chosen with more care. This can indeed be necessary for jump processes with low jump rates, as we will see later in Section 4.4. For jump processes, the initial T_s quantizers can be initialized as the set of the process value where no jump has happened in combination with $T_s - 1$ optimal quantizers of the jump distribution. This initialization helps to stabilize the convergence of the tree even with rare events. Finally, the third advantage comes into effect when the dimension of the value process ξ is $m = 1$. In [50], two algorithms are presented that can be used in exactly this case and when the probability distribution of a random variable is known. They use this known distribution to find the optimal quantizers more easily. If some of the factors of the model have an explicit form of their density and distribution, and others do not, optimal quantizers for the former can be computed much faster. We present both algorithms below. Algorithm 8 is used for the Wasserstein distance of order $r = 1$, whereas Algorithm 9 is optimal for quantizers when using the Wasserstein distance of order $r = 2$. Both algorithms proceed such that they use a set of initial quantizers q_i, $i = 1, ..., T_s$. This set is gradually refined. In case of Algorithm 8, this refinement step takes place by first finding the break points in between the quantizers. For those, the distribution values

Number	Structure GOU	Structure GBM	Nested Distance
1	(1, 1, 1)	(1, 24, 24)	21.86
2	(1, 1, 2)	(1, 24, 12)	18.61
3	(1, 1, 3)	(1, 24, 8)	17.77
4	(1, 1, 4)	(1, 24, 6)	17.53
5	(1, 1, 6)	(1, 24, 4)	17.39
6	(1, 1, 8)	(1, 24, 3)	17.39
7	(1, 1, 12)	(1, 24, 2)	17.34
8	(1, 1, 24)	(1, 24, 1)	17.21
9	(1, 2, 1)	(1, 12, 24)	14.90
10	(1, 2, 2)	(1, 12, 12)	10.55
11	(1, 2, 3)	(1, 12, 8)	9.15
12	(1, 2, 4)	(1, 12, 6)	8.63
13	(1, 2, 6)	(1, 12, 4)	8.33
14	(1, 2, 8)	(1, 12, 3)	8.25
15	(1, 2, 12)	(1, 12, 2)	8.13
16	(1, 2, 24)	(1, 12, 1)	7.89
17	(1, 3, 1)	(1, 8, 24)	12.67
18	(1, 3, 2)	(1, 8, 12)	8.09
19	(1, 3, 3)	(1, 8, 8)	6.56
20	(1, 3, 4)	(1, 8, 6)	5.95
21	(1, 3, 6)	(1, 8, 4)	5.59
22	(1, 3, 8)	(1, 8, 3)	5.50
23	(1, 3, 12)	(1, 8, 2)	5.36
24	(1, 3, 24)	(1, 8, 1)	5.13
25	(1, 4, 1)	(1, 6, 24)	12.02
26	(1, 4, 2)	(1, 6, 12)	7.38
27	(1, 4, 3)	(1, 6, 8)	5.82
28	(1, 4, 4)	(1, 6, 6)	5.23
29	(1, 4, 6)	(1, 6, 4)	4.87
30	(1, 4, 8)	(1, 6, 3)	4.78
31	(1, 4, 12)	(1, 6, 2)	4.61
32	(1, 4, 24)	(1, 6, 1)	4.41
33	(1, 6, 1)	(1, 4, 24)	11.46
34	(1, 6, 2)	(1, 4, 12)	6.77
35	(1, 6, 3)	(1, 4, 8)	5.23
36	(1, 6, 4)	(1, 4, 6)	4.65
37	(1, 6, 6)	(1, 4, 4)	4.29
38	(1, 6, 8)	(1, 4, 3)	4.25
39	(1, 6, 12)	(1, 4, 2)	4.07

Table 4.20: Result table for Example 4.32 with nested distances for structure $(1, 24, 24)$, lowest value in bold face.

Number	Structure GOU	Structure GBM	Nested Distance
40	(1, 6, 24)	(1, 4, 1)	3.88
41	(1, 8, 1)	(1, 3, 24)	11.55
42	(1, 8, 2)	(1, 3, 12)	6.97
43	(1, 8, 3)	(1, 3, 8)	5.53
44	(1, 8, 4)	(1, 3, 6)	5.02
45	(1, 8, 6)	(1, 3, 4)	4.75
46	(1, 8, 8)	(1, 3, 3)	4.65
47	(1, 8, 12)	(1, 3, 2)	4.53
48	(1, 8, 24)	(1, 3, 1)	4.39
49	(1, 12, 1)	(1, 2, 24)	11.08
50	(1, 12, 2)	(1, 2, 12)	6.48
51	(1, 12, 3)	(1, 2, 8)	5.08
52	(1, 12, 4)	(1, 2, 6)	4.57
53	(1, 12, 6)	(1, 2, 4)	4.38
54	(1, 12, 8)	(1, 2, 3)	4.28
55	(1, 12, 12)	(1, 2, 2)	4.12
56	(1, 12, 24)	(1, 2, 1)	**3.63**
57	(1, 24, 1)	(1, 1, 24)	10.10
58	(1, 24, 2)	(1, 1, 12)	5.69
59	(1, 24, 3)	(1, 1, 8)	4.55
60	(1, 24, 4)	(1, 1, 6)	4.23
61	(1, 24, 6)	(1, 1, 4)	4.22
62	(1, 24, 8)	(1, 1, 3)	3.96
63	(1, 24, 12)	(1, 1, 2)	3.75
64	(1, 24, 24)	(1, 1, 1)	3.65

Table 4.21: Continued: Result table for Example 4.32 for nested distances for structure $(1, 24, 24)$, lowest value in bold face.

Number	Structure GOU	Structure GBM	Nested Distance
1	(1, 1, 1)	(1, 12, 12)	21.95
2	(1, 1, 2)	(1, 12, 6)	18.88
3	(1, 1, 3)	(1, 12, 4)	18.11
4	(1, 1, 4)	(1, 12, 3)	17.88
5	(1, 1, 6)	(1, 12, 2)	17.81
6	(1, 1, 12)	(1, 12, 1)	17.58
7	(1, 2, 1)	(1, 6, 12)	15.19
8	(1, 2, 2)	(1, 6, 6)	11.07
9	(1, 2, 3)	(1, 6, 4)	10.03
10	(1, 2, 4)	(1, 6, 3)	9.57
11	(1, 2, 6)	(1, 6, 2)	9.37
12	(1, 2, 12)	(1, 6, 1)	9.12
13	(1, 3, 1)	(1, 4, 12)	13.89
14	(1, 3, 2)	(1, 4, 6)	9.64
15	(1, 3, 3)	(1, 4, 4)	8.58
16	(1, 3, 4)	(1, 4, 3)	8.09
17	(1, 3, 6)	(1, 4, 2)	7.93
18	(1, 3, 12)	(1, 4, 1)	7.52
19	(1, 4, 1)	(1, 3, 12)	12.84
20	(1, 4, 2)	(1, 3, 6)	8.53
21	(1, 4, 3)	(1, 3, 4)	7.69
22	(1, 4, 4)	(1, 3, 3)	7.11
23	(1, 4, 6)	(1, 3, 2)	6.94
24	(1, 4, 12)	(1, 3, 1)	6.52
25	(1, 6, 1)	(1, 2, 12)	13.03
26	(1, 6, 2)	(1, 2, 6)	8.90
27	(1, 6, 3)	(1, 2, 4)	8.01
28	(1, 6, 4)	(1, 2, 3)	7.67
29	(1, 6, 6)	(1, 2, 2)	7.56
30	(1, 6, 12)	(1, 2, 1)	7.05
31	(1, 12, 1)	(1, 1, 12)	10.99
32	(1, 12, 2)	(1, 1, 6)	7.24
33	(1, 12, 3)	(1, 1, 4)	6.22
34	(1, 12, 4)	(1, 1, 3)	6.30
35	(1, 12, 6)	(1, 1, 2)	5.92
36	(1, 12, 12)	(1, 1, 1)	**5.53**

Table 4.22: Result table for Example 4.32 for nested distances for structure $(1, 12, 12)$, lowest value in bold face.

Result: The output are optimal quantizers for random variables with dimension $m = 1$ and the Wasserstein distance of order $r = 1$.

begin

Input: Initial quantizers q_i, $i = 1, ..., T_s$ ordered by size, threshold τ, distribution function F.

Set breakpoints $b_0 = -\infty, b_{T_s} = \infty$ and $d = \tau + 1$.

while $d > \tau$ **do**

Find breakpoints $b_i = \frac{1}{2}(q_i + q_{i+1})$ for $i = 1, ..., T_s - 1$.

Find values $g_i = F(b_i)$ for all i.

Find new medians of F conditioned on the intervals determined by breakpoints b_i, i.e.

$$\tilde{q}_i = F^{-1}\Big(\frac{g_{i-1} + g_i}{2}\Big).$$

Set $d = \sum_i |\tilde{q}_i - q_i|$.

Set $q_i = \tilde{q}_i$.

end

Compute probabilities $p_i = g_i - g_{i-1}$ for $i = 1, ..., T_s$. **return** *The optimal quantizers q_i and their corresponding probabilities p_i.*

end

Algorithm 8: Optimal quantizers for dimension $m = 1$ and Wasserstein distance of order $r = 1$

are computed, and the mean of both is then mapped back through the inverse of the distribution function. This step defines the new values of the quantizers as the median of the distribution function conditioned on the corresponding intervals. Their order does not change through this step. Finally, when the difference between old and new quantizer set is small enough, the corresponding probabilities for all quantizers are computed and both sets are returned. In comparison to that, Algorithm 9 opts for the conditional mean instead of the median, as the mean is what minimizes the expected quadratic error.

Depending on the order of the Wasserstein distance of interest, one of the algorithms can be used when the probability distribution of the corresponding stochastic process is easily computable. In the special case of the model in Equation (4.4), this holds true for the first two factors - the first is lognormally and the second normally distributed. In comparison to that, the third factor is defined through a Lévy process whose probability distribution for each point in time indeed can be written down, but contains an infinite sum of convolutions and is thus quite difficult to compute and program. Therefore, it does make sense to compute optimal quantizers for the first two factors with Algorithm 8 or 9, and to use Algorithm 2 for the third factor.

Result: The output are optimal quantizers for random variables with dimension $m = 1$ and the Wasserstein distance of order $r = 2$.

begin

Input: Initial quantizers q_i, $i = 1, ..., T_s$ ordered by size, threshold τ, distribution function F.

Set breakpoints $b_0 = -\infty, b_{T_s} = \infty$ and $d = \tau + 1$.

while $d > \tau$ **do**

Find breakpoints $b_i = \frac{1}{2}(q_i + q_{i+1})$ for $i = 1, ..., T_s - 1$ and define intervals $I_i = b_i - b_{i-1}$ for $i = 1, ..., T_s$.

Find values $g_i = F(b_i)$ for all i.

Compute probabilities $p_i = g_i - g_{i-1}$ for $i = 1, ..., T_s$.

Find new means of F conditioned on the intervals determined by breakpoints b_i, i.e.

$$\tilde{q}_i = \frac{1}{p_i} \int_{I_i} u \, dF(u).$$

Set $d = \sum_i |\tilde{q}_i - q_i|$.

Set $q_i = \tilde{q}_i$.

end

return *The optimal quantizers q_i and their corresponding probabilities p_i.*

end

Algorithm 9: Optimal quantizers for dimension $m = 1$ and Wasserstein distance of order $r = 2$

4.3 Stochastic optimization

After discussing the basis for an optimization, i.e. a method to find a discrete representation of the underlying model, it remains to choose a method to optimize strategies based on the discretized model. Since the setting that we consider includes a look-ahead period of one week, it is necessary to consider methods that can handle this. In the following, we therefore take a look at stochastic optimization methods.

Stochastic optimization is concerned with incorporating uncertainty in optimization problems, either in the objective function or in the constraints. Thus, the term "stochastic optimization" is an umbrella term for a collection of methods used to pose and solve optimization problems with random elements. As the area of stochastic optimization has been worked on by many different disciplines and research communities, it has developed a variety of strategies for dealing with uncertainty in some of the parameters of an optimization problem. Their methods are distinguished by a number of differences, e.g., in notation, in decision epochs, in how randomness is incorporated and which elements may or may not be affected by it, in how transition to another state is handled, and so on. These differences immediately suggest that there is not one best method for solving stochastic optimization problems, but rather many, each with its own area of excellence. There is dynamic programming, which searches for Markov decision rules that optimally move from one state to another over the entire time horizon. Then there is stochastic programming, which focuses on optimally solving the first-stage decision based on scenarios that capture all randomness. There is also reinforcement learning and others, all of which approach the problem in their own way, but all of which have similarities to each other.

Stochastic optimization may be used to optimize decisions under uncertainty where a one-time decision has to be made. An example for this is to calibrate parameters of a model to a data set. It also contains techniques that appear when a one-time decision must be made followed by a recourse decision. Here, one can think of the placements of warehouses as the one-time decision and the actual transportation management from the warehouses to customers as recourse decision after demand has become known. A recourse decision is in common language of the stochastic programming community the second-stage decision in a two-stage problem, that can be made in response to the revelation of a stochastic exogenous factor after the first-stage decision. In the example, this exogenous factor would be actual demand from customers, and the recourse decision manages which customer is supplied from which warehouse. As the setting we consider does not contain such a one-time decision, but rather a repetition of the same decision under different conditions over several days, all explanations following will be restricted to this area. As a further consequence, the functional that will be optimized is the expectation rather than a lower or upper bound quantile – for repeated decisions and thus based on the law of large numbers, it makes sense to minimize the incurred costs on average. For the detailed explanation below, we follow [50].

We pose the stochastic optimization problem for this setting without settling for a specific area just yet. For that purpose, we define the following:

- **Decision epoch**: $t \in [0, ..., T]$ denotes the decision epoch from the decision epoch set $[0, ..., T]$, where $T \in \mathbb{N} \cup \infty$ represents the final decision epoch. The decision epochs and the passing time do not have to coincide, but as both are equivalent in this thesis, t will also be referred to as time.
- **Exogenous information**: ξ_t, $t \in [0, ..., T]$, denotes a stochastic process that represents exogenous information. It is thus our source of randomness. The evolution or history of ξ over time is denoted by $\xi_{0:t} = (\xi_0, \xi_1, ..., \xi_t)$. We assume ξ_0 to be known; it is thus not a random variable.
- **State**: $\mathcal{S}(t)$ is the state of the system at time t stemming from the set of states $\mathbb{S}_t$. It contains information about the system at time t that can depend on $\xi_{0:t}$ and on the actions $x_0, ..., x_{t-1}$, and will be introduced in more detail in the following subsections.
- **Action**: $x_t \in \mathbb{X}_t$ represents a feasible action, i.e. an action that fulfills all given constraints with probability 1, from action set $\mathbb{X}_t := \mathbb{X}_t(x_{t-1}, ..., x_0, \xi_{0:t})$. $\mathbb{X}_t$ depends on all taken decisions and realized observations up to time t and contains only actions for which the expected costs are defined for all system states. In the following, x_t will also be called decision, and $\mathbb{X}_t$ decision set at time t.
- **Cost**: $C_t(s, x_t, \xi_t)$, $t = 0, ..., T$, represents the cost of choosing action x_t based on the realization of ξ_t at decision epoch t, when the system state is given by $\mathcal{S}(t) = s$. For $t = 0$, it holds true that $C_0(\mathcal{S}(0), x_0, \xi_0)$ does not depend on stochastic input, as ξ_0 is known.

The optimization problem to be solved is then given through

$$\begin{aligned} &\min_{x_0 \in \mathbb{X}_0} C_0(\mathcal{S}(0), x_0, \xi_0) + f_T(1), \\ &\qquad f_T(t) := \mathbb{E}\Big[\inf_{x_t \in \mathbb{X}_t(x_{t-1}, ..., x_0, \xi_{0:t})} C_t(\mathcal{S}(t), x_t, \xi_t) + f_T(t+1)\Big], \\ &\qquad f_T(T+1) := 0. \end{aligned} \tag{4.33}$$

Depending on the literature, the system state is used, and then it usually contains ξ already, or it is not used, then the focus is on ξ only. Nonetheless, to make the explanation applicable for both approaches, we itemize the system state and the exogenous information separately for now. This is equivalent to the following equation, which stresses the nested nature of the problem:

$$\begin{aligned} \min_{x_0 \in \mathbb{X}_0} C_0(\mathcal{S}(0), x_0, \xi_0) + \mathbb{E}\Big[\inf_{x_1 \in \mathbb{X}_1(x_0, \xi_1)} C_1(\mathcal{S}(1), x_1, \xi_1) + \\ \mathbb{E}\Big[... + \mathbb{E}\big[\inf_{x_T \in \mathbb{X}_T(x_{T-1}, ..., x_0, \xi_{0:T})} C_T(\mathcal{S}(T), x_T, \xi_T)\big]\Big]\Big]. \end{aligned} \tag{4.34}$$

It is important to mention that the action x_t, which is taken at time t, may not depend on future observations of the stochastic process. This property is called *nonanticipativity*. Nonetheless, x_t may very well depend on the process $\xi_{0:t}$, i.e. on all information available up to time t, and as a consequence becomes a stochastic process itself. The optimization problem is called *linear* if the objective function as well as the constraints are linear.

Equation (4.33) can be reformulated in a second way that switches away from its focus on the nested structure of the problem. The reformulation stresses the view of actions $x_t := x_t(\xi_{0:t})$, $t = 1, ..., T$, as functions of the stochastic process ξ up to time t, which in conclusion denote a sequence of mappings. This sequence is called *policy* or *decision rule*. A policy is *feasible* if it fulfills all constraints for every possible realization of ξ, i.e.

$$x_t(\xi_{0:t}) \in \mathbb{X}_t(x_{t-1}(\xi_{0:t-1}), \xi_t), \qquad t = 2, ..., T, \text{ with prob. } 1.$$

Then, an equivalent formulation to Equation (4.33) is the following:

$$\min_{x_0, x_1, ..., x_T} \mathbb{E}\Big[C_0(\mathcal{S}(0), x_0, \xi_0) + C_1(\mathcal{S}(1), x_1(\xi_{0:1}), \xi_1) + ... + C_T(\mathcal{S}(T), x_T(\xi_{0:T}), \xi_T)\Big], \tag{4.35}$$

$$\text{s.th. } x_t \in \mathbb{X}_t \quad \text{feasible.}$$

Here, the optimization is performed over feasible policies, i.e. over functions instead of single actions. Consequently, if the process ξ does not have a finite number of observations, the optimization problem becomes infinite dimensional.

Starting from this general formulation, we introduce with dynamic and stochastic programming two classical stochastic optimization methods in more detail in the next subsections. Following those, we introduce with reinforcement learning a third, more recent method of how to tackle stochastic optimization problems. One property these three methods all have in common is that they either rely on the Bellman equation or can be translated to it. Therefore, we introduce it beforehand. Here, the following additional definition is necessary:

- **Value**: $V_t(s)$ is the value of being in state $\mathcal{S}(t) = s$ combined with the value of all following states under the assumption that only optimal decisions are taken from t onward.

The Bellman equation for the value of state $\mathcal{S}(t)$, that now contains the development of the exogenous information ξ, has the following form:

$$V_t(\mathcal{S}(t)) = \min_{x_t \in \mathbb{X}_t} \Big(C_t(\mathcal{S}(t), x_t) + \mathbb{E}\big[V_{t+1}(\mathcal{S}(t+1))\big] \Big), \tag{4.36}$$

with $V_{T+1}(\cdot) := 0$. The Bellman equation connects the costs incurred by an action taken at time t and conditioned on being in system state $\mathcal{S}(t)$ with the minimal expected costs

of the possible future costs, where the minimum is taken over the actions that are possible from that point onward. Thus, it defines a recursive relationship for the values of subsequent stages.

In order to make the different methods more tangible, we now briefly present a simple example that we will use for each of the three methods.

Example 4.33. In our example, there is a curry sausage vendor who stands in her stall every day selling curry sausages. She has her standard butcher who makes her a good price p_1 for his sausages. She orders x sausages from him, and then resells them to her customers for the price of p_2 per sausage with $p_2 > p_1$. When her customers' demand d is bigger than expected and they want to buy more sausages than she has prepared for the day, i.e. $x < d$, she can quickly buy the missing amount from a second butcher just on the other side of the street - his sausages are more expensive and sold to her at price p_3 with $p_3 > p_1$, but they help her to cover the demand that she did not account for. When on the other side, she bought more sausages than people came around, i.e. $x > d$, she can store the remaining sausages r in her fridge and, because they are high-quality, she can still sell them on the next day. Nonetheless, the storing leads to operating costs of p_4 per sausage. In total, her costs on day t are

$$p_1 \cdot x_t + p_3 \cdot [d_t - (x_t + r_{t-1})]_+ + p_4[(x_t + r_{t-1}) - d]_+ - p_2 \cdot d_t,$$

where d is the amount of sold sausages and x the amount of bought ones. We define cost instead of earning, because it is custom to minimize instead of maximize.

For the following paragraphs, we orientate ourselves on [54] in their explanations.

4.3.1 Dynamic programming

The approach of dynamic programming is mainly based on the work of Richard Bellman [2], who shifted the view from the analysis of the entire time horizon of an optimization problem to the analysis of recurring, single sub-problems in multi-period decision situations with a certain structure. This transformation was known earlier from physics, but until then it was not considered in the context of solving an optimization problem.

Bellman assumed that the underlying system in which optimization is performed can be described by its state.

Definition 4.34. A *state* is defined in [53] as "the minimally dimensioned function of history that is necessary and sufficient to compute the decision function, the transition function, and the contribution function".

In [53], a state is furthermore characterized by its physical resources, by its information, and by its assumptions. In addition, there is the possibility of influencing the state through an action.

Example 4.35. In the case of the curry sausage vendor, the physical resources would be the amount of sausages in the fridge as well as the additional amount bought for the day. The information state covers her knowledge about the prices. Finally, the assumptions state contains assumptions about her customers' demand, possibly its distribution based on empirical observations from the days before. She has the possibility to influence the state through her decision about how many sausages to buy from her first butcher.

Based on the collected information contained in a state, the state's value is to be ascertained. This value is usually not something that can be seen just by knowing a particular state - instead, the value of a state is calculated from its setup for the future: If there is little current profit to be made from a state, but there is the possibility of reaching other states in which much profit can be made, that state may have a higher value than a state in which there is a great deal of profit to be made, but only states with little profit can be reached afterwards. Thus, the value of a state contains the prospect of how much profit can be made in total. These future profits are of course weighted with the probability of their occurrence. Consequently, the value of a state is equal to the optimal solution of the entire optimization problem that is conditioned on starting in that state and is shortened to the time interval from this decision epoch to the last one. Thus, knowing the value of the starting state means knowing the solution to the original optimization problem.

Example 4.36. Again based on the curry sausage vendor, we can make the value of a state with the following example more tangible: Let us assume that her main butcher, that sells his sausages for a price p_1, will go on a vacation for a few days. Then, buying today an amount of sausages that exceeds today's demand in order to have enough to sell on the next days might have a higher value than just buying enough for today's demand, even though the cost incurred today would be higher. The reason for that is, that on the days to come, the curry sausage vendor does not have to buy her sausages at the higher price p_2 from the second butcher but can use the ones she prospectively bought beforehand and stored in her fridge.

Following this line of thought, only the value of the last stage can be calculated without calculating further values of other stages - and when that is known, the value of the stage before it can be calculated. This leads to a recursive computing structure, where, starting with the last stage, the values of all states can be calculated. This is where dynamic programming finds its place: Following [7], dynamic programming is usually only applied when a problem has *optimal substructure* and *overlapping subproblems*. Optimal substructure is given when an optimal solution to a subproblem of the original problem will also be contained in the optimal solution of the original problem itself. Furthermore, the solutions to the subproblems must be independent from each other for optimal substructure to apply. [7] formulate it such that the subproblems cannot share resources in their solutions: A solution to one subproblem in combination with the solution of a second subproblem must be a solution for the problem containing the two as well. Now, turning to overlapping subproblems, this occurs when a recursive algorithm would solve the same problem over and over again while trying to solve the original

problem. A typical example for that is the calculation of Fibonacchi numbers, where for example for the computation of the 5th Fibonacchi number, the Fibonacchi numbers of 4 and 3 are needed, and again for the calculation of the 4th Fibonacchi number, those for 3 and 2 are needed. The same calculations reappear and can be reused is stored in a smart fashion. Consequently, DP usually takes advantage of overlapping subproblems by storing once-calculated results and reusing them if necessary.

We write down this approach in a formalized way, sticking to the notation introduced before and following [54]. Again, we define:

- **State**: $\mathcal{S}(t)$ be the state of the system at time t stemming from the set of states $\mathbb{S}_t$.
- **Action**: x_t be a feasible action at decision epoch t stemming from the action set $\mathbb{X}_t$.
- **Cost**: $C_t(\mathcal{S}(t), x_t)$ be the cost of taking action x in state $\mathcal{S}(t)$ at time t.
- **Transition probability**: $p(s' \mid s, x_t)$ be the probability to move to state $\mathcal{S}(t+1) = s'$ when being in state $\mathcal{S}(t) = s$ and using action x_t.
- **Value**: $V_t(s)$ be the value of being in state $\mathcal{S}(t) = s$ combined with the value of all following states under the assumption that only optimal decisions are taken from that decision epoch onward.
- **Decision epoch**: $t = 1, ..., T$ be the decision epochs, i.e. the time points at which actions can be taken.

The corresponding optimal policy formulation, ranging over all decisions x_t, $t = 0, ..., T$, is consequently given by

$$\pi(s) = \underset{x_0 \in \mathbb{X}_0, ..., x_T \in \mathbb{X}_T}{\arg\min} \Big(C_0(\mathcal{S}(0), x_0) + \mathbb{E}\big[V_1(\mathcal{S}(1))\big] \Big). \tag{4.37}$$

With $\mathbb{S}_t$ being finite for all t, this can be rewritten to

$$\begin{aligned} V_t(\mathcal{S}(t)) &= \min_{x_t \in \mathbb{X}_t} \Big(C_t(\mathcal{S}(t), x_t) + \sum_{s' \in \mathbb{S}_{t+1}} p(s' \mid \mathcal{S}(t), x_t) V_{t+1}(s') \Big), \\ \pi(s) &= \underset{x_0 \in \mathbb{X}_0, ..., x_T \in \mathbb{X}_T}{\arg\min} \Big(C_0(\mathcal{S}(0), x_0) + \sum_{s' \in \mathbb{S}_1} p(s' \mid \mathcal{S}(0), x_1) V_1(s') \Big). \end{aligned} \tag{4.38}$$

The Bellman equation itself is not restricted to applications where the underlying stochastic process is Markovian, i.e. stagewise independent, and the optimization problem has optimal substructure or overlapping subproblems. Nonetheless, solving these equations in closed form can be very difficult or even impossible for generally distributed underlying stochastic processes. Even without closed form, the computation of these equations for non-Markovian processes can quickly exceed feasible limits. Consequently, numerical

approximations or simplifying assumptions like the stochastic process being Markovian are therefore resorted to quite often. Nonetheless, if that is not given, then a search over the whole state-action-space needs to be carried out - that is possible using dynamic programing equations, but does not bear advantages regarding computation time in comparison to regular full state-action-space searches any longer.

Remark 4.37. Typically, the dynamic programming community tries to model the states in such a way that they are Markovian. Citing [54], they even state "All properly modeled dynamic programs are Markovian". A typical solution for processes with history-dependent variables is to enlarge the state space by the history. Nonetheless, depending on the problem at hand, it might be easier to use another solution method like stochastic programming, instead of massively enlarging the state variable such that it contains all relevant information.

Remark 4.38. It is important to notice that due to its structure, dynamic programming can integrate differing transition probabilities from one state to another conditioned on the decision that was taken. This will not be the case for stochastic programming, where the scenarios are prepared beforehand and are independent from the taken decisions.

4.3.2 Stochastic programming

We now introduce stochastic programming and follow [42] and [4] hereafter: For stochastic programming, again, the goal is to perform optimization under uncertainty. Its origin lies with [8] roughly 80 years ago, where first answers on how to incorporate uncertain demands into an optimization problem were discussed. The standard approach is to approximate the parameters considered as uncertain by their (empirical) distribution, to choose scenarios from this distribution, and to optimize on them as basis.

Remark 4.39. The intrinsic assumption that at least an empirical distribution is known is often justified in practice, since past data of the same or a similar problem can be used. Furthermore, this approach rules out the possibility to redefine scenarios based on decisions that were taken; consequently, the underlying stochastic processes should be independent of the decision to be taken.

The typical problem in stochastic programming is the two-stage recourse problem, which assumes that a decision must be made in the here and now under the impression of an uncertain future, and a second decision - the recourse decision - can be made when whatever future parameter the problem depends on has materialized.

Example 4.40. Returning to the example of the curry sausage vendor, a possible decision she could face is that of whether she should invest in a bigger sign for her stall to increase her visibility and thus possibly increase the demand. This decision is about an action that is performed only once. Therefore, it can be formulated as a two-stage problem, where in the first stage the decision about which sign to buy is taken. In the second stage, knowledge about actual increases of customer demand builds and a recourse decision becomes possible, as for example to increase the amount of sausages that are ordered daily from her standard butcher. As is immediately visible, the first part

of the problem contains a one-time decision. In comparison, the second part actually contains an intrinsic multi-periodicity, which should also be taken into account in the modeling - not necessarily when deciding over the investment strategy regarding the new sign where the future demand patters need not be detailed out, but afterwards for the everyday order.

We formalize the approach in the following. The two-stage formulation of stochastic programming is in its linear form for $x \in \mathbb{R}^m$ given by

$$\begin{aligned} &\min_{x \in \mathbb{X}} \ \Big(C(x, \xi_0) + \mathbb{E}[Q(x, \xi_1)] \Big), \\ &\text{s.th. } \ Ax = b, \ \ x \geqslant 0. \end{aligned} \tag{4.39}$$

Here, A is a matrix of size $n \times m$ and b a vector of size n. The equation system defines the n constraints of the problem. $Q(x, \xi_1)$ is the optimal value of the second-stage problem

$$\begin{aligned} &\min_{\tilde{x} \in \tilde{\mathbb{X}}} \ \tilde{C}(\tilde{x}, \xi_1), \\ &\text{s.th. } \ T(\xi_1)x + W\tilde{x}(\xi_1) = h(\xi_1), \ \ \tilde{x} > 0, \end{aligned} \tag{4.40}$$

where the constraints now partly depend on the realization of ξ_1 and represent a combination of the first-stage and the second-stage decision. W is also called *recourse matrix*. Here, it is assumed to be fix and non-dependent of the realization of the stochastic process. Equation (4.39) is equivalent to the *deterministic equivalent program*. It is called this way because all stochasticity is contained in the constraints and therefore it can just be solved like a deterministic program. The main computational work lies with the computation of $\mathbb{E}[Q(x, \xi_1)]$, see [4]; consequently, the higher the amount of scenarios, the bigger the constraint matrices and the longer the computation of this expectation.

Remark 4.41. Equation (4.39) works for ξ being a discrete random variable, but also if it is a continuously distributed random variable. Nonetheless, fast computation is only possible if the continuous random variable has specific properties, or is sufficiently small in dimension such that numerical integration is computationally feasible, see [4]. Consequently, the practical applicability of stochastic programming for continuous variables has its limits. Knowledge about it for discrete variables is important all the more, as the possibility exists to approximate a continuous random variable by a finite - and thus discrete - number of realizations. This is where the methods discussed in Section 4.2 come into play.

The theoretical setting of two-stage problems can be extended to several periods. Concentrating on the case of a time-discrete stochastic process, a finite horizon framework and fixed recourse, its multi-stage version is based on scenarios $w \in \Omega_t$ that represent possible developments of the random influences. Constraints are omitted for now, but will be explained later. A multistage stochastic programming model beginning in t and

looking forward up to time $t + H$ is given by

$$\min_{x_t \in \mathbb{X}_t} C_t(x_t, \xi_t) + \mathbb{E}\Bigg[\min_{x_{t+1} \in \mathbb{X}_{t+1}(x_t, \xi_{t+1})} C_{t+1}(x_{t+1}, \xi_{t+1}) + \mathbb{E}\Big[\dots + \mathbb{E}\Big[\min_{x_{t+H} \in \mathbb{X}_{t+H}(x_{t+H-1}, \xi_{t+H})} C_{t+H}(x_{t+H}, \xi_{t+H})\Big]\Big]\Bigg]. \tag{4.41}$$

Consequently, we optimize over the expected costs for all decision time points up to time $t+H$ in order to make the best decision for the here-and-now. An example for a solution method is e.g. Bender's decomposition [56], if applicable. The actual computation of the multistage model is often computationally intractable though, if the number of scenarios is adequately high [54].

Remark 4.42. Equation (4.41) can actually be rewritten in a recursive manner, such that the costs of future periods are, like with the Bellman equation, contained in a second-stage value function. The corresponding format is usually called dynamic programming equation, see e.g. [50], [4].

Because of the tractability issues of many multistage programs, an approximation to Equation (4.41) is often used in practice [54]: A reduction of computation time is reached through the two-stage approximation of the multistage model, which is given in the following:

$$\min_{x_t, (x_{t'}(w), t < t' \leqslant t+H), \forall w \in \Omega_t} \left(c_t x_t + \sum_{w \in \Omega_t} p(w) \sum_{t'=t+1}^{t+H} c_{t'}(w) x_{t'}(w) \right). \tag{4.42}$$

Again, we assume that at time t we make a decision based on information known up to t. The big difference, however, is that after time t a scenario w occurs, which contains all the information up to time $t+H$. Consequently, we can optimally make all decisions from $t+1$ to $t+H$ when we arrive at $t+1$. Due to the simplicity of this approach combined with its prospect to nonetheless integrate assumptions about future behavior of uncertain parameters, and because it is the approach mainly adopted in practice according to [54], we choose to work with this in the following.

Remark 4.43. In the case of an originally continuous stochastic process that has been approximated by a finite number of scenarios, usually only the first from all found decisions over the whole time horizon is actually implemented. This is due to the fact that, no matter which observation realizes itself, the probability that one of the scenarios from the scenario tree equals it is zero. Consequently, the optimization for the next period must be carried out from the beginning.

Remark 4.44. The computation of the original multi-stage model is simplified when the interactions between stages are sufficiently weak. Multi-stage programs with this property are called *block separable*, see [4], and exploiting it can reduce the amount of computation work immensely.

4.3.3 Reinforcement Learning

Both stochastic programming and dynamic programming are classical methods for solving optimization problems with random elements. Reinforcement Learning (RL), on the other hand, is a machine learning method that only gained importance in the last few decades and is subject of current research. Nonetheless, its core ideas are based on dynamic programming and the Bellman equation, so similarities do exist between both methods. For an extensive introduction to RL, we refer to [58] and base our explanations on it from here on.

RL is referred to as a computational approach to learning by interacting with a system and its possibilities. Specifically, it does not solve an optimization problem like classical methods, but instead tries to find the solution through educated "trial-and-error": It considers a learning agent that can perform certain actions in the given environment and gains knowledge about the interplay of actions and environment states from these interactions. Depending on the state of its environment, the agent receives a reward for the action it chooses and can thus refine its knowledge about this interplay. The agent's overall goal is to maximize these accumulated rewards over the course of the program. Consequently, RL tries to reach the same results as classical solving techniques, but with less computational effort and less assumptions to the optimization problem, see [58]. For that, it relies on the same idea as DP, namely on the Bellman equation. We go into detail regarding this statement after introducing relevant assumptions and definitions.

RL is commonly based on the following hypothesis:

Definition 4.45. The *reward hypothesis* states that all goals can be expressed through maximizing an expected reward.

A key component of the Bellman equation and thus DP as well as RL is the idea to use a value function that depends on this reward in order to determine which policy to follow in the course of the optimization problem. In RL, the value function of a system state – corresponding to its definition in the other contexts – contains the expected accumulated reward under the optimal policy from that system state onward. In order to formulate the RL setting, we add the following definitions to the one made beforehand:

- **Environment state**: At every point in time, the environment state S_t^e is the environment's representation of itself, on whose basis the environment reacts to actions and moves from one state to another. This representation does not have to be known to the agent, and usually is not known.

- **Agent state**: The agent state S_t^a is the agent's representation of the system and the basis of its decisions.

- **Reward signal**: A reward signal $r(s, x) \in \mathcal{R} \subset \mathbb{R}$ provides the agent with a numerical, scalar feedback to the chosen action x in system state s in the short term.

- **Model**: A model of an environment provides the agent with a way of estimating how the environment will behave as a result of a particular action. It is used for planning, i.e. to estimate in advance which possible future situations will result from which actions. Methods that make use of models are also referred to as model-based methods. Techniques that are not model-driven are called model-free.

Remark 4.46. The classical RL framework is that of finite Markov decision processes, so the optimal decision must not be dependent on past states. Some extensions to the classical RL methods do yield the possibility to broaden this framework, but as this one fits our thesis well, we do not go into detail here.

In our case, the agent state and the environment state will be identical to each other, and are again represented by $\mathcal{S}(t)$. At every decision epoch $t = 0, .., T$, the agent observes the system state $\mathcal{S}(t) = s$, then decides for an action that is conditioned on the system state, i.e. $x_t \in \mathbb{X}(s)$, receives a reward $r_{t+1} \in \mathcal{R}$ and then, the next state is revealed based on a transition matrix with prespecified transition probabilities $p(s' \mid s, x_t)$ for all possible following system states. Consequently, the resulting trajectories are of the form $\mathcal{S}(0), x_0, r_1, \mathcal{S}(1), x_1, ..., r_T$, and the history of observations, actions and rewards up to time t is respectively defined as

$$H_t = (\mathcal{S}(0), x_0, r_1, \mathcal{S}(1), x_1, ..., r_t).$$

Bases on this definition of history, the state can be defined as a function of history, i.e. $\mathcal{S}(t) = f(H_t)$.

Example 4.47. In our example of the curry sausage vendor, the environment contains information about the amount of stored sausages in the fridge and the prices of the two butchers. The reward signal of one day is obviously the negative cost of that day. Her actions are as before given through how many sausages she buys from her main butcher. The model contains her estimate of the customers' demand distribution on coming days.

Now, the agent walks through the environment and collects these trajectories. The knowledge that is gained by this is supposed to be translated to a value and a policy function, just as was the case for DP. To define them again, their optimal values are given through

$$V_t^*(s) = \max_{x_t \in \mathbb{X}_t} \mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t} r \mid \mathcal{S}(t) = s \right] \quad \text{for all } s \in \mathbb{S}_t, \tag{4.43}$$

$$q_t^*(s, x) = \max_{x_t \in \mathbb{X}_t} \mathbb{E}\left[\sum_{k=0}^{T} \gamma^{k-t} r \mid \mathcal{S}(t) = s, x_t = x \right], \tag{4.44}$$

where the latter is called *optimal action-value function* and $\gamma \in (0, 1)$ represents a discount factor. Here, $q_t^*(s, x) = \mathbb{E}[r_{t+1} + \gamma V_t^*(\mathcal{S}(t+1)) \mid \mathcal{S}(t) = s, x_t = x]$ translates to the Bellman equation with discount factor for future rewards. Now, as RL does not optimize

in a classical sense, these equations cannot be computed in a classical sense either. The procedure that is followed here is that of either *policy iteration* or *value iteration*. Both start with guesses of the optimal policy or optimal value function, respectively, and from there contain a sequence of evaluation and improvement steps that let these guesses converge to the actual optimal values. We take a closer look at value iteration: This, as stated in [58], is actually nothing else but using the Bellman equation as updating rule for the value function. It is given through the following Algorithm 10, where we assume that the state space contains all possible states independent of t and again follow [58].

Result: Optimal value function $V^*(s)$ and optimal policy $\pi^*(s)$

begin
  Input: Threshold τ.
  Initialize the value function $V(s)$ arbitrarily for all states s.
  Set $d = \tau + 1$.
  while $d > \tau$ **do**
    Set $d = 0$.
    foreach *State* s **do**
      Save old value $\tilde{V} = V(s)$.
      Calculate the new value from maximizing over all actions x in state s:
      $V(s) = \max_x \sum_{s'} p(s'|s,x)\,(r(s,x) + \gamma V(s'))$.
      Calcute the change in value $d = \max\{d, |V(s) - \tilde{V}|\}$.
    end
  end
  Return the converged value function and the optimal policy $\pi^*(s)$, which is found by
  $\pi(s) = \arg\max_x \sum_{s'} p(s'|s,x)\,(r(s,x) + \gamma V(s'))$.
  return $V^*(s)$, $\pi^*(s)$.
end

Algorithm 10: Value Iteration

The value function and policy of this algorithm converge to the their optimal values without actually optimizing the whole problem. Instead, each single step is optimized on its own. This method saves from having to implement sometimes tedious constraint matrices, but has its own caveats: As for any machine learning method, there are certain hyper parameters present, e.g. τ, γ, a learning rate and others that need hyper parameter optimization, that does take some time on its own. Nonetheless, this method yields the opportunity to solve difficult multi-stage optimization problems.

4.4 Case Study on the German day ahead market

Having now described three fundamental solution approaches to our optimization problem, we will devote this section to the comparison of their performance and suitability for the practical application. To remind the reader, we concentrate on the time interval

from the years 2015 to 2020 based on the analysis from Section 2.1. The first subsection features the setting in which the stochastic optimization will take place. This is followed by the scenario tree generation for the model in Equation (3.6) that is calibrated on the year 2020. In order to test the tree generation as well as the tree generation heuristic for different initial situations, we have selected both a winter and a summer period on which to run the overall procedure. The winter period starts at the 1st of February, and the summer period on the 1st of August. Finally, the presented optimization methods are compared on the generated trees in a rolling horizon setting of one week each.

4.4.1 Setting

The general setting that we are looking at is one of an electricity trader owning a battery. The trader does, at this moment, not have the means to produce for or take electricity from the battery, but can only buy and sell at the market. To link this setting to reality, one could imagine a cold storage that needs temperatures to be below a certain threshold at all times, or a constantly running water pump of a slowly flooded salt pit that needs to keep the water level below a threshold. In the second example, the water is constantly running back into the pit tunnels. The water pump operator has the option to let the pump run at a base level that equals outflow to inflow and leads to a fixed water level at a certain height at all times. Then again, the option exists to pump out more water than is flowing in when electricity is cheap, or to let the water flood in until the maximum is reached when it is expensive. Both scenarios could follow one after the other, with breaks of base pumping in between where the water level remains constant – then, again, inflow and outflow cancel each other out. These examples both present processes that work like batteries, as the cold storage as well as the water pump have a range in which it is possible for them to work more or work less. Energy efficiency is something that could to be taken into account for the cold storage, but we will not include it in our optimization.

Based on that setting, we assume that the considered battery has a maximal capacity b_{max} MW as well as a minimal capacity b_{min} MW, where without loss of generality we can set $b_{\text{min}} = 0$. Furthermore, the battery's level $b(t)$ is for all t in $[b_{\text{min}}, b_{\text{max}}]$, and the speed at which the battery can at maximum be filled or emptied is b_a MW/h.

Example 4.48. We demonstrate the similarity of a water pumped salt pit to a battery based on these definitions. Figure 4.23 contains water levels over time as well as electricity used in a salt pit. We notice three intervals where the base pumping speed is used and inflow equals outflow, i.e. the time intervals $[1, 2], [4, 5]$ and $[8, 9]$. For those three intervals, the water levels remain constant. In the example, we assumed the water levels to be contained in $I = [0\text{cm}, 15\text{cm}]$. Keeping the water level fixed needs 20 MW/h, and its maximum pumping speed yields 5cm an hour at a cost of 40 MW/h. In combination with I, this leads us to $b_{\text{min}} = 0$ MW/h, $b_{\text{max}} = 120$ MW/h and $b_a = 40$ MW/h.

We do not include any necessity to have a certain amount of electricity stored at a specified deadline into the model. Therefore, the trader's only goal is to buy when prices are low and sell when prices are high, i.e. to minimize the incurred costs, all while staying in the capacity range of the battery. As specified before, in order to include weekly seasonalities of electricity prices, the model has a look ahead period of seven days and contains hourly prices. Formulating this objective and its constraints in an optimization problem yields

$$\begin{aligned} \min_{v \in \mathcal{V}} \quad & \sum_{t=1}^{7*24} S(t)v(t) && (4.45) \\ \text{s.th.} \quad & b_{\text{min}} \leqslant b(0) + \sum_{t=1}^{M} v(t) \leqslant b_{\text{max}}, && M = 1, 2, ..., 7 * 24, \\ & |v(t)| \leqslant b_a && t = 1, 2, ..., 7 * 24, \end{aligned}$$

where $S(t)$ is the spot price quoted and $v(t)$ the volume traded on the day ahead market at time t. The volume strategy v needs to be contained in the set of feasible strategies $\mathcal{V}$, i.e. it must fulfill all given constraints at all times. The traded volume is positive when electricity is bought and negative when it is sold. Formulated like this, Equation (4.45) is a regular linear problem that can be solved with standard methods.

Remark 4.49. We do assume that the trades we place on the market do not shift prices in any way. This is reasonable, as through the blind auction format, other people cannot see the prices that were offered and therefore cannot react to them.

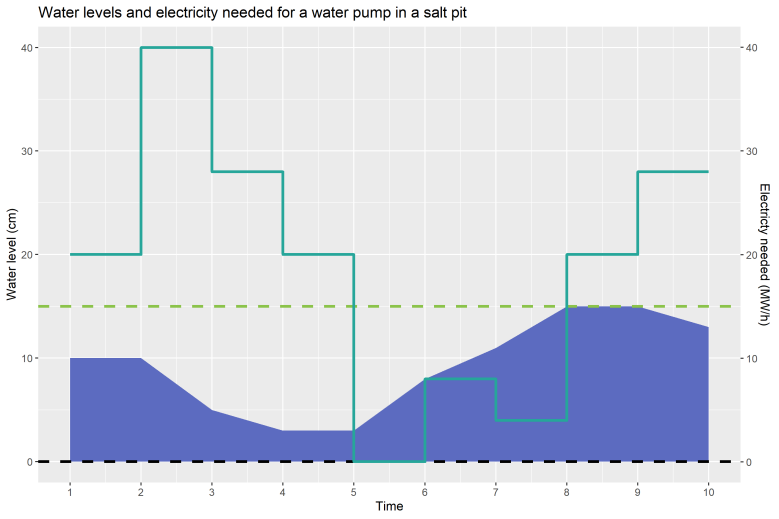

Figure 4.23: The water level in cm over time for a salt pit, represented by the blue area. Its maxmium and minimum height are depicted by the green and black dashed lines, respectively. Finally, the turquoise line represents the amount of electricity needed to reach the specified water levels.

Of course, the setting in Equation (4.45) is a deterministic one, where all prices $S(t), t = 1, ..., 7 * 24$, would have to be known beforehand. As this will never be the case in practice, the uncertainty of price forecasts should be taken into account when deciding for a volume strategy. The part of the uncertainty that is inherent in the realization of volumes on the market is leveraged by a simple mechanism: Since in the day ahead auction the market clearing price results from the intersection of supply and demand, see Section 2.1, which is ultimately paid by all or received by all whose bids are awarded, the maximum price can always be bid as the price bid. This means that the bidder's own bid and thus the submitted volume is always awarded, but the price to be paid is still only the market clearing price. This would not hold true if there was more total demand than there is total supply or vice versa, but this is something which – to our knowledge – basically never occurs, so we do not take it into account. Consequently, uncertainty in realized volumes can be neglected. Of course, this does not apply to price uncertainty, which is why Equation (4.45) has to be adapted to fit the mathematical area of stochastic optimization. As a result, we first look at the generation of scenario trees for this specific setting and then reformulate (4.45) to make it accessible to stochastic optimization methods.

Remark 4.50. Even though we do not require the battery to meet a specific demand at a certain point in time, we do require the optimization to plan in such a way that the final battery level after the seven days meets some predefined value $b(24)$. This keeps the strategies from planning to sell all remaining electricity at the seventh day.

4.4.2 Scenario tree for the electricity market model

In order to generate scenario trees for the German electricity market, we apply the presented algorithms from Subsection 4.2.4 to the introduced electricity market model. Our goal is to reach a tree with an acceptable amount of nodes that provides a good fit to the original stochastic process. Following the presented procedure, we first calculate trees with many scenarios. As elaborated on beforehand, a look-ahead of a week seems sensible in a periodic market like electricity. Furthermore, it is a general consensus in stochastic optimization for market prices that a reasonable approach to pricing contains a more detailed view of the near future and a coarser view of the more distant future. As a consequence, we introduce a starting tree structure of $(1, 8, 4, 3, 2, 2, 2)$. This structure is chosen based on the fact that for real applications, such a tree will only be used to find a decision for the next day, and is then re-calibrated based on the new realized values. As the amount of scenarios is given by the accumulated product of the node structure, this results in a total of 768 considered scenarios. Furthermore, we simulate $N = 1,000,000$ paths for each of the factors and apply the nested clustering algorithm, i.e. Algorithm 2, on them. This results in the trees given in Figure 4.24 for the winter period and Figure 4.25 for the summer period with the mentioned structure for the whole model in Equation (4.4).

The computation of these trees is possible, but takes some time, as Algorithm 2 splits the one million paths repeatedly to find good cluster means. To test whether the heuristic could support in terms of fast computation, we set $T_s = 3$ and determine a structure

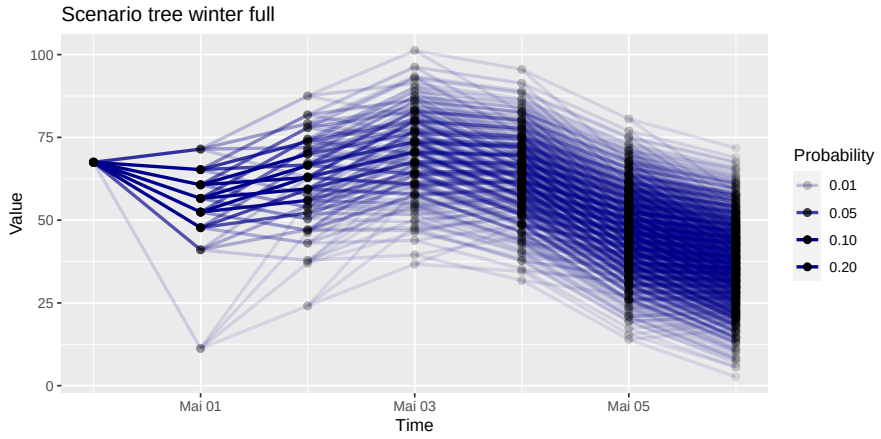

Figure 4.24: A scenario tree based on the model in Equation (4.4) calculated with Algorithm 2 and $1,000,000$ trajectories. The tree is calculated for the 1st of February.

of $(1, 8, 4)$. The results from the computation of the heuristic method presented in Algorithm 7 to test whether it yields comparable results with less computational costs are found in Tables 4.26 and 4.27 in detail for the summer period and the l_1-distance. We find that with a distance of 12.64, the worst approximation stems from putting all emphasis on the jump Ornstein-Uhlenbeck process, i.e. factor 3 from Equation (4.7). In comparison to that, the closest tree is produced by strengthening the second factor from Equation (4.6), i.e. the regular Ornstein-Uhlenbeck process, by putting all emphasis on it with a structure of $(1, 8, 4)$. This setting yields a distance of 1.01. This is mirrored by the results from the l_2-distance, where still the structure $(1, 8, 4)$ for the regular Ornstein-Uhlenbeck process yields the smallest nested distance with 3.06.

For the winter period, results are identical - again, the smallest distance of 1.13 for the l_1-distance and 3.05 for the l_2-distance are reached when putting all path flexibility into the second factor. Consequently, we calculate the added trees for all seven stages based on the results of the heuristic for both summer and winter with a structure of

$$
\begin{aligned}
&(1, 1, 1, 1, 1, 1, 1) && \text{for the GBM, i.e. factor 1,}\\
&(1, 8, 4, 3, 2, 2, 2) && \text{for the GOU, i.e. factor 2,}\\
&(1, 1, 1, 1, 1, 1, 1) && \text{for the JOU, i.e. factor 3.}
\end{aligned}
$$

The resulting added trees are found in Figure 4.28 for the winter period and in Figure 4.29 for the summer period. Computation times are given in Table 4.30 for distances with $r = 1$ and l_1-norm, respectively, and in Table 4.31 for distances with $r = 2$ and l_2 norm. It is visible that the computation of the tree that is based on the heuristic from Algorithm 7 and then added is much faster in comparison to the original method. The distances incurred between the trees stemming from the original method and their corresponding added counterparts are given in Table 4.32. It is visible that the incurred error on this level of the trees is not too large. Because the usage of the l_1-norm for the

Structure Factor GBM	Structure Factor OU	Structure Factor JOU	Nested Distance
(1, 1, 1)	(1, 1, 1)	(1, 8, 4)	12.64
(1, 1, 1)	(1, 1, 2)	(1, 8, 2)	11.09
(1, 1, 1)	(1, 1, 4)	(1, 8, 1)	10.35
(1, 1, 2)	(1, 1, 1)	(1, 8, 2)	12.43
(1, 1, 2)	(1, 1, 2)	(1, 8, 1)	10.84
(1, 1, 4)	(1, 1, 1)	(1, 8, 1)	12.28
(1, 1, 1)	(1, 2, 1)	(1, 4, 4)	9.15
(1, 1, 1)	(1, 2, 2)	(1, 4, 2)	7.14
(1, 1, 1)	(1, 2, 4)	(1, 4, 1)	5.99
(1, 1, 2)	(1, 2, 1)	(1, 4, 2)	8.85
(1, 1, 2)	(1, 2, 2)	(1, 4, 1)	6.95
(1, 1, 4)	(1, 2, 1)	(1, 4, 1)	8.73
(1, 1, 1)	(1, 4, 1)	(1, 2, 4)	7.52
(1, 1, 1)	(1, 4, 2)	(1, 2, 2)	5.31
(1, 1, 1)	(1, 4, 4)	(1, 2, 1)	3.57
(1, 1, 2)	(1, 4, 1)	(1, 2, 2)	7.16
(1, 1, 2)	(1, 4, 2)	(1, 2, 1)	4.92
(1, 1, 4)	(1, 4, 1)	(1, 2, 1)	7.07
(1, 1, 1)	(1, 8, 1)	(1, 1, 4)	5.81
(1, 1, 1)	(1, 8, 2)	(1, 1, 2)	3.55
(1, 1, 1)	(1, 8, 4)	(1, 1, 1)	**1.01**
(1, 1, 2)	(1, 8, 1)	(1, 1, 2)	5.42
(1, 1, 2)	(1, 8, 2)	(1, 1, 1)	3.12
(1, 1, 4)	(1, 8, 1)	(1, 1, 1)	5.38
(1, 2, 1)	(1, 1, 1)	(1, 4, 4)	12.02
(1, 2, 1)	(1, 1, 2)	(1, 4, 2)	10.44
(1, 2, 1)	(1, 1, 4)	(1, 4, 1)	9.53
(1, 2, 2)	(1, 1, 1)	(1, 4, 2)	11.80
(1, 2, 2)	(1, 1, 2)	(1, 4, 1)	10.18
(1, 2, 4)	(1, 1, 1)	(1, 4, 1)	11.65
(1, 2, 1)	(1, 2, 1)	(1, 2, 4)	8.76
(1, 2, 1)	(1, 2, 2)	(1, 2, 2)	6.75
(1, 2, 1)	(1, 2, 4)	(1, 2, 1)	5.42
(1, 2, 2)	(1, 2, 1)	(1, 2, 2)	8.45
(1, 2, 2)	(1, 2, 2)	(1, 2, 1)	6.47
(1, 2, 4)	(1, 2, 1)	(1, 2, 1)	8.32
(1, 2, 1)	(1, 4, 1)	(1, 1, 4)	7.00
(1, 2, 1)	(1, 4, 2)	(1, 1, 2)	4.81
(1, 2, 1)	(1, 4, 4)	(1, 1, 1)	2.81
(1, 2, 2)	(1, 4, 1)	(1, 1, 2)	6.65
(1, 2, 2)	(1, 4, 2)	(1, 1, 1)	4.40
(1, 2, 4)	(1, 4, 1)	(1, 1, 1)	6.57

Table 4.26: Distances between added trees and original tree for the summer period and with Wasserstein distance of order 1, both with a structure of (1,8,4); the lowest value is printed in bold-face.

Structure Factor GBM	Structure Factor OU	Structure Factor JOU	Nested Distance
(1, 4, 1)	(1, 1, 1)	(1, 2, 4)	11.90
(1, 4, 1)	(1, 1, 2)	(1, 2, 2)	10.30
(1, 4, 1)	(1, 1, 4)	(1, 2, 1)	9.46
(1, 4, 2)	(1, 1, 1)	(1, 2, 2)	11.61
(1, 4, 2)	(1, 1, 2)	(1, 2, 1)	10.07
(1, 4, 4)	(1, 1, 1)	(1, 2, 1)	11.55
(1, 4, 1)	(1, 2, 1)	(1, 1, 4)	8.52
(1, 4, 1)	(1, 2, 2)	(1, 1, 2)	6.54
(1, 4, 1)	(1, 2, 4)	(1, 1, 1)	5.15
(1, 4, 2)	(1, 2, 1)	(1, 1, 2)	8.22
(1, 4, 2)	(1, 2, 2)	(1, 1, 1)	6.22
(1, 4, 4)	(1, 2, 1)	(1, 1, 1)	8.12
(1, 8, 1)	(1, 1, 1)	(1, 1, 4)	11.81
(1, 8, 1)	(1, 1, 2)	(1, 1, 2)	10.25
(1, 8, 1)	(1, 1, 4)	(1, 1, 1)	9.44
(1, 8, 2)	(1, 1, 1)	(1, 1, 2)	11.61
(1, 8, 2)	(1, 1, 2)	(1, 1, 1)	10.01
(1, 8, 4)	(1, 1, 1)	(1, 1, 1)	11.50

Table 4.27: Continued: Distances between product trees and original tree for the summer period and with Wasserstein distance of order 1, both with a structure of (1,8,4).

Full model tree s	Full model tree w	Added tree s	Added tree w
1,358s	1,045s	0.99s	1.01s

Table 4.30: Computation times of the big trees for summer (s) and winter (w) period, both for the full model as well as the added model based on the results from the heuristic in Algorithm 7. The l_1 norm is used for distance calculation.

Full model tree s	Full model tree w	Added tree s	Added tree w
1,399s	1,251s	0.46s	0.46s

Table 4.31: Computation times of the big trees for summer (s) and winter (w) period, both for the full model as well as the added model based on the results from the heuristic in Algorithm 7. The l_2 norm is used for distance calculation.

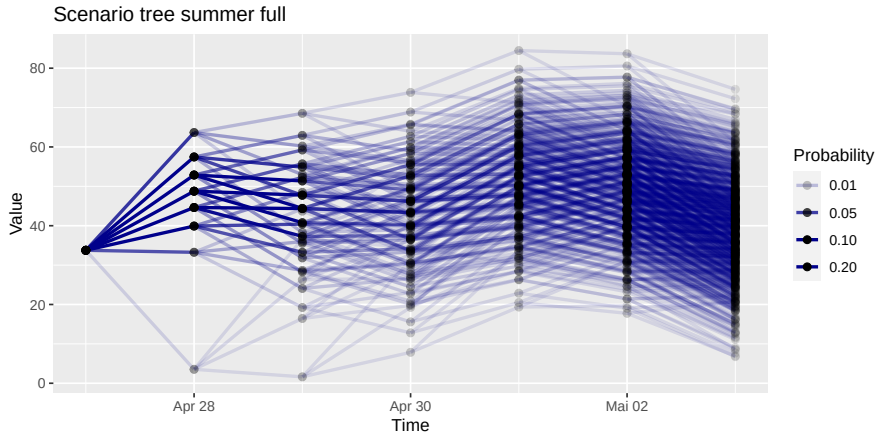

Figure 4.25: A scenario tree based on the model in Equation (4.4) calculated with Algorithm 2 and $1,000,000$ trajectories. The tree is calculated for the 1st of August.

Season	Treenorm	Wasserstein norm	Distance
Summer	1	1	7
Summer	2	2	6.55
Winter	1	1	6.91
Winter	2	2	6.49
Summer	1	2	3.59
Summer	2	1	13.91
Winter	1	2	3.59
Winter	2	1	13.77

Table 4.32: Distances between added trees and original trees for different norms.

tree generation yields smaller distances between the resulting trees no matter whether we use the nested distance of order 1 or the nested distance of order 2, we decide to continue with the trees generated from it.

In the second and third step, all four trees are reduced with Algorithm 5 and then finally improved with Algorithm 6. The resulting trees for the original method are given in Figure 4.33 for the winter and in Figure 4.34 for the summer period.

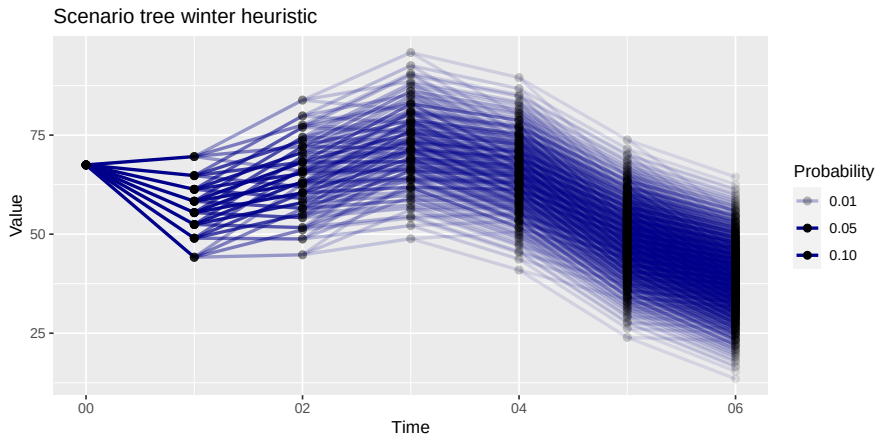

Figure 4.28: A scenario tree based on the model in Equation (4.4) based on the results from Algorithm 7 and using Algorithms 8 and 9. The tree is calculated for the 1st of February.

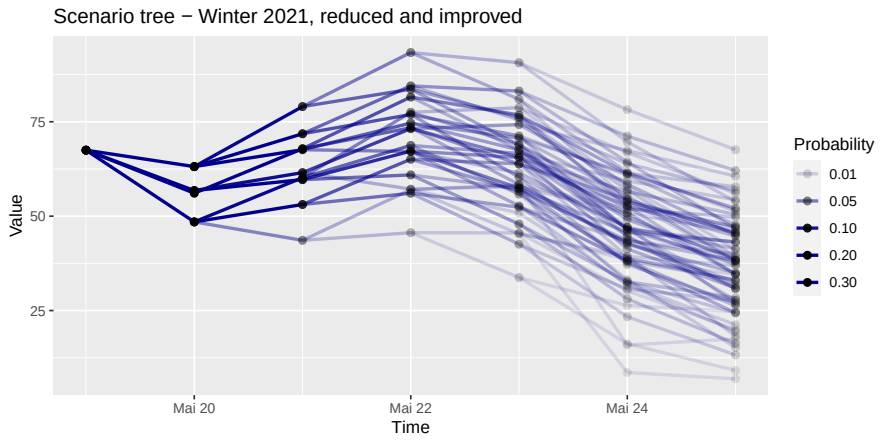

Figure 4.33: The reduced and improved version of the tree stemming from the original method for the winter period from Figure 4.24.

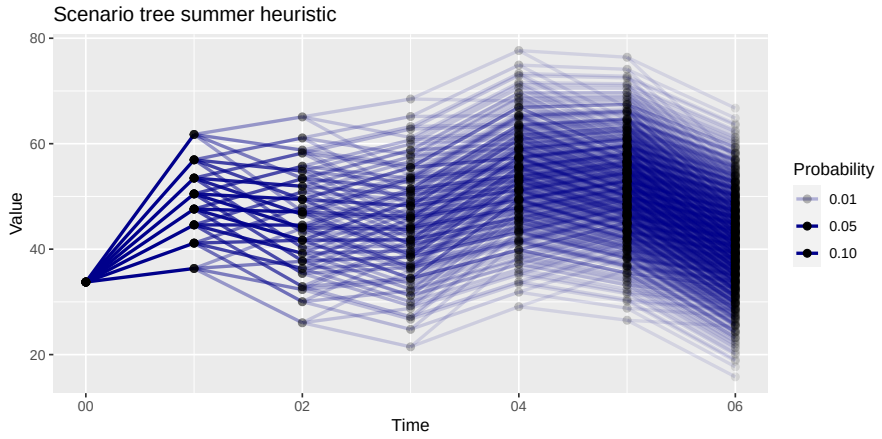

Figure 4.29: A scenario tree based on the model in Equation (4.4) based on the results from Algorithm 7 and using Algorithms 8 and 9. The tree is calculated for the 1st of August.

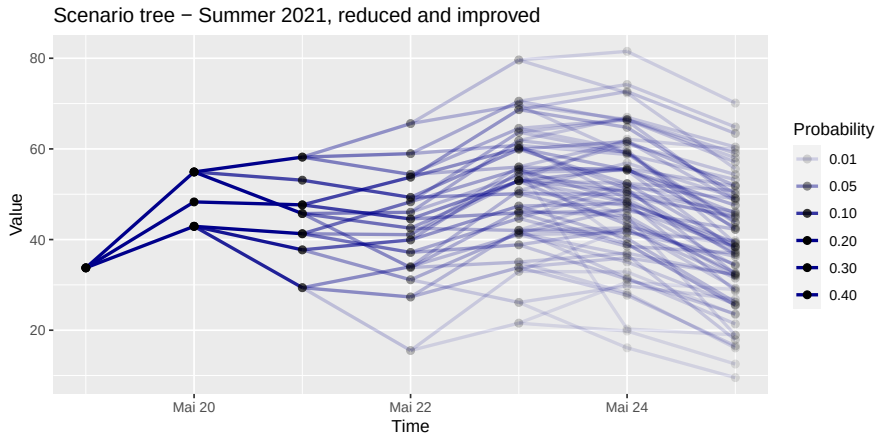

Figure 4.34: The reduced and improved version of the tree stemming from the original method for the summer period from Figure 4.25.

In comparison, resulting trees for the reduced and improved added trees are given in Figure 4.35 for the winter and in Figure 4.36 for the summer.

Calculating the distances to their corresponding big trees as well as to each other, matching summer and winter trees, respectively, yields the results given in Table 4.37. Here, the trees reduced with the Wasserstein distance of order 2 show a smaller distance to their big trees for distance measures based on the l_1-norm as well as on the l_2-norm. Consequently, we choose to go further with the trees that were reduced with this method.

To sum up, our trees generated as follows: The trees with many nodes are created while using the l_1-norm, and then reduced using the nested distance of order 2. As the

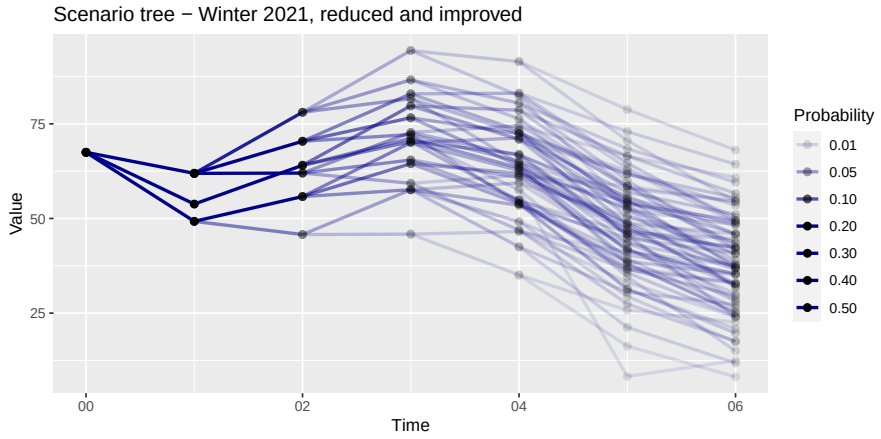

Figure 4.35: The reduced and improved version of the tree from the adding method for the winter period from Figure 4.28.

Season	Type	Wasserstein order (red.)	Distance order 1	Distance order 2
Summer	Original	1	32.6	16.4
Summer	Added	1	31.9	15.8
Summer	Original	2	30.4	14.8
Summer	Added	2	29.6	14.3
Winter	Original	1	31.0	15.5
Winter	Added	1	32.5	16.2
Winter	Original	2	30.3	14.6
Winter	Added	2	29.5	14.4

Table 4.37: Distances between all improved trees and their corresponding big trees.

algorithm for improving trees follows the same line of thought as the creation of a big tree, here again the l_1-norm is used.

Remark 4.51. The factors in Equation (4.4) have a Markovian representation; therefore, we could have implemented a scenario grid instead. In order to keep the procedure general and the number of paths overseeable, we nonetheless decided for the regular scenario tree generation.

Hourly prices Now that we have built scenario trees for our electricity price model, we know which average daily electricity prices are contained in them. Nonetheless, the real spot market does not yield one price per day, but 24. In order to incorporate this hourly price structure into the scenario trees, we use the type days defined in Section 4.1 as well as calendar information and find representatives for all of them based on historical data. Then, all daily prices in the trees are mapped to their corresponding representatives, which yields a vertical movement of these hourly prices depending on the correspond-

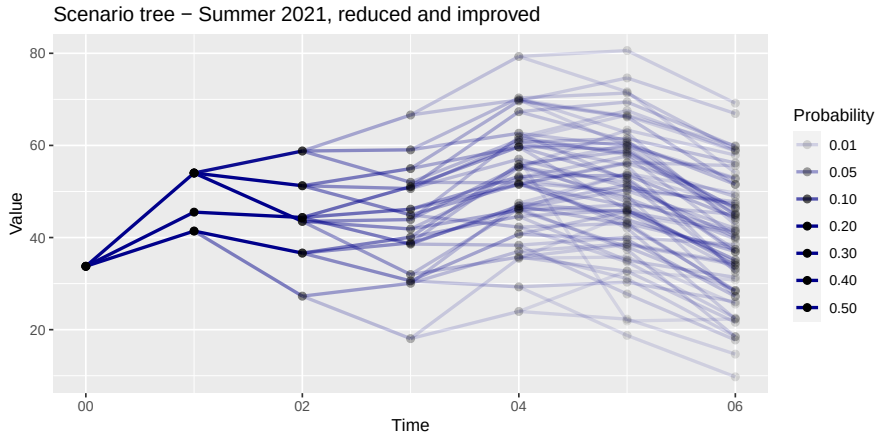

Figure 4.36: The reduced and improved version of the tree from the adding method for the summer period from Figure 4.29.

ing daily average.

In detail, our approach works as follows: Inspired by [63], we choose the following categories as possible predictors:

- Quarter of the year
- Month of the year
- Weekday, starting with 1 as Monday to 7 as Sunday
- Typeday, where the types are Mondays, Tuesdays to Thursdays, Fridays, Saturdays and bridge days and partly national holidays, Sundays and national holidays
- Hour of the day
- Quarter-Hour as variable capturing the interdependence between quarter and hourly behavior
- Month-Hour as variable capturing the interdependence between month and hourly behavior
- Weekday-Hour as variable capturing the interdependence between weekdays and hourly behavior
- Typeday-Hour as variable capturing the interdependence between typedays and hourly behavior

These are used as predictors in a categorical regression. Standard methods for such a regression are for example ANOVA or group LASSO. ANOVA stands for analysis of variance and is mainly used for categorical regression with one or two predictors. Group

LASSO, in comparison, offers the possibility of including all the categories that are potentially relevant in the regression and then reduces them to the predictors that explain the most variance in the data. Therefore, we use this method, which was first presented by [69]. In group LASSO, entire categories are always included in or excluded from the regression. This prevents only certain elements from a category from being included in the regression, e.g., only two out of four annual quarters. The objective function in the group LASSO for a problem with J groups is defined as

$$\frac{1}{2} \mid\mid Y - \sum_{j=1}^{J} X_j \beta_j \mid\mid^2 + \lambda \sum_{j=1}^{J} ||\beta_j||_{K_j}, \tag{4.46}$$

where $||a||_K := (a'Ka)^{1/2}$ for a vector $a \in \mathbb{R}^d$ and the $K_j, j = 1, ..., J$, are symmetric positive definite matrices with dimension $d \times d$. The first part of the sum represents the typical least-squares regression, were the latter contains a penalty term. Here, the design matrix X and covariance vector β have been replaced by collections of J design matrices and J covariance vectors, respectively. In addition, the penalty term changes in that it is now defined by a sum over the l_2 norm, which in turn are defined per chosen group by a positive definite matrix K_j.

Now, starting from the reduced and improved scenario trees generated above, we translate them to their hourly counterpart. Therefore, we first specify a time period of historical spot data for the model's calibration. The whole period from the years 2015 to 2020 is selected to serve this purpose. With the method described above, we find that from all possible defined categories, month, typeday, hour, month-hour and typeday-hour provide the most relevant information, and thus enter the hourly model. The resulting model is applied to the days for which we need an hourly simulation. As a result, we obtain increments to be added or deducted from the daily mean, which makes the breakdown of daily to hourly values possible. Figure 4.38 presents such a mapping of the daily improved scenario tree for the original model and the summer period from Figure 4.34 to its hourly version. As the increments are added to the daily average, the intensity of the oscillations around it do not depend on its magnitude.

Remark 4.52. An alternative would be to compute factors that are multiplied with the daily average price instead of adding increments. Our analysis of the data from 2015 to 2019 in comparison with the data from the years 2021 to 2022, that saw much higher prices than the years before, showed that these oscillation factors stayed within the same magnitude as before, wherefore absolute deviations of course changed a lot. Consequently, this method also has its advantages. We tested both methods, and the increment version yielded better fits for our purpose, so we decided to move further with it.

Equipped with the hourly scenario trees, the next step is to optimize the trading of the virtual battery based on them. But before we finally turn to that, we make a digression towards when and how DP is applicable for our setting.

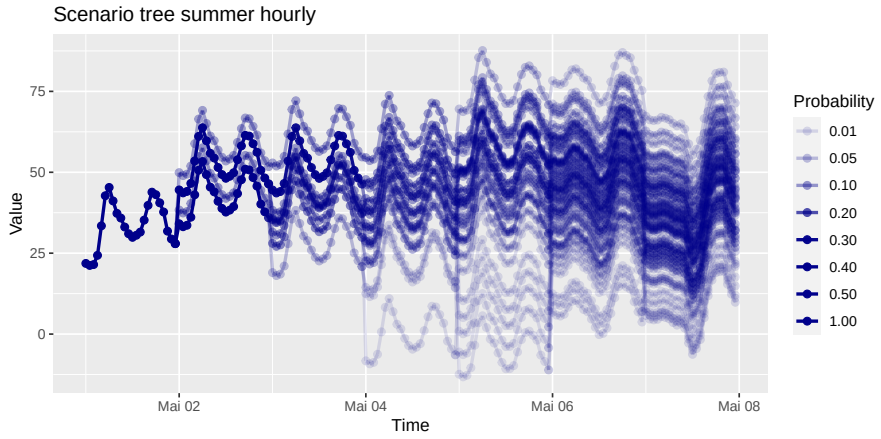

Figure 4.38: A scenario tree which was transformed from daily to hourly values, corresponding to the daily tree in Figure 4.34.

4.4.3 Conditions for optimality of dynamic programming in our setting

As it was explained in detail in Section 4.3, dynamic programming is based on Bellman's optimality principle. Furthermore, the common problem that dynamic programming is applied to has a certain structure, i.e. an optimal substructure. In addition, dynamic programming problems are often assumed to also have overlapping subproblems. When an optimization problem has these two properties, dynamic programming can speed up the calculation remarkably.

Overlapping subproblems are given in our setting: For each stage, the increments, that were calculated in order to map the daily average price to the corresponding hourly values, will be equal for all nodes on that stage. Consequently, knowing the optimal battery operation mode conditioned on a starting battery level for one node on that level equals knowing it for all nodes on that level. We did remark though, that there are other methods to map from daily averages to hourly values, see Remark 4.52. For example, when looking at factors instead of increments, this special answer does not hold any longer. Nonetheless, the overlapping subproblem property is given: When all optimal strategies conditioned on all possible starting battery levels are calculated for one node, these results can be reused for optimization problems containing that node.

The property of optimal substructure is in comparison not fulfilled in our setting: Even though one might guess that this is the case, since at each time point only the starting battery level, the battery constraints, and the possible prices are relevant, the starting battery level is also associated with costs that depend on the decisions made in the previous period. This information cannot be included in the current state, which means that common dynamic programming is not applicable by default in our setting. Nonetheless, we want to use dynamic programming in order to compare the different

optimization strategies. Therefore, we first introduce an approximation to the original dynamic programming program, and then discuss when it still might be applicable.

Remark 4.53. Following Remark 4.37, it would indeed have been possible to make this problem Markovian as well by including fixed starting battery levels into the system state. As this would have remarkably enlarged the state space as well as the underlying scenario tree, we decided against it.

As a first approach to the application of the idea from dynamic programming, a penalty for the optimization to use a certain battery starting state is introduced into the optimization. We use the node-based approach for its definition:

$$\min_{v\in\mathcal{V},B_n(0)} \quad \sum_{\tilde{n}\in N_+(n)} p(\tilde{n}) \sum_{i=1}^{24} S(i,\tilde{n})\, v(i,\tilde{n}) - \bar{S}(n)\cdot B_n(0) \tag{4.47}$$

$$\begin{aligned}
\text{s.th.} \quad & b_{\min} \leqslant b_n(0) + \sum_{i=1}^{M} v(i,n) \leqslant b_{\max}, \quad M = 1,2,...,24,\ n = 1,...,K,\\
& |v_i(n)| \leqslant b_a, \quad i = 1,...,24,\ n \in N,\\
& b_{\tilde{n}}(0) = B_n(0), \quad \forall \tilde{n} \in N_+(n), n = 1,...,K,\\
& b_{\tilde{n}}(0) = B_0(0), \quad \tilde{n} = 0,\\
& b_n(0) + \sum_{i=1}^{24} S(i,\tilde{n})\, v(i,\tilde{n}) = B_n(0), \quad \forall n \in N : n \notin N_T.
\end{aligned}$$

Here, $\bar{S}(n)$ represents the average electricity price from one stage earlier than the stage of n, and $B_n(0)$ represents the end battery level on node n and consequently also the starting battery level for its successor nodes. For each node, electricity costs are minimized conditioned on the fact that all nodes with the same parent start at the same battery level. As described above, a penalty term regarding this battery starting level is introduced in order to capture the cost to fill the battery to this level on the day before. This is represented by the product of the starting battery level times the average electricity price from one day earlier. As not all levels are optimized at the same time but rather level by level and node by node, this does not yet fully qualify as a dynamic program, but yields an approximation to it.

Concerning the constraints, the first and the second auxiliary constraint ensure the same as in the case of two-stage approximation optimization: Compliance to the battery's statistics. The third constraint serves the fact that all children nodes from a node are supposed to start with the same battery level into the optimization. This battery level is now known beforehand except for the case $n = 0$, where the battery level at the beginning of the current day is known. Consequently, it is allowed to optimize over all but this one starting battery levels in order to find the best compromise starting battery level for each node. Specifying battery starting levels for $n = 0$ happens in the fourth constraint. Finally, the fifth constraint ensures that battery levels at a node n also end at $B_n(0)$, the starting battery level for its children nodes.

The implementation of penalty dynamic programming equations is straightforward, but it provides only an approximation of the optimal solution and not necessarily the optimal solution itself. However, we can show that true dynamic programming is also possible in our specific optimization setting under certain specifications, as shown by the following proposition:

Proposition 4.54. *Consider a two-stage optimization over nodes n and $\tilde{n} \in N_+(n)$, i.e. the children of node n. The children have their end battery levels $b_{\tilde{n}}(24)$ fixed and given. In contrast to that, the starting battery level $b_n(0)$ at node n is unknown, and the same holds true for the end battery level $b_n(24)$ at node n which equals the starting battery level $b_{\tilde{n}}(0)$ for nodes $\tilde{n} \in N_+(n)$.*

Then, the following holds true: For every realization of prices in nodes n and $\tilde{n} \in N_+(n)$ and fixed battery minimum b_{min} and maximum b_{max}, there exists a battery speed b_a^* such that for every speed $b_a \geqslant b_a^*$, the optimal end battery level at node n, i.e. $b_n(24)$, is independent of the starting battery level at node n, i.e. $b_n(0)$. Furthermore, there exists an optimal battery speed b_a^{**} such that for every speed $b_a \geqslant b_a^{**}$, the optimal end battery level at node n, i.e. $b_n(24)$, is also independent of the end battery levels $b_{\tilde{n}}(24)$ at nodes $\tilde{n} \in N_+(n)$.

Proof. The existence of b_a^* and b_a^{**} is secured through the trivial battery speed in which $b_a^* = b_a^{**} = b_{\text{max}} - b_{\text{min}}$. This is the battery speed at which the battery can be completely filled or emptied in every hour.

Going into detail for the trivial battery speed, the first case considered is that all prices $S(i,n)$, $i = 1, ..., 24$, and $S(j,\tilde{n})$, $j = 1, ..., 24$ are equal. In this case, it is obviously optimal to buy once or to sell once in order to reach the demanded battery level $b_{\tilde{n}}(24)$ at the end; flexibility cannot be used to gain more reward. If all prices are equal, then the point of time when this transaction is made is not important, it just needs to lie in the available time interval. That leads to the fact that each strategy making this trade at node $\tilde{n}$ can be shifted to a strategy that trades during node n. Consequently, we find that there are optimal strategies that are all aligned before reaching their final battery level of node n.

Now, the second case considered is where all prices over the two nodes except one are equal, i.e.

$$\exists (i,m) \,:\, S(i,m) \neq S(j,\tilde{m}) \text{ for all } (j,\tilde{m}) \in \{(1,n),(2,n),...,(24,\tilde{n})\} \backslash (i,m).$$

For further analysis we distinguish between $S(i,m) > (S,\tilde{m})$ and $S(i,m) < (S,\tilde{m})$. Considering the first scenario, we furthermore look at three separate options: The first is $(i,m) = (1,n)$, the second is $(i,m) = (24,\tilde{n})$ and the third is $(i,m) \neq (1,n) \wedge (i,m) \neq (24,\tilde{n})$. Now, for the first scenario with $(i,m) = (1,n)$, all strategies except the one starting with an empty battery sell everything in the battery at $(1,n)$, and are thus equal after this. In the second scenario with $(i,m) = (24,\tilde{n})$, all strategies except the one starting with a full battery fill up the battery fully in order to sell $b_{\text{max}} - b_{\tilde{n}}(24)$ at $(\tilde{n},24)$. Filling up the battery takes place during a period of only equal prices and thus is based on the same argument as the first case. Finally, for $(i,m) \neq (1,n) \wedge (i,m) \neq (24,\tilde{n})$, all strategies except the one starting with a full battery fill up the battery at a time point

earlier than (i, m). Then, at (i, m), everything is sold and finally, at a time point later than (i, m), the battery is filled such that the battery level equals $b_{\tilde{n}}(24)$. Consequently, again we can find strategies starting from each battery level that are optimal and aligned before passing $(24, n)$. The argument chain works analogously for $S(i, m) < (S, \tilde{m})$.

For all cases with more unequal prices, the argument can be reduced to the arguments from the case with at least one different price. This is due to the following process: The existence of a first price which is different from the other prices always immediately triggers a reaction for all strategies. Either this first different price is higher than the prices before, then - like in the case with one unequal price - all batteries immediately fill the battery completely, or it is lower than the prices before, then all batteries immediately empty the battery completely. As a consequence, all strategies have the same battery level afterwards, and no matter what prices are observed then, they all use the same strategy from that point onward. Consequently, the existence of the trivial battery speed $b_a^* = b_{\text{max}} - b_{\text{min}}$ is secured.

The same holds true for b_a^{**} under the assumption that at least two different prices exist for electricity on nodes n and $\tilde{n}$. In this case, we skip the scenario with all prices being equal. As all strategies exploit the difference between the prices, they align and therefore move unisono afterwards, no matter where the final battery level is placed. Consequently, we also find $b_a^{**} = b_{\text{max}} - b_{\text{min}}$. □

Remark 4.55. The battery speeds b_a^* and b_a^{**} need not be equal. Furthermore, a battery speed $b_a > b_{\text{max}} - b_{\text{min}}$ makes sense in real world applications, but is not relevant for a market setting in the day ahead market, where the smallest traded time interval is an hour.

Remark 4.56. If the optimal strategy is independent of $b_n(0)$ and $b_{\tilde{n}}(24)$, a scenario tree extending beyond one day is unnecessary as all necessary information for optimal decision-making at node n is available.

Example 4.57. Naturally, there are batteries with a battery speed that does not surpass any of the two battery threshold speeds on a node. An example is the following, in which we consider a battery with $b_{\text{min}} = 0\,\text{MW}$, $b_{\text{max}} = 120\,\text{MW}$ and $b_a = 7\,\text{MWh}$. The aspired end battery level on the children nodes is $60\,\text{MW}$. For a given tree, the optimal strategies are then given in Figure 4.39.

A battery whose battery speed is greater than or equal to b_a^* at every node of the considered scenario tree is thus eligible for computation with dynamic programming, because no matter where a node starts, its end - depending on the final battery level of its children - will still be the same. Of course, b_a^* depends on the nodes under consideration, and thus can vary over the range of the tree. A notation like $b_a^*(n)$ would keep this dependence on the node n in mind, but we will omit it for notational convenience.

In many applications, battery speeds smaller than the trivial battery speed are of interest. Therefore, we present some procedures to find the corresponding battery speeds for a given tree and battery, or to test whether the existing battery speed is greater than the corresponding thresholds b_a^* or b_a^{**} for a node.

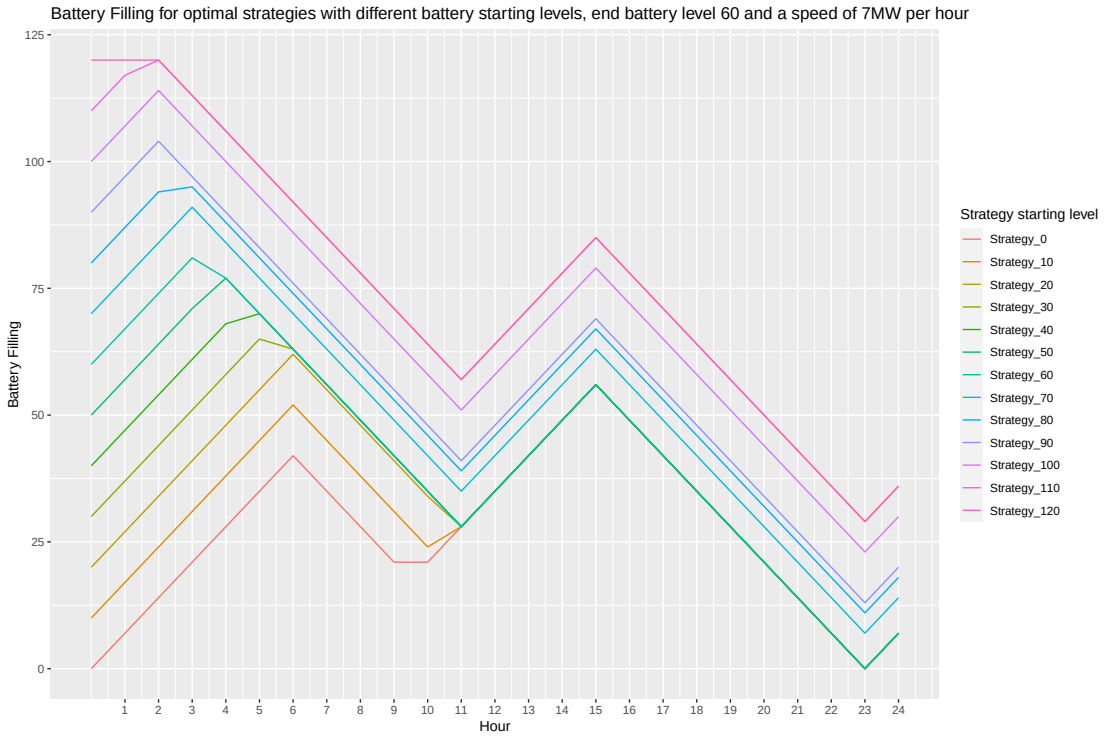

Figure 4.39: Optimal battery strategies from Example 4.57 depicted through their battery levels, all starting from different start battery levels.

Algorithm 11 tests the latter for threshold b_a^*. It starts by taking in inputs such as battery statistics (maximum and minimum battery levels, and battery speed), a tree snippet with one parent node and corresponding children nodes along with their values and probabilities, and the final battery levels for the children nodes. It then builds two vectors, one called b_{start} with the minimum and maximum battery levels and another vector $b_n(24)$ with values of 0. Now, the algorithm loops through the two values of vector b_{start} and optimizes over the two-stage tree, inserting the current value of b_{start} into the optimization as the starting battery level. The resulting optimal volume strategy for the node is then added to b_{start} to calculate $b_n(24)(1)$ and afterwards $b_n(24)(2)$. If the values are equal, the algorithm returns True, otherwise False. The returned value indicates whether the existing battery speed exceeds the threshold speed.

To check whether the battery speed exceeds the other threshold b_a^{**}, Algorithm 12 can be used, which is a modified version of Algorithm 11 with a double loop to check for two different final battery levels at stage two, namely $b_{\max}$ and $b_{\min}$. With that, again the two possibilities with the biggest distance between them are checked.

It is left to explain why it is enough to check the limit cases of a full and of an empty starting battery. The reason for that is given by the following proposition.

Result: The algorithm returns TRUE if the battery speed is greater than or equal to the corresponding threshold, and FALSE if it is smaller.

begin
Input: Battery statistics $b_{\text{max}}, b_{\text{min}}, b_a$, tree snippet with one parent node n and corresponding children nodes $\tilde{n} \in N_+(n)$ that contains node values and probabilities, final battery levels for nodes $\tilde{n}$.
Build vector $b_{\text{start}} = (b_{\text{min}}, b_{\text{max}})$.
Build vector $b_n(24) = (0, 0)$.
for $i \in (1, 2)$ **do**
Optimize over the two-stage tree, e.g. with the two-stage approximation algorithm. Insert $b_{\text{start}}(i)$ into the optimization as starting battery level.
$b_n(24)(i) =$ sum of $b_{\text{start}}(i)$ and the returned optimal volume strategy for node n.
end
if $b_n(24)(1) = b_n(24)(2)$ **then**
$ret =$ TRUE.
end
else
$ret =$ FALSE.
end
return ret.
end

Algorithm 11: Test whether existing battery speed exceeds the threshold battery speed b_a^* in a two-stage optimization problem.

Proposition 4.58. *Consider a two-stage optimization based on Equation* (4.48) *over nodes n and $\tilde{n} \in N_+(n)$, i.e. the children of node n. The starting battery level $b_n(0)$ at node n is unknown, whereas the end battery level $b_n(24)$ at node n which equals the starting battery level $b_{\tilde{n}}(0)$ for nodes $\tilde{n} \in N_+(n)$ as well as end battery levels $b_{\tilde{n}}(24)$ are given.*

Then, it holds true for battery strategies starting at $b_n^1(0)$ and $b_n^2(0)$ that there always exist two optimal strategies π_1 and π_2 starting from the respective battery levels, that for the same optimization problem do not cross each other.

Proof. We assume without loss of generality that $b_n^1(0) < b_n^1(0)$. There are two possible cases in which the strategies could cross each other: The first is where both strategies' battery levels align at some point and then, the strategy that started with the higher starting battery level, i.e. π_2, moves below the strategy starting with the lower initial battery level, i.e. π_1. The second case is that both strategies do not align but rather cross each other between two time points. Both cases can be proved with similar arguments, and we start with the first case.

In the first case, battery levels of π_1 and π_2 align at some point in time t with $t \in \{1, .., 23\}$. Then, we know by definition that π_1 must be optimal for the time horizon $[0, 48]$ when starting out from battery level $b_n^1(0)$. Due to the optimality principle from Bellman, it follows that π_1 must also be optimal for the time horizon $[t, 48]$ when starting

Result: The algorithm returns TRUE if the battery speed is greater than or equal to the corresponding threshold, and FALSE if it is smaller.

begin
Input: Battery statistics $b_{\text{max}}, b_{\text{min}}, b_a$, tree snippet with one parent node n and corresponding children nodes $\tilde{n} \in N_+(n)$ that contains node values and probabilities.
Build vectors $b_{\text{start}} = b_{\text{end}} = (b_{\text{min}}, b_{\text{max}})$.
Build vector $b_n(24) = (0, 0, 0, 0)$.
Counter $k = 1$.
for $i \in (1, 2)$ **do**
for $j \in (1, 2)$ **do**
Optimize over the two-stage tree, e.g. with the two-stage approximation algorithm. Insert $b_{\text{start}}(i)$ into the optimization as starting battery level, and $b_{\text{end}}(i)$ as end battery level.
$b_n(24)(k) =$ sum of $b_{\text{start}}(i)$ and the returned optimal volume strategy for node n.
Update counter $k = k + 1$.
end
end
if $b_n(24)(1) = b_n(24)(2) = b_n(24)(3) = b_n(24)(4)$ **then**
$ret =$ TRUE.
end
else
$ret =$ FALSE.
end
return ret.
end

Algorithm 12: Test whether existing battery speed exceeds the threshold battery speed b_a^{**} in a two-stage optimization problem.

out from battery level $b_n^1(t)$, which is the battery level reached after making all optimal decisions at times $\{0, 1, ..., t-1\}$. Therefore, no other strategy can be found with lesser costs for time horizon $[t, 48]$, and adapting π_1 must be optimal for π_2 as well. Now we consider the case where a crossing of battery levels takes place inside a time interval except for at one of its ends. Here, the same argument as before can be used: If the strategies have the option to cross each other during one hour and end at different battery levels, they could have actually arranged themselves to end on the same level, and consequently adaptation to each other must be optimal as well. □

Remark 4.59. If there are equal prices, several optimal strategies can exist that also could cross each other. Nonetheless, these strategies would not yield a better objective value. Consequently, we accept that this could be the case, but concentrate on those optimal strategies that do not cross each other. A possibility to enforce the latter could be to

allocate transaction costs to single trades.

We state two further behaviors that are exhibited by an optimal strategy:

Proposition 4.60. *In the optimization setting given in Equation* (4.48)*, the following must be true for the optimal strategy π:*

(i) Within two times of touching the same battery limit – b_{min} or b_{max} – all prices where π sells must be higher than the prices where it buys.

(ii) For points in time where π does not change its battery level, e.g. during the interval $[t, t+h], h \in \mathbb{N}$, one of the following must hold true: Either $S(t-1) < S(t), ..., S(t+h) < S(t+h+1)$, or $S(t-1) > S(t), ..., S(t+h) > S(t+h+1)$.

Proof. In order to prove (i), we w.l.o.g. consider the case of b_{min}, where we assume the time between hitting it twice is $[t, t+h]$. For each point in this interval, we know that $\pi(s)$ is buying, i.e. $s \in \{s : s \in [t, t+h],\ \pi(s) > 0\}$, selling, i.e. $s \in \{s : s \in [t, t+h],\ \pi(s) < 0\}$, or doing nothing, i.e. $s \in \{s : s \in [t, t+h],\ \pi(s) = 0\}$. Now, let us assume there exist $s_1 \in \{s : s \in [t, t+h],\ \pi(s) > 0\}$ and $s_2 \in \{s : s \in [t, t+h],\ \pi(s) < 0\}$ with $S(s_1) > S(s_2)$. Then, a strategy $\tilde{\pi}$ exists that equals π for all t except $\{s_1, s_2\}$ with $\tilde{\pi}(s_1) = -\pi(s_1)$ and $\tilde{\pi}(s_2) = -\pi(s_2)$. It earns strictly more than strategy π due to $S(s_1) > S(s_2)$, which is a contradiction.

Moving onward to (ii), we consider the case $S(t-1) < S(t+h+1)$. We know that $\forall s \in [t, t+h] : \pi(s) = 0\}$. Now, if there exists $s_1 \in [t, t+h]$ with $S(s_1) < S(t-1)$, then we can define a strategy $\tilde{\pi}$ that is the same as π except for $\{t-1, s_1\}$. There, it is defined as $\tilde{\pi}(t-1) = 0$ and $\tilde{\pi}(s_1) = \pi(t-1)$. Due to $S(s_1) < S(t-1)$, this is a contradiction to the optimality of π. The same holds true for the case $S(s_1) > S(t+h+1)$. Here, we define strategy $\tilde{\pi}$ that is the same as π except for $\{s_1, t+h+1\}$. There, it is defined as $\tilde{\pi}(t+h+1) = 0$ and $\tilde{\pi}(s_1) = \pi(t+h+1)$, which again leads to a contradiction. □

We have not yet addressed the problem of finding the threshold battery speeds for a given battery. Algorithm 13 presents a way to calculate the threshold battery speed b_a^* for a given battery minimum and maximum. It calculates the threshold battery speed b_a^* for a two-stage optimization problem. The algorithm takes as input the battery statistics b_{max} and b_{min} and a tree snippet with parent node n and corresponding children nodes $\tilde{n} \in N_+(n)$ that contains node values and probabilities, plus final battery levels for the nodes $\tilde{n}$.

The main logic of the algorithm is inside a while loop, which in each iteration tests whether b_a already equals or exceeds the threshold battery speed b_a^*. The result of this test is stored in a variable called res. If res equals true, the value of b_a^1 is updated, and if res equals false, the value of b_a^2 is updated. Then, the value of b_a^3 is calculated as the floor of the average of b_a^1 and b_a^2, and b_a is re-calculated.

Finally, after the while loop ends, the function returns the value of b_a from the last row of the table tbl, which equals the battery speed that would have been necessary to apply DP without limitation. The algorithm follows a divide and conquer strategy: It tries to

Result: The algorithm returns the threshold battery level b_a^*.

begin
Input: Battery statistics $b_{\text{max}}, b_{\text{min}}$, tree snippet with one parent node n and corresponding children nodes $\tilde{n} \in N_+(n)$ that contains node values and probabilities, final battery levels for nodes $\tilde{n}$.
Build vector $b_{\text{start}} = (b_{\text{min}}, b_{\text{max}})$.
Build vector $b_n(24) = (0, 0)$.
Set $b_a^1 = 48,\ b_a^2 = 0,\ b_a^3 = (b_a^1 + b_a^2)/2,$
$b_a = (b_{\text{max}} - b_{\text{min}})/b_a^3$.
Define table $tbl =$ table$(bat,\ factor,\ val)$, set $log =$ FALSE.
while $log =$ *FALSE* **do**
Use Algorithm 11 to test whether b_a already equals or exceeds the threshold battery speed b_a^*. The algorithm returns res.
Add row $(b_a,\ b_a^3,\ res)$ to table tbl.
if $res =$ *TRUE* **then**
if $(b_a^3 - 1) \in tbl.factor$ **then**
Set $log =$ TRUE.
end
Set $b_a^1 = b_a^3$.
end
else
if $(b_a^3 + 1) \in tbl.bat$ **then**
Set $log =$ TRUE.
end
Set $b_a^2 = b_a^3$.
end
Set $b_a^3 = floor[(b_a^1 + b_a^2)/2],\ b_a = (b_{\text{max}} - b_{\text{min}})/b_a^3$.
end
return *Batteryspeed* bat *from last row of table* tbl.
end

Algorithm 13: Find threshold b_a^* for a two-stage optimization problem.

find the threshold battery level b_a^* by dividing the possible range of values into smaller subranges, testing the midpoint of each, and updating the subrange based on the results of the tests until the threshold value is found.

Remark 4.61. When testing the battery from Example 4.57 with $b_{\text{min}} = 0\,\text{MW}$ and $b_{\text{max}} = 120\,\text{MW}$, we find the following: The optimal battery speed from which on the battery strategies align, is $b_a = b_{\text{max}}/13 \approx 9.23$.

We still have to shed light on why it is enough to check the battery speeds $b_a \in [(b_{\text{max}} - b_{\text{min}})/48, (b_{\text{max}} - b_{\text{min}})/1]$. The answer for the upper bound $(b_{\text{max}} - b_{\text{min}})/1$ was given in Remark 4.55 already. For the lower bound $(b_{\text{max}} - b_{\text{min}})/48$, it holds true that this is the natural limit given through the 24-step structure of one day: As every strategy can at most move 24 times on one day, the lower bound is reached when a strategy starting

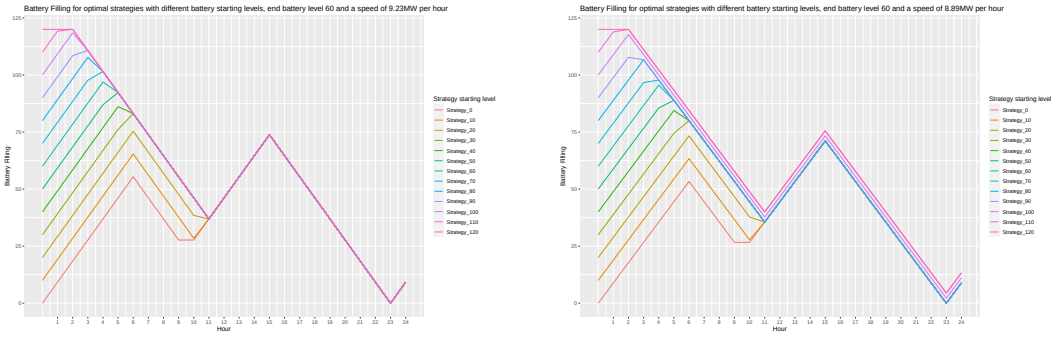

Figure 4.40: Both figures contain optimal battery strategies. On the left, the battery has a speed of $b_a = b_{\max}/13$ and all strategies align. On the right, the battery's speed was reduced to $b_a = b_{\max}/13.5$; now, the strategies do not align any longer.

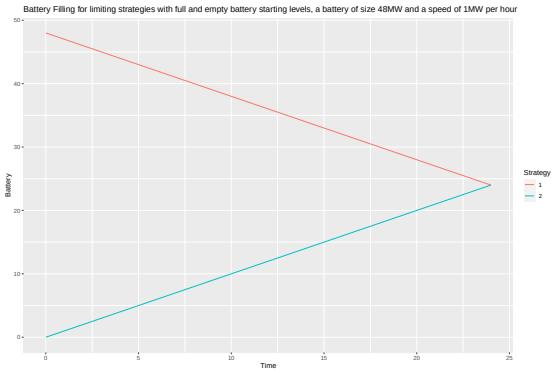

Figure 4.41: A battery with 48MW capacity and a speed of 1MW/h is depicted as well as two battery strategies, one starting with a full battery, the other with an empty one. Both battery levels meet at the end of the considered time period.

with full battery sells at maximum at every step in combination with a strategy starting from an empty battery that buys at every step. In this case, both strategies still meet after all trades of the first stage node. This is physically not possible for even smaller battery speeds. A plot depicting this situation is given in Figure 4.41. Furthermore, it is not only enough to concentrate on $[(b_{\max} - b_{\min})/48, (b_{\max} - b_{\min})/1]$, but the set to consider shrinks further to $\{(b_{\max} - b_{\min})/48, (b_{\max} - b_{\min})/47, (b_{\max} - b_{\min})/1\}$. This is caused by the fact that all strategies can only align after a full step, and as we are interested in the smallest battery speed needed to fulfill that, only these battery speeds need to be considered.

This subsection shows that there exist batteries for which the direct application of dynamic programming techniques without any approximation techniques is indeed pos-

sible. Nonetheless, whether or not that is an option for a battery with given battery speed also depends on the underlying scenario tree.

The results of this section can also be useful in guiding decisions when buying new batteries for trading purposes. The presented algorithms allow to test on historical data which battery speed would have performed in such a way that dynamic programming would have been applicable. That is not only relevant because it is in general the fastest of all three presented optimization methods, but also because the optimization of such a battery needs less price forecasting and thus reduces incurred forecast uncertainty.

4.4.4 Application of stochastic optimization to the German day ahead market

In order to make the stochastic optimization approaches comparable and more accessible, all three of them are applied to the problem setting from Equation (4.45). First, we apply two-stage approximation to the mentioned setting, followed by dynamic programming in its true form and then in its approximated form with penalty. Thus, we formulate a forward as well as two backward optimizations for the problem. Lastly, we formulate the problem for reinforcement learning. The second step is then to apply the introduced optimization algorithms.

Based on the problem described in Equation (4.45), we formulate a stochastic programming counterpart. We use the N scenarios contained in the scenario tree, with Ω^N being the set of all possible paths w. Their probability is given by $p(w)$ for all $w \in \Omega^N$. Furthermore, we assume the spot prices $S(t), t = 1, ..., 24$, of the actual day to be forecasted precisely enough to be viewed as deterministic instead of stochastic. This is a common assumption in energy price modelling, see e. g. [27]. Consequently, we reach the following optimization problem for the two-stage approximation:

$$\begin{aligned} \min_{v \in \mathcal{V}} \quad & \sum_{t=1}^{24} S(t)v(t) + \sum_{w \in \Omega^N} p(w) \sum_{t'=25}^{7*24} (S(t'))(w)\ (v(t'))(w) \qquad (4.48) \\ \text{s.th.} \quad & b_{\text{min}} \leqslant b(0) + \sum_{t'=1}^{M} (v(t'))(w) \leqslant b_{\text{max}}, \quad M = 1, 2, ..., 7*24,\ \forall w \in \Omega^N, \\ & |(v(t))(w)| \leqslant b_a, \quad t = 1, 2, ..., 7*24,\ \forall w \in \Omega^N. \end{aligned}$$

The first set of constraints ensures that the battery level never crosses battery boundaries for all points in time considered, whereas the second constraint set keeps the volume of a single hour between the limits given through the battery speed. Here, $|(v(t))(w)|$ represents the decision variable, i.e. the traded volume, at time t during scenario w.

Now, we describe the equations in the dynamic programming case. Due to its recursive nature, the focus now shifts from realizations w to nodes $n \in N$ and their predecessors

or children. Be reminded that $n = 0$ represents the root node and each tree level is defined through $N_t, t = 0, ..., T$. The optimization problem is formulated as follows:

$$\begin{aligned} \min_{v \in \mathcal{V},\, b_n(0)} \quad & \sum_n p(n) \sum_{i=1}^{24} S(i,n)\; v(i,n) \\ \text{s.th.} \quad & b_{\text{min}} \leqslant b_n(0) + \sum_{i=1}^{M} v(i,n) \leqslant b_{\text{max}}, \quad M = 0, 1, ..., 24,\ n \in N, \\ & |v(i,n)| \leqslant b_a, \quad i = 1, ..., 24, \\ & \sum_{i=1}^{24} v(i,\tilde{n}) + b_{\tilde{n}}(0) = b_n(0), \quad n \in N \backslash N(0),\ \tilde{n} \in N_-(n). \end{aligned} \tag{4.49}$$

Here, the node formulation is chosen again in order to simplify the recursion. Prices are thus specified by $S(i,n)$, where $i = 1, ..., 24$ represents the corresponding hour and n specifies the node under consideration. As before, the first set of constraints ensures that the battery's limits are adhered to, the second one makes sure that the battery's speed is not surpassed, and the last one sets the battery level of a stage equal to the sum of the battery level of the previous stage plus the volume accumulated in that stage.

Following the same structure, but with added penalty term, we formulate

$$\begin{aligned} \min_{v \in \mathcal{V},\, b_n(0)} \quad & \sum_n p(n) \Bigg(\sum_{i=1}^{24} S(i,n)\; v(i,n) - \bar{S}(\tilde{n} \in N_-(n)) \cdot b_n(0) \Bigg) \\ \text{s.th.} \quad & b_{\text{min}} \leqslant b_n(0) + \sum_{i=1}^{M} v(i,n) \leqslant b_{\text{max}}, \quad M = 0, 1, ..., 24,\ n \in N, \\ & |v(i,n)| \leqslant b_a, \quad i = 1, ..., 24, \\ & \sum_{i=1}^{24} v(i,\tilde{n}) + b_{\tilde{n}}(0) = b_n(0), \quad n \in N \backslash N(0),\ \tilde{n} \in N_-(n) \end{aligned} \tag{4.50}$$

as the DP approximation with penalty. Here, $\bar{S}(\tilde{n} \in N_-(n))$ represents the average electricity price of the parent node of n.

Finally, we want to apply reinforcement learning via Algorithm 10 to update an originally randomly initialized value matrix. Because RL as we defined it needs a Markov decision process, we define the corresponding matrix Q containing the optimal action-value estimates in the following way: It contains three columns, that represent actions, namely "buy", "hold" or "sell", and as many rows as there are time points where something can be bought, held or sold, i.e. $7 * 24$, times the number of possible battery states that the battery could start in. Alternatively, we could have decided to implement the problem in such a way as to look at each day as one point in time only, but we did not pursue this idea. Consequently, the update equation for system state $\mathcal{S}(t)$ and action x_t

is

$$Q[\mathcal{S}(t), x_t] = Q[\mathcal{S}(t), x_t] + \alpha\Big((S(t)v(t) + \lambda \max_x Q[\mathcal{S}(t+1), \cdot]) - Q[\mathcal{S}(t), x_t]\Big).$$

This can easily be translated to the value matrix in Algorithm 10. Here, α is a learning rate that regulates how much a value-action-pair changes after an iteration. In order to let the RL agent abide to the battery restrictions, the actions it can choose from are determined conditioned on the distance of the current battery level to the battery limits. E.g., if the battery is filled up to b_{max}, the agent does not have the option to choose "buy" as its action. Regarding the hyper parameters that need to be chosen, we did a simple grid search to determine the combinations that led to the best results on a test tree.

Before progressing, let us briefly comment on the computation of the optimal bidding strategy. Through the usage of scenario trees covering seven days, the computed optimal bidding strategy does not only focus on the next day, but on the whole next week. However, since newer and possibly better forecasts are available after one day has passed, the bidding strategy is only executed for today. By recomputing an optimal bidding strategy for the look-ahead period, we are in a setting in which the bidding strategy adapts each day to the new information about the price forecasts.

To describe this rolling horizon setting in more detail, we follow [20] and denote the points in time at which decisions have to be made by $t_0, t_1, \ldots$. We also denote the days of the planning horizon by T. At t_0, an optimal schedule for the entire planning horizon is computed and executed for the following day. At the next time t_1, the procedure is repeated. Regardless of whether new price forecasts are available, a new optimal schedule is computed for the entire planning horizon. Thus, at each point in time, a schedule covering T days is computed and the first day of the schedule is executed. Figure 4.42 presents a schematic representation of the rolling horizon.

In order to apply and compare the four methods, we first define two battery settings that we want to analyze:

- **Fast-charger**: Its minimum capacity is $b_{\text{min}} = 0$ and its maximum capacity $b_{\text{max}} = 100$. It starts with an empty battery $b(0) = 0$ and has a charging speed of $b_a = 100$. Consequently, it can fully charge or discharge in one hour.
- **Slow-charger**: Its minimum capacity is $b_{\text{min}} = 0$ and its maximum capacity $b_{\text{max}} = 100$. It starts with an empty battery $b(0) = 0$ and has a charging speed of $b_a = 5$. Consequently, it can fully charge or discharge in 20 hours when using them to go fully in one direction.

For all optimization methods except the DP approximation, the fast-charger should reveal the same results, as the battery speed in this setting makes preparation for a medium-term future unnecessary. In contrast to that, the slow-charger's battery speed is much smaller than the total battery capacity; therefore, planning with future prices does

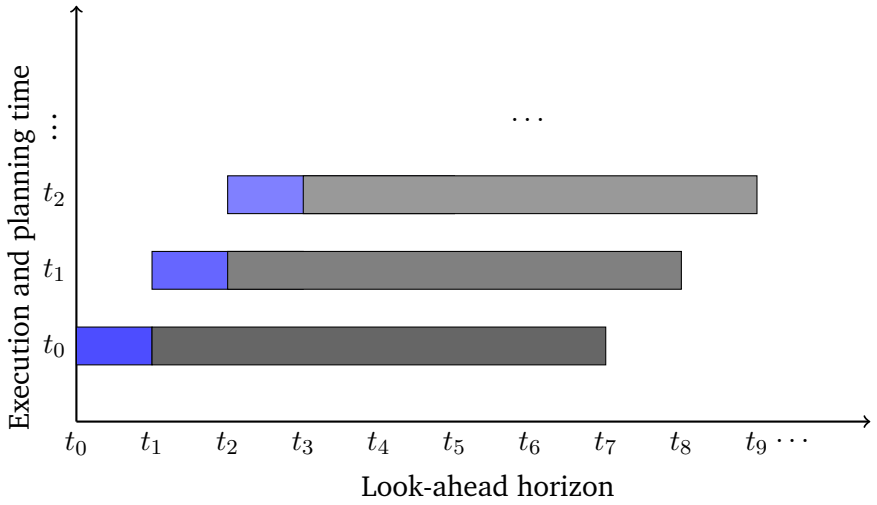

Figure 4.42: Schematic representation of the mechanics of a rolling horizon approach with a look-ahead horizon of seven time points. The blue period always marks the day where a decision is executed, and the grey areas depict the time horizon that is taken into account for that decision in combination with the blue period, aka the look-ahead horizon. As new information arrives after a day has passed, everything is recomputed with new execution day and new look-ahead horizon.

become more attractive. Regarding DP, it must hold true for the fast-charger that the battery speed surpasses b^* and b^{**}, and thus it is readily applicable. In comparison to that, the slow charger's battery speed is nearly never greater than b^*. Nonetheless, we apply the original DP algorithm and in case of several optimal battery end values of the first stage in a two-stage problem, we choose the first calculated locally optimal solution and proceed. This method thus does not yield a global optimum in this case, but rather a compilation of locally optimal solutions.

Now, we analyze the results for the summer period, first looking at the fast-charger, and then moving on to the slow-charger. Then, the winter period is considered shortly. Finally, we analyze whether the used methods are suitable for the given task and how their results compare to each other. The resulting costs for each strategy are presented in Table 4.43.

Figure 4.44 shows the optimal charging rates given by all four methods for the summer period and the tree based on the full model. It is visible that the two-stage stochastic programming, DP and RL all reach exactly the same solution, as was predicted. In contrast to that, the DP approximation method deviates once from the common path for the 5th of August. The picture strongly changes when we move to the slow-charger: Figure 4.45 presents the operation mode for the battery as given by the four methods. Both, the two-stage approximation as well as DP make use of the prices on the first days in order to sell them for profit later. Both strategies move closely to each other, but are not

Season	Model	Battery	Stochastic.P.	DP	DP.Approx.	RL
Summer	original	fast	-225	-225	-219.7	-225
Summer	original	slow	-52.9	-52.7	-43.6	-46.7
Summer	added	fast	-225	-225	-219.7	-224.9
Summer	added	slow	-53	-54.7	-43.6	-46.8
Winter	original	fast	-209.7	-209.7	-238.8	-204.6
Winter	original	slow	-26.9	-26.8	-45.5	-43.8
Winter	added	fast	-209.7	-209.7	-238.8	-202.1
Winter	added	slow	-27	-27	-45.5	-41.3

Table 4.43: Resulting optimal values for all optimization methods, both seasons, both tree building methods, and both batteries. All values are given in 10^2 €, and lower values indicate better strategies.

identical. In contrast to that, RL and the DP approximation also move closely together, but in contrast to the other two strategies, they never use the upper half of the battery. Regarding earnings, two-stage stochastic programming and DP outperform RL and the DP approximation, see Table 4.43. This outperformance is even stronger than visible in this table as the latter two strategies end with a battery level of 0, whereas the former two have some stored electricity left.

Now, turning to the same analysis on the added trees, Figure 4.46 and 4.47 present the optimized modi operandi for the battery by all four methods for the summer period and the tree based on the adding method for the fast-charger and the slow-charger, respectively. The battery strategies in Figure 4.46 for the tree stemming from the adding method are nearly identical to the strategies that are reached when optimizing on the tree stemming from the original model. Again, the DP approximation deviates once from the paths given by two-stage stochastic programming and DP, but here, RL also moves away once. Regarding the resulting cost, they equal the cost from before except for RL that pays the deviation with 10€. In comparison to that, the resulting methods for the fast-charger look very similar to before, but actually yield either lower or equal costs for all four strategies.

Considering the winter period, we again first look at the trees stemming from the original method: Figures 4.48 and 4.49 contain the optimal strategies for the fast-charger and the slow-charger, respectively. Here, we find that in case of the fast-charger, RL and the DP approximation have a harder time following the other strategies optimally in comparison to the summer period. Nonetheless, we find that the DP approximation yields remarkably lower cost than the other three strategies in Table 4.43. The reason for that again lies with the battery level after the seven days: All other strategies end with a battery level of 100, whereas DP approximation finishes with 0.

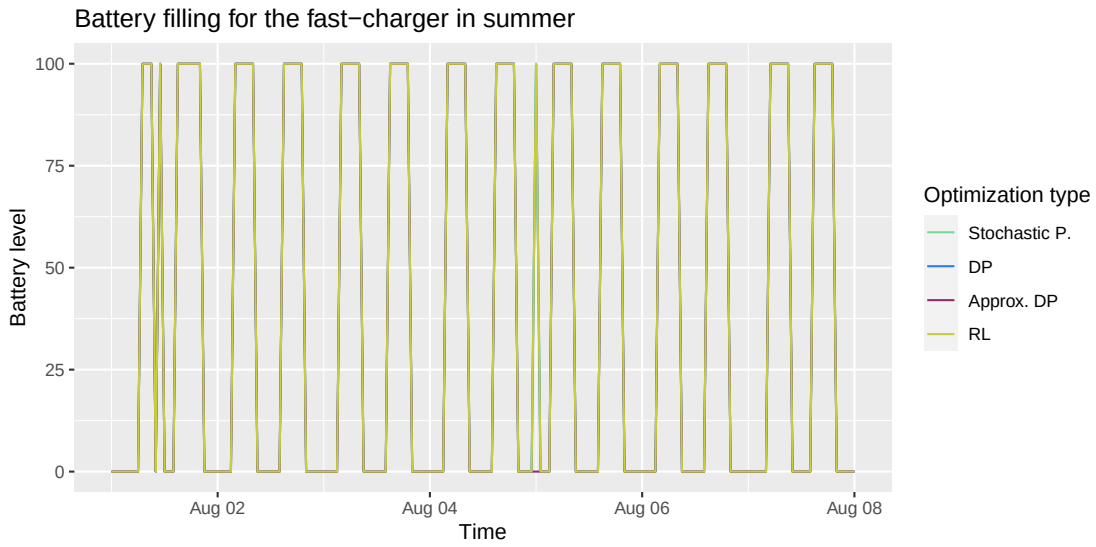

Figure 4.44: Battery filling over time for the summer period with the fast-charger and the tree based on the original method.

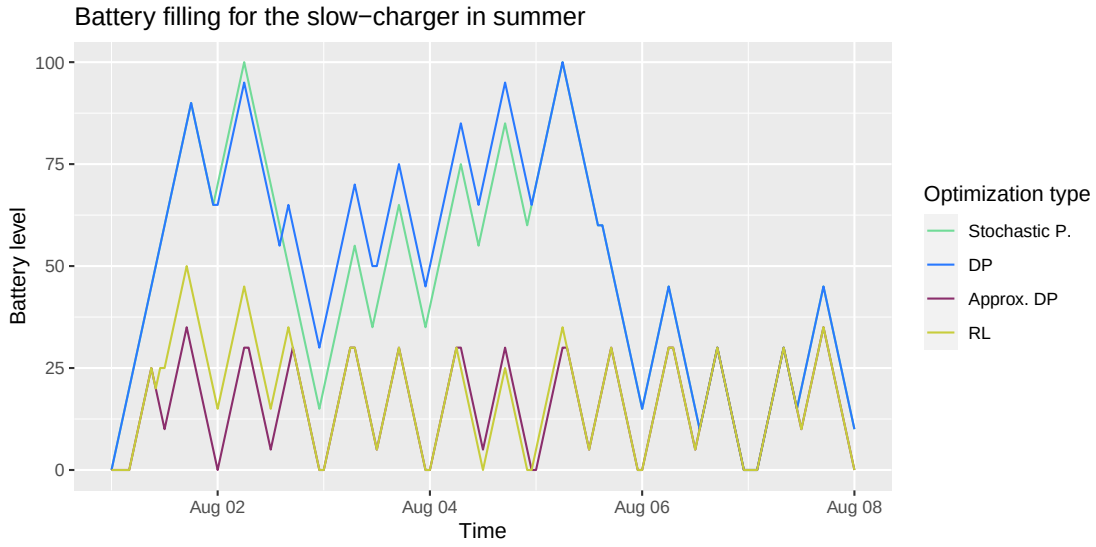

Figure 4.45: Battery filling over time for the summer period with the slow-charger and the tree based on the original method.

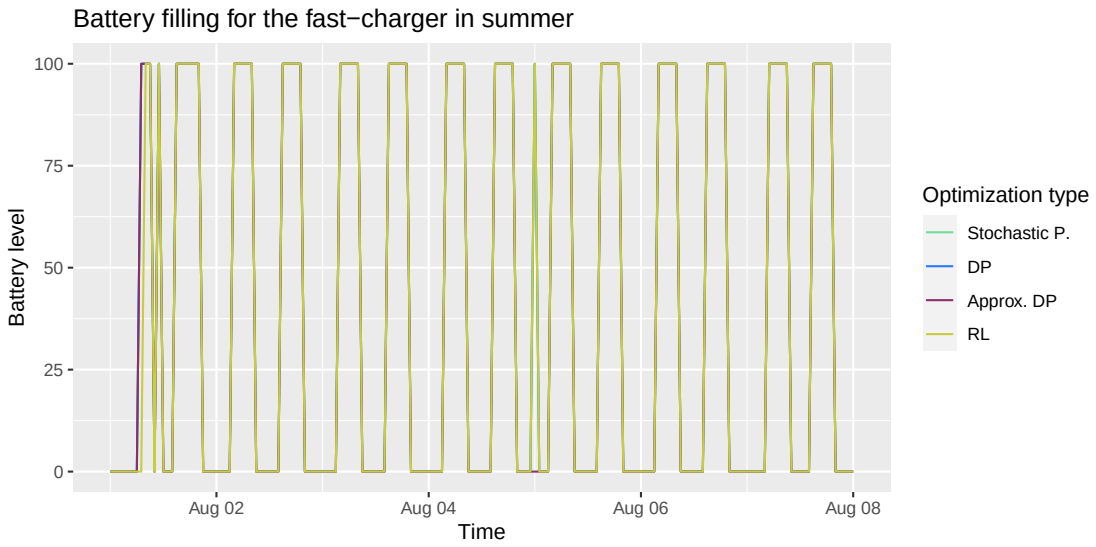

Figure 4.46: Battery filling over time for the summer period with the fast-charger and the tree based on the adding method.

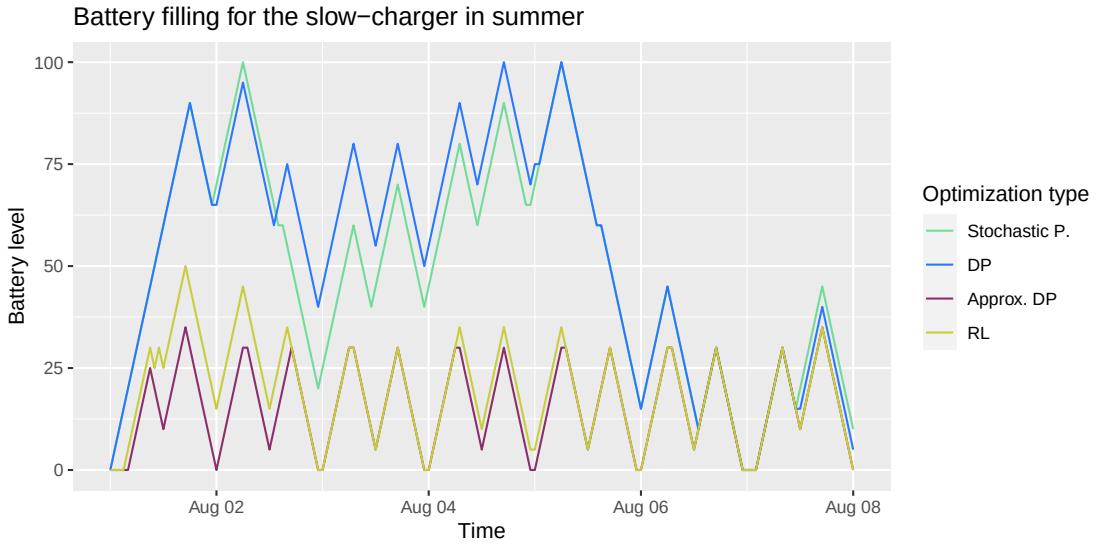

Figure 4.47: Battery filling over time for the summer period with the slow-charger and the tree based on the adding method.

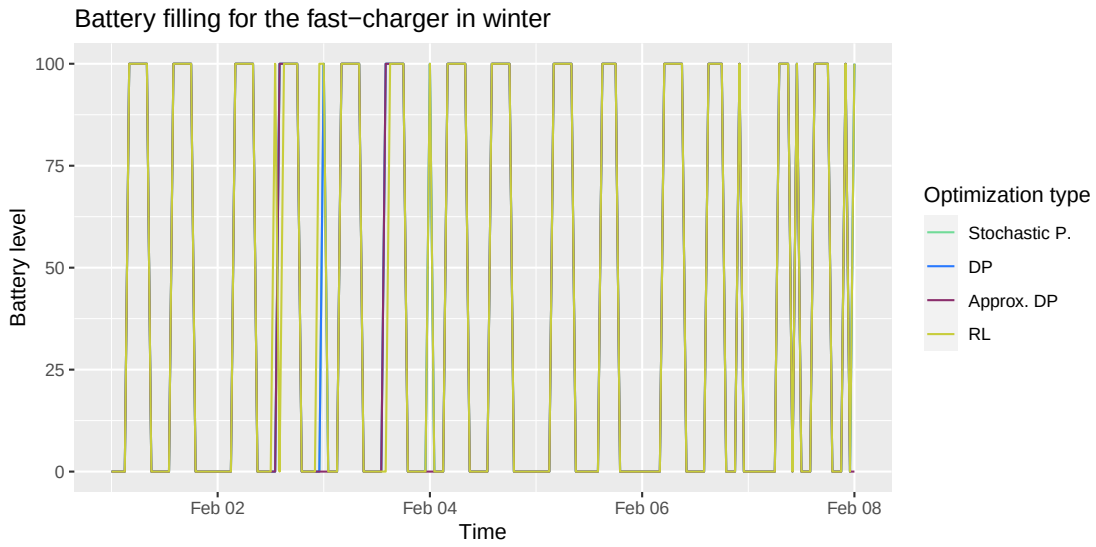

Figure 4.48: Battery filling over time for the winter period with the fast-charger and the tree based on the original method.

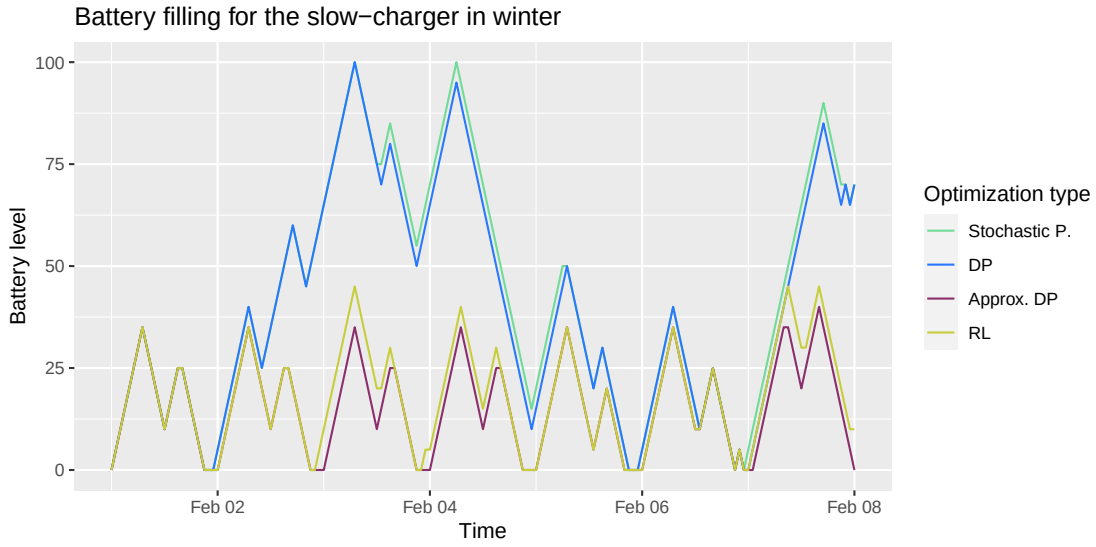

Figure 4.49: Battery filling over time for the winter period with the slow-charger and the tree based on the original method.

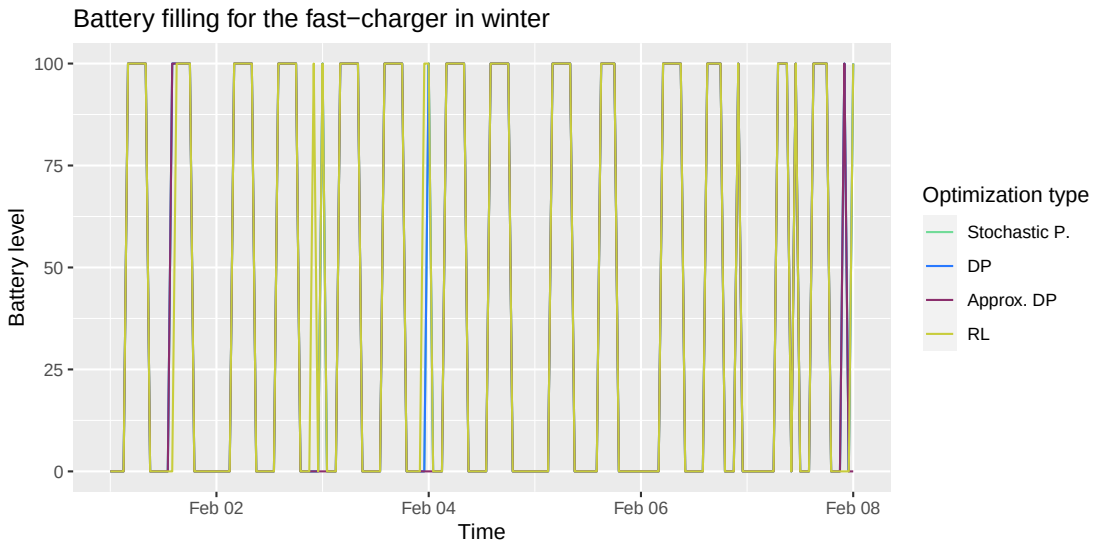

Figure 4.50: Battery filling over time for the winter period with the fast-charger and the tree based on the adding method.

Turning to the slow-charger, we find that on the first day, all strategies move unisono, but afterwards begin to again split into the groups two-stage stochastic programming and DP, and RL and DP approximation. Again, the latter two never move into the upper half of the battery. As the former two finish the week with a battery level around 70, and DP approximation and RL empty or nearly empty the battery at that point, we find higher returns for the latter two in Table 4.43.

For the same analysis on the added trees, Figure 4.50 and 4.51 present the optimized battery strategies for all four methods for the winter period and the tree based on the adding method for the fast-charger and the slow-charger, respectively. For both batteries, the resulting strategies are mostly the same. For the fast-charger, only RL receives higher costs of 250€ in comparison to trees stemming from the original method. For the fast-charger, two-stage approximation, DP and the DP approximation all yield equal or lower cost, and only RL again has higher costs of 250€. One reason for this difference is that it ends with a higher battery level when optimizing on the added trees in comparison to when optimizing on the original trees.

For all tested trees and optimization methods, the returned costs were negative, so the batteries did make money. Of course, the total amount is much higher for the fast-charging battery, because it could buy or sell 20 times more in one hour than the slow-charger. In comparison to that, the computed rewards for the slow-charger are not smaller by a factor of 20, but rather around 4.5 times smaller in the summer period and around 8 times smaller in the winter period. That hints at the fact that flexibility is worth more in the summer, where solar electricity uses flexibility more than during

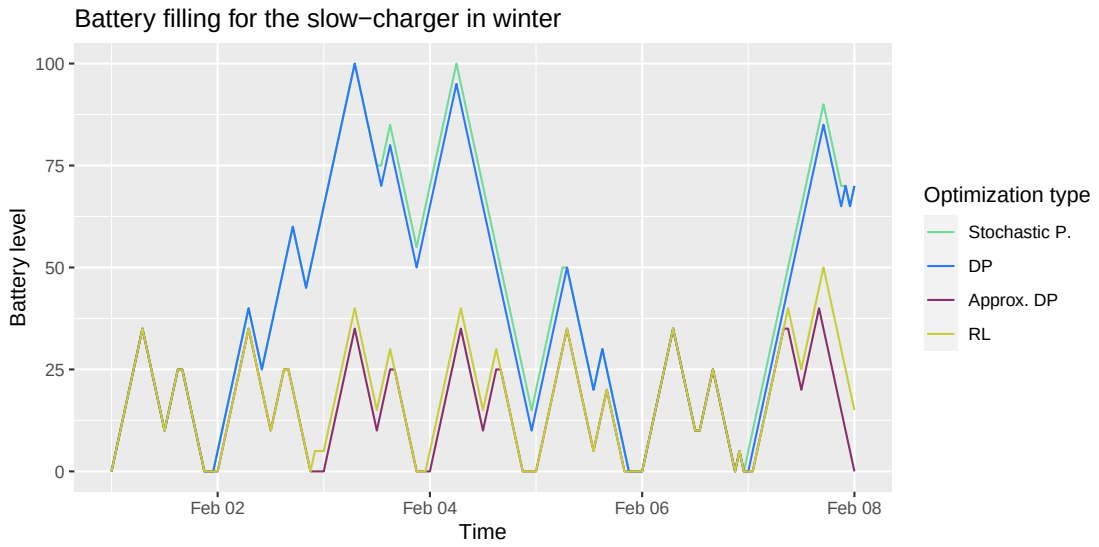

Figure 4.51: Battery filling over time for the winter period with the slow-charger and the tree based on the adding method.

winter. Furthermore, we see that RL and DP approximation did not perform on par with the other two optimization methods, which is already visible for the fast-chargers where the optimal solution is supposedly easy to find. For the DP approximation, the result depends on the chosen penalty, and for RL, it depends on the hyper parameter tuning. Probably, the latter would have required more hyper parameter tuning or more trajectories to refine its value matrix. Nonetheless, we tested around 50.000 hyper parameter constellations and calculated $90,000$ trajectories through the trees for each planning day, and as the first day is always fixed, all 90.000 trajectories passed that first day and added information to the value matrix here.

Regarding similarities between the strategies, we find that two-stage approximation and DP yield very similar results, even when DP is supposedly not optimal on the slow-chargers. As the latter requires much less computational resources due to less needed storage – there are no large constraint matrices as is the case for stochastic programming – it yields a valid alternative to the widely-used two-stage approximation, even when it is not yielding the globally optimal solution. Furthermore, we find that RL and the DP approximation yield very similar results in our example. This might indicate that the "look-ahead" over which RL admits negative rewards is similar to DP approximation with two days. Changing the reward signal for RL could be an idea to counter that: Apparently, it was problematic for RL to incur high losses in the beginning, so it rarely chose to move into this direction. If the buy strategy in the beginning did not yield such a high negative reward, RL could actually manage to use the whole battery for its trading instead of only the lower half.

Finally, we notice that the performance of all strategies on the added trees in comparison to the original trees was very similar and showed in all but two cases equal costs or even an outperformance of the results from the original trees. Consequently, the added trees with much faster calculation do indeed yield an alternative to the original method.

4.5 Conclusion

The objective of this chapter was to profitably market a battery in the German day ahead market. To achieve this goal, we first looked at the day ahead market in the period from 2015 to 2020 and analyzed whether characteristics attributed to spot prices in the literature also play a role in the period we considered. The results of this analysis yield the basis for our choice of a spot price model from the field of stochastic factor models.

Since the final calibrated model produces price trajectories with continuous components in time and space, the next step is to discretize the model. To do this, we introduce the approximation of continuous random variables by discrete random variables and then, based on this, the approximation of continuous processes by discrete processes using the nested distance. For an application to our electricity price model, we also discuss how the nested distance relates to scenario trees and present algorithms to generate optimal, computationally tractable scenario trees for the chosen electricity price model. One of these algorithms is the “adding” method, which yields a coarser approximation, but has significant speed advantages.

In our final theory chapter, stochastic programming, dynamic programming, and reinforcement learning as stochastic optimization algorithms are reviewed and related to each other.

We apply the introduced concepts to a real battery in Section 4.4. In order to test the concepts in two different time periods, one week in February and one week in August are chosen to optimize the battery deployment in the spot market. For this purpose, scenario trees are generated for the presented stochastic factor model for both periods using the presented methods. Before starting the optimization on these trees, we discuss conditions for the applicability of dynamic programming in our setting. We show that there are realistic battery properties for which dynamic programming is readily applicable. Furthermore, we show how, for given batteries and spot market prices, it is possible to test whether dynamic programming provides only an approximation to the optimal solution or actually the true optimum.

The four presented methods are then applied to two different battery types and the results are compared on each of the four scenario trees. We see that dynamic programming, even if it is applied in a setting where it does not necessarily yield the optimal result, gives results on par with the commonly used two-stage stochastic programming method and takes much less time to do so. Furthermore, we see that RL as well as

the DP approximation yield worse overall results compared to the classical optimization methods. Therefore, for this rather simple optimization problem, we recommend using the classical optimization variants. Compared to them, RL does not offer any advantage in our setting, neither in terms of results nor in terms of computation time, and the DP approximation has a computational time advantage in comparison to the two-stage stochastic programming, but not in comparison to the original DP. In any case though, all four methods achieve profits in the day ahead market and manage to take advantage of the possibility to sell flexibility profitably and in line with the feed-in profiles of renewable energies.

Finally, the results that the strategies achieve on the trees stemming from the tree addition method are equal or better than the results based on the original tree method except for two cases. Consequently, we do recommend this method for our presented setting.

Bibliography

[1] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2010.118.

[2] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954. doi: bams/1183519147.

[3] Fred Espen Benth, Jūratė Šaltytė Benth, and Steen Koekebakker. *Stochastic Modeling of Electricity and Related Markets*. Number 6811 in World Scientific Books. World Scientific Publishing Co. Pte. Ltd., April 2008. ISBN ARRAY(0x4f8c14d8). URL `https://ideas.repec.org/b/wsi/wsbook/6811.html`.

[4] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer New York, NY, 2 edition, June 2011. ISBN 9781461402374. doi: https://doi.org/10.1007/978-1-4614-0237-4.

[5] Bundesministerium für Wirtschaft und Technologie (BMWi). Energiekonzept für eine umweltschonende, zuverlässige und bezahlbare Energieversorgung, 2010. Available at `https://www.bmwi.de/Redaktion/DE/Downloads/E/energiekonzept-2010.pdf?__blob=publicationFile&v=5`, version: 2010.

[6] Bundesnetzagentur. Pressemitteilung: Bundesnetzagentur veröffentlicht Daten zum Strommarkt 2022, 2023. Available at `https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Allgemeines/Presse/Pressemitteilungen/2023/20230104_smard.pdf;jsessionid=F3B8B2EAC7ABA5C8447C9E26715C3CF1?__blob=publicationFile&v=3`, accessed: 2023-03-22.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 4 edition, 2022. ISBN 026204630X.

[8] George B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3-4):197–206, 1955. doi: 10.1287/mnsc.1.3-4.197.

[9] ENTSO-E. Developing balancing systems to facilitate the achievement of renewable energy goals, 2011. Available at `https://eepublicdownloads.entsoe.eu/clean-documents/pre2015/position_papers/111104_RESBalancing_final.pdf`, accessed: 2022-09-16.

[10] ENTSO-E. Tyndp 2022 – system needs study: System dynamic and operational challenges, 2023. Available at `https://eepublicdownloads.blob.core.`

windows.net/public-cdn-container/tyndp-documents/TYNDP2022/public/syst-dynamic-operational-challenges.pdf, accessed: 2022-03-19.

[11] EPEX SPOT. New id3-price index on continuous german intraday market, 2015. Available at https://www.epexspot.com/en/news/new-id3-price-index-continuous-german-intraday-market, accessed: 2021-07-26.

[12] EPEX SPOT. Epex spot to publish separate prices and volumes for the austrian and german day-ahead markets, 2018. Available at https://www.epexspot.com/sites/default/files/download_center_files/181001_EPEXSPOT_Press_Release_Split-DE-AT.pdf, accessed: 2023-03-14.

[13] EPEX SPOT. Epex spot history, 2019. Available at https://www.epexspot.com/en/10years, accessed: 2021-08-02.

[14] EPEX SPOT. Epex spot operational rules, 2021. Available at https://www.epexspot.com/sites/default/files/download_center_files/EPEXSPOTMarketRules_7.zip, version: 2021-06-16.

[15] EPEX SPOT. Epex spot annual market review 2020, 2021. Available at https://www.epexspot.com/sites/default/files/download_center_files/2021-01-14_EPEXSPOT_AnnualPressRelease-2020_final.pdf, accessed: 2021-10-10.

[16] EPEX SPOT. Trading on epex spot 2021, 2021. Available at https://www.epexspot.com/sites/default/files/download_center_files/21-03-09_TradingBrochure.pdf, accessed: 2021-07-26.

[17] EPEX SPOT. Day-ahead and intraday — the backbone of the european spot market, 2022. Available at https://www.epexspot.com/en/basicspowermarket#day-ahead-and-intraday-the-backbone-of-the-european-spot-market, accessed: 2022-09-14.

[18] EPEX SPOT. Epex spot annual market review 2022, 2023. Available at https://www.epexspot.com/sites/default/files/download_center_files/2023-01-192020EPEX20SPOT_Annual20Press20Release-2022_final.pdf, accessed: 2023-03-23.

[19] European Commodity Clearing AG. Clearing circular 14/2022 — increase of price limit on mrc day-ahead auctions, 2022. Available at https://www.ecc.de/fileadmin/Global/News/ECC/ECC_Circulars/2022/20220411_ECC_Clearing_Circular_No._14_Increase_of_Price_Limit_on_MRC_Day-Ahead_Auctions_01.pdf, accessed: 2023-03-14.

[20] Elisabeth Finhold, Christoph Gärtner, Ria Grindel, Till Heller, Neele Leithäuser, Elias Röger, and Florian Schirra. Optimizing the marketing of flexibility for a virtual battery in day-ahead and balancing markets: A rolling horizon case study. *Applied Energy*, 04 2023. In major revision.

[21] Fraunhofer ITWM. Technischer Anhang Monte Carlo. Technical document, 2016.

[22] Kirsting Ganz, Timo Kern, Louisa Wasmeier, and Serafin von Roon. Veränderungen der Merit Order und deren Auswirkungen auf den Strompreis, 2022. Available at `https://www.ffe.de/veroeffentlichungen/veraenderungen-der-merit-order-und-deren-auswirkungen-auf-den-strompreis/`, accessed: 2023-03-22.

[23] Hélyette Geman and Andrea Roncoroni. Understanding the fine structure of electricity prices. *The Journal of Business*, 79(3):1225–1261, 2006. ISSN 00219398, 15375374.

[24] Ria Grindel and Nikolaus Graf von Luckner. Forecasting of the ID3 using limit order book data. *SSRN eLibrary*, 2022. doi: 10.2139/ssrn.4017248.

[25] 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`.

[26] David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291, 1997. ISSN 0169-2070. doi: 10.1016/S0169-2070(96)00719-4.

[27] Holger Heitsch. Szenariobaumapproximation für stochastische Optimierungsprobleme in der Energiewirtschaft. *Optimierung in der Energiewirtschaft, VDI-Berichte 2080*, pages 45–60, 2009.

[28] Wieger J. Hinderks and Andreas Wagner. Pricing german energiewende products: Intraday cap/floor futures. *Energy Economics*, 81:287–296, 2019. ISSN 0140-9883. doi: 10.1016/j.eneco.2019.04.005.

[29] Wieger J. Hinderks and Andreas Wagner. Factor models in the german electricity market: Stylized facts, seasonality, and calibration. *Energy Economics*, 85:104351, 2020. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2019.03.024.

[30] Wieger Johan Hinderks, Andreas Wagner, and Prilly Oktoviany. Energy Risk Management in Practice. In Stéphane Goutte and Duc Khuong Nguyen, editors, *HANDBOOK OF ENERGY FINANCE Theories, Practices and Simulations*, World Scientific Book Chapters, chapter 14, pages 319–341. World Scientific Publishing Co. Pte. Ltd., June 2020. URL `https://ideas.repec.org/h/wsi/wschap/9789813278387_0014.html`.

[31] Lion Hirth and Ranga Yogeshwar. Licht aus, Preise rauf: Wie funktioniert der Energiemarkt?, 2022. URL https://www.youtube.com/watch?v=mpbWQbkl8_g#t=20m15s.

[32] Markéta Horejšová, Sebastiano Vitali, Miloš Kopa, and Vittorio Moriggia. Evaluation of scenario reduction algorithms with nested distance. *Computational Management Science*, 17(2):241–275, 2020. doi: 10.1007/s10287-020-00375-.

[33] IBM. Timestamp arithmetic, 2021. Available at https://www.ibm.com/docs/en/i/7.1?topic=sql-timestamp-arithmetic, accessed: 2023-03-24.

[34] *German Intraday Electricity Market Analysis and Modeling Based on the Limit Order Book*, 2018. Institute of Electrical and Electronics Engineers, IEEE.

[35] Fraunhofer ISE. Installierte Netto-Leistung zur Stromerzeugung in Deutschland in 2023, 2023. Available at https://www.energy-charts.info/charts/installed_power/chart.html=de&c=DE&chartColumnSorting=default&legendItems=1000111111111, accessed: 2023-03-22.

[36] Tim Janke and Florian Steinke. Forecasting the price distribution of continuous intraday electricity trading. *Energies*, 12:4262, 11 2019. doi: 10.3390/en12224262.

[37] Kirui Kipngeno, Georg Pflug, and Alois Pichler. New algorithms and fast implementations to approximate stochastic processes. *arXiv e-prints*, 12 2020. doi: 10.48550/arXiv.2012.01185.

[38] Christopher Koch and Lion Hirth. Short-term electricity trading for system balancing: An empirical analysis of the role of intraday trading in balancing germany's electricity system. *Renewable and Sustainable Energy Reviews*, 113:109275, 2019. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2019.109275. URL https://www.sciencedirect.com/science/article/pii/S1364032119304836.

[39] Emil Kraft, Marianna Russo, Dogan Keles, and Valentin Bertsch. Stochastic optimization of trading strategies in sequential electricity markets. *European Journal of Operational Research*, 308(1):400–421, 2022. ISSN 0377-2217. doi: 10.1016/j.ejor.2022.10.040.

[40] Marcel Kremer, Rüdiger Kiesel, and Florentina Paraschiv. Intraday electricity pricing of night contracts. *Energies*, 13(17), 2020. ISSN 1996-1073. doi: 10.3390/en13174501.

[41] Jonathan Law and John Smullen. *A Dictionary of Finance and Banking*. Oxford University Press, 2008. ISBN 9780191726668. doi: 10.1093/acref/9780199229741.001.0001.

[42] Can Li and Ignacio E. Grossmann. A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering*, 2, 2021. ISSN 2673-2718. doi: 10.3389/fceng.2020.622241.

[43] Katarzyna Maciejowska, Weronika Nitka, and Tomasz Weron. Day-ahead vs. intraday—forecasting the price spread to maximize economic benefits. *Energies*, 12(4), 2019. ISSN 1996-1073. doi: 10.3390/en12040631.

[44] Katarzyna Maciejowska, Bartosz Uniejewski, and Tomasz Serafin. Pca forecast averaging—predicting day-ahead and intraday electricity prices. *Energies*, 13(14), 2020. ISSN 1996-1073. doi: 10.3390/en13143530.

[45] Grzegorz Marcjasz, Bartosz Uniejewski, and Rafał Weron. Beating the naive—combining lasso with naive intraday electricity price forecasts. *Energies*, 13(7), 2020. ISSN 1996-1073. doi: 10.3390/en13071667.

[46] Thilo Meyer-Brandis and Peter Tankov. Multi-factor jump-diffusion models of electricity prices. *International Journal of Theoretical and Applied Finance*, 11(05): 503–528, 2008. doi: 10.1142/S0219024908004907.

[47] Michał Narajewski and Florian Ziel. Econometric modelling and forecasting of intraday electricity prices. *Journal of Commodity Markets*, 19:100107, 2020. ISSN 2405-8513. doi: 10.1016/j.jcomm.2019.100107.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] Georg Ch. Pflug. Version-independence and nested distributions in multistage stochastic optimization. *SIAM Journal on Optimization*, 20(3):269–293, 2010. doi: 10.1137/080718401.

[50] Georg Ch. Pflug and Alois Pichler. *Multistage Stochastic Optimization*. Springer, Cham, 2014. ISBN 978-3-319-38267-8. doi: 10.1007/978-3-319-08843-3.

[51] Georg Ch. Pflug and Alois Pichler. Dynamic generation of scenario trees. *Computational Optimization and Applications*, 62:641–668, 2015. doi: 10.1007/s10589-015-9758-0.

[52] Alois Pichler and Michael Weinhardt. The nested sinkhorn divergence to learn the nested distance. *Computational Management Science*, 19:1619–6988, 2022. doi: 10.1007/s10287-021-00415-7.

[53] Warren Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2nd edition, 08 2011. ISBN 9780470604458. doi: 10.1002/9781118029176.

[54] Warren Powell. *Clearing the Jungle of Stochastic Optimization*, pages 109–137. INFORMS, 2014. ISBN 978-0-9843378-5-9. doi: 10.1287/educ.2014.0128.

[55] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL `https://www.R-project.org/`.

[56] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[57] Eduardo S. Schwartz. The stochastic behavior of commodity prices: Implications for valuation and hedging. *The Journal of Finance*, 52(3):923–973, 1997. doi: https://doi.org/10.1111/j.1540-6261.1997.tb02721.x.

[58] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[59] Umweltbundesamt. Erneuerbare Energien in Deutschland – Daten zur Entwicklung im Jahr 2020, 2021. ISSN 2363-829X. Available at `https://www.umweltbundesamt.de/sites/default/files/medien/5750/publikationen/2021_hgp_erneuerbareenergien_deutsch_bf.pdf`, accessed 2021-08-12.

[60] Bartosz Uniejewski, Grzegorz Marcjasz, and Rafał Weron. Understanding intraday electricity markets: Variable selection and very short-term price forecasting using lasso. *International Journal of Forecasting*, 35(4):1533–1547, 2019. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.02.001.

[61] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[62] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN 9780821833124. URL `https://books.google.de/books?id=idyFAwAAQBAJ`.

[63] Andreas Wagner, Enislay Ramentol, Florian Schirra, and Hendrik Michaeli. Short- and long-term forecasting of electricity prices using embedding of calendar information in neural networks. *arXiv e-prints*, 2020. doi: 10.48550/ARXIV.2007.13530.

[64] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021.

[65] 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`.

[66] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

[67] Marcus Willems. Quantization and scenario trees and their application in finance, 2021. Master thesis, RPTU Kaiserslautern.

[68] Rolf Wüstenhagen and Michael Bilharz. Green energy market development in germany: effective public policy and emerging customer demand. *Energy Policy*, 34(13):1681–1696, 2006. ISSN 0301-4215. doi: https://doi.org/10.1016/j.enpol.2004.07.013. URL https://www.sciencedirect.com/science/article/pii/S030142150400237X.

[69] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006. doi: https://doi.org/10.1111/j.1467-9868.2005.00532.x.

[70] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67 (2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.

[71] Stig Ødegaard Ottesen, Asgeir Tomasgard, and Stein-Erik Fleten. Multi market bidding strategies for demand side flexibility aggregators in electricity markets. *Energy*, 149:120–134, 2018. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2018.01.187.

Lebenslauf

2020 - 2023	Doktorandin an der Reinland-Pfälzischen Technischen Universität Kaiserslautern Landau am Standort Kaiserslautern, DE
2020 - 2023	Hilfswissenschaftlerin der Abteilung Finanzmathematik am Fraunhofer Institut für Techno- und Wirtschaftsmathematik in Kaiserslautern, DE
2017 - 2020	Wissenschaftliche Mitarbeiterin der Abteilung Finanzmathematik am Fraunhofer Institut für Techno- und Wirtschaftsmathematik in Kaiserslautern, DE
2016 - 2017	Hilfswissenschaftlerin am Lehrstuhl für Stochastik und ihre Anwendungen der Universität Mannheim, DE
2014 - 2017	M. Sc. Wirtschaftsmathematik mit Schwerpunkt Finance an der Universität Mannheim, DE
2014 - 2015	Hilfswissenschaftlerin an der Graduate School for Economics and Social Sciences in Mannheim, DE
2011 - 2014	B. Sc. Wirtschaftsmathematik mit Schwerpunkt Betriebswirtschaftslehre an der Universität Mannheim, DE
2003 - 2011	Allgemeine Hochschulreife am Egbert-Gymnasium-Münsterschwarzach, DE

Scientific Career

2020 - 2023	Doctoral student at the Rhineland-Palatine Technical University Kaiserslautern Landau in Kaiserslautern, GE
2020 - 2023	Research Assistant at the Department of Financial Mathematics at the Fraunhofer Institute for Industrial Mathematics in Kaiserslautern, GE
2017 - 2020	Scientific Employee at the Department of Financial Mathematics at the Fraunhofer Institute for Industrial Mathematics in Kaiserslautern, GE
2016 - 2017	Research Assistant at the Department of Stochastics and its Applications, University of Mannheim, GE
2014 - 2017	M. Sc. Mathematics in Business and Economics with focus on Finance at the University of Mannheim, GE
2014 - 2015	Research Assistant at the Graduate School for Economics and Social Sciences in Mannheim, GE
2011 - 2014	B. Sc. Mathematics in Business and Economics with focus on business administration at the University of Mannheim, GE
2003 - 2011	General university entrance qualification at the Egbert-Gymnasium-Münsterschwarzach, GE