Valuing Cross-Border Capacity as a Real Option

An IOHMM Approach

Mats Leenders

Pricing cross-border capacity
for the Germany-France
and Netherlands-Germany borders

Valuing Cross-Border Capacity as a Real Option

An IOHMM Approach

by

Mats Leenders

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday June 27, 2022 at 9:00 AM.

Student number:	4548140	
Project duration:	July 1, 2021 – June 27, 2022	
Thesis committee:	Prof. dr. ir. A. Papapantoleon,	TU Delft, supervisor
	Prof. dr. ir. K. Vuik,	TU Delft
	Ir. V. Kotzev,	Northpool B.V.

Preface

The work before you is written for the purpose of completing the Master of Applied Mathematics at the Delft University of Technology. It is my attempt to create a model that can find a fair value for the cross-border capacity for the Germany-France and the Netherlands-Germany borders. The research was conducted together with Northpool B.V.

I am thankful to Northpool B.V. and the TU Delft for the opportunity to research this topic. I would like to thank my supervisors Vassil and Teun from Northpool, and prof. Papapantoleon from the Applied Probability group at the TU Delft for their help; without your advice and patience, this work would not have come to fruition.

I would also like to thank my roommates in the House of Fortune, Bas, Brian, Gavin, Bart, and Cedric, as well as my girlfriend Marije. Without you this would not have been achievable either; this applies not only to the thesis before you, but to my entire study at the TU Delft.

Finally, and most importantly, I would like to thank my parents and my brother for their never ending support.

I am glad the reader has taken the time to read my thesis, and I hope you enjoy it.

Mats Leenders
Delft, June 2022

Abstract

The right to use a certain amount of capacity in an electrical cable between two countries for the purpose of trading energy is an asset that can be bought. Each hour of capacity can be seen as a real spread option with the energy prices of each country being the underlying processes. In this thesis we build a model to find the fair value of a month of cross-border capacity and use it to predict this value for the Germany-France and Netherlands-Germany border for the months of July-October 2021. Our model is implemented as an IOHMM with 4 states following a multinomial distribution and a non-stationary Normal Inverse Gaussian distribution for each output distribution. We use SPOT prices of each bordering country to create a 'spread process', which are then used as target values. As inputs various market drivers for each respective country were used. The model produces a mixture probability distribution for the spread for each hour of the month. Taking the discounted expected value then gives the hourly capacity value; certain Greek calculation is also possible. The results show that the capacities follow a comparable trend to the market. The predicted values for September and October fall short of the realized prices; we expect this to be due to unprecedented volatility in the energy market. Finally, we tested whether certain key drivers were mapped to the spreads as one would expect; this was the case for the Dutch and French drivers, but not for the German drivers.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In the last two decades, the distribution and sale of electricity has gone under major deregulation all around the globe. Where previously markets were controlled in a mostly deterministic way, electricity prices have recently become subject to the forces of the market [5, 9, 19]. This makes the pricing of electricity and electricity derivatives more complicated than in previous years [17]. Furthermore, the introduction of market dynamics to the electricity market brings the question of a ’fair’ price for such derivatives. This is the question we concern ourselves with for a specific type of electricity derivative in this thesis.

Say we have an electricity trader that wishes to trade between two different countries. These countries have separately organized electricity networks, however there exists a cable that physically connects these networks. This cable has the physical constraint of a maximum amount of capacity that can flow through it at any one time. This scarcity ensures that the capacity has a certain amount of value. The question this thesis aims to answer is, what premium should the trader pay for the use of the capacity in the cable? In practice the right to use this cable is auctioned by the Joint Allocation Office (JAO). The use of a certain amount of capacity in the cable can then be assigned through the auction for certain blocks, for example weeks or months [31, 32]. However, as we will see, this process does not necessarily show the true value of the asset. So then, how do we find this true value?

Naturally this problem is multifaceted. We immediately notice that as the trader plans on making a profit or loss with regards to the price of electricity in both countries, the premium must be related in some form to these prices. We can therefore see this capacity as a derivative of the electricity prices. This framework gives rise to two areas which can be explored: electricity pricing and derivative pricing. There are a variety of ways that electricity prices are calibrated and subsequently predicted [5, 9]. One such approach assumes the electricity prices follow a stochastic process of some kind; as will be shown, the capacity can be seen as a real spread option on the electricity price of the two countries. This leads to the world of Black-Scholes option pricing [34], and subsequently the Margrabe equation [29]. While we do review this method, we will also take a different approach.

In practice, the Margrabe method requires certain parameters of the price process to be calibrated to the actual electricity prices; however, due to illiquidity in the market this can be a difficult task [5]. A trader would then usually use experience and heuristics to analyse certain driving factors in the electricity market and estimate the parameters. We wish to replicate this process through some kind of quantifiable criterion. To do this, we analyse the electricity market in France, Germany, and the Netherlands to see which types of data 'drive' the prices; for example, daily wind, solar, or gas production each have a certain effect on daily electricity prices. We then employ machine learning algorithms in order to map these key market variables to the electricity prices. In doing this we create a different type of non-stationary price process than that of the Black-Scholes framework; however we can still use this price process to find an option value. This option value then represents the fair value of our cross-border capacity.

In this thesis we attempt to find a fair price for the monthly cross-border capacities between Germany and France, and the Netherlands and Germany. We first give a succinct overview of the electricity market in Europe, and so in the French, German, and Dutch markets. We define the cross-border capacity as a real option, and explore how this could be priced through the Black-Scholes framework. We then continue by defining our own model, a variant of an Input-Output Hidden Markov Model. We first define the model theoretically. We do this by showing the mathematics behind the machine learning principles used, as well by defining the Normal Inverse Gaussian distribution that was used within the model. We continue by showing implementation considerations specific to our problem. Given a trained model, we show the results of predicting the capacities for the months of July to October 2021. Finally, we discuss these results and any further research to be done in the area.

2
Background

In this chapter we discuss all the necessary background information regarding the problem this thesis attempts to solve. We start with an overview of the mechanics of the modern day electricity markets and the various parties involved. This leads to a more rigorous definition of the cross-border capacity that we wish to value. Using this definition, we show how this asset can be seen as a real option, from which an overview of the Black-Scholes option pricing framework follows naturally. We show the extension of the Black-Scholes framework to spread option to include the Margrabe formula. Finally, we give an overview of machine learning and how it operates. These tools will be used in subsequent chapters for the valuation of the cross-border capacity.

2.1. Electricity Markets

In this thesis we focus on the modelling of spot prices in European countries in order to value a type of electricity derivative also taking place in Europe. Therefore an overview of the electricity markets in this region seems appropriate. Throughout the following chapter, the words electricity and power will be used interchangably.

2.1.1. Various Types of Markets

The main markets where electricity is traded in Europe that we are interested in are the 'spot market', the 'forward market', and the 'IntraDay market' [5]. The reason for the existence of so many markets regarding a single commodity lies in an important quality that most other commodities do not possess: electricity cannot be stored [17]. These markets work together to still provide all the participants with electricity around the clock. The main market is the spot market, on which electricity is sold for the day ahead. However, due to the non-storability of electricity, operators of the electricity grid often need to trade electricity in real time. This is what the IntraDay market does. Furthermore, the forward market serves as a way of hedging the spot market, however due to the non-storability issue, this market is often sparse and quite illiquid. Naturally there are more ways in which electricity is traded in Europe, however seeing that these are not as relevant to this thesis as the markets mentioned above, these are left for the reader to explore [5].

2.1.2. Spot Markets in Europe

We now take a closer look at the spot market for electricity in Europe, focusing specifically on the principle of market coupling. As stated before, participants in the spot market trade electricity for the day ahead. Market coupling is defined as 'the merging of individual, national markets to render possible the trade of electricity across a large geographical area' [38]. All EU countries follow the marginal costs system, whereby in the absence of market coupling, the country with the highest marginal cost has the highest price for a specified hour in the day ahead market. The last 10 years, the European Union has been working towards a fully coupled electricity market between all of its member countries. The reason behind this is to reduce congestion rates and making transaction costs economically efficient, ultimately improving overall welfare for consumers and producers of electricity [23]. However, this fully coupled market is not quite available yet.

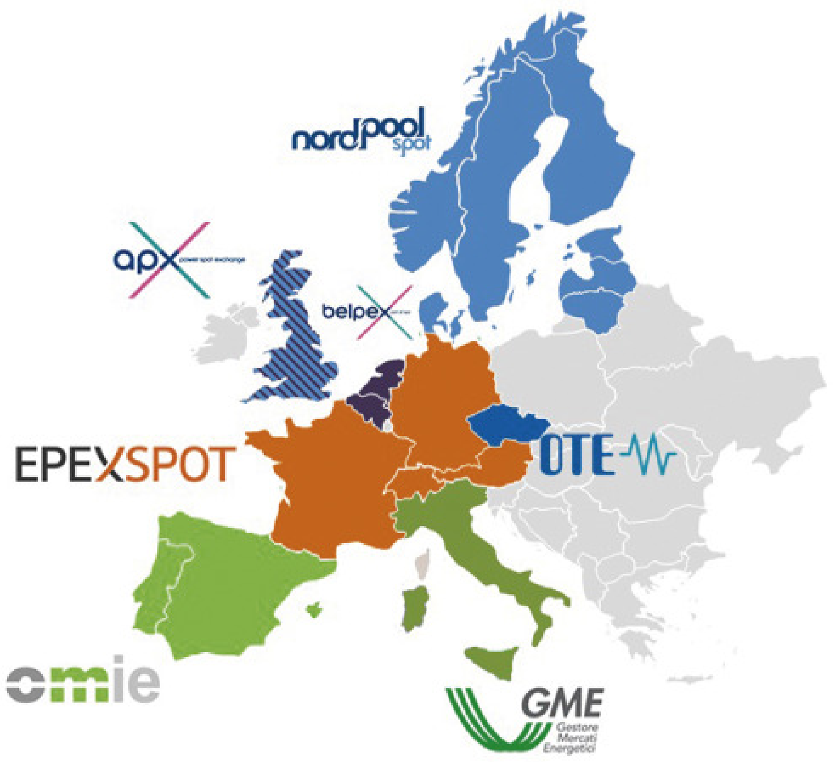


Figure 2.1: The Day Ahead (Spot) Markets for the European Union [33].

In Figure 2.1 the current spot markets for electricity are displayed. Clearly some countries are already participating in a fully coupled market, for example the Nord Pool exchange in Scandinavia. This means that in those regions, there is a free market for electricity between all contained geographical areas, assuming the electricity network restriction are respected. A market participant simply has to input an order with their power exchange and they take care of the transmission capacity to carry out the transaction [23].

So how does trading between the fully coupled markets in Figure 2.1 work? Let us first look at the case in which these markets are actually coupled. The majority of the European power exchanges have agreed to carry out an implicit auction everyday for the delivery of electricity in the next day, in which they decide which markets will be coupled for which hours of the day [23]. This is done using an algorithm called EUPHEMIA, which aims to maximize consumer and producer welfare while simultaneously minimizing the congestion rent of the entire electricity network [10]. In the case that EUPHEMIA couples the markets, the cross-border transmission capacities can be used freely to to arbitrage away any price difference for every hour of the day, as in a fully coupled system described before. For more information on how the EUPHEMIA algorithm works, see [10]. For more information regarding the benefits of market coupling, see [23].

Naturally it also occurs that certain markets are not coupled for certain hours of the day. In this case, trading between independently fully coupled markets is a bit more difficult. The Transmission System Operators (TSOs) between the countries then only attempts to optimize their own grids. Furthermore, there is a physical limitation to the amount of capacity that can flow through the cross-border transmission cables between these markets that is not being optimized for one network. This means that there are additional costs that need to be accounted for when using these cables [23, 38]. The combination of the limited cross-border capacity and the differences in the prices of marginal costs of electricity between countries is the main factor leading to a different price of electricity between countries. In general, this leads to inefficiencies for all market participants, either through increased congestion or increased costs. For more information, see [38].

2.1.3. Forward Markets in Europe

We now take a closer look at the forward market for electricity in Europe, focusing on the mechanics behind the cross-border capacities briefly described for the spot market. Participants in the forward market trade in electricity contracts for the longer term, which we define as a minimum of a week and a maximum of a year [37]. Unlike the spot market, there is no EUPHEMIA algorithm to increase the coupling of markets; however the forward market can be seen as a prediction of the EUPHEMIA algorithm, as these are the prices that will eventually settle as the spot prices. This means that there is a separate mechanism for managing the cross-border transmission capacities between the standard markets. This is done by the Joint Allocation Office (JAO), through a form of auction [18, 32]. This works as follows. The cross-border capacities are auctioned off in blocks of months, quarters or years, depending on the particular border. For a block of, for example, a month, there is a certain amount of capacity (given in Megawatt hours, Mwh) that is available for every hour of that month. Participants in the auction give bids for how much they would be willing to pay for 1 Mwh and how many Mwh they would pay at this price. Different bids for different amounts of Mwh for a single participant are admitted. JAO then takes the largest of all combined bids, and allocates the requested MWh to the appropriate participant; then the amount of MWh with the second largest bid is allocated to appropriate participant. This continues until all the available capacity in the transmission cable is allocated; the price of the final allocated capacity is then the price that all the participants pay for their allocated capacities. To summarize, at the end of the auction, all participants that had capacity allocated have this capacity for every hour of the auctioned month, and all participants pay the lowest allocated price for each Mwh of capacity. The auctions for the week, quarter, and year capacities are performed in a similar manner. The EU has approved harmonized rules, which define two types of forward capacity rights: “Financial Transmission Right Option” and “Physical Transmission Right”. The former provides a full option to the owner by guaranteeing the payment of the positive difference in the prices for the respective border for every hour in the day ahead auction. The latter guarantees only the right to transfer electricity across border through the physical cable [18, 32] Given that a participant in the forward market then wishes to perform a cross-border trade, it can do so with the allocated capacity awarded in the auction [37]. For more information about the regulations and specific products of these auctions, see [18, 31, 32].

2.1.4. Cross-Border Capacity

We can now define the product that we wish to price.

Definition 1 *The* cross-border capacity *between two markets (usually countries) is the amount of Mwh that can flow through the transmission cable from one network to the other.*

In this thesis we focus on the monthly capacity for the border between Germany and France and the border between The Netherlands and Germany; both of these border fall in the "Financial Transmission Right Options" category [18, 32]. Keeping in mind the auction process previously described, we wish to find the market value of 1 Mwh of capacity for an entire month for these particular borders. We do this by assuming the cross-border capacity is a real option and so incorporate techniques from financial mathematics and machine learning to price it.

2.2. Real Options

We show that the cross-border transmission capacity described in the previous section can be seen as an example of a real option.

Definition 2 *A* real option *is a type of investment that gives the buyer the right, but not obligation, to undertake some kind of business decision at a later time.*

The main difference between real options and financial options is that financial options are usually traded securities based on some explicit underlying financial product, while real options do not have this restriction [27].

Real options often rely on some physical conditions, as is indeed the case with the cross-border capacity, as this capacity has physical restrictions which are creating its worth. Let us assume a German power trader wishes to trade on the forward market in France, for a specific hour one month from now. The idea the trader has, is that if the price of electricity for that hour is higher in France than in Germany, buying in Germany and selling in France would give him the difference in price as profit. The trader knows however that if he is wrong, selling German electricity in France will make him lose money, in which case he will simply stay in the German market. In order to do this, the trader would have to participate in the monthly auction for this particular border, and assuming he bids high enough, pay a premium for using the transmission cable. Therefore, we see that the trader pays a premium now, giving him the right, but not obligation, to trade between the German and French markets for the specific hour he requires. Thus the cross-border capacity, and more specifically the “Financial Transmission Rights Option”, can be seen as a real option.

There are some specifics that should be mentioned in this framework, as the auction from JAO imposes some restrictions. We see that the monthly auction that we wish to value is in fact not a single option but a bundle of options. When the trader wins the auction and is allocated the monthly capacity, he or she can choose the individual hours of the month that they wish to trade at a later date; the trader may choose to do this as different hours may be more profitable than others. Therefore, each hour is a separate real option, and the monthly auction we value is the sum of these options. Furthermore, a physical limitation of the transmission cable is that the electricity 'flows' in one directions; this means that there is a separate auction for power flowing from Germany to France and from France to Germany [18]. In this thesis we aim to value these 'flows' independently, as this is more applicable to trading strategy.

We see that as the cross-border capacity is a bundle of real options, we can price it as such. The 'physical investment' we are valuing is the right to use the transmission cable in a specific direction for every hour of a specific month. The underlying asset in this case is the difference (or spread) between the price of electricity in the two bordering countries, as this is where the trader earns their profit. Ideally, the forward market prices would be used for the underlying. However, as stated earlier, the forward market suffers from illiquidity, meaning that a true 'market price' for every hour or even every day is quite impossible to obtain. Therefore instead of forward market prices, we use the spot market prices as an underlying asset. As is shown in [37], this substitution is justified as spot markets show close proximity to forward markets and are deemed the 'driving factor' of the electricity markets as a whole.

2.3. Option Valuation

One method of pricing real options is by using the Black-Scholes model that is typically used for financial options [11]. In this section we briefly describe the general Black-Scholes framework and show the extension to the use of spread options. Certain Greeks are also shown in this setting. It should be noted that this method of pricing the cross-border capacity is one of the methods that our final model will be compared to.

2.3.1. The Black-Scholes Model

The Black-Scholes model can be applied to real options by assuming that the real option is in fact a financial option on the underlying asset, in this case the price of electricity. A European call option on an underlying asset $S(t)$ is a contract that gives the holder the right to buy a certain amount of the asset for a price K (referred to as the strike price), at a predefined later time T (referred to as the expiry time). The pay-off function of a European call option is defined as

$$H(T,S) = \max(S(T) - K, 0) \tag{2.1}$$

Now we let the $\mathscr{F} = (\mathscr{F}_t)_{t\in[0,T]}$, be the filtration generated by the sequence $S(t)$, i.e. $\mathscr{F}_t := \sigma(S_k \mid k \leq t)$ are σ-algebras and so $\mathscr{F}(T)$ contains all the information generated by $S(t)$ up to time T. Then we define the time t value of the European call option to be the discounted conditional expectation

$$V(t,S) = e^{-r(T-t)}\mathbb{E}[H(T,S)|\mathscr{F}(T)] \tag{2.2}$$

where $e^{-r(T-t)}$ is the discount factor, with r being the interest rate [34].

The Black-Scholes model assumes that the underlying asset follows a Geometric Brownian Motion (GBM), which is a stochastic process defined by

$$\mathrm{d}S(t) = \mu S(t)\mathrm{d}t + \sigma S(t)\mathrm{d}W^{\mathbb{P}}(t) \tag{2.3}$$

where μ is the the drift, σ is the volatility, and $W^{\mathbb{P}}(t)$ is a Wiener Process under some real world measure $\mathbb{P}$. Now the Black-Scholes model assumes the options price $V(t,S)$ is in fact a function that solves for the Black-Scholes partial differential equation

$$\frac{\partial V}{\partial t} + r\frac{\partial V}{\partial X} + \frac{1}{2}\sigma^2\left(-\frac{\partial V}{\partial X} + \frac{\partial^2 V}{\partial X^2}\right) - rV = 0 \tag{2.4}$$

together with the terminal condition

$$V(S,T) = H(T,S)$$

There are several ways to solve this PDE, some methods are explained in [34]. For a European call option, this PDE can be solved in closed form, giving the option price

$$V_c(t,S) = S(t)\Phi(d_1) - K\mathrm{e}^{-r(T-t)}\Phi(d_2), \tag{2.5}$$

where

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\log\frac{S(t)}{K} + \left(r + \tfrac{1}{2}\sigma^2\right)(T-t), \\ d_2 &= \frac{1}{\sigma\sqrt{T-t}}\log\frac{S(t)}{K} + \left(r - \frac{1}{2}\sigma^2\right)(T-t) = d_1 - \sigma\sqrt{T-t}, \end{aligned} \tag{2.6}$$

and Φ the cumulative distribution function of a Standard Normal random variable [34].

The Greeks of a financial option measure how sensitive the price is to its own parameters; these values can be helpful for traders to hedge their portfolios. In the Black-Scholes model, the Greeks for a European call option can be found in closed form [34]. The Greeks in regards to the asset price S, the time until maturity T, and the volatility σ are tabulated in Table 2.1. For more information regarding the derivation and use of the Greeks, see [34].

Name	**Symbol**	**Derivative**	**Value**
Delta	Δ	$\frac{\partial V}{\partial S}$	$\Phi(d_1)$
Gamma	Γ	$\frac{\partial^2 V}{\partial S^2}$	$\frac{\Phi'(d_1)}{S\sigma\sqrt{T-t}}$
Vega	ν	$\frac{\partial V}{\partial \sigma}$	$S\Phi(d_1)\sqrt{T-t}$
Theta	Θ	$\frac{\partial V}{\partial t}$	$-\frac{S\Phi'(d_1)\sigma}{2\sqrt{T-t}} - rKe^{-r(T-t)}\Phi(d_2)$

Table 2.1: Selected Greeks for a European call option.

2.3.2. The Margrabe Formula

We now show how the Black-Scholes model can be extended to a financial option that is based on the spread between two assets.

Definition 3 *An* exchange option *(or a* spread option*) is a financial option that, at maturity, gives the holder the right but not obligation to exchange one risky asset for another*

We assume that our two assets, $S_1(t)$ and $S_2(t)$, follow a correlated GBM

$$\begin{aligned} \mathrm{d}S_1(t) &= \mu_1 S_1(t)\mathrm{d}t + \sigma_1 S_1(t)\mathrm{d}W_1^{\mathbb{P}}(t),\\ \mathrm{d}S_2(t) &= \mu_2 S_2(t)\mathrm{d}t + \sigma_2 S_2(t)\mathrm{d}W_2^{\mathbb{P}}(t),\\ \mathrm{d}W_1\mathrm{d}W_2 &= \rho \mathrm{d}t, \end{aligned} \tag{2.7}$$

where ρ is the correlation coefficient between the two assets [36]. We can see these assets as the price of electricity in countries with markets that are not coupled. We assume that the holder of the option also has possession of asset $S_2(t)$, meaning that at maturity they have the right to exchange $S_2(t)$ for $S_1(t)$. In the case of $S_1(T) > S_2(T)$, the holder can make a profit if the exchange is performed. Similarly, the holder of the option has no reason to perform the exchange in the case that $S_2(T) > S_1(T)$. Therefore we see that at maturity, the option has a pay-off function of

$$H(T, S_1, S_2) = \max(S_1(T) - S_2(T), 0). \tag{2.8}$$

The option value is then described by the conditional expectation as before

$$V(t, S_1, S_2) = e^{-r(T-t)}\mathbb{E}[H(T, S_1, S_2)|\mathscr{F}(T)]. \tag{2.9}$$

In this sense it is also an option based on the spread between $S_1(t)$ and $S_2(t)$ [29, 36]. Now in the Black-Scholes framework, assuming a constant interest rate of r, the price of this option at time t is

$$V(S_1, S_2, t) = S_1(t)\Phi(d_1) - S_2(t)\Phi(d_2). \tag{2.10}$$

with

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\log\left(\frac{S_1(t)}{S_2(t)}\right) + \frac{\sigma^2}{2}(T-t)\right]\\ d_2 &= \frac{1}{\sigma\sqrt{T-t}}\left[\log\left(\frac{S_1(t)}{S_2(t)}\right) - \frac{\sigma^2}{2}(T-t)\right] = d_1 - \sigma\sqrt{T-t}\\ \sigma &= \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2} \end{aligned} \tag{2.11}$$

and Φ the cumulative distribution function of a Standard Normal random variable [29, 36]. Formula (2.10) is known as the Margrabe formula [29]. We see that this option value is the value of the cross-border capacity in the Black-Scholes framework.

The Margrabe formula also has Greeks corresponding to the assets' parameters [36]. The Greeks at time 0 corresponding to the first asset price S_1 and volatility σ_1, the second asset price S_2 and volatility σ_2, and the time until maturity T have been tabulated in Table 2.2, where $\phi(x)$ is the probability density function of a Standard Normal variable. Derivation of these Greeks can be found in Appendix A.

Name	Sign	Derivative	Value
Delta 1	Δ_1	$\frac{\partial V}{\partial S_1}$	$\Phi(d_1)$
Delta 2	Δ_2	$\frac{\partial V}{\partial S_2}$	$-\Phi(d_2)$
Gamma 11	Γ_{11}	$\frac{\partial^2 V}{\partial S_1^2}$	$\frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_1)}{S_1(t)}$
Gamma 22	Γ_{22}	$\frac{\partial^2 V}{\partial S_2^2}$	$\frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_2)}{S_2(t)}$
Gamma 12	Γ_{12}	$\frac{\partial^2 V}{\partial S_1 S_2}$	$-\frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_1)}{S_2(t)}$
Vega 1	ν_1	$\frac{\partial V}{\partial \sigma_1}$	$S_1(t)\sqrt{T-t}\phi(d_1)\frac{\sigma_1-\rho\sigma_2}{\sigma}$
Vega 2	ν_2	$\frac{\partial V}{\partial \sigma_2}$	$S_1(t)\sqrt{T-t}\phi(d_1)\frac{\sigma_2-\rho\sigma_1}{\sigma}$
Theta	Θ	$\frac{\partial V}{\partial t}$	$\frac{S_1(t)\sigma}{2\sqrt{T-t}}\phi(d_1)$

Table 2.2: Selected Greeks for a spread call option.

The Margrabe formula, while an excellent starting point for the valuation of cross-border capacity, does have a few deficiencies. First of all, seeing that it is based on the Black-Scholes model, it is bound to suffer from the volatility smile, which occurs through the parameterization of the underlying assets. One of these parameters, the volatility σ, cannot be directly inferred and so must be implied through market prices. However, as has been noted many times, the Black-Scholes implied volatility usually does not conform with market prices, leading to the possibility of inaccuracies in the valuation of the options [34]. It is natural to assume that this deficiency should also carry through to the Margrabe formula used on power markets. For this thesis, alternative stochastic processes were first considered to solve the issue of the Black-Scholes implied volatility; eventually however, a different route was chosen. For more information regarding such processes as applied to spread options, see [6, 12, 16, 24, 26, 35]

The second deficiency however, shows that we cannot even know this for certain. Due to the illiquidity of the forward markets and the low number of auctions, there is not a fully operated market to which this implied volatility can be compared [5]. This means that there is no direct numerical method to find the volatility parameter needed for the valuation. Common practice for traders is that the volatility is instead inferred from market data in different markets, usually on a heuristic basis. This is one of the deficiencies that the model in this thesis hopes to circumvent, and the reason that using different stochastic processes within the Black-Scholes framework was rejected. Even if the implied volatility could be solved, the market data that would be necessary to calibrate such processes is not available. We found that different types of data needed to be applied to the market prices that were available, which could be done through neural networks as will be shown in later chapters.

2.4. Machine Learning

In this model we wish to find a way to map certain data from other markets to the data of the market in which we find our real option. In this way we hope to feed the information that a trader would use to estimate certain parameters of the model directly to it. We achieve this through a type of machine learning model, which will be discussed in the next chapter. Here we give an introduction to the mechanics of machine learning.

2.4.1. Components of a Machine Learning Algorithm

A machine learning algorithm can often be generalized as a combination of a problem that needs to be solved, a cost function describing the problem, a model used to solve the problem, a dataset with information from the problem, and an optimization technique used to optimize the cost function. These problems can be split up in supervised or unsupervised learning. We are interested in supervised learning:

Definition 4 *A* supervised learning algorithm *is an algorithm that attempts to map a training set of inputs* $\mathbf{x}$ *to specific outputs* $\mathbf{y}$.

By contrast, an unsupervised learning algorithm does not have realized outputs $\mathbf{y}$ to directly map to. The most common cost function that comes with this type of learning is the negative log-likelihood function, which the model should try to minimized through the optimization technique. The chosen optimization technique is heavily dependent on the type of model that is used. Should a linear model be employed, the optimization technique can often find a closed form solution to minimizing the cost function. However, in the case of non-linear models, a different approach is needed [22].

2.4.2. Stochastic Gradient Descent

The most popular method of optimizing in a non-linear environment is through Stochastic Gradient Descent (SGD) (or some variation of SGD). As previously stated, optimizing is usually done by minimizing (or maximizing) some cost function for the training data; this is done using gradient descent, which we define below.

Definition 5 *The* gradient *of a (cost) function* $f(\mathbf{x})$*, denoted* $\nabla_{\mathbf{x}} f(\mathbf{x})$*, is the vector containing all of the partial derivatives of* f.

Definition 6 *The method of* (stochastic) gradient descent *proposes that we minimize* f *through using the gradient of* f *together with a* learning rate ϵ_t *to iterate to a solution:*

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \epsilon_t \nabla_{\mathbf{x}} f(\mathbf{x}) \tag{2.12}$$

How the learning rate changes through time is called a *learning rate schedule.* The method of gradient descent works well for non-linear problems when direct computation of $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$ is not possible. However, machine learning algorithms often use very large sets of data for which gradient descent quickly becomes computationally expensive. In order to counteract this, SGD is used instead. SGD turns the gradient into an expectation, which can be approximated using uniformly drown samples from the entire training set. These samples are called *minibatches*, and can range from 1 to a few hundred. Using this stochastic gradient in the gradient descent equation (2.12) then leads to similar results for a much lower computation time [22]. The specifics of the choice of minibatches, learning rate, and a few other characteristics then decides the type of SGD optimization algorithm that is used.

2.4.3. Common Issues in Machine Learning

Naturally the rich area of machine learning is not perfect, and this approach to pricing the cross-border capacity does come with some complications. We briefly mention some of these here, with more detailed solutions presented in the next chapters. For example, choosing which specific types of data should be used as input can be difficult. While many different forms of data may be appropriate, increasing the dimension of your inputs makes the learning process slow down or even stagnate [22]. Choosing the right data is known as feature selection, which we do using a Principle Feature Analysis (PFA) [28]. SGD (and other variants) often suffer from *underflow* or *overflow*, which essentially means that the gradient that is being minimized either stops moving, or explodes [22]. This is solved using an appropriate activation function [30] and clipping the gradient where necessary [22]. Finally, a well known danger of supervised learning is over-fitting the data on the training set. This means that algorithm starts to predict patterns that are only available in the training set, but are not part of the whole problem [22]. This is solved by splitting the data into a training and testing set, where the model is not exposed to the testing data until already fully trained to see if over-fitting has occurred. There are of course many more considerations, which will become apparent in the following chapters.

3 Model

In this chapter we present the methods used for building the valuation model for the cross-border capacity. We begin with a section outlining the motivation behind the methodology and a quick overview of the suggested approach. We then show the workings behind the type of model, an Input Output Hidden Markov Model (IOHMM), from a mathematical and machine learning perspective. We continue with an overview of the Normal Inverse Gaussian process. Finally we reiterate the mathematics of the valuation through the expected value of the price process our model produces, as well as an overview of certain Greeks that can be produced with this price process.

3.1. Suggested Approach

As was explained in the previous chapter, the Margrabe valuation method that is most often employed in the pricing of the cross-border capacity lacks a method of quantifying certain parameters, most notably the volatility of the underlying electricity prices. The way traders still find a way to value the asset is by reviewing data that drives the volatility in these markets and then using their experience in the market to estimate the volatility. In a sense, the traders are mapping input data to output data, albeit without a quantifiable criterion. The model we have created intends to mimic this mapping, only this time quantifying and optimizing the procedure.

The Margrabe approach also attempts to model the two electricity prices separately into two correlated Geometric Brownian Motions, and subsequently taking the discounted expected value of the spread. We have chosen to instead model the spread between the prices directly and taking its discounted expected value. This way we have a single process that needs to be calibrated to a market instead of two. In order to do this, a different underlying process is needed, for which we have chosen a variant of the Normal Inverse Gaussian process.

Given our new spread process, we then need to find a way to calibrate the parameters to the necessary market. As stated previously, our target values are the spreads between the spot prices between the two countries for which we value the cross-border capacity. We then use market data that influences these prices and map them through our price process to the target values via a machine learning algorithm. The type of market data that we use as inputs is reviewed in the next chapter. The model is trained using historical data pairings between inputs and outputs in order to calibrate how the input data influences the output data. When this training is finished, new inputs can be used to make predictions about the distributions of the outputs. The discounted expected value of these predictive distributions are then used for the cross-border capacity valuation, as will be shown in the remainder of this chapter.

3.2. Input-Output Hidden Markov Models

Input-Output Hidden Markov Models (IOHMMs) combine Hidden Markov Models (HMMs) and mapping of inputs and outputs through machine learning [2], as will be shown in this section. HMMs use a 'state variable' to describe some kind of market or otherwise unknown aspect that follows an unknown distribution. Conditioned on this state distribution, inputs are mapped to outputs through a separate distribution resulting in a total market distribution that is dependent on the state distribution as well as the inputs that are provided. Initially these types of models were designed for sequence processing and applied to grammatical inference [2]. However subsequently they were found to be applicable to financial processes as well, as can be seen in [3]. Furthermore, an example of modelling and predicting electricity spot prices on the Spanish market using IOHMMs can be found at [21].

3.2.1. Hidden Markov Models

We begin with mathematically defining Markov models in order to give a motivation for state variables. We begin with a series of definitions:

Definition 7 *A* Markov model of order k *is a probability distribution over some sequence of variables* $x_1^t = \{x_1, x_2, \dots x_t\}$ *which follows the Markov property of order k:*

$$\mathbb{P}\left(x_t | x_1^{t-1}\right) = \mathbb{P}\left(x_t | x_{t-k}^{t-1}\right)$$

The probability described in definition 7 can be decomposed into

$$\mathbb{P}\left(x_1^T\right) = \mathbb{P}\left(x_1^k\right) \prod_{t=2}^{T} \mathbb{P}\left(x_t | x_{t-k}^{t-1}\right)$$

In the special case of a Markov model of order 1, this becomes

$$\mathbb{P}\left(x_1^T\right) = \mathbb{P}\left(x_1\right)) \prod_{t=2}^{T} \mathbb{P}\left(x_t | x_{t-1}\right)$$

In the case that k is larger, we note that the number of parameters needed for $\mathbb{P}\left(x_1^T\right)$ becomes quite large as well. However, for most lengthy sequences the Markov property is not applicable and so modelling such sequences with a Markov model directly is also intractable. This was the motivation behind creating a type of variable that described not the sequence itself but something close to the sequence [4].

In practice, the sequence we are trying to model is the spread between two price processes, which indeed cannot be accurately modelled by a Markov model. Now consider the market (or markets) that our spread is based in. While we do not have a directly observable sequence to represent this market, we could still make some assumptions about the information at each time step. Such sequences are often referred to as *hidden*, as we cannot observe them directly. One important assumption that HMMs make is that this hidden variable can in fact be modelled by a Markov model, and that it is closely related to our observable sequence. Such variables are often referred to as *hidden state variables*. In the case of the state variable being a Markov model of order 1, that state at time t can be seen as carrying all the 'relevant' information about our observable sequence required to find the distribution the observable sequence at time $t+1$ [4]. This is similar to how traders would look at indicators in the market at the current time in order to predict the price in the future. We now formalize such a model mathematically.

Given a sequence $y_1^t = \{y_1, y_2, \dots y_t\}$, we now do not assume that the observable sequence itself follows the Markov property for a relatively low order. However, we do assume that there exists some unobserved sequence that is closely related to y_1^t that follows the Markov property for a low order. We call this the *hidden state variable* and denote it by x_t. Below we define the relationships in this new model:

Definition 8 *A* Hidden Markov Model (HMM) *is a joint probability distribution over some sequence of observed variables $y_1^t = \{y_1, y_2, \dots y_t\}$ and an unobserved state variable x_t which follows a Markov model of order 1. The conditional independence assumptions are*

$$\mathbb{P}\left(y_t|x_t\right) = \mathbb{P}\left(y_t|x_1^t, y_1^{t-1}\right)$$

$$\mathbb{P}\left(x_{t+1}|x_t\right) = \mathbb{P}\left(x_{t+1}|x_1^t, y_1^{t-1}\right)$$

The joint distribution of these variables can then be defined as

$$\mathbb{P}\left(y_1^T, x_1^T\right) = \mathbb{P}(x_1) \prod_{t=1}^{T-1} \mathbb{P}\left(x_{t+1}|x_t\right) \prod_{t=1}^{T-1} \mathbb{P}\left(y_t|x_t\right)$$

This joint distribution is then completely defined by

1. $\mathbb{P}(x_1)$, *the* initial state probability
2. $\mathbb{P}\left(x_{t+1}|x_t\right)$, *the* transition probability
3. $\mathbb{P}\left(y_t|x_t\right)$, *the* emission probability

For ease of use, we have defined the hidden state to be a Markov model of order 1 instead of order (small) k. While it is certainly possible to have higher orders than 1 for this distribution, the order 1 is often taken as increasing the number of hidden states can usually recreate the effect of higher orders [4].

These conditional probabilities are often more easily understood through a Bayesian network; such a network shows how sequences are related through time. Figure 3.1 can be interpreted as follows. At time t, information has been passed from hidden state x_{t-1} to hidden state x_t. Hidden state x_t then contains all the information needed for observable variable y_t, and subsequently for the next hidden state x_{t+1}. Note that no information is passed between the observable sequence that does not also flow through the hidden state sequence. Thus at each time, the hidden state contains all the necessary information [4].

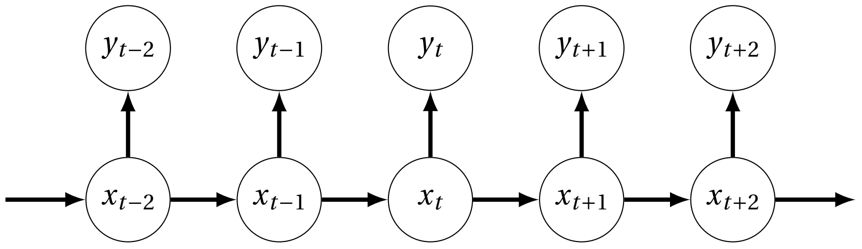


Figure 3.1: Bayesian network showing dependencies through time between hidden state variable x_t and observed variable y_t.

There are many possible choices for the initial state, transition, and emission probabilities [4]. However due to the scope of this paper we will focus on the most popular and subsequently the one that is most applicable to us. The hidden state variable x_t is most often seen as a discrete variable, with a multinomial distribution given the previous state. This means that initial state probabilities and transition probabilities defined above also follow a multinomial distribution. Furthermore the emission probabilities are taken to be continuous random variables, as will be shown in subsequent chapters. We assume that these are not homogeneous for each time step and so change through time. For more information regarding various types of HMMs and their workings, see [4].

3.2.2. Mapping Inputs to Outputs

We now take a step back from HMMs and focus on machine learning. Recall that we wish to find a mapping from certain inputs involving market indicators to an output of spreads.. One way of creating such a mapping is through a neural network. We give a definition of a simple feedforward network with a single hidden layer.

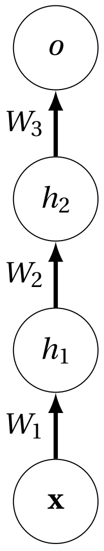
Definition 9 *Say we have output y with input $\mathbf{x}$ such that $y = f(\mathbf{x})$ for some unknown mapping f. A* feedforward neural network *(or* multilayer perceptron*) with a single hidden layer then defines an approximate mapping $f^*(\mathbf{x})$ as*

$$f^*(\mathbf{x}) = f_2(W_2 f_1(W_1\mathbf{x})))$$

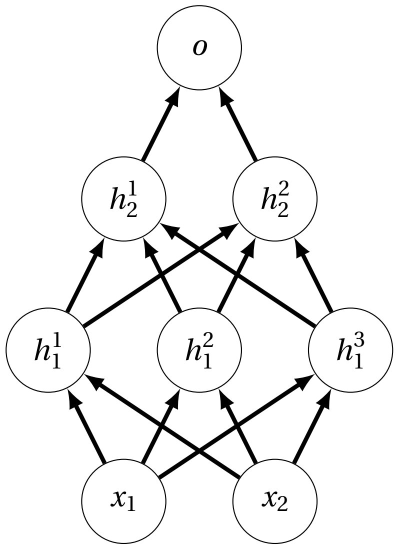
where f_1 is non-linear and referred to as the hidden layer *(or* activation function*), f_2 is referred to as the* output layer*, and W_i for $i = 1,2$ are matrices corresponding to the weights of each layer.*

This definition can be expanded to a 'deeper' neural network by increasing the number of hidden layers. The hidden layers are called 'hidden' because the values that they produce are not directly compared to a value; they are simply carried on to the next layer, until the output layer is compared to the goal value. We show a brief example of a neural network with two hidden layers in the directed graphs shown in Figure 3.2.

Figure 3.2a shows that the information flows from the input layer x, through the hidden layers h_1, and h_2, to the output layer o. The hidden layers h_1 and h_2 correspond to the activation functions f_1 and f_2 respectively, and o corresponds to the function f_3. Shown at the respective edges are the corresponding weight matrices. Figure 3.2b shows the expanded network. The number of nodes per layer represents the dimensionality of the respective function, i.e. $f_1 \in \mathbb{R}^3$, so the first hidden layer has three nodes. Furthermore, we note that



(a) Information flow



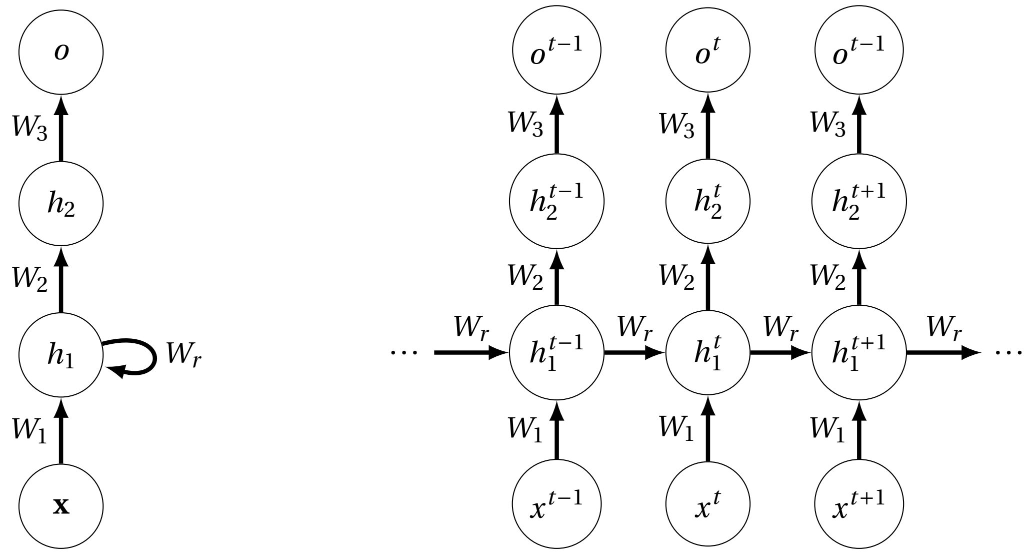
(b) Expanded view of nodes

Figure 3.2: Graphical representation of a multilayer perceptron with 2 hidden layers

all information only flows from the bottom to the top; this is necessary for it to be called a feedforward network. However, as will be seen later, there are other possibilities for different types of neural networks [22].

Usually the activation and output functions are chosen from a family of functions with a specific parameter set; these families often differ per layer and depend on the goal of the neural network. Once these are chosen and all the information has flown from left to right, we have our approximation $f^*(\mathbf{x})$ defined by our layer functions and weights. This is called *forward propagation*. Having our estimate, we then need a cost function to compare our estimate to the actual value; the choice of cost function is dependent on the goal of the network. Then in order to train, the derivative is taken from this cost function with respect to the weights W_i in every layer. This is done using an algorithm called *back-propagation*, which applies the chain rule to the layer and activation functions repeatedly. Training then involves minimizing the cost function; as this is done through the weights, this can be seen as giving connections through certain hidden nodes more preference if they contain the 'correct' answer. For many activation functions the gradient cannot be found in closed form, so Stochastic Gradient Descent (or a similar algorithm) is applied in order to minimize the cost function to an acceptable error level. When the network has successfully trained, the model can be used, for example for predicting unknown outputs based on new inputs. For more information on specific implementations of back-propagation, see [22]. The choice for the activation functions, output function, and cost function are very dependent on the problem the model is trying to solve; we explore these more in later sections and chapters.

The choices of how many layers, nodes, and their connections are called the architecture of the neural network; using recurrent connections between nodes is one such architecture. Recurrent neural networks (RNNs) are an extension of feedforward neural networks and are often used for sequence modelling, as using recurrent connections gives the



(a) Information flow (b) Unrolled view of nodes

Figure 3.3: Graphical representation of a RNN with 2 hidden layers

network the ability to have a 'memory' about previous input-output pairs. The recurrent connection in RNNs can be implemented in various ways through out the network; here we will focus on a self recursive hidden unit. In Figure 3.3a we see an RNN with 2 hidden units, one of which is recurrent. We see that unlike for feedforward networks, information is allowed to go back a layer. In Figure 3.3b, we see this network 'unfolded' through each input-output pair to give a clear view of how the information propagates forward. Back-propagation is then performed by going backward in this graph. It should be noted that for feedforward networks, parallel computation of input-output pair back-propagation is possible and often employed. However, for most RNNs this is not possible, as the order in which the back-propagation occurs for the entire dataset is fixed [22].

3.2.3. Architecture

Having explained HMMs and the basics behind (recurrent) neural networks, we now combine these two for our final model: an Input-Output Hidden Markov Model (IOHMM). We first define the state space of the model. We have, at time t, an *input vector* $\mathbf{u}_t \in \mathbb{R}^i$, an *output vector* $\mathbf{y}_t \in \mathbb{R}^o$, and a *discrete state* $x_t \in \{1,2,3,\dots n\}$. Here i is the number of inputs, o the number of outputs (usually 1), and n the number of states in our model. This state space has the following dynamics

$$\begin{aligned} x_t &= f(x_{t-1}, \mathbf{u}_t) \\ \mathbf{y}_t &= g(x_t, \mathbf{u}_t)), \end{aligned}$$

where we call f the *state transition function*, and g the *output function*. If we assume a probabilistic view, these dynamics have the form

$$S_t = \Phi(\mathbf{u}_t)S_{t-1},$$

where S_t represents the probability distribution of the state at time t, and $\Phi(\mathbf{u}_t)$ the matrix of probabilities of transitioning between states at time t. Therefore throughout time, our state space moves linearly through the discrete state probabilities, and non-linearly through inputs $\mathbf{u}_t$ [2].

Having defined the dynamics of the state space, we move on to define the non-linear functions and how they interact. This is best done through a graphical interpretation. Fig-

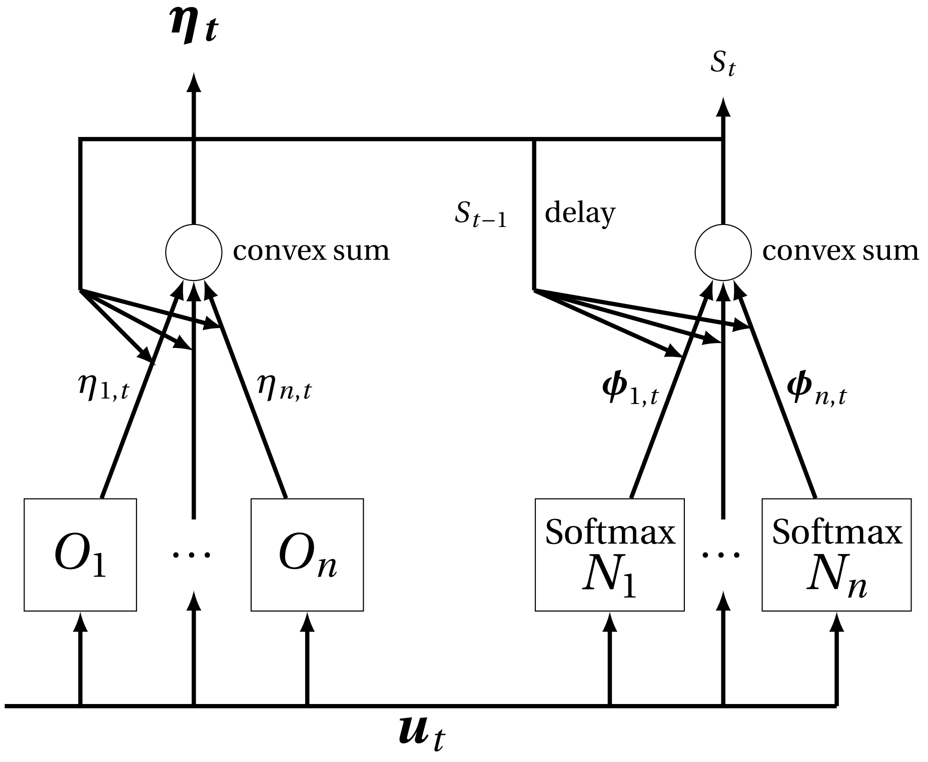


Figure 3.4: Graphical representation of an IOHMM.

ure 3.4 shows a graph of the IOHMM architecture. While there are multiple possibilities in which this model can be applied, we shall focus on the probabilistic interpretation as this is the one that we use. We begin with $\mathbf{u}_t$, the inputs at time t. These inputs flow into a series of subnetworks: N_j the *state network* for state $j \in \{1,2,3,\ldots n\}$, and O_j the *output network* for state $j \in \{1,2,3,\ldots n\}$. The state networks N_j have the task of predicting $\mathbb{P}(x_{t+1}|x_t = j, \mathbf{u}_t)$ the next state probability, given the input and that the current state is j [2]. While there are many types of forms these state networks take, we have chosen them to be multilayer perceptrons with a single hidden layer. At each time t, these state networks N_j then have an output of

$$\boldsymbol{\phi}_{j,t} = (\phi_{1j,t}, \phi_{2j,t}, \ldots, \phi_{nj,t})^{\top},$$

where $\phi_{ij,t}$ represents the probability that at time $t+1$ the state changes from state j to state i. Thus there are n state networks, and each state network N_j has n outputs at each time t. In order to ensure that $\boldsymbol{\phi}_{j,t}$ is a proper probability vector, a softmax function is added to the output layer of each state network.

Definition 10 *The* softmax function *is an activation function that applies the following* *nonlinear transformation to a vector* $\mathbf{w} \in \mathbb{R}^n$

$$\sigma(\mathbf{w})_i = \frac{e^{w_i}}{\sum_{k=1}^{n} e^{w_k}}$$

Applying this function to the output layer of N_j ensures that

$$\sum_{i=1}^{n} \phi_{ij,t} = 1, \quad \forall j, t.$$

While each state network attempts to find the transitioning probabilities at each timestep, combining these transition probabilities gives rise to a multinomial state distribution. This is represented by the state variable S_t. After the state variable is initialized at time $t = 0$, the probability of being in each state at time t is computed recursively by the previous state probability and the transition probability

$$S_t = \sum_{j=1}^{n} S_{j,t-1} \boldsymbol{\phi}_{j,t}$$

where $S_{j,t-1}$ the probability of being in state j at time $t-1$, and $\boldsymbol{\phi}_{j,t}$ the probability vector of transitioning from state j at time t as defined before. The relationships between state networks the recursive nature of the state distribution can also be seen in Figure 3.4 [2].

The output networks, while having the same input as the state networks, have a different goal. As with the state networks, each output network O_j is associated to the state $j \in \{1, 2, 3, \dots n\}$. The output networks compete to find some parameters describing the actual output $\mathbf{y}_t$. There are various possibilities for what these parameters could be; we have chosen them to be parameters of the distribution that we believe the realized output follows. In a sense we see then that each output network O_j attempts to find the following probability distribution at time t

$$\eta_{j,t} = \mathbb{P}(\mathbf{y}_t | x_t = j, \mathbf{u}_t)$$

This is again done through multilayer perceptrons with a single hidden layer.

Combining these output distributions with the state distribution that the state networks provide, we get the final output of the model

$$\begin{aligned} \boldsymbol{\eta}_t &= \sum_{j=1}^{n} S_{j,t} \eta_{j,t} && (3.1) \\ &= \sum_{j=1} \mathbb{P}(x_t = j | \mathbf{u}_t^1) \mathbb{P}(\mathbf{y}_t | x_t = j, \mathbf{u}_t) && (3.2) \\ &= \mathbb{P}(\mathbf{y}_t | \mathbf{u}_t^1) && (3.3) \end{aligned}$$

We see from this formula that our model produces a probability distribution for $\mathbf{y}_t$, the output at time t, given the entire input sequence $\mathbf{u}_t$. We see that the model will produce a different distribution for every t, as the input sequence on which it is conditioned will have changed. Thus the total distribution of the sequence we are trying to model is assumed to be non-homogeneous through time [2].

The resulting distribution $\boldsymbol{\eta}_t$ can be interpreted as follows. We see in equation 3.2 that the distribution is composed of the convex weighted sum of the various output distribution associated with each state, where the weights correspond to the probability of being in that state [2]. We note that as the state variable that is produced by the state networks have no observable sequence with which it can be compared, it is hidden as within an HMM. We therefore take the interpretation of the states in the model as we did before: we assume that all the states $\{1,2,3,\dots n\}$ represent various modes the market of the spread can be in. For example, we could have two states representing the market having either high or low volatility. The state variable S_t then represents the probability of the market being in each state. Furthermore, each output network then attempts to find the distribution for each market state at that time step. Finally, these distributions are combined into an overall spread distribution weighted on the states and conditional on the input.

In order to make computation feasible in the proposed model, some conditional independence must be introduced throughout the variable structure. We once again explain this through a Bayesian network. In Figure 3.5 we see that the Bayesian network for a

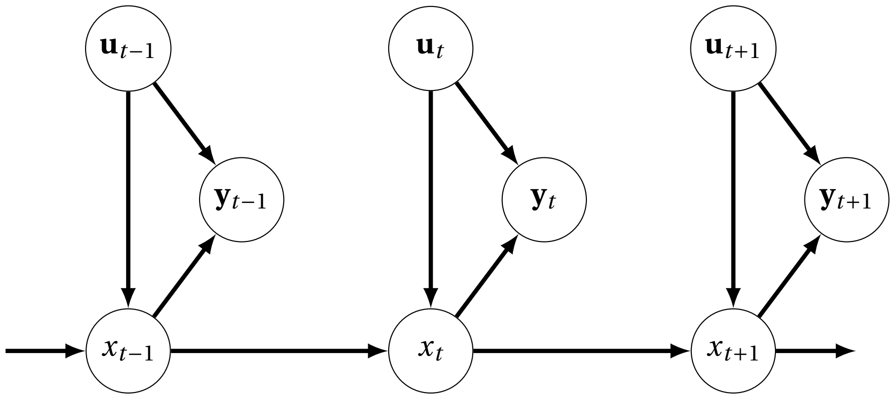


Figure 3.5: Bayesian network showing dependencies through time between hidden state variable x_t, input sequence $\mathbf{u}_t$ and observed sequence $\mathbf{y}_t$.

IOHMM is similar to that of a HMM. Once again we only need the input at time t and the previous state to find the current state. Furthermore, we see that while the input adds information to multiple places, this does not have an effect on the time dependency [2].

In the description of the subnetworks O_j and N_j so far we have stated them as multi-layer perceptrons with a single hidden state. However within that hidden state there is still a need for an activation function. In the next chapter we define which activation functions we have used for these networks. We excluded the softmax function from this postponement as this was relevant to the model architecture itself; the remainder of the activation functions are an implementation choice and so will be discussed later. Finally it should again be noted that the model which has been described here is a probabilistic version of the IOHMM; a broader definition of this type of model can be found at [2, 4].

3.2.4. Training

IOHMM can be trained using the Expectation Maximization (EM) algorithm, which we will explain here. The reference paper [2] from which this model stems then proceeds to find a formulated algorithm specifically for IOHMMs based on the EM algorithm. However, we have implemented our IOHMM in a machine learning program called Keras [8], and therefore do not use this specific variation of the EM algorithm. We therefore briefly show the general application of the EM algorithm to IOHMMs also described in [2], and leave the specifics of our implementation for the next chapter.

The EM algorithm is closely related to maximum likelihood estimation. The algorithm assumes that we have parameters of the model Θ, the data of the model D, the 'missing' or 'hidden' data of the model X and the set $D \cup X$ the completed dataset. Naturally in our case, Θ are the parameters of our output distribution, D is the set of input data $\mathbf{u}_t^1$, and X the state distribution S_t. Because X is not known, the log-likelihood $l(\Theta; D \cup X)$ of our model with the complete dataset is a random variable. We then define the following function through an expected value under the distribution of X:

$$Q(\Theta, \bar{\Theta}) = \mathbb{E}_X(l(\Theta; D \cup X)|D, \bar{\Theta}),$$

which can be seen as the expected value of the completed data log likelihood, given the known data. The task is then to iteratively maximize this function, as shown in Algorithm 1.

Algorithm 1 The Expectation Maximization (EM) Algorithm.

```
for $k = 1, 2, 3, \dots$ do
  Estimate:
    Compute
      $Q(\Theta, \Theta^k) = \mathbb{E}_X(l(\Theta; D \cup X)|D, \Theta^k)$
  Maximize:
    Update the parameters as
      $\Theta^{k+1} = \arg\max_{\Theta} Q(\Theta, \Theta^k)$
end for
```

In general, this function cannot always directly be maximized, in which case Θ is iterated such that $Q(\Theta, \bar{\Theta})$ increases.

Algorithm 2 The Generalized Expectation Maximization (GEM) Algorithm.

```
for $k = 1, 2, 3, \dots$ do
    Estimate:
        Compute
            $Q(\Theta, \Theta^k) = \mathbb{E}_X(l(\Theta; D \cup X) | D, \Theta^k)$
    Maximize:
        Update the the parameters as
            $\Theta^{k+1} = M(\Theta^k),$
        where
            $Q(M(\Theta^k), \Theta^k) \geq Q(\Theta^k, \Theta^k)$
end for
```

We then speak of the Generalized Expectation Maximization algorithm, shown in Algorithm 2 [2].

In the case of IOHMMs, the log-likelihood of the total probability is employed. We have the likelihood function for all input-output pairs $D = (\mathbf{u}_T^1, \mathbf{y}_T^1)$

$$\begin{aligned} L(\Theta|D) &= \prod_{t=1}^{T} \mathbb{P}(\mathbf{y}_t | x_t, \mathbf{u}_t; \theta_j^O) \mathbb{P}(\mathbf{x}_t | \mathbf{x}_{t-1}; \theta_j^N) \\ &= \prod_{t=1}^{T} \prod_{i=1}^{n} \prod_{j=1}^{n} \mathbb{P}(\mathbf{y}_t | x_t = i, \mathbf{u}_t; \theta_j^O)^{z_{i,t}} \mathbb{P}(\mathbf{x}_t = i | \mathbf{x}_{t-1} = j; \theta_j^N)^{z_{i,t} z_{j,t-1}} \end{aligned}$$

Here we take an indicator variable $z_{j,t} = \mathbb{1}_{x_t = j}$. Furthermore we take the parameters Θ to be the combination of θ_j^N and θ_j^O, the weights in the state networks and output networks respectfully. Taking the logarithm we obtain the log-likelihood

$$l(\Theta|D) = \sum_{t=1}^{T} \sum_{i=1}^{n} z_{i,t} \log \mathbb{P}(\mathbf{y}_t | x_t = i, \mathbf{u}_t; \theta_j^O) + \sum_{j=1}^{n} z_{i,t} z_{j,t-1} \log \mathbb{P}(\mathbf{x}_t = i | \mathbf{x}_{t-1} = j; \theta_j^N) z_{i,t} z_{j,t-1} \tag{3.4}$$

This log-likelihood is what the algorithm attempts to maximize when training. Algorithm 3 summarizes the specific functions and computations that occur when training. As can be seen, this is an example of the GEM, as the likelihoods which the IOHMM creates can become quite complicated [2].

3.3. The Normal Inverse Gaussian distribution

From equation 3.2 we now have several distributions that we need to consider. As we have already seen, the state variable has a multinomial distribution which is governed by an initialized multinomial instance propagated through the recursive transition probabilities generated by the state networks. Then, for each time t, each output network generates a distribution instance for each state. In historical applications of the IOHMM model, often a Gaussian distribution has been used for the output distributions, leading to a Gaussian

Algorithm 3 A Generalized Expectation Maximization Algorithm for an IOHMM.

```
for $k = 1, 2, 3, \dots$ do
    Estimate:
        for each pair $(\mathbf{y}_T^1, \mathbf{u}_T^1)$ do
            for each state $j \in \{1, 2, 3 \dots n\}$ do
                Compute $\phi_{t,j}$ and $\eta_{t,j}$ by running forward propagation through subnetworks $N_j$
                and $O_j$
            end for
            Compute $S_t$ with $\phi_{t,j}$ through recursion and past state value.
            Compute $\eta_t$ through convex sum $\phi_t$ and $\eta_{t,j}$
                $$\eta_t = \sum_{j=1}^{n} S_{j,t} \eta_{j,t}$$
        end for
    Maximize:
        for each state $j \in \{1, 2, 3 \dots n\}$ do
            Adjust the parameter $\theta_j^N$ in the state network $N_j$ in order to increase the function
            3.4
        end for
        for each state $j \in \{1, 2, 3 \dots n\}$ do
            Adjust the parameter $\theta_j^O$ in the state network $N_j$ in order to increase the function
            3.4
        end for
end for
```

mixture for the final distribution [2]. Indeed, in the applciation of an IOHMM on the Spanish electrcitiy market in [21], Gaussians were also used. We, however, have chosen a different route.

Gaussian distributions, while very applicable, have historically shown to not be empirically accurate for financial data. Often this is due to tail width and kurtosis constraints [14]. In the case of electricity prices, this is also true; not only are the tails not heavy enough, there are also instances of jumps between prices that are not modelled accurately with regular Gaussian distributions [13]. Therefore, we have chosen to implement each output distribution as a Normal Inverse Gaussian (NIG). The NIG distribution has been shown to have heavy tails, as well as a jump diffusion element [14]. Furthermore, application to electricity prices [13], as well as use in a Black-Scholes option pricing framework [39] has proven to be successful. Therefore using this distribution in pricing an electricity price spread option seems appropriate.

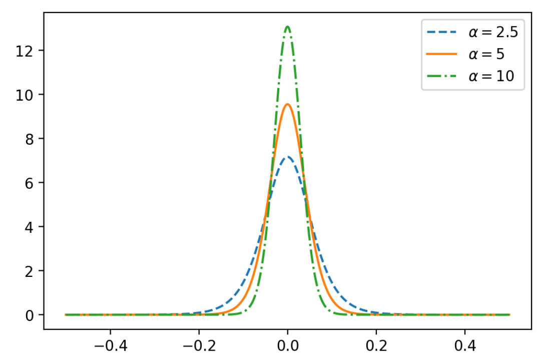
The NIG distribution was originally constructed through a mean-variance mixture of an Inverse Gaussian. Below we define the probability density function (PDF).

$$\mathrm{NIG}(x; \alpha, \beta, \mu, \delta) = \frac{\alpha\delta}{\pi} \frac{K_1\left(\alpha\sqrt{\delta^2 + (x-\mu)^2}\right)}{\sqrt{\delta^2 + (x-\mu)^2}} e^{\delta\gamma + \beta(x-\mu)} \tag{3.5}$$

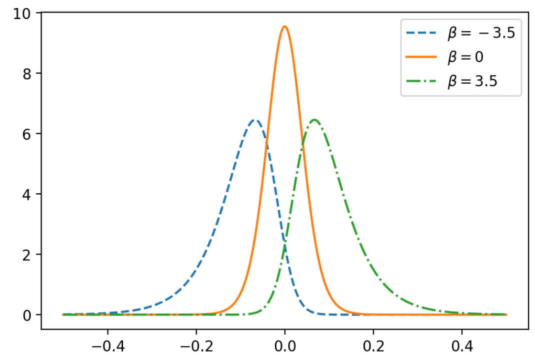
Here $\gamma = \sqrt{\alpha^2 - \beta^2}$, and the modified Bessel function of the third kind

$$K_\lambda(x) = \frac{1}{2}\int_0^\infty u^{\lambda-1} e^{-\frac{1}{2}x(u^{-1}+u)}\,du$$

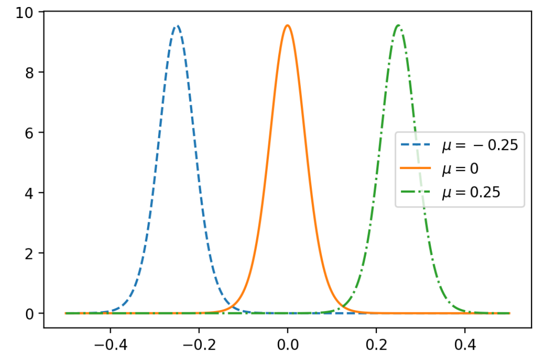
The PDF in equation 3.5 has four parameters, each with its own constraints and effect on the overall function. The shape is decided by $\alpha > 0$. The skewness is decided by β, for which we have $|\alpha| > |\beta|$. The location and scale are decided by $\mu \in \mathbb{R}$ an $\delta > 0$ respectfully. These effects are illustrated in Figure 3.6.



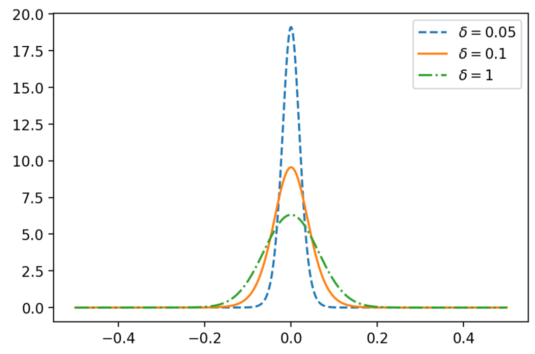
(a) α



(b) β



(c) μ



(d) δ

Figure 3.6: PDFs of NIG distributions with certain parameters changed.

Each parameters is changed while the other parameters are kept a standard value. The standard values are $\alpha = 50$, $\beta = 0$, $\mu = 0$, and $\delta = 0.1$. We note that in the case that $\beta = 0$, the PDF is symmetric around 0.

The cumulative distribution function (CDF) and the survival function of the NIG distribution are defined below.

$$\mathrm{NIG}_{\mathrm{CDF}}(x;\alpha,\beta,\mu,\delta) = \int_{-\infty}^{x} \mathrm{NIG}(y;\alpha,\beta,\delta,\mu)\,dy$$

$$\mathrm{NIG}_{\mathrm{SF}}(x;\alpha,\beta,\mu,\delta) = \int_{x}^{\infty} \mathrm{NIG}(y;\alpha,\beta,\delta,\mu)\,dy$$

These functions will be used in the valuation process. Unfortunately, there is no closed form solution for these integrals, so numerical methods will have to be used for their evaluation. For more information regarding the derivation and properties of functions related to the NIG distribution, see [1].

3.4. Valuation

Having defined the NIG distribution, we come back to our model output distribution defined in equations 3.1-3.3. We let the output distribution for each output network O_j, with $j \in \{1,2,3\dots n\}$ be

$$\eta_{j,t} = \mathbb{P}(\mathbf{y}_t | x_t = j, \mathbf{u}_t) = \text{NIG}(x; \alpha_{j,t}, \beta_{j,t}, \delta_{j,t}, \mu_{j,t}),$$

as defined in equation 3.5. We then see that our model output at time t is $(\boldsymbol{\eta})_t = \mathbb{P}(\mathbf{y}_t|\mathbf{u}_t^1)$, which is a mixture of NIG distributions. Thus we have a time dependent distribution for the price spreads between two countries $\mathbf{y}_t$ which is conditional on some input sequence $\mathbf{u}_t$.

We compare this distribution process with the option valuation method for spread options described in the previous chapter. There the prices were modelled separately according to correlated GBMs, and subsequently subtracted within the pay-off function described in equation 2.8 to create the spread. We have chosen to model the spread directly. We note that this could not have been done with a single GBM, as these cannot produce negative values, whereas the NIG distribution can become negative. Furthermore, the parameters of the GBMs used in the Margrabe formula are stationary, whereas the parameters in the mixture NIG distribution are updated after every timestep t. We also note that when using a GBM, the underlying process is a log-asset price process. This means that with the corresponding exponential representation of the process, the characteristic function of the process can be used when taking the expected value. This is however not the case when taking the spread directly, as this is not a log-asset process; this means that characteristic function has no use when taking expected values. Finally the mixture aspect of our final distribution introduces an underlying market structure that facilitates possible jumps. No such structure is available in a GBM.

Despite the differences in underlying process, the subsequent valuation remains similar. We see that as $(\boldsymbol{\eta})_t$ already represents the spread, our pay-off function at time T is

$$H(T, \boldsymbol{\eta}) = \max(\boldsymbol{\eta}_T, 0) \tag{3.6}$$

Now we let the $\mathscr{F}(t)$ be the filtration generated by the sequence $\boldsymbol{\eta}_t$, i.e. $\mathscr{F}(T)$ contains all the information generated by $\boldsymbol{\eta}_T$ up to time T. Then we define the time t value of the European call option to be the discounted conditional expectation

$$V(t, T, \boldsymbol{\eta}) = e^{-r(T-t)}\mathbb{E}[H(T, \boldsymbol{\eta})|\mathscr{F}(T)] = e^{-r(T-t)}\mathbb{E}[\max(\boldsymbol{\eta}_T, 0)]$$

where $e^{-r(T-t)}$ is the discount factor, with r being the interest rate [34]. Here the final equality comes from the fact that all the information carried in the filtration $\mathscr{F}(T)$ is also contained in the mixture distribution $\boldsymbol{\eta}_T$ through the hidden states x_t, as described previously. Now we note that for Y following a mixture distribution with component PDFs p_{Y_i} for $i = 1,2,3\dots n$, if we have a function $H(x)$ such that $H(Y_i)$ exists, we have

$$\begin{aligned}\mathbb{E}(H(Y)) &= \int H(x) \sum_i^n w_i p_{Y_i}(x) dx = \sum_{i=1}^n w_i \int H(x) p_{Y_i}(x) dx \\ &= \sum_{i=1}^n w_i \mathbb{E}(H(Y_i))\end{aligned}$$

We apply this to our option value and definition of $\boldsymbol{\eta}_t$, where we take $w_i = S_{j,T}$; then we get

$$\begin{aligned}
V(t,T,\boldsymbol{\eta}) = e^{-r(T-t)}\mathbb{E}[\max(\boldsymbol{\eta}_T,0)] &= \mathbb{E}\left[\max\left(\sum_{j=1}^{n} S_{j,T}\eta_{j,T},0\right)\right] \\
&= e^{-r(T-t)}\sum_{j=1}^{n} S_{j,T}\mathbb{E}(\max(\mathrm{NIG}(x;\alpha_{j,T},\beta_{j,T},\delta_{j,T},\mu_{j,T},),0) \\
&= e^{-r(T-t)}\sum_{j=1}^{n} S_{j,T}\int_{-\infty}^{\infty}\mathbb{1}_{x>0}x\frac{\alpha\delta}{\pi}\frac{K_1\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)}{\sqrt{\delta^2+(x-\mu)^2}}e^{\delta\gamma+\beta(x-\mu)}dx
\end{aligned}$$

Here we have used the fact that the $\max(x,0)$ function can be rewritten as $\mathbb{1}_{x>0}x$. Some subscripts (j,T) have intentionally been left out for clarity. Then finally we have our option value

$$V(t,T,\boldsymbol{\eta}) = e^{-r(T-t)}\sum_{j=1}^{n} S_{j,T}\int_{0}^{\infty}x\frac{\alpha\delta}{\pi}\frac{K_1\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)}{\sqrt{\delta^2+(x-\mu)^2}}e^{\delta\gamma+\beta(x-\mu)}dx \tag{3.7}$$

The integrals within the summation in equation 3.7 cannot be solved analytically, so numerical methods will be applied. In this thesis we used the *SciPy* library for numerical integration [40]. Should computation time be an issue when applying this model, the COS method could also be employed [34].

Using equation 3.7, we can find the time 0 value of the cross-border capacity for any expiration T. Recall from the previous chapter that we wish to value the cross-border capacity for an entire month. Through the described auction process, we then have a bundle of options that is being auctioned together; each option having a different expiry such that there is an option for every hour of the month. Say we have a month with m days. Using our model, we find the option value at time 0 for each of these options in the monthly bundle

$$V_{\text{month}}(\boldsymbol{\eta}) = \sum_{i}^{24m} V(0,T_i,\boldsymbol{\eta}),$$

where T_i for $i = 1,2,3\ldots24m$ is an appropriately scaled time parameter. Averaging this value then gives the appropriate price of the cross-border capacity in the monthly auction

$$V_{\text{auction}}(\boldsymbol{\eta}) = \frac{1}{24m}V_{\text{month}}(\boldsymbol{\eta}) = \frac{1}{24m}\sum_{i}^{24m} V(0,T_i,\boldsymbol{\eta}) \tag{3.8}$$

3.5. Greeks

The Greeks are, as explained earlier, sensitivities with regards to the option value in a Black-Scholes framework. In the Magrabe spread option valuation, much of the original Black-Scholes framework was used, and so comparative Greeks could be found. However the model we have described so far is somewhat different from Black-Scholes in many aspects. Nevertheless, we can still derive certain sensitivities from our option value, namely Delta and Gamma. Note however that these new sensitivities (which we still call 'Greeks') may differ slightly from the original Black-Scholes Greeks. Furthermore, as the dependency of the modelled spread $\boldsymbol{\eta}$ to time parameter t is encoded in the non-stationary parameters, there is no direct way of calculating Theta. Similarly, as there is no volatility parameter in the option value function, Vega is also not derivable in closed form. These parameters could be found numerically, however this beyond the scope of this work.

We begin with some preliminary equations regarding some random variable X and function $H(x)$. We define A to be the expected value of $H(X)$

$$A = \mathbb{E}(H(X))$$

Then assuming that $\mathbb{E}(|H(X)|) < \infty$, the Dominated Convergence Theorem (DCT) states we have the following

$$\frac{\partial A}{\partial X} = \mathbb{E}\left(\frac{\partial}{\partial X} H(X)\right)$$
$$\frac{\partial^2 A}{\partial X^2} = \mathbb{E}\left(\frac{\partial^2}{\partial X^2} H(X)\right)$$

We can apply this to our option valuation function using the pay-off function defined in equation 3.6. We see that

$$\frac{\partial}{\partial x} \max(x, 0) = \mathbb{1}_{x>0},$$
$$\frac{\partial^2}{\partial x^2} \max(x, 0) = \delta(x),$$

where $\delta(x)$ is the Dirac Delta function. Therefore, we see that

$$\begin{aligned}\Delta = \frac{\partial V}{\partial \boldsymbol{\eta}} &= e^{-r(T-t)}\mathbb{E}\left[\frac{\partial V}{\partial \boldsymbol{\eta}} \max(\boldsymbol{\eta}_T, 0)\right] = e^{-r(T-t)}\mathbb{E}[\mathbb{1}_{\boldsymbol{\eta}>0}] \\ &= e^{-r(T-t)} \sum_{j=1}^{n} S_{j,T}\mathrm{NIG}_{\mathrm{SF}}(0; \alpha_{j,T}, \beta_{j,T}, \delta_{j,T}, \mu_{j,T})\end{aligned}$$

Here we have used that the expected value of an indicator variable is the probability of the event occurring, so $\mathbb{E}[\mathbb{1}_{\boldsymbol{\eta}>0}] = \mathbb{P}(\boldsymbol{\eta} > 0)$, which gives the survival function at point 0. We then also have

$$\begin{aligned}\Gamma = \frac{\partial^2 V}{\partial \boldsymbol{\eta}^2} &= e^{-r(T-t)}\mathbb{E}\left[\frac{\partial^2 V}{\partial \boldsymbol{\eta}^2} \max(\boldsymbol{\eta}_T, 0)\right] = e^{-r(T-t)}\mathbb{E}[\delta(\boldsymbol{\eta})] \\ &= e^{-r(T-t)} \sum_{j=1}^{n} S_{j,T}\mathrm{NIG}(0; \alpha_{j,T}, \beta_{j,T}, \delta_{j,T}, \mu_{j,T})\end{aligned}$$

Here we have used that, for random variable X with PDF $f(x)$, the expected value the Dirac Delta function is

$$\mathbb{E}(\delta(X)) = \int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0)$$

Applying this to the PDF of $\boldsymbol{\eta}_T$ and simplifying gives the result.

4
Implementation

In this chapter we show how the model we presented in the previous chapter is implemented. We begin with an overview of the deeplearning API Keras, and how a model is trained. We continue with an overview of the output data and the structure behind it. We show that as the temporal structure between input and output data differs, multiple instances of the model need to be initialized. We reiterate the need for activation functions within the hidden and output layers of our subnetworks, and show for every parameter the choices we have made in that regard. Error measures are defined beyond the loss function that is optimized. We show the various options for input data, and perform a feature selection using Principle Feature Analysis. Finally, we define the hyperparameters around the model and show how the optimal choice for these parameters was made.

4.1. From Training to Valuation

For the implementation of the IOHMM we have used the deeplearning API Keras. This is an open-source platform backed by the machine learning backend Tensorflow [8]. Model creation in Keras is done using a layer building API, where the output of a layer becomes the input of the next layer. Our model consists of a series of neural nets (or Dense layers), which are recurrently applied and subsequently multiplied to provide a convex sum of probabilities making a final total probability. The final model in the Keras API is then a chain of layers from input to final output layer. A graph of the Keras variant of our IOHMM can be found in Appendix B.

We now reiterate how the EM algorithm from the previous chapter occurs in Keras. Once we have our model implemented, training works as follows. For the entire set of input-output pairs ($\mathbf{u}_T^1, \mathbf{y}_T^1$, the inputs are put into the input layer. These inputs then follow the transformations provided at each chain in the layering structure, finally reaching the output layer. This is the forward propagation. These inputs then have the form found in equation 3.1. Then the log-likelihood is calculated as in equation 3.4. This is then turned into the negative log-likelihood and is known as the *loss function.* Subsequently, Keras has built-in back-propagation which differentiates this negative log-likelihood, now moving backwards through the layering chain. For some optimizer, Keras applies a change in the model weights such that the negative log-likelihood is reduced. We define this as follows.

Definition 11 *An* epoch *is a single cycle of forward propagation, backward propagation, and weight augmentation for all input-output pairs.*

After a certain amount of epochs, the model is considered trained. This is either because it has reached the predetermined number of epochs or because the loss function has reached a low enough error. When the model is trained, we can predict output values. This is done by feeding the input layer new inputs and running these along the layered chain. This time, the weights are not updated and the output layer returns the parameters for the distribution.

Having our predicted parameters, we can define the hourly option price by equation 3.7. The integrals in equations 3.7 represents the expected value

$$\mathbb{E}[\max(\boldsymbol{\eta}_T, 0)].$$

Now we train the model with the spreads computed in a certain 'direction', for example from Germany to France. For this training, the spreads are positive if the German price is higher and negative if the French price is higher; in this case we want the expected positive values and so compute the integral from 0 to ∞. We can repurpose this training for the other direction. If we have a model trained on the Germany to France spreads, the information for the France to Germany spreads is contained within as well. In this case, we wish to know all the expected negative values, as we are interested in the Germany to France spread being in France's favor (i.e. negative). In these cases, we take the integrals in 3.7 from $-\infty$ to 0.

Having our hourly option values at the time of their maturity, we need to discount them. We take the interest rate $r = 1$. Furthermore, as we want to know the time 0 value of the option, we have $t = 0$. Then we assume that for an options with maturity i days from now, our expiry will be $T = \frac{i}{365}$; the difference between the hourly maturities is ignored, as these values are computed in separate models. So our final discount factor for an option with maturity i days from now is

$$e^{-r(T-t)} = e^{-\frac{i}{365}}$$

4.2. Optimizers

As we have stated in previous chapters, once back-propogation is performed, there is a gradient that needs to be minimized. The backbone of modern optimizers that perform this minimization is the SGD algorithm defined in previous chapters. There are however many more variants of this algorithm [7]. We have used the ADAM algorithm [25], whose update rule is defined below.

Definition 12 *The* ADAM algorithm *proposes that we minimize f through using the stochas-*

tic gradient of f together with parameters α_t, β_1, β_2,ϵ to iterate to a solution:

$$\begin{aligned}
&m_0 = v_0 = 0\\
&m_{t+1} = \beta_1 m_t + \left(1-\beta_1\right)\nabla f\left(x_t\right)\\
&v_{t+1} = \beta_2 v_t + \left(1-\beta_2\right)\nabla\left(x_t\right)^2\\
&b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}\\
&x_{t+1} = x_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}}+\epsilon} b_{t+1}
\end{aligned}$$

As can be seen when compared to equation 2.12, the ADAM algorithm is somewhat more complicated than SGD. ADAM uses estimates of the first and second moments of the stochastic gradient together with a learning rate parameter α_t to create adaptive learning rates for each network weight parameter. The parameters β_1 and β_2 govern the exponential decay rate of the first and second moments respectuflly. Finally, ϵ is added for numerical stability. It is ideally suited for problems with large datasets, as well as for modelling non-stationarity, and computationally efficient, all of which are available in our problem setting [25]. It has also been shown to outperform SGD and other comparable algorithms [7]. Furthermore, it is built into Keras for easy implementation.

The final option we have within our optimizer choice is 'gradient clipping'. As mentioned before, machine learning models occasionally suffer from 'exploding gradients'. This essentially means that when the back propagated gradient of the loss function becomes expectantly large. This can cause numerical overflow within the optimizer trying to adapt the gradient for the optimization process; this is especially a problem with recurrent models as large gradients often get reused, extrapolating the problem. One solution for this is defined below.

Definition 13 *An optimizer performs* gradient clipping *when it resizes the gradient to some predetermined maximum (or minimum) size if the gradient it receives is too large (or too small).*

Gradient clipping can also be applied such that the norm of the gradient does not exceed some value; this is the method we have chosen. While gradient clipping does nothing to improve the performance of the model itself, it improves numerical stability so that gradients do not explode [22]. All the values in regards to optimizer parameters that we used are shown in Table 4.1.

Parameter	**Value**
α	0.001
β_1	0.9
β_2	0.999
ϵ	1e-7
Max norm	1.0

Table 4.1: Parameter values used for the ADAM optimizer.

4.3. Activation Functions

As previously described, activation functions are non-linear functions used in the hidden nodes of neural networks [22]. As we saw with the state networks, these functions can also be used within the output layer in order to mold the output into a specific range. For the output networks, the hidden units were kept linear, and so no activation functions were used. However each output network represent a single instance of a NIG distribution, or more importantly, the parameters of such a distribution. Therefore activation functions were used to keep the output from the output networks in a specific range. Notably the location parameter had a range of $\mu \in \mathbb{R}$, and so required no activation function. The parameters together with the complete used activation functions are shown in Table 4.2. For more information regarding the use of activation functions in deep learning, see [30].

We begin with the activation function used in the hidden units.

Definition 14 *The* sigmoid function *is defined by*

$$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$$

The sigmoid function is one of the most common activation functions, especially for logistic probability problems, as well as classification problems [30]. It was also used in the current setting in the analysis of the Spanish electricity market in [21]. Some concern has been shown in recent years in the use of the sigmoid function in deeper networks, however as we only have a single hidden layer, this should not be a worry [30].

The shape and scale parameters α and δ have the constraints $\alpha > 0$ $\delta > 0$. We considered the functions defined below. We note that the ranges of these functions do have negative values; however as they are bounded below, a simple shift in the function shifts the range to only positive values as well.

Definition 15 *The* Exponential Linear Unit (ELU) function *is defined by*

$$\text{elu}(\alpha, x) = \begin{cases} x & \textit{if } x > 0 \\ a\left(e^x - 1\right) & \textit{otherwise} \end{cases}$$

for the parameter $a > 0$

Definition 16 *The* Scaled Exponential Linear Unit (SELU) function *is defined by*

$$\text{selu}(x) = \begin{cases} cx & \textit{if } x > 0 \\ ca\left(e^x - 1\right) & \textit{otherwise} \end{cases}$$

for the parameter $a = 1.67326324$ *and* $c = 1.05070098$.

The ELU function was proposed as a way of increasing the speed of learning [30]. This, together with the lower bound and lack of upper is why it was considered for both α and δ. However when implemented in our model, numerical stability problems were found, as both these variables displayed exploding gradients during training. The SELU function however, works in a similar manner but has been proven to withstand vanishing and exploding gradients [30]. When this was implemented for α, the exploding gradients in both parameters ceased. We also considered 'capping' the scale parameter at $\delta = 1$, however this restricted the learning process too much.

Finally, we have the asymmetry parameter β with restriction $|\beta| < \alpha$. To satisfy this restriction, we had to find an activation function $\sigma(x)$ such that $\sigma(\beta) \in (-1, 1)$, after which we set $\beta = \alpha\sigma(\beta) - \epsilon$, where ϵ is some small constant. This way the restriction was respected, and the different gradient produced by $\sigma(x)$ and $\text{selu}(x)$ ensured that the values of α and β could differ. For this we considered the following activation functions

Definition 17 *The* Hyperbolic Tangent (tanh) function *is defined by*

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Definition 18 *The* softsign function *is defined by*

$$\text{softsign}(x) = \frac{x}{|x| + 1}$$

We first tried tanh, however this led to vanishing gradients during training for β, meaning that β would consistently be equal to $\alpha - \epsilon$. After implementing the softsign function however, the issue was resolved.

Parameter	**Activation function**
α	$1.67326324 \times 1.05070098 + 1\text{e-}5 + \text{selu}(\alpha)$
β	$\alpha\,\text{softsign}(\beta) - 5\text{e-}6$
μ	μ
δ	$1.0 + 1e - 5 + \text{elu}(\delta)$

Table 4.2: Activation function for each parameter of the NIG distribution.

4.4. Error Measures

The purpose of the model in this work is to find a predictive density that accurately describes the realized spread $\mathbf{y}_t$. To do this, a *scoring rule* is employed for the predicted density and realized spreads to determine the level of accuracy over the entire data set. We wish to find a scoring rule that considers not only a point mass estimate, but also if the probability of an observation was accurately allocated. These error measures are returned after every training epoch, as well as after prediction, should the observed spreads be available. We consider three scoring rules in this work.

For scoring the allocation of probability, we employ the use of the negative log-likelihood (NLL) as described in equation 3.4. This is after all the loss function that the model is training on. The point mass estimate refers to how far away the probability density is from the realized observation. This is usually done by finding some error function around a midpoint of the distribution, for example the mean or median. We have used the Mean Absolute Error defined below.

Definition 19 *The* Mean Absolute Error (MAE) *for a set of densities and observations* $(p_1^T, \mathbf{y}_1^T)$ *is defined by*

$$\text{MAE} = \frac{\sum_{t=1}^{T} |\mathbb{E}(p_t) - \mathbf{y}_t|}{T}$$

Finally, we consider scoring rule that considers the entire distribution, and so both of the criterion mentioned above; we define it below.

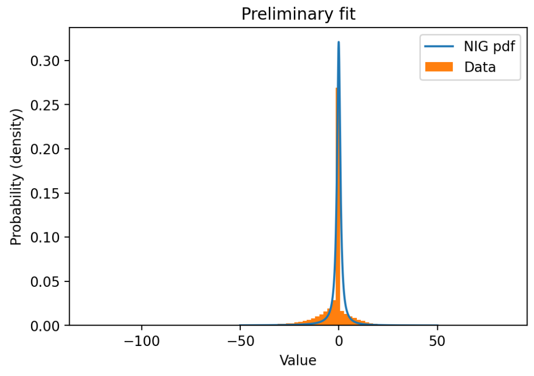
Definition 20 *The* Continuous Ranked Probability Score (CRPS) *of a distribution with CDF* F *and a single observation* y *is defined by*

$$\text{CRPS}(p, y) = \int_{-\infty}^{\infty} (F(x) - \mathbb{1}_{x \geq y})^2 dx$$

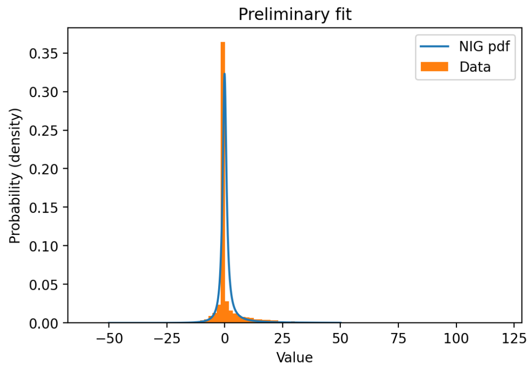
This is comparative to the integral (and so continuous) Brier Score, and is a proper scoring rule [20]. The CRPS of the entire data set would then be the average of all individual pairs of distribution and observation. We had wished to use this scoring rule in conjunction with the previous two. However, as we previously saw, numerical integration is already needed in order to find the CDF of a NIG distribution. This means the CRPS would again have to be numerically integrated; this proved to be too computationally expensive, even if only applied to the prediction results and not during training. Therefore we only used the MAE and the NLL as error measures.

4.5. Data Analysis

We now begin a preliminary analysis of the data and see what effects this has on our model. We recall that the output data is the singular sequence of spreads between the hourly spot prices of two countries, Germany-France and Netherlands-Germany. A preliminary fit of the data to a single NIG distribution is shown in Figure 4.1. This data is available from the JAO database at [18]. The input data are multiple sequences of indicators of the German and French or Dutch and German electricity markets. This data was received from Northpool B.V.



(a) Germany-France



(b) Netherlands-Germany

Figure 4.1: Fit of 1 January 2019 to 31 July 2021 spread data to a single NIG distribution

4.5.1. Temporal Structure

As stated, the output data sequence has a point for every hour. However, for some of the input data, only daily sequences are available. This is an issue, as input-output pairs are fed into the model together during training. There are two ways to solve this: either average the hourly prices to find a daily price, or expand the daily sequences by 24 so that there is a value for every hourly price. Seeing that we are pricing options by the hour, the second solution is preferable. This however brings a new problem, as we now have multiples of 24 of the same values for some input sequences. Recall that the model works with weights within neural networks. When the model is trained and we feed the model a previously unknown input sequence, these weights are no longer updated. Therefore a sequence of multiples of 24 of the same values will then return a predicted output sequence of multiples of 24 of the same values. So despite training being able to take place, predicted distributions will be equal for every hour of a certain day.

We solve this by changing the temporal structure of the information flow in the model. Say we have n days of data; in this case we assume all the input data is only available on a daily basis, so we have $\mathbf{u}_i$ for $i = 1,2,3\ldots n$. Our output data is then of the form $\mathbf{y}_{i,j}$ for $i = 1,2,3\ldots n$ days and $j = 1,2,3\ldots 24$ hours. The initial temporal structure then flows from $(\mathbf{u}_1,\mathbf{y}_{1,1})$ to $(\mathbf{u}_n,\mathbf{y}_{n,24})$ as shown in Figure 4.2. We change this by changing the number of

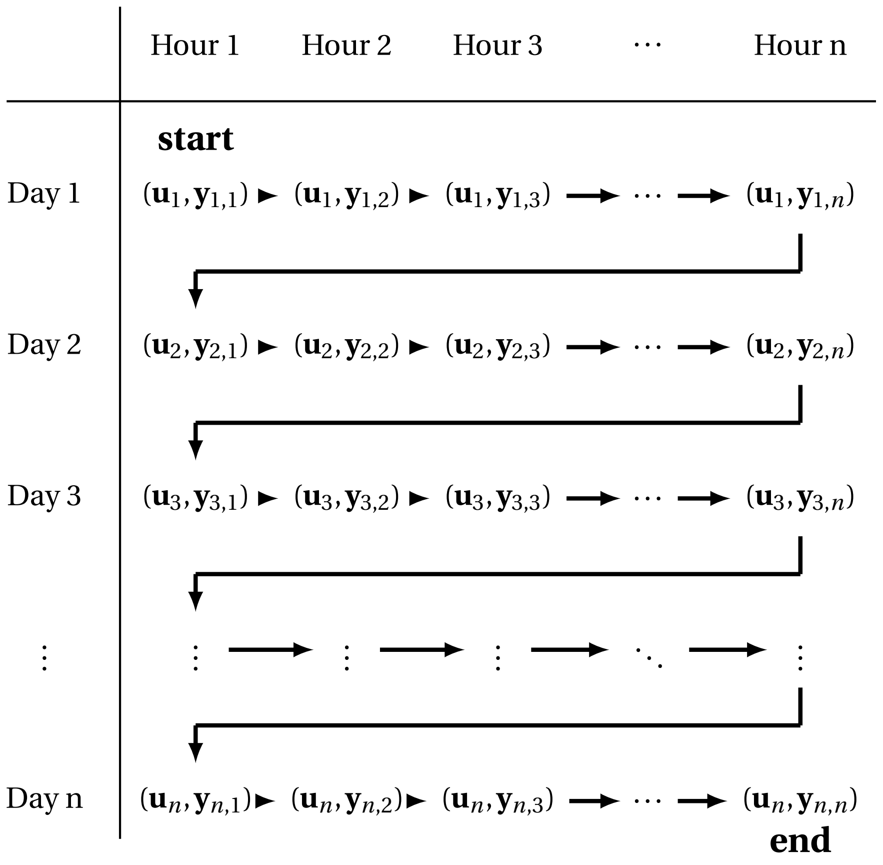

Hour 1
Hour 2
Hour 3
⋯
Hour n
start
Day 1
$(\mathbf{u}_1,\mathbf{y}_{1,1})$
$(\mathbf{u}_1,\mathbf{y}_{1,2})$
$(\mathbf{u}_1,\mathbf{y}_{1,3})$
$(\mathbf{u}_1,\mathbf{y}_{1,n})$
Day 2
$(\mathbf{u}_2,\mathbf{y}_{2,1})$
$(\mathbf{u}_2,\mathbf{y}_{2,2})$
$(\mathbf{u}_2,\mathbf{y}_{2,3})$
$(\mathbf{u}_2,\mathbf{y}_{2,n})$
Day 3
$(\mathbf{u}_3,\mathbf{y}_{3,1})$
$(\mathbf{u}_3,\mathbf{y}_{3,2})$
$(\mathbf{u}_3,\mathbf{y}_{3,3})$
$(\mathbf{u}_3,\mathbf{y}_{3,n})$
Day n
$(\mathbf{u}_n,\mathbf{y}_{n,1})$
$(\mathbf{u}_n,\mathbf{y}_{n,2})$
$(\mathbf{u}_n,\mathbf{y}_{n,3})$
$(\mathbf{u}_n,\mathbf{y}_{n,n})$
end

Figure 4.2: Temporal structure of information flow for a single initialization of an IOHMM model.

models that we initialize. In Figure 4.2, all the information flows through a single model. Instead, we initialize 24 models, one for each hour of the day. We then only feed the n pairs of data that describes hour 1 into the model for hour 1, only hour 2 data into the model of hour 2, etc. The new temporal structure is shown in Figure 4.3. Initially it may seem like a

$$\begin{tikzcd}
 & \text{Hour 1} & \text{Hour 2} & \text{Hour 3} & \cdots & \text{Hour n} \\
 & \textbf{start} & \textbf{start} & \textbf{start} & \textbf{start} & \textbf{start} \\
\text{Day 1} & (\mathbf{u}_1,\mathbf{y}_{1,1}) \arrow[d] & (\mathbf{u}_1,\mathbf{y}_{1,2}) \arrow[d] & (\mathbf{u}_1,\mathbf{y}_{1,3}) \arrow[d] & \cdots \arrow[d] & (\mathbf{u}_1,\mathbf{y}_{1,n}) \arrow[d] \\
\text{Day 2} & (\mathbf{u}_2,\mathbf{y}_{2,1}) \arrow[d] & (\mathbf{u}_2,\mathbf{y}_{2,2}) \arrow[d] & (\mathbf{u}_2,\mathbf{y}_{2,3}) \arrow[d] & \cdots \arrow[d] & (\mathbf{u}_2,\mathbf{y}_{2,n}) \arrow[d] \\
\text{Day 3} & (\mathbf{u}_3,\mathbf{y}_{3,1}) \arrow[d] & (\mathbf{u}_3,\mathbf{y}_{3,2}) \arrow[d] & (\mathbf{u}_3,\mathbf{y}_{3,3}) \arrow[d] & \cdots \arrow[d] & (\mathbf{u}_3,\mathbf{y}_{3,n}) \arrow[d] \\
\vdots & \vdots \arrow[d] & \vdots \arrow[d] & \vdots \arrow[d] & \ddots \arrow[d] & \vdots \arrow[d] \\
\text{Day n} & (\mathbf{u}_n,\mathbf{y}_{n,1}) & (\mathbf{u}_n,\mathbf{y}_{n,2}) & (\mathbf{u}_n,\mathbf{y}_{n,3}) & \cdots & (\mathbf{u}_n,\mathbf{y}_{n,n}) \\
 & \textbf{end} & \textbf{end} & \textbf{end} & \textbf{end} & \textbf{end}
\end{tikzcd}$$

Figure 4.3: Temporal structure of information flow for multiple 'per hour' initialization of an IOHMM model.

lot of information may be lost in this setup. After all, we now have 24 models that are not sharing information. However, we now also have 24 times as many weights as before. These weights are more capable of retaining information than with a single model. Furthermore, we have now imposed a structure in the data that the model could not have inferred by itself; the 'memory' of the data that the model keeps was not sufficient.

Now, for one day's worth of hourly prices, we have 24 models attempting to find a price for their own hour, for which they have been specifically trained. Naturally, this does increase necessary memory capacity. However the amount of time it takes to train all the data stays roughly the same. The initializing of 24 models naturally takes slightly longer, however when compared to total training time this is negligible. We also now have 24 pairs of error measures being produced after each epoch. We average these error measures to obtain a single pair after each epoch. However it should be noted that as the error measures are computed separately for each hour, the 'best' epoch that is saved for each hour could be different.

4.5.2. Input types

Recall from previous chapters the motivation behind the modelling procedure: we wish to quantify the methodology of a trader. Power traders use various electricity market indicators to try to predict the electricity prices, and so the spreads between electricity prices. We have compiled a list of these market indicators for the Germany-France and Netherlands-Germany borders which can be found in Appendix C. We now explain the various data types and what they represent.

We begin with energy ratings for certain fuel types. We consider sequences of data involving information about certain fuels, for example coal. However fuel data is split up not only into what specific type of fuel it is, but also what the energy rating is for how that fuel is used. Intuitively this makes sense, as energy made by 'low quality' coal has a higher environmental impact and so is typically less wanted than 'high quality' coal. This rating is denoted by a number; the higher the number, the higher the quality. These ratings are also interchangeable between fuels, for example Gas57 has a higher quality than Coal44.

The different types of input data we can encounter are fuel availability, fuel marginal costs, power demand, and actual fuel production. *Fuel availability* is the availability of a certain type of fuel for a specific day for a country, for example how much Gas57 there is available in Germany on a specific day. *Fuel marginal costs* are the difference in costs for specific fuels. While higher quality fuel is preferred, it is often also more expensive. An example of a fuel marginal costs is the difference between Gas57 and Coal44 on a specific day. *Power demand* is the amount of total electricity needed for a specific day in a country. This is usually quite seasonal; considerably more energy is used in winter than in summer in Europe. Finally, *actual fuel production* is the amount of fuel that was used to make electricity in a country on a specific day. Unlike the other fuel related data, the energy rating is not included in this data; here we are talking, for example, of the actual gas production in a country on a specific day.

4.5.3. Feature Selection

Unfortunately, using all 44 possible inputs would make training infeasible. Furthermore, there is the question whether the IOHMM possesses the ability to differentiate between so much data. A study on various IOHMM implementations for predicting financial returns is done in [3]. There the highest number of inputs was chosen to be 14, with the results showing no difficulty in analyzing that amount of data. Therefore we have decided push this even further and use 25 different types of inputs for our model. The choosing of these inputs is called *feature selection*.

We select our input features using a process called Principle Feature Analysis (PFA). The method is based on Principle Component Analysis (PCA). In PCA, a data set of n features are linearly transformed into *n principle components* such that the variance between all components is maximized. One could then pick the top $n-j$ components and have the maximal amount of variance using $n-j$ components that the data will allow. The idea behind PFA is similar, but instead of a list of transformed components with the highest variance, the features with the highest variance with respect to the entire data set are selected. This is preferable for us, as we do not wish to loose the underlying structure of each respective feature. PFA is shown in Algorithm 4 [28]. The reasoning behind this is that the K-mean algorithm clusters the features based on their variability across the chosen dimension. The chosen features closest to the means of the clusters represent their cluster's variability the most. This way we maximize on the variability in the remaining feature set

Algorithm 4 The Principle Feature Analysis (PFA) Algorithm.

```
Require: $X$ = 0-mean matrix containing $n$ features as columns.
1: Compute $\Sigma = \mathrm{Cov}(X)$
2: Compute A such that $\Sigma = A \Lambda A^T$, where diag($\Lambda$) contains the eigenvectors of X.
3: Choose $q$ and compute $A_q$, the matrix comprised of the first $q$ columns of A. The retained variability if your features will be the ratio of the sum of the first $q$ eigenvalues to the sum of all the eigenvalues.
4: Compute the $n$ Principle Components $V_i$. $V_1, V_2, V_3, \dots V_n \in \mathbb{R}^q$ which are the first $n$ rows of $A_q$.
5: Cluster the absolute principle components $|V_1|, |V_2|, |V_3|, \dots |V_n| \in \mathbb{R}^q$ to $p \geq q$ cluster via a K-means clustering algorithm.
6: For each cluster $1, 2, 3, \dots p$, find the vector $V_i$ that is 'closest' to the mean of the cluster. Feature $x_i$ is then the feature you choose, resulting in $p$ features.
```

while still holding as much of the original data as possible. The PFA was performed on the 44 inputs after they were scaled and missing values were interpolated. The chosen 25 are shown in Appendix C.

4.5.4. Data preprocessing

We have now considered all the data and are ready to define how it is fed into the IOHMM model. Recall that this occurs in input-output pairs $(\mathbf{u}_t, \mathbf{y}_t)$, and so that we need the same number of data points for all 25 input sequences in $\mathbf{u}_t$ and the observed sequence $\mathbf{y}_t$. Our training data begins on 1-1-2019, and ends somewhere in the second half of 2021, depending on which month we are trying to predict. In reality, no data set is perfect and so we can encounter a missing value in one of these 26 sequences. We solve this through linear interpolation of the missing values.

We also note that while we have already implemented techniques to reduce the chance of exploding gradients, this is still a possibility. If for example, a particularly large input value gets propagated forward, the resulting back propagated value could start a chain of every increasing gradients. Therefore we normalize our input sequences. We do this per sequences, meaning each input type is normalized separately. We use the following normalizing formulae for each feature u_i

$$\mathrm{norm}([u_i]_1^T) = \frac{[u_i]_1^T - \min([u_i]_1^T)}{\max([u_i]_1^T) - \min([u_i]_1^T)}$$

This further reduces the chance that the gradient becomes unreasonably large.

4.5.5. The Margrabe model

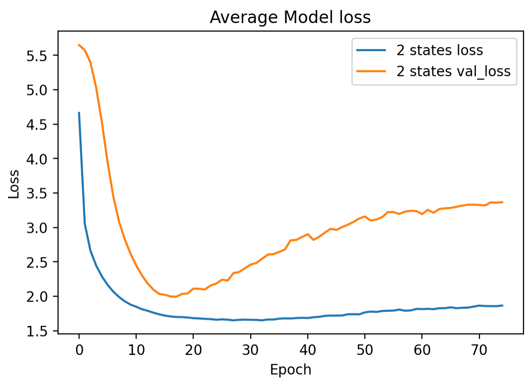
We note here a key difference in the data used by the Margrabe model to that of the IOHMM. The spread data used for the IOHMM model is hourly. However, the Margrabe model uses a type of averaged data. In a given week, all hours can be split into three groups: peaks, off-peaks, and weekends. The peaks consist of the weekday hours of 10:00-22:00; the off-peaks consist of the weekday hours of 00:00-10:00 and 22:00-00:00; the weekends consists of all the hours in the weekend. For each data, the appropriate hours are computed for a peak, off-peak, and/or weekend price. Naturally there is no weekend price for a weekday and no peak or off-peak price in the weekend. The resulting option value is then calculated using these averaged prices, and multiplying them by the amount of hours for which they are applicable that month. Adding these values gives the total monthly option value, similarly to the IOHMM model.

4.6. Hyperparameters

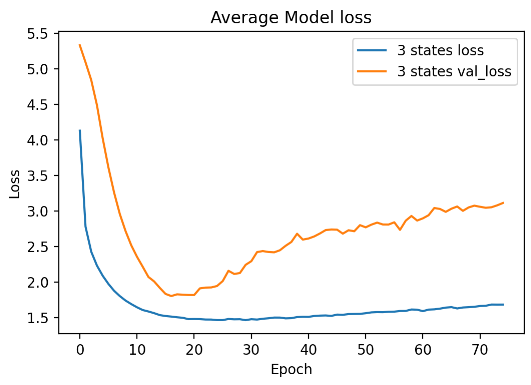
The final steps in our implementation are the hyperparameters: the batch size, the number of epochs, and the number of states. These are the parameters that describe the model itself, and must be decided before attempting to run our final problem. We begin by letting the batch size equal to 1. As mentioned before, the batch size can be used to speed up training through producing stochastic gradients [22]. However due to the recurrent structure of our model, the continuity of how the sequences are processed is important. Therefore, despite the implications for the training time, we keep it at 1. The remainder of the hyperparameters need to be tested with regards to some data.

The number of times the model can 'see' the training data is decided by the number of epochs. For each epoch, the model trains the weights a little further with respect to the entire training set. With this comes the possibility of over-training: the model weights are then over-fit to the training data set instead of to the problem as a whole. In order to find out how many epochs are sufficient, we split the training data into a *testing set* and a *validation set*. We then train the model only on the testing set and evaluate it on validation set as if it were new data. When the error measures for the testing set show better results than the validation set, we know the model has been over-fit. The validation set is typically between 10-20% of the training set [22]. In our case we used 94 days for validation.

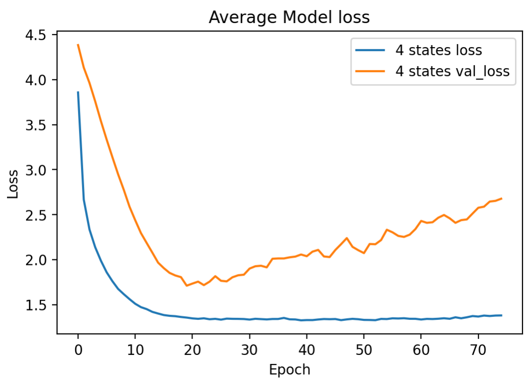
Finally, we have the number of states that we use in the model. The number of states used in an IOHMM are very specific to the problem it is trying to model. We assume the the state variable represents and electricity market in Europe. The IOHMM application to the Spanish electricity market in [21] assumed the market consisted of 4 states. We tested this while using this value as a starting point. We let the model train with 2, 3, 4, and 5 states implemented and selected the number of states with the lowest error measures. Simultaneously we tested the number of epochs by running the models for 75 epochs and graphing the resulting error measures. The hyperparameter testing was done on the Germany-France border for the dates 1-1-2019 to 30-6-2021. The results are shown in Figures 4.4 and 4.5.



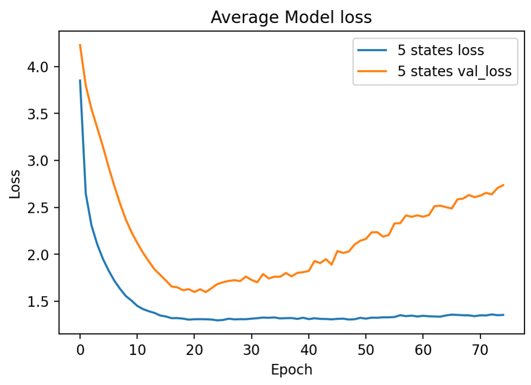
(a) 2 states



(b) 3 states



(c) 4 states



(d) 5 states

Figure 4.4: Average loss and validation loss for 2-5 states

As can be seen in Figures 4.4 and 4.5, 4 states show the best results. Furthermore, the validation error measures show degradation around the 25 epoch mark. In the final implementation, we use 30 epochs and we use an attribute called *callbacks*. This entails that even during training the final model we keep a validation set. The error measure values of this validation set are recorded after each epoch. Then the weights of the epoch with the best validation set are saved for prediction.

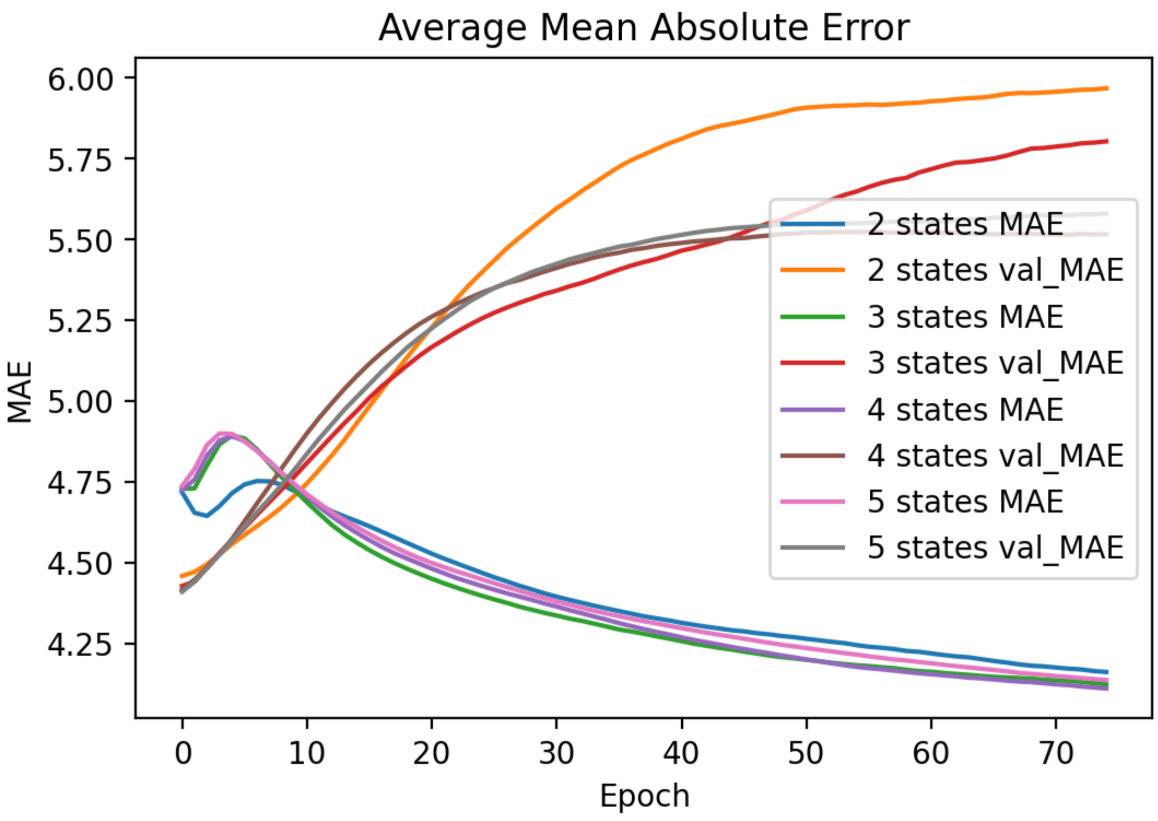


Figure 4.5: Average MAE and validation MAE for 2-5 states

5 Results

We present the results of the model described in the past chapters applied to the cross-border valuation of the Germany-France border and the Netherlands-Germany border. The model was trained using the data from 2019-1-1 to 2021-06-30, as described in the last chapter. The models were trained for 30 epochs, using 4 market states; the training of each model took approximately 1 hour. The average error measures of the 'best' average epoch for each model's validation set are shown in Tables 5.1.

Month	**Germany-France**		**Netherlands-Germany**	
	Loss	MAE	Loss	MAE
July	2.21360	5.76510	1.71822	3.87431
August	1.73384	5.11663	1.27253	4.02100
September	1.94822	5.77923	1.63691	3.99040
October	2.18150	7.86455	2.08989	5.61418

Table 5.1: Average error measures for best training epoch for each model's validation set.

Subsequently, hourly point mass predictions were computed in batches of 1 month; we computed the months of July, August, September, and October. These hourly predictions were used to compute conditional expectations and Greeks for both 'directions' of the capacity flow as described in previous chapters. These values were used to find the monthly cross-border auction price and were then compared to the Margrabe valuation method as well as the JAO auction value. Furthermore, surface plots of the hourly cross-border values and the Greek values were created for each month; this was done for our model as well as the Margrabe model and were then compared. These surface plots for the hourly option price for the month August are included in this chapter; the Greeks for August and the surface plots for the remaining months are shown in Appendix D. Note that the Margrabe Greeks's are computed per separate price process of each country, while the IOHMM Greeks are computed per spread process. Furthermore, as Vega and Theta cannot be computed directly for the IOHMM, these surface plots have also not been produced for the Margrabe model. Finally, we tested whether the model had accurately mapped the market through the shifting of key inputs.

5.1. Valuation

5.1.1. Germany-France

The monthly valuation of cross-border capacity for the Germany to France flow using the IOHMM, Margrabe model, and JAO auction are shown in Table 5.2 and Figure 5.1.

Month	**IOHMM**	**Margrabe**	**JAO**
July	0.85	1.73	1.79
August	1.46	1.28	1.62
September	1.48	3.69	3.51
October	1.59	16.11	18.27

Table 5.2: Comparison of the cross-border capacity from the Germany to France valuation.

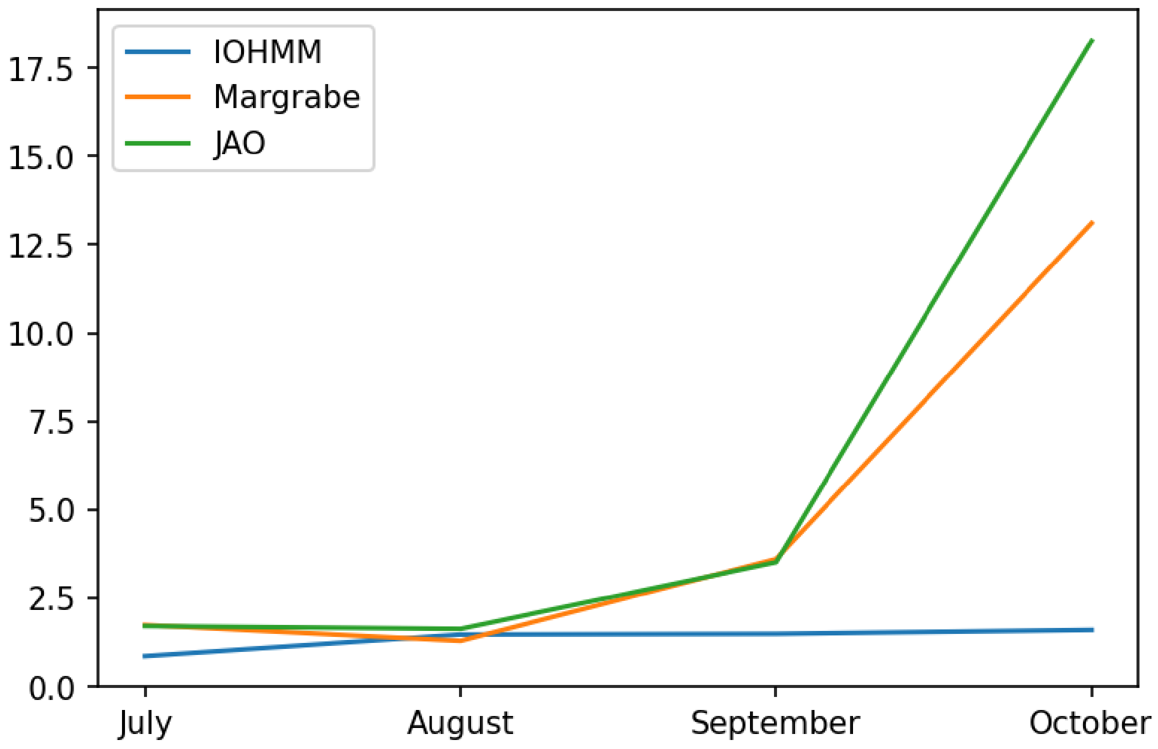
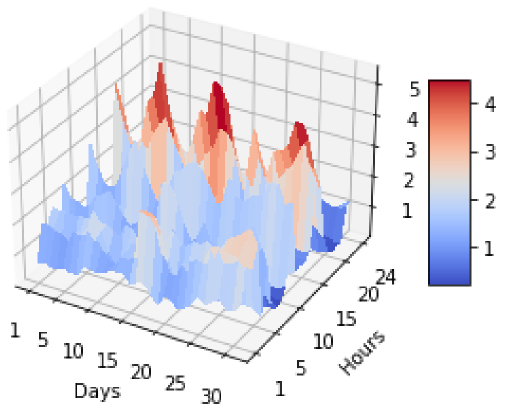


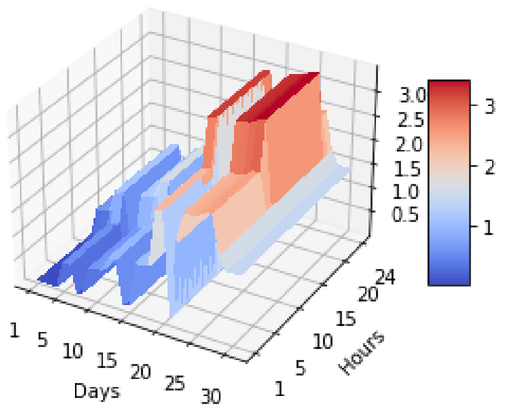
Figure 5.1: Comparison of the cross-border capacity from the Germany to France valuation.

Figures 5.2 and 5.4 show surface plots for the hourly cross-border value generated by the IOHMM and the Margrabe model for the month August for Germany to France and France to Germany, respectively.

The monthly valuation of cross-border capacity for the France to Germany flow using the IOHMM, Margrabe model, and JAO auction are shown in Table 5.3 and Figure 5.3.



(a) IOHMM



(b) Margrabe

Figure 5.2: Germany to France spread option valuation surface plots for August.

Month	IOHMM	Margrabe	JAO
July	1.05	2.00	2.57
August	3.27	2.26	3.03
September	1.95	2.48	2.36
October	2.63	0.87	1.08

Table 5.3: Comparison of the cross-border capacity from the France to Germany valuation.

5.1.2. Netherlands-Germany

The monthly valuation of cross-border capacity for the Netherlands to Germany flow using the IOHMM, Margrabe model, and JAO auction are shown in Table 5.4 and Figure 5.5.

Month	IOHMM	Margrabe	JAO
July	2.50	1.18	1.09
August	1.45	0.61	0.73
September	1.20	0.68	0.55
October	2.95	1.48	1.51

Table 5.4: Comparison of the cross-border capacity from the Netherlands to Germany valuation.

Figures 5.6 and 5.8 show surface plots for the hourly cross-border value generated by the IOHMM and the Margrabe model for the month August, for Netherlands to Germany and Germany to Netherlands, respectively.

The monthly valuation of cross-border capacity for the Germany to Netherlands flow using the IOHMM, Margrabe model, and JAO auction are shown in Table 5.5 and Figure 5.7.

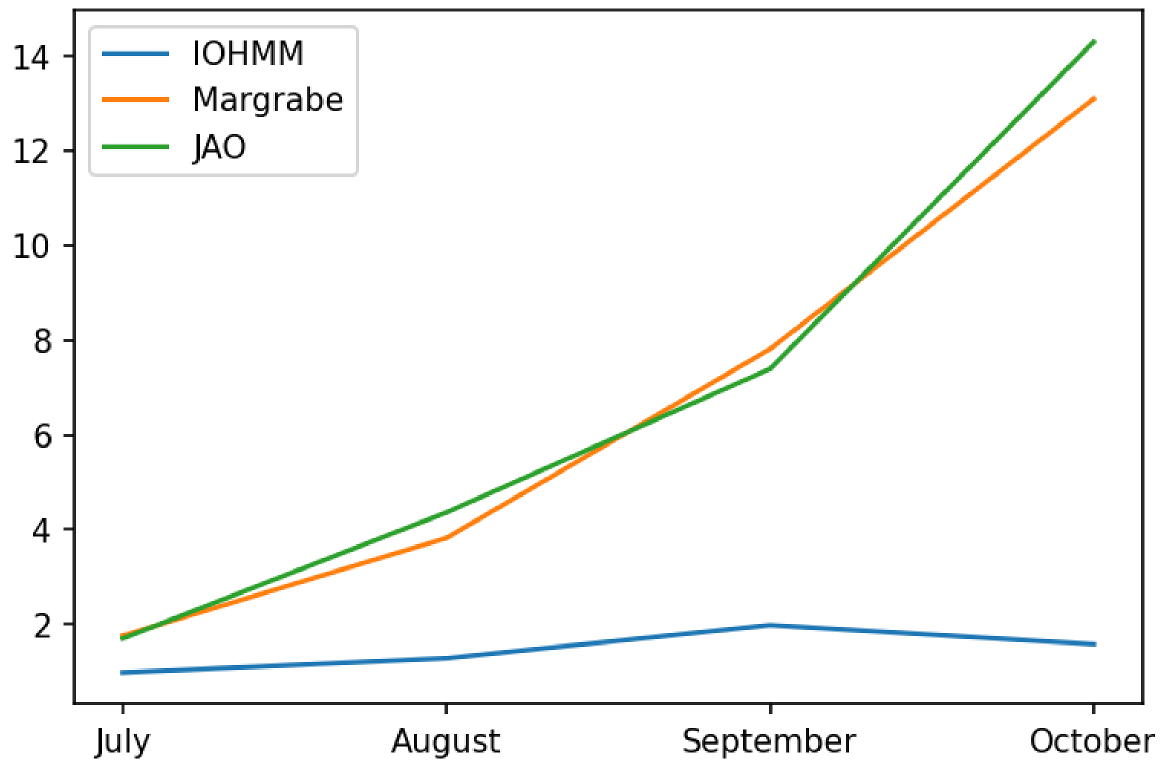


Figure 5.3: Comparison of the cross-border capacity from the France to Germany valuation.

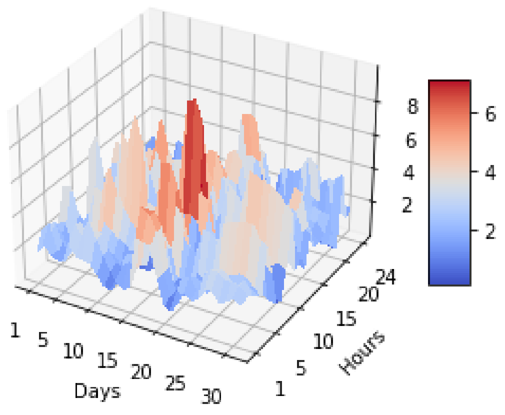
Month	**IOHMM**	**Margrabe**	**JAO**
July	0.98	1.76	1.70
August	1.28	3.82	4.36
September	1.98	7.81	7.39
October	1.58	13.09	14.29

Table 5.5: Comparison of the cross-border capacity from Germany to Netherlands valuation.

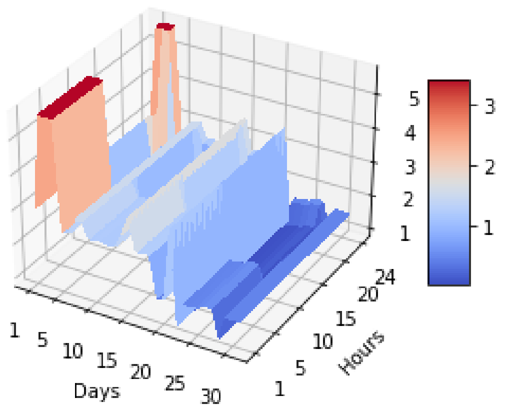
5.2. Shifting of Inputs

Part of the goal of our model is to map certain inputs to the spreads of our respective spot prices. However due to the black-box nature of our machine learning approach, these mappings cannot be observed directly. We can however, shift certain inputs which we know to be important in the market and see how our point mass prediction for the month responds. For example, if we assume that German Demand drops for the entire month, economic theory tells us that the price of German electricity will go down. Therefore, the spread between Germany-France should go down, and the spread between France-Germany should go up.

We test these mappings by shifting key inputs in the markets by 5% and see how our predicted spread react. In the Tables 5.6 and 5.6 show the results. A '✓' shows the spreads move in the expected direction, a '✗' shows the spreads move in the opposite direction, and an empty cell shows the movement was negligible. We used the predicted month of August 2021 for testing.



(a) IOHMM



(b) Margrabe

Figure 5.4: France to Germany spread option valuation surface plots for August.

Action (5%)	**Expected**		**Actual**	
DE Demand ↓	DE-FR ↓	FR-DE ↑		✗
FR Demand ↓	DE-FR ↑	FR-DE ↑	✓	✓
DE Wind ↓	DE-FR ↑	FR-DE ↓		✗
DE Solar ↓	DE-FR ↑	FR-DE ↓		✗
FR Nuclear ↓	DE-FR ↓	FR-DE ↓	✓	✓

Table 5.6: Expected and actual response of the Germany-France IOHMM valuation to a 5% decrease in key inputs.

Action (5%)	**Expected**		**Actual**	
NL Demand ↓	NL-DE ↓	DE-NL ↑	✓	✓
DE Wind ↓	NL-DE ↓	DE-NL ↑		
DE Solar ↓	NL-DE ↓	DE-NL ↑	✗	
NL Gas ↓	NL-DE ↑	DE-NL ↓	✓	✓

Table 5.7: Expected and actual response of the Netherlands-Germany IOHMM valuation to a 5% decrease in key inputs.

An increase in these key inputs show the same mapping relationship results and have therefore been omitted.

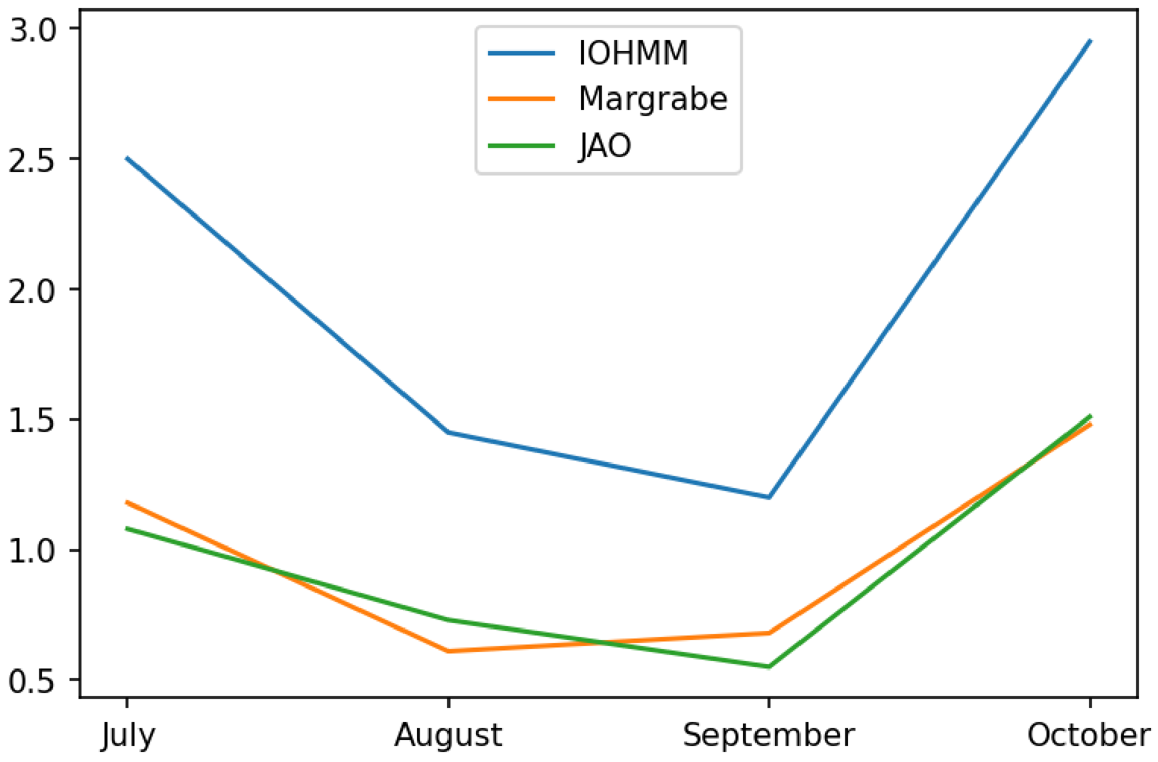
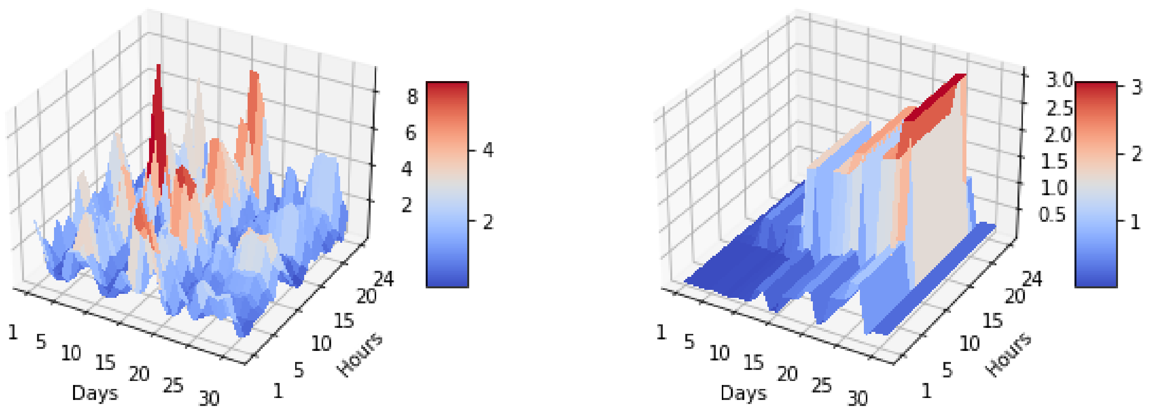


Figure 5.5: Comparison of the cross-border capacity from the Netherlands to Germany valuation.



(a) IOHMM

(b) Margrabe

Figure 5.6: Netherlands to Germany spread option valuation surface plots for August.

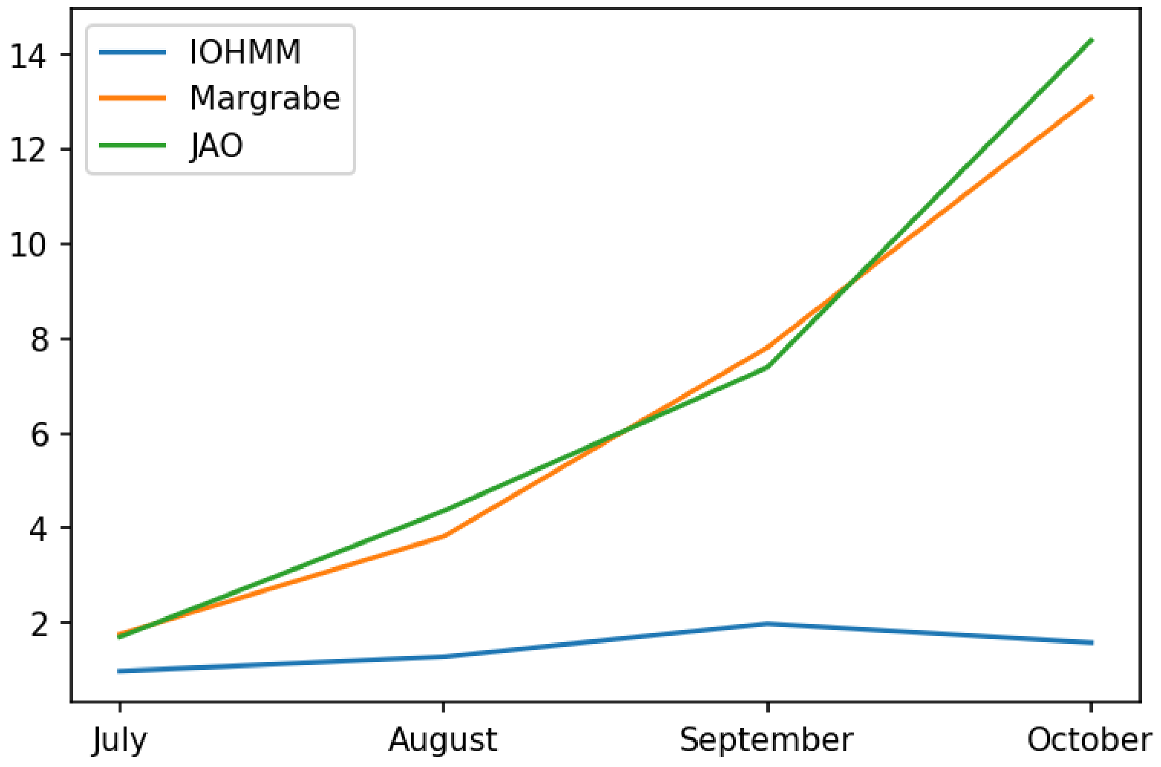
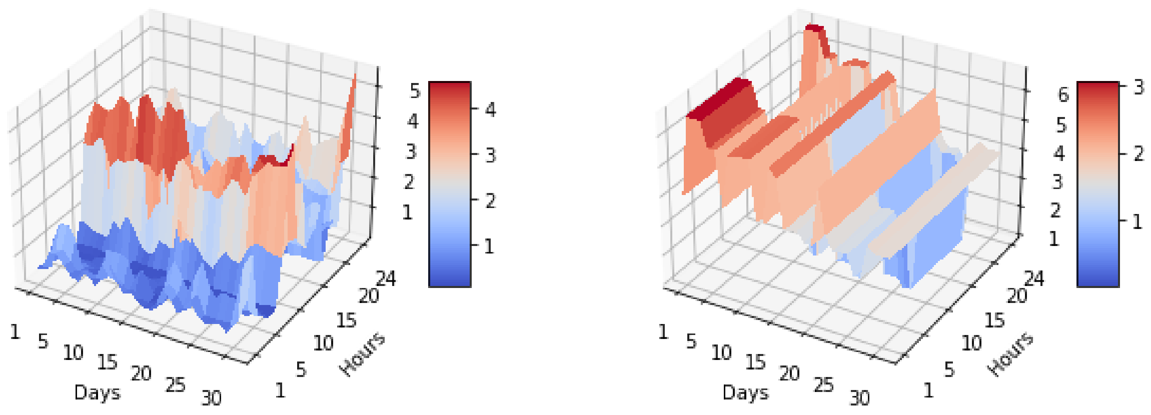


Figure 5.7: Comparison of the cross-border capacity from the Germany to Netherlands valuation.



(a) IOHMM

(b) Margrabe

Figure 5.8: Germany to Netherlands spread option valuation surface plots for August.

6 Conclusion

In this chapter we discuss the results of the previous chapter and draw some conclusions as to the performance of the model. We compare the model results for the months of August until October to the Margrabe model and the realized JAO results. Furthermore, we show some improvements to the Margrabe pricing strategy of the cross-border capacity. Possible motivation for the deviation within these results is also discussed. Finally, we discuss how the mapping of inputs to output has fared in the results.

In this thesis we attempt to find the ’true’ value for the cross-border capacity between two countries. As explained previously, the auctioning method that the market uses with JAO does not necessarily reflect this true value. The Margrabe method attempts to predict the JAO price and therefore also does not necessarily show the true value. This explains the discrepancy in the monthly values shown in Tables 5.2-5.5. As the Margrabe method attempts to predict the JAO price, it produces results that for almost all months lie closer to this value than the IOHMM prices. While in some case (for example, the valuation from France to Germany in August as shown in Table 5.3) the predicted model value lies quite close to the JAO value, for the majority of the months the model values were considerably different. We note from Figure 5.5 that the IOHMM model values the cross-border capacity higher than the JAO auction only for the Netherlands to Germany capacity; all other capacities are consistently valued lower.

Looking at Figures 5.1-5.7, we see the the ’trend’ within the market in these months. We see that from Figure 5.5 that for the Netherlands to Germany capacity the market trend was captured accurately; the IOHMM model trend follows the JAO trend more accurately than even the Margrabe model. For the other borders, however, this is not the case. There are two reasons for this. Firstly, it is possible that the IOHMM model has not captured the market accurately and can therefore not predict the trend accurately; we will explore this when considering the shifting of input results. Secondly, the data for the months of September and October is most likely different than that of the training data. As can be seen when comparing electricity prices of the last decade, the electricity market is showing previously unknown levels of volatility [18]. These levels of volatility lead to very high prices, and so very high valuations of the cross-border capacity. This is an issue, as it is difficult for a machine learning algorithm to predict values so far away from what it has seen before.

The Margrabe model, while good at providing a single price for the entire month of cross-border capacity, does lack certain insights to the specific hourly capacities. Here the IOHMM model is a large improvement. Figures 5.2-5.8 show surface plots for the IOHMM model and the Margrabe model valuations for all border for August. We see that there is much more accurate representations of which hourly options provide the value of the monthly option bundle. For example, where the Margrabe model show which weekday peaks have a high value, the IOHMM model shows which specific hour is most valuable. This type of data can help in better predicting the underlying market dynamics, as well as help with trading strategy. This increased accuracy is also extended to the Greeks, for which an hourly surface plot has also been created. Having knowledge of the hourly value of the Greeks as compared to 12 or 24 hour bundles as with the Margrabe model can improve hedging strategy. These plots as well as the surface plots for July, September, and October can be found in Appendix D.

As shown in Tables 5.6-5.7, we have also tested whether the model accurately captured the model by shifting key inputs for each border and seeing whether the capacity price shifts accordingly. We see that the German inputs cause some trouble, while the French and Dutch inputs have been mapped accurately. We therefore conclude that some part of the German market has not been mapped properly. This can be due to using the incorrect inputs; if some key component has not been included a correct mapping would not be possible. The problem of the Germany market could also be too difficult for the model to learn. For example, electricity prices show a certain amount of seasonality throughout the year. However, the weights in the IOHMM are kept static outside of each training epoch; this means that the model attempts to fit each weight to the entire year, essentially ignoring this seasonality. If the seasonality has a big enough impact, this could make the problem too difficult for the model to learn. This is most likely also why the months of September and October show such different results when compared to the JAO auction.

7 Discussion

In this chapter we briefly discuss how the work in this thesis can be continued. We begin with research pertaining to the electricity market. We also explore certain mathematical approaches, especially through different processes. Finally, we touch upon expansions of the model through machine learning.

As explained in the previous chapters, understanding how a market works is key to predicting prices and derivatives in that market. In this work we have done a preliminary analysis in the underlying features of the electricity market and used these features as inputs for our model. A more careful approach to this input selection could be beneficial in the valuing of the cross-border capacity, whether these features are used an inputs or not. For example, analysis on the seasonality of the input data or realized prices would be proactive in the predictive process. Certain inputs could be shown to be relevant 'all-year' or only in certain months. Furthermore, using deseaonalized prices could aid in capturing trends.

The choice of underlying process has a large influence on the predictive power of a model. Where we used a Normal Inverse Gaussian as a price process, others could be used as well. As referred to in previous chapters, the stochastic processes with stochastic volatility offer a wide variety of modelling opportunities. Given such a process, certain seasonal components could then be implemented directly; furthermore research in the fitting of such a process to prices is also needed should this approach be used. Processes with a time parameters (i.e. non-stationary processes) should also be considered. We used a stationary process where non-stationarity was introduced through the IOHMM outputs. Using a non-stationary process directly seems like a logical extension of this. Multiple temporal structures could be created; through the time parameter and through the IOHMM output. Furthermore, should a volatility parameter also be added, then Theta and Vega could also be computed, as with a Black-Scholes framework.

Finally, the IOHMM architecture that was used can also be expanded. In the analysis of the Spanish electricity market [21], an online updating implementation of the IOHMM was used. This means that, with each new month of data, only the new data needs to be trained, not the entire data set. This would greatly speed up training, should it be combined with our implementation. The recurrent aspect of an IOHMM has the ability 'remember' information from the previous hour. There are, however, certain machine learning architectures that can remember across multiple points; these are called long-term short-term memory models (LSTM) [22]. Should such a model be combined with an IOHMM architecture, more information could be used from the past when trying to model market dynamics and predict prices. This could, for example, help the model solve the seasonality of the market while training.

A
Appendix: Derivation of Margrabe Greeks

We refer to the following formulas throughout the entire derivation [29].

$$V(S_1, S_2, t) = S_1(t)\Phi(d_1) - S_2(t)\Phi(d_2), \tag{A.1}$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\log\left(\frac{S_1(t)}{S_2(t)}\right) + \frac{\sigma^2}{2}(T-t)\right], \tag{A.2}$$

$$d_2 = \frac{1}{\sigma\sqrt{T-t}}\left[\log\left(\frac{S_1(t)}{S_2(t)}\right) - \frac{\sigma^2}{2}(T-t)\right] = d_1 - \sigma\sqrt{T-t}, \tag{A.3}$$

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}, \tag{A.4}$$

where $\Phi(x)$ is the cumulative distribution function of a standard Normal distribution.

A.1. Delta 1

We begin from equation (A.1)

$$\Delta_1 = \frac{\partial V}{\partial S_1(t)} = \Phi(d_1) + S_1(t)\frac{\partial d_1}{\partial S_1(t)}\phi(d_1) - S_2(t)\frac{\partial d_2}{\partial S_1(t)}\phi(d_2),$$

where $\phi(x)$ is the probability density function of a standard Normal distribution. We see from equations (A.2) and (A.3) that

$$\begin{aligned}
\phi(d_2) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}d_2^2\right) \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left[\frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma\sqrt{T-t}} - \frac{\sigma\sqrt{T-t}}{2}\right]^2\right) \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left[\frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma\sqrt{T-t}} + \frac{\sigma\sqrt{T-t}}{2}\right]^2 + \log\left(\frac{S_1(t)}{S_2(t)}\right)\right) \\
&= \frac{S_1(t)}{S_2(t)}\phi(d_1).
\end{aligned} \tag{A.5}$$

Substituting this gives

$$\Delta_1 = \Phi(d_1) + S_1(t)\frac{\partial d_1}{\partial S_1(t)}\phi(d_1) - S_2(t)\frac{\partial d_2}{\partial S_1(t)}\phi(d_1).$$

We also note from equations (A.2) and (A.3) that

$$\frac{\partial d_1}{\partial S_1(t)} = \frac{\partial d_2}{\partial S_1(t)} = \frac{1}{S_1(t)\sigma\sqrt{T-t}}, \tag{A.6}$$

and so we see that

$$\Delta_1 = \Phi(d_1), \tag{A.7}$$

A.2. Delta 2

We begin from equation (A.1)

$$\Delta_2 = \frac{\partial V}{\partial S_2(t)} = S_1(t)\frac{\partial d_1}{\partial S_2(t)}\phi(d_1) - \Phi(d_2) - S_2(t)\frac{\partial d_2}{\partial S_2(t)}\phi(d_2),$$

where $\phi(x)$ is the probability density function of a standard Normal distribution. Using equations (A.5), we see that

$$\Delta_2 = \frac{\partial V}{\partial S_2(t)} = S_2(t)\frac{\partial d_1}{\partial S_2(t)}\phi(d_2) - \Phi(d_2) - S_2(t)\frac{\partial d_2}{\partial S_2(t)}\phi(d_2).$$

We also note from equations (A.2) and (A.3) that

$$\frac{\partial d_1}{\partial S_2(t)} = \frac{\partial d_2}{\partial S_1(2)} = \frac{1}{S_2(t)\sigma\sqrt{T-t}},$$

and so we see that

$$\Delta_2 = -\Phi(d_2), \tag{A.8}$$

A.3. Gamma 11

Using equation (A.7) and (A.6)

$$\Gamma_{11} = \frac{\partial \Delta_1}{\partial S_1(t)} = \frac{\partial d_1}{\partial S_1(t)}\phi(d_1) = \frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_1)}{S_1(t)}$$

A.4. Gamma 22

Using (A.8) we see that

$$\Gamma_{22} = \frac{\partial \Delta_1}{\partial S_1(t)} = \frac{\partial d_1}{\partial S_1(t)}\phi(d_1)$$

Then we see that

$$\frac{\partial d_1}{\partial S_2(t)} = \frac{\partial d_2}{\partial S_2(t)} = \frac{1}{S_2(t)\sigma\sqrt{T-t}}, \tag{A.9}$$

and so

$$\Gamma_{22} = \frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_2)}{S_2(t)}$$

A.5. Gamma 12

Using equation (A.8) and (A.9)

$$\Gamma_{12} = \frac{\partial \Delta_1}{\partial S_2(t)} = \frac{\partial d_1}{\partial S_2(t)}\phi(d_1) = \frac{1}{\sigma\sqrt{T-t}}\frac{\phi(d_1)}{S_2(t)}.$$

We note that the using equation (A.8) in combination with equation (A.6) would lead to $\Gamma_{21} = \Gamma_{12}$.

A.6. Vega 1

We begin from equation (A.1)

$$\nu_1 = \frac{\partial V}{\partial \sigma_1} = S_1(t)\phi(d_1)\frac{\partial d_1}{\partial \sigma_1} - S_2(t)\phi(d_2)\frac{\partial d_2}{\partial \sigma_1}.$$

Using (A.4), we see that

$$\frac{\partial d_1}{\partial \sigma} = -\frac{1}{\sigma^2\sqrt{T-t}}\left[\log\left(\frac{S_1(t)}{S_2(t)}\right) + \frac{\sigma^2}{2}T\right] + \frac{1}{\sigma\sqrt{T-t}}\sigma T = -\frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma^2\sqrt{T-t}} + \frac{\sqrt{T-t}}{2}, \tag{A.10}$$

and

$$\frac{\partial d_2}{\partial \sigma} = \frac{\partial d_1}{\partial \sigma} - \sqrt{T-t} = -\frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma^2\sqrt{T-t}} - \frac{\sqrt{T-t}}{2}. \tag{A.11}$$

Furthermore, we note that

$$\frac{\partial \sigma}{\partial \sigma_1} = \frac{1}{2\sigma}\left(2\sigma_1 - 2\rho\sigma_2\right) = \frac{\sigma_1 - \rho\sigma_2}{\sigma}.$$

Using these equations in combination with equation (A.5), we see that

$$\begin{aligned}
\nu_1 &= S_1(t)\phi(d_1)\left(\frac{\partial d_1}{\partial \sigma} - \frac{\partial d_2}{\partial \sigma}\right)\frac{\partial \sigma}{\partial \sigma_1} \\
&= S_1(t)\phi(d_1)\left(-\frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma^2\sqrt{T-t}} + \frac{\sqrt{T-t}}{2} + \frac{\log\left(\frac{S_1(t)}{S_2(t)}\right)}{\sigma^2\sqrt{T-t}} + \frac{\sqrt{T-t}}{2}\right)\frac{\sigma_1 - \rho\sigma_2}{\sigma} \\
&= S_1(t)\phi(d_1)\sqrt{T-t}\frac{\sigma_1 - \rho\sigma_2}{\sigma}
\end{aligned}$$

A.7. Vega 2

We begin from equation (A.1)

$$\nu_2 = \frac{\partial V}{\partial \sigma_2} = S_1(t)\phi(d_1)\frac{\partial d_1}{\partial \sigma_2} - S_2(t)\phi(d_2)\frac{\partial d_2}{\partial \sigma_2}.$$

Furthermore, we note that

$$\frac{\partial \sigma}{\partial \sigma_2} = \frac{1}{2\sigma}\left(2\sigma_2 - 2\rho\sigma_1\right) = \frac{\sigma_2 - \rho\sigma_1}{\sigma}.$$

Using this and equations (A.10) and (A.11) we see that

$$\begin{aligned}
\nu_2 &= S_1(t)\phi(d_1)\left(\frac{\partial d_1}{\partial \sigma} - \frac{\partial d_2}{\partial \sigma}\right)\frac{\partial \sigma}{\partial \sigma_2} \\
&= S_1(t)\phi(d_1)\sqrt{T-t}\frac{\sigma_2 - \rho\sigma_1}{\sigma}
\end{aligned}$$

A.8. Theta

We begin from equation (A.1)

$$\Theta = \frac{\partial V}{\partial t} = S_1(t)\phi(d_1)\frac{\partial d_1}{\partial t} - S_2(t)\phi(d_2)\frac{\partial d_2}{\partial t}.$$

From equation A.5 we see that this becomes

$$\Theta = S_1(t)\phi(d_1)\left(\frac{\partial d_1}{\partial t} - \frac{\partial d_2}{\partial t}\right)$$

Now we see from equations A.2 and A.3 that

$$\frac{\partial d_2}{\partial t} = \frac{\partial d_1}{\partial t} - \frac{\sigma}{2\sqrt{T-t}},$$

and so

$$\Theta = \frac{S_1(t)\sigma}{2\sqrt{T-t}}\phi(d_1)$$

B
Appendix: Keras Implementation of IOHMM

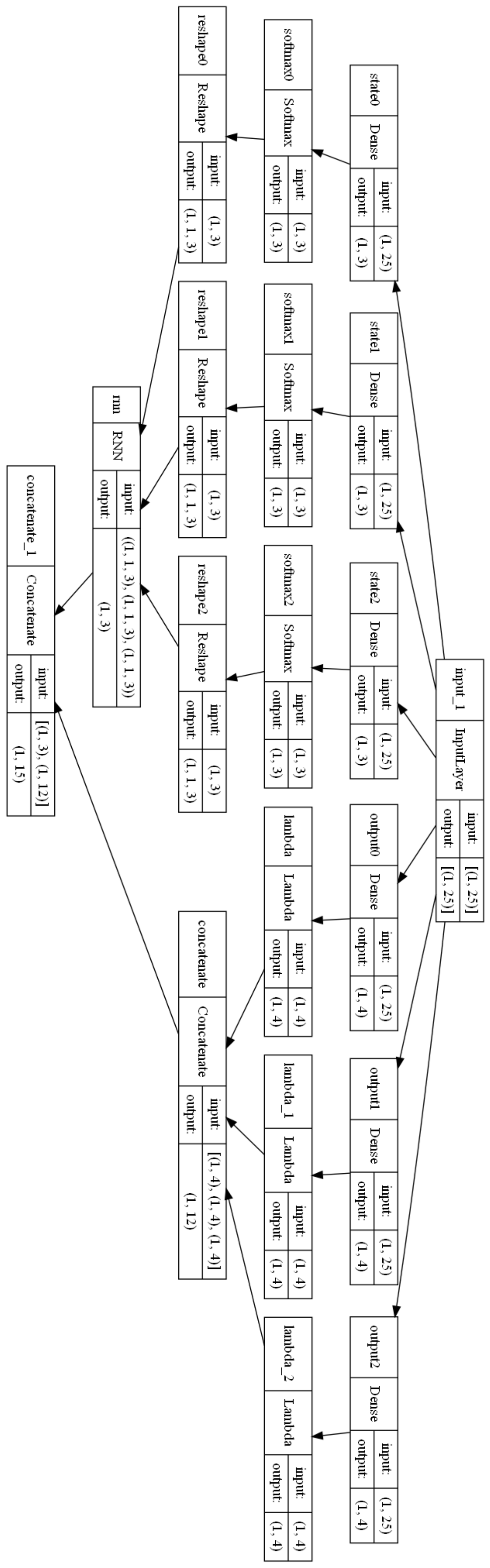

input_1 | InputLayer | input: [(1, 25)] | output: [(1, 25)]
state0 | Dense | input: (1, 25) | output: (1, 3)
state1 | Dense | input: (1, 25) | output: (1, 3)
state2 | Dense | input: (1, 25) | output: (1, 3)
output0 | Dense | input: (1, 25) | output: (1, 4)
output1 | Dense | input: (1, 25) | output: (1, 4)
output2 | Dense | input: (1, 25) | output: (1, 4)
softmax0 | Softmax | input: (1, 3) | output: (1, 3)
softmax1 | Softmax | input: (1, 3) | output: (1, 3)
softmax2 | Softmax | input: (1, 3) | output: (1, 3)
lambda | Lambda | input: (1, 4) | output: (1, 4)
lambda_1 | Lambda | input: (1, 4) | output: (1, 4)
lambda_2 | Lambda | input: (1, 4) | output: (1, 4)
reshape0 | Reshape | input: (1, 3) | output: (1, 1, 3)
reshape1 | Reshape | input: (1, 3) | output: (1, 1, 3)
reshape2 | Reshape | input: (1, 3) | output: (1, 1, 3)
concatenate | Concatenate | input: [(1, 4), (1, 4), (1, 4)] | output: (1, 12)
rnn | RNN | input: ((1, 1, 3), (1, 1, 3), (1, 1, 3)) | output: (1, 3)
concatenate_1 | Concatenate | input: [(1, 3), (1, 12)] | output: (1, 15)

Figure B.1: Graphical representation of the IOHMM layering chain in Keras

C

Appendix: All Possible Inputs

Chosen Inputs	**Rejected Inputs**
DE Demand	DE Coal40 availability
DE Gas36 availability	DE Gas50 availability
DE Gas57 availability	DE Gas42 availability
DE Uranium availability	DE Lignite
DE Coal37 availability	FR Wind
DE Lignite availability	FR Gas50 availability
DE Coal32 availability	FR Nukes
DE Gas production	Gas50-Coal40 (Marginal Cost Spread)
DE Solar production	Gas50-Coal44 (Marginal Cost Spread)
DE Coal production	Gas50-Lig32 (Marginal Cost Spread)
DE Wind production	Gas50-Lig36 (Marginal Cost Spread)
FR Solar production	Gas50-Lig40 (Marginal Cost Spread)
FR Hydro production	Gas57-Coal40 (Marginal Cost Spread)
FR Demand	Gas57-Lig32 (Marginal Cost Spread)
FR Coal37 availability	Coal40-Coal44 (Marginal Cost Spread)
FR Uranium availability	Coal44-Lig36 (Marginal Cost Spread)
Gas57-Coal44 (Marginal Cost Spread)	Coal44-Lig40 (Marginal Cost Spread)
Coal40-Lig32 (Marginal Cost Spread)	Lig32-Lig36 (Marginal Cost Spread)
Coal44-Lig32 (Marginal Cost Spread)	Lig32-Lig40 (Marginal Cost Spread)
Gas57-Lig40 (Marginal Cost Spread)	
Lig36-Lig40 (Marginal Cost Spread)	
Coal40-Lig36 (Marginal Cost Spread)	
Coal40-Lig40 (Marginal Cost Spread)	
Gas50-Gas57 (Marginal Cost Spread)	
Gas57-Lig36 (Marginal Cost Spread)	

Table C.1: Chosen and rejected inputs by the PFA for the Germany-France border

Chosen Inputs	**Rejected Inputs**
NL Demand	NL Coal40 availability
NL Coal production	DE Lignite availability
NL Wind Offshore production	DE Gas57 availability
NL Wind Onshore production	DE Gas50 availability
NL Gas production	DE Demand
NL Uranium availability	DE Gas production
NL Gas50 availability	Gas50-Coal44 (Marginal Costs Spread)
NL Gas57 availability	Gas50-Lig32 (Marginal Costs Spread)
DE Coal production	Gas50-Lig36 (Marginal Costs Spread)
DE Lignite production	Gas50-Lig40 (Marginal Costs Spread)
DE Solar production	Gas57-Coal40 (Marginal Costs Spread)
DE Wind production	Gas57-Coal44 (Marginal Costs Spread)
DE Coal40 availability	Gas57-Lig32 (Marginal Costs Spread)
DE Coal37 availability	Coal40-Coal44 (Marginal Costs Spread)
DE Uranium availability	Coal40-Lig32 (Marginal Costs Spread)
DE Gas36 availability	Coal40-Lig36 (Marginal Costs Spread)
DE Coal32 availability	Coal40-Lig40 (Marginal Costs Spread)
DE Gas42 availability	Coal44-Lig32 (Marginal Costs Spread)
Gas50-Coal40 (Marginal Costs Spread)	Lig32-Lig40 (Marginal Costs Spread)
Coal44-Lig36 (Marginal Costs Spread)	Lig36-Lig40 (Marginal Costs Spread)
Gas57-Lig36 (Marginal Costs Spread)	
Coal44-Lig40 (Marginal Costs Spread)	
Gas57-Lig40 (Marginal Costs Spread)	
Gas50-Gas57 (Marginal Costs Spread)	
Lig32-Lig36 (Marginal Costs Spread)	

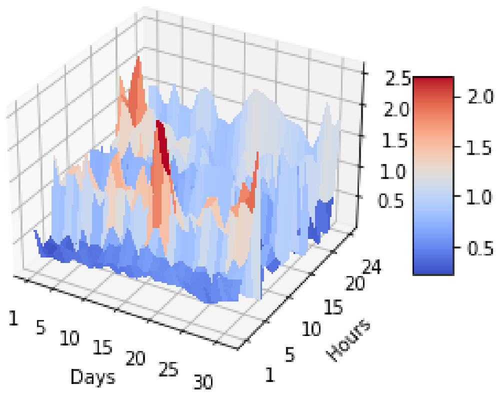
Table C.2: Chosen and rejected inputs by the PFA for the Netherlands-Germany border

D

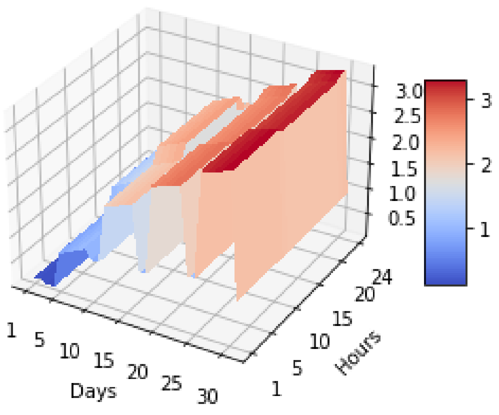
Appendix: Surface Plots

D.1. July

D.1.1. Germany-France

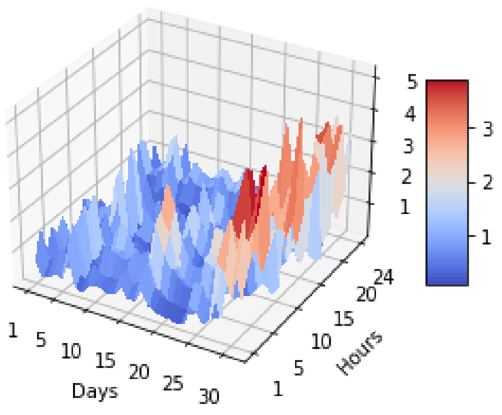


(a) IOHMM

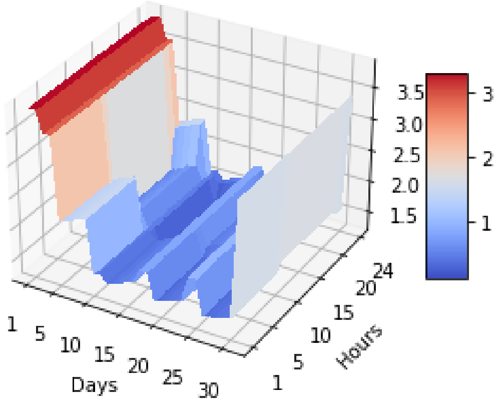


(b) Margrabe

Figure D.1: Germany to France spread option valuation surface plots for July.

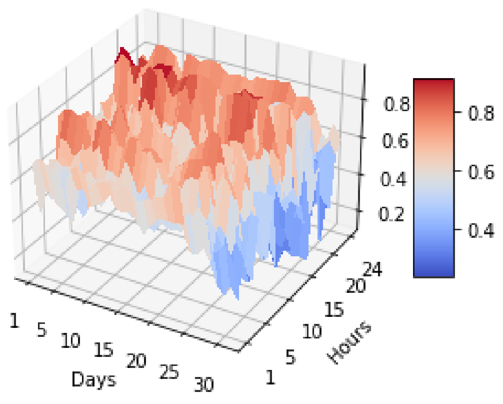


(a) IOHMM

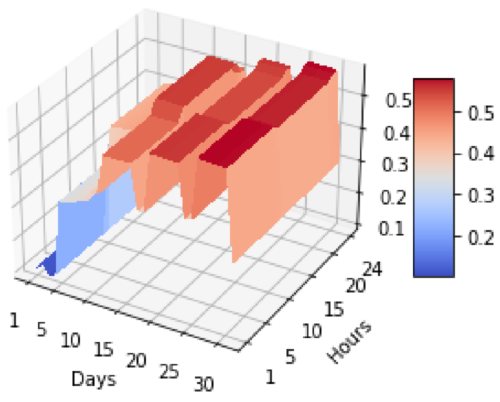


(b) Margrabe

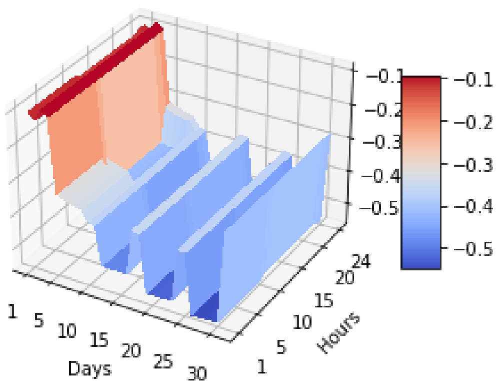
Figure D.2: France to Germany spread option valuation surface plots for July.



(a) IOHMM

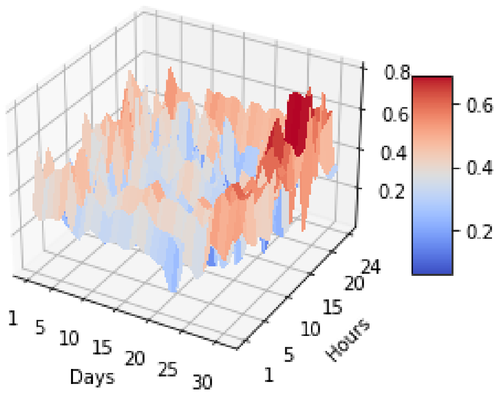


(b) Margrabe Delta 1

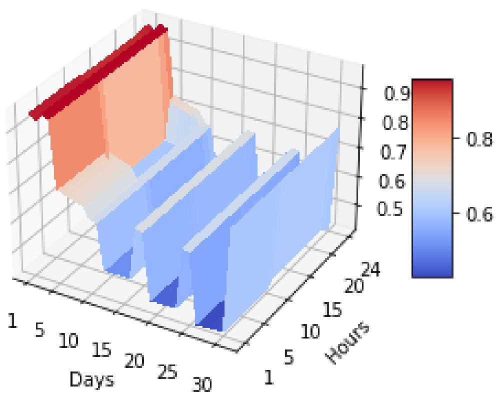


(c) Margrabe Delta 2

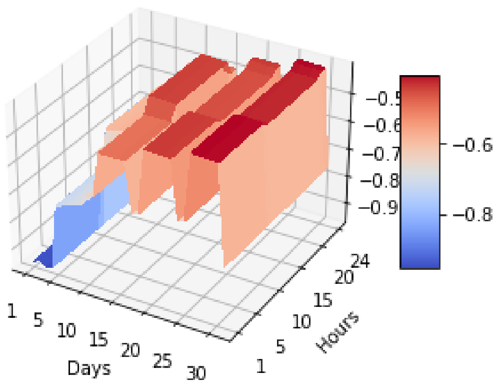
Figure D.3: Germany to France Delta surface plots for July.



(a) IOHMM

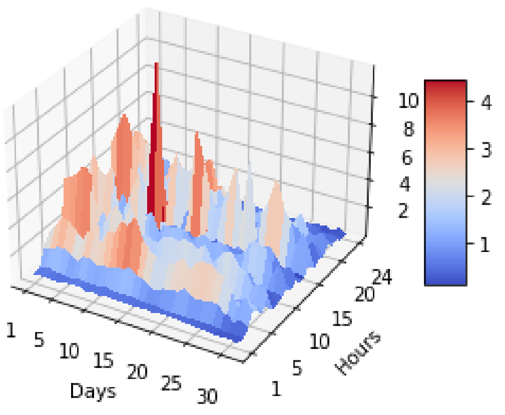


(b) Margrabe Delta 1

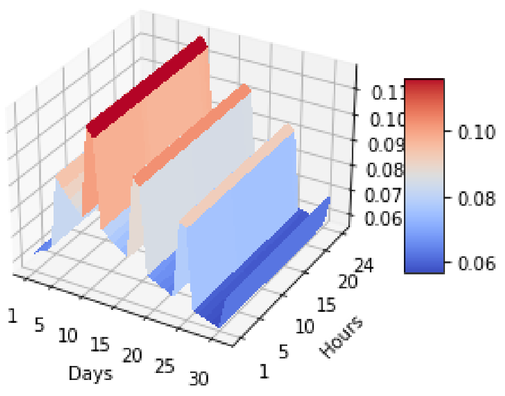


(c) Margrabe Delta 2

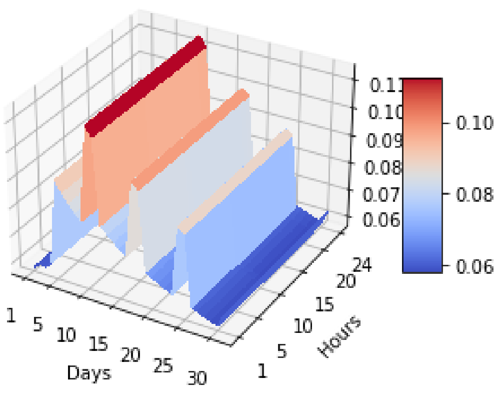
Figure D.4: France to Germany Delta surface plots for July.



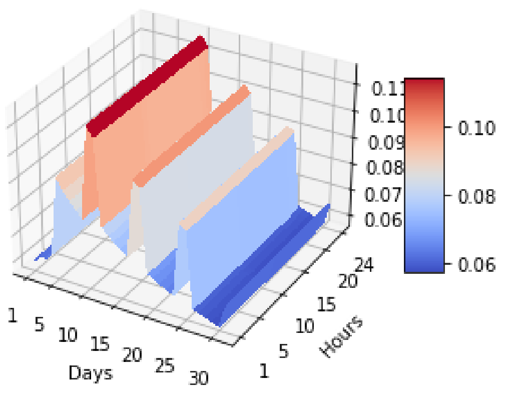
(a) IOHMM Gamma



(b) Margrabe Gamma 11



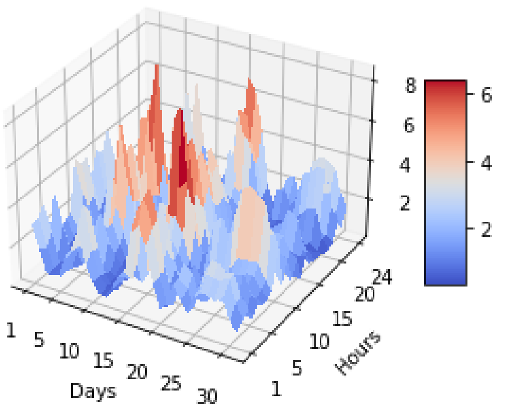
(c) Margrabe Gamma 22



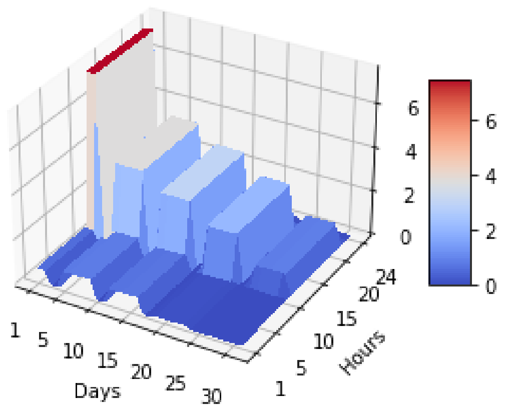
(d) Margrabe Gamma 12

Figure D.5: Germany-France Gamma surface plots for July.

D.1.2. Netherlands-Germany

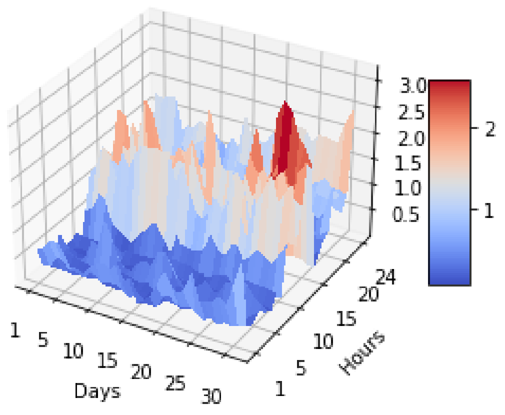


(a) IOHMM

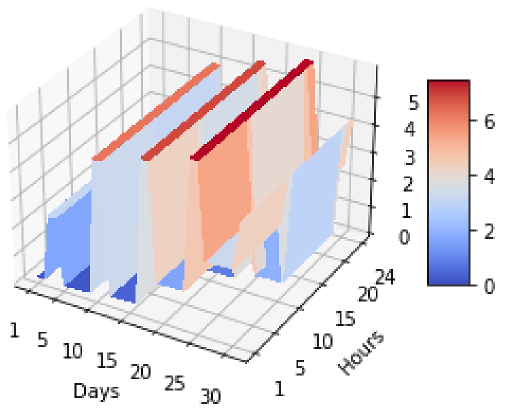


(b) Margrabe

Figure D.6: Netherlands to Germany spread option valuation surface plots for July.

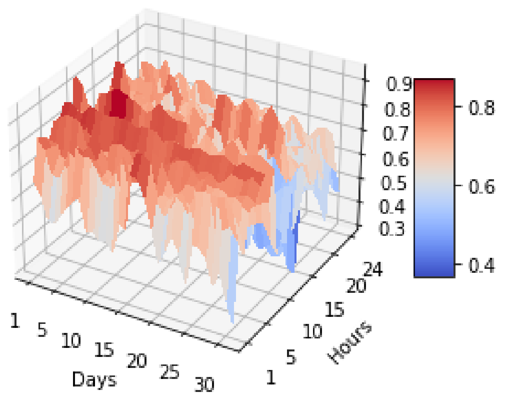


(a) IOHMM

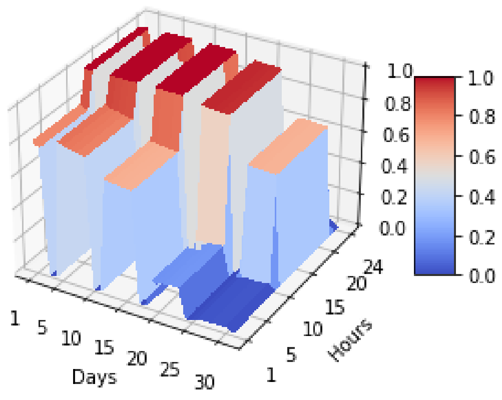


(b) Margrabe

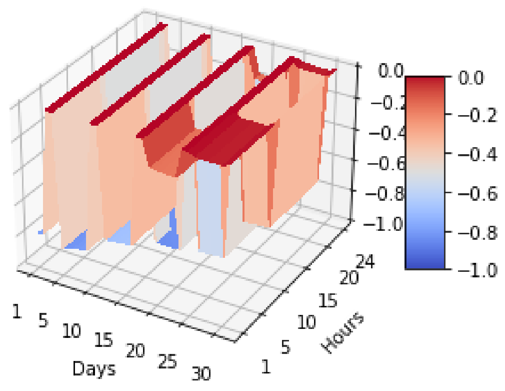
Figure D.7: Germany to Netherlands spread option valuation surface plots for July.



(a) IOHMM

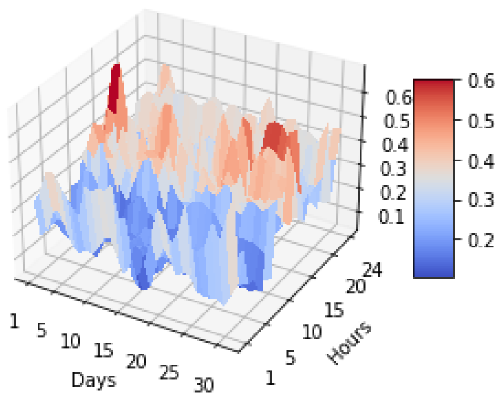


(b) Margrabe Delta 1

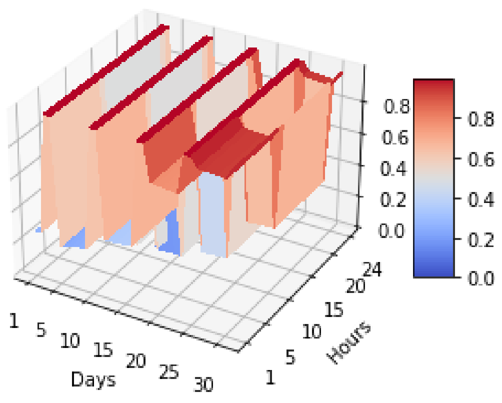


(c) Margrabe Delta 2

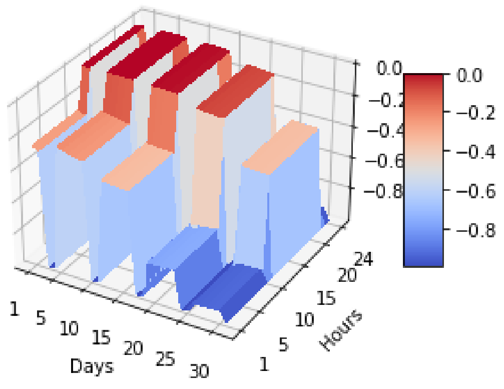
Figure D.8: Netherlands to Germany Delta surface plots for July.



(a) IOHMM

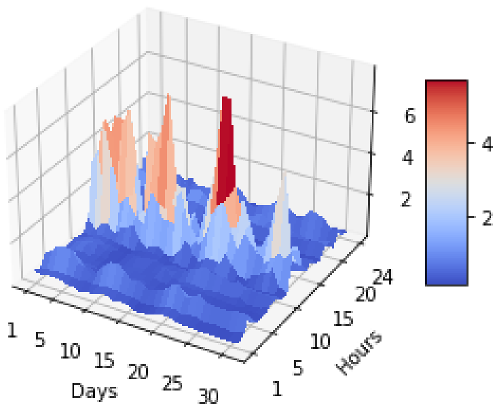


(b) Margrabe Delta 1

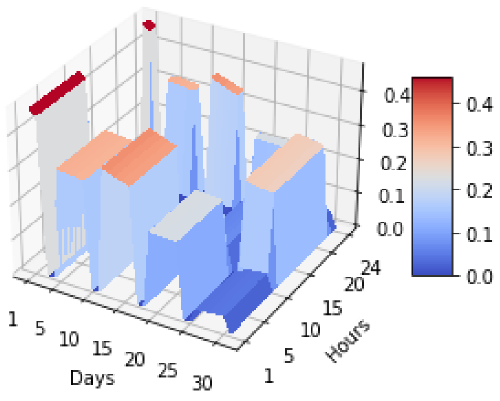


(c) Margrabe Delta 2

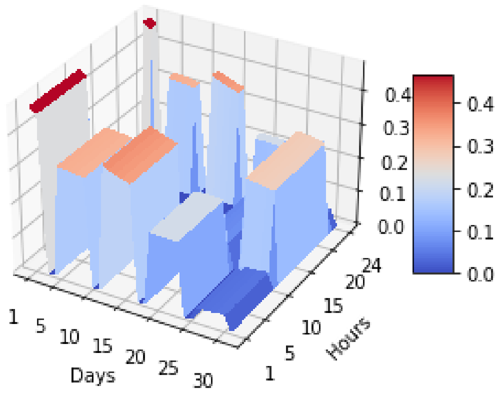
Figure D.9: Germany to Netherlands Delta surface plots for July.



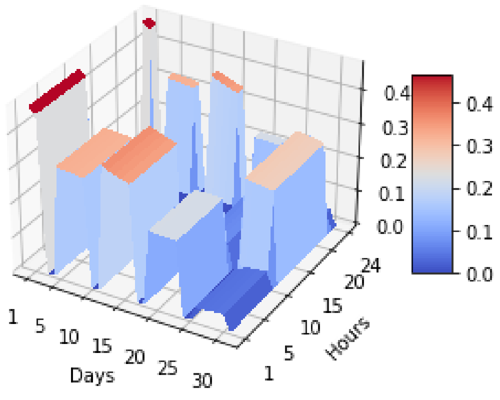
(a) IOHMM Gamma



(b) Margrabe Gamma 11



(c) Margrabe Gamma 22

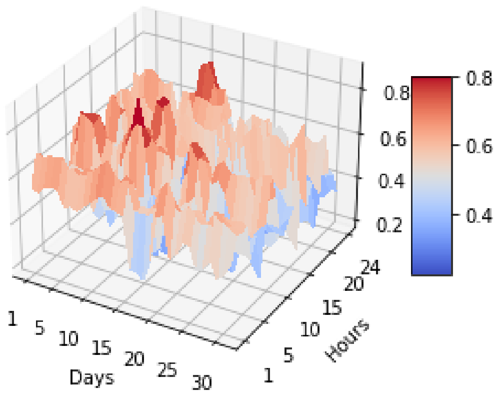


(d) Margrabe Gamma 12

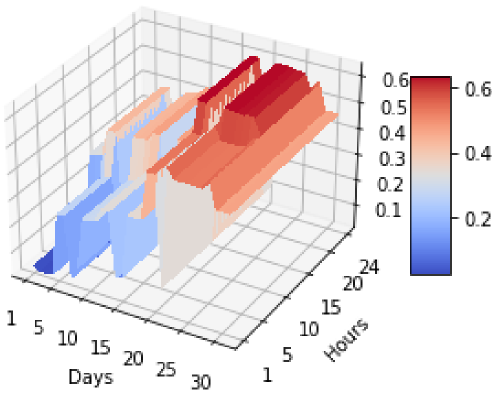
Figure D.10: Netherlands-Germany Gamma surface plots for July.

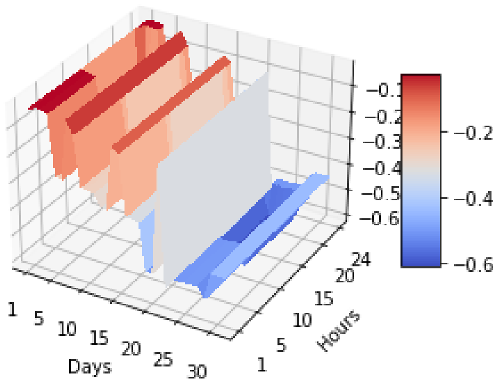
D.2. August

D.2.1. Germany-France



(a) IOHMM

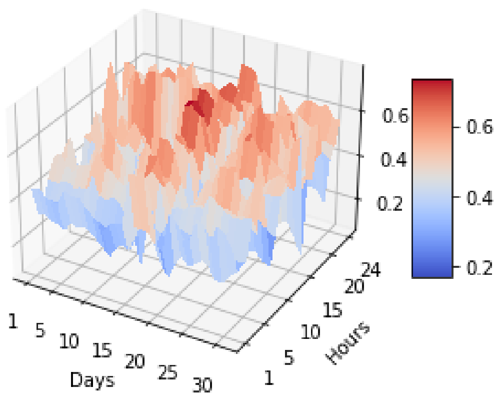




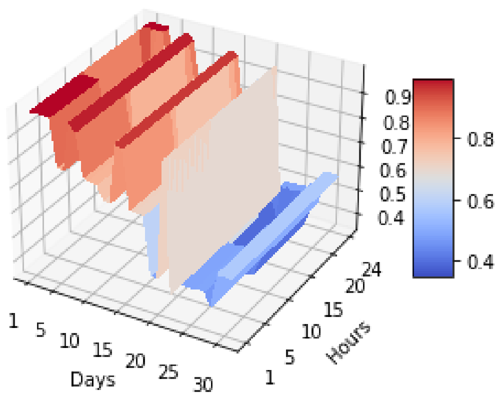
(b) Margrabe Delta 1

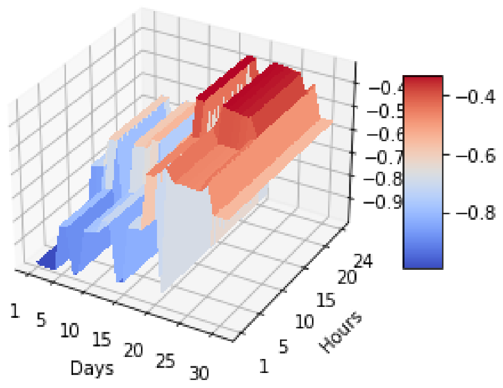
(c) Margrabe Delta 2

Figure D.11: Germany to France Delta surface plots for August.



(a) IOHMM

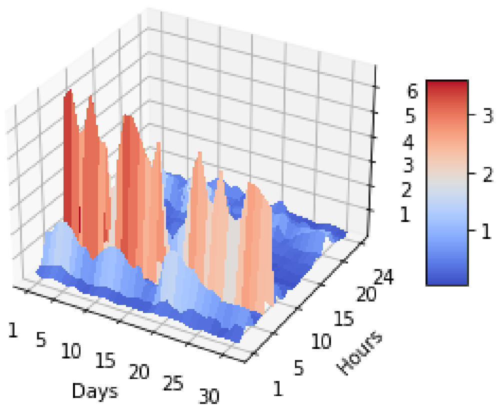




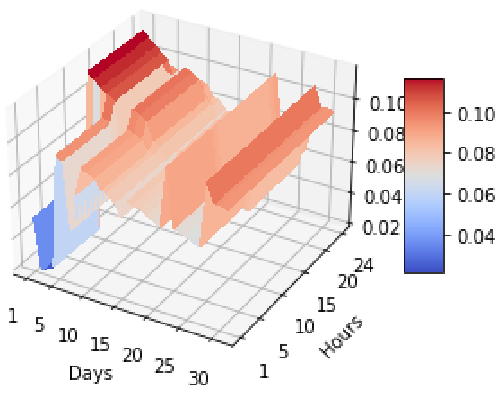
(b) Margrabe Delta 1

(c) Margrabe Delta 2

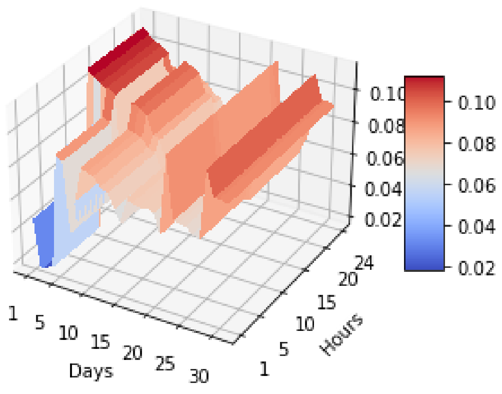
Figure D.12: France to Germany Delta surface plots for August.



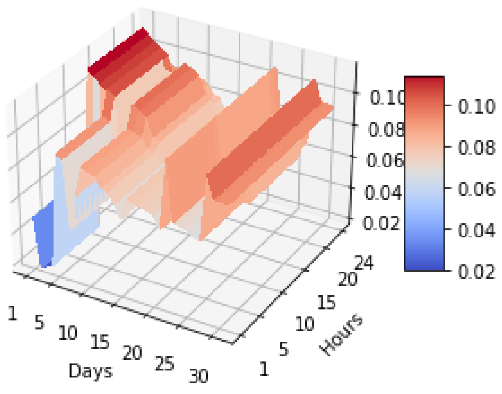
(a) IOHMM Gamma



(b) Margrabe Gamma 11



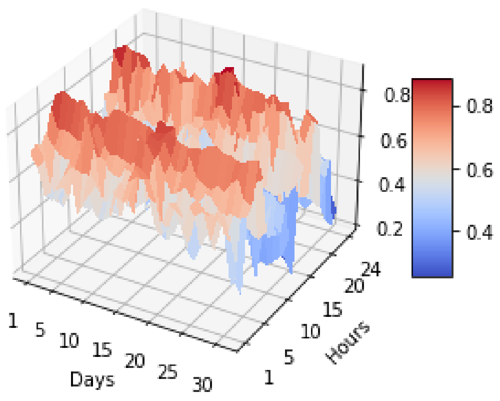
(c) Margrabe Gamma 22



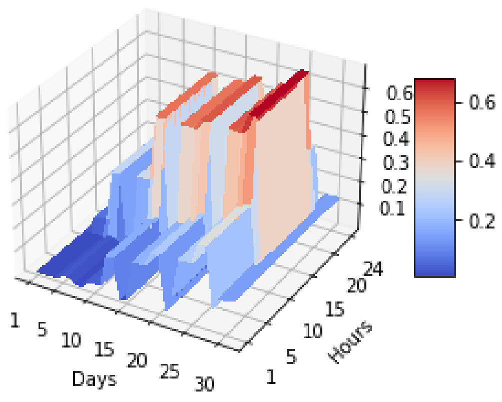
(d) Margrabe Gamma 12

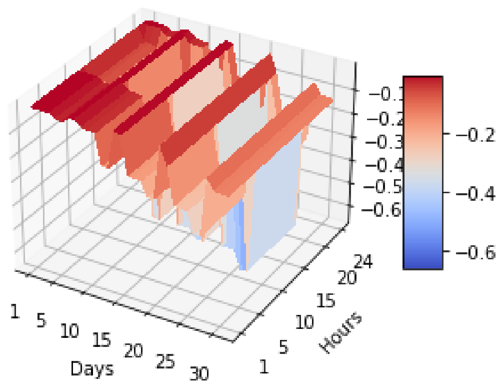
Figure D.13: Germany-France Gamma surface plots for August.

D.2.2. Netherlands-Germany



(a) IOHMM

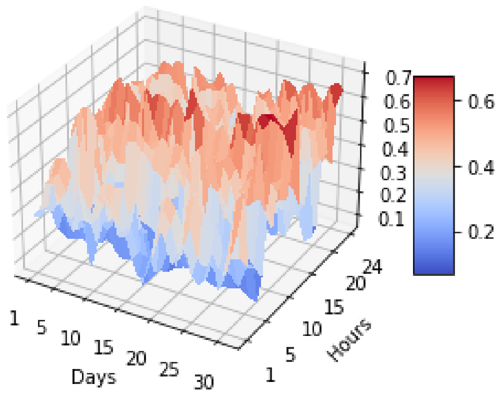




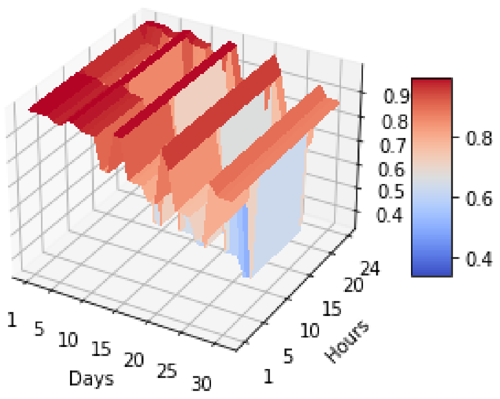
(b) Margrabe Delta 1

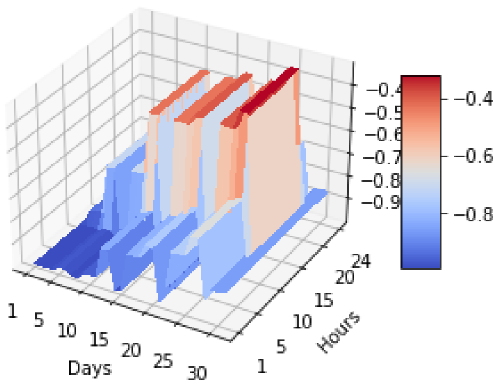
(c) Margrabe Delta 2

Figure D.14: Netherlands to Germany Delta surface plots for August.



(a) IOHMM

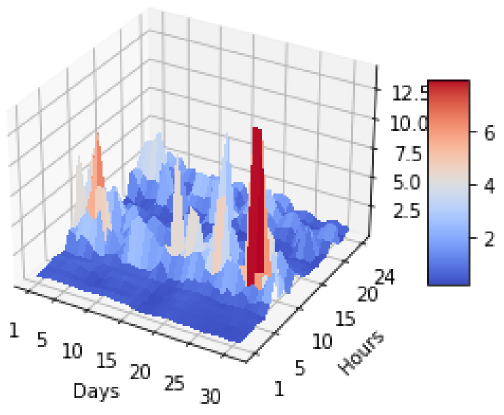




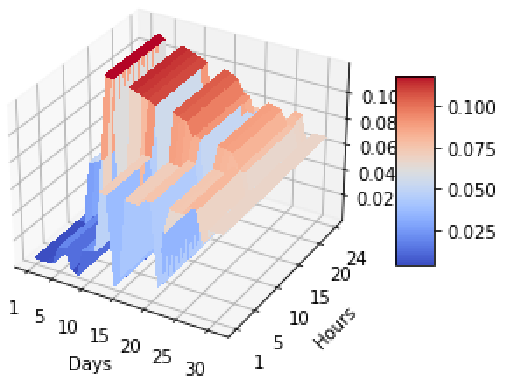
(b) Margrabe Delta 1

(c) Margrabe Delta 2

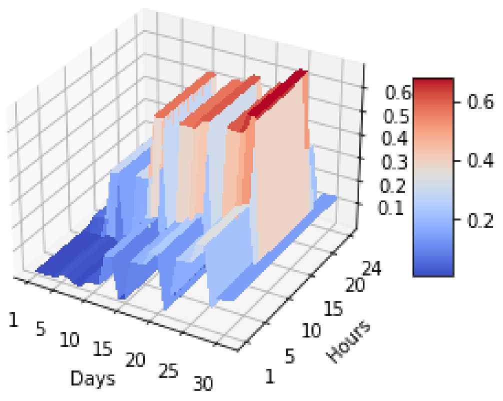
Figure D.15: Germany to Netherlands Delta surface plots for August.



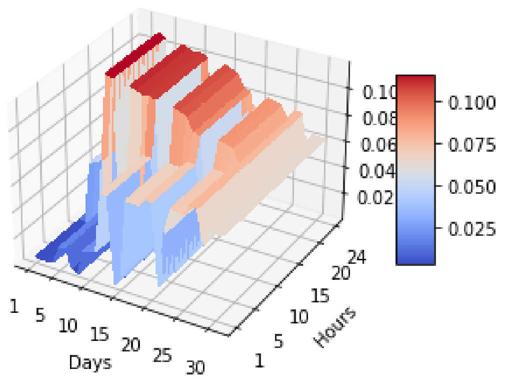
(a) IOHMM Gamma



(b) Margrabe Gamma 11



(c) Margrabe Gamma 22

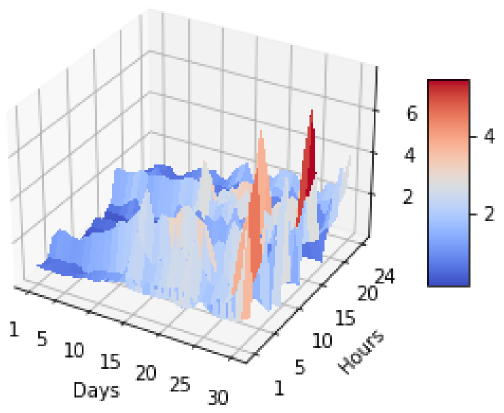


(d) Margrabe Gamma 12

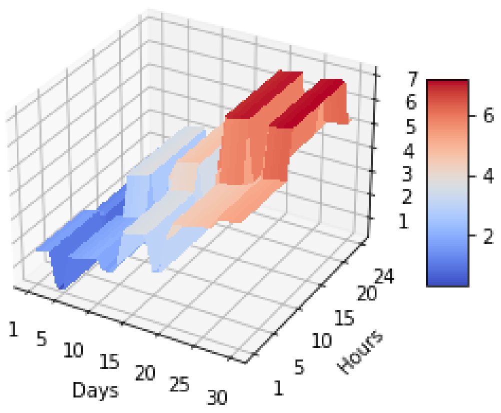
Figure D.16: Netherlands-Germany Gamma surface plots for August.

D.3. September

D.3.1. Germany-France

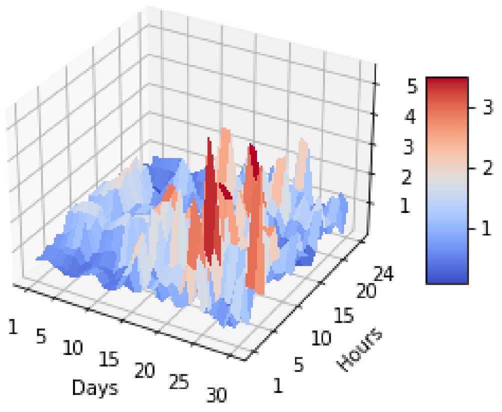


(a) IOHMM

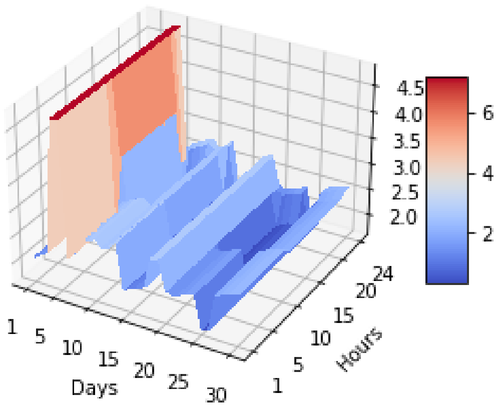


(b) Margrabe

Figure D.17: Germany to France spread option valuation surface plots for September.

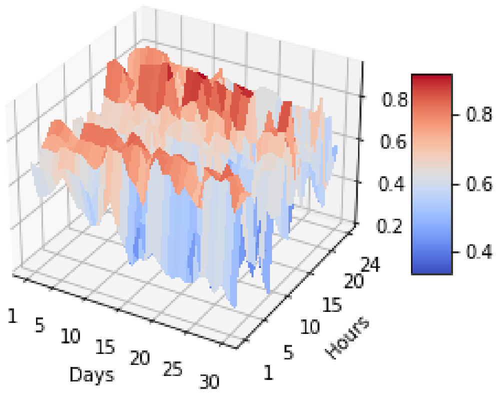


(a) IOHMM

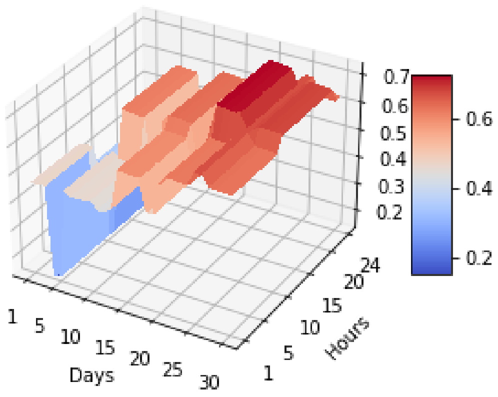


(b) Margrabe

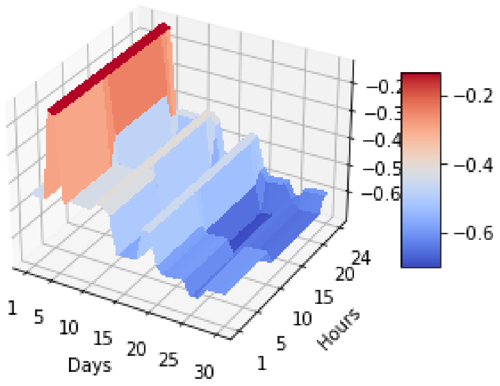
Figure D.18: France to Germany spread option valuation surface plots for September.



(a) IOHMM

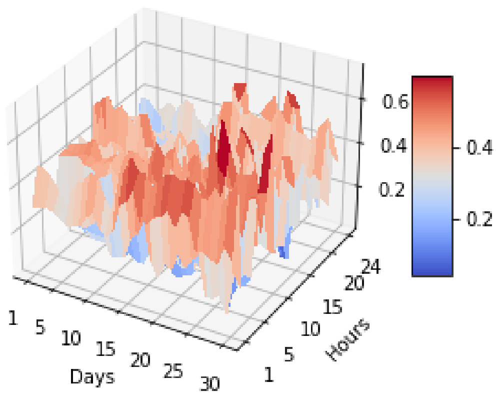


(b) Margrabe Delta 1

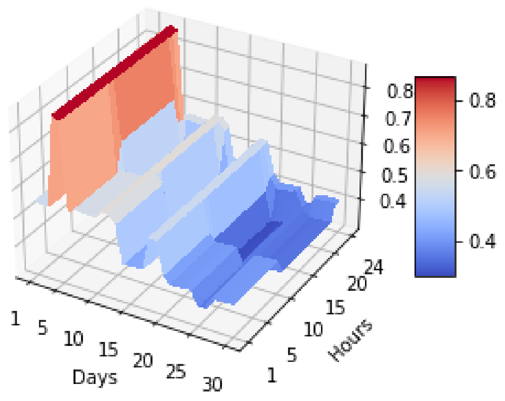


(c) Margrabe Delta 2

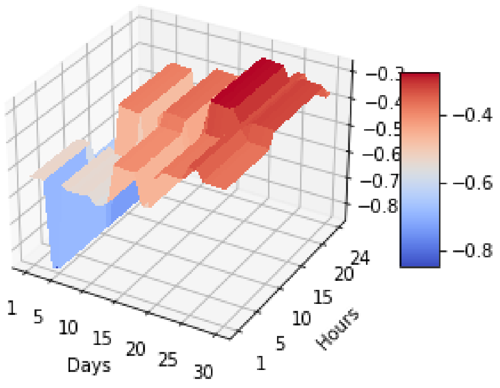
Figure D.19: Germany to France Delta surface plots for September.



(a) IOHMM

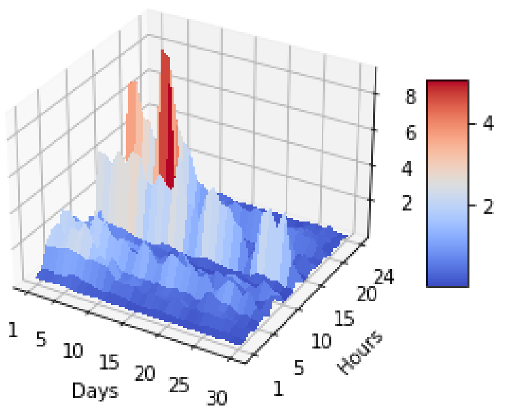


(b) Margrabe Delta 1

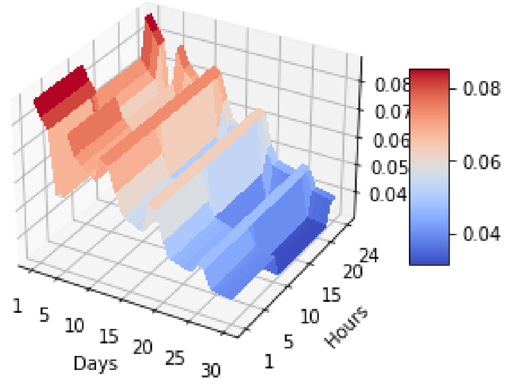


(c) Margrabe Delta 2

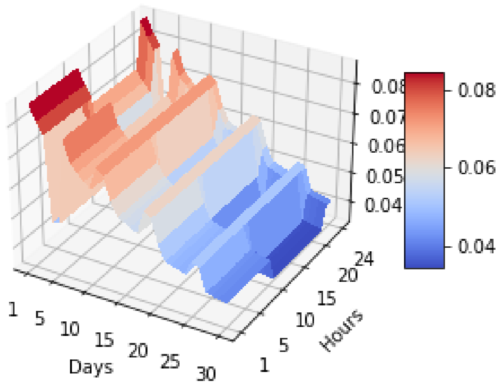
Figure D.20: France to Germany Delta surface plots for September.



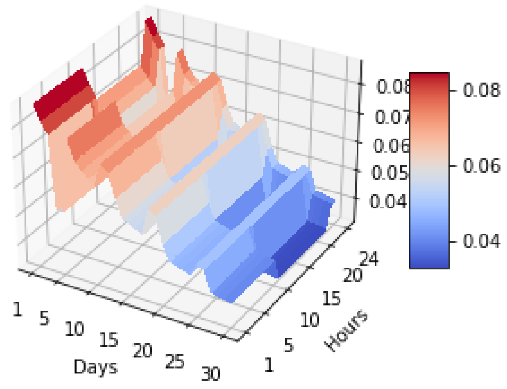
(a) IOHMM Gamma



(b) Margrabe Gamma 11



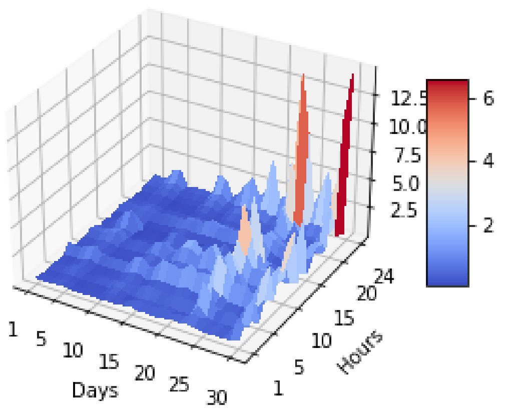
(c) Margrabe Gamma 22



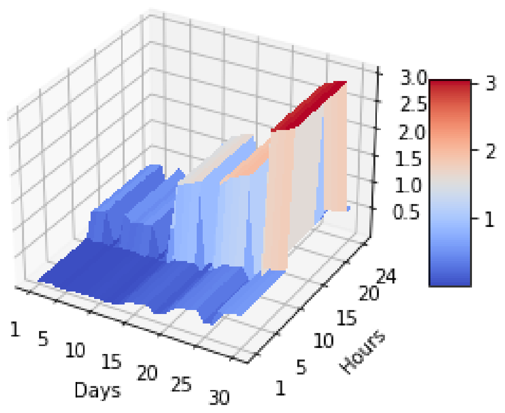
(d) Margrabe Gamma 12

Figure D.21: Germany-France Gamma surface plots for September.

D.3.2. Netherlands-Germany

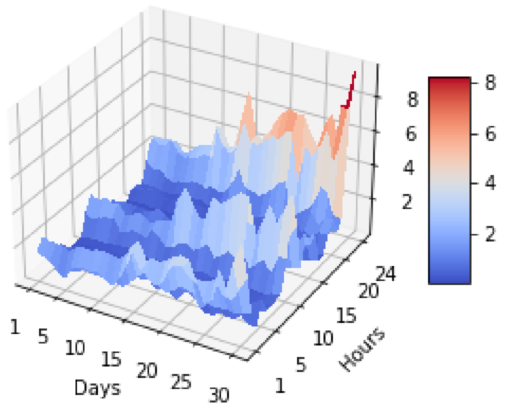


(a) IOHMM

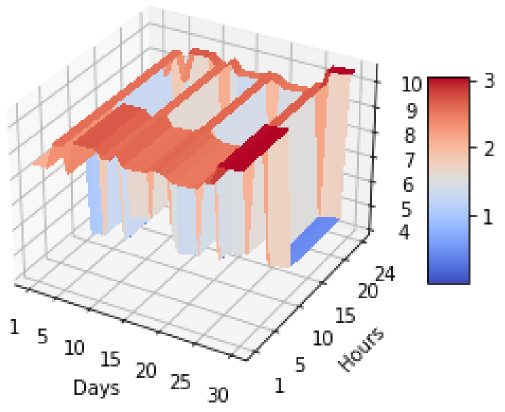


(b) Margrabe

Figure D.22: Netherlands to Germany spread option valuation surface plots for September.

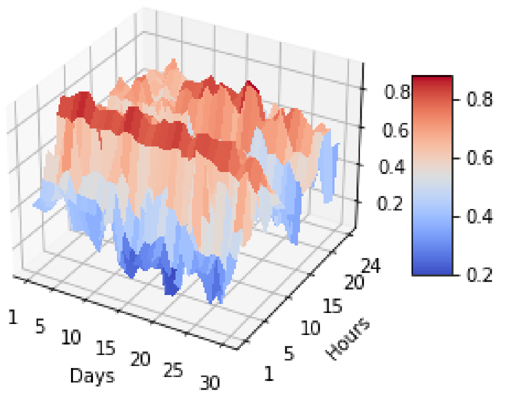


(a) IOHMM

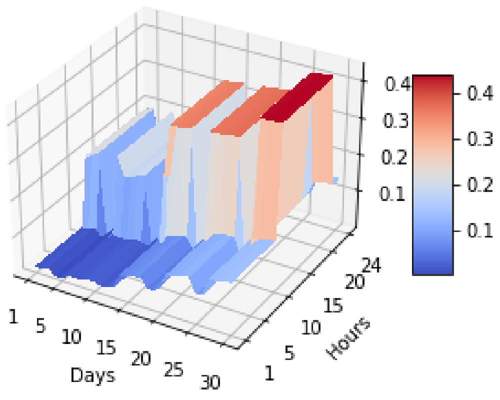


(b) Margrabe

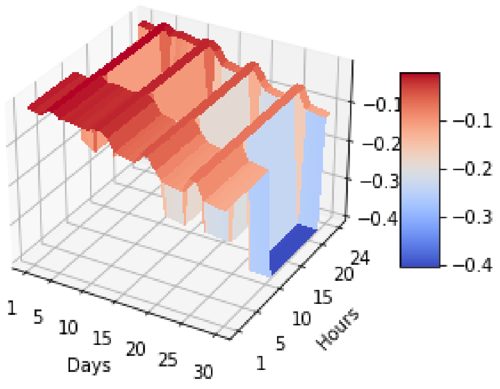
Figure D.23: Germany to Netherlands spread option valuation surface plots for September.



(a) IOHMM

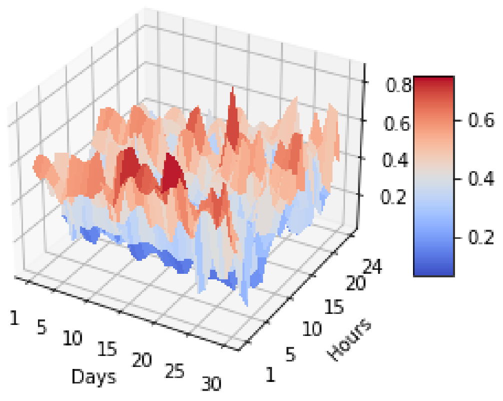


(b) Margrabe Delta 1

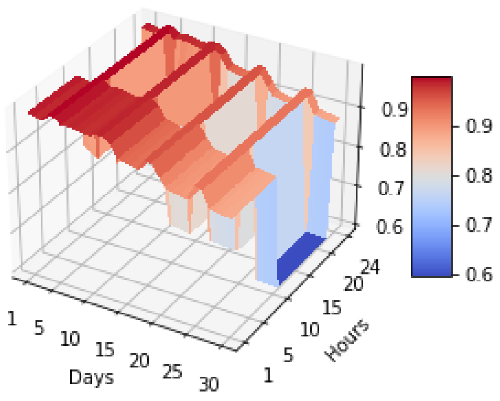


(c) Margrabe Delta 2

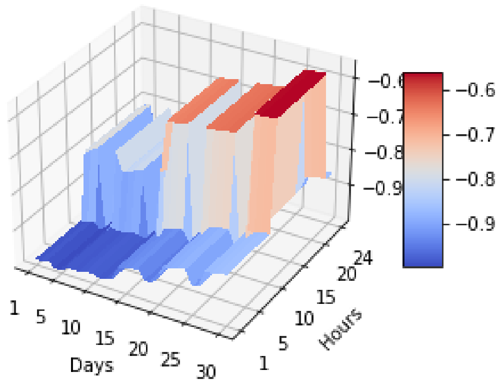
Figure D.24: Netherlands to Germany Delta surface plots for September.



(a) IOHMM

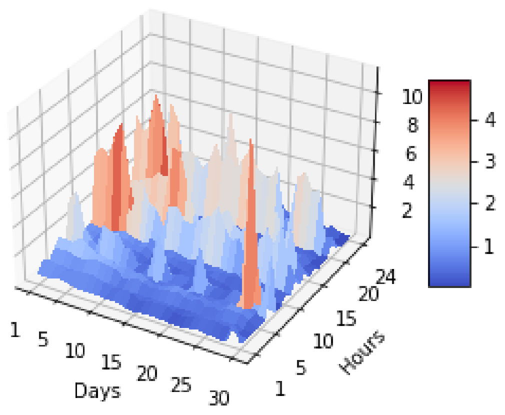


(b) Margrabe Delta 1

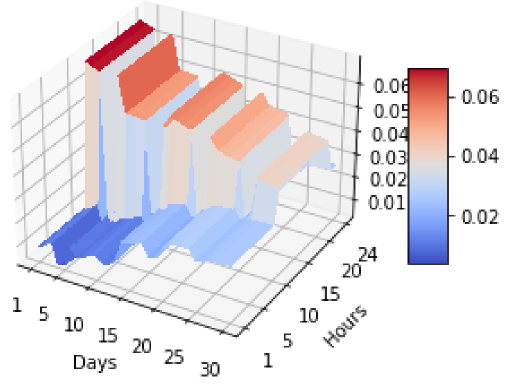


(c) Margrabe Delta 2

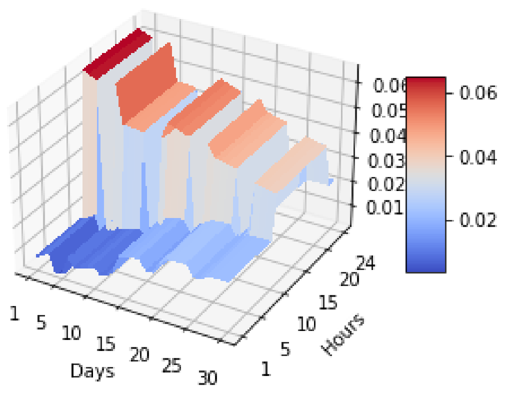
Figure D.25: Germany to Netherlands Delta surface plots for September.



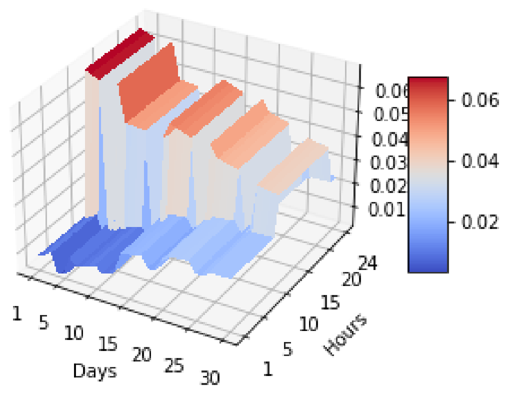
(a) IOHMM Gamma



(b) Margrabe Gamma 11



(c) Margrabe Gamma 22

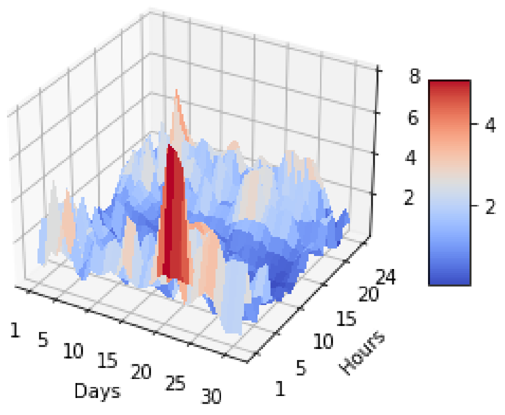


(d) Margrabe Gamma 12

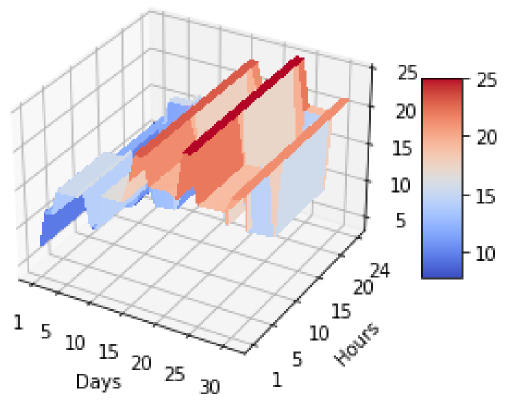
Figure D.26: Netherlands-Germany Gamma surface plots for September.

D.4. October

D.4.1. Germany-France

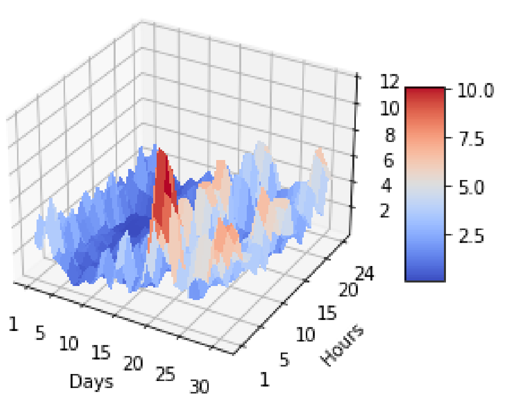


(a) IOHMM

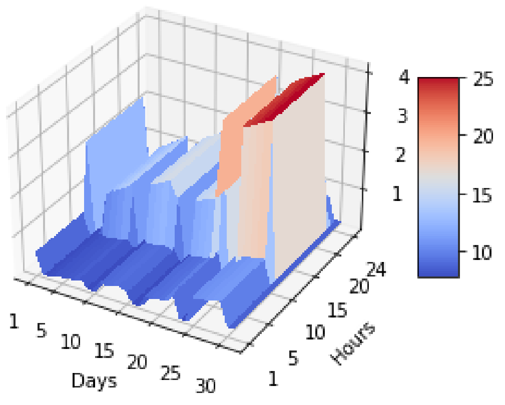


(b) Margrabe

Figure D.27: Germany to France spread option valuation surface plots for October.

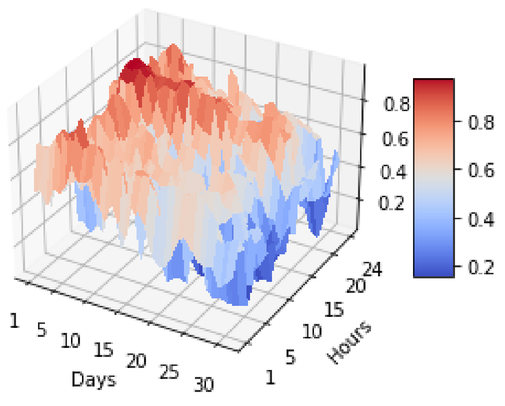


(a) IOHMM

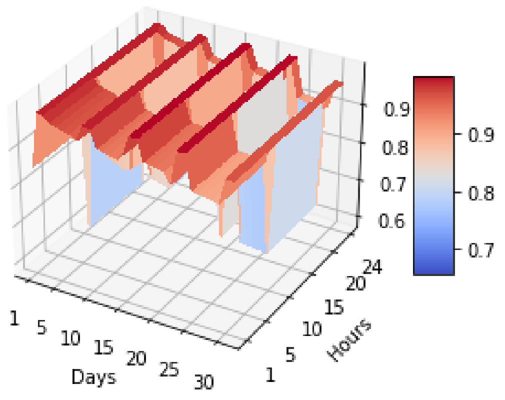


(b) Margrabe

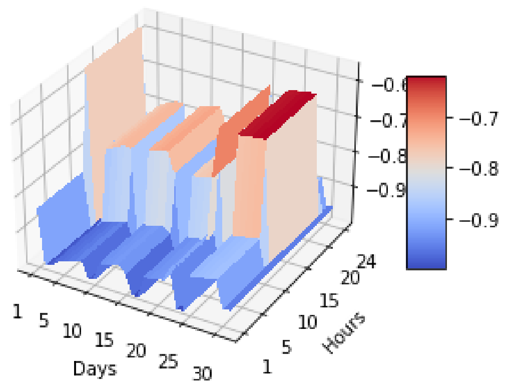
Figure D.28: France to Germany spread option valuation surface plots for October.



(a) IOHMM

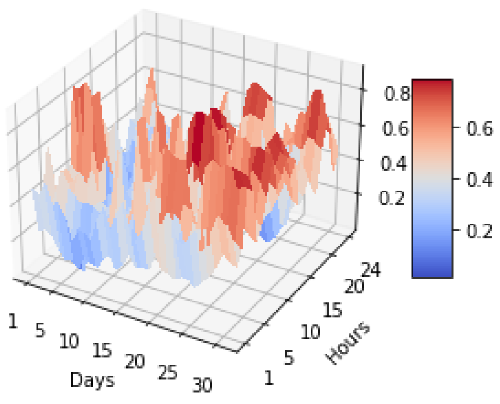


(b) Margrabe Delta 1

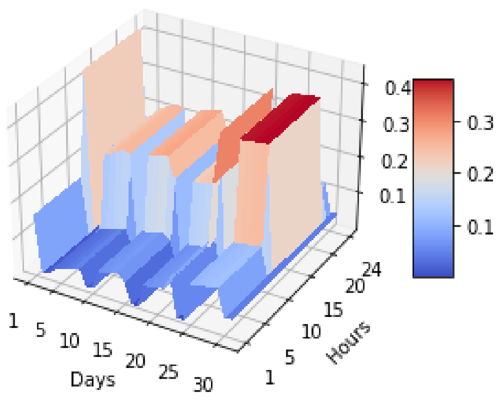


(c) Margrabe Delta 2

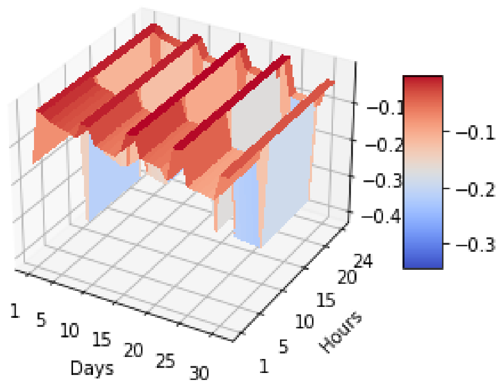
Figure D.29: Germany to France Delta surface plots for October.



(a) IOHMM

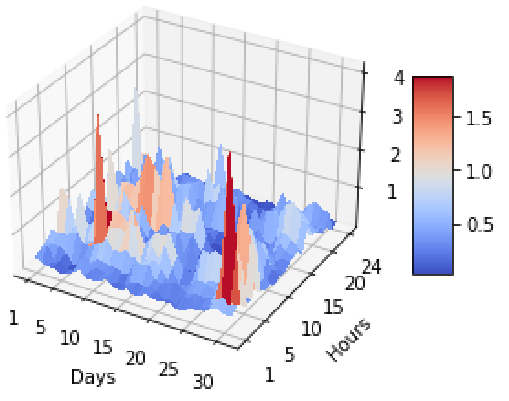


(b) Margrabe Delta 1

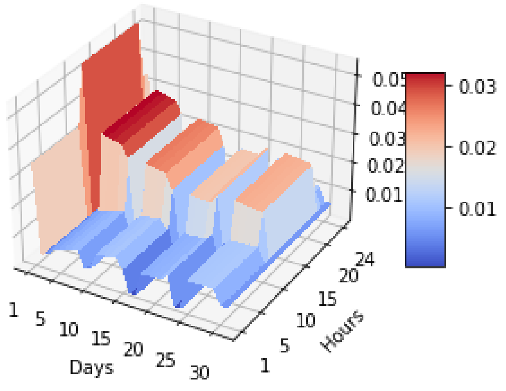


(c) Margrabe Delta 2

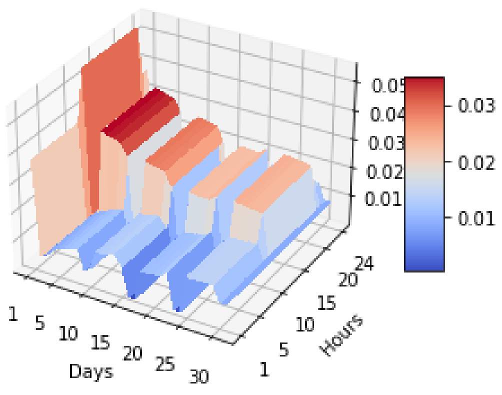
Figure D.30: France to Germany Delta surface plots for October.



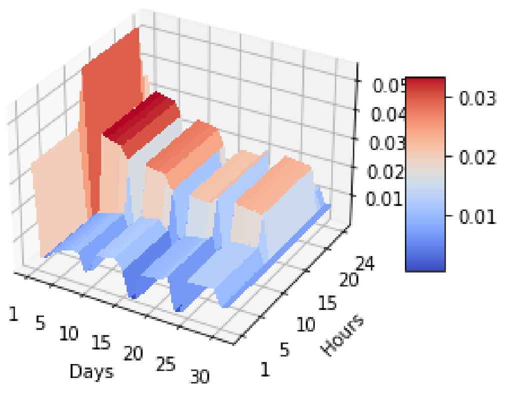
(a) IOHMM Gamma



(b) Margrabe Gamma 11



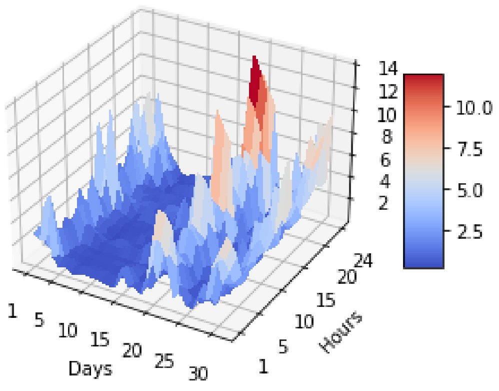
(c) Margrabe Gamma 22



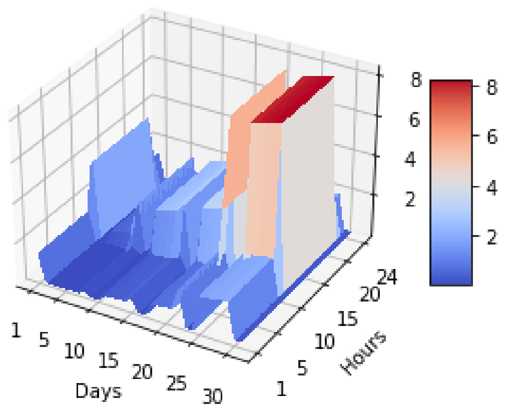
(d) Margrabe Gamma 12

Figure D.31: Germany-France Gamma surface plots for October.

D.4.2. Netherlands-Germany

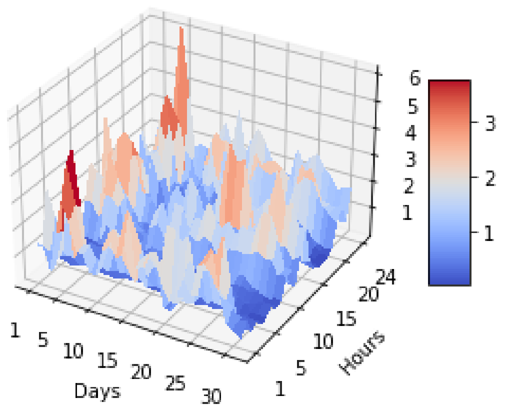


(a) IOHMM

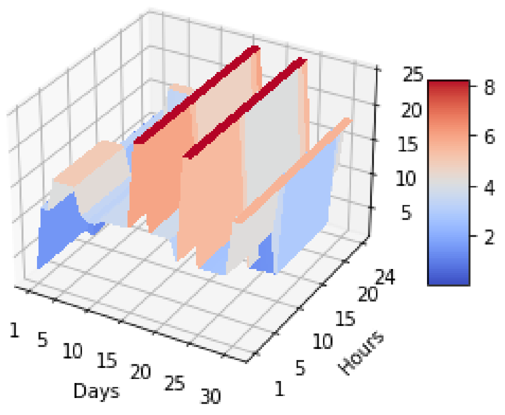


(b) Margrabe

Figure D.32: Netherlands to Germany spread option valuation surface plots for October.

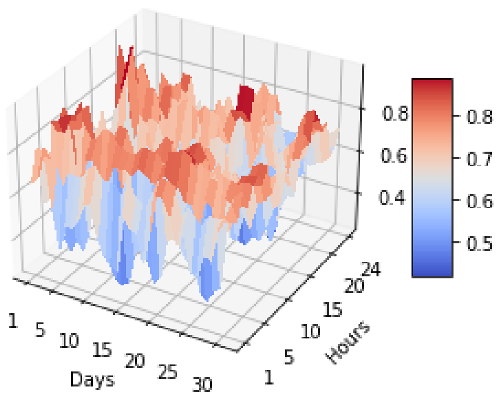


(a) IOHMM

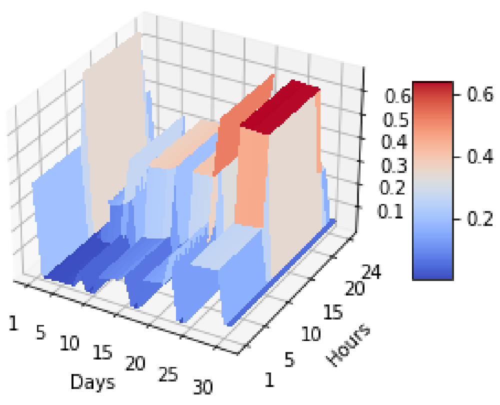


(b) Margrabe

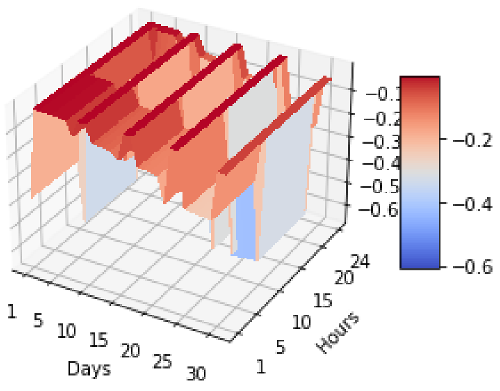
Figure D.33: Germany to Netherlands spread option valuation surface plots for October.



(a) IOHMM

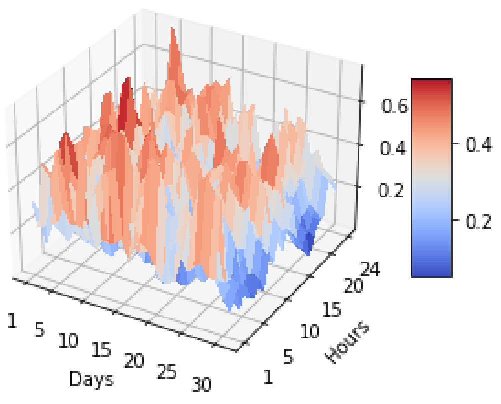


(b) Margrabe Delta 1

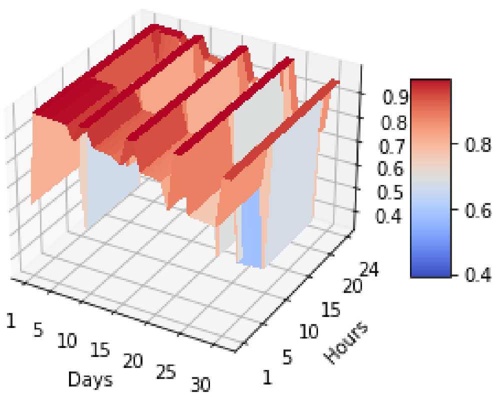


(c) Margrabe Delta 2

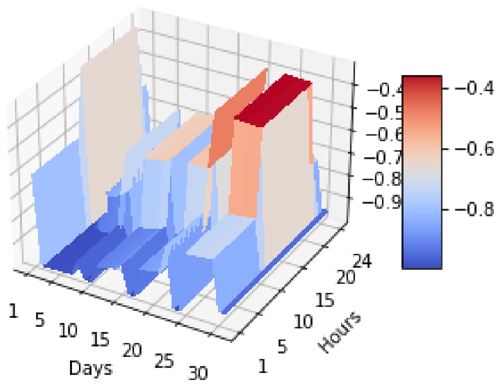
Figure D.34: Netherlands to Germany Delta surface plots for October.



(a) IOHMM

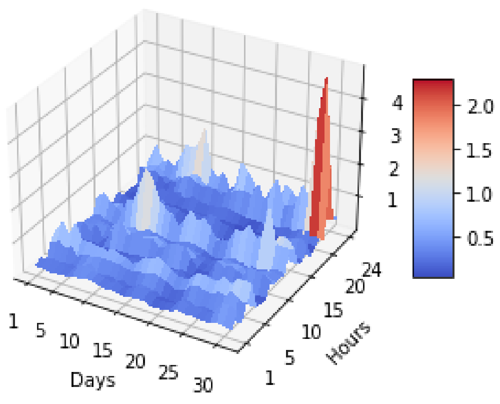


(b) Margrabe Delta 1

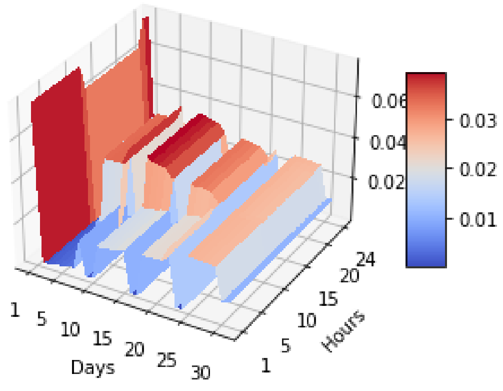


(c) Margrabe Delta 2

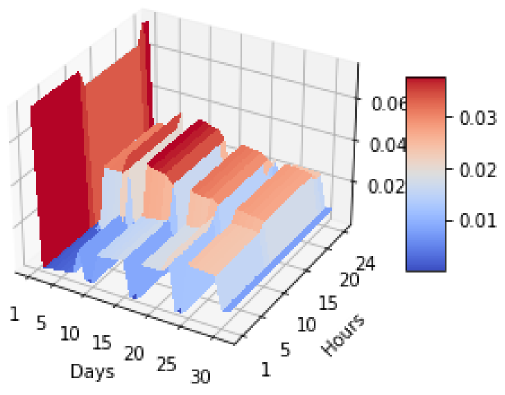
Figure D.35: Germany to Netherlands Delta surface plots for October.



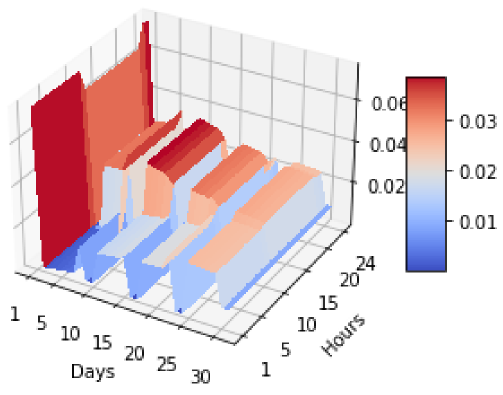
(a) IOHMM Gamma



(b) Margrabe Gamma 11



(c) Margrabe Gamma 22



(d) Margrabe Gamma 12

Figure D.36: Netherlands-Germany Gamma surface plots for October.

Bibliography

[1] Ole E Barndorff-Nielsen. Normal inverse gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of statistics*, 24(1):1–13, 1997.

[2] Yoshua Bengio and Paolo Frasconi. Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[3] Yoshua Bengio, V-P Lauzon, and Réjean Ducharme. Experiments on the application of iohmms to model financial returns series. *IEEE Transactions on Neural Networks*, 12(1):113–123, 2001.

[4] Yoshua Bengio et al. Markovian models for sequential data. *Neural computing surveys*, 2(199):129–162, 1999.

[5] Fred Espen Benth, Valery A Kholodnyi, and Peter Laurence. Quantitative energy finance. *Modelling, pricing, and hedging in energy and commodity markets. Springer*, 2014.

[6] R Carmona and V Durrleman. Pricing and hedging spread options in a log-normal model (technical report: Department of operations research and financial engineering). *Princeton, NJ: Princeton University*, 2003.

[7] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

[8] François Chollet et al. Keras. `https://keras.io`, 2015.

[9] Les Clewlow and Chris Strickland. *Energy derivatives: pricing and risk management.* Lacima Publ., 2000.

[10] Nemo Committee et al. Euphemia public description: Single price coupling algorithm. *URL: https://www. epexspot. com/sites/default/files/2020-02/Euphemia_Public% 20Description_Single% 20Price% 20Coupling% 20Algorithm_190410. pdf*, 2019.

[11] Aswath Damodaran. The promise and peril of real options. 2005.

[12] Shijie Deng. *Stochastic models of energy commodity prices and their applications: Mean-reversion with jumps and spikes.* University of California Energy Institute Berkeley, 2000.

[13] Ernst Eberlein and Gerhard Stahl. Both sides of the fence: a statistical and regulatory view of electricity risk. *Energy and Power Risk Management*, 8(6):32–36, 2003.

[14] Ernst Eberlein, OE Barndorff-Nielsen, T Mikosch, and S Resnick. Lévy processes: theory and applications, 2001.

[15] Student Energy. Electrical grid, January 2020. URL https://studentenergy.org/site/assets/uploads/2020/01/Electric-Grid.jpg. [Online; accessed 1 May, 2022].

[16] Seyyed Ruhollah Etesami. Spread options: From margrabe to kirk. *Available at SSRN 3665654*, 2020.

[17] Alexander Eydeland and Krzysztof Wolyniec. *Energy and power risk management: New developments in modeling, pricing, and hedging*, volume 97. John Wiley & Sons, 2002.

[18] European Union Agency for the Cooperation of Energy Regulators. Harmonised allocation rules for long-term transmission rights, Nov 2021. URL https://www.jao.eu/sites/default/files/2021-12/EU%20HAR%202022%20with%20annexes.pdf.

[19] Hélyette Geman and Andrea Roncoroni. Understanding the Fine Structure of Electricity Prices. *The Journal of Business*, 79(3):1225–1262, May 2006. doi: 10.1086/500675. URL https://ideas.repec.org/a/ucp/jnlbus/v79y2006i3p1225-1262.html.

[20] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[21] Alicia Mateo González, AM Son Roque, and Javier García-González. Modeling and forecasting electricity prices with input/output hidden markov models. *IEEE Transactions on Power Systems*, 20(1):13–24, 2005.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[23] Pedro Mejía Gómez. Chapter 10 - benefits of market coupling in terms of social welfare. In Alessandro Rubino, Maria Teresa Costa Campi, Veronica Lenzi, and Ilhan Ozturk, editors, *Regulation and Investments in Energy Markets*, pages 185–198. Academic Press, 2016. ISBN 978-0-12-804436-0. doi: https://doi.org/10.1016/B978-0-12-804436-0.00010-2. URL https://www.sciencedirect.com/science/article/pii/B9780128044360000102.

[24] Jeong-Hoon Kim and Chang-Rae Park. A multiscale extension of the margrabe formula under stochastic volatility. *Chaos, Solitons & Fractals*, 97:59–65, 2017.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Minqiang Li, Shi-Jie Deng, and Jieyun Zhoc. Closed-form approximations for spread option prices and greeks. *The Journal of Derivatives*, 15(3):58–80, 2008.

[27] Giorgio Locatelli, Mauro Mancini, and Giovanni Lotti. A simple-to-implement real options method for the energy sector. *Energy*, 197:117226, 2020.

[28] Yijuan Lu, Ira Cohen, Xiang Sean Zhou, and Qi Tian. Feature selection using principal feature analysis. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 301–304, 2007.

[29] William Margrabe. The value of an option to exchange one asset for another. *The journal of finance*, 33(1):177–186, 1978.

[30] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[31] Joint Allocation Office. Auctions, 2022. URL `https://www.jao.eu/auctions#/`.

[32] Joint Allocation Office. List of bidding zone borders, March 2022. URL `https://www.jao.eu/sites/default/files/2022-03/List%20of%20Bidding%20Zone%20borders_2022_CORE.pdf`.

[33] PCR Project OMIE. Da markets in the eu, 2016. URL `https://ars.els-cdn.com/content/image/3-s2.0-B9780128044360000102-f10-02-9780128044360.jpg`. [Online; accessed March 15, 2022].

[34] Cornelis W Oosterlee and Lech A Grzelak. *Mathematical Modeling and Computation in Finance: With Exercises and Python and Matlab Computer Codes*. World Scientific, 2019.

[35] Puneet Pasricha and Anubha Goel. Pricing power exchange options with hawkes jump diffusion processes. *Journal of Industrial & Management Optimization*, 17(1):133, 2021.

[36] Rolf Poulsen. The margrabe formula. *Encyclopedia of Quantitative Finance*, pages 1118–1120, 2009.

[37] Christian Redl, Reinhard Haas, Claus Huber, and Bernhard Böhm. Price formation in electricity forward markets and the relevance of systematic forecast errors. *Energy Economics*, 31(3):356–364, 2009. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2008.12.001. URL `https://www.sciencedirect.com/science/article/pii/S0140988308001862`.

[38] David Schönheit, Michiel Kenis, Lisa Lorenz, Dominik Möst, Erik Delarue, and Kenneth Bruninx. Toward a fundamental understanding of flow-based market coupling for cross-border electricity trading. *Advances in Applied Energy*, 2:100027, 2021. ISSN 2666-7924. doi: https://doi.org/10.1016/j.adapen.2021.100027. URL `https://www.sciencedirect.com/science/article/pii/S2666792421000202`.

[39] DP van de Wiel BSc. Valuation of insurance products using a normal inverse gaussian distribution. Master's thesis, Tilburg University, 2015.

[40] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0:

Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272, 2020. doi: 10.1038/s41592-019-0686-2.