ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Dottorato di ricerca in Matematica

Ciclo XXXV

Settore concorsuale di afferenza: 01/A3

Settore scientifico disciplinare: MAT/06

Portfolio optimization in the energy market

Tesi di dottorato presentata da: Elisa Raspanti

Coordinatrice dottorato:
Chiar.ma Prof.ssa
Valeria Simoncini

Relatore:
Chiar.mo Prof.
Andrea Pascucci

Esame finale anno 2023

Abstract

Let's put ourselves in the shoes of an energy company. Our fleet of electricity production plants mainly includes gas, hydroelectric and waste-to-energy plants. We also sold contracts for the supply of gas and electricity. For each year we have to plan the trading of the volumes needed by the plants and customers: better to fix the price of these volumes in advance with the so-called forward contracts, instead of waiting for the delivery months, exposing ourselves to price uncertainty. Here's the thing: trying to keep uncertainty under control in a market that has never shown such extreme scenarios as in recent years: a pandemic, a worsening climate crisis and a war that is affecting economies around the world have made the energy market more volatile than ever. How to make decisions in such uncertain contexts?

There is an optimization problem: given a year, we need to choose the optimal planning of volume trading times, to meet the needs of our portfolio at the best prices, taking into account the liquidity constraints given by the market and the risk constraints imposed by the company. Algorithms are needed for the generation of market scenarios over a finite time horizon, that is, a probabilistic distribution that allows a view of all the dates between now and the end of the year of interest. Algorithms are needed to solve the optimization problem: we have proposed more than one and compared them; a very simple one, which avoids considering part of the complexity, moving on to a scenario approach and finally a reinforcement learning approach.

Acknowledgements

Questa tesi ha visto diversi attori in scena.

Il capitolo Reinforcement Learning riporta un lavoro di collaborazione che ha coinvolto l'ufficio "Modeling and Pricing" e l'Università di Pavia, in particolare il docente Marco Piastra e la docente Andrea Pedrini. Il capitolo Scenario Optimization è nato dall'evolvere di un progetto di approfondimento al termine di un corso tenuto al Politecnico di Milano dalla professoressa Maria Prandini. Il capitolo Price Simulations è stato portato avanti all'interno dell'ufficio "Modeling

and Pricing", in particolare con il contributo fondamentale di Andrea Cafforio. Il capitolo Introduction racconta un progetto inter-ufficio: "Modeling and Pricing", "Market Analysis and Price Forecasting", "Commodity Optimization", "Energy Risk Control".

Sono stata comunque dottoranda all'Università di Bologna e ringrazio innanzitutto il professore Andrea Pascucci, il mio relatore e la mia guida accademica in questo percorso; il professore Andrea Cosso che mi ha proposto come correlatrice per la laureanda Teresa Angeli; un saluto ai dottorandi e ai professori esterni e interni dei corsi e dei seminari che ho seguito: grazie per la formazione di alta qualità, interessante spesso al di là del problema oggetto della tesi, e per l'ambiente stimolante perché ricco di persone appassionate e talentuose.

Al netto dell'alternarsi tra lavoro in presenza e da casa, sono sempre stata un'impiegata in A2A: ringrazio il referente aziendale per questo progetto, nonché responsabile dell'ufficio "Modeling and Pricing", Andrea Marziali; tutti i miei colleghi, in particolare l'ufficio "Modeling and Pricing", Alessandra Trolli, Fiorella Merola, Andrea Cafforio, Elisa Cambiaso, Andrea Marziali per la disponibilità nell'assecondare i miei impegni da dottoranda; i già citati uffici "Energy Risk Control", "Commodity Optimization", "Market Analysis and Price Forecasing"; le colleghe di alcuni uffici dell'area "People and Trasformation" a cui mi sono rivolta per tutte le questioni burocratiche. Un grazie anche al dirigente dell'area "Market Analysis, Modeling and Pricing", Alberto Ravasi e alla dirigente di tutto il "Portfolio Management and Trading", Annamaria Arcudi: assieme ad Andrea Marziali sono stati gli sponsor interni di questo progetto di collaborazione con l'università.

Infine un grazie al supporto da casa, umano e canino.

Contents

Glossary

CCGT Combined cycle gas turbine plant. A plant that produces electricity through gas; on the one hand it sells electricity, on the other it pays gas and EUA and some fixed costs according to a given formula, like $\gamma_{gas} \cdot Price_{gas} + \gamma_{eua} \cdot Price_{eua} + \gamma_{fixed}$, where $\gamma_{gas}, \gamma_{eua}, \gamma_{fixed}$ are given coefficients.. 6

EUA European Emission Allowances. It represents the price you have to pay in order to emit one tonne of carbon dioxide equivalent (CO_2) during a specified period (https://www.emissions-euets.com/).. 4, 6, 31, 33, 42, 91

MWh MegaWatt-hour. A unit for measuring power that is equivalent to one million watts used for one hour.. 10

PaR Profit at risk. A risk measure commonly used in the energy sector.. iii, 7, 16, 17, 22–24, 26, 67, 69, 73

PFOR An Italian acronym that stands for Prezzo di FORnitura. A gas index that depends on TTF and it indicates the coverage of natural gas procurement costs; it is published by ARERA (Autorità di Regolazione per Energia Reti e Ambiente - Italian acronym that stands for Authority for Energy, Networks and the Environment) before the start of each quarter of the year. Starting from October 2022, this price no longer exists as ARERA has changed the methods of calculating the gas price for the protected market.. 3, 6, 7, 91

PL Profit and Loss.. 4, 5, 17, 23, 72, 91, 92

PSV An Italian acronym that stands for Punto di Scambio Virtuale. Price of the Italian gas.. 3, 6, 7, 30, 31, 33, 35–37, 42, 74, 75, 91

PUN An Italian acronym that stands for Prezzo Unico Nazionale. Price of the Italian power.. 6, 10, 32, 48

PUNBASE PUN is an Italian acronym that stands for Prezzo Unico Nazionale (i.e. price of the Italian power); BASE refers to the fact that it is the average price over all hours of each day of the given period.. 3, 6, 7, 30–32, 34, 36, 42, 44, 91

PUNPEAK PUN is an Italian acronym that stands for Prezzo Unico Nazionale (i.e. price of the Italian power); PEAK refers to the fact that it is the average price over the ”peak” hours of each day of the given period, i.e. from 8 a.m. to 8 p.m of the working days.. 3, 6, 7, 30–32, 36, 42, 44, 46, 74, 75, 91

Q Q stands for Quarter; for example Q4 is the forth quarter of the year.. 10, 11, 13, 67, 68

TTF An acronym that stands for Title Transfer Facility. Price of the Dutch gas.. 3, 6, 7, 30, 31, 33–35, 42, 91

Chapter 1

Introduction

In this chapter we describe the main ingredients of the real world problem and the easiest approach.

1.1 Our portfolio optimization problem

Starting from the 1990s, the liberalization of the gas and electricity markets began around the world, in particular in Europe, in particular in Italy (see [1]). In addition to gas and electricity, other relevant products in the energy market are *oil*, one of the most traded commodities, by far, not only in the energy sector; *coal*, since in the world a large percentage of electricity production is given by coal; CO_2, in the sense that when the production of electricity involves CO_2 emissions, you have to pay a certain amount to produce.
Our fleet of electricity production plants mainly includes CCGTs, hydroelectric plants and waste-to-energy plants; the subdivision of our electricity production into technologies reflects the national one, while it differs from that of other European countries (see Figure 1.1 and 1.2). We also have customers with whom we enter into contracts for the supply of gas and electricity.

Given a year of interest not yet concluded (typically the current one or the next one), the production of the plants and the contracts with the customers create the need to purchase fuel for the plants, where necessary, and to sell the energy pro-

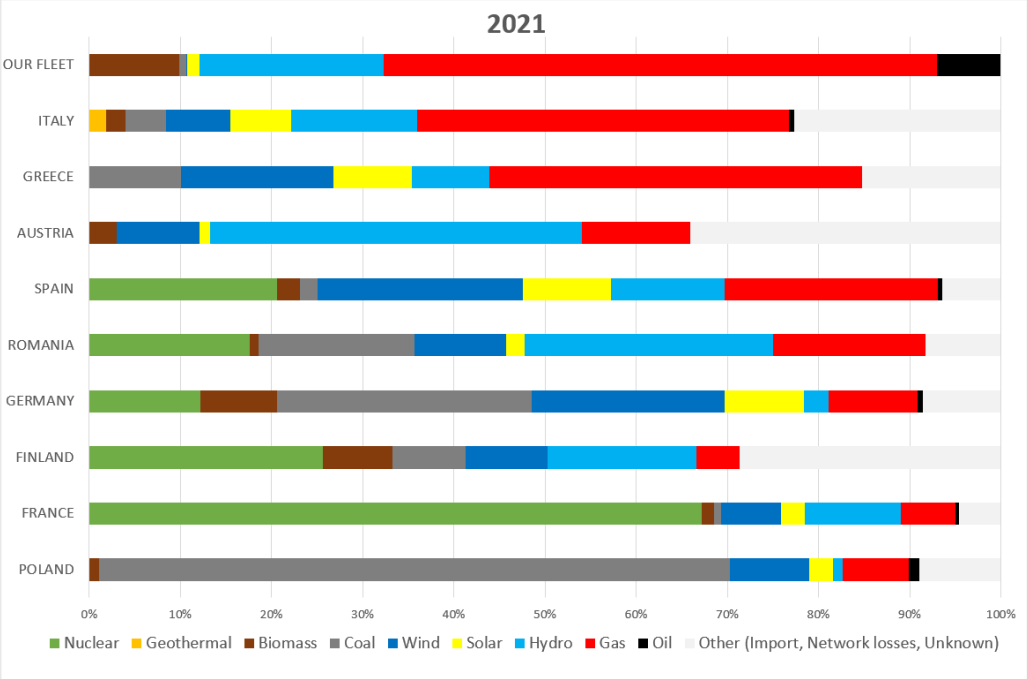


Figure 1.1: The figure shows the subdivision of our electricity production into technologies compared to that of Italy and that of other European countries by aggregating the data for 2021 ([2]).

duced during the months of the year, to supply gas and electricity to its customers during the months of the year of interest. Waiting to arrive at the month of delivery exposes you to the risk of the spot price, in the sense that we cannot know for sure what price will form in the future, so we usually prefer to keep that risk under control through forward contracts that fix today the price of something which will be used in a certain future month. The performance of the portfolio is evaluated with respect to a level called ”budget level” which corresponds to the value of all the volumes to be exchanged for that year of interest multiplied by the known and fixed prices (the budget price curve): on the one hand we try to keep the risk under control by closing open positions, fixing prices with forward contracts, on the other hand we try to find the best prices compared to those of the budget curve.

So we have a portfolio of forward exposures relating to power plants and cus-

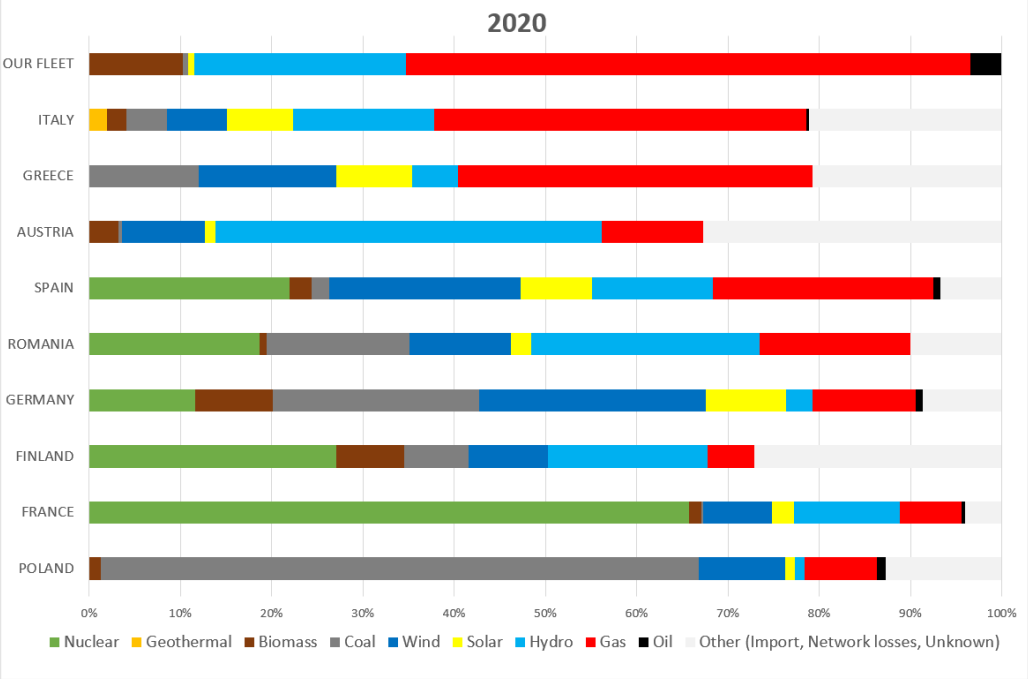


Figure 1.2: The figure shows the subdivision of our electricity production into technologies compared to that of Italy and that of other European countries by aggregating the data for 2020 ([2]).

tomer supplies: we have to decide how to divide the prescribed volumes up to each monthly delivery using the tradable products of each trading date t (one trading date every month), taking into account of liquidity and the risk constraints. The goal is choosing the best way to trade the prescribed volumes up to the given deliveries in order to optimize total profit, taking into account of all the constraints.

Given a certain time horizon of interest spanning T years and L underlyings, $x_1, ...x_L$, we consider M delivery dates, $d_1, ...d_M$, each one associated to a month of time horizon of interest. As we focus on a portfolio of forward exposures related to the production and supply of energy for the Italian market, the underlyings of interest are mainly Italian electricity prices (PUNBASE, PUNPEAK) and Italian gas prices (PSV, PFOR); in addition there is also the Dutch gas price (TTF), given that there are contracts that depend on this price which is a reference price for the European gas market; finally there is also the price that a plant has to pay to emit

CO_2 (EUA).

For each $i \in \{1, ...L\}$, for each $j \in \{1, ...M\}$ we have a certain volume exposure $\overline{V}_{ij}$, i.e. a quantity of the commodity x_i that we have to buy or to sell within the delivery date d_j. Decisions on the amount of volume exposure to trade and how to trade it are taken at the trading dates, t_k, $k = 1, 2, \ldots, K$; in general, the set of delivery dates is a subset of the set of trading dates, which means that a trading date can be earlier than all delivery dates.

The goal is to decide the amounts $V_{ij}(t_k)$, $k = 1, \ldots, K$, of the total volume exposure $\overline{V}_{ij}$ to trade from the current trading month up to the delivery date for every underlying x_i, $i = 1, \ldots, L$, and every delivery date d_j, $j = 1, \ldots, M$, so as to maximize the profit over the reference time horizon.

In particular, at the trading date t_k, for each underlying x_i and delivery date $d_j > t_k$, we have the following options for the (residual) volume

$$V_{ij}^k = \overline{V}_{ij} - \sum_{h:t_h<t_k} V_{ij}(t_h) \tag{1.1}$$

- trade all the volume at the current forward price $F_{ij}(t_k)$;
- trade the volume at the delivery month at the spot price $S_{ij} = F_{ij}(d_j)$;
- trade the volume at some intermediate month between the current and the delivery month.

If $t_k = d_j$, then, the delivery date d_j is met and we have to close the position, i.e., satisfy constraint

$$\sum_{k:t_k \leq d_j} V_{ij}(t_k) = \overline{V}_{ij} \tag{1.2}$$

by trading all the residual volumes V_{ij}^k at the spot price S_{ij}.

The profit P is computed with respect to a baseline value associated to a policy where the entire position is closed at a known *budget price level* that is denoted with B_{ij} for the underlying x_i with delivery date d_j; this means that we evaluate the performance of the portfolio by comparing the PL of the chosen actions with respect to the PL that we would have had by closing the entire position at the budget

price level: the goal is to outperform the PL at the budget price level, which we can calculate at any time since the budget price level is provided as an input.
More precisely, P is given by

$$P = P_{fwd} + P_{spot} \tag{1.3}$$

where the forward and the spot components are obtained as follows

$$P_{fwd} = \sum_{i=1}^{L} \sum_{k=1}^{K} P_{fwd,i}(t_k)$$
$$P_{spot} = \sum_{i=1}^{L} \sum_{k=1}^{K} P_{spot,i}(t_k)$$

with

$$P_{fwd,i}(t_k) = \sum_{j:t_k<d_j \wedge \overline{V}_{ij}\neq 0} (F_{ij}(t_k) - B_{ij})\, V_{ij}(t_k) \tag{1.4}$$

$$P_{spot,i}(t_k) = \sum_{j:t_k=d_j \wedge \overline{V}_{ij}\neq 0} (S_{ij} - B_{ij})\, (\overline{V}_{ij} - \sum_{t_h<t_k} V_{ij}(t_h)). \tag{1.5}$$

Note that, given t_k, if not exists j such that $t_k = d_j$ (i.e. the trading date t_k is before any delivery date), then $P_{spot,i}(t_k) = 0$. The sign of the price will be positive when selling and negative when buying.

Note that the splitting of the total profit P into P_{fwd} and P_{spot} is a consequence of the constraints (1.2), since at the delivery date d_j the position must be closed according to the initial volume exposure $\overline{V}_{ij}$ for all the underlyings x_i.

In order to maximize the profit we need to appropriately set the values of the traded volumes $V_{ij}(t_k)$, $k = 1, \ldots, K$, for all underlyings and delivery dates. The traded volumes are subject not only to the position constraints (1.2), but also to additional constraints due to the market liquidity and admissible profit at risk as explained in the next Sections 1.1.3 and 1.1.2.

So for every underlying x_i and every delivery d_j, from the current month up to the delivery date, we have to decide which fraction of the total volume exposure $\overline{V}_{ij}$ we should trade.
Observe that the first kind of uncertainty that arises in this problem is given by the

prices: if we knew exactly the evolution of the market, we could choose the optimal combination of volumes, which allows us to close the position and satisfy all the other constraints of the problem. Note that there is another source of uncertainty: the initial volumes $\overline{V}_{ij}$ are taken as an input, but actually they depend also on the forecast of what will be produced by our power plants and this might change because of the weather (for example, drought could force us to stop hydroelectric production) and because of the market conditions (if the costs of producing with CCGT are higher than the price of electricity, we do not turn on the CCGTs). Usually the price movements between one month and another are more relevant than adjustments in forecast volumes and this is the reason why we have focused more on price uncertainty.

1.1.1 Portfolio description

We are interested in a portfolio in the energy market. The main underlyings involved are the ones typical of the Italian energy market:

- PUN, i.e. the Italian power price, in particular PUNBASE and PUNPEAK;
- PSV, i.e. the Italian gas price;
- EUA (related to CCGT);
- TTF, i.e. the Dutch gas price, the most traded product in the European gas market;
- PFOR, i.e. an Italian index that depends on TTF.

On one side there is power plant production, in particular:

- hydroelectric power plants and waste-to-energy plants: they have to sell the electricity (PUNBASE, PUNPEAK) they planned to produce;
- CCGTs: they have to sell the electricity (PUNBASE, PUNPEAK) they planned to produce; at the same time they have to buy the gas (PSV) and the permission to emit CO_2 (EUA).

On the other side there are customer supplies, in particular:

- power supplies (PUNBASE, PUNPEAK);
- gas supplies, in particular:
  - PSV;
  - the spread PSV - TTF and the spread PSV - PFOR; these two spreads are related to the gas sales indexed to TTF and PFOR, respectively.

It is mandatory not to speculate on prices, but instead to hedge the position in the most convenient way.

1.1.2 Risk measure: PaR (Profit at Risk)

The company defines the maximum limit of total economic capital for market risks, on an annual basis, which qualifies the appetite for risk associated with a certain level of return. This value determines the maximum accepted variation in EBITDA[1] due to adverse commodity price developments over the reference time interval and given a predefined confidence level. With the passage of time, the level of risk on the horizon of the reference accounting year is reduced by reducing the open positions and increasing the portion of realised PL accrued on closed positions. Over time, price volatility is also reduced due to the shortening of the time horizon; then decreasing monthly risk limits will be defined. These limits will be proposed by the risk manager on the basis of the calculation of the risk reduction by time, in line with the annual limit.

For this reason, for each trading date t_k, we need to impose a bound on the risk by using the Profit at Risk (PaR), which is the p-th percentile (typically $p = 1$) of the distribution of the profit for the portfolio associated to a policy where all volume exposures of all underlyings are traded at the delivery month at the spot price instead of being traded at the forward price curve $F_{ij}(t_1)$, $j = 1, \ldots, M$, known at the current time t_1 when portfolio optimization is performed.

[1] A company's earnings before interest, taxes, depreciation, and amortization.

This method for assessing the risk for a portfolio has been used since the start of the electricity market ([3]).

In order to assess the PaR, we can estimate the profit distribution at the trading date t_k by Monte Carlo simulation using the profit expression

$$\sum_{i=1}^{L} \sum_{\{j:\, d_j > t_k\}} \left(S_{ij} - F_{ij}(t_1)\right) V_{ij}^k, \tag{1.6}$$

and extracting multiple samples of the spot prices S_{ij}, $i = 1 \ldots, L$, $j = 1, \ldots, M$. We can then consider the first percentile ($p = 1$) corresponding to the minimum value of the profit guaranteed with confidence $(100 - p)\%$ and impose that it is not smaller than a certain target value. Alternatively, we can adopt an analytic approach similar to that used for the Value at Risk ([4]): we compute the standard deviation of the random variable (1.6) and multiply it by the inverse of the cumulative Gaussian probability distribution for a given $(100 - p)\%$ confidence by making the simplifying assumption (see [5]) that (1.6) is a Gaussian random variable because the spot prices S_{ij}, $i = 1 \ldots, L$, $j = 1, \ldots, M$, are jointly Gaussian.

By collecting in vector V_k all the residual volumes V_{ij}^k, $i = 1, \ldots, L$, $j \in \{1, \ldots, M, \text{ such that } d_j > t_k\}$, defined in (1.1), (1.6) can be rewritten in compact form as:

$$\sum_{i=1}^{L} \sum_{\{j:\, d_j > t_k\}} \left(S_{ij} - F_{ij}(t_1)\right) V_{ij}^k = S_k^\top V_k$$

where S_k is a suitably defined vector whose elements are of the form $S_{ij} - F_{ij}(t_1)$, $i = 1, \ldots, L$, $j \in \{1, \ldots, M, \text{ such that } d_j > t_k\}$. The variance of (1.6) can then be expressed as

$$V_k^\top \Sigma_k V_k$$

where Σ_k is the variance matrix of S_k, which can be estimated by extracting multiple independent realizations of the spot prices S_{ij} $i = 1 \ldots, L$, $j \in \{1, \ldots, M, \text{ such that } d_j > t_k\}$.

As a result, the PaR constraint at t_k rewrites as

$$\alpha \sqrt{V_k^\top \Sigma_k V_k} \leq \eta_k \iff V_k^\top \Sigma_k V_k \leq \frac{\eta_k^2}{\alpha^2}, \tag{1.7}$$

where α is the inverse of the cumulative Gaussian probability distribution for the desired confidence level $(100-p)\%$ and η_k is the upper bound on the PaR at time t_k.

Note that the numerical formula (the p-th percentile of the distribution of the profit) is more precise, but it is not a quadratic constraint. So we calibrate parameter α in Formula (1.7) so that the two PaR values (the numerical and the analytic one) are the same at the beginning, then we look for the optimal decision using the analytical formula. This approximation could be justified by the fact that typically the PaR does not change much during the month, so we remain in the neighborhood of the initial value.

1.1.3 Other constraints

In addition to the risk constraint, there are other constraints:

- liquidity,
- tradability,
- position.

The liquidity constraint is simply a maximum quantity given for each tradable product of each underlying on each trade date; note that since there is only one trading date per month, the liquidity constraint refers to the maximum tradable quantities in the entire month. The position constraint is the one given by Formula (1.2). We dedicate a subsection for the tradability.

Tradable products

We cannot trade any monthly forward contract at any trading date because, in general, energy markets are not liquid enough.

At each trading date t_k, for each underlying x_i, we can choose between nine tradable products:

- the next four monthly forward contracts;

- the next four quarterly forward contracts;

- the next one yearly forward contract.

Given a trading date t_k, for each underlying x_i, usually we consider less than nine products, because we are interested only in the products that involve the year of interest. In the examples below the year of interest is 2021.

Figure 1.3: The 9 products that can be traded at the trading date in February 2020: the next four months, none of them is on 2021; the next four quarters, i.e. Q_2^{20} (the second quarter of 2020), Q_3^{20} (the third quarter of 2020), Q_4^{20} (the fourth quarter of 2020), Q_1^{21} (the first quarter of 2021); if we are interested only in 2021, we have only one quarter available, Q_1^{21}; the next year (i.e. 2021). Therefore there are only two products for the year of interest: Q_1^{21} and the whole year.

Figure 1.4: The 9 products that can be traded at the trading date in November 2020: the next four months, i.e. December 2020, January 2021, February 2021, March 2021 (only three of them are in 2021); the next four quarters, i.e. the four quarters of 2021; the next year (i.e. 2021). In this case there are eight products for the year of interest: January 2021, February 2021, March 2021, Q_1^{21}, Q_2^{21}, Q_3^{21}, Q_4^{21} and the whole year.

When a period begins it is no longer tradable. For instance, as shown in Figure 1.4, when the trading date is in January 2021, year 2021, quarter Q_1^{21}, and January 2021 are no longer tradable. Also, trading a forward contract over a period like a quarter means that we divide the volume by each month of the period. For example, buying 30 000 MWh of PUN of the Q_4^{21} means buying 10 000 MWh for each month from October 2021 to December 2021.

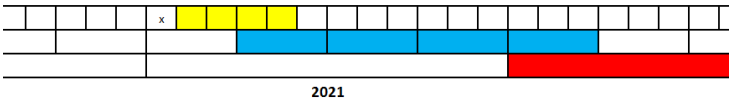


Figure 1.5: The 9 products that can be traded at the trading date in January 2021: the next four months, i.e. February 2021, March 2021, April 2021, May 2021; the next four quarters, i.e. Q_2^{21}, Q_3^{21}, Q_4^{21}, Q_1^{22} (only three of them are in 2021); the next year (i.e. 2022); as soon we are in 2021, we cannot trade the year 2021. So there are seven products available: February 2021, March 2021, April 2021, May 2021, Q_2^{21}, Q_3^{21}, Q_4^{21}.

Products are not mutually exclusive and we can trade different products in the same month. For instance, in Example 1.4, we can trade the product with delivery in January 2021 with a monthly (yellow), quarterly (blue) and yearly (red) product. In order to compute the residual exposure at the trading date in November 2020 for the delivery in January 2021, we must then consider three products, the monthly, the quarterly and the yearly one.

Each traded volume $V_{ij}(t_k)$ needs to be parameterized in terms of tradable products, that represent the decision variables of the optimization problem.

For each delivery date d_j we denote by M_j, Q_j, and Y_j respectively the month, quarter, and year to which d_j belong. Correspondingly, $V_{M_j}^{ij}(t_k)$, $V_{Q_j}^{ij}(t_k)$ and $V_{Y_j}^{ij}(t_k)$ are the amount of monthly, quarterly and yearly products. Then, the cumulative volume for underlying x_i with delivery date d_j at the trading date $t_k \leq d_j$ is given by

$$V_{ij}(t_k) = V_{M_j}^{ij}(t_k) \cdot \mathbf{1}_{M_j}(t_k) + \frac{V_{Q_j}^{ij}(t_k)}{3} \cdot \mathbf{1}_{Q_j}(t_k) + \frac{V_{Y_j}^{ij}(t_k)}{12} \cdot \mathbf{1}_{Y_j}(t_k) \qquad (1.8)$$

where

- $\mathbf{1}_{M_j}(t_k) = 1$ if t_k is either month M_j or one of the 4 months before M_j, and 0 otherwise;
- $\mathbf{1}_{Q_j}(t_k) = 1$ if t_k is between twelve months before the start of Q_j and the month before the start of Q_j, and 0 otherwise;

- $\mathbf{1}_{Y_j}(t_k) = 1$ if t_k is between twelve months before the start of Y_j and the month before the start of Y_j, and 0 otherwise.

Formula (1.8) can be rewritten in compact form as

$$V_{ij}(t_k) = A_{ij}(t_k)^\top X_{ij}(t_k) \tag{1.9}$$

where

$$X_{ij}(t_k) = \left[V^{ij}_{M_j}(t_k)\, V^{ij}_{Q_j}(t_k)\, V^{ij}_{Y_j}(t_k)\right]^\top \tag{1.10}$$

collects all volumes related to monthly, quarterly, yearly products for the considered underlying and delivery date for the trading date t_k and $A_{ij}(t_k)$ is a suitably defined row vector.

Position constraints in (1.2) then rewrite as

$$\sum_{k:t_k \le d_j} A_{ij}(t_k)^\top X_{ij}(t_k) = \overline{V}_{ij} \tag{1.11}$$

For each trading date t_k, for each underlying x_i, we have boundaries on the tradable quantity, which translated into the liquidity constraints

$$X_{ij}(t_k) \le \overline{X}_{ij}(t_k), \qquad i = 1, \dots, L, j = 1, \dots M, k = 1, \dots K, \tag{1.12}$$

which entail that usually we cannot close the position in a single trading date.

For each trading date t_k, for each underlying x_i, for each of the nine tradable products, we have boundaries on the tradable quantity; this means that usually we cannot close the position in a single trading date. Indeed, trading a huge volume of a given product on a single trading date would have a significant market impact (see [6]), and would possibly change the current market price.

1.1.4 Problem formulation

Our goal is to maximize the cumulated profit (1.3), without exceeding the risk limit (1.7) for each trading date. There is also the liquidity constraint, that acts like a

domain $\mathcal{D}$ for the decision variables; moreover there is the position constraint, i.e. we have a given total quantity $\overline{V}_{ij}$ for each underlying and each delivery. We can normalize the decision variables, that is considering the actions over time as a fraction of the total exposure. By changing the sign, we can think at minimizing instead of maximizing. So the problem is

$$\min_{(V_{ij}(t_k))_{ijk} \in \mathcal{D}} -P \quad \text{subject to (1.7), (1.2)} \quad \forall t_k$$

By looking at Formula (1.3) one might notice that

- the volume exposures $\overline{V}_{ij}$ and the budget level B_{ij} are given as input for each couple underlying-delivery;

- our decision variables are the volumes to be traded, $X_{ij}(t_k)$, that in Formula (1.3) are translated as their cumulative action on delivery dates, $V_{ij}(t_k)$ (see Formulas (1.9) and (1.10));

- the forward prices $F_{ij}(t_k)$ are not given for $t_k > t_1$, they are uncertain variables; spot prices are only a special case of forward prices, when the trade date and the delivery date coincide; for $t_k = t_1$, $F_{ij}(t_k)$ represent the current forward prices, so they are known.

By knowing for sure the evolution of the forward curves we could guess the optimal sequence of actions in order to optimize (1.3) without violating the constraints. We know for sure the current forward price and the current spot price; for example if the current trading date is on January of the year of interest, for each underlying x_i we know the prices of the current forward tradable products (February, March, April, May, Q_2, Q_3, Q_4) and the spot price of the current month (January), but what about the prices of the next trading dates? How can we choose a strategy without knowing the value of some variables? The fact is that we do not know the evolution of the forward curves, so we have to use an approach that can deal with uncertainty.

1.2 A deterministic approach

1.2.1 Stylized objective function

Before considering the whole problem, we start from a naive approach: let's suppose that we can trade only today, at the current forward prices, using the current tradable actions, otherwise we will trade at delivery (see Figure 1.6). So we do not consider intermediate dates, it is a simplified approach. Moreover we use a forecast (called *view*) for the price at delivery: in this way everything it is known in advance, there are no uncertain variables, it is a deterministic optimization, where the objective function becomes

$$PL_{naive}(t_1) = PL_{currentfwd}(t_1) + PL_{view}(t_1) \tag{1.13}$$

$$PL_{currentfwd}(t_1) = \sum_{i=1}^{L} \sum_{j:d_j>t_1} (F_{ij}(t_1) - B_{ij})V_{ij}(t_1) \tag{1.14}$$

$$PL_{view}(t_1) = \sum_{i=1}^{L} \sum_{j:d_j>t_1} (View_{ij} - B_{ij})(\overline{V}_{ij} - V_{ij}(t_1)) \tag{1.15}$$

So instead of Formula (1.3), we have Formula (1.13). Note that in the rest of this chapter we can omit t_1, the current trading month, as there is no other t to get confused with in the deterministic approach.

Python package

This is a deterministic optimization, since there is not uncertainty on the involved variables. We can tackle this optimization using a python package: *scipy.optimize*; in particular we use the function *scipy.optimize.minimize* with the method SLSQP (**S**equential **L**east **SQ**uares **P**rogramming) that minimizes a function of several variables with any combination of bounds, equality and inequality constraints, i.e. a problem of this form:

$$\min_x f(x) \text{ subject to}$$
$$c_j(x) = 0 \; \forall j \in \mathcal{E}$$
$$c_k(x) = 0 \; \forall k \in \mathcal{I}$$

$$lb \leq x \leq ub$$

where $\mathcal{E}$ and $\mathcal{I}$ are sets of indices containing equality and inequality constraints respectively, lb stands for lower bound, ub stands for upper bound. The SLSQP method is the default choice for *scipy.optimize* when there are constraints; it is based on the algorithm originally implemented by Dieter Kraft [7], that is basically a Newtonian method for solving the Lagrange system; in order to understand it better see also the chapter "Sequential Quadratic Programming" of [8].
This is a sequential numerical algorithm easy to use and of general use; we are aware that there are more specific numerical algorithms (such as the one used in the Section 3.3) and it would be interesting to analyze the output by modifying the numerical algorithm. However the use of a sequential algorithm gives the advantage of obtaining a solution even if it is not possible to fully satisfy all the constraints; there are cases where having a result might be interesting even if it is not optimal, such as when the risk limit is far from the initial PaR (see Subsection 1.2.2). Furthermore we are forced to use an algorithm that does not require convexity for some variants such as that of Subsection 1.2.4.

1.2.2 Cluster analysis varying the initial guess

As said before this is a deterministic optimization that can be solved easily using a python package such as *scipy.optimize*. This means using an iterative method starting from an initial guess. So one might be wonder

- how we should choose the initial guess;
- whether the final outcome could differ according to the initial guess.

Initial guess

We check the correct sign (buy or sell) of each product i and delivery j according to the corresponding sign of exposure $\overline{V}_{ij}$.
Then we choose zero or the maximum liquidity for each product; considering all the possible combinations would be expensive, so we choose randomly a fixed

number N of combinations, i.e. $N = 1000$. This initial guesses satisfy the liquidity constraint by construction.
For each guess we compute the PaR, then the distance of the PaR with respect to the target. We want to penalize those that does not satisfy the risk constraint, so we modify the distance measure by doubling it up when the guess does not satisfy the constraint.
We consider as the best initial guess the one with the lower distance.

Cluster analysis

Instead of considering just one guess we can consider a sample of initial guesses as starting points of the iterative solving method. We choose the ones that satisfy the risk constraint if there are some, otherwise we choose the best $n = 100$ guesses according to the distance described in the paragraph before. Considering all the $N = 1000$ combinations generated at the previous step would cost too much time, moreover if we choose the best guesses it is more likely that the iterative method find the optimum.
We would like to dispel some doubt:

- is essentially unique or is there more than one optimum? Can we distinguish some clusterization?
- does the iterative method reach the optimum independently from the initial guess?
- the iterative method give in output a message that represents the final status such as ”Optimization terminated successfully”, ”Iteration limit reached”, ”Positive directional derivative for linesearch”, ”Inequality constraints incompatible” and so on and so forth: are there any pattern on the final output according to the final message?

In principle, there might be solutions that are equivalent in terms of PaR and the global value of the objective function, but that can be different because

- they allocate differently the volume between the tradable products of the same underlying,
- they allocate differently the volume between products of highly correlated underlyings,
- they allocate differently the PL between now ($PL_{currentfwd}$) and at delivery (PL_{view}).

As you can see in the examples below (see Figure 1.7 and 1.9), the final status "Optimization terminated successfully" corresponds to a unique solution, when the message is "Iteration limit reached" (with maximum iteration 1000) the solution is usually closer with respect to the case "Positive directional derivative for linesearch".
By taking a look at the cluster numerically found, if the solution is essentially unique, we expect a single cluster; in fact there are cases in which it is so (see Figure 1.10); in other cases more than one cluster is found (see Figure 1.8), but by a closer inspection there is just a transferring of volumes between the tradable products of the same underlying that are involved in the same deliveries (for example a quarter and its months). There are cases in which the initial PaR is too far from the target, due to disruptive changes in the market; in these cases the clusters might be more significant: for example there could be a cluster that has higher risk, but higher profit and another cluster that has lower risk, but lower profit (see Figure 1.11).

Python package

In order to perform clustering we used the method *KMeans* from the python package *sklearn.cluster* in order to choose centroids [2] that minimise the within-cluster sum-of-squares criterion, a measure of how internally coherent clusters are:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \left(\|x_i - \mu_j\|^2 \right)$$

[2] a centroid is the mean of the samples in the cluster

where μ_j are the centroids, x_i are all the given points. This algorithm is described in [9].

In order to plot the dendrogram we used the method *dendrogram* from the python package *scipy.cluster.hierarchy*; in order to construct the dendrogram we used the method *linkage* from the python package *scipy.cluster.hierarchy*, in particular we used the Ward variance minimization algorithm, where the distance between two clusters u and v is given by the following formula:

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 - \frac{|v|}{T}d(s,t)^2}$$

where $u = s \cup t$, v is an unused cluster in the forest, $T = |v| + |s| + |t$ where the symbol $|\cdot|$ represents the cardinality of its argument. For more details see [10].

We can use the dendrogram as a way of choosing the best number of clusters; but we can also use other metrics, like the silhouette score (see [11]). The silhouette score is calculated using the formula

$$\frac{(b-a)}{max(a,b)}$$

where for each sample a is the mean intra-cluster distance and b is the distance between a sample and the nearest cluster that the sample is not a part of (mean nearest-cluster distance). Observe that this score is only defined if number of labels is between two and the number of samples minus 1; moreover this score is a values between 1 and -1; if it is 0 there are overlapping clusters, when it is 1 the clusters are perfectly distinguishable, if is is negative it means that a sample has been assigned to the wrong cluster.

1.2.3 Variant: soft constraints

There might be times when it is hard to satisfy all the constraints, so we can relax the constraints. So, instead of imposing to verify the liquidity, position and risk constraints, we insert the constraints in the objective function (such as in equation (1.16)): every time we do not fulfill a constraint, we pay a penalty. We can define a weight (α_i $i = 1, ...4$) for the penalty of each constraint to represent the fact that

we might be interested in satisfy a constraint more than another.

$$\alpha_1(PL_{currentfwd} + PL_{view}) - \alpha_2(target - PaR)^+ + \\ -\alpha_3 \left(\sum_{il} (V_{x_i}(l) - ub_{il})^+ + (-lb_{il} - V_{x_i}(l))^+ \right) - \alpha_4 \left(\sum_{ij} (V_{ij} - \overline{V}_{ij})^+ \right) \tag{1.16}$$

where ub_{il} and lb_{il} are the upper bound and the lower bound of the tradable product l of underlying x_i (so they represent the liquidity constraint), $V_{x_i}(l)$ is the volume of underlying x_i and product l, V_{ij} is the cumulative action of the tradable products of underlying x_i for delivery d_j. Note that this objective function is based on the same idea of the one used in Section 4.4, where instead of constraints, we introduce a penalty proportional to the exceeding the limit value.

1.2.4 Variant: adding bid-ask cost

In order to avoid, for example, selling the quarter and buying its months, i.e. in order to avoid zero-sum actions, we can add a regularization term, in order to take into account that each trading action has a cost; so in Formula (1.13) we add a term:

$$Cost_{bid-ask} = \sum_{x_i} \sum_{l} |V_{x_i}(l)| c_{il} \tag{1.17}$$

where c_{il} is the semi-distance between bid and ask for the tradable product l of the underlying x_i. Observe that by adding this term to the objective function, on the one hand we can force the algorithm to prefer more liquid products, on the other hand the problem becomes a non-convex optimization, while most of the optimization algorithms require convexity.

1.2.5 Remarks on the deterministic approach

The main advantage of this deterministic approach is that it is the easiest to understand and implement. If a solution exists, then it is substantially unique regardless of the starting point (see Figures 1.7, 1.8, 1.9, 1.10), apart from changes in the distribution of volume for products that concern the same maturities (for example one

quarter and the corresponding three months). If there is no solution to the given constraints, the algorithm will end up in one of two clusters (see Figure 1.11): it either tries to find a better profit with higher risk or a lower risk with lower profit. We tested some variants for the objective function: when the constraints are too hard, you can relax them (see Section 1.2.3); to get more realism, you can add the bid-ask cost (see Section 1.2.4), but keep in mind that this way you lose convexity, so in general it is more difficult to find solutions.

The main drawback of this approach is that it leaves out many aspect of the complete problem, in particular the fact that there are many intermediate trading dates: we need simulations for the evolution of the forward curves (see Chapter 2) and methods that are able to extract a decision given those simulations (see Chapters 3 and 4).

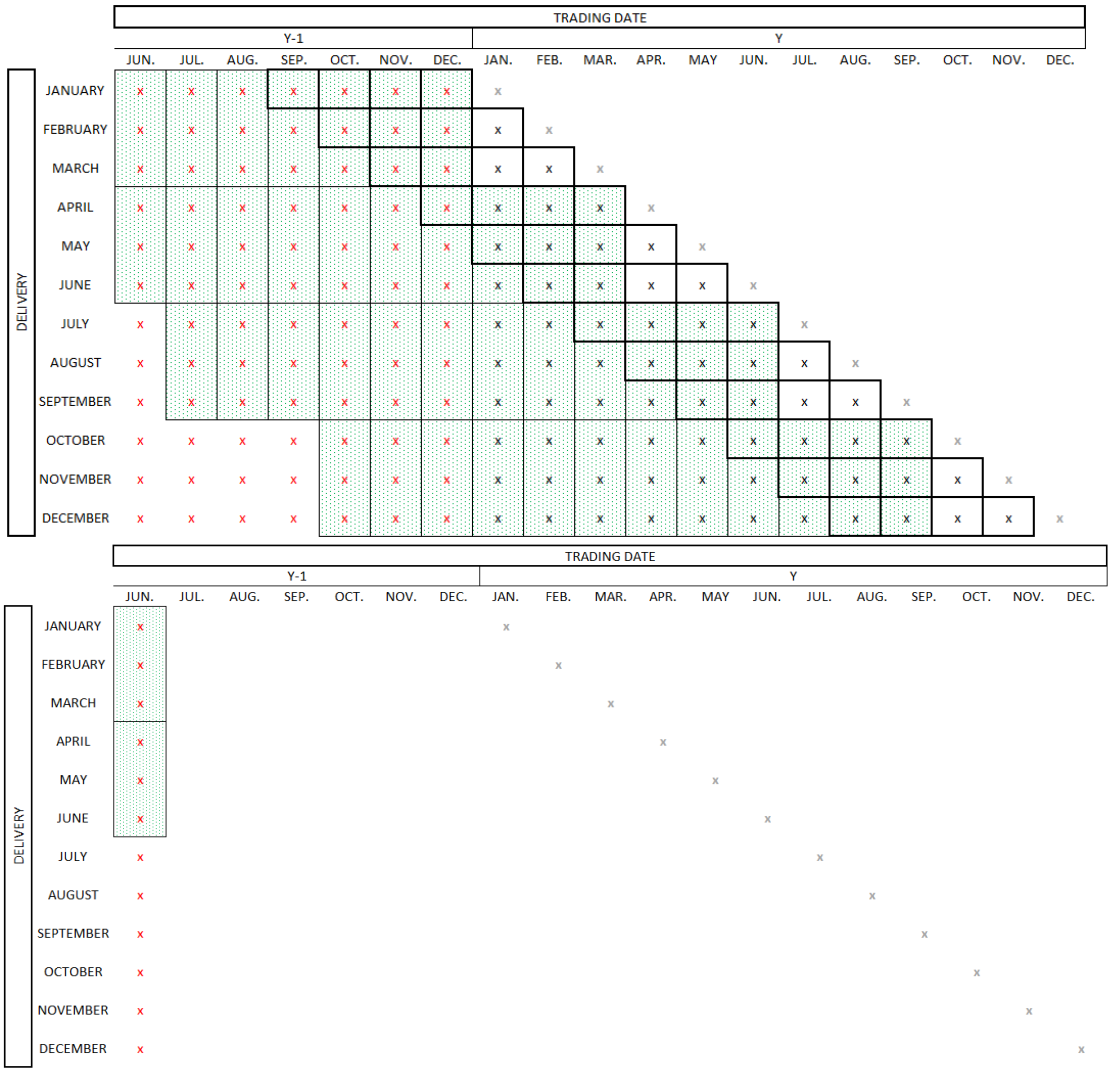


Figure 1.6: Let's say today is June of year Y. Year of interest: Y+1. Above: all the tradable actions for all trading dates (for a single underlying), the yearly products (red), the quarterly products (green block), the monthly products (thick black edges) and the spot (grey). Below: the tradable actions according to the naive approach: today and at delivery.

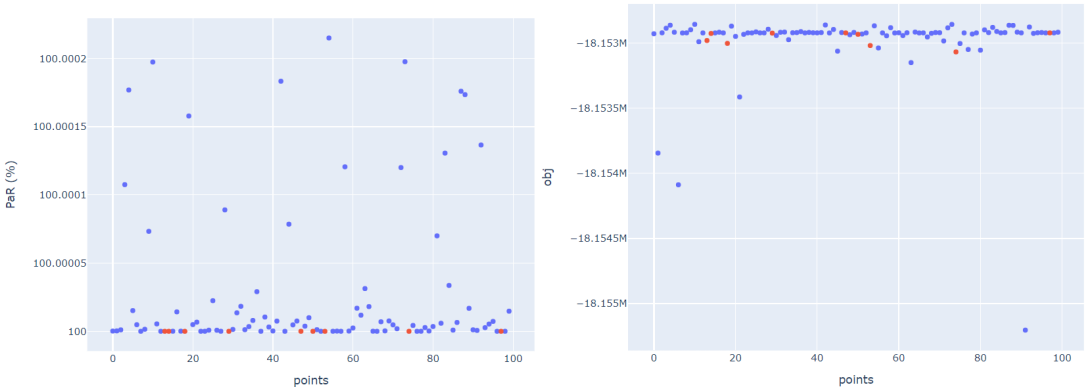


Figure 1.7: Trading month: October of year Y. Delivery months: the months of year Y+1. On the left: PaR value (reported as a percentage of the target PaR) after the optimization varying the initial guess (the best 100 initial guesses over 1000). On the right: objective function value according to the optimization varying the initial guess. Blue points correspond to the message ”Iteration limit reached”, red points correspond to the message ”Optimization terminated successfully”. As you can see, the blue points are more scattered, anyway all the values are very close one to each other; in particular the ones corresponding to the message ”Optimization terminated successfully” are essentially the same.

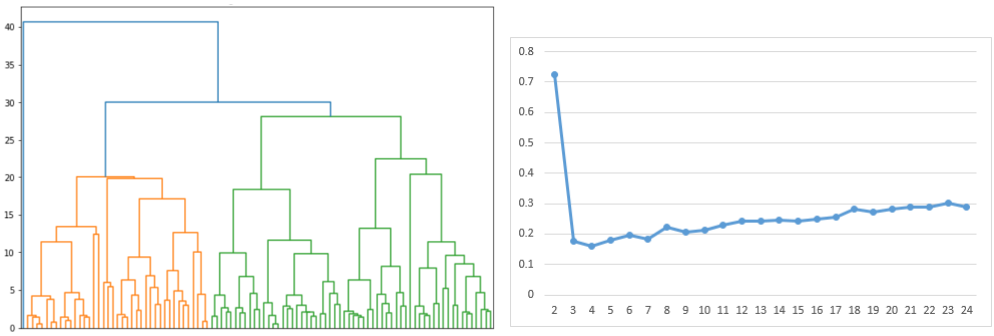


Figure 1.8: Trading month: October of year Y. Delivery months: the months of year Y+1. On the left: the dendrogram representing the clustering of all the solutions (on the x-axis) according to the distance (on the y-axis). By looking at the hugest jump in distance, we could pick two clusters, as suggested by the colors. On the right: the silhouette score for a number of clusters between 2 to 24; between these values the highest score is reached by two. By a closer inspection to the centroids found using two clusters, we found that the two solutions are equivalent in terms of PaR and total PL and also in terms of total volume traded at the forward price, aggregating by underlying: for the same underlying there is a different allocation of the same total volume between the tradable products, in particular between quarters and the calendar.

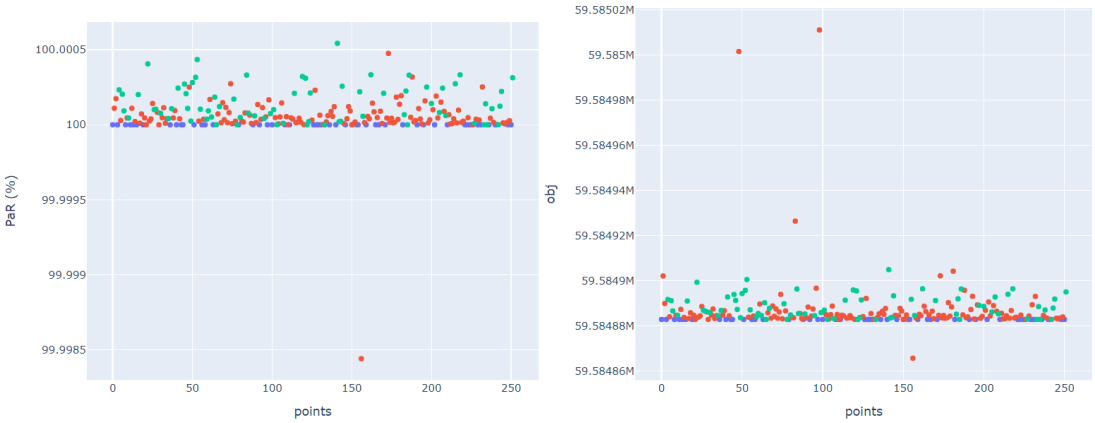


Figure 1.9: Trading month: November of year Y. Delivery months: the months of year Y+1. On the left: PaR value (reported as a percentage of the target PaR) after the optimization varying the initial guess; we used 252 initial guesses, all the ones (over 1000 random combinations) that satisfy the risk constraint. On the right: objective function value according to the optimization varying the initial guess. Blue points correspond to the message ”Optimization terminated successfully”, green points correspond to the message ”Iteration limit reached”, red points correspond to the message ”Positive directional derivative for linesearch”. As you can see, the red points are more scattered, anyway all the values are very close one to each other; in particular the ones corresponding to the message ”Optimization terminated successfully” are essentially the same.

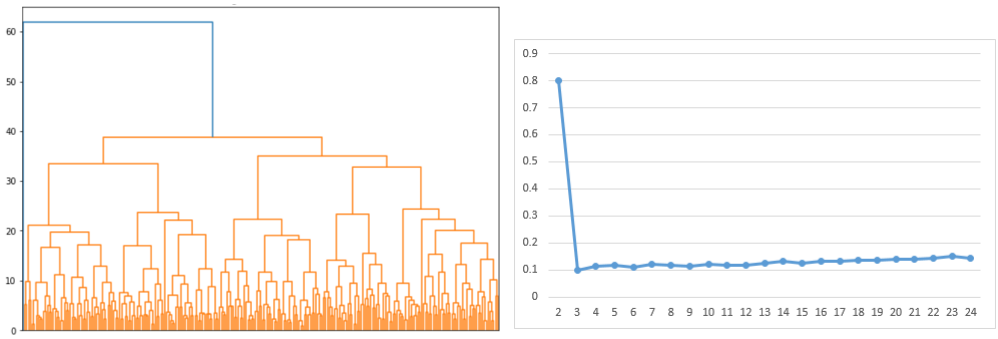


Figure 1.10: Trading month: November of year Y. Delivery months: the months of year Y+1. On the left: the dendrogram representing the clustering of all the solutions (on the x-axis) according to the distance (on the y-axis). By looking at the hugest jump in distance, we could pick one cluster, as suggested by the colors. On the right: the silhouette score for a number of clusters between 2 to 24; between these values the highest score is reached by two, the other scores are almost the same low value. As in the case of October, we could say that apart from a different allocation between quarters and the calendar, there is a unique solution.

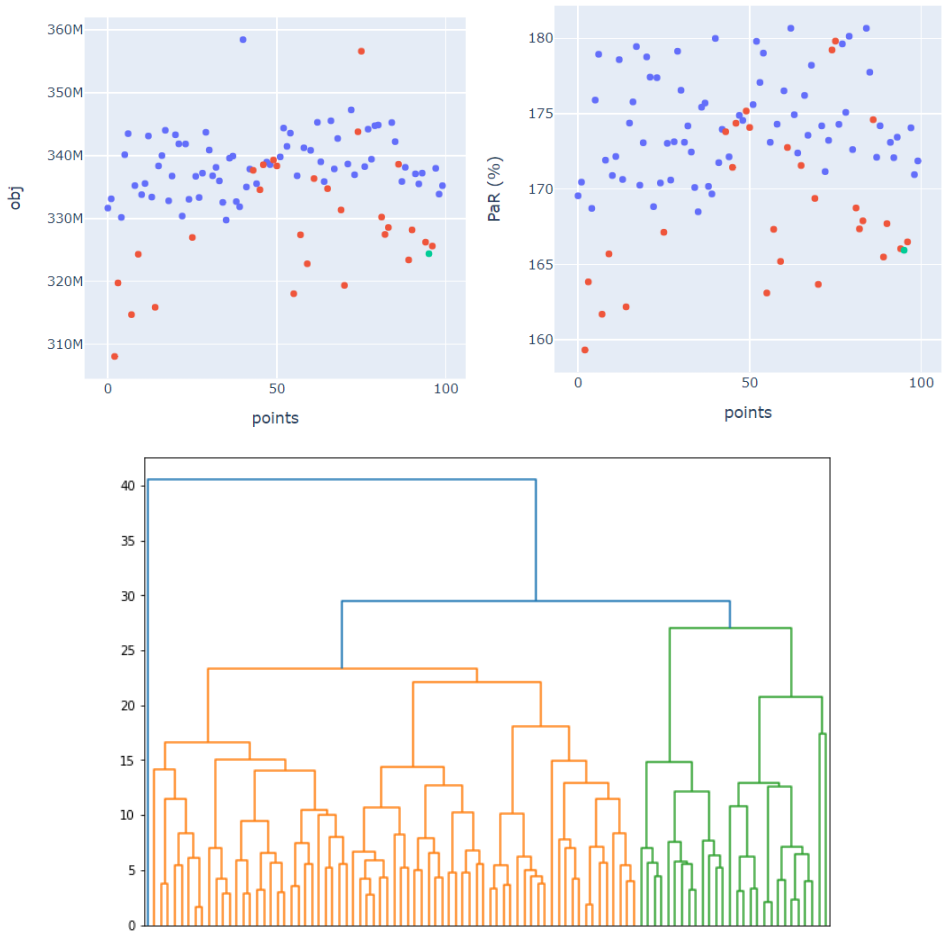


Figure 1.11: Trading month: December of year Y. Delivery months: the months of year Y+1. Starting from the top: the value of objective function (on the y-axis) for each different starting point (on the x-axis), coloured by the clusters you can see in the dendrogram of this figure. Figure on the middle: the same graph for PaR, reported as a percentage of the target PaR. Here you can distinguish that the there are two clusters (blue and red) corresponding one to higher risk, higher profit, the other to lower risk, lower profit. Figure on the bottom: the dendrogram representing the clustering of all the solutions (on the x-axis) according to the distance (on the y-axis). Here you can see that the there are two clusters: the orange one and the green one.

Chapter 2

Price simulations

In this chapter we propose a model for the simulation of correlated forward curves for a portfolio of the Italian energy sector.

We model forward prices directly using a Heath-Jarrow-Morton framework; spot prices are thought of as a particular case of the forward, so we can compare this model with a spot model. We point out some empirical facts about our portfolio of interest, then we point out how we have tried to separate the noise from the signal in order to have more realistic simulations. Finally we outline the market calibration of this model and the next possible steps to improve it.
The main novelty of this model is the fact that it addresses an *entire* portfolio of *forward* curves in the *Italian energy* market; moreover we propose a market calibration that has proved to be effective with a reduced computational cost. We compare the spot prices of this model with another model and we investigated the high dimensional noise pattern of the Italian energy prices. This model can be useful both for pricing and for optimization purposes.

The main idea comes from [12] and it takes shape from the Heath-Jarrow-Morton framework in interest rates ([13]): in the energy sector it is more natural to model directly the forward prices instead of starting from the spot prices process, since the derivation is not straightforward (see [14], [15], [16]).

The randomness source of this model is Gaussian; we are aware that it is not the only possible choice and that there are evidences of non-normality in the forward prices of electricity market (see [14]); a possible future development is to apply a Normal-Inverse Gaussian distribution (see [17]) to our data. It is well known (see [15]) that in the energy market, in particular in the electricity market, we need to consider more noise sources than for example in the interest rate market to describe a realistic forward curve: our empirical analysis (see Section 2.3) confirms this fact. The reasons for the presence of such a high-dimensional noise pattern are to be found in the multiple sources of information (weather, temperature, storage, seasonality, power plant shutdowns) that may affect the energy markets.
For the energy market, there are many references for the simulation of the spot prices ([18], [19], [20], [21], [22]), some reference for the simulation of forward curves ([23], [24], [25], [26], [17]), but there are still few references ([27], [28], [29], [30]) on how to simulate an entire portfolio of forward contracts in the energy market: we are interested in maintaining both a realistic correlation between different underlyings and realistic behavior for the specific underlying. There are references that speak of commodities in general ([26], [31]) but when it comes to energy, oil ([23], [24]) and gas ([25]) are the most common examples; electricity is rarely mentioned. Furthermore, we are interested in the Italian energy market, while most of the references for the simulation of forward curves do not use data from the Italian energy market; this could be due to the fact that the Italian gas and electricity market is not so liquid and was born more recently than, for example, the Nordic market ([20], [32]). The main contribution of this chapter is the selection and fitting of existing literature models for forward simulations for our particular portfolio, adding an example in a field where there are still few examples.

The rest of the chapter is structured as follows. We list the underlyings of interest and show some empirical facts between Section 2.1.1 and Section 2.1. We therefore present the proposed approach in Section 2.2, whose core is a Principal Component Analysis, which allows us to select the main factors that drive the evolution of each forward curve. In Section 2.3 we report some experiments aimed at overcoming the presence of simulations that have too large jumps for the power

products. We emphasize that this model could also be used for pricing thanks to a simple calibration to the market in Section 2.5.1 and, finally, we draw some conclusions in Section 2.6.

The content of this chapter was accepted at 8th Edition of the Energy Finance Italia Conference, at Politecnico di Milano.

2.1 Underlyings of interest and empirical facts

2.1.1 Forward curves/Spot curves

When you look at a market price, you might ask

- the trading date (t): at which time you can trade the underlying at that price;
- the maturity date (T): in which period the underlying will be delivered.

Describing a spot curve is easier than describing a forward curve (see Figure 2.1). Given a set of maturity dates $T_1, T_2, ...T_N$, a spot curve describes for each T_i, $i = 1...N$ the price at the maturity date for the maturity date itself, i.e. represents the price of the underlying at maturity, i.e. the trade date and the maturity date are the same[1].

Given the same set of maturity dates, a forward curve describes the prices at a specific trading time for all the maturity dates of interest: so you need another piece of information, the trading time.
Note that given a maturity date T_i, exists a unique spot price, S_{T_i}, but there exist as many forward prices $F_{T_i}(t)$ as trading dates $t \leq T_i$. When $t = T_i$, $F_{T_i}(t) = S_{T_i}$.
If you want to simulate the spot prices, you need a one-dimensional array with indices $T_1, T_2, ...T_N$ for each simulation (see Figure 2.1 on the right). On the other side, if you want to simulate the forward prices, you need another dimension (see

[1] We take as monthly spot price for PSV and TTF the monthly average of the Day Ahead prices, for PUN the monthly average of the MGP price (where MGP stands for Mercato del Giorno Prima), while for EUA we take the monthly average of the quotes of the futures with delivery in December of that year; this because the future with delivery in December is a particular reference price.

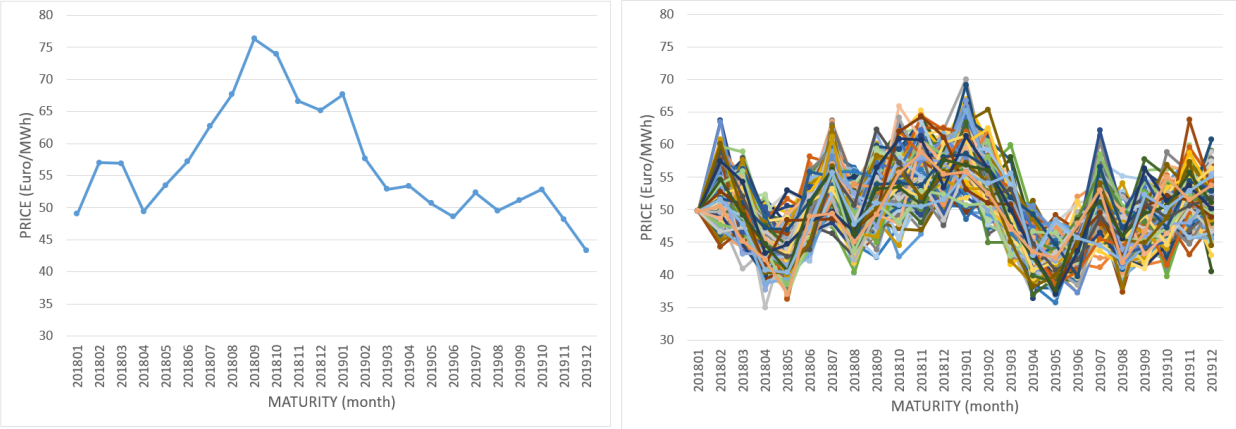


Figure 2.1: On the left: spot prices of Italian electricity for maturities from January 2018 to December 2019. On the right: simulations of the spot prices of Italian electricity for maturities from January 2018 to December 2019.

Figure 2.2): for each simulation you need a two-dimensional array with indices $T_1, T_2, ...T_N$ (maturity dates) on one dimension and $t_1, t_2, ...t_N$ (trading dates) on the other dimension.

Warnings:

- You expect that the forward curve at the trading time t_{i+1} is not independent on the forward curve at the trading time t_i, so you should not simulate separately each trading date.
- Moreover if you want to simulate a set of underlyings you should take into account of the correlations in order to be more realistic, so you should not simulate separately each underlying.

So we need a model that simulates the evolution of a set of correlated forward curves.

2.1.2 Underlyings of interest

Below are some examples of forward curves for the underlyings of interest in our analysis: PUNBASE and PUNPEAK (Figure 2.3), TTF and PSV (Figure 2.4),

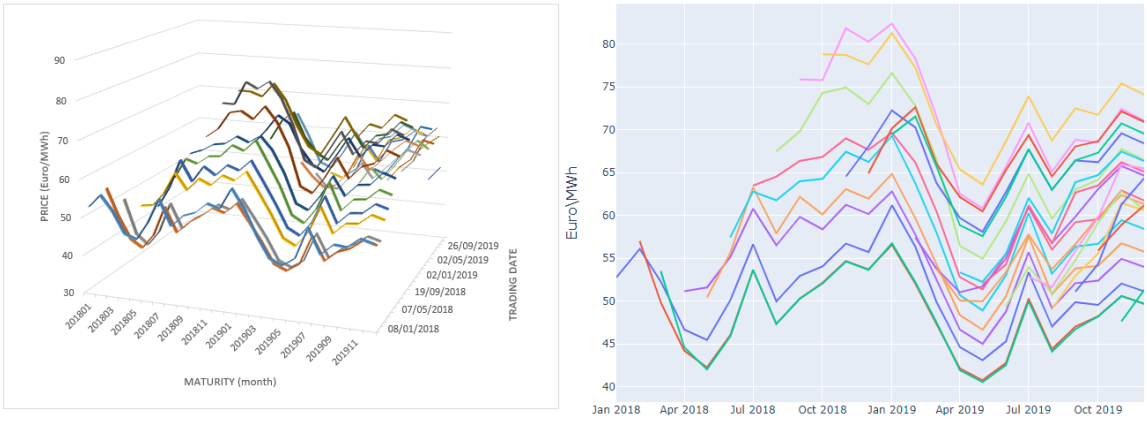


Figure 2.2: On the left: a 3D representation of the evolution of the forward curve of Italian electricity through different trading dates for the maturities from January 2018 to December 2019. On the right: the projection on the plane price-maturity. Note that this is a single scenario, but we want to simulate N evolutions of this kind.

EUA (Figure 2.5).

2.1.3 Empirical facts

By looking at these examples we can infer some empirical facts:

- the movements of PUNBASE and PUNPEAK are almost synchronous;
- to a certain extend there is synchrony also between TTF and PSV;
- EUA forward curves are linear;
- gas forward curves (PSV and TTF) have some seasonality (higher during winter, lower during summer);
- power forward curves (PUNBASE and PUNPEAK) show sharper movements than those of gas forward curves.

We underline the fact that usually in the energy markets, in particular for gas and power forward contracts, there are different delivery periods, for example months,

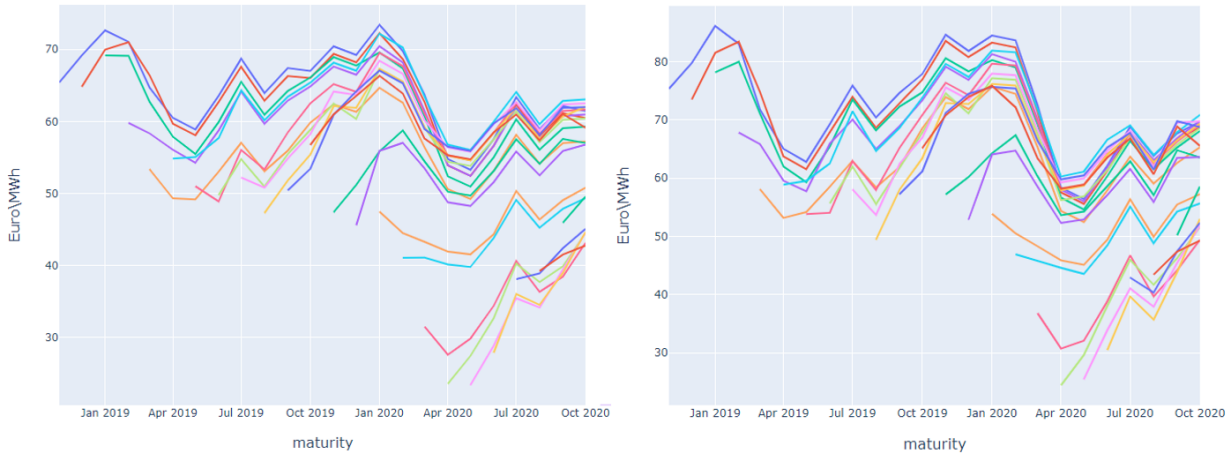


Figure 2.3: PUN (the Italian Electricity price), like most electrical market, is twofold (see https://www.eex.com/en/glossary). On the left: a base price (it refers to "the load profile of power deliveries with a constant output over 24 hours of every day of the delivery period"). On the right: a peak price (it refers to "a load profile for power deliveries with a constant output over twelve hours from 8 a.m. to 8 p.m. on any business day of the delivery period"). These are (respectively) PUNBASE and PUNPEAK forward curves of one (random) trading day per month from November 2018 to October 2020. Each line corresponds to a trading day.

quarters, years, and these periods can overlap; instead of simulating different periods and thus imposing non-arbitrage conditions (see [14]), we only simulated forward contracts with monthly delivery.

We underline also the fact that since we want to simulate the evolution of the forward curves with monthly deliveries, we take the historical forward curves with monthly deliveries; in most cases there is no provider that gives us a forward curve with monthly granularity, because especially in the case of the Italian electricity price there is not enough liquidity. For this reason we have an algorithm that every day builds the monthly forward curve given the closing prices for different delivery periods (months, quarters or years) published by a given provider; this algorithm translates quarterly prices into monthly or annual ones into monthly using appropriately calibrated weights; in this way we can extract historical forward

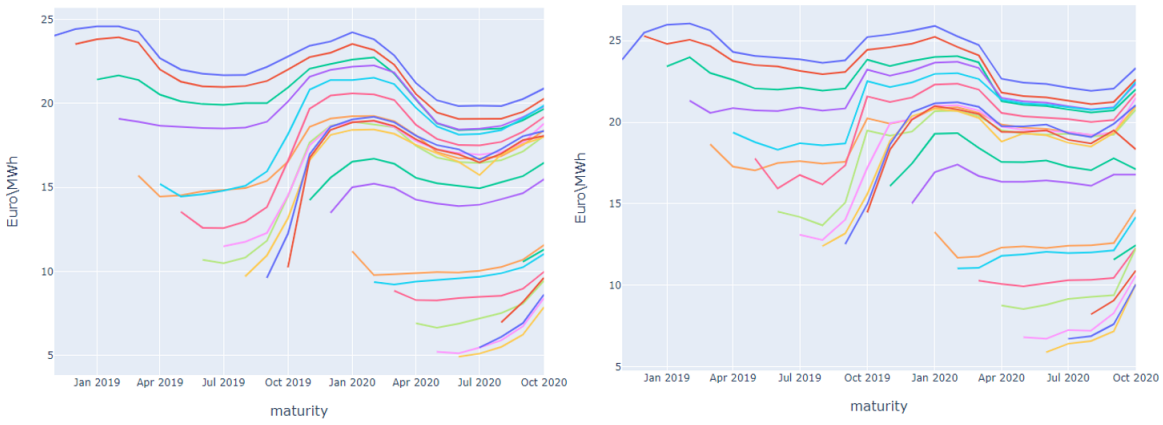


Figure 2.4: On the left: TTF forward curve of one (random) trading day per month from November 2018 to October 2020. Each line corresponds to a trading day. TTF refers to the Dutch Gas Price, the most traded Gas Price in the European market. On the right: the same for PSV; PSV refers to the Italian Gas Price.

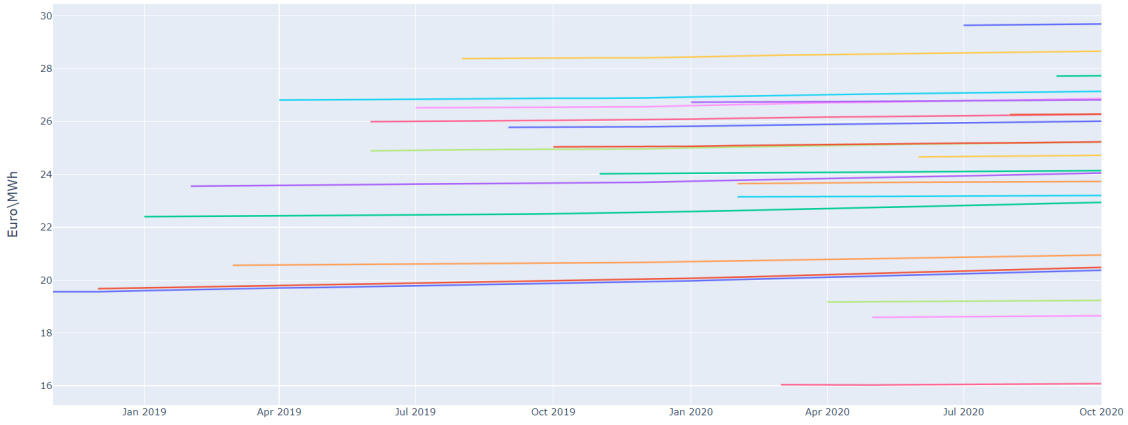


Figure 2.5: EUA forward curve of one (random) trading day per month from November 2018 to October 2020. Each line corresponds to a trading day. EUA refers to the price you have to pay if you emit CO_2; for example, you have to pay this certificate if you produce electricity with a CCGT (Combined Cycle Gas Turbine).

curves with monthly deliveries from our database.

How some events might affect price movements

The need for a simulation tool could arise for several reasons. One reason might be to test a certain strategy against different scenarios. Here you can see how important it can be to have such a tool, since reality has shown us that it is best to be prepared for a wide range of scenarios.
For example in Figure 2.6 it can be seen that starting from March 2020 the PUNBASE forward curve for deliveries in 2020 has dropped dramatically, probably due to a drop in demand due to travel and economic activity restrictions to stop the spread of Covid-19 (see [33], [34]).

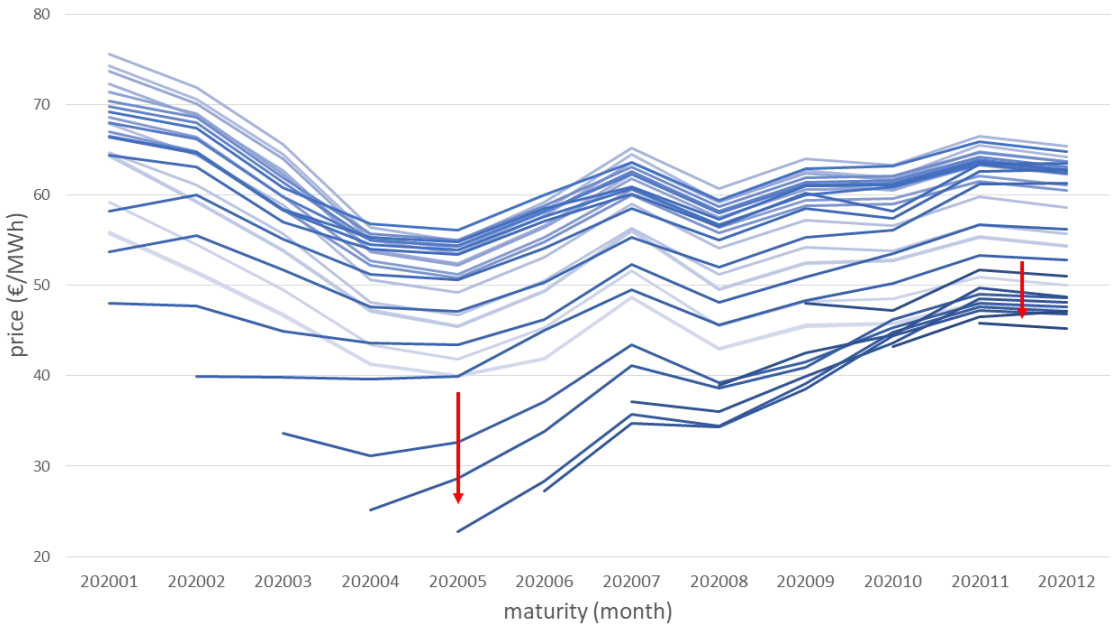


Figure 2.6: The evolution of the PUNBASE forward curve for deliveries in 2020, starting from February 2018, evaluating the monthly average on the trading dates.

In Figure 2.7, instead, it can be seen that starting from the summer 2021 the TTF forward curve for deliveries in 2021 has increased hugely, for a variety of contributing causes that end in a global energy crisis (see [35]); by looking at the historical spot prices, it can be seen how extreme the movement is (see Figure 2.8).

Moreover, because of the conflict in Ukraine, you can see a wild increase in

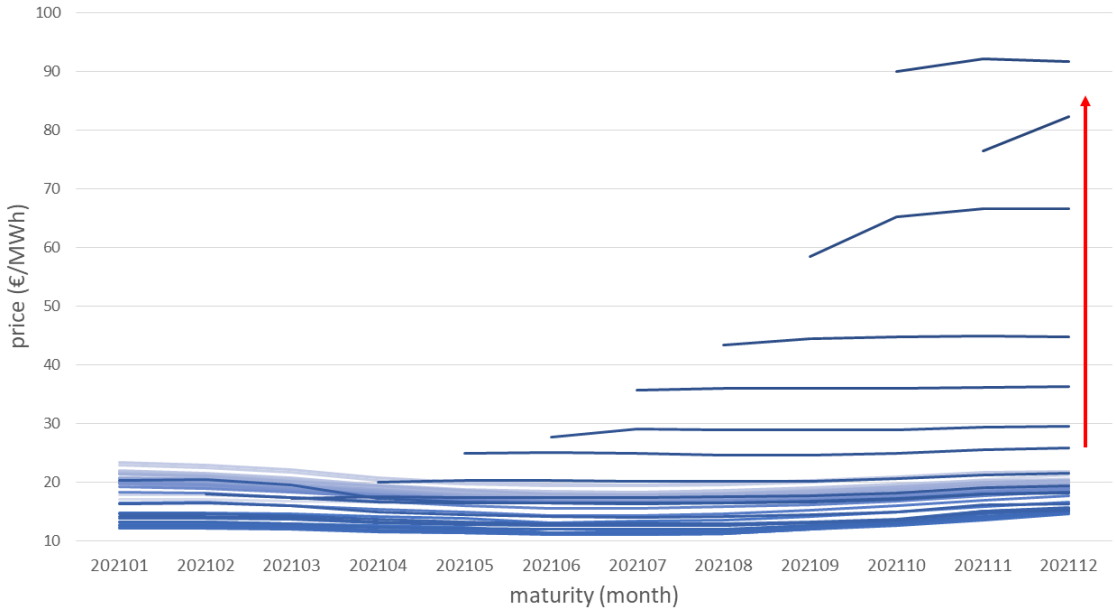


Figure 2.7: The evolution of the TTF forward curve for deliveries in 2021, starting from March 2018, evaluating the monthly average on the trading dates.

PSV forward curve for deliveries in 2022, comparing the monthly average on two consecutive months, February 2022 and March 2022 (see Figure 2.9).

2.2 Model

Starting from the model described in [12], we extracted the principal components of the covariance matrix of the logarithmic returns for each underlying of our portfolio; moreover we calibrated the deterministic movement given by seasonality ([36]). In order to simulate simultaneously all the underlyings of our portfolio we followed the approach described in [28], with the main difference that here we have more than two underlyings.
Observe that there is also a particular version of model from [12] with just two factors and with a specific functional form (see [37]), extended also for the case with more than one underlying (see [29], [30]).

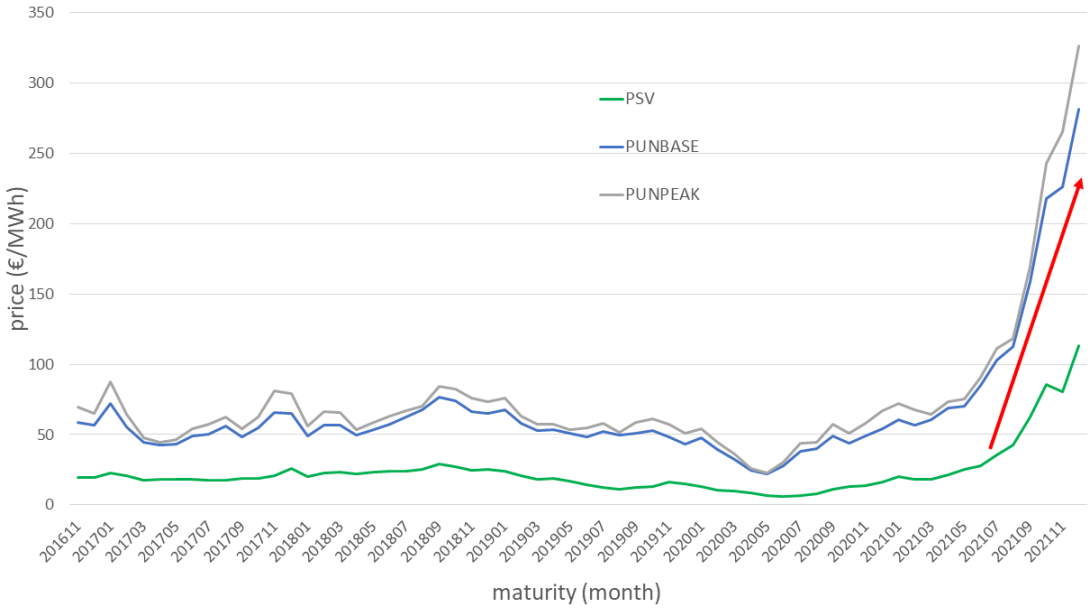


Figure 2.8: The evolution of the spot prices of PUNBASE, PUNPEAK and PSV, evaluating the monthly average starting from November 2016 until December 2021.

2.2.1 Find historical relationships

As a first step we wanted to find the historical relationship between the forward prices at different trading dates and/or different delivery. Following [12], we extracted four years of historical forward prices for the next $M = 24$ monthly maturities; i.e. for each historical trading dates t_z, $z = 1..N$ we extracted the prices

$$F(t_z, t_z + \tau_i)$$

of the next M monthly maturities τ_i, $i = 1..M$. Given this data, we computed a sample covariance matrix S of the logarithmic of the difference of the price between consecutive trading dates.

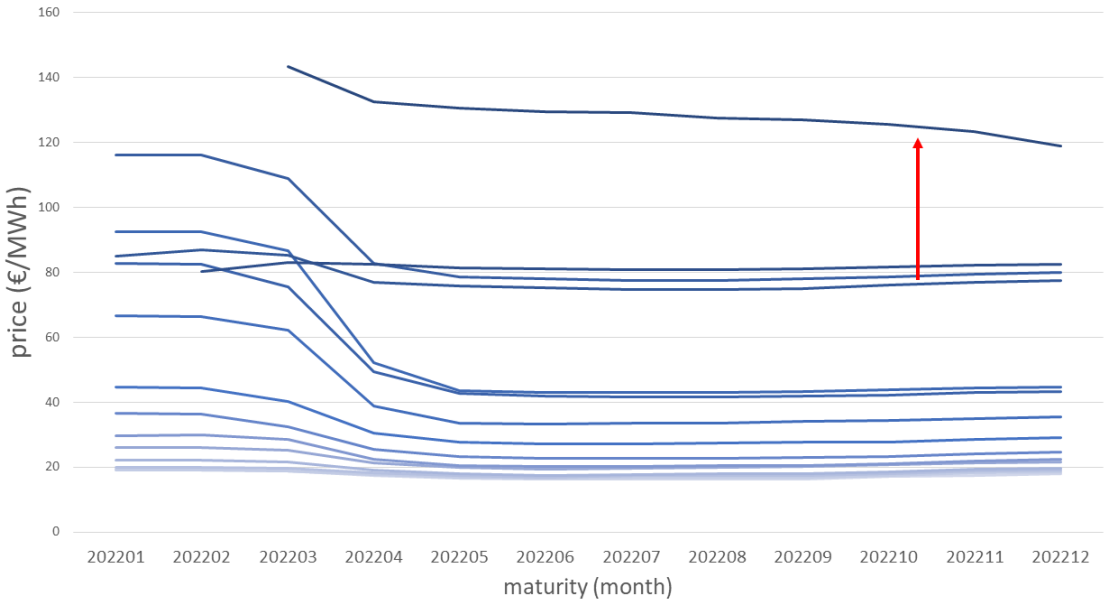


Figure 2.9: The evolution of the PSV forward curve for deliveries in 2022, starting from the trading month January 2021, evaluating the monthly average on the trading dates.

2.2.2 Consider a deterministic seasonal factor

In the energy market, *it is well known ([36], [12]) that consumption as well as production follows seasonal patterns*; so we considered a deterministic seasonal factor. For each underlying, we computed the standard deviation of the logarithmic returns and we collected these results in a matrix

$$\tilde{\Sigma}^k = \left(\tilde{\sigma}^k(M_a, M_b)\right)_{a,b}$$

where k represents the underlying, M_a represents the number of month in the year (January$=1$, February$=2$,..) and M_b represents the time-to-maturity measured in months.

2.2.3 Apply Principal Component Analysis (PCA)

Principal Component Analysis is one of the most widely used Dimensionality Reduction technique which finds directions of maximal variations through a eigendecomposition. So we found eigenvalues (w) and eigenvectors (v) of the covariance matrix S. Given w and v, we sorted them in descending order according to the w values, then we built another matrix by concatenating each eigenvector multiplied by the square root of its corresponding eigenvalue; so we obtained

$$\Sigma^k = \left(v_i^k \sqrt{w_i^k} \right)_i \tag{2.1}$$

where k represents the underlying, w_i^k represents the i-th eigenvalue of S and v_i^k its corresponding eigenvector (a column vector). Instead of taking all the columns of Σ, we chose the first nb columns. This means that we reduced the dimensionality, keeping the most significant factors. The number nb of components depends on the underlying.

2.2.4 Random core: correlated Gaussian variables

The model we used can be written as

$$\begin{cases} dF^k(t,T) & = F^k(t,T)\tilde{\sigma}^k(T,T-t)\sum_i \sigma_i^k(T-t)dW_i^k \\ dW_i^k dW_j^h & = \rho_{ij}^{kh}dt \end{cases} \tag{2.2}$$

where k and h represent underlyings and i and j represent indices of the components (chosen for the PCA); $\tilde{\sigma}^k(T,T-t)$ is the seasonality of the volatility for maturity T (a specific month) and time-to-maturity $T-t$ (a distance in month) and $\sigma_i^k(T-t)$ is the $i-th$ principal component of underlying k for time-to-maturity $T-t$; finally ρ_{ij}^{kh} is the historical correlation between each pair (k,h) of underlyings for each pair (i,j) of components. In the previous subsections we already described how to construct the seasonality ($\tilde{\Sigma} = (\tilde{\sigma})$) and the principal components of covariance ($\Sigma = (\sigma_i)$). Finally we have to compute the historical correlation between each pair (k,h) of underlyings for each pair (i,j) of components. Observe that we denote by W_i^k the Brownian motion for the component i

of the underlying k.
So this is a forward curve model in the sense that it explicitly models all forward prices rather than just spot prices; it follows the Heath-Jarrow-Morton framework, originally designed for forward interest rates, but potentially also working in the energy market.
Warning:

$$R = (\rho_{ij}^{kh})$$

is the correlation matrix, built element by element. While a directly computed correlation matrix is always positive definite, this reconstructed matrix may not be!

Let's consider N diffusion processes with correlated Brownian motions:

$$\begin{cases} dX_i(t) & = a_i(\mathbf{X}, t)dt + \sigma_i(\mathbf{X}, t)dW_i(t) \quad i = 1, ..., N \\ d\mathbf{W}(t) & \sim \mathcal{N}(0, \mathbf{R}dt) \end{cases}$$

$$dW_i(t)dW_j(t) = \rho_{ij}(t)dt$$

where $\mathbf{R} = (\rho_{ij}(t))_{ij}$ is the correlation matrix, as such it is symmetric and positive definite, with $\rho_{ij} \in]-1, 1[$ if $i \neq j$ and $\rho_{ii} = 1$.

You can transform the previous stochastic differential equations in order to obtain independent Brownian motions: you have to decompose **R**. You can use the Cholesky decomposition or the singular value decomposition (SVD).

Cholesky decomposition

Hypothesis on the matrix R for a Cholesky decomposition: Hermitian and positive definite.

Hermitian: a complex square matrix that is equal to its conjugate transpose, $R = R^H$; if R is real, $R = R^H$ implies $R = R^T$, i.e R is symmetric (a square matrix that is equal to its transpose);

Definite positive: a Hermitian matrix R such that $z^*Rz > 0$ each vector $z \neq 0$.

Other definitions:

- Since the eigenvalues of an Hermitian matrix are always real, an Hermitian matrix R is positive definite if and only if all its eigenvalues are strictly positive;
- R Hermitian matrix is positive definite if and only if its leading principal minors (the determinants of its upper-left kxk sub-matrix) are all positive;
- R hermitian matrix is positive definite if and only if the form $< x, y >= y^*Rx$ is an inner product on $\mathbb{C}^n$.

In this case, given **R** a real symmetric positive definite matrix, you can find **L**, a triangular inferior matrix such that

$$\mathbf{R} = \mathbf{L}\mathbf{L}^T$$

So the SDEs become

$$\begin{cases} dX_i(t) & = a_i(\mathbf{X}, t)dt + \sigma_i(\mathbf{X}, t)\mathbf{L}d\mathbf{Z}(t) \quad i = 1, ..., N \\ d\mathbf{Z}(t) & \sim \mathcal{N}(0, Idt) \end{cases}$$

Singular value decomposition (SVD)

Hypothesis on the matrix R for a SVD: the SVD can be applied to any matrix. Given a matrix R, the SVD is $R = U\Sigma V^T$ where

- U is an orthogonal matrix whose columns are the eigenvectors of RR^T;
- V is an orthogonal matrix whose columns are the eigenvectors of R^TR;
- Σ is an all zero matrix except for the first r diagonal elements (singular values σ_i) that are the square roots of the eigenvalues of RR^T (equivalently R^TR).

In the case of a real symmetric matrix, SVD is equivalent (apart from the signs of the vectors) to

Eigen-value decomposition : $R = X\Lambda X^{-1}$, where $X^{-1} = X^T$ (so $U = X$ and $\sigma_i = |\lambda_i|$);

Orthogonal decomposition : $R = PDP^T$, where P is a unitary matrix and D diagonal;

Schur decomposition : $R = QSQ^T$, where Q is a unitary matrix and S a diagonal matrix (if R is not symmetric, S is an upper triangular matrix).

In this case, given $\mathbf{R}$ a real symmetric matrix, you can find $\mathbf{\Gamma}$ (the orthogonal matrix of eigenvectors) and a diagonal matrix D (the corresponding eigenvalues) such that

$$\mathbf{R} = \mathbf{\Gamma} D \mathbf{\Gamma}^T = (\mathbf{\Gamma}\sqrt{D})(\mathbf{\Gamma}\sqrt{D})^T$$

So the SDEs become

$$\begin{cases} d\mathbf{X}(t) & = \mathbf{A}(\mathbf{X}, t)dt + diag(\sigma_i(\mathbf{X}, t))\mathbf{\Gamma}\sqrt{D}d\mathbf{Z}(t) \quad i = 1, ..., N \\ d\mathbf{Z}(t) & \sim \mathcal{N}(0, Idt) \end{cases}$$

So, on one side, the Cholesky decomposition makes the SDE simpler, since you have a triangular matrix; on the other side, SVD requires weaker hypothesis on R, in particular it does not require positive definiteness.

Regarding forward simulations, we compare the error between the historical and the simulated correlation of logarithmic returns of forward price: in the case of Cholesky there is a higher error: the absolute difference between historical and simulated correlation, averaging over different evaluation dates and all the pairs of underlyings, is about 35% with Cholesky and 9% with SVD (see Figure 2.10). On the other hand if you look at the decomposition error, both method perform very well, but Cholesky is better: the average distance (the mean absolute percentage error, averaging over all elements) between the original matrix and the one given by the decomposition is about 10^{-17} with Cholesky and 10^{-16} with SVD (see Figure 2.11). The problem is that Cholesky fails if the matrix is not positive definite.

On both cases there is an arbitrary choice: the column ordering. By changing the column ordering, it changes the decomposition. How this change can affect the

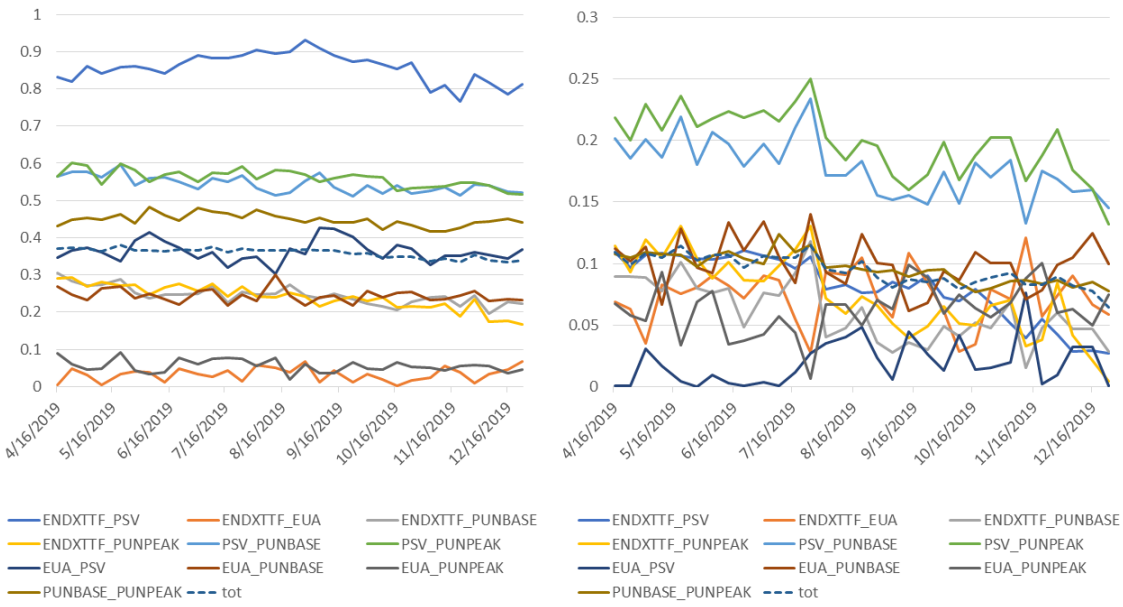


Figure 2.10: Absolute difference between historical and simulated correlation for all the pairs of underlyings for different evaluation dates. On the left: the result using Cholesky decomposition. On the right: using SVD. Here we used three components for the TTF, five for the PSV, one for the EUA, seven for the PUNBASE, seven for the PUNPEAK. We can see that in general this distance is lower for SVD decomposition.

simulations? According to our tests on several valuation dates, a combination appears to have the lowest error by taking the average across all pairs of instruments and five dates: TTF, EUA, PSV, PUNBASE, PUNPEAK. Therefore, bearing in mind that the advantage of SVD is that it does not require the matrix to be positive definite, the two methods perform both rather well in the approximation of the matrix, but not in the same way in the approximation of the historical correlation; as we currently have no theoretical explanation of this latter outcome, it could be subject to further investigation.

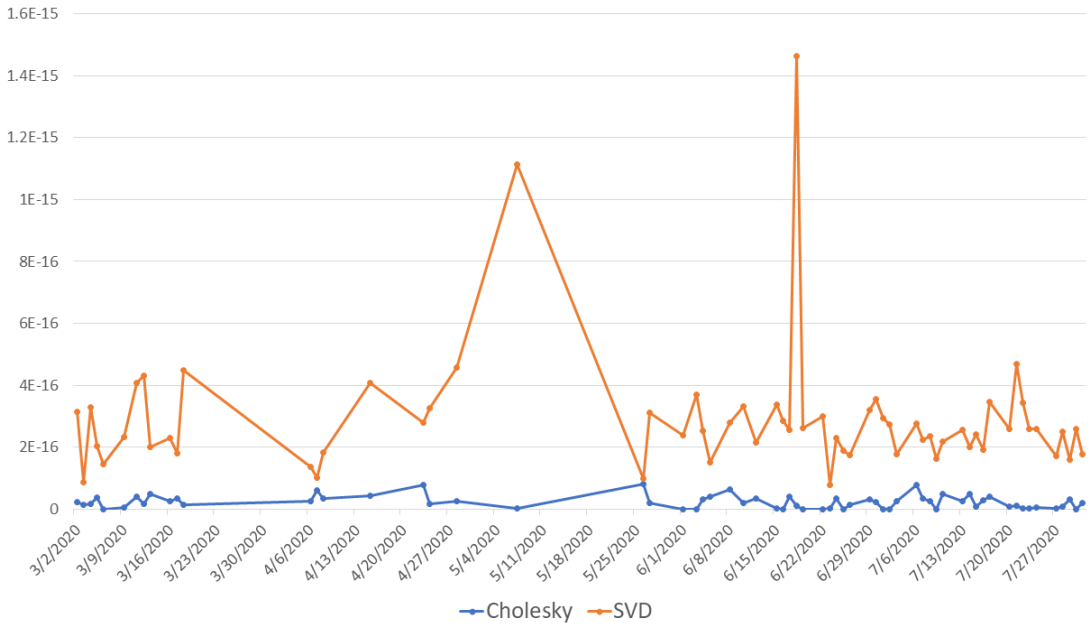


Figure 2.11: Distance between the original matrix and the one given by the decomposition according the two different decomposition methods. The distance is computed as a mean absolute percentage error, averaging over all elements.

2.3 Some issues on the model

The first tests we did were not satisfactory, in particular due to the behaviour of power simulations: it happened too often that we have gotten too large swings and too sharp movements (see Figure 2.12). We stressed the model by different perspectives in order to have more realistic simulations:

- we applied different types of regularization of the term $\tilde{\sigma}$;
- we applied different variations of the PCA algorithm in order to better separate signal from noise;
- we tested different combinations of number of principal component for each underlying.

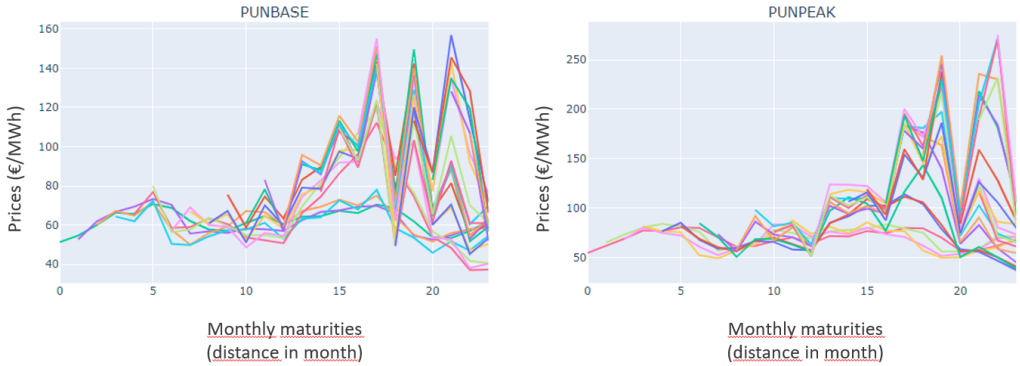


Figure 2.12: An example of bad simulation for PUNBASE and PUNPEAK. On the x-axis the monthly maturities (in number of months with respect to the first trading month), on the y-axis the prices; each different curve refers to a different trading month.

2.3.1 Seasonality

Seasonality is the deterministic part of the simulated forward curves. You might wonder if it would be better to approximate seasonality in order to increase the generalization of the simulation. You might also wonder if including seasonality gives you more realistic simulations or if it's better not to factor a term for seasonality instead.

Regularize seasonality

Nurbs Using the python package `geomdl` (see [38]) devoted to NURBS (Non-Uniform Rational Basis Spline), we approximated the seasonality surface with a NURBS. You have to specify which degree you want to use on each of the two directions of the surface; the degree have to be strictly lower than the number of points in that direction. Since we have twelve points in a directions (months in a year) and twenty-four along the other (the time-to-maturity), we tried all the possible combinations of degree for both the directions in order to find the lower approximation error (see Figure 2.13). The lowest error is for degree equal to one for both directions, while you

found the highest errors for the intermediate values; the behaviour is the same for all the underlyings.

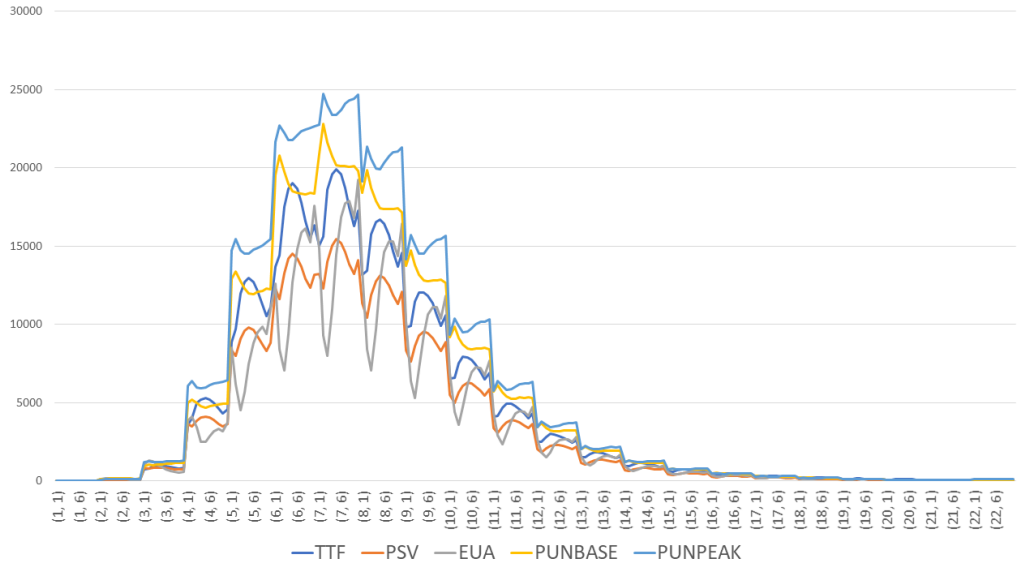


Figure 2.13: Approximation error (computed as mean absolute difference) between the surface describing seasonality and the NURBS approximation for all the underlyings, for all the combinations of degree of both directions.

Fourier Looking at the images of the implied volatility of electricity options (see [36]), one might be tempted to approximate seasonality with a sinusoid or a combination of sinusoid. So

- we found the best approximating plane and remove it;
- then we found the sinusoid components on both directions of the surface (using python package `scipy.fft`), calibrating without using the first $\kappa = 4$ time-to-maturity (because they are more irregular).

No regularization We compared the two cases above with the case with no regularization at all.

For an intuitive idea of the two approximations see Figure 2.14.

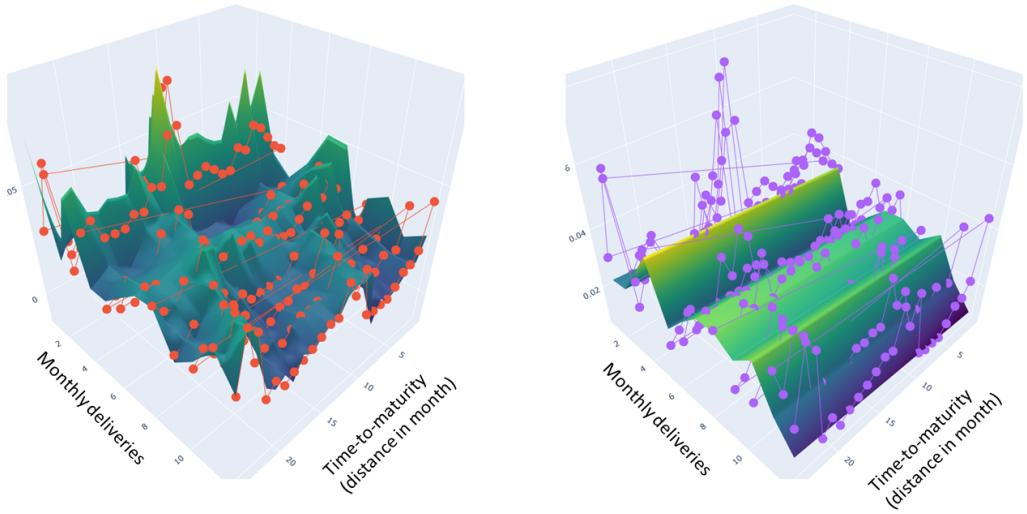


Figure 2.14: A comparison of two different approximation for the deterministic factor for PUNPEAK. On the left: NURBS. On the right: the step-by-step method based on Fourier transform (evaluation date: 19/12/2018).

Looking at the principal components, we found that NURBS stands out of the crowd with wider movements and peaks. We tracked down in the simulations the same unreliable behaviour for the NURBS case. On the other side neither Fourier overcomes the sharp movements for power simulations.

Seasonality: with or without?

Comparing the case with and without the seasonal factor, we came to the conclusion that it is better to include seasonality, because in this way there is a lower error between historical and simulated correlation (see Figure 2.15).

2.3.2 Regularize PCA

Can we distinguish between noise and signal? This is a frequently asked question while addressing PCA-approach. In our application, we are concerned that our historical dataset, especially Italian power prices, is noisy because the market is

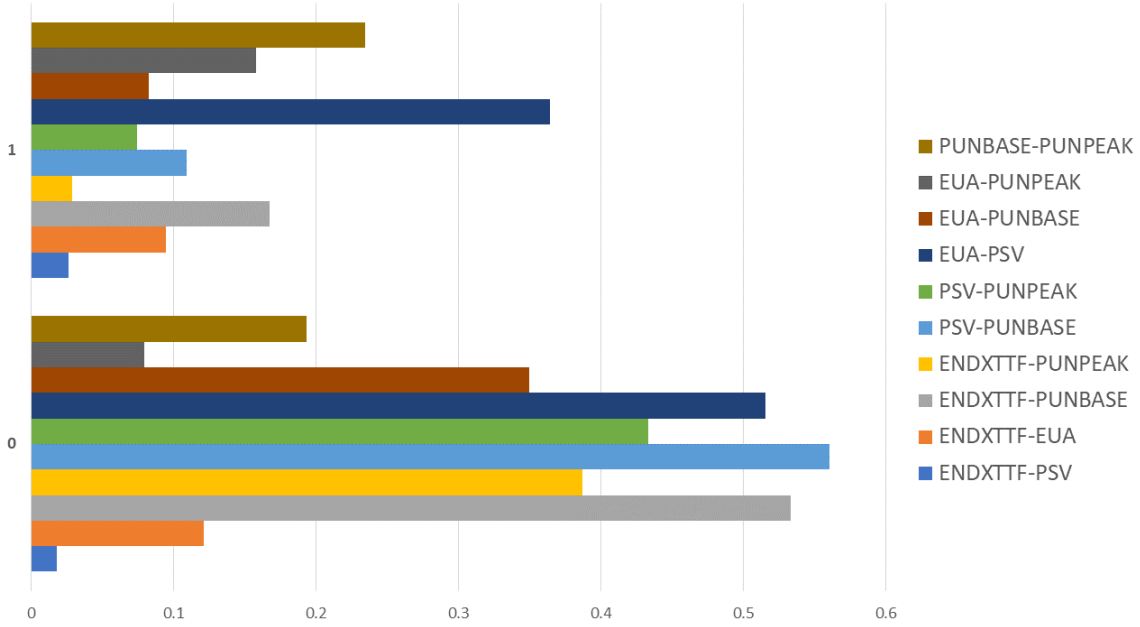


Figure 2.15: The average over five samples dates of the absolute difference between historical and simulated correlation for all the pairs of instruments. The label "0" refers to the case without seasonal factor, "1" with seasonal factor.

not that liquid, so there may be historical forward curves that do not follow a smooth evolution; it is difficult to distinguish these non-fluid movements a priori. So we tested two approach in order to regularize PCA.

Robust PCA PCA is a standard statistical procedure to recover a low-rank matrix from a given data matrix; but PCA is highly sensitive to outliers in the data. According to [39], given a matrix M, with a low rank, we can decompose that matrix, i.e.

$$M = L_0 + N_0$$

where L_0 *has low-rank and* N_0 *is a small perturbation matrix*. This decomposition can help you to distinguish between signal and noise and this might be crucial if you have perturbed measurements; the authors found that the decomposition is solvable and moreover *it can be solved by tractable convex optimization*. Through this algorithm the authors recover the "true" signal

L_0 starting from the perturbed observations given by M. In our case, the perturbation could be attributed to some asynchronous movements between the underlyings of the portfolio; so we recovered L_0 and then we applied PCA to that matrix.

Expectation-maximation PCA There is a paper ([40]) that addresses a similar problem: it supposes to have a *noisy dataset with missing values*. They present a method for performing PCA on this type of unreliable data. As the authors say, *a limitation of classic PCA is that it does not distinguish between variance due to measurement noise and variance due to genuine underlying signal variations*; so they present a method to *directly solve for the PCA eigenvectors with an iterative solution based upon expectation maximization (EM) PCA (EMPCA)*. EM consists in an E-step and a M-step:

E-step: find the expectation of the hidden variables, given the current parameters;

M-step: find the optimal parameters (maximize the likelihood) given the hidden variables.

In this case, *the parameters to solve are the eigenvectors, the latent variables are the coefficients for fitting the data using those eigenvectors*. So we apply this algorithm to our matrices.

No regularization We consider also the case in which we apply directly PCA, without regularization.

In order to compare these three cases we consider the behaviour of the principal components and of the simulations. Since the objective is to improve power simulations, we apply regularization only in the case of power instruments. The principal components of empca are similar to the one with no regularization, apart from the first component: the case of empca shows up and down movements. For both empca and rpca approaches it still appears the behaviour we want to avoid for PUN simulations.

2.3.3 Components of PCA

One of the choice is how many components for each instrument. According to [41] there can be different criteria; for example given a certain level, you can consider the number of components so that the cumulative percentage of the explained variance reaches that level; otherwise you can look for a cutoff point by looking at the eigenvalues in descending order. All these criteria are trying to answer the following question: is variance explained enough? The spectrum of the sample covariance matrix S (Figure 2.16) shows that we would need a huge amount of components especially for PUN, in order to reach a 90%: in particular it would fit three components for TTF, five for PSV, one for EUA, twelve for PUNBASE and thirteen for PUNPEAK.

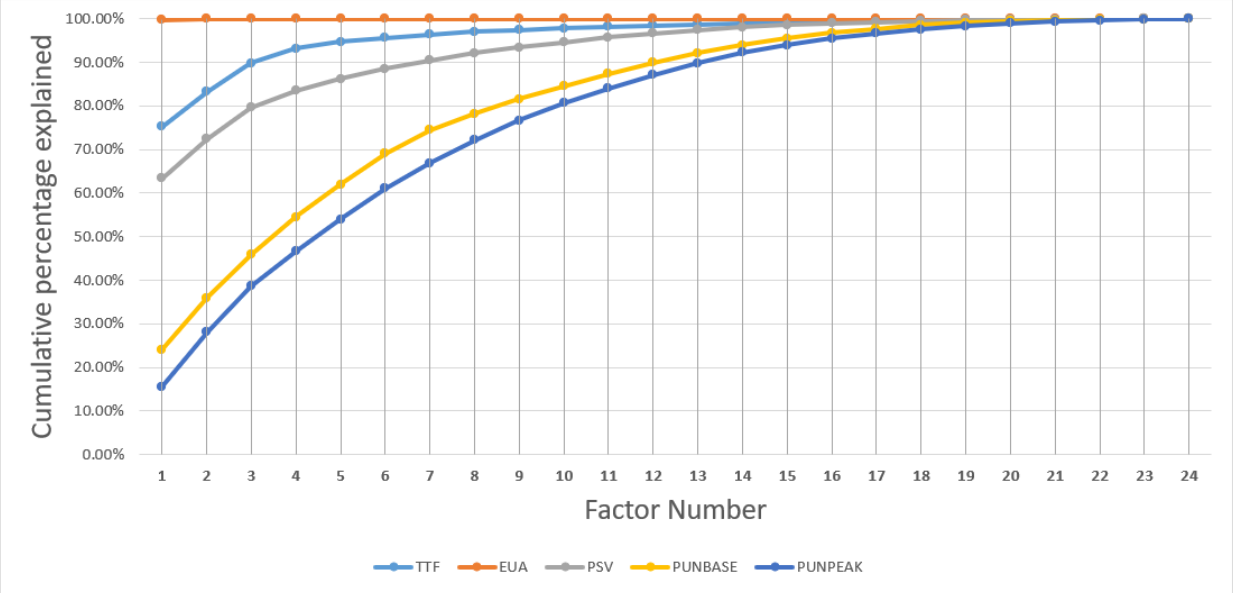


Figure 2.16: The cumulative percentage explained of the spectrum for each instrument; average of 19 different evaluation dates. By looking at this graph, you might say that you need three components for TTF, between five and seven for PSV, one for EUA, twelve for PUNBASE, thirteen for PUNPEAK in order to reach 90%.

We performed a simulation using different combinations of components for

ten different dates between 2019 and 2020. Based on our tests, we have come to these conclusions:

- if we compare the historical correlation with the one of the simulated prices, we saw that there are pairs of instruments easiest to mimic, for example the pair TTF-PSV;

- adding more components causes an increase in the correlation error, especially between PUN and PSV, so in general it is safer to keep low the number of components;

- we found that it is better when PUNBASE and PUNPEAK have the same number of components, otherwise the correlation error for this couple increases.

So, on the basis of our tests, one of the most effective way to reach a good match with reality both looking at the single underlying and looking at the couples of underlyings is to keep low (at most three) the number of components for each underlying and to keep the same number of components for high correlated underlyings.
Taking into account of these considerations, we chose one components for TTF, PSV, EUA and two for PUNBASE and PUNPEAK.
By reducing the number of components we can mitigate the effect of "bad simulation" (see Figure 2.12), i.e. extreme and unrealistic jumps. Another good reason to keep component counts low is the fact that this way we can easily scale to a larger portfolio. We are aware that each component is supposed to catch a particular movement of the forward surface, so we lose some realism by using only one component; in fact, as [12] said, the typical pattern obtained from PCA is the one in Figure 2.17: there are risk factors that act to "shift", "tilt" and "bend", and all these constitute a decomposition for an approximation of the overall volatility.

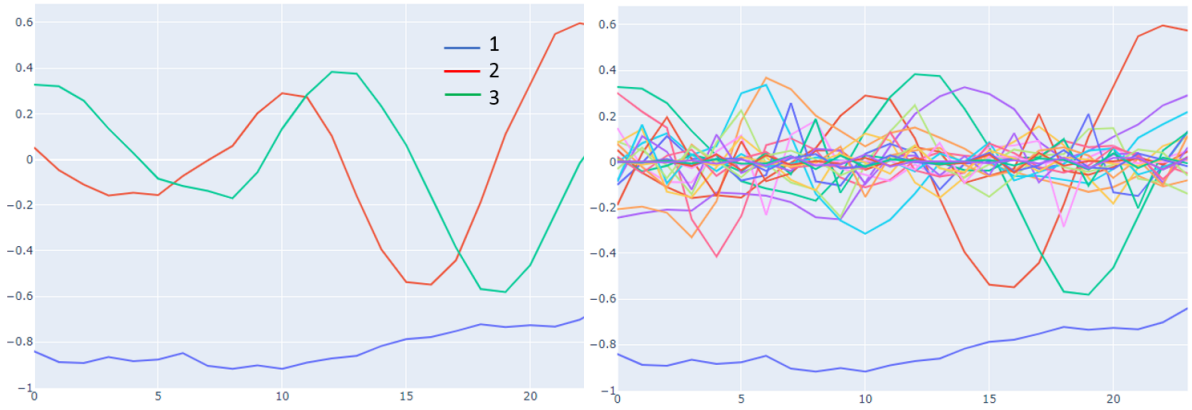


Figure 2.17: Left: the first three principal components for PSV. Right: all the components for PSV.

2.4 Comparison with another model: OU

When we simulate the evolution of a forward curve, we can consider the spot price as a particular point of that evolution, the last one. So we can compare the spot simulations induced by this model with the ones of a spot model, like the Ornstein-Uhlenbeck (OU) model (see [42], [43]), a mean-reverting model often used in the energy market:

$$\begin{cases} dS^k(t) & = \lambda^k(\mu^k - S^k(t))dt + \sigma^k dW^k \\ dW^k dW^h & = \rho^{kh} dt \end{cases}$$

where k and h represent underlyings. We observed the behaviour of this two model in different market situations:

- during the global energy crisis (see [35]) that started in Summer 2021;
- during the pandemic of Covid19 (see [33], [34]);
- before the pandemic.

We saw that since both models rely on the initial forward curve they follow the different market situation in a similar way (see Figure 2.18). They act differently when it comes to spread, i.e. in dealing with correlations (see Figure 2.19).

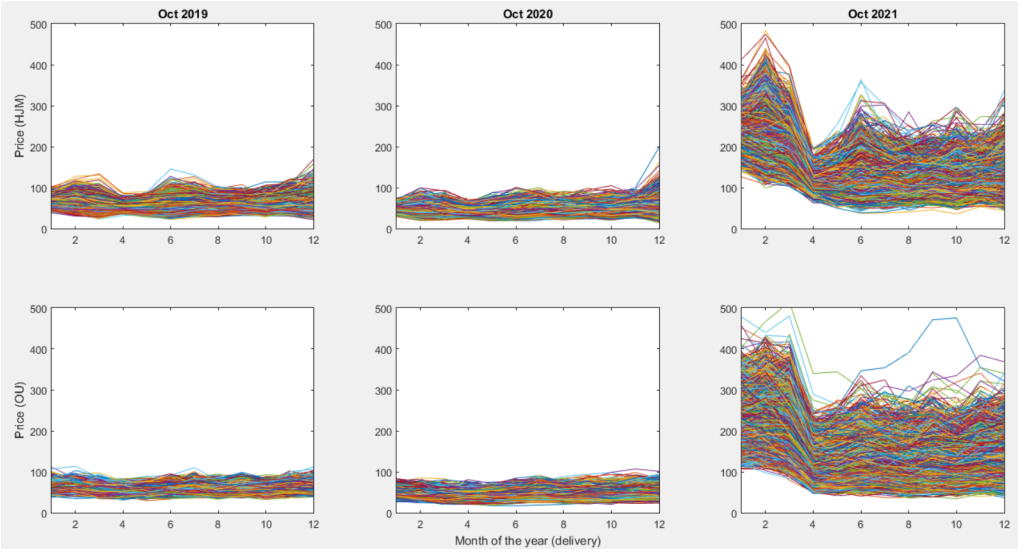


Figure 2.18: A visual comparison of N = 10^4 spot simulations of PUNBASE using our HJM-based model (top) and OU (bottom) model, considering three different year of interests (from the left: 2020, 2021, 2022), starting from the October of the year before. Here you can see that the two model acts in a similar way, the main change is due to a change in the forward market (the third column has higher prices due to the energy crisis started in Summer 2021).

2.5 Some applications

2.5.1 Calibrate to market: pricing

This model offers a closed formula for plain vanillas (see [12]). At date t the European call option price is

$$c(t, F(t,s); K, T, s) = E_t\left[P(t,T)max(0, F(T,s) - K)\right] \tag{2.3}$$

where $P(t,T)$ is the T-maturity discount factor and E_t represents the expectation under the risk neutral measure conditioned to filtration $\mathcal{F}_t$. From equation (2.2)

$$F(t,T) = F(0,T)\cdot exp\left[\sum_{i=1}^{nb}\int_0^t \tilde{\sigma}(T, T-u)du\left(-\frac{1}{2}\int_0^t \sigma_i(T-u)^2 du + \int_0^t \sigma_i(T-u)dz_i(u)\right)\right] \tag{2.4}$$

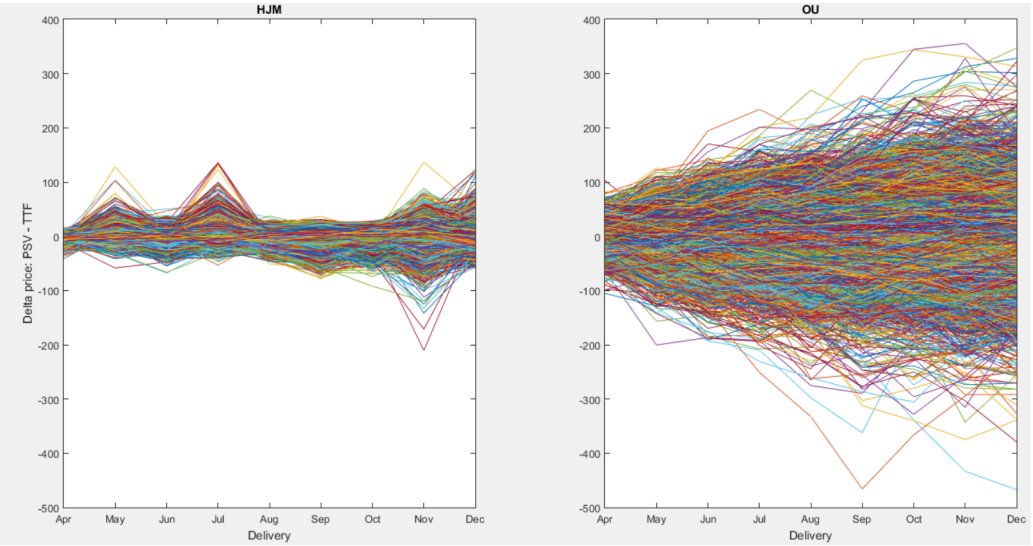


Figure 2.19: A visual comparison of the delta between PSV and TTF for N = 10^4 simulations using our HJM-based model (on the left) and OU (on the right) model, considering the deliveries of 2022 starting from March 2022. Observe that in this case the two model act in a different way, the spread is larger in the OU model, while the actual spread of the settlement price is historically less than 10 euro/MWh.

where nb is the number of components; so the natural logarithms of the forward prices at time T are normally distributed

$$\ln F(T,s) \sim \mathcal{N}\left(m_{\mathcal{N}}, s_{\mathcal{N}}\right) \tag{2.5}$$

$$m_{\mathcal{N}} = \ln F(t,s) - \frac{1}{2}\sum_{i=1}^{nb}\int_0^t \tilde{\sigma}(T,T-u)\sigma_i(T-u)^2 du \tag{2.6}$$

$$s_{\mathcal{N}} = \sum_{i=1}^{nb}\int_0^t \tilde{\sigma}^2(T,T-u)\sigma_i^2(T-u) du \tag{2.7}$$

where $\mathcal{N}(m_{\mathcal{N}}, s_{\mathcal{N}})$ is the normal distribution with mean $m_{\mathcal{N}}$ and standard deviation $s_{\mathcal{N}}$. So the solution of (2.3) is given by

$$c(t,F(t,s);K,T,s) = P(t,T)\left(F(t,s)N(h) - KN(h-\sqrt{w})\right) \tag{2.8}$$

where

$$h = \frac{\ln F(t,s)/K + \frac{1}{2}w}{\sqrt{w}} \qquad w = \sum_{i=1}^{nb} \int_t^T \tilde{\sigma}^2(T, T-u)\sigma_i^2(T-u)du$$

and N is the standard normal cumulative distribution function. Thanks to Formula (2.8) we can modify the columns σ_i of matrix Σ in order to minimize the approximation error between the plain vanillas given by Formula (2.8) and the plain vanillas listed on the market. In this way we can use the simulations of this model to evaluate numerically exotic derivatives.

So we can calibrate to market the columns σ_i of matrix Σ. In order to avoid numerical problems due to the large number of elements to be calibrated, since matrix Σ (see Formula (2.1)) is built from eigenvectors $(v_i)_i$ and eigenvalues $(w_i)_i$, we decided to change only the eigenvalues in the calibration: this allows us to save memory and time, while still having a good market calibration, i.e. a good fit to the prices of quoted plain vanillas (see Figure 2.20).

2.5.2 Calibrate to history: creating realistic scenarios

Market calibration is performed when you want simulated prices to match market information, usually plain vanilla prices; you need this consistency when using these simulations to evaluate an exotic financial product. On the other hand, if only historical information is used to choose the parameters of our model, the most direct way to use these simulations is to create realistic scenarios, for example to have a tail measure of the price distribution.

Having realistic scenarios could be crucial, for example, for the numerical solution of optimization problems or for measuring the risk of a portfolio considering the most extreme scenarios. In fact when you need to evaluate portfolio risk measures, for example PaR[2], usually you need a model that simulate *correlated* spot curves. Moreover, if you are looking at the evolution of a portfolio of future contracts, if your goal is finding the more profitable way to hedge your position while no

[2]PaR stands for Profit-at-Risk; it is the percentile (usually the worst first percentile) of the profit distribution of open positions in a portfolio of futures contracts (see Section 1.1.2).

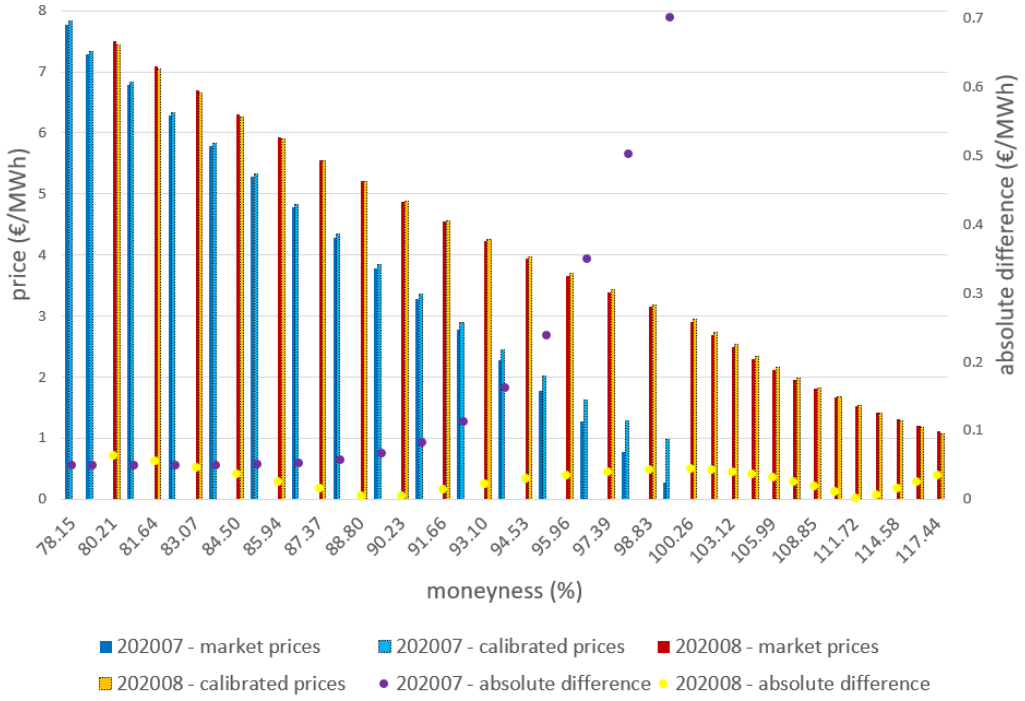


Figure 2.20: A comparison between the original option market price and the approximated one computed using our calibrated model for the Call options of PUN-BASE (see Formula (2.8)) according to the market of 25th June 2020 for different strikes and two different monthly maturities, July 2020 and August 2020.

exceeding a given target risk measure, you need a model that simulate *correlated forward* curves.

2.6 Conclusions

This model offers some pros:

- an effective tool to simulate correlated instruments for the Italian energy market; to the best of our knowledge, there is not much literature for such a tool for the Italian energy market and, more in general, for an entire portfolio of correlated underlyings;

- a closed formula to calibrate to market (useful for pricing); by choosing to change only the eigenvalues in the calibration we reach good performance while saving on computational cost;
- intuitive justifications for the behaviour of simulations (seasonality of the volatility, historical movements captured by the PCA);
- having a tool to simulate correlated instruments is crucial when it comes to portfolio risk measures, because in those cases you have to simulate taking account of the correlated movements of the instruments.

Here we outline some future research directions for our problem.

- We would like simulations that are indistinguishable from the real ones: can we improve them further? For example, we can make some tests by changing the random core of the model, for example the Normal Inverse Gaussian Lévy process (see [44]) instead of the Brownian Motion; on the one hand we would add more realism to the simulations, but on the other hand we would have to face some issue related to keep realism also in the correlated generations and we would have to look for a closed pricing formula in order to calibrate to market.
- There are also other methods for facing robust PCA, for example by considering the problem of robust PCA as a nonconvex optimization problem on the manifold of low-rank matrices (see [45]).
- Currently the literature on forward simulations for the Italian electricity market is scarce, so there are not many benchmarks: can we find a model for a market similar to the Italian electricity market in order to have a comparison?
- How can we further check the goodness of our simulations? Are there any other methods or criteria to measure the quality of the simulations? For example, there are papers (see [46], [47]) which propose some metrics for evaluating the performance of financial time series.

Chapter 3

Scenario optimization

We address portfolio optimization for futures contracts in the energy market. The goal is to decide when to buy the monthly deliveries of a given year for all underlyings in the portfolio so as to maximize profit subject to constraints on risk, position, and liquidity. This can be formulated as a chance constrained optimization problem with probabilistic guarantees on the optimal profit value. A chance constrained feasible solution is obtained using the sampling and discarding approach to scenario optimization, where constraints in probability are replaced with deterministic ones associated to different price realizations and those worsening the profit are discarded. The high memory requirement of the scenario approach implementation is addressed by using a compression scheme.

We consider the problem of setting a portfolio of futures in the energy market. Energy futures are contracts to buy energy products at a predetermined future date and price. They are meant to reduce the exposure of investors to price fluctuations of the underlying assets. Our goal is to best hedge the starting position for all monthly deliveries of a given year, for all underlyings in the portfolio, so as to maximize profit subject to constraints on risk, position, and liquidity.

Since prices for future dates are stochastic, uncertainty must be taken into account when choosing dates and volumes to be traded. This can be done according to three main different paradigms: maximize the profit on average, robustly over all uncertainty instances, or over all uncertainty instances except for a set of pre-

defined probability $\epsilon \in (0, 1)$.

The first paradigm is adopted in the early paper [48] on modern portfolio optimization, addressing a problem where the expected profit is maximized for a given level of risk expressed in terms of profit variance, thus realizing a trade-off between risk and return through the diversification of the portfolio among different assets. Variants of the model in [48] were suggested in the literature, considering the projections of future returns by investors when assessing the expected profit, [49], and accounting only for the negative deviations from the expected return in the risk assessment, see, e.g., [50, 51].

In this chapter, we adopt the third paradigm and formulate the problem as a chance constrained program where we allow for a possible degradation with respect to the target (maximal) profit but only over a set of uncertainty realizations of probability at most equal to a given ϵ. Conservativeness of the worst case approach accounting for all uncertainty instances, even those that are unlikely to occur, is thus avoided, while still providing guarantees on performance. Given that the so-obtained chance constrained optimization problem is non-convex and hard to solve, we propose to adopt the scenario approach [52] to provide a chance constrained feasible, though possibly sub-optimal, solution. To improve performance, we adopt the sampling and discarding scenario method in [53], which entails to first extract N prices realizations ("scenarios") using a suitably designed stochastic model, and then discard the $\kappa < N$ worst ones so as to best improve the profit objective function. If κ and N are suitably chosen according to the desired ϵ, then, the obtained optimal profit value is guaranteed to be the maximum over all price realizations except for a set with probability at most ϵ, with high confidence. By exploiting the fact that stochastic uncertainty enters the scenario optimization program linearly, we are able to cope with the memory requirement growth as a function of the time horizon and number of underlyings, through a compression scheme relying on the convex hull of the prices realizations. Interestingly, the proposed approach can be applied irrespective of the correlation structure of the stochastic prices, which do not need to obey a Markovian model as in those approaches to portfolio optimization based on dynamic programming (see [54] and

the references therein).

The rest of the chapter is structured as follows. The addressed problem was described in Section 1.1 and reformulated here in Section 3.1. After discussing possible solution concepts, we present the proposed scenario-based solution and its sampling and discarding variant in Section 3.2, together with the scenario compression scheme. After discussing how the scenarios are generated in Subsection 3.2.3, we finally report some numerical results in Section 3.3 and draw some conclusions in Section 3.4.

The content of this chapter was accepted at the European Conference on Stochastic Optimization - Computational Management Science (ECSO-CMS) 2022, at the Department of Economics of Ca' Foscari University of Venice.

3.1 Optimization problem formulation

In this section we formulate the addressed portfolio optimization problem.

We start by defining the decision vector X obtained by piling up all vectors $X_{ij}(t_k)$ in (1.10) corresponding to the tradable products for the underlying x_i, the delivery dates d_j and the trading date t_k with $i = 1, \dots, L$, $j = 1, \dots, M$ and $k = 1, \dots, K$.

The non-negativity component-wise constraint $X \geq 0$ jointly with the position and liquidity constraints (1.2) and (1.12) can be expressed compactly as a linear constraint

$$EX \leq F \tag{3.1}$$

for appropriately defined matrices E and F; to take into account the sign (buy or sell) we attribute it to prices. The PaR constraints (1.7) imposed at each trading date t_k can be jointly rewritten as a quadratic constraint

$$X^\top A_1 X + A_2^\top X \leq \beta \tag{3.2}$$

since V_k in (1.7) is a collection of the residual volumes V_{ij}^k which by (1.1) can be expressed as an affine function of $V_{ij}(t_k)$ and (given (1.9) and the definition of X) of X.

As for the profit P to be maximized (cf. (1.3)), it can be rewritten as a linear function of X

$$P = C(\delta)^\top X, \tag{3.3}$$

where $C(\delta)$ is a vector that depends affinely on vector δ whose elements are the forward and spot prices $F_{ij}(t_k)$ and $S_{ij} = F_{ij}(t_k)$, $i = 1, \ldots, L$, $j = 1 \ldots, M$, $k = 1, \ldots, K$.

Our goal is to set the decision vector X so as to maximize the profit (3.3), while satisfying the linear and quadratic constraints (3.1) and (3.2).

Note however that at the trading date when the decision is taken, only the prices of the current forward tradable products and the spot price of the current month are known, which makes the optimization problem affected by uncertainty through the price vector δ.

3.1.1 Chance constrained problem formulation

A possible approach to account for uncertainty is to look for a robust solution that is guaranteed for all uncertainty instances, i.e.:

$$\begin{aligned} &\min_{X,h} h \text{ subject to} \\ &-C(\delta)X \leq h, \forall \delta \in \Delta \\ &(3.1), (3.2) \end{aligned}$$

This requires knowledge of the set Δ where the uncertainty δ takes values and involves solving a semi-infinite optimization program with a finite number of optimization variables and an infinite number of constraints. More importantly, the obtained solution $(X^\star, h^\star)$ is typically conservative, i.e., with a low guaranteed profit $-h^\star$, since all uncertainty instances are accounted for, irrespectively of their likelihood to occur.

If we assume that δ is distributed over Δ according to some probability measure $\mathbb{P}$, we can adopt the so-called chance constrained approach where performance is optimized over a set of of probability at least equal to $1 - \epsilon$ where

$\epsilon \in (0,1)$ is an a-priori specified violation parameter. This allows to mitigate the conservativeness of the robust approach since the δ instances that deteriorate performance but has low probability to occur are neglected when assessing the profit, while at the same time defining the acceptable risk level through the choice of ϵ. The resulting optimization program

$$\begin{aligned} &\min_{X,h} h \text{ subject to} \\ &\mathbb{P}(\delta \in \Delta : -C(\delta)X \leq h) \geq 1-\epsilon \\ &(3.1), (3.2) \end{aligned}$$

is non-convex and hard to solve, in general, and the probability measure $\mathbb{P}$ needs to be known explicitly. This motivates the adoption of the proposed scenario-based solution to chance-constrained optimization.

3.2 The proposed scenario-based solution

Suppose that Δ and $\mathbb{P}$ are not known explicitly but only indirectly thorough independent and identically distributed samples $\{\delta^{(i)}\}_{i=1}^{N}$ (the scenarios) extracted from Δ according to $\mathbb{P}$.

We can then define the scenario program

$$\begin{aligned} &\min_{X,h} h \text{ subject to} \\ &-C(\delta^{(i)})X \leq h,\ i = 1,\dots,N \\ &(3.1), (3.2). \end{aligned} \tag{3.4}$$

This approach has many advantages: the setup is very general, it is computationally tractable, it is a direct approach (it does not attempt to estimate Δ or $\mathbb{P}$). The main problem is how to certify the solution: here the constraint feasibility is addressed empirically, but we want to provide a measure of the probability of violation of the constraint. It can be proved (see [55]) that given $\beta \in (0,1)$, then, with confidence $1-\beta$ it holds that

$$\mathbb{P}(\delta \in \Delta : -C(\delta)X \leq h) \geq 1-\epsilon, \tag{3.5}$$

where

$$\epsilon = 1 - \sqrt[N-d]{\frac{\beta}{\binom{N}{d}}} \tag{3.6}$$

with d denoting the number of decision variables; so we know that given enough simulations (N), fixed a certain degree of uncertainty (β), we can keep the error (ϵ) above a certain level.

In this way, chance-constrained feasibility is recovered.

In our problem, the scenarios are given by *price simulations*. We simulate the prices of the future trading dates for all the delivery dates of the time-horizon of interest; these price simulations represent random extraction of future price scenarios according to a model calibrated on historical prices (see Chapter 2 for more details).

Given a scenario, i.e., the entire forward price evolution for each underlying, each tradable product, each trade date, the problem becomes deterministic; we want to find the optimal combination of volumes to be traded from now up to the end of the year of interest, in order to close the given volume exposures minimizing the worst value of the loss, while satisfying risk and liquidity constraints for each trading date. So we deal with the uncertain part of the problem, i.e. the evolution of the forward prices, by extracting N scenarios and evaluating the maximum level of the loss varying the scenarios.

3.2.1 Sample discarding

Starting from the observation that the extracted scenarios can negatively affect performance, in order to improve performance, while preserving chance-constrained feasibility, we adopt the sampling and discarding approach, [53]. This involves discarding κ over N scenarios so as to best improve performance (making h smaller in our setting).

If κ and N satisfy

$$\binom{\kappa + d - 1}{\kappa} \sum_{i=0}^{\kappa+d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta \tag{3.7}$$

where $\epsilon, \beta \in (0, 1)$ are the violation and confidence parameter and d is the number of optimization variables, then, the chance-constraint (3.5) is satisfied with confidence $1 - \beta$. Observe that using Formula (3.7) with $\kappa = 0$, we have a tighter evaluation of epsilon with respect to Formula (3.6). Instead of solving numerically the Formula (3.7), we use the expression for κ in [53]:

$$\kappa = \left\lfloor \epsilon N - d + 1 - \sqrt{2\epsilon N \ln \frac{(\epsilon N)^{d-1}}{\beta}} \right\rfloor \tag{3.8}$$

where $\lfloor \cdot \rfloor$ determines the maximum integer smaller or equal than the argument.

Figure 3.1 shows an example where performance actually improves significantly: in the figure, the blue dots represent the value of the loss given by $-P$ (changing sign in the (1.3) formula) as the price scenarios vary, the horizontal green line represents the worst loss, while the dashed red line represents the updated worst loss after removing κ worst case scenarios.

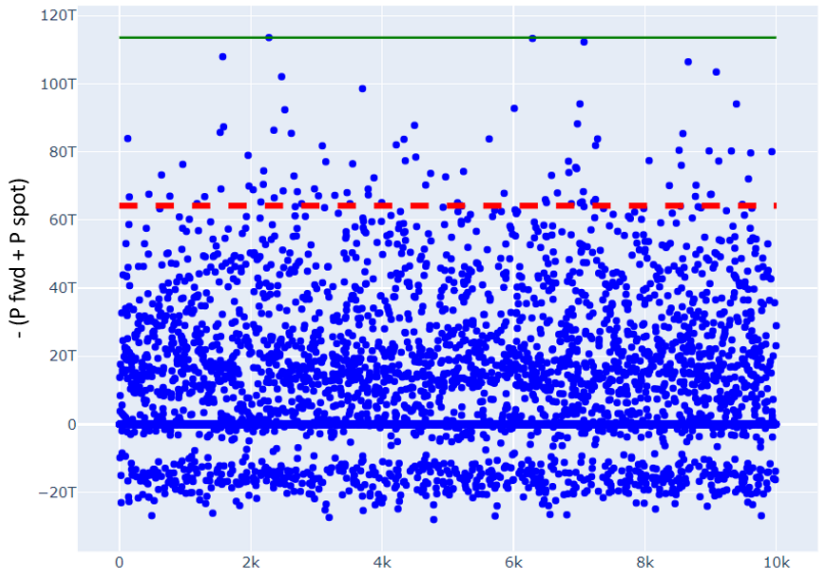


Figure 3.1: An example of applying the sampling and discarding approach that shows a profit improvement by discarding the scenarios that lead to the worst losses; for this test, we have $d = 43$ decision variables, we set $\epsilon = 0.1$, $\beta = 0.0001$, from which we get $N = 10000$, $\kappa = 93$.

3.2.2 Convex hull

As the reference time horizon for profit optimization grows, the number d of decision variables increases and, consequently, the number of scenarios needed to provide a solution with a certain ϵ and β guarantees grows (cf. (3.6)). This has an impact on the memory requirements and computational effort involved when solving the scenario program since each scenario maps into an inequality (see (3.4)). In order to mitigate this problem we can take advantage of the fact that those inequalities are affine in δ and impose the inequalities just for the vertices of the convex hull of all scenarios $\delta^{(i)}$, $i = 1, \ldots, N$, thus reducing the number of constraints.

Given a start date, each price simulation $\delta^{(i)}$ is a vector of fixed length, each element of the vector represents the price of a given underlying for a given product that can be traded on a given trade date; for example if the start date is December before the year of interest and if there are six underlyings, there are 390 decision variables, so each price scenario is a vector of length 390 (see Figure 3.2).

Unfortunately, most algorithms for finding the convex hull consider low dimensional cases (2 or 3 dimensions), motivated by many applications to image processing. Algorithms like QuickHull (see [56]) work in higher dimension, but usually no more than $d = 8$.
In Figure 3.2 we report the total number of decision variables as the trading start date varies: in our case we have up to 648 decision variables (see Figure 3.2), so instead of looking for the exact convex hull we adopt an approximation and use the algorithm in [57], which minimizes the approximation error for a fixed number of vertices and has already been used to reduce the number of constraints in the scenario solution to chance-constrained optimization (cf. [58]).

3.2.3 Scenarios generation

For a deeper description of the model we used for scenario generation see Chapter 2.

When a forward price is simulated, there are three reference dates: the current

date, the trading date and the delivery date.

- **current date:** The simulation is based on the information provided at a given time t_1, so t_1 is the current date.
- **trading date:** The simulation is referred to a market situation in the future, i.e., a market situation of a certain time t_2, with $t_2 > t_1$: usually all the following market situations from t_1 up to the end of the period of interest are simulated; if $t_2 \equiv t_1$, the forward prices of time t_1 are known, there is no need to simulate; we only consider one trade date for each month.
- **delivery date:** Given a certain trading date, the forward prices of a certain delivery t_3 are simulated; note that $t_3 \geq t_2$; usually for each t_2 the prices of several delivery dates t_3 are simulated.

It is necessary to simulate the evolution of correlated forward curves, from the current trading date, for all the following trading dates until the end of the relevant year, for all delivery dates of the relevant year; therefore there is a two-way evolution of prices: on deliveries and on trading dates.

In order to simulate forward prices, we consider the model in [28, Chapter 2]:

$$dF_{i,t_3}(t_2) = F_{i,t_3}(t_2)\tilde{\sigma}^i(t_3, t_3 - t_2) \sum_{z=1}^{nb(i)} \sigma_z^i(t_3 - t_2) dW_z^i$$

where

- i is the index for the underlying,
- z is the index for the component of Principal Component Analysis applied to the sample covariance matrix of historical forward prices; it ranges from 1 to a number $nb(i)$, chosen empirically,
- t_2 represents the trading date,
- t_3 represents the delivery date,
- $\tilde{\sigma}$ represents the seasonal component,

- σ_z represents the component of Principal Component Analysis applied to the sample covariance matrix of historical prices,
- W_z^i are random extraction from a multivariate Gaussian distribution where the standard deviation between the components z of each underlying i is historically estimated.

In practice, we first consider each underlying separately, estimating a seasonal component $\tilde{\sigma}$ of the evolution and then we apply a dimension reduction of the components σ_z to be calibrated via Principal Component Analysis. Then we estimate an historical correlation between the components of each underlying, for all the components of all the underlyings of the portfolio. In this way we have all the ingredients to run the scenarios generator.

Note: in [12, Chapter 8] the reader can also find an easy way to calculate plain vanillas using this model, so there is a way to implement a market calibration, but since we are not using simulations for pricing purposes, we are only using a historical calibration.

3.3 Results

In this section we present some numerical results obtained using the python package *cvxpy* (see [59], [60]), and, in particular, the solver ECOS - Embedded Conic Solver (see [61]): "*The main interior-point algorithm is a standard primal-dual Mehrotra predictor-corrector method with Nesterov-Todd scaling and self-dual embedding, with search directions found via a symmetric indefinite KKT system* [..]").

3.3.1 Comparison with a naive approach

We consider a reference baseline approach to the portfolio optimization problem and compare the results for a backtest for the whole portfolio of 2020 starting from the trading date of December 2019. For a deeper description of the naive approach see also Section 1.2.

The reference baseline approach consists in solving a *deterministic* optimization problem, it does not have uncertain variables: each time t_k the algorithm looks for the actions that keep the portfolio below the limit PaR for time t_k, satisfying liquidity and position constraints for time t_k, finding the best combination between a forward action today and a spot action at delivery, using a *forecast* of the spot price. So the goal is still to minimize the loss given by $P_{fwd} + P_{spot}$, but in this case we do not have price simulations, we just use one curve for current forward prices, $F_{ij}(t_1)$ for $i = 1, \dots, L$ and $j = 1, \dots, M$, and a forecast for spot prices, called view and denoted as View_{ij} for $i = 1, \dots, L$ and $j = 1, \dots, M$.

More formally,

$$\min_{V_{ij}} \sum_{i=1}^{L} \sum_{j:d_j>t_1} \left(F_{ij}(t_1) - B_{ij}\right) V_{ij}(t_1) + \sum_{i=1}^{L} \sum_{j:d_j>t_1} \left(\text{View}_{ij} - B_{ij}\right) \left(\overline{V}_{ij} - V_{ij}(t_1)\right) \text{ subject to}$$

(3.1), (3.2)

The baseline approach does not consider all the intermediate dates, so its forward profit P_{fwd} takes into account only of the current forward prices and the current tradable volumes; for each t_k it considers only the current forward action and the possible spot action, each time looking for the best combination between now and at delivery. On the contrary, the scenario approach, that we have described before, every time tries to find the best combination between now and all the possible times until delivery.

In order to do this backtest, for each trading month from December 2019 up to December 2020, we update all the prices and the residual volume exposures for each couple underlying - delivery of 2020. For example in December 2019, we have

- a forward price for each tradable product (the next year: 2020, the next four quarter: Q_1^{20}, Q_2^{20}, Q_3^{20}, Q_4^{20}, the next four months: January 2020, February 2020, March 2020, April 2020) of each underlying;

- a forecast for the spot price for each monthly delivery of 2020 and each underlying;

- N scenarios for the evolution of the forward price for each tradable product of each underling and each trading date from December 2019 until December 2020.

Then, we adopt both the scenario-based and the baseline approaches to compute the optimal volumes to be traded in December 2019 at the forward prices of December 2019. The next step is to move to the next month, January 2020, and update the residual volume exposure for both approaches; we update all the prices:

- the forward prices for each tradable product of January 2020 (the next quarters: Q_2^{20}, Q_3^{20}, Q_4^{20}, the next months: Febraury 2020, March 2020, April 2020, May 2020) of each underlying;
- a forecast for the spot price for each (remaining) monthly delivery of 2020 and each underlying (the forecast made in January 2020 is different from the one made in December 2019);
- the N scenarios for the evolution of the forward price for each tradable product of each underling and each trading date from January 2020 until December 2020; these scenarios are different from the ones made in December 2019, because each month the price simulator uses new data from the market.

Then we determine with both approaches the optimal volumes to be traded in January 2020 at the forward prices of January 2020. Since the year of interest is started, we can also compute the spot profit of January 2020, i.e. the profit referring to the remaining volumes of the delivery of January 2020 to be closed at the spot price. The next step is to move to the next month, February 2020: here we have to update the residual volume exposures and all the prices. And so on and so forth.

For this backtest, we set $\epsilon = 0.1$, $\beta = 0.0001$, from which we get $N = 10000$, $\kappa = 129$ for each scenario problem; this κ is found starting from the initial hypothesis of κ as in Formula (3.8), then iteratively checking Formula (3.7): if the inequality of Formula (3.7) is satisfied, we keep that value for κ, otherwise we reduce it by one and check the inequality (3.7) again.

In Figure 3.3 we compare the profit of the two components of the formula (1.3) for the two algorithms: the component given by the forward products (P_{fwd}) and the one given by the volumes left at delivery (P_{spot}); you can see that the two methods have similar results in terms of overall profit. In order to compare the two methods, the forward profit P_{fwd} that is shown in Figure 3.3 is the one for the forward price at time t_k, without the intermediate dates, even if the objective function for the scenario-based solution is still given by formula (1.3). The results are close and there is one method systematically lower than the other; but this example suggests that the scenario approach reaches less extreme values.

Figure 3.4 shows that both methods fulfill the PaR constraints (the red line). During the backtest, while the deterministic approach estimates the risk for the single trade date, the scenario approach estimates the risk for the time horizon after a trading date: this is why the deterministic approach has only one point for each trading date (the green points), while the scenario approach has a curve for each trading date (the blue lines with progressive length reduction).

From a practical point of view, the main advantage of the proposed approach is having on overview of all the trading months, even the future ones. This overview might be adjusted over the time, according to a receding horizon approach, using updated price realizations. Changes in suggested actions are expected when there are changes in the market, otherwise if the market remains stable, the output remains the same even after updating the price realizations; in Figure 3.5, for example, there are two different case: we focus on two consecutive trading dates for a backtest in 2020 looking at a single instrument and we find that the output of our optimization algorithm does not change; on the other hand, focusing on two consecutive trading date for a backtest in 2021 and looking at a single instrument, we find that the output changes. In fact, starting from March 2020, because of the pandemic ([34], [33]), prices in the energy market remained low for a few months, while starting from the middle of 2021 the energy crisis ([35]) arrived and prices in the energy market had a huge progressive increase (in Figure 3.6 we report the evolution of the actual forward curves for those instruments in those trading times).

3.4 Remarks on this approach

In this chapter we addressed an optimization problem for a portfolio of forward contracts in the energy market using the scenario optimization technique. The scenario optimization approach is a very intuitive tool that has strong theoretical foundations. Typically in a portfolio optimization problem one aims at maximizing the expected return (based on a certain objective function) while minimizing risk (according to some risk measure); the problem of portfolio selection has been known since [48], but in this case there is the constraint of closing the position within a given delivery, so here the problem consists in choosing the optimal sequence of fractions of the volume exposure, in order to choose the best prices, while satisfying liquidity and risk constraints. By the proposed scenario approach we can make a plan for the entire reference period and adjust it at each trading date based on the possible realizations of all the intermediate prices (including the spot) and not just on a forecasting of the spot. This allow a better tuning to the actual uncertainty entering the system.

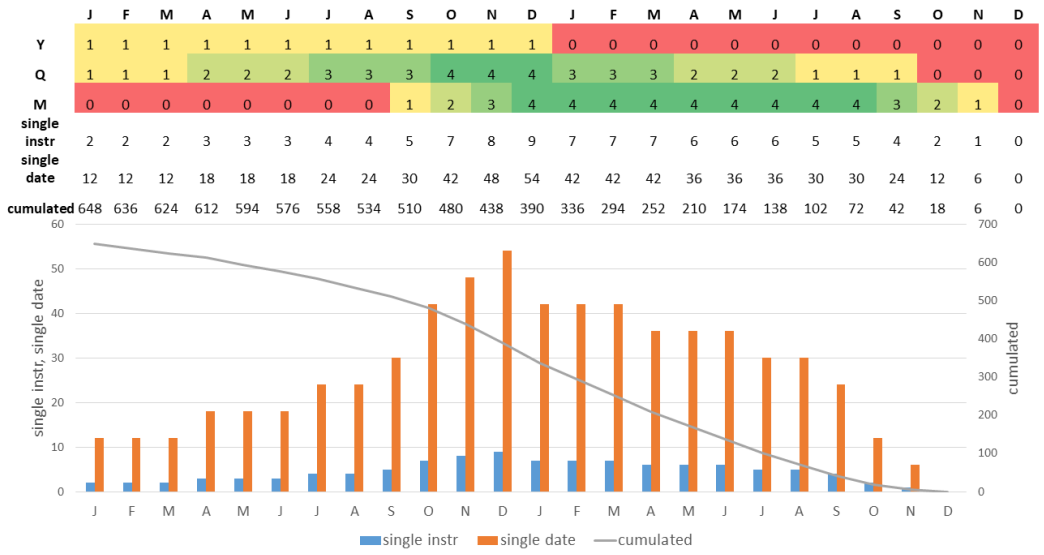


Figure 3.2: For each trading date, starting from January of the year preceding the year of interest we count the annual, quarterly and monthly tradable products for a single instrument: the annual product Y is one before the year of interest, otherwise it is zero; the number of quarterly products Q and the number of monthly products M depend on the distance between the trade date and the year of interest, since the only tradable quarters are the following four quarters and the only tradable months are the following four months, provided that they belong to the year of interest, so in general Q and M are between zero and four. By adding the rows Y, Q and M we find the number of tradable products for a single instrument for a single trading date (the row *single instr*, i.e. the blue columns); multiplying the count of a single instrument by six we find the number of tradable products for the whole portfolio for a single trading date (the row *single date*, i.e. the orange columns); summing up all tradable products (i.e. decision variables) from a given trading date to the end of the period of interest we find the *cumulated* gray curve.

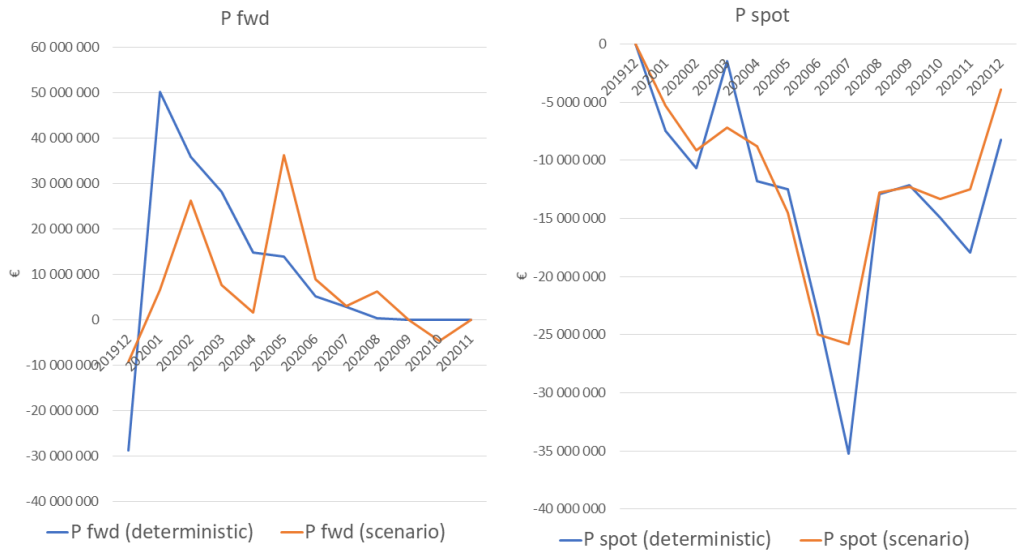


Figure 3.3: Comparison of the two methods: the forward and spot profit for each trading month between December 2019 and December 2020 for the year of interest 2020. Note that the scenario approach suggests what to trade on the current date and also provides an overview of what to trade on all upcoming trading dates, while the deterministic approach provides quantities to be exchanged only for the current date. Here we only consider the realized PL, the one given by the current date: therefore for each trading date of the backtest, we estimate the realized forward PL, i.e. the one given by the forward contracts of the current trading date, and we evaluate the realized spot PL, i.e. the one given by the residual volume exposures for monthly delivery which coincides with the current trading date.

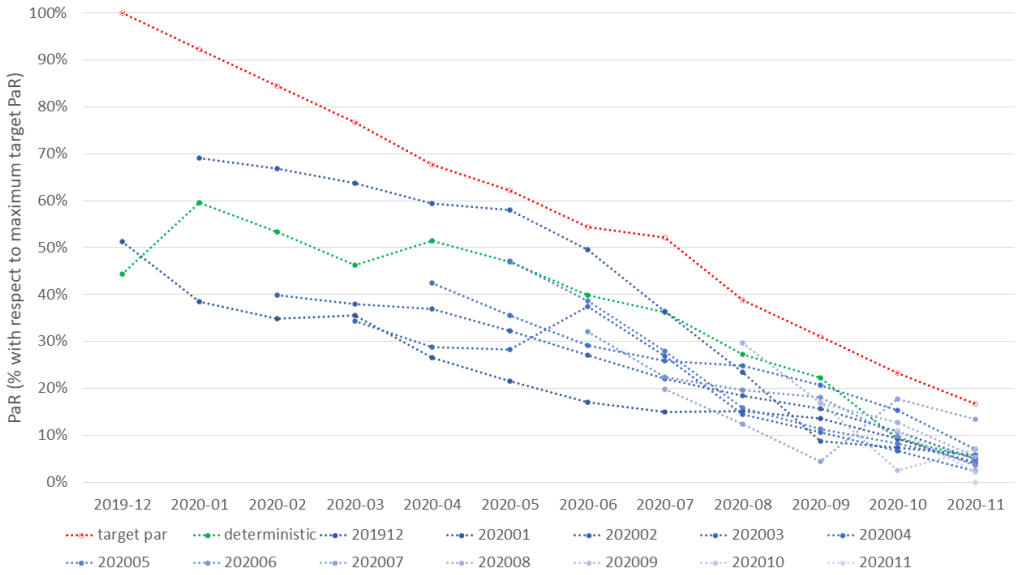


Figure 3.4: Comparison of the two methods: the PaR for each trading month between December 2019 and December 2020 for the year of interest 2020, reported as a percentage of the maximum target PaR. The red line is the constraint, the green line represents the PaR values of the baseline approach, each blue line represents the PaR value from a different starting month according to the scenario approach.

TRADABLE PRODUCTS	PSV	TRADING DATES Jul-20	Aug-20	Sep-20	Oct-20	Nov-20
	Aug-20	0%				
	Sep-20	0%	0%			
	Oct-20	0%	0%	0%		
	Nov-20		0%	0%	0%	
	Dec-20			0%	0%	0%
	Q_3^{20}	0%		0%		
	Q_4^{20}	0%	0%	0%		

TRADABLE PRODUCTS	PUNPEAK	TRADING DATES May-21	Jun-21	Jul-21	Aug-21	Sep-21	Oct-21	Nov-21
	Jun-21	-47.1%						
	Jul-21	0.0%	0.0%					
	Aug-21	0.0%	0.0%	0.0%				
	Sep-21	0.0%	0.0%	0.0%	0.0%			
	Oct-21		0.0%	0.0%	0.0%	0.0%		
	Nov-21			0.0%	0.0%	0.0%	0.0%	
	Dec-21				0.0%	0.0%	0.0%	0.0%
	Q_3^{21}	-2.8%	-2.8%					
	Q_4^{21}	-0.5%	-0.5%	-6.9%	-6.9%	-6.9%		

Figure 3.5: Left: the percentage difference over time of products of PSV to be traded according to the optimization algorithm for two consecutive trading months, June 2020 and July 2020. Right: the percentage difference over time of products of PUNPEAK to be traded according to the optimization algorithm for two consecutive trading months, April 2021 and May 2021. The tables are not full because we only consider tradable products. PSV is an Italian acronym that stands for Virtual Exchange Point and is the price of Italian petrol. PUN is an Italian acronym which stands for Prezzo Unico Nazionale (i.e. the price of Italian power); PEAK refers to the fact that it is the average price in the ”peak” hours of each day of the period in question, i.e. from 8:00 to 20:00 on working days.

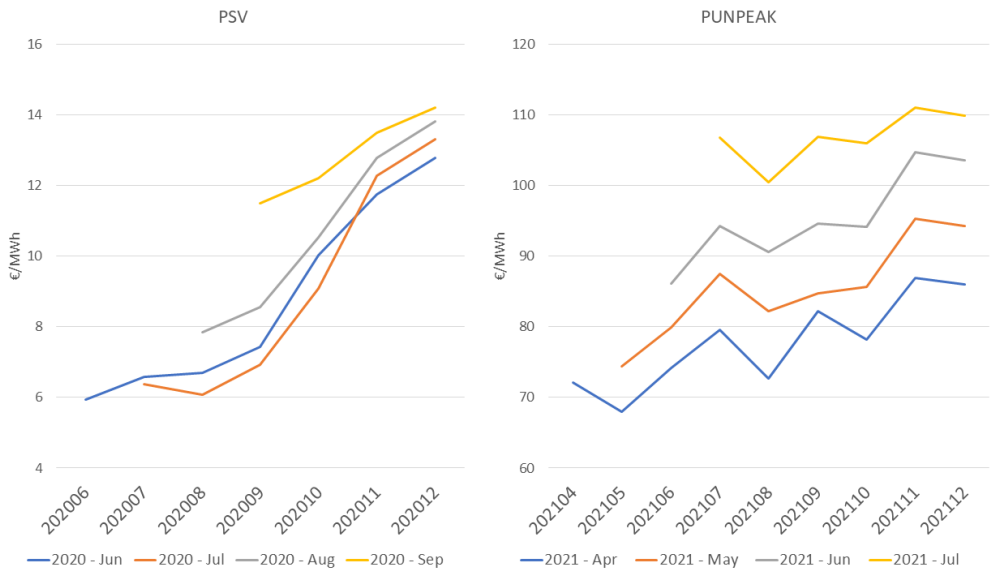


Figure 3.6: On the left: the forward curve for PSV with trade date spanning from June 2020 and September 2020 and delivery dates the monthly deliveries of 2020. On the right: the forward curve for PUNPEAK with trade date spanning from April 2021 and July 2021 and delivery dates the monthly deliveries of 2021.

Chapter 4

Reinforcement learning

Artificial Intelligence (AI) is the art of imitating human intelligence; a particular subset of AI is Machine Learning, the science that ”teaches” a computer a specific task through a *training* that calibrates certain parameters. Not many decades ago, Artificial Intelligence seemed to be in a dead end: now thanks to greater data availability, more hardware power, and new advances on the algorithmic side, it is an active research field.

Machine Learning usually falls into these branches: Supervised Learning, Unsupervised Learning, Reinforcement Learning.
We talk about Supervised Learning when you provide to the computer a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1..n}$ where each couple (x_i, y_i) represents an example of input and output; the goal is learning the relationship between x and y through samples, so that when a new $\tilde{x}$ arrives, the machine can suggest the correct $\tilde{y}$.
On the other hand, it is called Unsupervised Learning when the dataset of inputs is not labelled with the corresponding outputs; the machine has to learn a pattern from the given dataset $\mathcal{D} = \{x_i\}_{i=1..n}$. Note that according to the Unsupervised Learning approach we lack an external supervisor and this may be the case in some problems.
Finally, Reinforcement Learning (see [62]) starts from the idea that the most natural way to learn is by interacting with our environment; we want to learn a function

π that maps situations to actions, so to optimize a given function R. Function π is called *policy* and function R is called *reward*. So Reinforcement Learning is different from Supervised Learning because we do not have representative examples of the correct behaviour in order to learn the policy. At the same time Reinforcement Learning is different from Unsupervised Learning because we are trying to optimize a reward by learning the optimal policy, instead of trying to find some hidden structure.

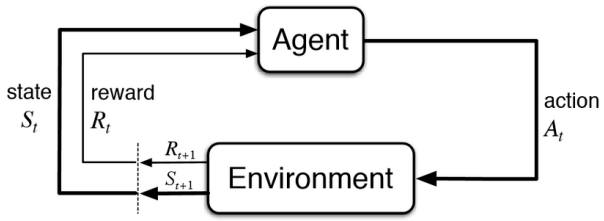


Figure 4.1: Image taken from [62]: the agent receives the state S_t from the environment and choose the action A_t; the environment changes the state into S_{t+1} and update the reward R_{t+1}.

Reinforcement Learning is a useful tool to represent and solve sequential decision-making processes: there is an *agent*, a decision-making entity, that has to choose an *action* according to the current *state* of the *environment* (everything outside of the agent) receiving a *reward*. We talk about *Markov Decision Process* (MDP) (see [63]) every time the next state s_{t+1} and reward r_{t+1} depend only on the current state s_t and action a_t, i.e.

$$P(s_{t+1}, r_{t+1}|s_1, a_1, r_1, ...s_t, a_t, r_t) = P(s_{t+1}, r_{t+1}|s_t, a_t).$$

So in order to define a MDP you need a state space $\mathcal{S}$, an action space $\mathcal{A}$ a reward function r, where

$$r : \mathcal{S} \text{ x } \mathcal{A} \to \mathbb{R}$$

and a transition operator that models the dynamics of the MDP; knowing this operator means knowing the probability distribution of the next state s_{t+1}, given the

current state s_t and action a_t:

$$P_{i,j,k} = P(s_{t+1} = i | s_t = j, a_t = k).$$

Some Reinforcement Learning systems are model-free, others are model-based. Using a model-based approach means knowing completely the behaviour of the environment; in this way you can plan by predicting the next state and the next reward given the current state and the action. So using a model-based approach means knowing the *transition probabilities*, which is generally quite unrealistic. On the opposite side, a model-free method is a kind of trial-and-error learner. We will use the model-free approach.

There is also another important distinction: whether the horizon is finite or not. When the horizon is finite, the *return* R is simply the sum of the total reward over all times t up to the final one T; when the horizon is not finite, the *return* is defined as a series $\sum_t r(s_t)\gamma^t$ where $0 < \gamma < 1$ so that the series is convergent, given that exists an upper bound independent from t for the single reward. In our problem the horizon is finite.

For each state s there is also a *value function*, $V^\pi(s)$, that measures the total amount of reward to be expected starting from the state s according to a given policy π, i.e. $E_\pi[R|s_t = s]$ where E_π is the expectation according to policy π; we have to measure this value in order to take action, but determine values is harder than determine rewards: the central role of *value estimation* is easily understood and there are many reinforcement learning algorithms driven by the value function. Another important function is the action-value function, $Q^\pi(s, a)$ that is the total amount of reward to be expected starting from the state s selecting action a and following policy π, i.e. $E_\pi[R|s_t = s, a_t = a]$. For each state s we can define the optimal value function $V^*(s)$ as the maximum $V^\pi(s)$ over all the policies π; in the same way, for each couple state s and action a we can define the optimal value-action function $Q^*(s, a)$ as the maximum $Q^\pi(s, a)$ over all the policies π. The optimal policy $\pi^*(s, a)$ is the one that realizes the optimal $Q^*(s, a)$.

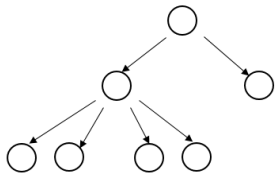
There is an important link between reinforcement learning and *Dynamic Program-*

ming ([64]): the optimal value function can be found using the *Bellman equation*:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s_i \in \mathcal{S}} P_{s_i,s,a} \left(E[r_{t+1} | s_{t+1} = s_i, s_t = s, a_t = a] + V^*(i) \right).$$

The dynamic programming approach involves the transition probabilities, it is model-based, while here we use a model-free approach.

4.1 Monte Carlo Tree Search (MCTS)



Let's consider a game represented by a tree, where

- each node represents a game status
- each arrow represents a possible game action.

At the end of the game there is a reward: we want to find the best policy, which is a function that given a game state can show us the best game action in order to increase the chances of having the highest reward.

A **brute force** strategy - systematically trying all possible actions - would immediately run into memory problems. Furthermore, the possible actions could be infinite or they could be chosen in a **continuous** space: even with huge memory resources a brute force strategy would be impossible. Also there could be **uncertainty** about the next state, given the current state and the action: this could happen due to an opponent or because there are some uncontrollable variables (e.g. I want to find the fastest route in a city, but I don't know if the traffic light will favor me). One game that shows uncertainty and one of the more stylized but yet effective instance of this framework is the *multi-armed bandit* (see [65], [66]): a K-armed bandit problem is defined by K gambling machines; if you play machine i n times

you will obtain a sequence of rewards independent and identically distributed according to an unknown law; the reward is immediate but stochastic and a policy suggests you the next machine to play balancing the knowledge from the previous plays (the average return obtained from each machine) and the desire to explore enough the potential of each machine.
An approach that deals with this kind of games is the Monte Carlo Tree Search (see [67], [68], [69]), based on sampling and selective iterative search.
Before of MCTS there was the simple *Monte Carlo* approach ([70]) to deal with a game with a huge branching factor like Go: the idea is to play a random game starting from the current position and evaluate the final outcome; repeat N times and compute a mean over the N simulations in order to have an evaluation of the expected final value from the current position. It was proved in [70] that this approach is winning against a beginner even with very few simulations. The problem of this Monte Carlo approach is that after a certain number of simulations the search reaches a plateau and it does not improve.
Then it was proposed in [68] the MCTS approach, a combination between tree search and Monte Carlo: this framework saves memory by sharing information between one move and the others and it introduces a selectivity that allows to devote more time to most promising nodes, reducing the probability of exploring the less promising ones, without letting this probability to zero.
The research interest of this algorithm is due to the success that has been proved with playing Go ([71]) and the fact that there are many potential applications, whenever there is a problem that can be described in terms of state and action and there is the possibility of extracting scenarios to simulate episodes.

4.1.1 Monte Carlo Tree Search: pseudo code

MCTS episode

1. Initialize time, t=0;
2. Initialize state, s_t;
3. MCTS step: find the best action a_t;

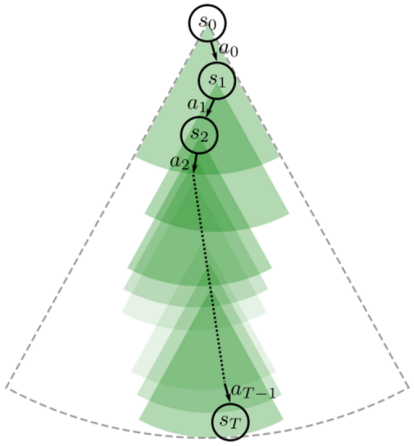


Figure 4.2: An imagine that gives an idea of MCTS episode: for each move there is a partial selective exploration (the cones have a limited depth); the information is shared between one move and the others (the cones are overlapping); the approach concentrates its efforts on the most promising nodes (the tree is not symmetrical).

4. $s_{t+1} = \tau(s_t, a_t)$, where τ is the transition function;

5. Update $t \mapsto t + 1$ and repeat from point 2 until the end of the game.

MCTS step

selection Start from current state and traverse the (already stored and possibly empty) tree by following the selection policy until a leaf node is encountered.

expansion Add the leaf node to the tree.

simulation + backpropagation Play a simulated game (according to the simulation policy) and backpropagate the result. Repeat n times.

After repeating m times, each time adding a node, find the best action according to the output policy.

4.1.2 Monte Carlo Tree Search: policies

In order to concentrate the efforts on the most promising nodes, we want to find the right balance between *exploration* and *exploitation*: we do not want to miss good alternatives, but we also want a criterion to remove suboptimal solutions. This balance is performed thanks to a selection policy called UCT (see [65]), that stands for Upper Confidence Bounds (UCB, see [66]) applied to Trees.

selection policy

$$\pi^{UCT}(s) = \underset{a \in \mathcal{A}(s)}{\text{argmax}} \left(\hat{Q}(s,a) + c(s) \sqrt{\frac{2\log(N(s))}{N(s,a)}} \right)$$

Here s represents a state, a an action, $N(s)$ it the number of transitions already performed from state s, $N(s,a)$ is the number of already performed selections of the action a starting from state s, $\hat{Q}(a,s)$ is the estimate of the outcome of action a starting from state s, $c(s)$ is a given function that controls the degree of exploration, $\mathcal{A}(s)$ is the action space in state s. Every time we visit a state s and choose an action a, we update all these values. This selection policy shows the two side of exploration-exploitation dilemma: if $\hat{Q}(a,s)$ prevails, we exploit the value given by the past evaluations, on the other side if the number of visit $N(s)$ is small enough with respect to the number $N(s,a)$ of actions from the state s, then exploration prevails.

The simulation policy is driven by a uniform probability distribution:

simulation policy

$$\pi^{sym}(s) = a \text{ with } P(s,a) = \frac{1}{|\mathcal{A}(s)|}$$

So we randomly choose in the space of admissible actions for the state s.

The output policy is a greedy policy, because it simply looks for the action $a \in \mathcal{A}(s)$ that has been chosen the most starting from state s:

output policy

$$\pi^{gre}(s) = \underset{a \in \mathcal{A}(s)}{\text{argmax}}\, N(s,a) \overset{m \to +\infty}{\longrightarrow} \underset{a \in \mathcal{A}(s)}{\text{argmax}}\, \hat{Q}(s,a)$$

So it is an empirical way of finding the best action. When the number m of repetitions of the procedure selection-expansion-simulation-backpropagation is big enough, the action that maximizes $N(s,a)$ should coincide with the one that maximize $\hat{Q}(s,a)$.

One drawbacks of the MCTS is the fact that the knowledge is shared between one move and another, but not between MC episodes; moreover there is no overall policy found, so you need something that saves the policy that you learnt during the MCTS. MCTS is inherently stochastic, instead we would like a deterministic policy to be able to measure performance.

4.2 Neural MCTS

So the biggest problem with MCTS is that no general policy is found because there is no knowledge transfer between episodes. We can use a deep neural network to allow for the transfer of knowledge between episodes of MC, so that a general policy can be found, through the **reinforcement learning** mechanism: this approach is called Neural MCTS ([71], [72]) and it is at the basis of AlphaZero algorithm by David Silver.
One of the main novelty introduced by this algorithm with respect to other algorithms that try to compete in strategic games is the fact that there is no need to insert domain-specific heuristics, it learns by self-play; in this sense it is general-purpose.

4.2.1 Neural MCTS: pseudo code

Note that here we use the tilde and the hat to distinguish between neural network outputs and empirical estimates, respectively.

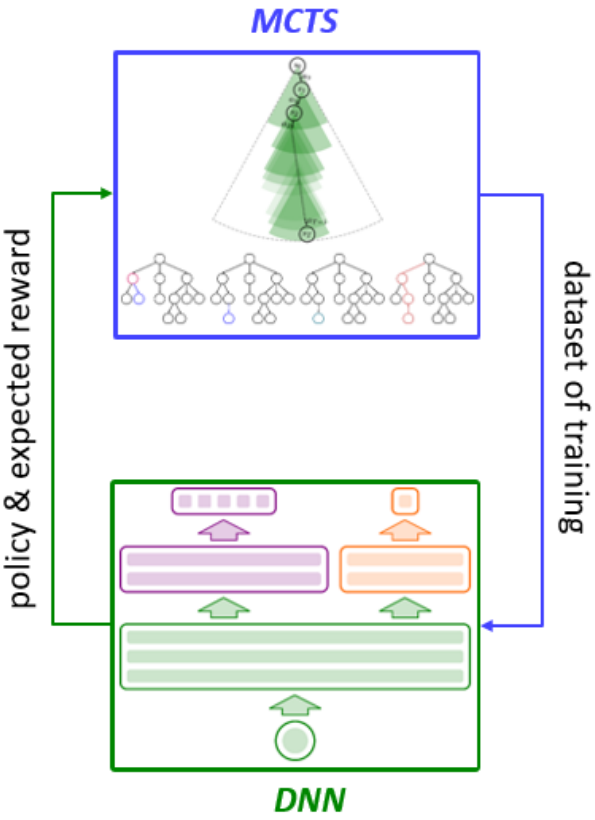


Figure 4.3: This image gives an idea of how Neural MCTS works. The dataset of training is given by episodes played with MCTS. Then there is a Deep Neural Networks (DNN) that is twofold: it learns the function that, given a state, maps an action into a probability; moreover it learns an esteem of the final expected reward, given a state. This outcomes are used for building the next set of training through other MCTS episodes.

Iteration: (repeat k times): play one MCTS episode $\mathcal{E}_j$ and collect data items $\mathcal{D}^{\mathcal{E}_j}$. Then train DNN with dataset $\mathcal{D} = \bigcup_{j=1}^{k} \mathcal{D}^{\mathcal{E}_j}$.

MCTS episode: play a whole game expanding the tree through MCTS steps; obtain $\mathcal{D}^{\mathcal{E}} := \left\{ s_i, \hat{P}(s_i), \hat{V}^{\mathcal{E}} \right\}$, where s_i represents all the states of the tree encountered during the MCTS episode, $\hat{P}(s_i)$ are the correspondent empir-

ical probabilities and $\hat{V}^{\mathcal{E}}$ the total empirical reward of the episode.

MCTS step : see Figure 4.4. This MCTS step is similar to the one of the previous paragraph, but it involves a neural network which has a Y shape: it takes a state s as input, processes the state through a common body that becomes twofold. From one side there is $\tilde{P}$, that allows you to compute the vector of the probabilities of all actions given the state s; this vector is used to select the best action. From the other side there is $\tilde{V}$, the value that anticipates the final reward and it allows you to avoid the simulating step.

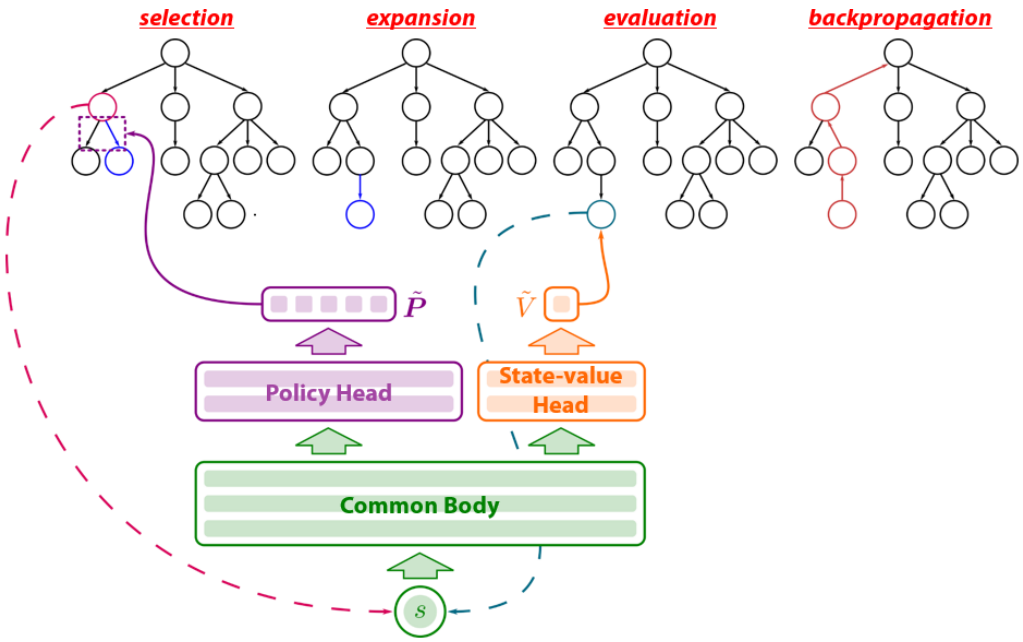


Figure 4.4: MCTS step

4.2.2 Neural MTCS: policies

The main difference with respect to the MCTS is the fact that the deep neural network is present in the policies, so it is like a self-powering mechanism: the episodes played with MCTS constitute the dataset for the training of the neural

network; at the same time, the MCTS is played using the neural network in some specific points. At the beginning, the parameter of the neural network are initialized randomly. The selection policy, called PUCT (where P stands for predictor), is a balance between exploitation and exploration, as in the MCTS case:

selection policy

$$\pi^{PUCT}(s) = \underset{a \in \mathcal{A}(s)}{\text{argmax}} \left(\hat{Q}(s,a) + c(s)\tilde{P}(a|s)\frac{\sqrt{N(s)}}{N(s,a)+1} \right)$$

Also in this case the exploitation part is given by $\hat{Q}(s,a)$, the empirical mean of the value of the next states starting from state s and choosing action a. Also in this case the exploration part depends on the number $N(s)$ of visit of state s and on the number $N(s,a)$ of the times we choose action a starting from state s. But there is another term: $\tilde{P}(a|s)$, the probability of choosing action a given state s according to the optimal policy trained by the neural network; this term helps to balance the exploration part.

There is no simulation policy:

simulation policy it does not exist.

In place of simulation we evaluate using $\tilde{V}(s)$, the deep neural network value that anticipates the final outcome of the reward of the MCTS episode given a state s.

Finally the output policy:

output policy

$$\pi^{out}(s) = \underset{a \in \mathcal{A}(s)}{\text{argmax}}\, \hat{P}(s,a) \quad \text{with} \quad \hat{P}(s,a) = \frac{N(s,a)}{N(s)}$$

So we simply use the choice frequency of action a as an empirical approximation for the probability that a gives the best result.

4.3 Neural MTCS in continuous spaces

What if action space $\mathcal{A}$ is continuous? Having a continuous space necessitates some changes to the algorithm in several places.
First of all, if the space is continuous, equivalently if the space is discrete but of high dimension, we cannot explore all the actions of a given state, it is numerically impossible; we need a method for exploring space. Moreover instead of having the list of all possible actions and their correspondent probabilities, we have a probability distribution; but it is an empirical one, we do not have the functional formula. So we need a way to sample from an empirical probability distribution. Finally we would like to pick the action with the highest probability; in the discrete case this is simple, because we can list all the possible actions, compute their probability and choose the action with the maximum probability: how should we do when the action space is continuous?

4.3.1 Progessive widening

How to numerically explore a continuous space? We choose a number κ of initial actions by sampling the probability $\tilde{P}(a|s)$. Then we use **progressive widening** ([73]): add new actions if $|\mathcal{A}(s)|^2 \leq N(s)$, so we decide when to add new actions by comparing the cardinality $|\mathcal{A}(s)|$ of the sample of actions of this state s and the number $N(s)$ of visits of the state s. In this way we devote more resources to the most promising research direction.

4.3.2 Neural importance sampling

How to numerically sample the probability $\tilde{P}(a|s)$? We use the **neural importance sampling** ([74]): we extract z from a uniform probability distribution in the interval $[0, 1]$, $z \sim \mathcal{U}(0, 1)$; we get $x = h(z|\theta)$ where h is a neural network with parameters θ so that x is distributed according our density: it is easier to train h than to numerically invert the cumulative distribution function.
We want that both h^{-1} and $\det\left(\frac{\partial h}{\partial \theta}\right)$, the determinant of the Jacobian of h with respect to θ, exist and are easy to calculate. Function h is given by the composition

of several functions:

$$h = f_1 \circ f_2 \circ f_3 ...$$

where each f_i is built so to make the inverse and the determinant of h well-defined and easy to calculate. In fact each $f_i \equiv f$ is defined in the following way:

$$y = f(z) \quad \text{where } z \in \mathbb{R}^d$$

We can normalize z, so that $z \in [0,1]^d$; moreover d could be represented ad the disjoint union of A and B: we split the indices into two disjoint sets, so each element of z could be split accordingly; in particular $y = (y^A, y^B)$, $z = (z^A, z^B)$ and f is defined as

$$\begin{cases} y^A = z^A \\ y^B = c(z^B, m(z^A|\theta)) \end{cases}$$

where c stands for coupling layer and m is a neural network with parameters θ. The inverse of such a function, $z = f^{-1}(y)$, exists and it is easy to calculate:

$$\begin{cases} z^A = y^A \\ z^B = c^{-1}(y^B, m(y^A|\theta)) \end{cases}$$

Since the neural network m is based on the constant part of the input, its invertibility is not a problem; so invertibility of h is ultimately discharged on c. The Jacobian $\frac{\partial f}{\partial z}$ is given by four blocks:

$$\frac{\partial f}{\partial z} = \begin{pmatrix} I^A & 0 \\ ... & \frac{\partial c}{\partial z^B} \end{pmatrix}$$

where I^A represents the identity matrix for the components of sets A. We want to define $c(z^B) = (c_1(z_1^B), c_2(z_2^B), ...)$ such that $\frac{\partial c}{\partial z_B}$ is a diagonal matrix $diag\left(\frac{\partial c_1}{\partial z_1^B}, \frac{\partial c_2}{\partial z_2^B}, ...\right)$, so that

$$\det \frac{\partial f}{\partial z} = \det \frac{\partial c}{\partial z^B} = \prod_i \frac{\partial c_i}{\partial z_i^B}$$

Apart from the identity, the easiest transformation that satisfy this property is a translation, $c(z^B, m(z^A|\theta)) = z^B + m(z^A|\theta)$. But since h, at the end of the day, should represent a change in measure, a translation would not be so significant; so we choose the exponential function, $c(z^B, m(z^A|\theta)) \equiv c(z^B, (s,t)) = e^s z^B + t$.

4.3.3 Cross entropy maximization

How to numerically find $\text{argmax}_a \tilde{P}(a|s)$? We use the **cross entropy maximization** (CEM), an iterative method often used for solving problem in a continuous domain using the reinforcement learning.
We initialize a random mean μ_0 and a random variance σ_0 in the space of the action a; then we sample m elements according to the normal distribution $\mathcal{N}(\mu_0, \sigma_0)$. We compute the probability for all these elements, then we consider the $k < m$ with the highest probability. At this point we find the parameter μ_1 and σ_1 that give the best fit for the distribution $\mathcal{N}(\mu_1, \sigma_1)$ according the set of k elements of highest probability. Then we repeat with another extraction from this normal distribution, we compute the probability, then find the subset with highest probability and we fit another normal distribution to this new subset. We repeat the procedure until a convergence criterion is met (a maximum number of iterations or a minimum variation from one step to the other). In this way we find a maximum in a continuous space for an empirical probability distribution: it is the final value of the mean of the normal distribution at the end of the iterative process, i.e. μ_n.
Note that this algorithm has many parameters: we have to choose the maximum number of iterations; for each iteration we repeat the calibration procedure for a given number of batches, extracting a given number of samples each time, so we have to choose the number of batches and the number (m) of samples for each batch; finally we have to choose k, the size of the subset of the elements of maximum probability.

4.4 Bringing our problem in this framework

We want to tackle a portfolio optimization of future contracts in the Italian energy market (see Chapter 1). There is a finite number of dates in which we can trade and we have some liquidity and risk constraints for each date. The deliveries of the contracts of our portfolio all belong to a specific year of interest. For instance, let's fix our starting date in the December of the year before of the year of interest; if we suppose that there is a single trading decision for each month, we have twelve

time steps; when we are in December of the year of interest there are no choices left, we have to close all the positions and hedging all we did not hedge before.
We have a tool (see Chapter 2) that simulates the evolution of the forward prices for the six underlyings of interest of our portfolio for all the deliveries of the year of interest: PUNBASE, PUNPEAK, PSV, TTF, PFOR, EUA. This tool will feed the different MCTS episodes, allowing to build a dataset for the neural network in the Neural MCTS algorithm.
We have to translate into code the definition of state, reward, action and environment. All the remaining algorithmic part is independent from the specific problem, apart from the choice of the hyperparameters that allow us to have a good convergence rate.
Each *state* represents the set of all the information we can have at a given date; in our case there are the volume exposure, the forward prices at which we can trade on that date, the spot prices (if the current date belongs to the deliveries of the year of interest), the budget prices, the covariance matrix of the components (see Subsection 2.2.3) of each underlying, the covariance matrix of all the couples underlying-delivery, the liquidity constraints and the target limit for the risk. All these value have to be concatenate in a tensor that will constitute the input for the neural network. Note that over time there might be some elements that reduce their size. For example the tradable products are the next four months, the next four quarters and the next year and we consider only the products that belong to our year of interest; so, there is not the same number of tradable products for each trading date. So we have to pad our vector with zeros in order to have the same size for the input on every date. The choice of the set of information that defines the state is one of the choices that can modify the performance of the algorithm; for example, one test we could do is to add information to the state, such as including not only current forward prices, but also some past forward prices; this could be more informative on the evolution of forward prices, which is the most significant aspect for deciding when certain actions must be taken. As for the *reward*, being a finite time horizon, we evaluate only the final return, which in this case means the sum of all the forward PL, those directly achieved with forward contracts, and

the sum of all the spot PL, those we evaluate at each monthly delivery, for all the monthly deliveries of the year of interest; when we have reached the end of the time horizon we can sum up and judge the final value of the strategy along the entire time horizon. In defining the *action* we must consider the rule of tradability (see Subsection 1.1.3). In order to have a complete definition of the Markov Decision Process, we should define also the transition probability, i.e. the probability of the next state, given the current state and the chosen action; but in our problem we do not fully know the behaviour of the environment, in particular we do not know the transition probabilities.
The rate of convergence of the neural network is better when the input is normalized; but we have to take into account that there are certain functions that have to be translated from the normalized to the real world: for example the target limit for the risk is a unique number for all the portfolio; we can normalize the volumes and the prices for the single underlying, but we can not normalize this target limit, so the risk has to be computed in the non-normalized world.
In the Neural MTCS framework there is no a natural definition of constraints, so instead of imposing hard constraints we introduced them in a soft way, as penalties of the objective function we want to maximize:

$$\alpha_1 PL - \alpha_2 (PaR - target\_PaR)^+ - \alpha_3 (traded\_volume - maximum\_volume)^+$$

where α_1, α_2, α_3 are hyperparametric choices and $(\cdot)^+$ is the positive part; in this way, whenever the constraints are not satisfied, the value of the objective function is reduced as much as the limit is exceeded; when all the constraints are satisfied, no penalty must be paid. So it may happen that some constraints are not met, but we expect this effect to be minimized by training.

It is very difficult to appreciate a convergence improvement in our neural network and we believe that the weak link is the cross entropy maximization (see Section 4.3.3): when we challenge the CEM to find the maximum of a Gaussian distribution in a multidimensional space, the performances are good enough; but when we add a sinusoidal noise to the Gaussian distribution (see Figure 4.5), the performances become soon poor as the dimension increase (see Figures 4.6, 4.7).

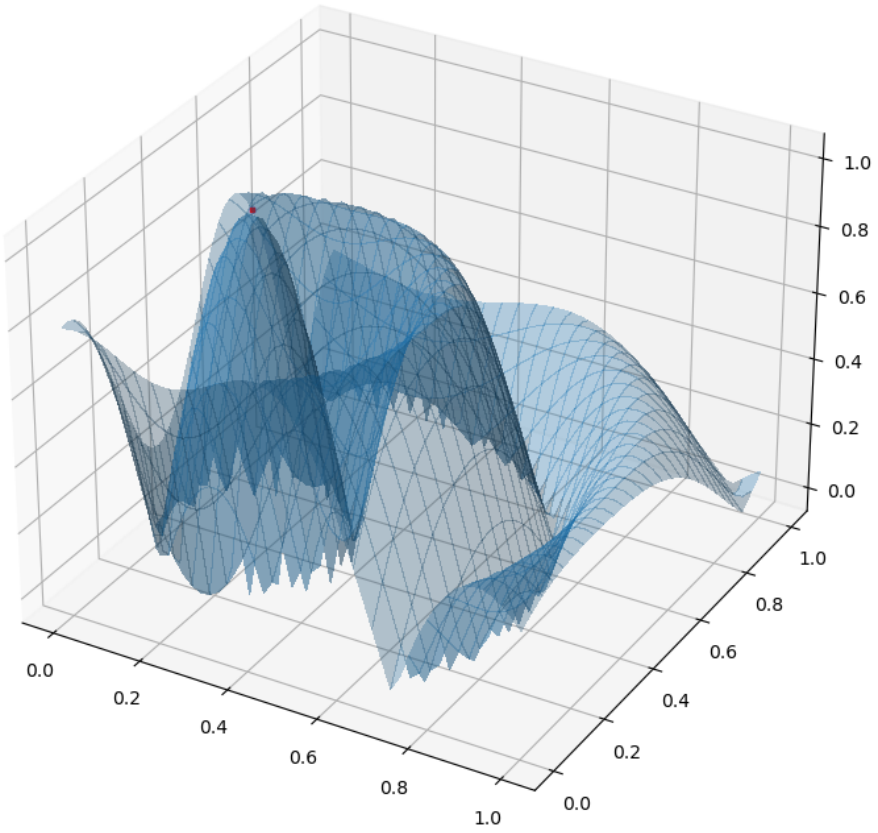


Figure 4.5: This is a graph for input dimension $N = 2$ of the function $f(x) = |cos(b * \|x\|)|e^{-\frac{\|x\|}{a^2}}$, where $\|x\| = \sqrt{\sum_{i=1}^{N}(x_i - c)^2}$, where $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}^N$ are parameters, $x \in \mathbb{R}^N$. In this dimension, the global maximum (the red dot) is easily identified by the cross entropy maximization algorithm.

So the tricky point of this Continuous Neural MCTS is the neural importance sampling because by increasing the size of the action space, we have to increase the number of simulations to have a reliable result, that is, to be sure that we have found the best policy.

Another open point is given by one approximation we made: we considered only the stochasticity on prices, but actually also the volume exposure might be adjusted over time, for example because a pandemic provokes a drop in the energy demand, or because a global energy crisis raises prices dramatically or because a

war together with a climate crisis changes completely the energy market; even in the absence of a global crisis, there could be adjustments to the estimated production of power plants and to customer demand. In order to consider also these movements, we would need to feed the MCTS episode with scenarios in which there are changes in the volume exposures, so we need to simulate prices and volumes together. We tried to test ([75]) the relationship between the variation in the forward prices and the variation in the volume exposures using a residual neural network that has an interpretation in the stochastic optimal control (see [76]): the method was not successful, so the best approximation we have up to know is to add a random noise to the volumes in order to have a simulation of the state that takes into account also of the variation in the volume exposure.

INPUT DIMENSIONS \ NUMBER OF SAMPLES IN A BATCH	1.E+02	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	1.E+08
2	1.17E-09	1.06E-10	1.82E-10	1.51E-10	6.28E-11	3.01E-10	1.46E-10
4	1.52E-01	1.31E-08	9.21E-11	7.08E-11	9.28E-11	6.87E-11	Allocation exceeds 10% of free system memory.
6	1.02E-01	9.34E-02	1.39E-08	2.28E-10	8.61E-11	8.71E-11	Allocation exceeds 10% of free system memory.
8	7.23E-02	8.19E-02	6.06E-02	9.53E-09	6.50E-10	3.22E-11	Allocation exceeds 10% of free system memory.
10	7.69E-02	8.11E-02	7.22E-02	3.65E-05	1.83E-08	7.93E-10	Allocation exceeds 10% of free system memory.
12	7.92E-02	7.60E-02	6.61E-02	8.37E-09	6.48E-07	8.71E-11	Allocation exceeds 10% of free system memory.
14	7.20E-02	6.97E-02	6.99E-02	5.61E-02	6.83E-02	4.41E-05	Allocation exceeds 10% of free system memory.
16	5.70E-02	6.16E-02	6.45E-02	7.07E-02	5.73E-02	5.00E-02	Allocation exceeds 10% of free system memory.
18	5.32E-02	4.66E-02	5.15E-02	5.02E-02	4.61E-02	5.30E-02	Allocation exceeds 10% of free system memory.
20	5.14E-02	5.12E-02	5.45E-02	4.84E-02	4.93E-02	5.18E-02	Allocation exceeds 10% of free system memory.
22	4.90E-02	4.54E-02	5.23E-02	4.65E-02	5.43E-02	4.64E-02	Allocation exceeds 10% of free system memory.
24	4.58E-02	4.41E-02	4.34E-02	4.56E-02	4.84E-02	4.69E-02	Allocation exceeds 10% of free system memory.

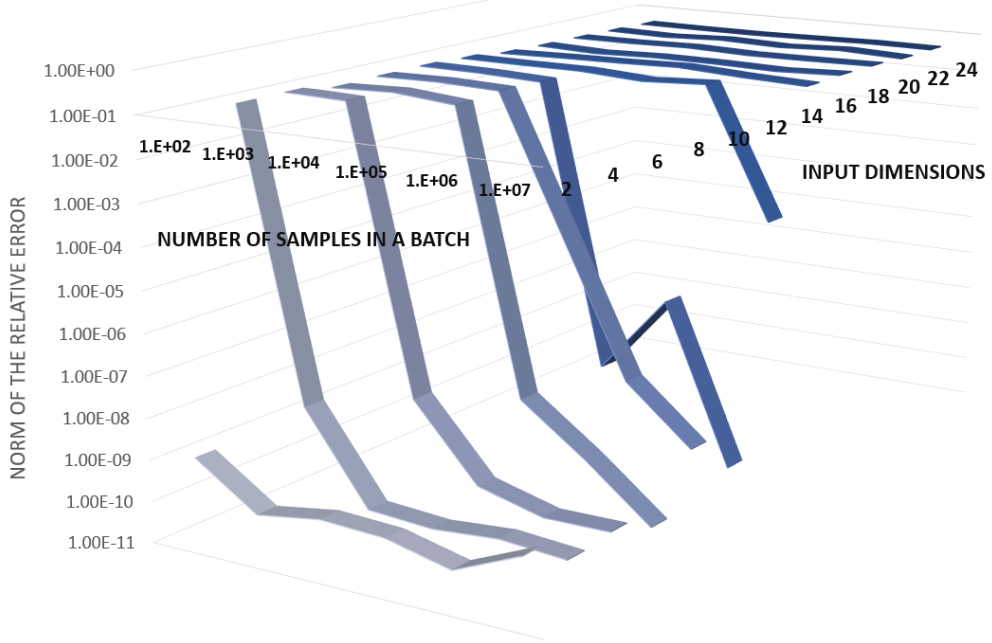


Figure 4.6: The norm of the relative error by varying the input dimension and the number of samples in a batch. Note that when input dimension is greater than 12, more samples than 10^7 would be needed to have low error; at the same time, when the number of samples in a batch reaches 10^8, even for low input sizes, there are memory problems.

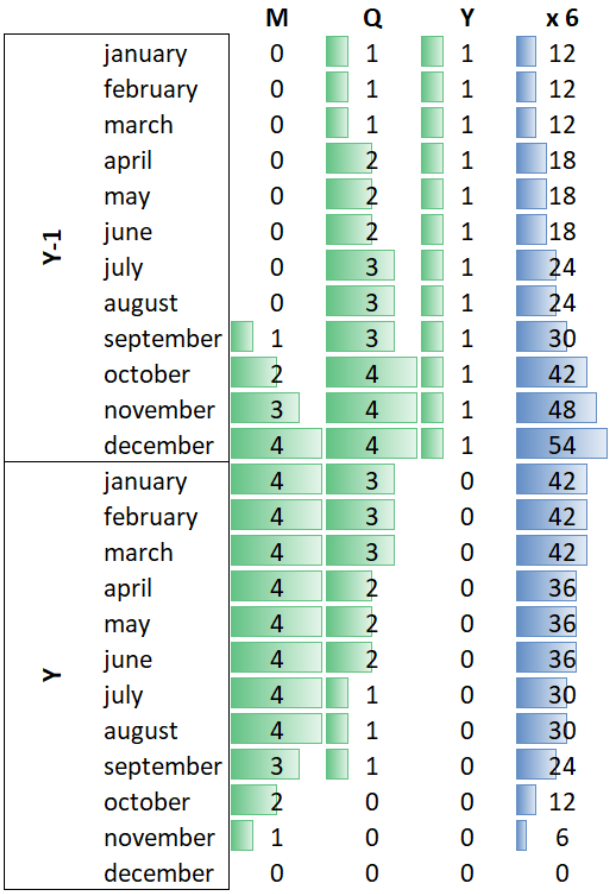


Figure 4.7: This is a count of the input dimension (i.e. the total number of tradable volumes, i.e. the number of actions) by trade date. Possible trading dates start from the year before the year of interest, where Y is the year of interest. Size is the sum of tradable products for a portfolio of six underlyings, considering M, Q and Y, where M are the monthly products (the next four months belonging to the year of interest), Q are the quarterly products (the next four quarters belonging to the year of interest), and Y is the yearly product (as soon as the year begins it is no longer tradable). Note that the input dimensions range from 6 to 54, but Figure 4.6 suggests that for input sizes greater than 12 there is either a large error or memory leaks.

Chapter 5

Final remarks and next steps

5.1 Comparison between the different approaches

The deterministic approach described in Section 1.2 is simple to interpret and implement. It is based on a forecast (called ”view”) of the the spot price that encompasses market and macroeconomic information: this might be a double-edged sword, because the view usually adds information to that extracted directly from historical and current prices, but if the resulting spot price is very far from what the view predicted, perhaps because something unpredictable happened, the performance will be poor. Furthermore, this approach does not consider the intermediate dates from today to delivery, so it does not consider the whole complexity of the problem.

The scenario approach described in Chapter 3 overcomes this issue, but in order to implement this approach we had also to develop a tool to simulate correlated forward curves (see Chapter 2); this tool adds an important piece to our puzzle also because, to the best of our knowledge, it is not trivial to find a model to simulate correlated forward curves with the additional difficulty of having underlying assets in the Italian energy market, which is not very liquid.

A good point of the scenario approach is having theorems that measure the reliability of the result based on the size of the problem; this approach is relatively easy to implement, it looks for the set of actions for all intermediate dates that

meet constraints and minimize worst profit based on price scenarios; but there are memory problems which we have approached with an approximation of the convex envelope.
Also the reinforcement learning approach (Chapter 4), as the scenario approach, is a stochastic approach that considers the whole complexity of the problem and uses the tool to simulate correlated forward curves, but it it works very differently. It is based on the training of a deep neural network that calibrates its parameters in order to optimize the final profit of the portfolio, with a penalty proportional to the overcoming of the liquidity, risk and position constraints. This deep neural network takes inspiration from the famous AlphaZero (winning in the game of Go and chess) and the training set is given by simulated games, where the game here is given by trading volumes to hedge our initial portfolio starting from a certain date up to the end of the year of interest; we need price simulations to create these simulated stories. The challenge was given by the fact that AlphaZero was thought for discrete actions and discrete states, while our problem lives in a continuous space. In particular we have to deal with a continuous numerical probability distribution, so we need a tool to extract a sample from this distribution and to find the mode of this distribution; so we tried to take inspiration from an article (see [74]), but we faced some numerical issues, in particular our deep neural network did not converge.

5.2 Possible future developments

We could extend the deterministic approach described in Section 1.2 by taking into account all intermediate dates, assuming, for example, that the view and the forward curve of each underlying do not change over time. In this way we would answer how to keep the risk below the limit for each date, while having liquidity constraints, with the assumption that prices do not change; this approach would be easier to interpret than stochastic approaches, but the main drawback would be the fact that prices actually change a lot, so we should take that into account.
As we said at the beginning, there is another source of uncertainty, in addition to

prices: volume exposures depend on the forecast of customer demand and power plant production and since it is a forecast it could be adjusted over time. We tried to find a relationship between price changes and volume changes (see [75]), but it seemed to be almost random, so we could add random noise to the volumes to mimic reality and we could use this set to train the deep neural network of the Chapter 4.

We could explore the link between reinforcement learning and optimal control (see [77], [78]): stochastic calculus tools could provide theoretical justification and interpretability to these algorithms, for example a convergence analysis and more generally analytical results.

We could overcome the memory issue of the scenario approach of Chapter 3 by rewriting the problem as a Quadratic Unconstrained Binary Optimization (QUBO, see [79]):

$$\min_x x^T Q x$$

where x is a vector of binary decision variables and Q is a square matrix of constants. This model can embrace a variety of optimization problems, most of which are NP-hard: by combining classical and quantum computing, it is possible to approach the optimum, finding a high-quality solution in polynomial time. We can embody our problem in this framework even if we have constraints: we can consider the risk constraint (see Formula (1.7)) as a penalty, while the position and liquidity constraints (see Formula (1.2) and Formula (1.12)) define the domain.

Bibliography

(1) Giachetti Fantini, M. La liberalizzazione del mercato dell'energia elettrica e del gas naturale: il caso italiano nel panorama europeo. *ApertaContrada* **2017**, 1–103.

(2) Electricity Maps, https://app.electricitymaps.com/, Accessed: 2022-12-29.

(3) Longva, P.; Keers, G. Risk Management in the Electricity Industry. *IAEE 17th Annual International Conference* **1994**, *15*.

(4) Adamko, P.; Spuchľáková, E.; Valášková, K. The history and ideas behind VaR. *Procedia Economics and Finance* **2015**, *24*, 18–24.

(5) Gambaro, A. M.; Secomandi, N. A Discussion of Non-Gaussian Price Processes for Energy and Commodity Operations. *Production and Operations Management* **2021**, *30*, 47–67, DOI: https://doi.org/10.1111/poms.13250.

(6) Farmer, J. D.; Gerig, A.; Lillo, F.; Waelbroeck, H. In 2008.

(7) Kraft, D. *A software package for sequential quadratic programming*; tech. rep. DFVLR-FB 88-28; Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen: Institut fuer Dynamik der Flugsysteme, 1988.

(8) Nocedal, J.; Wright, S. J., *Numerical optimization*; Springer, New York, NY.: 2006.

(9) Arthur, D.; Vassilvitskii, S. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

(10) Müllner, D. Modern hierarchical, agglomerative clustering algorithms, 2011.

(11) Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* **1987**, *20*, 53–65, DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

(12) Clewlow, L.; Strickland, C., *Energy derivatives: Pricing and risk management*; Lacima Publications: 2000.

(13) Heath, D.; Jarrow, R.; Morton, A. Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Valuation. *Econometrica* **1992**, *60*, 77–105.

(14) Benth, F. E.; Benth, J. S.; Koekebakker, S., *Stochastic Modeling of Electricity and Related Markets*; World Scientific Publishing Co. Pte. Ltd.: 2008.

(15) Koekebakker, S.; Ollmar, F. Forward curve dynamics in the Nordic electricity market. *Managerial Finance* **2005**, *31*, 73–94.

(16) Benth, F.; Koekebakker, S. Stochastic Modeling of Financial Electricity Contracts. *Energy Economics* **2008**, *30*, 1116–1157, DOI: 10.1016/j.eneco.2007.06.005.

(17) Barth, A.; Benth, F. E. The forward dynamics in energy markets – infinite-dimensional modelling and simulation. *Stochastics* **2014**, *86*, 932–966, DOI: 10.1080/17442508.2014.895359.

(18) Cai, W.; Pan, J. Stochastic Differential Equation Models for the Price of European CO2 Emissions Allowances. *Sustainability* **2017**, *9*, 207, DOI: 10.3390/su9020207.

(19) Oyuna, D.; Yaobin, L. Forecasting the Crude Oil Prices Volatility With Stochastic Volatility Models. *SAGE Open* **2021**, *11*, 215824402110262, DOI: 10.1177/21582440211026269.

(20) Noorani, I.; Mehrdoust, F.; Lio, W. Electricity spot price modeling by multi-factor uncertain process: a case study from the Nordic region. *Soft Computing* **2021**, *25*, DOI: 10.1007/s00500-021-06083-8.

(21) Ogbogbo, C. P. Stochastic Model Of Crude Oil Spot Price Process As A Jump-Diffusion Process. *Applied Mathematics and Information Sciences* **2019**, *13*, 1029–1037.

(22) Burger, M.; Klar, B.; Müller, A.; Schindlmayr, G. A spot market model for pricing derivatives in electricity markets. *Quantitative Finance* **2004**, *4*, 109–122, DOI: 10.1088/1469-7688/4/1/010.

(23) Hosseini, M. In 2007.

(24) Ladokhin, S.; Borovkova, S. Three-factor commodity forward curve model and its joint P and Q dynamics. *Energy Economics* **2021**, *101*, Publisher Copyright: © 2021 The Authors Copyright: Copyright 2021 Elsevier B.V., All rights reserved., 1–15, DOI: 10.1016/j.eneco.2021.105418.

(25) Chiarella, C.; Clewlow, L.; Kang, B. Modelling and Estimating the Forward Price Curve in the Energy Market. *Quantitative Finance Research Centre, University of Technology, Sydney, Research Paper Series* **2009**.

(26) Higgins, M. A Two Factor Forward Curve Model with Stochastic Volatility for Commodity Prices, 2017, DOI: 10.48550/ARXIV.1708.01665.

(27) Li, A. Covariance Matrix Extrapolation for Energy Forward Prices. *SSRN Electronic Journal* **2003**, DOI: 10.2139/ssrn.938083.

(28) Sclavounos, P. D.; Ellefsen, P. E. Multi-factor model of correlated commodity forward curves for crude oil and shipping markets. **2009**.

(29) Edoli, E.; Tasinato, D.; Vargiolu, T. Calibration of a multifactor model for the forward markets of several commodities. *Optimization* **2013**, *62*, DOI: 10.1080/02331934.2013.854786.

(30) Russo, F. Commodity risk management: a two-factor model with long-term dependency, Ph.D. Thesis, Università degli studi Roma Tre, 2017.

(31) Pirrong, C., *Commodity Price Dynamics: A Structural Approach*; Cambridge University Press: 2011.

(32) Audet, N.; Heiskanen, P.; Keppo, J.; Vehviläinen, I. In *Modelling Prices in Competitive Electricity Markets*, Bunn, D., Ed.; John Wiley & Sons Inc.: United States, 2004.

(33) Ghiani, E.; Galici, M.; Mureddu, M.; Pilo, F. Impact on Electricity Consumption and Market Pricing of Energy and Ancillary Services during Pandemic of COVID-19 in Italy. *Energies* **2020**, *13*, DOI: 10.3390/en13133357.

(34) Lazo, J.; Aguirre, G.; Watts, D. An impact study of COVID-19 on the electricity sector: A comprehensive literature review and Ibero-American survey. *Renewable and Sustainable Energy Reviews* **2022**, *158*, 112–135, DOI: https://doi.org/10.1016/j.rser.2022.112135.

(35) Gilbert, A.; Bazilian, M. D.; Gross, S. THE EMERGING GLOBAL NATURAL GAS MARKET AND THE ENERGY CRISIS OF 2021-2022. *Foreign Policy* **2021**.

(36) Fanelli, V.; Schmeck, M. D. On the seasonality in the implied volatility of electricity options. *Quantitative Finance, DOI: 10.1080/14697688.2019.1582792* **2018**.

(37) Kiesel, R.; Schindlmayr, G.; Boerger, R. A two-factor model for the electricity forward market. *Quantitative Finance* **2009**, *9*, 279–287, DOI: 10.1080/14697680802126530.

(38) Bingol, O. R.; Krishnamurthy, A. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX* **2019**, *9*, 85–94.

(39) Candes, E. J.; Li, X.; Ma, Y.; Wright, J. Robust Principal Component Analysis?, 2009.

(40) Bailey, S. Principal Component Analysis with Noisy and/or Missing Data. *Publications of the Astronomical Society of the Pacific* **2012**, *124*, 1015–1023, DOI: 10.1086/668105.

(41) Hair, J. F.; Anderson, R. E.; Tatham, R. L.; Black, W. C., *Multivariate data analysis*; Fourth edition, Prentice Hall (Englewood Cliffs, New Jersey).: 1995.

(42) Schwartz, E. The Stochastic Behavior of Commodity Prices: Implications for Valuation and Hedging. *The Journal of Finance* **1997**, *52*, 923–973.

(43) Guerini, A.; Marziali, A.; Nicolao, G. MCMC calibration of spot prices models in electricity markets. *Applied Stochastic Models in Business and Industry* **2019**, *36*, DOI: 10.1002/asmb.2471.

(44) Barndorff-Nielsen, O. Normal Inverse Gaussian Distribution and Stochastic Volatility Modelling. *Scandinavian Journal of Statistics* **1997**, *24*, 1–13.

(45) Zhang, T.; Yang, Y. Robust PCA by manifold optimization. *Journal of Machine Learning Research* **2018**, *19*.

(46) Buehler, H.; Horvath, B.; Lyons, T.; Perez Arribas, I.; Wood, B. A Data-Driven Market Simulator for Small Data Environments. **2020**.

(47) Buehler, H.; Horvath, B.; Lyons, T.; Perez Arribas, I.; Wood, B. Generating Financial Markets With Signatures. **2020**.

(48) Markowitz, H. Portfolio Selection. *The Journal of Finance , Mar., 1952, Vol. 7, No. 1 (Mar., 1952), pp. 77-91* **1952**.

(49) Black, F.; Litterman, R. Global Portfolio Optimization. *Financial Analysts Journal* **1992**, *48*, 28–43.

(50) Hallerbach, W.; Grootveld, H. Variance versus downside risk: is there really that much difference? *European Journal of Operational Research* **1999**, *114*, 304–319.

(51) Gökgöz, F.; Atmaca, M. E. Financial Portfolio Optimization in Electricity Markets: Evaluation via Sharpe Ratio. **2016**, DOI: 10.5281/zenodo.1127190.

(52) Campi, M.; Garatti, S.; M.Prandini The scenario approach for systems and control design. *Annual Reviews in Control* **2009**, *33*, 149–157.

(53) Campi, M. C.; Garatti, S. A Sampling-and-Discarding Approach to Chance-Constrained Optimization: Feasibility and Optimality. *Journal of Optimization Theory and Applications 148(2):257-280 DOI: 10.1007/s10957-010-9754-6* **2011**.

(54) Das, S. R.; Ostrov, D.; Radhakrishnan, A.; Srivastav, D. Dynamic optimization for multi-goals wealth management. *Journal of Banking and Finance* **2022**, *140*, 106–192, DOI: https://doi.org/10.1016/j.jbankfin.2021.106192.

(55) Margellos, K.; Falsone, A.; Garatti, S.; Prandini, M. Distributed Constrained Optimization and Consensus in Uncertain Networks via Proximal Minimization. *IEEE Transactions on Automatic Control* **2018**, *63*, 1372–1387, DOI: 10.1109/TAC.2017.2747505.

(56) Barber, C. B.; Dobkin, D. P.; Huhdanpaa, H. The Quickhull algorithm for convex hulls. *Acm Transactions on Mathematicl Softward* **1996**, *22*, 469–483.

(57) Sartipizadeh, H.; Vincent, T. L. Computing the Approximate Convex Hull in High Dimensions, 2016.

(58) Sartipizadeh, H.; Açikmeşe, B. In *2018 Annual American Control Conference (ACC)*, 2018, pp 4700–4705, DOI: 10.23919/ACC.2018.8430936.

(59) Agrawal, A.; Verschueren, R.; Diamond, S.; Boyd, S. A rewriting system for convex optimization problems. *Journal of Control and Decision* **2018**, *5*, 42–60.

(60) Diamond, S.; Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* **2016**, *17*, 1–5.

(61) Domahidi, A.; Chu, E.; Boyd, S. In *European Control Conference (ECC)*, 2013, pp 3071–3076.

(62) Sutton, R. S.; Barto, A. G., *Reinforcement Learning: An Introduction*, Second; The MIT Press: 2018.

(63) Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; Wiley Series in Probability and Statistics; Wiley: 1994, DOI: 10.1002/9780470316887.

(64) Bellman, R., *Dynamic Programming*; Dover Publications: 1957.

(65) Kocsis, L.; Szepesvári, C. In 2006; Vol. 2006, pp 282–293, DOI: 10.1007/11871842_29.

(66) Auer, P.; Cesa-Bianchi, N.; Fischer, P. Finite-time Analysis of the Multi-armed Bandit Problem. *Machine Learning* **2002**, *47*, 235–256, DOI: 10.1023/A:1013689704352.

(67) Kozelek, T. Methods of MCTS and the game Arimaa. *Master's thesis, Department of Theoretical Computer Science and Mathematical Logic* **2009**.

(68) Coulom, R. In 2006; Vol. 4630, DOI: 10.1007/978-3-540-75538-8_7.

(69) Gelly, S.; Wang, Y. Exploration exploitation in Go: UCT for Monte-Carlo Go. **2006**.

(70) Brügmann, B. *Monte Carlo Go*; tech. rep.; Munich: Max-Planke-Inst. Phys., 1993.

(71) Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144, DOI: 10.1126/science.aar6404.

(72) Silver, D. Reinforcement Learning and Simulation-Based Search in Computer Go. *PhD thesis* **2009**.

(73) Chaslot, G.; Winands, M.; Herik, H.; Uiterwijk, J.; Bouzy, B. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation* **2008**, *04*, 343–357, DOI: 10.1142/S1793005708001094.

(74) Müller, T.; Mcwilliams, B.; Rousselle, F.; Gross, M.; Novák, J. Neural Importance Sampling. *ACM Transactions on Graphics* **2019**, *38*, 1–19, DOI: 10.1145/3341156.

(75) Angeli, T. Applicazione delle Reti Neurali residuali ai Mercati Energetici e Loro Interpretazione come problema di Controllo Ottimo stocastico, MA thesis, Alma Mater Studiorum - Universitá di Bologna - Corso di studio in Matematica, 2021.

(76) Ee, W.; Han, J.; Li, Q. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences* **2018**, *6*, DOI: 10.1007/s40687-018-0172-y.

(77) Huré, C.; Pham, H.; Bachouch, A.; Langrené, N. Deep Neural Networks Algorithms for Stochastic Control Problems on Finite Horizon: Convergence Analysis. *SIAM Journal on Numerical Analysis* **2021**, *59*, 525–557, DOI: 10.1137/20M1316640.

(78) Wang, H.; Zariphopoulou, T.; Zhou, X. Y. Reinforcement Learning in Continuous Time and Space: A Stochastic Control Approach. *Journal of Machine Learning Research* **2020**, *21*, 1–34.

(79) Glover, F. W.; Kochenberger, G. A. A Tutorial on Formulating QUBO Models. *CoRR* **2018**, *abs/1811.11538*.