

Master of Science
Master thesis at Department of Applied Mathematics and Computer Science

 **DTU Compute**
Institut for Matematik og Computer Science

Real-Time Forecasting in Renewable Energy Production: Improving Wind and Solar Forecasts through Advanced Error Correction Methods

A Collaboration with QUENT

Author: Bastian Schmidt Jørgensen

9th July 2024



Supervisor: Prof. Henrik Madsen
Co-supervisor: Peter Nystrup and Jan K. Møller

DTU Compute

Department of Applied Mathematics and Computer Science

Technical University of Denmark

Building 341

2800 Kongens Lyngby, Denmark

Preface

This thesis has been written over approximately 23 weeks: 1st of February 2024 to 9th of July 2024, at the department DTU Compute at the Technical University of Denmark in Lyngby, Denmark. The thesis is conducted as part of the education for a degree in Master of Science, MSc. The thesis amounts to 32.5 ECTS points. A few paragraphs of this thesis have used AI for proofreading.

It is assumed that the reader is familiar with multivariate statistics, probability theory, likelihood estimation and time series analysis.

Acknowledgments

I am grateful to QUENT ApS for assistance regarding datasets that made these studies possible. I would like to thank my supervisor and co-supervisors for their support, great ideas and suggestions. It has been a true pleasure. Thanks!

Kongens Lyngby, 9th July 2024



Bastian Schmidt Jørgensen

Abstract

This thesis leverages neural networks, time-adaptive quantile regression and stochastic differential equations to improve day-ahead forecasts for wind- and solar power production using a model-free approach. We developed a three-step corrective procedure to effectively error-correct forecasts and forecast-ensembles. Neural networks are used to find the optimal basis function for the time-adaptive quantile regression algorithm. The number of steps you apply in the procedure will depend on your application. In almost all the areas and type of energy combinations presented in the results, improvements are observed. The improvements are measured by the CRPS, Variogram and Quantile scores. We show that in almost all cases, our methodology improves the forecasts. Case studies showed potential profitability of improved forecasts, though limited sample size affected statistical significance. The set of new methodologies for error correction of energy forecasts, contributes to reliable renewable energy systems in the future. We will publish the developed Python package for use by forecasters and practitioners by end of year.

Contents

Preface	i
Contents	iii
1 Introduction	1
1.0.1 The Danish Energy Market	2
1.1 Previous Work	4
1.1.1 A Historical Prelude	4
1.1.2 Similar Work and Contributions	5
1.1.3 Problem Statement	6
1.2 Thesis Structure and New Work	6
1.2.1 Nomenclature, Abbreviations and Notation	7
2 Dataset	9
3 Methods & Theory	16
3.1 Quantile regression	16
3.1.1 Simple Quantile Regression	16
3.1.2 Non-linear Quantile Regression	22
3.1.3 Inference for Quantile Regression	23
3.2 Time-Adaptive Quantile Regression	24
3.2.1 Notation	24
3.2.2 Algorithm	26
3.3 General Error Measures and Statistical Tools	29
3.4 Quantifying the Performance of Quantiles	30
3.4.1 CRPS- & Skill Score	31
3.4.2 Variogram Score	32
3.4.3 Quantile Score	34
3.5 Neural Networks	34
3.5.1 Feed Forward Neural Network	34
3.5.2 Long Short-Term Memory Network	36
3.5.3 Batch Normalization	39
3.5.4 Adam Optimizer	40

3.6	Updating Basis Functions	42
3.7	Stochastic Differential Equations	46
3.7.1	Ornstein-Uhlenbeck Process	48
3.7.2	Predictions from SDEs	49
3.7.3	Training and Test Set	50
3.7.4	Fixing Correlation Structure	51
4	Results	53
4.0.1	Model Selection	56
4.1	Corrected Ensembles	58
4.2	Time-Adaptive Quantile Regression	65
4.2.1	ACF, PACF and QQ Plots	67
4.3	SDE Corrected Ensembles	69
4.3.1	TAQR and SDE	74
4.4	Python- Implementation and Package	76
5	Discussion	78
5.1	Comment on Results	78
5.2	Validation of Ensembles by Trading Strategy	81
5.2.1	Case study 1: Own Wind Power Plant - Fixed Risk	81
5.2.2	Case study 2: Energy-Trader	83
5.3	Future Work	85
5.3.1	Going from Offline to Online Implementation	87
5.3.2	Ideas for Improvement	88
6	Conclusion	91
	Bibliography	93
	Appendices	98
	Appendix A	100
A.1	Code Overview	100
A.2	Methods & Theory	101
A.2.1	Proof for Quantile Regression	101
A.2.2	Equations for Simplex Algorithm	103
A.2.3	Equations for Updating Step in Simplex Algorithm	103
A.2.4	Illustration of Simplex	104
A.2.5	SDE Parameters	106
A.3	Results	109
A.3.1	Additional Statistics for Corrected Quantiles	113
A.3.2	Probe (b) LSTM	113
A.3.3	Further Statistics on TAQR Made from SDE-CE	114
A.4	Discussion	117
A.4.1	Casestudy 1	117

CHAPTER 1

Introduction

The world is witnessing a growing urgency to address the issue of climate change. Governments, corporations, countries and individuals alike are setting ambitious targets to achieve carbon neutrality by 2030 or 2050 (Ministry of Foreign Affairs of Denmark, 2024). This global effort, often referred to as the green transition, demands the collective participation of all stakeholders in creating a sustainable future. Among the various sectors that can contribute to this endeavor, energy trading and renewable energy forecasting stands out as critical players, wielding the potential to drive significant change.

Concurrently, artificial intelligence is rapidly evolving, but still many language models remain closed-source and inaccessible, despite promises of transparency and open access (OpenAI, 2024). This (lack of) transparency presents a significant challenge for researchers and practitioners seeking to improve and refine these models. Similarly, many proprietary mathematical models are not shared with the public eye either. To address this issue, the concept known as *model-free error corrections*, has emerged as a promising approach. By developing techniques to correct models without requiring intimate knowledge of their internal architecture, we can make progress in enhancing their performance and reliability for a wide range of forecasts; frankly almost all forecasts that are based on predictions from a basis could gain from this approach.

Accurate forecasts for solar and wind power are crucial for maintaining the stability and reliability of the power grid. As renewable energy sources are variable and dependent on weather conditions, precise predictions help grid operators balance supply and demand, preventing blackouts and ensure a steady flow of electricity as well as a grid frequency that stays within the desired range. Correcting renewable energy forecasts enhances economic efficiency by reducing the need for expensive backup power sources. By accurately predicting renewable energy output, grid operators can optimize the use of existing resources, minimizing costs and emissions associated with running additional fossil fuel-based (backup) power plants, see e.g. (Kandpal et al. (2024), McKinsey (2020), International Renewable Energy Agency (IRENA) (2020)). This aligns with global efforts to combat climate change and supports the goals set in international agreements such as the Paris Agreement (Kolkowska, 2023).

Reliable forecasts build confidence among investors and developers in the renewable energy sector. Accurate data on potential energy output and grid performance encourages further investment and development of solar- and wind power projects, driving technological advancements and increasing the overall capacity of renewable energy sources (Institute for Energy Economics and Financial Analysis, 2023).

This thesis aims to explore the intersection of two crucial domains – model-free error corrections and energy trading – and their combined potential to support the green transition. In this thesis we will focus on energy data from Denmark. We seek to develop new ways to correct and optimize forecasts, ultimately enabling more efficient and effective energy trading strategies. Based on the profits for energy-trading companies in 2022, there is a huge financial motivation to get energy forecasts right. Therefore, we will not introduce the danish energy market.

1.0.1 The Danish Energy Market

We will take offset in Nord Pool as one of the leading power markets/exchanges of Europe, especially covering the Nordics and therefore Denmark. There are three main ways you can participate in the energy markets: Day-ahead, intra-day and the balancing market. As the name suggest the *day-ahead market* accepts bid and offers from market participant up until 12:00 CET on day $d - 1$ for delivery on day d . The price is found as an optimization problem, where social welfare is maximized, see (Nord Pool Group, 2023a).

The *intraday market* is a continuous market that provide a mix of 15-, 30- and 60 minute blocks in 16 different countries throughout Europe via the Nord Pool exchange. Here bids are matched with offers in real time and the price is not available to us in a simple to access dataset, see (Nord Pool Group, 2023b).

The *balancing market* ensures balance and stability on the grid. In Denmark, Energinet makes sure there is always balance, and therefore your balancing order will always be filled. If you have a huge surplus sell order that needs balancing, you are to expect a bad price for your energy. We will be using the balancing market in our case-studies in Section 5, since these prices, albeit volatile, is available to us. The volatility is asymmetric, meaning that the price can go very high, but rarely becomes significantly negative. The law of supply and demand applies, almost as good as in theory, in the balancing market. If you forecast a big surplus of energy - compared to consensus - one day in advance, then you would expect the balancing price to be lower than the day-ahead price and vice versa, see (Nord Pool Group, 2023c).

The foundation of our work is connected to European Centre for Medium-Range

Weather Forecasts (ECMWF), which generates 50 distinct scenarios of potential weather outcomes by perturbing initial conditions slightly on their "control ensemble", they call it the *Ensemble Forecast Suite* (ENS), and it can be accessed here (ECMWF, 2023). They note that if they average the 50 ensembles they have lower skill than the control ensemble, but any of the individual ensemble might have higher skill.

A forecast provider uses these weather scenarios to provide us with forecast ensembles of how much energy production is stemming from wind - both onshore and off-shore - along with solar power in the two areas DK1 and DK2. DK1 and DK2 make up the entirety of Denmark, see Figure 1.1. These forecasts ensembles, will be referred to as the *ensembles*.



Figure 1.1. Areas DK1 and DK2, that make up Denmark. Figure from Energinet (2023).

The ensembles alongside prices from Energinet is the most promising data source for our thesis, offering a good foundation for developing advanced error correction techniques. Furthermore, Energinet provides a superb API solution, see (Energidataservice, 2024).

The primary objective of this thesis is to develop a methodology for correcting energy forecasts, yielding the optimal description of 50 key quantiles in the day-ahead markets of wind- and solar in DK1 and DK2. By achieving a high level of accuracy in our forecast ensembles, we unlock the potential to trade on these corrected quantiles, potentially outperforming other participants in a combination of the day-ahead and balancing market. This competitive edge can be transformative, for e.g. individual traders. However we do also believe that insights are valuable for the broader energy sector, as it drives innovation.

By grounding our investigation in the real-world challenges and opportunities of the Danish energy market, we aim to contribute meaningfully to the ongoing dialogue surrounding model-free error corrections and their application in the green transition, see e.g. the paper by Stavrev and Ginchev (2024), which uses reinforcement learning in a model-free approach to optimize energy systems.

The Danish TSO, Energinet, has recently introduced the *P90 requirement*, see (Gade et al., 2024). This requirement allows for a probabilistic approach to allow ancillary services, such as batteries, or in the case of Ireland, high-speed spinning flywheels, which at a moments notice is ready to keep the grid frequency constant, to make reserve bids in the market. The key condition is that the reserve bid must be fully available at least 90% of the time. Energinet typically reviews the past three months to assess how often a reserve provider has experienced a shortfall. If a provider fails to meet the promised amount more than 10% of the time, they are no longer eligible to participate in the P90 service (Lunde et al., 2024). This requirement directly motivates the use of quantiles, specifically the 90%-quantile, as it highlights the need for accurate day-ahead quantile forecasts of energy production.

1.1 Previous Work

1.1.1 A Historical Prelude

The median is a simple concept and can often be thought of as the middle, when ordering sticks by length or sorting people by height, the median is *the "data-point" in the middle*. The idea of the median is age old, however estimation of the median in a more mathematical context, was proposed circa 1757-1760 by Ruder Josip Bošković, a Jesuit Catholic priest from Dubrovnik (Stigler, 1984). Bošković was studying how to find the equator on a rotating planet from three observations, developing on some of Newton's work. His work precedes some of Legendre's work on Least Squares presented in 1805. Later, Laplace starting building upon the work of Bošković, and developed his "methode de situation", which basically laid out the groundwork for Bayesian statistics and probability as we know it today. Following this, Francis Edgeworth introduced his "plural median - a geometric approach to median regression".

Standing on the shoulders of Bošković, Laplace and Edgeworth, Roger Koenker and Gilbert Bassett, Jr. famously published and popularized Regression Quantiles, which presented Quantile regression as we know it today, (Koenker & Bassett Jr., 1978).

1.1.2 Similar Work and Contributions

Here, we will discuss previous work on quantile regression and neural networks, followed by a description of this thesis' contributions.

In H. A. Nielsen, Nielsen, et al. (2006) they try to go from wind forecasts to wind ensembles and then to quantile forecast. Their approach include statistical modelling, since they argue that they can not go directly from the wind power ensembles to quantile forecasts. In H. A. Nielsen, Madsen, and Torben (2006) they use quantile regression to try and extend their forecast. Lauret et al. (2017) uses quantile regression models to model solar power intra-day. They show that using Numerical Weather Predictions (NWP) improves their quantile regression models. However, in this thesis we will restrict ourselves. The forecasts we assume given to us, of course rely on NWP, but when we correct these forecasts we do so in a completely weather- and model-free environment, which is solely data-driven.

This goes to show that quantile regression and wind- and solar forecast goes hand in hand and probabilistic forecasts make sense in this setting.

We will now turn our attention to an interesting aspect of this thesis, which, although not directly applicable, has garnered attention in several papers. Specifically, this involves using neural networks for quantile regression. Note, that our application is not the same, but the motivation for the multidimensional quantile loss functions stems from here. Cannon (2011) implements quantile regression with neural networks in R, and in Bremnes (2020) the quantile functions are specified by linear combinations of Bernstein basis polynomials, while their coefficients are assumed to be related to ensemble forecasts by means of a highly adaptable neural network. This article provided good inspiration for this thesis. Instead of Bernstein basis polynomials Alitasb and Salau (2024) is a good example of using Radial basis functions instead. In Fakhoury et al. (2022) they have developed an Expressive Spline-based Neural network that rely on the combined ideas of the very recently published Kolmogorov Arnold Networks, ensembles of probabilistic trees, and multivariate B-spline representations.

We can conclude that using neural networks with quantile regression also makes sense.

To do a time adaptive quantile regression model we will employ the simplex method for quantile regression described in J. Møller et al. (2006). We note that a similar approach is also found in numerous other articles with different notation, see e.g. (Van de Geer, 2003, Alhamzawi et al., 2011, Hubert et al., 2004).

In the development of Probabilistic Forecasts of Wind Power Generation, J. K. Møller et al. (2016) uses stochastic differential equations to model the residuals. This is a neat trick developed at the Dynamical Systems department at DTU, to the best of the

authors knowledge. We have also drawn inspiration from the work in E. B. Iversen et al. (2017) and E. B. Iversen et al. (2016).

1.1.3 Problem Statement

Based on the introduction and previous work developed in this field, we will in this thesis investigate the following

How can we leverage time-adaptive data-driven neural networks and quantile regression to obtain improved day-ahead forecasts for wind and solar power production using a model-free approach and is it then possible to make a profitable energy-trading strategy under some fair assumptions?

1.2 Thesis Structure and New Work

The target audience for this thesis is an avid energy trader or energy forecaster with a deep mathematical understanding and a solid core knowledge in the fields of statistics and probability. Thus we aim to keep the thesis as self-contained as possible while still maintaining conciseness.

The thesis will start by introducing the dataset in Section 2. Here we will briefly show summary statistics and plots of the data.

In Section 3 we will present the relevant theory. We will only focus on the quintessential parts of the theory used, since we will cover Neural Networks, Time-Adaptive Quantile Regression algorithms and Stochastic Differential Equations. These topics are extremely rich and many books have been written on them, some of which we will reference in the before-mentioned section.

We will present the results of applying this theory in Section 4. Here we will explore many scores to find the best framework for finding the optimal quantiles. In this section we will also explore and review the results.

In Section 5, we will provide perspective on the results by introducing two case studies. We will also cover topics that are relevant for future work and explain what it would take to take this thesis from the current offline implementation to an online implementation.

Finally, we end with a recapitulating conclusion in Section 6.

This thesis will be using Definition, Theorem, Example and Summary environments that will start with **bold text** and include the *relevant statement italicized*. A Summary is added at the end of a (sub)section when it is deemed necessary.

We have in Section 1.1.2 presented previous work utilizing neural networks (NN), time-adaptive quantile regression (TAQR), and stochastic differential equations (SDE) in primarily separate cases. We use these methods in a "synergic" combination in our thesis. This thesis introduces new findings on how our "three-step-rocket" (NN-SDE-TAQR) can significantly improve forecasting in a model-free setting without external inputs. Our approach achieves upwards of 66% improvement in Variogram scores and more than 44% improvement in CRPS scores, while also enhancing the quantile loss score. These improvements put the work of this thesis at the forefront of advanced error correction in a model- and input free environment.

All development has been conducted within a Python framework, emphasizing reproducibility, ease of use, and transparency. We plan to release this Python package to the public by the end of the year. For a way to access the preliminary code see Section 4.4.

1.2.1 Nomenclature, Abbreviations and Notation

Symbol	Definition
$\mathbb{R}$	The set of real numbers.
$\mathbb{Z}$	The set of all integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$.
$\mathbb{R}_+$	The set of positive real numbers.
$\mathbb{R}_0$	The set of non-negative real numbers.
$\mathbb{R}^n$	The n -dimensional real vector space.
$\mathbb{R}^{n \times m}$	The set of $m \times n$ real matrices.
CDF	Continuous Distribution Function.
pdf	Probability Density Function.
iid	Independent Identically Distributed.
IQR	Inter Quartile Range.
A, X	Bold face upper case letters are used for matrices. X will be used for the design matrix.
A _{·,s}	The s 'th column of the matrix, s can be a set of indexes.
A (h)	The h ' row or rows of the matrix, h can be a set of indexes.
diag(A)	A vector with the diagonal elements of the matrix A .
diag(a)	A matrix with the diagonal elements equal to a and all other elements equal to zero.
y (h), y _{$\mathcal{B}$}	Vector containing the elements of index set h or $\mathcal{B}$.

Table 1.1. Mathematical Symbols and their definitions.

Acronym	Definition
OE	Original Ensembles
CE	Corrected Ensembles
TAQR	Time-Adaptive Quantile Regression
TP	TAQR Predictions
TP-OE/TAQR-OE	TP based on OE
TP-CE/TAQR-CE	TP based on CE
SDE	Stochastic Differential Equations
SDE-CE	SDE applied to CE
TAQR-SDE-CE	TP based on SDE-CE
NWP	Numerical Weather Prediction
WPPT	Wind Power Prediction Tool
ECMWF	European Center for Medium-Range Weather Forecast
CRPS	Continuous Ranked Probability Score
QSS	Quantile Skill Score or Quantile Loss Function Score
DK1, DK2	The two areas the energy grid is split into in Denmark
NN	Neural Network
FFNN	Feed Forward Neural Network
LSTM	Long Short-Term Memory (network)

Table 1.2. List of Acronyms and their definitions.

CHAPTER 2

Dataset

For our dataset, before we do anything else with it, we put it through our data preprocessing line, shown in Figure 2.1. Since we believe that the pipeline is quite self explanatory we will here only briefly mention, that the reason we divide by the max value in the dataset, is to effectively have applied a *min-max* transformation to the dataset, which helps - and is extremely crucial - even a necessity for the success of training the LSTM network, we will be presenting later. In general, we apply this preprocessing to be able to train our network and obtain meaningful plots and statistics e.g. free of NaN-values.

In this section we will present the data we primarily use in this thesis, namely the power production from DK1 and DK2 based on offshore wind, onshore wind and solar.

The *ensembles* that we talked about in the introduction stems from the European Centre for Medium-Range Weather Forecast (ECMWF, 2024). The forecasted weather ensembles are available and have been transformed to show the power production as aforementioned.

The control ensemble will here be dubbed the *High Probability Ensemble*, which is what ECMWF believes to be the most likely scenario i.e. the one from which they perturb the initial conditions.

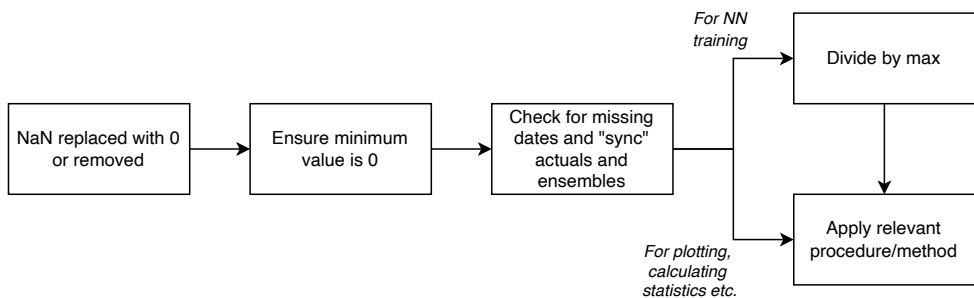


Figure 2.1. Data preprocessing pipeline

These ensembles are made available at 12:00 CET at day $d - 1$, covering the 24 hours in day d .

The actual observations are available from Energinet through their data provider Energi Data Service (Energidataservice, 2024). The actual observation are available on both 5 minute and 1 hour time frames. Energi Data Service also provide day-ahead, intra-day, 1 hour, 5 hour and current forecasts for the power production. All of this data is made readily available through their simple, yet capable API solution.

Table 2.1 summaries the statistics for the data. We will here comment on the aspects we find interesting. The data consists of 18718 hourly data points corresponding to approximately 2.14 years worth of data. One would not expect huge differences across the two regions DK1 and DK2, but still there are installation differences; for example DK1 has a much higher max output for Onshore Wind Power than DK2.

DK1's Onshore wind power production is the biggest production amongst the six power productions.

Kurtosis is used as a measure for how heavy the tails are in a probability distribution, where distributions with a heavy tail has high kurtosis. For comparison the standard normal distribution has a kurtosis of 3. No extreme kurtosis values are to be found in the dataset with DK1 Solar power being the biggest kurtosis at 4.23.

In Figure 2.2 we present the raw data plotted versus time. The solar power has daily and yearly seasonality, while the wind power does not; see Figure 2.5. In general the observations are highly autocorrelated. The solar power shows negative autocorrelation in lag 8-16, and multiples hereof, which corresponds to the typical day/night-cycle throughout the year in Denmark. The autocorrelation function is defined in Definition 3.6.

After having displayed the data in Figure 2.2, we now focus on some interesting descriptive statistics of our dataset, that will be relevant in our Section 4 and 5.

In Figure 2.3 the histograms of the actual hourly observations are plotted. This is included for completeness. Note, the sharp fall of at the end of the power output spectrum is related to the power curve of offshore wind mill farms, see e.g. J. Iversen et al., 2017, Figure 1, 2. Here the wind curve peaks at a certain max threshold wind speed, after which it dies off, due to safety regulations.

We notice that there is some negative observations for both the Offshore Wind Power and Solar Power. The reason is simply an error in the dataset which was confirmed by Energi Data Service on their website: "NOTE: Data are based on upscaled real-time power measurements from the SCADA system. Errors will occur, and will generally not be corrected.", see (Energidataservice, 2024). We use this dataset, since this is

		OffshoreWindPower	SolarPower	OnshoreWindPower
DK1:	Count	18718.00	18718.00	18718.00
	Mean	574.68	195.62	1047.10
	Std	392.57	338.79	870.59
	Skewness	0.07	2.11	0.85
	Kurtosis	-1.40	4.23	-0.18
	Min	-8.93	-0.31	4.40
	25%	201.11	0.42	308.61
	50%	564.34	5.73	824.94
	75%	941.14	248.75	1589.04
	Max	1327.72	1879.88	3562.69
		OffshoreWindPower	SolarPower	OnshoreWindPower
DK2:	Count	18718.00	18718.00	18718.00
	Mean	422.98	83.29	191.09
	Std	329.50	138.33	151.19
	Skewness	0.24	1.76	0.84
	Kurtosis	-1.43	2.12	-0.33
	Min	-6.34	-0.48	5.20
	25%	104.47	-0.15	62.01
	50%	368.11	2.24	145.77
	75%	746.24	113.59	291.53
	Max	982.52	654.99	634.85

Table 2.1. Table showing summary statistics of the hourly dataset used in this thesis.

the dataset that is available *live*. Energi Data Service also provide a *Production and Consumption - Settlement* dataset, which have been corrected for errors, but data is delayed between 9-15 days.

We quickly include a normal QQ plot, to show the normality - or lack thereof of our data, see Figure 2.4. The data is not normally distributed, which is also evident from the fact that 0 is a natural lower bound. This can also be seen from the histograms in Figure 2.3.

Lastly, we will touch upon the autocorrelation in the actual raw observations. In Figure 2.5 we clearly see a daily seasonality in the solar data as one would expect. There is also autocorrelation present in the other datasets.

In Figure 2.6, we present an example of the ensembles and actual observations for all three types of power production alongside the high probability ensemble and the

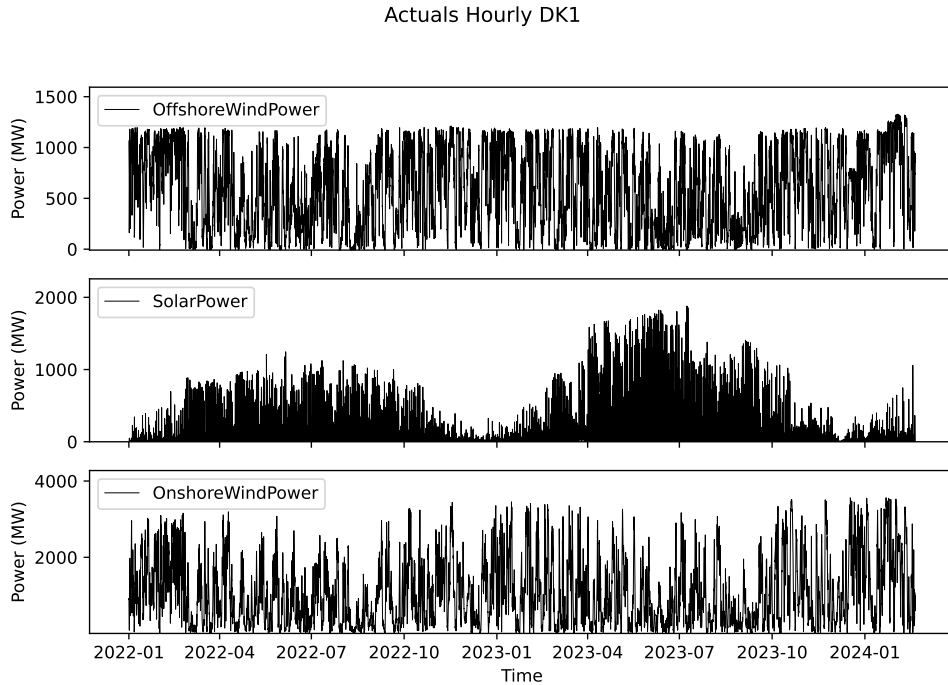


Figure 2.2. The actual hourly observations from Solar, Offshore Wind and Onshore Wind Power Production in DK1

day-ahead forecast from Energi Data Service.

We will take a good look at this figure, as it provides us with many insights, that we later can use. We will note that this figure shows a representative sample of the data regarding the following comments.

The first thing we notice is that Energi Data Service’s day-ahead forecast on the solar power always undershoots both the high probability ensemble, but even more concerning the actual observations. We see in most cases that the ensembles encapsulates how much solar power is generated.

Regarding the offshore wind power it seems that the ensembles can not capture the maximum output from the offshore wind farms. Here the day-ahead forecast is usually better than the ensembles.

In the onshore wind power we notice both better max and min values compared to

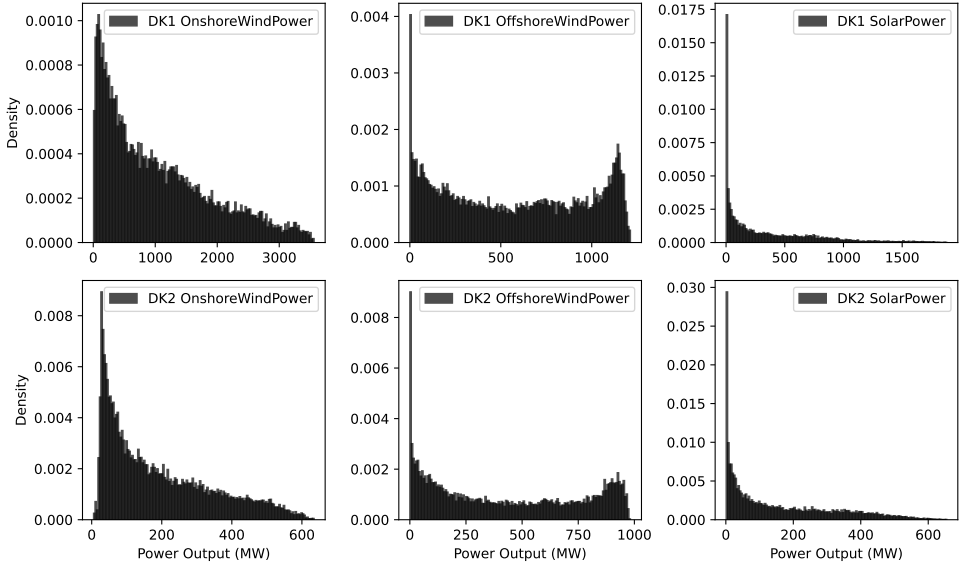


Figure 2.3. Histogram plots of the raw actual observations. Here 0-observations are omitted from the Solar power, to make the plot more readable.

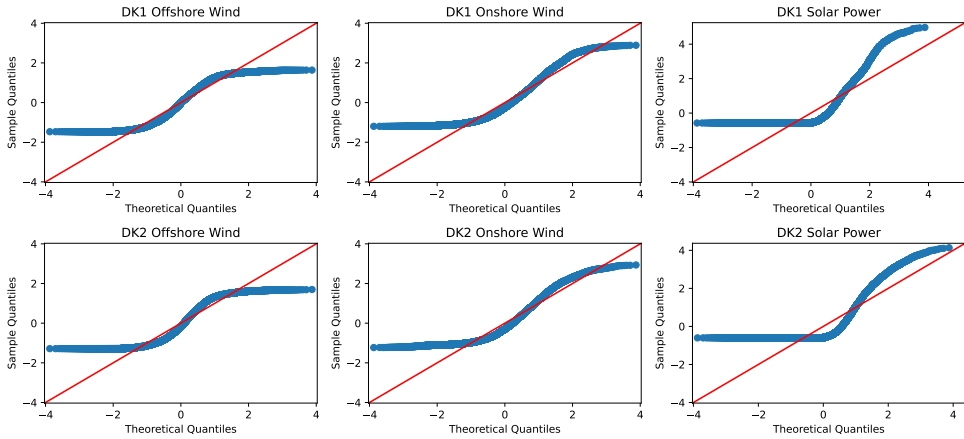


Figure 2.4. QQplots of our actual observations.

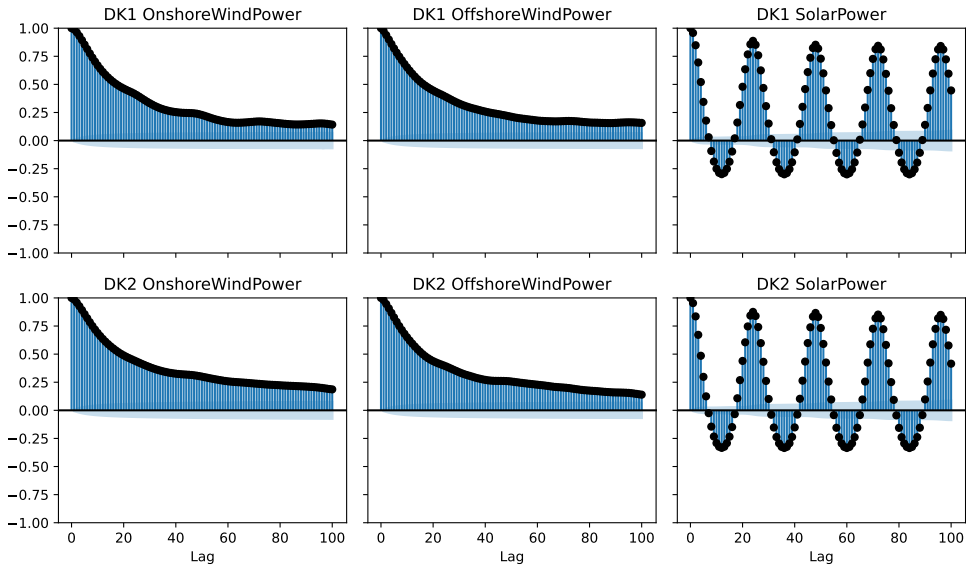


Figure 2.5. ACF plots of the raw actual observations.

offshore, but also that the ensembles, and especially the high probability ensemble, are very close to the actual observations. This can generally be accounted to that the onshore wind power production is much more stable, and thus can be better predicted, see (Clark, 2024).

In general it seems like there is room for improvement, especially in the solar and offshore wind power prediction. We will have this in mind when considering the results in Section 4.

In Figure 2.2 we see that a small bump in the maximum value is seen at the beginning of year 2024. The reason was confirmed by e-mail correspondence with Energinet, where they noted that the cause was the new offshore wind mill park "Vesterhav Syd" (Danish: "havmølleparken Vesterhav Syd"). To correct for this we down-scaled the entire period after the installation, so the maximum of the offshore wind power matched than in the training set.

Lastly, we will quickly define what we in this thesis mean by *quantile* and *percentile*. Even though the literature differs a lot in the definition and actual use of these terms, we define quantile to be values that split a data group into *equal* sized groups. Percentiles are quantiles that divide the data into 100 equally sized groups. Thus, if we then have the data points consisting of the integers from 1 to 200, we would call the data point 50 the 25% quantile or the 25th percentile.

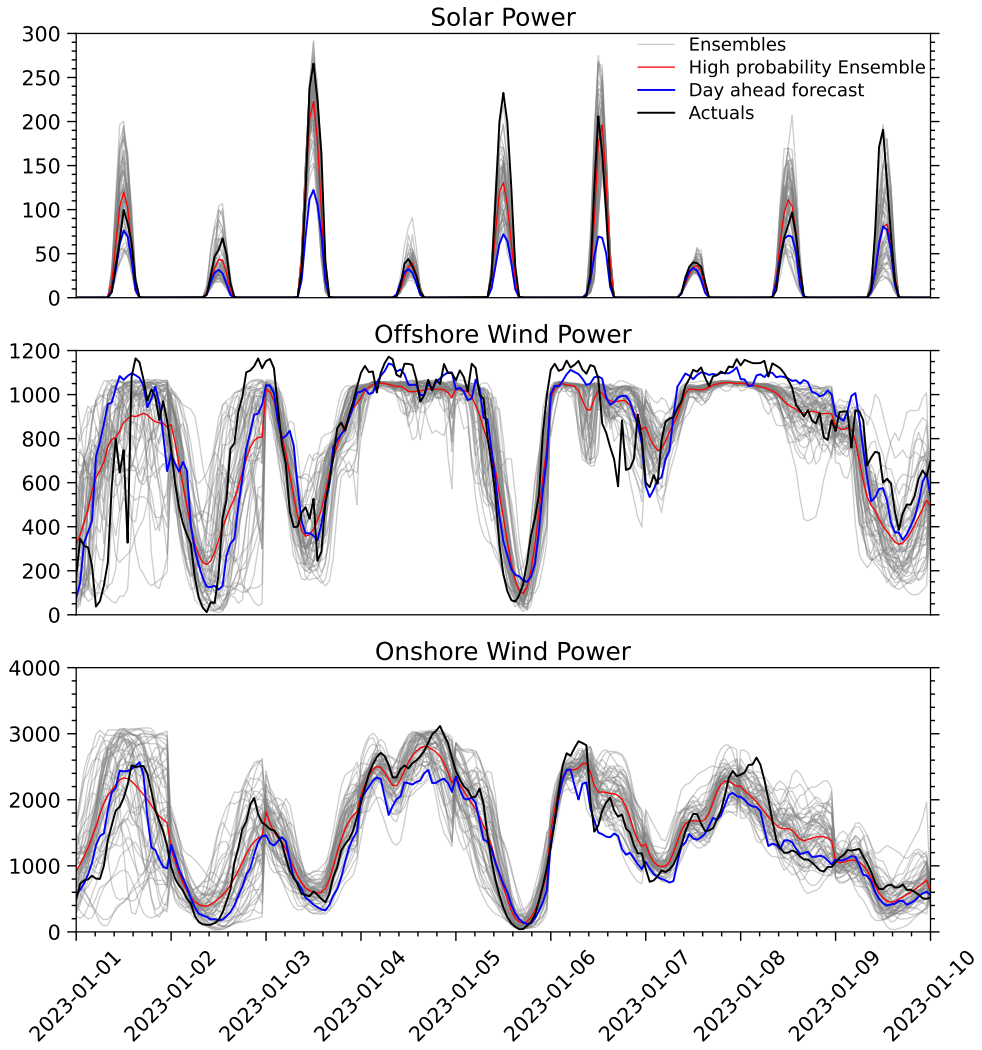


Figure 2.6. The 50 ensembles in grey, alongside the high probability ensemble in red, Energi Data Service's Day ahead forecast in blue and the actual observations from Energi Data Service in black for the first 10 days of 2023.

CHAPTER 3

Methods & Theory

The first thing we will cover in this section is quantile regression. The reason our focus fell here, is explained partly by the P90 requirement and partly due to need for quantile estimates in the risk management side of energy trading strategies.

3.1 Quantile regression

The original paper on quantile regression is (Koenker & Bassett Jr., 1978) with an updated book published by the same author in 2005, (Koenker, 2005), and a modernized version from 2018, (Giles et al., 2018).

We can no longer, as is the case in linear regression, write the closed analytic form for quantile regression due to the absolute loss function. This is the reason that we need linear programming to find the optimum for quantile regression.

In the previous sections we found that we are interested in both the actual mean prediction, but also the quantiles. We start off by explaining vanilla quantile regression.

3.1.1 Simple Quantile Regression

Let Y be a real-valued random variable with $F_Y(y) = P(Y \leq y)$. The τ 'th quantile of Y is then given by

$$q_Y(\tau) = F_Y^{-1}(\tau) = \inf \{y : F_Y(y) \geq \tau\}, \quad (3.1)$$

where $\tau \in (0, 1)$.

We assume that the reader is familiar with Least Squares minimization, and the ideas behind it. Quantile regression, is in fact very similar, in the sense that the only thing

different is its loss function. Instead of the usual MAE or MSE loss function, we, for quantile regression, use the following loss function as $L_\tau(m) = m(\tau - \mathbb{I}_{(m < 0)})$, where $\mathbb{I}$ is an indicator function and $m \in \mathbb{R}$, (Koenker, 2005). If we minimize the expected loss of $Y - f(x)$ with respect to $f(x)$, a specific quantile τ is found

$$q_Y(\tau) = \arg \min_{f(x)} \mathbb{E}(L_\tau(Y - f(x))) \quad (3.2)$$

$$= \arg \min_{f(x)} \left\{ (\tau - 1) \int_{-\infty}^{f(x)} (y - f(x)) dF_Y(y) + \tau \int_{f(x)}^{\infty} (y - f(x)) dF_Y(y) \right\}. \quad (3.3)$$

This can be shown by computing the derivative of the expected loss with respect to $f(x)$ via an application of the Leibniz integral rule, setting it to 0, and isolating $f(x)$. In the Appendix, Section A.2.1, we find

$$F_Y(q_\tau) = \tau,$$

and thus $q_\tau = F^{-1}(\tau)$. If the solution $q_\tau = F^{-1}(\tau)$ is not unique, then we will take the smallest such solution to obtain the τ 'th quantile of the random variable Y , (Koenker, 2005).

We have in Figure 3.1 illustrated a few examples of how this loss function looks. It is for apparent reason also called the check-mark loss.

For a sample the loss function can be written for a single point, y , as

$$\begin{aligned} E &= y - f(x) \\ L_\tau(E) &= \begin{cases} \tau E, & E > 0 \\ (1 - \tau)(-E), & E < 0 \end{cases} \end{aligned} \quad (3.4)$$

Here E is the error term, and L_τ is the loss function for the quantile τ , that we also defined earlier. For n data points, we thus have that

$$\begin{aligned} \hat{q}_\tau &= \arg \min_{f(x_i) \in \mathbb{R}, \forall i} \sum_{i=1}^n L_\tau(y_i - f(x)), \\ &= \arg \min_{f(x_i) \in \mathbb{R}, \forall i} \left[(1 - \tau) \sum_{y_i < \tau} (y_i - f(x)) + \tau \sum_{y_i \geq \tau} (y_i - f(x)) \right], \end{aligned}$$

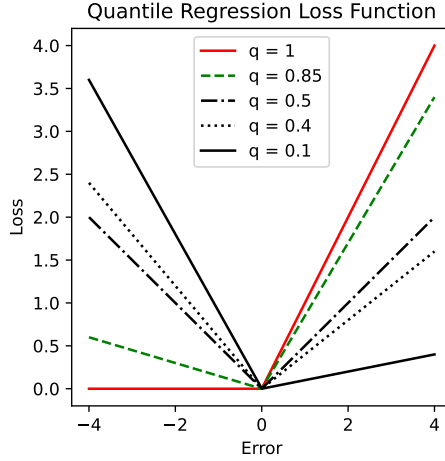


Figure 3.1. Illustration of the quantile loss function, L_q , for errors, E , in the range $[-4, 4]$ and for four different quantiles (0.1, 0.4, 0.5, 0.9).

where the function L_τ is the check mark loss function. We note here again, that this is a linear optimization problem. This will be important, once we describe the time adaptive approach to quantile regression.

We can define $S = (1 - \tau) \sum_{y_i < \tau} (y_i - f(x)) + \tau \sum_{y_i \geq \tau} (y_i - f(x))$ to be the loss function, and we note that this is convex. On a bounded domain, the minimum therefore exists but is not necessarily unique. Note, that in the generalized version above, $f(x)$ takes values in all of $\mathbb{R}$, but we only allow $f(x)$ to take values in a subset $U \subsetneq R$, where U is a bounded domain e.g. $U = [0, \max(\text{WindPower})]$, and thus the minimum exists in practical applications.

In a more general setting, consider the set $\mathcal{J} = 1, 2, \dots, N$ and let $\mathcal{H}$ represent the K -element subsets of $\mathcal{J}$, where K is the number of columns in the matrix $\mathbf{X}$. Let $B^*(\tau)$ denote the set of solutions to the quantile regression problem defined by (3.3). Define $H = \{h \in \mathcal{H} \mid \text{rank } \mathbf{X}(h) = K\}$. The following theorem, attributed to (J. K. Møller, 2006), provides key insights.

Theorem 3.1. *If the matrix $\mathbf{X}$ has rank K , then the set of regression quantiles, $B^*(\tau)$, contains at least one element of the form*

$$\beta^*(\tau) = \mathbf{X}(h)^{-1} \mathbf{y}(h) \quad (3.5)$$

for some $h \in H$. Furthermore, $B^*(\tau)$ is the convex hull of all solutions of this form.

Additionally, if the distribution function F of Y is continuous, then with probability one, β^* is a unique solution if and only if

$$(\tau - 1)\mathbf{e}_K^T < \sum_{t \in \bar{h}} \left(\frac{1}{2} (1 - \text{sign}(y_t - x_t \beta^*)) - \tau \right) x_t \mathbf{X}(h)^{-1} < \tau \mathbf{e}_K^T \quad (3.6)$$

where $\mathbf{e}_K$ is a K -vector of ones and $\bar{h} = \mathcal{J} \setminus h$.

Proof. We will here outline the main ideas behind the proof.

1. **Existence of a Solution:** We need to show that if $\mathbf{X}$ has rank K , then for some $h \in H$, the subset $\mathbf{X}(h)$ is invertible.

Let $\mathbf{X} \in \mathbb{R}^{N \times K}$ be a matrix of rank K . By definition, there exists a K -element subset $h \subseteq \mathcal{J}$ such that $\text{rank}(\mathbf{X}(h)) = K$. This implies that the matrix $\mathbf{X}(h)$ is invertible.

Most of the time we do check if the condition number of $\mathbf{X}$ is too big, if that is the case, as we will explain in Section 5, we sometimes omit some explanatory variables, such that they can no longer be written as a linear combination of each other (e.g. scaling).

We then need to construct a candidate solution $\beta^*(\tau) = \mathbf{X}(h)^{-1} \mathbf{y}(h)$ and verify that it satisfies the necessary conditions for being a quantile regression solution. Since $\mathbf{X}(h)^{-1}$ exists, $\beta^*(\tau)$ is a candidate solution. To show this we need to verify that it fulfills

$$B^*(\tau) = \arg \min_{\beta} \sum_{i=1}^N L_q(y_i - \mathbf{x}_i^T \beta),$$

Since $\mathbf{X}(h)$ is invertible and h corresponds to a K -element subset where $\mathbf{X}(h)\mathbf{X}(h)^{-1} = \mathbf{I}$, we have:

$$\mathbf{X}(h)\beta^*(\tau) = \mathbf{y}(h).$$

Thus, $\beta^*(\tau)$ provides a perfect fit to the subset h of the data. For other indices $i \notin h$, $\beta^*(\tau)$ minimizes the weighted absolute deviations as per the definition of the quantile loss function L_q . Therefore, $\beta^*(\tau)$ satisfies the conditions for being a quantile regression solution, establishing the existence of such a solution in $B^*(\tau)$.

2. **Convex Hull Characterization:** We need to demonstrate that the set of all solutions of the form $\mathbf{X}(h)^{-1} \mathbf{y}(h)$ for $h \in H$ forms a convex set.

Let $\beta^*(h_1, \tau) = \mathbf{X}(h_1)^{-1} \mathbf{y}(h_1)$ and $\beta^*(h_2, \tau) = \mathbf{X}(h_2)^{-1} \mathbf{y}(h_2)$ be two solutions corresponding to subsets $h_1, h_2 \in H$.

For any $\lambda \in [0, 1]$, consider the linear combination:

$$\beta^\lambda = \lambda\beta^*(h_1, \tau) + (1 - \lambda)\beta^*(h_2, \tau).$$

We need to show that β^λ is also a solution in $B^*(\tau)$. Since the quantile regression problem is convex, any convex combination of solutions is also a solution. Hence, β^λ lies in the convex set formed by the solutions $\beta^*(h, \tau)$ for $h \in H$.

Now we need to show that $B^*(\tau)$, being the set of all regression quantiles, is the convex hull of these solutions.

By definition, the convex hull of a set of points is the smallest convex set that contains all the points. For any $\beta^* \in B^*(\tau)$, we can express β^* as a convex combination of solutions $\beta^*(h, \tau)$:

$$\beta^* = \sum_i \lambda_i \beta^*(h_i, \tau),$$

where $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$.

This means that any regression quantile can be written as a combination of the solutions $\beta^*(h, \tau)$ for $h \in H$, confirming that $B^*(\tau)$ is indeed the convex hull of the solutions of the form $\mathbf{X}(h)^{-1}\mathbf{y}(h)$.

Thus, we have shown that $B^*(\tau)$ is the convex hull of all solutions having the form $\beta^*(\tau) = \mathbf{X}(h)^{-1}\mathbf{y}(h)$ for $h \in H$.

3. **Uniqueness Condition:** For a proof on uniqueness we refer to p. 16 in (J. K. Møller, 2006).

□

Thus we have existence of quantiles solutions and if (3.6) holds, and the CDF F_Y is continuous, we have uniqueness.

Remark 3.2. We know from Least squares that the sample mean solves the problem

$$\min_{\mu \in \mathbb{R}} \sum_{i=1}^n (y_i - \mu)^2. \quad (3.7)$$

Thus, if we express the conditional mean of y given x as $\mu(x) = x^\top \beta$, then β may be estimated by solving

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n (y_i - x_i^\top \beta)^2. \quad (3.8)$$

In the case of quantile regression, the problem is similar since, the q 'th sample quantiles, $\hat{\alpha}_\tau$, solves

$$\min_{\alpha \in \mathbb{R}} \sum_{i=1}^n L_\tau(y_i - \alpha), \quad (3.9)$$

This leads us to specifying the conditional quantile function $Q_y(\tau|x) = X^\top \beta(\tau)$, and $\hat{\beta}_\tau$ solving

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n L_\tau(y_i - x_i^\top \beta). \quad (3.10)$$

What we basically are showing here is that the quantile regression produces a generalization of the sample quantile.

For application purposes, it should be noted that the loss for a single point, y , can be rewritten as

$$\begin{aligned} E &= y - f(x) \\ L_\tau(E) &= \max \begin{cases} \tau E \\ (\tau - 1)E \end{cases} \end{aligned} \quad (3.11)$$

The good reason to rewrite it with the max operator is that the TensorFlow library and the tensors it works with, knows how to handle a max operator much better than a conditional statement, when doing backpropagation.

Let us consider the quantile loss function for the $\tau = 0.8$ quantile. Now, the loss penalizes under-estimated ($f(x) < y \Rightarrow E > 0$) predictions $0.8/0.2 = 4$ times more than over-estimated, since

$$L_{0.8} = \begin{cases} 0.8E, & E > 0 \\ (0.2)(-E), & E < 0 \end{cases}$$

Thus the model is more likely to predict higher values, resulting in a negative error, and thus a smaller penalty. The result is that the fitted model on average will estimate $f(x) > y$ in approximately 80% of cases and in 20% of the cases $y > f(x)$.

We will now show, that the quantile loss function actually finds the median, when we are considering $\tau = 0.5$.

Theorem 3.3. *Quantile regression finds the median, when considering the 50th percentile.*

Proof. Consider the case, $\tau = 0.5$, then the loss function reduces to,

$$L_{0.5}(E) = \begin{cases} 0.5E & \text{if } E \geq 0 \\ -0.5E & \text{if } E < 0 \end{cases}.$$

This is exactly the same as $L_{0.5}(E) = (1/2)|E|$, which has the same minimum as $|E|$. It is a well known facts that minimizing the sum of absolute deviations is exactly what the median does. To see this consider $L'_{0.5}(E) = \text{sign}(E)$, then summing over all observations we see that this error is minimized once there is equally many observations on each side of the considered observation, y , which is the exact definition of the median. If the number of observations is even, the median is defined as the mean of the two observations that create an equal number of points on each side of them. $\square$

Note, in Python, `LinearRegression` is a least squares approach minimizing the root mean squared error (RMSE) between the training and predicted targets. In contrast, `QuantileRegressor` with $q = 0.5$ minimizes the mean absolute error (MAE) instead. However, for $q \neq 0.5$, we still need to use the check-mark loss function. Both of these measures is defined in Definition 3.4 and 3.5 respectively.

In Section 3.6 the loss function is extended to be multidimensional.

3.1.2 Non-linear Quantile Regression

Assume your quantile regression model would predict the τ -th quantile of the actual wind observation, Y , as a function of your ensemble forecasts $(X_1, X_2, \dots, X_{50})$. You might start with a model structure similar to this,

$$\hat{Q}_q(Y | X_1, \dots, X_{50}) = \hat{\alpha}(q) + \sum_{j=1}^{50} \hat{f}_j(X_j) \beta_j,$$

where $\hat{f}_j(X_j)$ could be a linear or non-linear function approximated by spline basis functions, depending on the relationship you observe between each ensemble forecast and the actual wind power.

For each independent variable (ensemble forecast), we approximate the non-linear relationship using spline basis functions. This means each $\hat{f}_j(X_j)$ is represented as:

$$\hat{f}_j(X_j) = \sum_{l=1}^{n_j} b_{j,l}(X_j),$$

where the spline-coefficients $b_{j,l}$ should be estimated.

Let $n_j = 7$, i.e. we have 7 knots (including endpoints), for our spline basis functions. Fitting such a model seems to capture the top of the ensembles better, see Figure 3.2. We indeed see that the actuals above the ensembles are very well captured. But we also notices times where the actuals are below the 10% quantile, and even though the ensembles capture this move on the 24th of february 2022, our model does not.

This section is included to be able to show the power of non-linear quantile regression and will serve a purpose in Section 5.3.

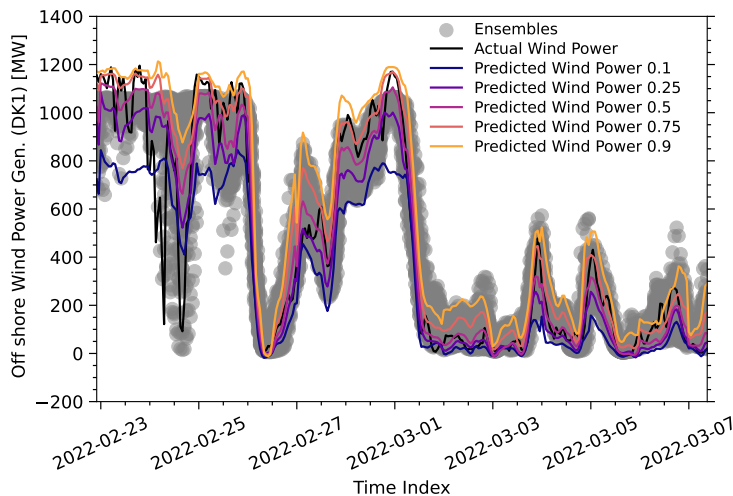


Figure 3.2. $q = 0.1, 0.25, 0.5, 0.75, 0.9$ non-linear quantile model is fitted to the offshore wind power in DK1. The results presented here are in-sample.

3.1.3 Inference for Quantile Regression

Regarding the *inference* of quantile regression, we will just briefly mention that there are many checks one can do, for example Koenker, 2005(Section 3.3.1, 3.3.2, 3.5.1, 3.6) presents Wald Tests for testing for equality between interquantile ranges of two samples or the more general linear hypotheses, which can be used on e.g coefficient tests or joint tests involving several covariates and several distinct quantiles. Furthermore, it includes rank tests for two-sample location shifts and likelihood ratio tests. However, in this thesis, we will not be doing any parameter tests, since we will be studying the performance of the produced quantiles and not the underlying parameters producing the quantiles. That is to say, we will not be doing statistical test

on an individual parameter level, but rather evaluating the entire quantile produced. We refer to Section 3.4.

3.2 Time-Adaptive Quantile Regression

To be able to properly describe the Time-Adaptive Quantile Regression (TAQR) algorithm, we need to introduce some notation. This will be kept to a minimum here, but for more details we refer to (J. Møller et al., 2006). We will be closely following what is outlined in (J. Møller et al., 2006) and (J. K. Møller, 2006) in the coming section.

3.2.1 Notation

Let the positive and negative part of the residuals, $(y - \hat{Q})$, be given by $\mathbf{r}^+ = \{\max(r_i, 0)\}_{i=1, \dots, n}$ and $\mathbf{r}^- = \{\max(-r_i, 0)\}_{i=1, \dots, n}$ respectively. We can then write the LP formulation of the quantile regression problem for the τ 'th quantile as

$$\min_{(\mathbf{r}^+, \mathbf{r}^-, \beta) \in \mathbb{R}_+^{2n} \times \mathbb{R}^K} \left\{ \tau \mathbf{1}^T \mathbf{r}^+ + (1 - \tau) \mathbf{1}^T \mathbf{r}^- : \mathbf{X}\beta + \mathbf{r}^+ - \mathbf{r}^- = \mathbf{y}, (\mathbf{r}^+, \mathbf{r}^-) \in \mathbb{R}_+^{2n}, \beta \in \mathbb{R}^K \right\}, \quad (3.12)$$

as seen in Koenker (2005) p. 10.

Another typical way of writing this can be seen in Appendix, Equation (A.1).

We will adopt the notation from (J. Møller et al., 2006), to be consistent. We will need the following more compact notation, to be able to describe the simplex step in a concise way. We rewrite (3.12) to

$$\min_{(\mathbf{r}^+, \mathbf{r}^-, \beta) \in \mathbb{R}_+^{2n} \times \mathbb{R}^K} \left\{ \mathbf{c}^T \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{y}, (\mathbf{r}^+, \mathbf{r}^-) \in \mathbb{R}_+^{2n}, \beta \in \mathbb{R}^K \right\}, \quad (3.13)$$

where

$$\mathbf{c} = \begin{bmatrix} \mathbf{0}_K \\ \tau \mathbf{e} \\ (1 - \tau) \mathbf{e} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \beta \\ \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} \quad \mathbf{A} = [\mathbf{X}, \mathbf{I}, -\mathbf{I}].$$

Note, a solution to (3.13) must lie on the boundary of the feasible region (a vertex) i.e. at a point, where the number of active constraints are equal to $\dim(\mathbf{x})$, (J. K. Møller, 2006).

We note that at least $n + K$ elements of $\mathbf{x}$ must be zero. In the optimum we will have that at least K elements of $\mathbf{r}$ is zero, since $\beta \in \mathbb{R}^K$. If we then imagine we do the "splitting" of $\mathbf{r}$ into $\mathbf{r}^+$ and $\mathbf{r}^-$ after this we have that K elements in each are zero i.e. $2K$ elements are zero. Now, the remaining $2(n - K)$ elements of $\mathbf{r}$ has by definition $n - K$ zeros. Thus the total number of zeros must be at least $2K + n - K = n + K$.

We have now argued that all the zeros should be in $\begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix}$, but by definition there will always be n zeros, thus there are K zeros in $\mathbf{r}$. The indices of these *original*¹ K zeros are collected in the index set h , which we will be using a lot. h is the index set which characterizes the solution of to the quantile regression problem. We now use this index set to define new index sets to describe $\mathbf{x}$. Let $\mathcal{B}$ be given such that,

$$\mathbf{x}_{\mathcal{B}} = \begin{bmatrix} \beta \\ |\mathbf{r}(\bar{h})| \end{bmatrix}, \quad (3.14)$$

where $\bar{h}$ is the complement of h , i.e. the indices of the non-zeros and thus $\mathbf{x}_{\mathcal{B}} \geq 0$ and we choose $\mathcal{C}$ such that $\mathbf{x}_{\mathcal{C}} = [\mathbf{r}(h)] = 0$. $\mathbf{x}^k$ will denote the k 'th iteration through the simplex algorithm.

We now split $\mathbf{A}$ into columns that are multiplied by $\mathbf{x}_{\mathcal{B}}$ and $\mathbf{x}_{\mathcal{C}}$. Let $\mathbf{B} = \begin{bmatrix} \mathbf{X}(h) & \mathbf{0} \\ \mathbf{X}(\bar{h}) & \mathbf{P} \end{bmatrix}$.

We get, that the constraints at a solution, can be written as

$$\mathbf{B}\mathbf{x}_{\mathcal{B}} = \begin{bmatrix} \mathbf{X}(h) & \mathbf{0} \\ \mathbf{X}(\bar{h}) & \mathbf{P} \end{bmatrix} \mathbf{x}_{\mathcal{B}} = \begin{bmatrix} \mathbf{y}(h) \\ \mathbf{y}(\bar{h}) \end{bmatrix},$$

where $\mathbf{P}$ is a diagonal matrix with diagonal elements

$$\mathbf{p} = \text{sign}(\mathbf{r}(\bar{h})) = [\mathbb{I}(r_i \geq 0)] - [\mathbb{I}(r_i < 0)].$$

For clarity and later reference, we will let $\mathbf{X}$ denote all the ensembles. $\mathbf{X}_n \in \mathbb{R}^N$ denote an ensemble and all the associated time points $t = 1, 2, \dots, N$. $\mathbf{X}^t \in \mathbb{R}^{50}$ denote all the quantiles to a specific time point. One specific scalar data point can then be denoted X_n^t , where $n \in \{1, \dots, 50\}$, and t is a specific time point in the index set T , which covers all of the data.

Now that we have the notation in place, let us continue to the algorithm.

¹I.e. those not stemming from the definition of $\mathbf{r}^+, \mathbf{r}^-$. These will be found as an initial solution as described in Section 3.2.2.

3.2.2 Algorithm

The simplex method for quantile regression described in (J. Møller et al., 2006) is also found in numerous other articles with different notation, see e.g. (Van de Geer, 2003, Alhamzawi et al., 2011, Hubert et al., 2004). In this section we will primarily show things, that we will cover in the code that will be discussed in Section 4.4, where our new Python implementation is presented.

The basic assumption relies on the idea that we need an initial vertex in the constraint polyhedron to start with.

A 2D example of some simplex steps for a simple example can be seen in Appendix, Figure A.1, where also a download link is available for the .gif-file, to be able to see the steps animated.

Previously, this was accomplished using the `rq` command in R (J. Møller et al., 2006). In the Python implementation that we will be presenting later, we initially attempted to use the `QuantileRegressor` class from `sklearn`. However, the results were not as good as those obtained with `rq`, and the approach proved to be impractical due to poor initialization and slow convergence. As a result, we opted to run the `rq` R-code directly from within Python using the `subprocess` module. This method is described in Section 4.4. In the simplex algorithm we need a descent direction, $\mathbf{h}$, step size, α , a reduced cost $\mathbf{d}$. We introduce $\mathbf{g}$ to make notation lighter. We present the formulas for the main ingredients here:

$$\begin{aligned} \mathbf{d} &= \mathbf{c}_C - \mathbf{C}^T \mathbf{g} \quad , \quad \mathbf{g} = \mathbf{B}^{-T} \mathbf{c}_B \\ \mathbf{h} &= \mathbf{B}^{-1} \mathbf{C}_{:,s} \quad , \quad \alpha = \min \{ \sigma_1, \dots, \sigma_m \} , \end{aligned} \quad (3.15)$$

with σ_j given by

$$\sigma_j = \begin{cases} \left(\mathbf{x}_B^{(k)} \right)_j / h_j & \text{if } h_j > 0, \\ \infty & \text{if } h_j \leq 0. \end{cases}$$

Now we are ready to do a standard step in the simplex algorithm, but we will first make some notes regarding stopping criteria. There are a few ways the algorithm can terminate. 1) We have an optimal solution: If $\mathbf{d} \geq \mathbf{0}$, we terminate. 2) If $\mathbf{x}^k$ is not optimal, we choose a negative element d_s in $\mathbf{d}$, and change a (zero) element of $(\mathbf{x}_C)_s$ of $\mathbf{x}_C$. The basic variables, $\mathbf{x}_B$ are decreased in direction $\mathbf{h}$, we continue until we meet a new vertex, which corresponds to a variable constraint. Now we have the step size, and the objective $\mathbf{c}^T \mathbf{x}$ is decreased by αd_s . 3) If $\alpha = +\infty$, the problem is unbounded. This never happens in the case of quantile regression, see (J. Møller et al., 2006). 4) If $\alpha = 0$, and then there is no step to be taken. However, we still need the algorithm to continue. A simple fix to this is to follow Bland's rule, which

states that if we choose the entry with the smallest index, we are sure to converge, see e.g. (Avis & Chvátal, 1978). However, in this custom regime we are in, there is a better way. Update

$$\begin{aligned} \mathbf{x}_{\mathcal{B}}^{(k+1)} &= \mathbf{x}_{\mathcal{B}}^{(k)} - \alpha \mathbf{h} \\ \left(\mathbf{x}_{\mathcal{B}}^{(k+1)} \right)_q &= \alpha, \end{aligned} \tag{3.16}$$

where $q = \arg \min_j \sigma_j$. We also do the swap²

$$\begin{aligned} temp &= \mathcal{B}_q, \\ \mathcal{B}_q &= \mathcal{C}_s, \\ \mathcal{C}_s &= temp. \end{aligned}$$

Furthermore, we keep track of where we have been. This eliminate the chance of visiting vertices we already have visited again.

The most expensive part of this is calculating the inverse of $\mathbf{B}$, but this can be done efficiently with

$$\mathbf{B}^{-1} = \begin{bmatrix} \mathbf{X}(h)^{-1} & \mathbf{0} \\ -\mathbf{P}\mathbf{X}(h)\mathbf{X}(h)^{-1} & \mathbf{P} \end{bmatrix}, \tag{3.17}$$

which gets more and more important as the number of observations grow.

We present the steps of the algorithm in pseudocode in Algorithm 1.

Algorithm 1 The Simplex Algorithm

```

while  $i < \text{max\_iter}$  do
    Compute  $\mathbf{d}$ . ▷ Step 1
    if  $\mathbf{d} \geq \mathbf{0}$  then
        Return.  $\mathbf{x}$  is optimal.
    else
        Choose  $s$  such that  $d_s < 0$ .
    end if
    Compute  $\mathbf{h} = \mathbf{B}^{-1}\mathbf{A}_{:, \mathcal{C}(s)}$ . ▷ Step 2
    if  $\mathbf{h} \leq \mathbf{0}$  then
        Return. The problem is unbounded.
    end if
    Compute  $\alpha$  and choose  $q$  such that  $\sigma_q = \alpha$ .
    Swap  $\mathcal{B}(q)$  and  $\mathcal{C}(s)$ .
    Set  $\mathbf{x}_{\mathcal{B}} := \mathbf{x}_{\mathcal{B}} - \alpha \mathbf{h}$ 
    Set  $(\mathbf{x}_{\mathcal{B}})_q := \alpha$ 
     $i++$ 
end while

```

²This can be accomplished in Python with the neat trick $\mathbf{a}, \mathbf{b} = \mathbf{b}, \mathbf{a}$.

We refer to (J. Møller et al., 2006) for more information on each step. For completeness the most important calculations in Algorithm 1, can be seen in Appendix A.2.2.

We will now turn our focus to handle the problem of oscillation between two variables that need to leave the basis, but is in the index set h .

How to handle oscillations? The updating procedure.

Updating the design matrix is tricky. There are many possible solutions, and more thought should be given to this, we refer to the section on further work, Section 5.3. We will here note that, how we do the updating is not relevant to the success of the simplex method.

Algorithm 2 Updating Design Matrix and Simplex Steps

- | | |
|---|----------|
| Decide which row, l , must leave the design matrix. | ▷ Step 1 |
| if $l \in h_t$ then | |
| Take one simplex step such that $l \notin h_t$. | ▷ Step 2 |
| end if | |
| Update $\mathbf{X}, \mathbf{P}, h$, and $\mathbf{c}_B$. | ▷ Step 3 |
| Perform the simplex steps needed to get to the optimal solution at time $t + 1$. | |
-

In Algorithm 2, Step 1, we have simply chosen to let the oldest observation leave the design matrix. This is a natural choice, since we do not work with splines.

Regarding Step 2 in Algorithm 2: If the leaving variable is in the index set h , we can not remove it, since that would mess up our basis. Let the leaving variable be l . We change the objective function for the new simplex step. We set $\mathbf{c}_{\{l, l+N\}} = \mathbf{0}$. This in turns mean that we change the l 'th element in $\mathbf{d}$, to $\mathbf{d}^l = \begin{bmatrix} -\mathbf{g}^{(l)} \\ \mathbf{g}^{(l)} \end{bmatrix}$. Now we have the descent direction for the new objective function, and thus our index s , which is what we use for the swap. This can cause problems, since we do not care about the condition number of $\mathbf{X}(h)$ in this case, see Section 5.3.

Again, for completeness we refer to Appendix A.2.3 for calculations regarding Step 3 in Algorithm 2.

This is a brief explanation of the TAQR implemented in MATLAB in (J. Møller et al., 2006), and in Python in this thesis. The Python implementation will briefly be presented in Section 4.4.

3.3 General Error Measures and Statistical Tools

In this section we will define well known quantities, which we will use to score, compare and discuss the results in Section 4 and 5. The point of defining these is that the thesis becomes self-contained.

Definition 3.4 (Root Mean Square Error (RMSE)). Assume $y \in \mathbb{R}^N$ is the true observations, and $\hat{y} \in \mathbb{R}^N$ is our predictions. The RMSE is then defined as, see (Karunasingha, 2022):

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}.$$

Definition 3.5 (Mean Absolute Error (MAE)). Assume $y \in \mathbb{R}^N$ is the true observations, and $\hat{y} \in \mathbb{R}^N$ is our predictions. The MAE is then defined as the mean of the absolute error e.g. the Manhattan distance divided by the sample size, see (Rajawat et al., 2022):

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|.$$

Definition 3.6 (Autocorrelation Function (ACF)). Let $\{X(t)\}$ be a stochastic process and t_1, t_2 two points in time, then the ACF is defined as

$$\rho_{XX}(t_1, t_2) = \rho(t_1, t_2) = \frac{\gamma_{XX}(t_1, t_2)}{\sqrt{\sigma^2(t_1) \sigma^2(t_2)}},$$

where $\gamma_{XX}(t_1, t_2) = \gamma(t_1, t_2) = \text{Cov}[X(t_1), X(t_2)]$ is the autocovariance function. If $\{X(t)\}$ takes on discrete values, $\{X(1), X(2), \dots, X(n)\}$, and we know the true mean, μ , and variance σ^2 , we can calculate the unbiased estimate of ACF as

$$\hat{\rho}(k) = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} (X_t - \mu)(X_{t+k} - \mu),$$

for any value positive integer $k < n$, see (Nounou & Bakshi, 2000).

Definition 3.7 (Partial Autocorrelation Function (PACF)). The PACF, ϕ_{kk} , is defined as, see (Madsen, 2007):

$$\phi_{kk} = \text{Cor}[Y_t, Y_{t+k} \mid Y_{t+1}, \dots, Y_{t+k-1}].$$

The PACF can be seen as the ACF between Y_t and Y_{t+k} given all the observations between Y_t and Y_{t+k} .

We refer to (Madsen, 2007) for properties of the ACF and PACF.

3.4 Quantifying the Performance of Quantiles

There is no universally accepted method for evaluating quantiles, so we will present several measures. The main issue with these measures is the lack of statistical tests to determine if the new model is "good enough." This means we cannot easily assess how close we are to the true model (J. K. Møller, 2006).

We will here assume that our data has been split into a training and test set, see Section 3.7.3 for more information on how this have been split. We will apply these scores to the test set e.g. out of sample data, that the models have not seen before. The reason is, that usually the scores will be better in the training set, which the parameters are fitted to, and thus one would need to take into account the number of model parameters.

If we consider the subset of data points that is under a given quantile, q , then the fraction should be (at least) close to the quantile value i.e. for the 70th percentile, approximately 70% of the data should be below this quantile. We give a formal definition in Definition 3.8.

Definition 3.8 (Reliability). *For our data $\mathbf{y} \in \mathbb{R}^N$ let $\mathcal{A}$ be a perturbation of the index set $\{1, 2, \dots, N\}$, s.t. $y_{\mathcal{A}_1} \leq y_{\mathcal{A}_2} \leq \dots \leq y_{\mathcal{A}_N}$ i.e. $\mathbf{y}_{\mathcal{A}}$ is the order statistics of $\mathbf{y}$. For a given value of y let*

$$p(y) = \begin{cases} 1 & \text{if } y_{\mathcal{A}_{(1)}} > y \\ \operatorname{argmax}_i \{y_{\mathcal{A}_{(i)}} \leq y\} & \text{otherwise} . \end{cases}$$

By using $p(y)$, you can effectively gauge where y stands among the observations $\mathbf{y}$ in terms of their sorted order.

Sharpness measures the separation of symmetric quantiles around the median (0.5). For reliable intervals, we aim for these intervals to be as narrow as possible. In this context, the Interquartile Range (IQR) is used, see Definition 3.9.

Definition 3.9 (Interquartile Range (IQR)). *IQR is defined as the difference between the 75th and 25th percentiles, e.g.*

$$IQR = q_{75} - q_{25}.$$

In general we are interested in the mean IQR, denoted $\overline{IQR}$.

Notice, that we (hopefully) observe that the IQR is positive, however, if the crossing problem is very pronounced, then we may see $IQR < 0$ for some data points along the quantiles. Furthermore, we will introduce $\mathbb{E}(IQR|IQR < 0)$, $\mathbb{E}(IQR|IQR > 0)$ and $\min(IQR)$ as a way to gauge the significance of any potential crossings.

3.4.1 CRPS- & Skill Score

First we will present the *Continuous Ranked Probability Score* (CRPS) definition, and later explain how we actually implement it in Python.

Definition 3.10 (Continuous Ranked Probability Score (CRPS)). *Consider the univariate time series y , and let probabilistic (ensemble) forecasts be given in the form of a CDF F_Y . The continuous ranked probability score is then defined as*

$$CRPS(F, y) = \int_{-\infty}^{\infty} (F(u) - \mathbb{1}(u \geq y))^2 du \quad (3.18)$$

The multivariate generalization of CRPS is simply given by $CRPS(F, \mathbf{y}) = \int_{-\infty}^{\infty} (F(\mathbf{u}) - \mathbb{1}(\mathbf{u} \geq \mathbf{y}))^2 d\mathbf{u}$.

We are interested in the average CRPS, denoted $\overline{CRPS}(f, \mathbf{y})$.

Often it is a good idea to split up the integral into the following two parts

$$CRPS(F, y) = \int_{-\infty}^y F(u)^2 du + \int_y^{\infty} (F(u) - 1)^2 du. \quad (3.19)$$

This can alleviate some pain when doing the integral.

In (3.18), we only scored a single observation/forecast pair. But to reliably evaluate a forecast model, you need to average these scores across many observations. Fortunately, all scoring rules in the Python package `properscoring` happily accept and return observations as multi-dimensional arrays. We will use the `crps_ensemble` function from the `properscoring` module.

Once you calculate an average score, S , it is often useful to normalize them relative to a baseline forecast, to calculate a so-called "skill score". In Definition 3.11, we will define what we mean by skill score in this thesis³.

Definition 3.11 (Skill Score). *Assume S_b is the score for your baseline model. S_m is your new model. The Skill Score is then,*

$$SS_{b \setminus m} = \frac{S_m}{S_b}.$$

³This (may) differ from the usual skill score. However, we went with this approach so the improvement in score could be compared almost as a "percentage score".

Since the measures we will be using in this thesis to quantify the quality of our forecast is some transformation of the Lebesgue measure, lower is always better. What we mean by this, is that the measures we are using is a loss function of some sort, in the sense that it penalizes distance to the true value. This means that if we have a perfect model, our skill score would be 0. The skill score would be 1 if the model performs the same as the baseline and > 1 if the new proposed model is worse than the baseline. This definition differs from the usual skill score, defined as $\text{SkillScore}_{\text{usual}} = 1 - \frac{\text{Model}_{\text{new}}}{\text{Model}_{\text{baseline}}}$.

Properties of CRPS. The CRPS integrates the squared differences between the forecast CDF and the step function representing the observed value, thus indirectly emphasizing accuracy around the median of the forecast distribution. A lower CRPS indicates that the forecast distribution F places higher probability mass closer to the actual observation y , rewarding forecasts that are more centered around the median when the observation aligns with it.

Example 3.12. Let us assume our true model is a $\text{Beta}(2,5)$, and for illustration purposes let the wrong/estimated model be $\text{Beta}(1,5)$. Figure 3.3 illustrates that observations close to the median (0.264 for the true model), shown in green, is rewarded. This example is inspired from (Bjerregård et al., 2021). We see that when the two integrands from (3.19) are intersecting at the median, this is when the CRPS is minimized.

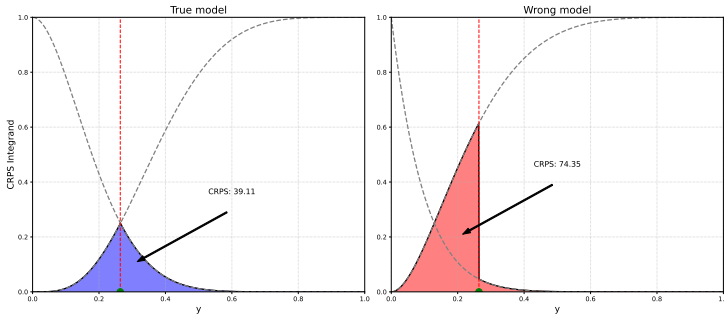


Figure 3.3. CRPS score for true $\text{Beta}(2,5)$ and wrong model $\text{Beta}(1,5)$.

3.4.2 Variogram Score

We use the Variogram score to detect correlation structures in our data. The variogram score stems from the *variogram* also referred to as the *structure function* in geostatistics, see (Scheuerer & Hamill, 2015). Here they study the quantity

$\gamma_2(i, j) = \frac{1}{2} \mathbb{E}_{\mathbf{Y}} |Y_i - Y_j|^2$. Let the (multivariate) distribution of $\mathbf{Y}$ have finite second moments, and let $\mu_i = \mathbb{E}(Y_i)$, $\sigma_i^2 = \text{var}(Y_i)$ and $\rho_{ij} = \text{corr}(Y_i, Y_j)$. Then it is easy to show that,

$$\begin{aligned} \mathbb{E} |Y_i - Y_j|^2 &= \mathbb{E}(Y_i^2 + Y_j^2 - 2Y_i Y_j) \\ &= \mu_i^2 + \sigma_i^2 + \mu_j^2 + \sigma_j^2 - 2(\text{COV}(Y_i Y_j) + \mu_j \mu_i) \\ &= \mu_i^2 + \sigma_i^2 + \mu_j^2 + \sigma_j^2 - 2(\sigma_i \sigma_j \rho_{ij} + \mu_j \mu_i) \\ &= (\mu_i - \mu_j)^2 + (\sigma_i^2 - 2\sigma_i \sigma_j \rho_{ij} + \sigma_j^2). \end{aligned}$$

More generally, we can consider $\gamma_p = \frac{1}{2} \mathbb{E} |Y_i - Y_j|^p$. For $p = 1$ and $p = 0.5$ the score is called madogram and rodogram respectively. Variograms of order p can in general be defined whenever the p th absolute moment exist (Scheuerer & Hamill, 2015). We will now define the variogram score of order p , that we will be using here.

Definition 3.13 (Variogram Score). *Consider the multivariate time series $\mathbf{y}_t$ and let probabilistic (ensemble) forecasts be given in the form of a density $f_{\mathbf{X}}$, then the variogram score of order p is*

$$\text{VarS}_p(f_{\mathbf{X}}, \mathbf{y}) = \sum_{i=1}^{k-1} \sum_{j=i+1}^k w_{ij} (|y_i - y_j|^p - \mathbb{E}[|X_i - X_j|^p])^2, \quad (3.20)$$

where w_{ij} are nonnegative weights⁴. For implementation purposes the expected pairwise difference for all horizon-pairs (i, j) can be estimated as

$$\mathbb{E}[|X_i - X_j|^p] \approx \frac{1}{m} \sum_{l=1}^m |x_i^{(l)} - x_j^{(l)}|^p. \quad (3.21)$$

Implementing this scoring algorithm has time complexity $\mathcal{O}(mk^2)$, where k is the length of our forecast and m is the our ensemble size. It is fair to say that this scoring rule is much faster to calculate than e.g. the CRPS (Bjerregård et al., 2021).

In Bjerregård et al. p. 5-6, they note that $p = 0.5$ is a good choice, at least for gaussian distributions. When looking at our pairwise differences between ensembles, we conclude that they are close enough to gaussian, so that we continue with $p = 0.5$. A t -distribution is actually a better fit. Both the fitted gaussian- and t -distribution can be found in Appendix, Figure A.2 and A.3. Furthermore, they argue at p.6-7 that when the mean and variance is correctly modeled (or known), then getting the correct correlation structure minimizes the variogram score.

For an example of the Variogram properties, we refer to the paper by Bjerregård et al. (2021), Section 2.1.3, *Example 3: a closer look at the properties of VarS*.

⁴In this thesis $w_{ij} = 1, \forall i, j$.

3.4.3 Quantile Score

In this thesis, we will define the *Quantile Score* (QS) as the check mark loss function from (3.4) applied to our predictions compared to the actual observations. We here define the quantile skill score

Definition 3.14 (Quantile Score). *Assume the actual observations are $y \in \mathbb{R}^N$, and our predictions are $f(x) \in \mathbb{R}^N$, then the Quantile Score for the quantile τ is defined as*

$$QS(y, f(x)) = \sum_{i=1}^N L_{\tau}(E_i),$$

where L_{τ} is defined in (3.4) and $E_i = y_i - f_i(x)$. The *Quantile Skill Score*, is readily defined as $QSS = QS_{new}/QS_{baseline}$.

Thus, if the new corrected model is better, we will observe $QS_C < QS_R$, resulting in $QSS < 1$. In the results, Section 4, we will sometimes use "QSS" in the legend name, when we actually only mean the Quantile Score (i.e. not the QSS), this will be clear from the caption of the figure.

In assessing quantiles, various measures can be considered, including sharpness, reliability, IQR, CRPS, Variogram, and Quantile Score. We have evaluated all these scores for many of the ensembles but determined that CRPS, Variogram, and Quantile Score align best with the perceived quality of the ensembles. These measures also translate most effectively to meaningful interpretations, such as those in the case studies presented in Section 5. Consequently, we will only present these measures in the Results.

3.5 Neural Networks

3.5.1 Feed Forward Neural Network

Feed Forward Neural Networks (FFNNs) can be described as a mapping of our data $\mathbf{x} \mapsto \mathbf{y}$, with a function f , i.e. $f(\mathbf{x}) = \mathbf{y}$. Between the input layers and the output layer we introduce weights $\mathbf{w} = \{w_{ij}\}$, from node i to node j . Then our output can be written as $\mathbf{y} = \mathbf{w} \odot \mathbf{x}$, which actually is a linear map. We then introduce non-linear activation functions between the layers. In this report we apply the ReLu and Sigmoid activation function, given as $\text{ReLu}(x) := \max(0, x)$ and $\text{Sigmoid}(x) := \frac{1}{1 + \exp^{-x}}$.

Let us denote a general activation function by $\psi(x)$. The weights and neurons in a FFNN find the gradient to the objective/loss function by backpropagation, which basically is the chain rule of differentiation applied to the weights in the network. Gradient descent is performed in the "loss-landscape" with the Adam optimizer, which is a well renowned optimizer for such problems, see (M. Nielsen, 2015).

For the loss function we of course use the multi dimensional version of the quantile loss function, defined in Algorithm 3.

It is also possible to introduce a bias at each node, which can be helpful if we want to apply the linear map $x \mapsto ax + b$ at each node. We do not want this, since we rather would like a correction e.g. scaling of the ensembles that we have described.

An example of such a FFNN using Sigmoid activation function can be seen in Figure 3.4.

The forward pass through such a network can be written explicitly like

$$\mathbf{y} = \psi_L \left(\mathbf{w}_L^\top \psi_{L-1} \left(\mathbf{w}_{L-1}^\top \cdots \psi_1 \left(\mathbf{w}_1^\top \mathbf{x} + b_1 \right) \cdots + b_{L-1} \right) + b_L \right), \quad (3.22)$$

where, $\mathbf{x}$ is the input vector,

$\mathbf{y}$ is the output vector,

ϕ_i is the activation function for the i -th layer,

$\mathbf{w}_i$ is the weight matrix for the i -th layer,

b_i is the bias vector for the i -th layer,

L is the total number of layers in the network.

TensorFlow is the implementation framework that has been used in this thesis to implement neural networks. It is an end-to-end solution for implementing Machine Learning models in e.g. Python, see (Abadi et al., 2015). We also refer to the very

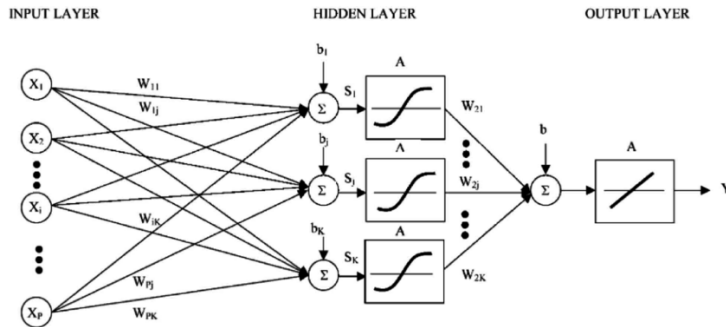


Figure 3.4. Example of a FFNN using Sigmoid activation functions. Picture from (Vuckovic et al., 2002).

well made TensorFlow Playground, see (Smilkov & Carter, 2023).

The choice of using TensorFlow was made since TensorFlow was the only framework where the multidimensional quantile function could be implemented in parallel and the training procedure could be sped up via GPU acceleration on the M1-metal architecture by Apple.

3.5.2 Long Short-Term Memory Network

We will base our explanation of the inner workings of an LSTM on Figure 3.5. The basic idea behind an LSTM network is that we have some sequence of variables in $\mathbb{R}^n$, that make up the long-term memory of this network. In Figure 3.5 this is, as in other literature, called the cell state. The hidden state, or the short-term memory, of the LSTM is what runs in the bottom of each LSTM module. Each module works for each time step. Now we will dive into the each *gate* in the module. The short-term memory concatenated with a user-specified input serves as the input to the Input Gate. We send this through a sigmoid-activation function and multiply with the previous cell state, or the long-term memory. Since the activation function here is a Sigmoid activation function this input gate determines how much of cell state we pass on through this module. This trick of sending the concatenated input through a sigmoid activation function, is used multiple times in an LSTM module. Basically, it

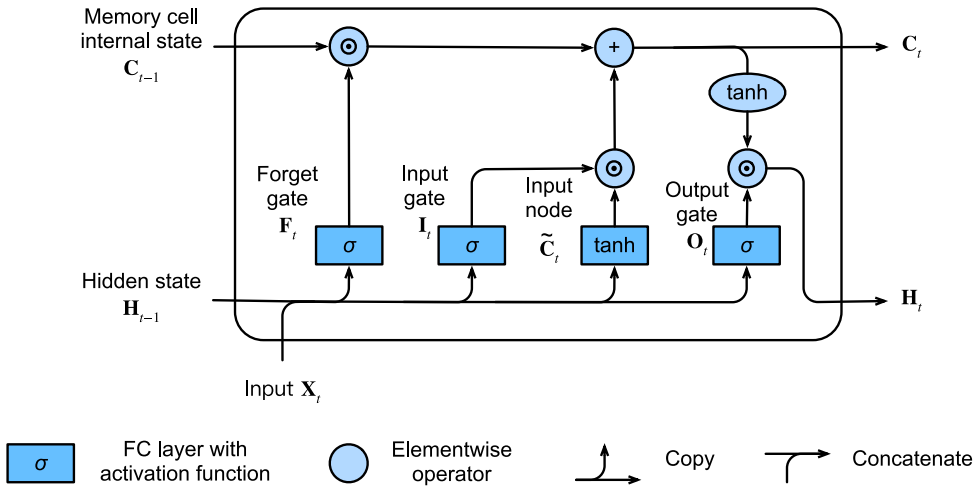


Figure 3.5. General scheme of an Long Short-Term Memory neural networks (LSTM). Picture from (Sinkin, 2024).

allows the network to choose how many % of the input, it is multiplied by, it should remember.

We see this %-trick used again in the Forgotten Gate, where the concatenated input first is sent through the hyperbolic tangent activation function that maps it in the range $[-1, 1]$. We add the output from this multiplication to the cell state.

In the Output Gate the current cell state is sent through the tanh activation function and the %-trick determines how much we output from the module as the hidden state. The cell state is also outputted directly as the long-term memory.

This wraps up the inner-workings of one LSTM module.

We will here present a more mathematical description of the different gates just mentioned.

Each LSTM cell has a cell state C_t and a hidden state h_t . The LSTM cell uses three gates to control the flow of information: The input gate i_t , the forget gate f_t , and the output gate o_t . b_i, b_f, b_o are the biases for the respective gates.

1. Input Gate.

The input gate controls how much of the new input x_t and the previous hidden state h_{t-1} should be used to update the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

2. Forget Gate.

The forget gate controls how much of the previous cell state C_{t-1} should be retained.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

3. Cell State Update.

The cell state is updated by forgetting some part of the previous cell state and adding the new candidate cell state $\tilde{C}_t$, which is modulated by the input gate.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

4. Output Gate.

The output gate controls how much of the cell state should be output as the new hidden state h_t .

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

Connection Between LSTM Cells.

In an LSTM network, the hidden state h_t and the cell state C_t are passed from one LSTM cell to the next in the sequence. This allows the network to maintain information over long sequences.

Training an LSTM Network.

When training an LSTM network, we use backpropagation through time (BPTT) to update the weights W_i, W_f, W_C, W_o and biases b_i, b_f, b_C, b_o based on the error gradients.

This wraps up the mathematical equations governing an LSTM network, see Hochreiter and Schmidhuber, 1997 for the original article.

In Tensorflow we build an LSTM layer in our framework, which consists of an LSTM cell or in Tensforflow lingo, LSTM layer. An LSTM cell is then a collection of `num_units` LSTM modules.

`num_units` can be interpreted as the analogy of a hidden layer from the FFNN.

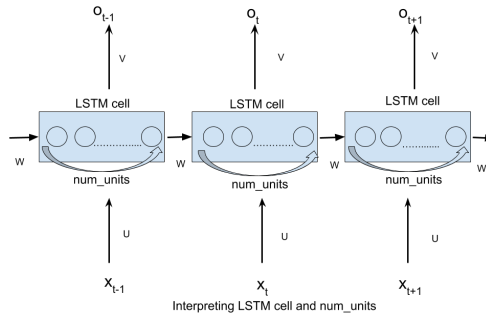


Figure 3.6. Overview of entire LSTM layer. Picture from (Chhabra, 2017).

Given the relatively simple nature of the LSTM architecture, it is impressive how

powerful the network can be. It effectively captures both long-term and short-term trends.

LSTM networks excel in handling sequential data due to their ability to maintain and manage memory over extended periods. Unlike traditional Feed Forward Neural Networks, which process inputs independently, LSTMs have a built-in mechanism to remember previous inputs through their cell state and various gates. This makes LSTMs particularly effective for tasks involving time series, natural language processing, and any context where the order of inputs matters.

FFNNs lack the inherent capability to retain information about previous states, making them less suitable for sequential data.

Since LSTMs have excellent applications in temporal studies, we also apply one in this thesis. We suspect that the ensembles exhibit some degree of autocorrelation, and we are optimistic that employing an LSTM will allow us to effectively capture this.

This means that we for each batch, with size `batch_size`, in training, will give the model a number of time steps, `n_timesteps`, where each entry has `num_units` of data. Our data will thus have the input size given by the tuple (`batch_size`, `num_units`, `n_timesteps`). Since we are interested in the model given us a many-to-one output, in the sense that the `n_timesteps` previous time steps are used to say something about the current day, we set the model input `return_sequences = False`.

3.5.3 Batch Normalization

Consider a neural network composed of L layers. Each layer l receives an input vector $\mathbf{x}^{(l-1)} \in \mathbb{R}^{n_{l-1}}$, performs a linear transformation parameterized by weights $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and biases $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, followed by a nonlinear activation function $\phi(\cdot)$. Mathematically, the output of layer l , denoted as $\mathbf{x}^{(l)} \in \mathbb{R}^{n_l}$, is given by,

$$\mathbf{x}^{(l)} = \phi \left(\mathbf{W}^{(l)} \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)} \right).$$

Neural networks tend to train efficiently when the distribution of the data remains consistent, ideally normalized such that $\mu = 0$ and $\sigma = 1$. Without batch normalization, the training process, which updates the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$, causes the distribution of each layer's inputs $\mathbf{x}^{(l-1)}$ to change dynamically. This phenomenon, known as *internal covariate shift* (Ioffe & Szegedy, 2015), can significantly slow down the training process due to the fluctuating input distributions across layers and epochs.

Batch Normalization addresses this issue by normalizing the inputs of each layer on a per-feature basis. Specifically, for a given mini-batch $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^m$, the normalization for each feature k is performed as follows

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mathbb{E}_{\mathcal{B}}[x^{(k)}]}{\sqrt{\text{Var}_{\mathcal{B}}[x^{(k)}] + \epsilon}},$$

where $\mathbb{E}_{\mathcal{B}}[x^{(k)}]$ and $\text{Var}_{\mathcal{B}}[x^{(k)}]$ denote the mean and variance of the k -th feature in the mini-batch, and ϵ is a small constant added for numerical stability.

To ensure that the transformation retains the expressiveness of the network, batch normalization introduces two learnable parameters, $\gamma^{(k)}$ and $\beta^{(k)}$, which scale and shift the normalized value:

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)}$$

The output $\mathbf{y} = (y_1, y_2, \dots, y_n)$ is then used as the input for the subsequent layer. Through this process, batch normalization stabilizes the distribution of layer inputs, thereby mitigating internal covariate shift and facilitating faster and more stable training.

We note here, that this is not the way we do it in this thesis, as we do not want to standardise our network to a Z-score, per se, i.e. introduce negative values. The solar- and wind power has a natural lower bound of 0 MW and a maximum that is dependent on the size of the wind/solar park, how much the wind blows or sun shines and many more things, however, it is a fixed number. Therefore we introduce a min-max standardization, $x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} = x/x_{\max}$, since $x_{\min} = 0$. We do this manually before training and inverses the transformation by multiplying the output by $x_{\max}$ after training and testing.

Summary 3.5.3. *Batch Normalization enhances neural network training by maintaining consistent input distributions across layers, normalizing inputs to be in the range $[0, 1]$.*

3.5.4 Adam Optimizer

The Adam optimizer, introduced in Kingma and Ba, 2017, is a first-order gradient-based optimization algorithm designed for efficient stochastic optimization of objective functions. Adam is particularly effective for problems with "large datasets and

high-dimensional parameter spaces due to its adaptive estimates of lower-order moments, computational efficiency, and minimal memory requirements”.

Adam combines the advantages of two popular methods: AdaGrad and RMSProp. It adapts the learning rates for each parameter individually based on estimates of the first and second moments of the gradients.

The algorithm updates exponential moving averages of the gradient (m_t) and the squared gradient (v_t) using the following equations

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad (3.23)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \quad (3.24)$$

where g_t is the gradient at time step t , and β_1 and β_2 are the exponential decay rates for the respective moment estimates. To correct the initialization bias, Adam computes bias-corrected estimates as

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad (3.25)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \quad (3.26)$$

The parameters are then updated as follows

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \quad (3.27)$$

where α is the learning rate and ϵ is a small constant to prevent division by zero.

The advantages of the Adam optimizer and why it is used here are many, but we will focus on the following, see (Kingma & Ba, 2017).

- **Invariance to diagonal rescaling of the gradients:** Adam’s adaptive learning rate helps maintain consistency in the updates regardless of the scale of the gradients. This is helpful to us, even though we rescale our data, we do not know exactly what the range of future data could be. We could for example have days with very low winds, or a month with non-stop gusts. Also, more wind- or solar power plants could be installed in the future changing the scale of the data and therefore potentially the gradients.
- **Suitable for noisy or sparse gradients:** The method’s design makes it robust to noisy gradient estimates and efficient in handling sparse gradients. Even though gradients in backpropagation is theoretically exact, noise is introduced when using stochastic gradient descent methods such as Stochastic Gradient Descent (SGD) or the Adam optimizer. The reason is that in stochastic gradient descent methods do not use the full dataset, when evaluating the gradient. This actually improves performance and is advantageous to get out of local minima.

- **Efficient for non-stationary objectives:** Adam can adjust the learning rates for each parameter dynamically, making it well-suited for problems where the objective function changes over time. We like this feature of the Adam optimizer. It means, that we do not have to manually adapt learning rate algorithms, such as hot-starts or Cosine Annealing, which usually helps the learning, but in this case, it is data-driven, which we prefer in our report here.
- **Minimal hyper-parameter tuning:** The default parameters often work well, reducing the need for extensive hyper-parameter tuning. This makes it possible for us to focus on other aspects of the modelling.
- **Computationally efficient:** Adam has low memory requirements and is computationally efficient, which is beneficial for large-scale and high-dimensional problems.

Empirical results demonstrate that Adam performs well in practice, often outperforming other stochastic optimization methods in terms of convergence speed and robustness.

For our implementation in TensorFlow, we used the `tf.keras.optimizers.Adam` class, which provides an interface to configure and apply the optimizer during training of our model. The key hyperparameters include:

- **learning_rate:** The step size for each parameter update, typically set to 0.001.
- **beta_1:** The exponential decay rate for the first moment estimates, typically set to 0.9.
- **beta_2:** The exponential decay rate for the second moment estimates, typically set to 0.999.
- **epsilon:** A small constant for numerical stability, typically set to 10^{-7} .

We used the typical values for the hyperparameters, as described above.

For further theoretical and empirical details, we refer to the original paper, see (Kingma & Ba, 2017) and the TensorFlow documentation, see (Abadi et al., 2015).

3.6 Updating Basis Functions

This section will be used to describe the theory behind how we update the energy production ensembles from the forecast provider. We also refer to these as *basis*

functions, since they are what constitutes the design matrix in the TAQR algorithm, and some of these values will be part of the basis in the simplex algorithm at a given point in time. Thus, we will describe how we use machine learning and neural networks to *data-drivenly* find the *optimal basis functions*.

The reason for using neural networks to update the basis functions, is that we then get data-driven basis functions, and *not* fixed basis functions, which for example could be the case for spline basis functions or a polynomial basis for that matter.

We will now briefly introduce some notation.

Notation.

Let $\mathbf{X}$ denote all the ensembles. $\mathbf{X}_n \in \mathbb{R}^N$ denotes an ensemble and all the associated time points $t = 1, 2, \dots, N$. $\mathbf{X}^t \in \mathbb{R}^{50}$ denotes all the quantiles to a specific time point. One specific scalar data point can then be denoted X_n^t , where $n \in \{1, \dots, 50\}$, and t is a specific time point in the index set T , which covers all of the data. The ensembles do not come presorted from the provider, but we will most often assume that we sort them; if not, we will explicitly state so.

In Figure 3.7 we have outlined the main idea behind how we update our basis functions.

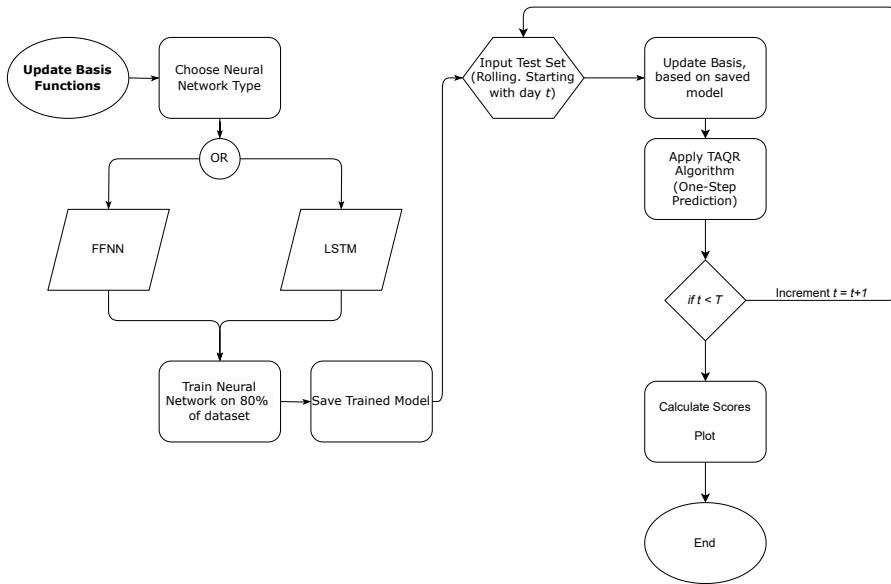


Figure 3.7. Overview of the pipeline that updates basis functions.

In the flowchart, circles indicate start and end points. Rounded boxes shows actions. Parallelograms refer to inputs from users and diamonds refer to (automatic) decisions points. Figure 3.7 will be a sub diagram in the overall flowchart that will indicate the entire code flowchart, seen in Section 4.

First, we need to input what network type the user wants to use for the updating. For simplicity in the walk-through, we start with the FFNN choice here, and will cover the LSTM later.

We thus create a FFNN in Tensorflow, as described in Section 3.5. The network takes in $\mathbf{X}^t$ for every time point. The network outputs the corrected ensembles $\hat{\mathbf{X}}^t$.

Secondly, to train the network (on 75% of the entire dataset) we use a custom loss function, which will be outlined here in pseudocode shortly.

During training we input $[\mathbf{X}^t, \mathbf{y}^t]^T$ to correct the ensembles in the loss function, where $\mathbf{y}^t$ is the actual observation. To be able to follow the pseudocode, we need this definition of what we mean by the τ -function, which we will need to find the correct percentile.

Definition 3.15 (τ -function to find the correct percentile). *Let Q denote a linear interpolation between the 0.01 and the 0.99 quantile, divided into 50 equidistant points. Formally, if $Q = \{q_i\}_{i=1}^{50}$ represents these quantiles, then q_i is given by:*

$$q_i = 0.01 + \frac{i-1}{49} \times (0.99 - 0.01)$$

for $i \in \{1, 2, \dots, 50\}$. The function $\tau : (q, X) \mapsto X_q$ is defined to find the q -th percentile of the input given to it, as described in Equation (3.1). In Algorithm 3, that is $[\mathbf{X}^t, \mathbf{y}^t]$. This function τ is capable of processing multiple quantiles simultaneously i.e. if q is a vector with more than length 1, this is not a problem in the implementation.

Let us see how this works in the following pseudocode.

Algorithm 3 Pseudocode for quantile loss function for a specific t

Ensure: $\mathbf{X}_1^t < \mathbf{X}_2^t < \dots < \mathbf{X}_{50}^t, \forall t$

for $q \in Q$ **do**

$\hat{y}^t \leftarrow \tau(q, [\mathbf{X}^t, \mathbf{y}^t]^T)$ $\triangleright$ This is the value the quantile should have had

$\varepsilon \leftarrow \hat{y}^t - y_q^t$ $\triangleright y_q^t$ is the predicted value for the q 'th quantile from the FFNN

$l \leftarrow l + \max(q \cdot \varepsilon, (q - 1) \cdot \varepsilon)$

end for

Return $l/50$

In Algorithm 3 we have outlined how our multidimensional loss function works. Even though we do not model the gap between the quantiles directly and optimize the loss function such that the gap is greater than zero, we still do never get crossings in our corrected ensembles. The reason is that our τ -function picks out the correct ensemble from the sorted ensembles and the actual observation. Thus the FFNN has incentive to model non crossing quantiles to every time step, since this would minimize the loss. By doing it this way we teach the network how the ensembles should be corrected, and only inputting the actual observations into the loss function makes sure that we do not use look-ahead bias or the observations when we are updating our ensembles for the test set.

There is quite some power to the subtlety of this loss function. Normally when doing quantile regression with neural networks you model each quantile individually. Then you could sort all the quantiles after to alleviate the problem of crossings, however, that solution is not smooth, since you could experience jump in your quantiles. This is not usually done in the literature to the best of the author's knowledge. Our loss function targets the specified quantile, and not just randomly sorts "some" produced quantiles. Further more, the for-loop presented in Algorithm 3 can easily be parallelized in TensorFlow speeding up the calculation significantly. This speeding up may be needed, since summing the loss for all quantiles at once, means we usually needs more epochs to get the same accuracy, compared to if we trained each quantile separately.

This was how we did it with the FFNN. We are also correcting ensembles with the LSTM network presented earlier. We are using the exact same approach. The only difference is that the LSTM is a form of Recurrent NN which is made to handle multiple lagged inputs. If we are trying to correct the ensembles for day t , we input $[\mathbf{X}^t, \mathbf{X}^{t-1}, \mathbf{X}^{t-2}, \mathbf{X}^{t-3}, \mathbf{X}^{t-6}, \mathbf{X}^{t-12}, \mathbf{X}^{t-24}, \mathbf{X}^{t-48}]^T$. This is chosen such that we put more emphasis on recent observations but still give the network the possibility of picking up on some daily or bi-daily seasonality. To be able to train this model properly in the time allotted to this thesis, we did not include more data than this in the final model. For more on this, we refer to Section 4.0.1.

Note, even though the input to the LSTM network includes more lagged data points, the loss function in Algorithm 3 does not. The reason is that we still only need to predict 20 or 50 quantiles for time t , and not the lagged inputs, since these will be realized in real-time updating. Thus the (multidimensional) loss-function only is concerned with $\mathbf{y}^t$.

Going back to Figure 3.7, we then save the trained model, where we have used the multidimensional loss function to train the network.

Preparing the Online Implementation.

Assume we have arrived at day $t < T$. We will then run the appropriate inputs

through the selected network, to obtain the updated basis to be used for the TAQR algorithm.

We do a one-step prediction using our previous end point on the simplex polyhedron i.e. we use our last solution to the quantile regression problem as our initial guess at a solution. Usually the algorithm converges fast, and we thus have obtained our adaptive quantile regression estimates for day t . If $t < T$, we increment t and repeat the process. This is the process we use on our dataset. In the online real-time implementation we would wait until new observation becomes available.

The online implementation looks a bit different and will be discussed in the further work section in Section 5. Here we will also discuss the possibility of implementing more time adaptive neural nets, that do not need retraining - say - every two to four weeks. We will introduce the concepts of Liquid Neural Nets, N-BEATS and N-HiTS.

Summary 3.6. *We have introduced a multidimensional loss function and presented the algorithm we use for doing the neural network updating of the original ensembles.*

As we will clearly see in Section 4, the correlation structure breaks once we update the basis functions with neural networks. This can be observed, for example, by comparing the ACF and PACF of the original data with those of the corrected ensembles. To fix this we will now introduce Stochastic Differential Equations as a way to fix this. We emphasize that it is not always that you do care about the correlation structure (or the Variogram score) breaking down, and sometimes we might only care about getting our forecasts close to the median (improving the CRPS score). If we do not care about the correlation structure, we should of course *not* try to fix it, if we end up breaking other good scores e.g. the CRPS.

3.7 Stochastic Differential Equations

When modelling complex phenomena, as wind- and solar power production are, there are many ways to go about this. We have previously introduced neural networks as a way to do this. In Section 4 we will see that these networks destroy the autocorrelation structure of our ensembles. From the papers introduced in the introduction (J. K. Møller et al. (2016), E. B. Iversen et al. (2017), E. B. Iversen et al. (2016)), we get the inspiration to try and "repair" this structure with SDEs. As described in the previous sections, this is not always a concern. If the Variogram score or the correlation structure is not of interest, then applying the SDEs for correction may not be necessary.

Therefore, we now introduce SDEs.

If we consider an Ordinary Differential Equation, of the form,

$$\frac{dx_t}{dt} = \mathbf{f}(x_t, u_t, t, \theta), \quad (3.28)$$

our first idea would be to add a time dependent stochastic part, leading to,

$$\frac{dx_t}{dt} = \mathbf{f}(x_t, u_t, t, \theta) + w_t, \quad (3.29)$$

where w_t has zero mean and independent increments, x_t is the data, u_t could be some input or force acting on the system, t is time and θ is the parameters. If we assume w_t to be gaussian, this will define a SDE. Since we are working in continuous time, nothing but an integral interpretation works, and so we will introduce a general SDE and the Itô integral interpretation, following Øksendal, 2003,

Definition 3.16 (Stochastic Differential Equation (SDE)). *A SDE has the general form*

$$dx_t = \mathbf{f}(x_t, u_t, t; \theta) dt + \sigma(x_t, u_t, t, \theta) dw_t$$

where $t \in \mathbb{R}_0$ is time, $x_t \in \mathbb{R}^n$ is the state, $u_t \in \mathbb{R}^r$ is an input vector, $\mathbf{f}(\cdot)$ is a vector function into $\mathbb{R}^n$, $\sigma(\cdot)$ is a matrix function into $\mathbb{R}^{n \times m}$, $w_t \in \mathbb{R}^m$ is the standard Brownian motion (i.e. $w_t \sim N(0, \mathbf{I}t)$), and $\theta \in \mathbb{R}^p$ is a parameter vector. The Itô integral interpretation is

$$X_t = X_0 + \int_0^t \mathbf{f}(x_t, u_t, s; \theta) ds + \int_0^t \sigma(x_t, u_t, s, \theta) dB_s. \quad (3.30)$$

Since, we here have given an introduction to something continuous we need to comment on how we go from our discrete observations to the continuous solution and vice versa. Assume we are given N observations $\mathbf{y}_N = \{y_{t_1}, \dots, y_{t_k}, \dots, y_{t_N}\}$, from which we want to find the optimal parameters, θ , and the best structure of $\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$ and $\sigma(\cdot)$ in (3.31).

$$\begin{aligned} dX_t &= \mathbf{f}(X_t, u_t, t, \theta) dt + \sigma(X_t, u_t, t, \theta) dW_t, \\ \mathbf{y}_{t_k} &= \mathbf{h}(X_{t_k}, u_{t_k}, t_k, \theta) + e_{t_k}. \end{aligned} \quad (3.31)$$

(3.31) is known as a continuous-discrete time model, $\mathbf{y}_{t_k}$ is the observation at time t_k , u_{t_k} is the input at time t_k , $\{e_{t_k}\}_{k=1}^N$ is a sequence of i.i.d normal random variables i.e. a simple discrete random field. $\mathbf{f}$ and σ is known as the drift and diffusion term respectively. The drift term is what we know from standard ODEs. W_t is in general a Wiener process, but we will here restrict ourselves to only consider $W_t \equiv B_t$, where B_t is Brownian motion.

The theory on SDEs are extremely extensive, and we will only focus on the most essential parts for this thesis. This may mean we will have to omit non-obvious facts to be concise; here we will focus on the Ornstein-Uhlenbeck (OU) process.

3.7.1 Ornstein-Uhlenbeck Process

Definition 3.17 (Ornstein-Uhlenbeck (OU) process). *The simplest example of when a linear system is driven by noise, is given by,*

$$dX_t = -\lambda X_t dt + \sigma dB_t, \quad (3.32)$$

where $\lambda > 0, \sigma > 0$. The stationary solution to (3.32) is called the OU process. The OU process has expectation 0 and variance $\mathbf{V}X_t = \sigma^2/(2\lambda)$, see (Thygesen, 2023).

Example 3.18. *An example of a simulation of the OU process, which has zero mean for its stationary solution, can be seen in Figure 3.8. The parameters are $\lambda = -0.11, \sigma = 0.53, X_0 = 1$.*

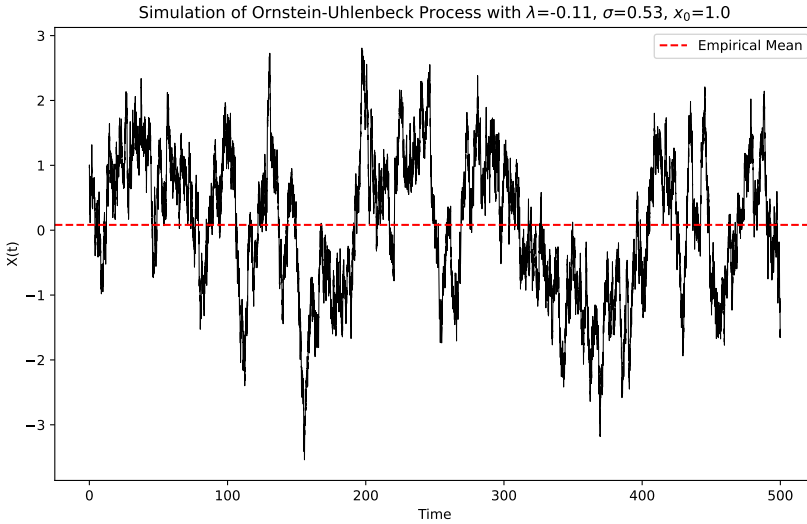


Figure 3.8. Simulation of the Ornstein-Uhlenbeck process with $\lambda = -0.11, \sigma = 0.53, X_0 = 1$, for 50,000 timesteps of length 0.01 each.

To estimate the parameters $\theta = (\lambda, \sigma)$ in the OU process, we will apply the closed form solution to the Maximum Likelihood Estimate, given in Valdivieso et al., 2009. We will present the MLE estimate in Theorem 3.19.

Theorem 3.19. *Let the OU process from Definition 3.17 be given.*

Let $X(t_0), X(t_1), X(t_2), \dots, X(t_n)$ be a sample of the stationary OU process, $x_0, x_1, x_2, \dots, x_n$ their observed values and $\Delta_k = t_k - t_{k-1}$. Since the process is Markovian and Gaussian the likelihood function of this sample is explicitly given by,

$$\begin{aligned} L(\theta) &= f_{X(t_0)}(x_0) \prod_{k=1}^n f_{X(t_k)|X(t_{k-1})=x_{k-1}}(x_k) \\ &= \frac{1}{\sigma\sqrt{\pi}} \exp\left(-\frac{x_0^2}{\sigma^2}\right) \prod_{k=1}^n \frac{\exp\left(\frac{-(x_k - \exp(-\lambda\Delta_k)x_{k-1})^2}{\sigma^2(1 - \exp(-2\lambda\Delta_k))}\right)}{\sigma\sqrt{\pi(1 - \exp(-2\lambda\Delta_k))}}. \end{aligned}$$

Hence, the observed maximum likelihood estimator can be easily found by numerically maximizing the log-likelihood function

$$\begin{aligned} l(\theta) &= -\frac{(n+1)}{2} \log(\pi\sigma^2) - \frac{x_0^2}{\sigma^2} - \frac{1}{2} \sum_{k=1}^n (1 - \exp(-2\lambda\Delta_k)) \\ &\quad - \sum_{k=1}^n \frac{(x_k - \exp(-\lambda\Delta_k)x_{k-1})^2}{\sigma^2(1 - \exp(-2\lambda\Delta_k))}. \end{aligned} \tag{3.33}$$

We refer to Valdivieso et al., 2009 p. 3-4 for further details. We use the numerical L-BFGS-B optimizer from the `scipy`-package in Python.

Summary 3.7.1. *We explained that we will use a closed form Maximum Likelihood Estimate to find the parameters in the Ornstein-Uhlenbeck process.*

3.7.2 Predictions from SDEs

We now understand how to fit the parameters in the OU process. Next, we will explain how to discretize and simulate SDEs, as well as make predictions using the OU process, which serves as an example of a solution to an SDE.

Discretizing the SDE. To be able to simulate an SDE, we will introduce the Euler-Maruyama scheme. We will here take the SDE from (3.32).

Definition 3.20 (Euler-Maruyama Scheme). *For the SDE (3.32), the Euler-Maruyama scheme is given as, see (Thygesen, 2023),*

$$X_{t+h} = X_t - \lambda h + \sigma (B_{t+h} - B_t), \tag{3.34}$$

where B_t is Brownian Motion, X_0 is given and $h > 0$.

Definition 3.20 clearly shows how to simulate an SDE, given an initial condition. From this it is possible to obtain kernel density estimates of the 1-step predictive densities.

Now, it is clear that 1-step predictions, which MLE ensures the SDE parameters are optimized for, can be readily obtained via the Euler-Maruyama scheme. Assume our true data, i.e. actual observations, are given as $\mathbf{y}_N = \{y_1, \dots, y_N\}$, then the $h = 1$ step predictions is obtained as $X_{t+h} = y_t - \lambda \cdot 1 + \sigma (B_{t+h} - B_t)$.

Note, the one-step predictions we are interested in, is what our parameters are going to be optimal for. The reason is that when we do likelihood estimation, it is based on the one-step ahead error, $\varepsilon_i = Y_i - \hat{Y}_{i|i-1}$, as mentioned in (Löwe et al., 2014).

3.7.3 Training and Test Set

In this thesis we will be training neural networks and fitting SDE parameters. Therefore we have split the data up into the following parts, which we will clarify to the reader before our results section, Section 4. Figure 3.9 shows how many percent of our observations we include in each of the sets.

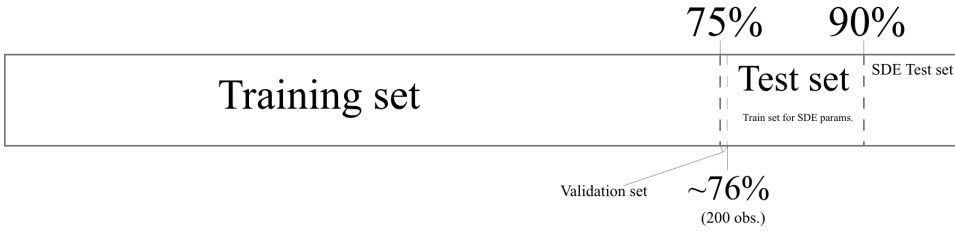


Figure 3.9. Illustration of how the data is split.

The first observation is from 2022-01-01 00:00.

- **Training set.** The original training set, which we will use to train neural networks is 75% of the entire length of the data, i.e. 14038 data points for the actual observations within each group. The training set ends at 2023-08-09 00:00.
- **Test set.** The test set we will use to evaluate our neural networks is 15% of the total data, i.e. 2807 observations. Of these 2807 observation, the first 200 observations (approximately 1%) will be used as *validation set* for plotting

during training. The 76% of the data ends at 2023-08-16 19:00, while the 90% ends at 2023-12-04 00:00.

- **SDE Test set.** The last 10% of our data 1871 observations is the set we will be using to evaluate our SDE corrected ensembles. The dataset ends on 2024-02-19 23:00.

The reason we are splitting up the data like this, is that we need a lot of data to train our parameter heavy neural networks. We can *not* use the in-sample training set to get realistic corrected ensembles, since the networks have seen this data during training. The SDE parameters needs less data than the network, due to only two parameters that needs estimating. To be able to evaluate the SDE corrected ensembles we need an out-of out-of-sample, so to speak, i.e. the SDE test set in Figure 3.9.

3.7.4 Fixing Correlation Structure

In Section 4, we will see that when we consider our CE, the Variogram score, that measures the correlation structure, is often worse in the CE compared to the OE. This makes sense, since we alter the ensembles from the weather model, this will be discussed here and in Section 4.

We know from e.g. Time Series Analysis, that if our model is "good" we capture everything (i.e. the underlying dynamics) and thus the residuals should be white noise (Madsen, 2007). If this is not the case, and we see autocorrelation in the residuals, a possible solution is to add stochasticity to the process modelling the residuals, as suggested in (J. Møller, 2011).

Therefore we here introduce a method for fixing the correlation structure using SDEs. In Algorithm 4 we will describe how we use SDEs to adjust a corrected ensemble, corresponding to a specific quantile, q . We denote this CE_q in the algorithm. Furthermore, y corresponding to the actual observations. $T_{SDE} \subset T$, where T is the test set (75-100%) and T_{SDE} is the SDE test set (90-100%) of the data. Then $T \setminus T_{SDE}$ is the training set for the SDE parameters.

Algorithm 4 Fixing correlation structure with SDE

```

1: Inputs:  $CE_q, y \in T \setminus T_{SDE}$ 
2: Outputs:  $\widehat{CE}_q$ 
3:  $r = y - CE_q$ 
4:  $\lambda, \sigma \leftarrow \text{FitSDE}(r)$ 
5:  $X \leftarrow \text{SimulateSDE}(\lambda, \sigma)$  ▷ Simulate entire SDE history
6: for each  $t \in T_{SDE}$  do
7:    $\widehat{CE}_q^t \leftarrow CE_q^t + X_t$ 
8: end for
9: Return  $\widehat{CE}_q$ 

```

In Algorithm 4 $\text{FitSDE}(\cdot)$ and $\text{SimulateSDE}(\cdot, \cdot)$ are very standard function that uses the MLE theory from previously to estimate the SDE parameters and simulates values from the SDE using the the Euler-Maruyama scheme from Definition 3.20. All the fitted SDE parameters can be seen in Appendix, Table A.2

Summary 3.7.4. *Algorithm 4 takes an initial corrected ensemble and actual observations to fit an SDE model to the residuals. It then uses this SDE to simulate a correction process, which is applied to the test set, resulting in an adjusted corrected ensemble that better fits the correlation structure of the actual data.*

To present the final model, let $f(\cdot)$ denote the neural network transformation, $\mathbf{X}_t$ is the original ensembles to time t and y_t is the actual observations to time t , then for a specific quantile $(\cdot)^\tau$, we have the SDE for which we can estimate parameters, for $t \in T \setminus T_{SDE}$,

$$d\mathbf{X}_t^\tau = -\lambda(f(\mathbf{X}_t)^\tau - y_t)dt + \sigma dB_t. \quad (3.35)$$

(3.35) is suitable for $\tau = 0.5$, the median. Here we expect the residuals to have mean value 0, which aligns with the expected value of the stationary OU process. For $\tau \neq 0.5$, we should incorporate a bias-term that could handle the offset the residuals will have from the actual observations. This is left for future work.

We now have all the theory outlined that we will be using in this thesis. The three step correction procedure; data-drivenly correcting basis functions (i.e. the ensembles), running the TAQR algorithm and finally fixing correlation structure with SDEs. The theory will be applied in the following section: Results.

CHAPTER 4

Results

In this Results section, many results will be covered, so an outline of what is being considered is provided. In the three-step corrective procedure, there is an option to do evaluation at many points. To avoid confusion, a graphical representation is presented in Figure 4.1. Here, the probe indicates where in the process the results will be presented.

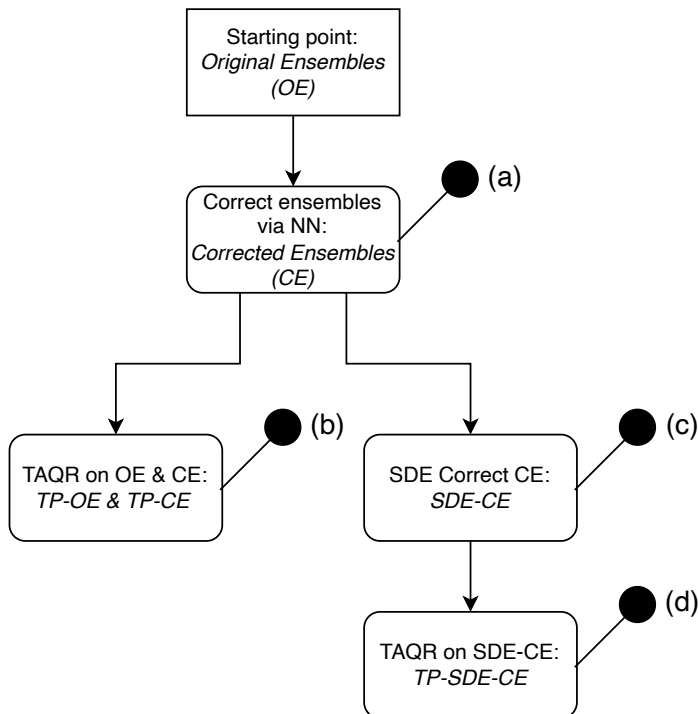


Figure 4.1. Overview of where the results will be "measured".

1. The starting point is the Original Ensembles (OE), these are given.

2. We then do a data-driven correction of the original ensembles. This produces the *corrected ensembles (CE)*. We have two different neural network architectures working here, the FFNN and LSTM-network. This was described in Section 3.6.
 - We will present results on evaluating these CE compared to the OE. This is **Probe (a)**.
3. Based on these CE we run the TAQR algorithm to obtain our best bet on where the actual observation will end tomorrow. This was described in Section 3.2. This produces our *TAQR predictions based on the CE (TP-CE)*. We evaluate if our TP-CE is better than our TP-OE. This is **Probe (b)**.
4. We then apply our SDE correction procedure on the residuals from our CE, and present these results in the hopes that this fixes the variogram score. This part of the theory was presented in Section 3.7. This gives us the *SDE-CE*. This is **Probe (c)**.
 - The TAQR algorithm is also applied to the SDE-CE to see if any metrics are improved here. This is **Probe (d)**.

Before we begin analyzing the results from Probe (a)-(d) we will present our main algorithm with a flowchart, see Figure 4.2. We use the symbols defined in (Broughton, 2024).

We will now meticulously go through the steps in the flowchart from Figure 4.2 and make observations.

1. **Start node:** Initiating the process.
2. **Load Data:** The process starts by loading the data specific to a given area and type, such as “Onshore Wind Power”, “Offshore Wind Power” or “Solar Power”. This involves retrieving the relevant datasets, ensuring they are correctly indexed and formatted for further processing.
3. **Decision: Model Type?:** The next step is a decision point where the type of neural network to be used is determined. This can be either a Feed-Forward Neural Network (FFNN) or a Long Short-Term Memory (LSTM) network. The reason we include this decision is with thoughts on the future, where this input can be mixed up with additional implementations of neural networks, see Section 5.3.
4. **Train FFNN Model:** If FFNN is chosen, the data is preprocessed, including handling missing values and normalization. Quantiles are defined, and the FFNN model is trained using 80% of the data.

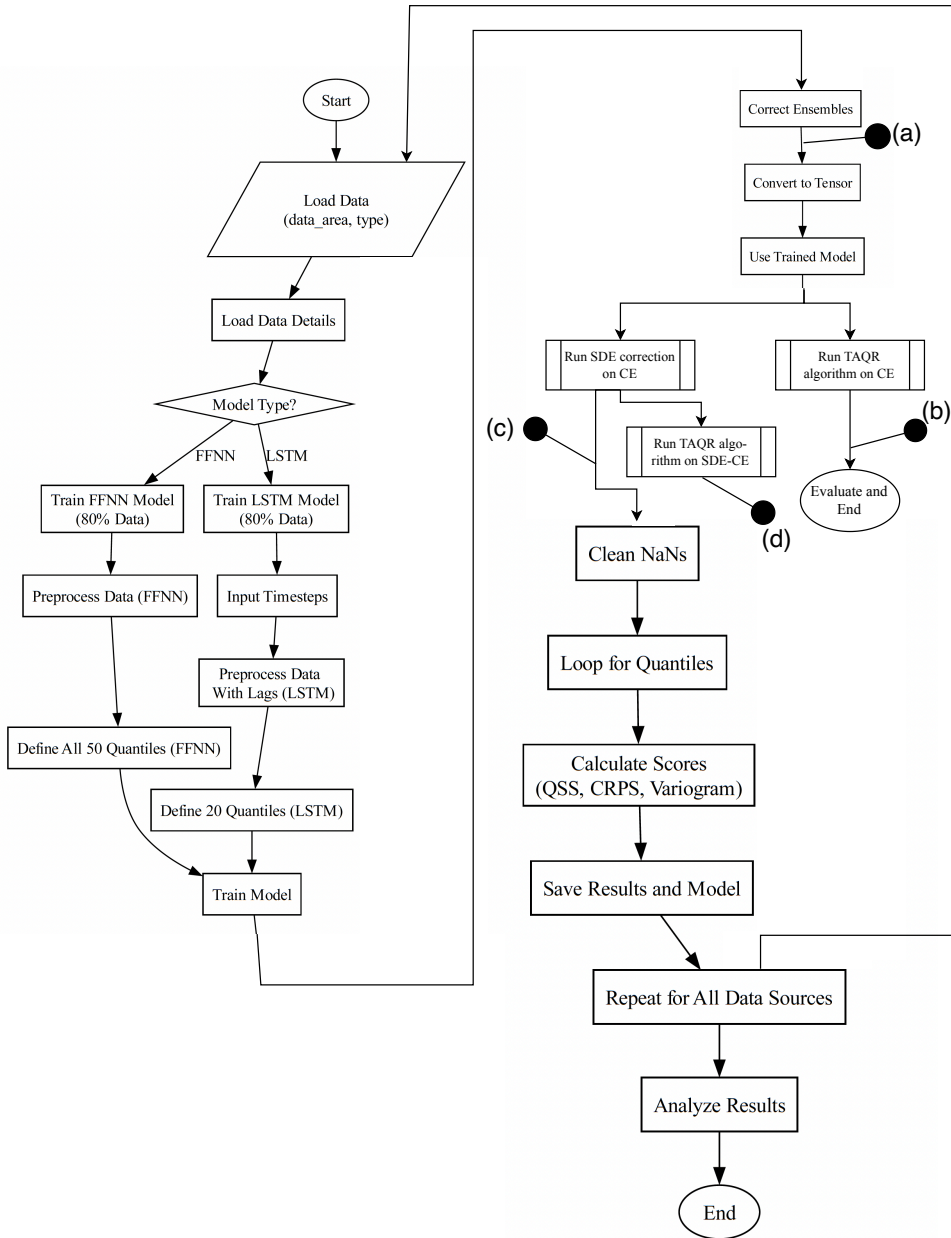


Figure 4.2. Flowchart of our entire Code Pipeline. Here QSS refer to the Quantile Score.

5. **Train LSTM Model:** If LSTM is chosen, additional preprocessing is done to include lagged data as specified by the "Input Timesteps". Similar to FFNN, quantiles are defined, and the LSTM model is trained.
6. **Correct Ensembles:** Post training, the ensembles are corrected using the trained model. This involves converting the data to tensors and applying the model to generate corrected ensembles. This is what is specified in the flowchart from Figure 3.7.
7. **a) Run TAQR Algorithm:** This step, indicated as a subroutine, involves running the TAQR algorithm using the corrected ensembles. It cleans the data for NaNs and loops through the quantiles to produce the quantile predictions. We refer to Section 3 for a deeper explanation. We run the TAQR on the CE and evaluate these as described with Probe (b).

b) Run SDE Correction: Furthermore, we also run the SDE correction on CE, producing our SDE-CE. On the SDE-CE we also have the option to run the TAQR algorithm again to be able to produce the results that corresponds to Probe (d). If we do not do that we can evaluate results at Probe (c). See Algorithm 1, 2 and 4.
8. **Evaluate/Calculate Scores:** The results from the TAQR algorithm and SDE correction are evaluated using various scoring metrics. The Quantile Score (QSS), CRPS and Variogram score are calculated to assess the performance of the predictions.
9. **Save Results and Model:** After evaluation, the scores, corrected ensembles, TAQR results, and the trained model are saved for future reference and analysis.
10. **Repeat for All Data Sources:** The process is repeated for all the different data sources specified, ensuring a comprehensive analysis across various datasets.
11. **Analyze Results:** Finally, the results are analyzed, providing insights into the performance and accuracy of the models and predictions.
12. **End:** The pipeline concludes, having processed, analyzed, and saved the necessary data and results.

4.0.1 Model Selection

In Section 3 we present both the FFNN and LSTM framework. What we will see is that the more sophisticated of the two, the LSTM network, is better at correcting

the ensembles based on QS and Variogram, which is why we will focus on the "model-selection" process of the LSTM. Model-selection here refers to choosing the correct lagged values, the size (`num_units`) of the LSTM cells and the amount of cells in a layer. We will briefly mention that the FFNN network had an obvious structure, namely to include the number of quantiles as hidden neurons in the hidden layers. We started with only one hidden layer with 50 neurons, but had to add an additional two hidden layers with non-linear activation functions (`relu`, `tanh`, `relu` respectively) to get any usable results. The output layer is the number of quantiles (50) as well.

A full visualization of the FFNN network can be seen in Appendix, Figure A.6.

We started with the FFNN network architecture since it is the easiest to set-up and get a minimum viable starting point. After going through the model selection process and reaching model 3, see Table 4.1, we started experimenting with the LSTM architecture. As we will see, this gave dramatically better results on two out of three measures.

Note, for both of these model selections in Table 4.1 and 4.2, the Layer 0 in TensorFlow, always support a `batch_size` parameter, which handles the batch size dynamically. Thus, in the LSTM case the LSTM layer would have the input size ((units), (batch_size, n_quantiles, timesteps)). To keep the thesis somewhat concise, we will not be displaying training- and validation loss curves for all models here. These have, of course, been created, examined, and compared to each other, and were deemed satisfactory.

FFNN (n_quantiles = 50 always)						
Model	Layer 0	Layer 1	Layer 2	Output Layer	Act. Functions	# Weights
1	50			50	relu	2500
2	50	50		50	relu, tanh	5000
3	50	50	50	50	relu, tanh, relu	7500

Table 4.1. FFNN model selection process.

Due to time constraints and the above arguments, we stopped focusing on the FFNN architecture and shifted our focus to the LSTM network. In Table 4.2 we present our model selection process for the LSTM network. Note, activation functions for Dense 1 and 2 are sigmoid and relu, respectively. `units` = 50 always.

We started out very simple by trying to recreate what we have learned from the FFNN network: Model 1. Remember that the `num_units` parameter in the LSTM case mimics that of a hidden layer in FFNN. To correct the ensembles on day t , we only inputted the lagged values for day $t - 1$, which is what the second input in the input size specifies in Table 4.2. Adding a few more lagged time steps, i.e.

$(t-1, t-2, t-3)$ did not change much for Model 2. So, maybe the LSTM structure in and of itself was not enough. Hence, we added two dense layers, which is the same as a hidden layer in the FFNN case. Also, more lagged time steps was added, now including the last 12 hours of data, to hopefully capture more correlation structure. This is Model 3. The training time of Model 3 is already in the hours. For Model 4, we went all-in. Adding all previous 48 hours of data for each time point, increasing the number of neurons in the dense layers, thus creating a model with millions of weights. We soon had to realize that training enough epochs on this model would not be possible. Therefore we decreased the number of lags for Model 5. We also decreased the number of neurons in the dense layers to model that of the FFNN. Now, the training time hovered around 2-3 hours on a 2021 MacBook Pro M1 Pro laptop. This concludes our model selection. We reached an equilibrium between performance and time for training that was viable within the scope of this thesis and thus we accept Model 5 as our LSTM network.

A full blown parameter specification for our final model (Model 5) is presented in Table 4.3. We see that the LSTM layer has layer-, recurrent- and bias weights, while the dense layers has layer- and bias weights only.

The total number of trainable weights for our final Model 5 is $(52 \cdot 1024) + (256 \cdot 1024) + 1024 + 256 \cdot 20 + 20 + 20 \cdot 20 + 20 = 321,976$.

LSTM				
Model	LSTM Layer	Dense Layers		Output Layer
		Dense 1	Dense 2	
1	(n_quantiles, 1)			(n_quantiles)
2	(n_quantiles, [1,2,3])			(n_quantiles)
3	(n_quantiles, [1,2,...,12])	(n_quantiles)	100	(n_quantiles)
4	(n_quantiles, [1,2,3,...,48])	200	200	(n_quantiles)
5	(n_quantiles, [1,2,3,6,12,24,48])	(n_quantiles)	(n_quantiles)	(n_quantiles)

Table 4.2. LSTM model selection process.

4.1 Corrected Ensembles

In this section we will focus on how the final LSTM model, see Table 4.3, performs during training and validation. We will show the respective loss curves as well as a plot depicting how the ensembles changed throughout training. Figure 4.3(a)-(c) shows the training loss along side the validation loss for 200 epochs during training on the training set, defined in Section 3.7.3. The training looks satisfactory. The

Layer Name	Weight Type	Shape
lstm	Layer Weights	(52, 1024)
lstm	Recurrent Weights	(256, 1024)
lstm	Bias Weights	(1024,)
dense	Layer Weights	(256, 20)
dense	Bias Weights	(20,)
dense ₁	Layer Weights	(20, 20)
dense ₁	Bias Weights	(20,)

Table 4.3. Model 5 parameter specification for the LSTM network.

validation set will in most cases not perform better or on par with the training set, but as long as we see improvements, it is acceptable. We note that the training- and validation loss plateaus and we conclude that 200 epochs is enough.

Regarding Figure 4.3 a general remark here is that we choose Epoch 1 (left side of the right plots), so that the models have some time to adjust the parameters, since we would just get pure randomness in the beginning due to the random initialization of the weights.

To the right in Figure 4.3 the fitted ensembles for three quantiles are shown. In all cases we see big improvements after running 200 epochs.

The right plots in Figure 4.3(a-c) are mostly included to prove the point that we see big improvements. It is important to note that the datetime to the left and right of the vertical red line is not the same. Actually, the first 200 time points are the first 200 hours in the dataset, while the 200-400 time points are for the last 200 hours in the training set. This figure could be enhanced by including the same datetimes on both sides of the red line; however we still believe that the figure proves the point it was included for.

A few not-so-obvious comments is that for the Onshore Wind Power we see that at Epoch 1 we have the bottom ensembles to be identical to 0 at all times. This is of course not desirable and we see that it is fixed during training. However, for the Solar Power, we start out with only one ensemble being different from 0. The general quality and the middle ensemble is lifted from Epoch 1 to the last epoch. We are however not able to get the last ensemble away from zero. Usually, it would be the case that 1-3 of the ensembles were identical 0 for all time points. The model apparently did not care enough about these ensembles to correct them. Introducing some boundary penalty might be a fix. More effort should be put into this, which we will discuss in Section 5.3.

A similar figure can be found for DK2, in Appendix, Figure A.5. The comments are similar.

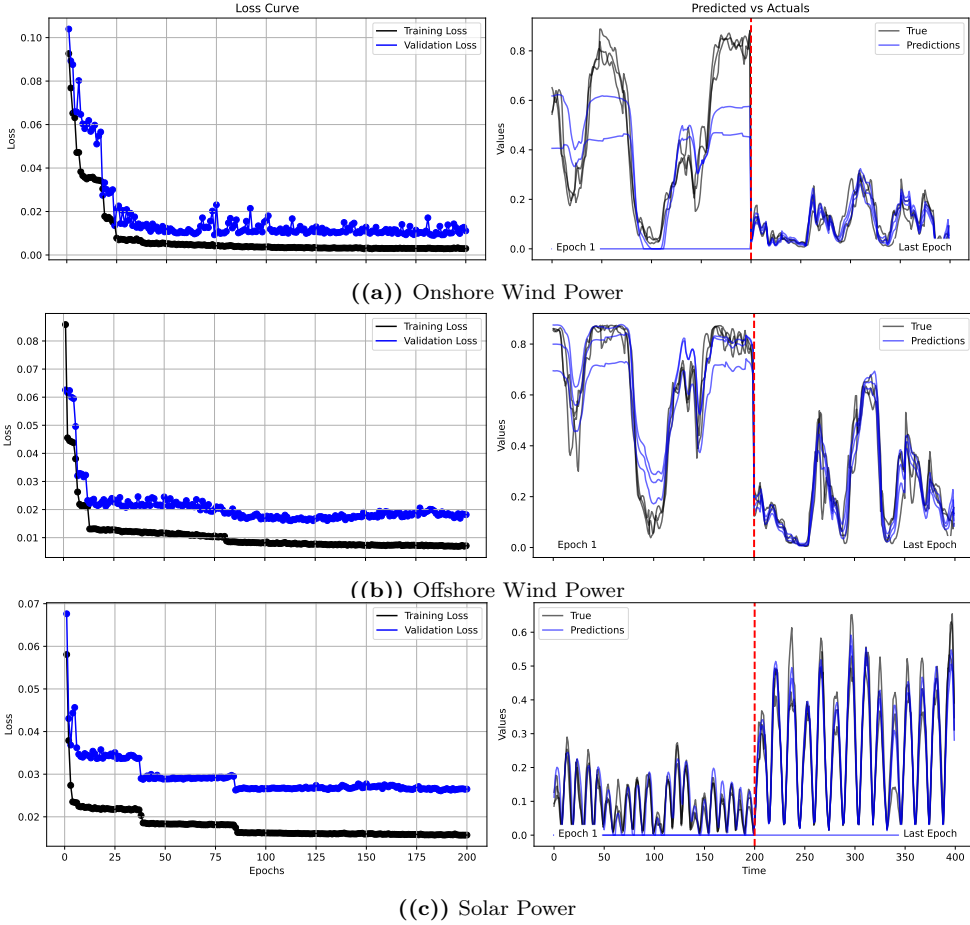


Figure 4.3. Model Training Analysis for the three different power sources for DK1.

In this section we will be showing the results of correcting our basis with our approach using neural networks to find the optimal basis. This theory was described in Section 3.6.

Consider Figure 4.4, which has 4 subplots; DK1 On- and Offshore wind power production as well as DK2 On- and Offshore wind power production. We have plotted the actual observations in black, the original ensembles in a faded grey, the FFNN corrected ensembles in red and the LSTM corrected ensembles in blue. For the DK1 Onshore wind power production we are plotting the 10-, 50- and 90%-quantile. Since

we do not have crossings in the original ensembles due to sorting, and not in our corrected ensembles due to the multidimensional loss function in Algorithm 3, we do not label the 10-, 50- and 90-percentile respectively.

In the middle two plots we are plotting the 19- and 23/24th-percentile of the ensembles. The reason is that our LSTM network sometimes - especially for low quantiles - predict that the most optimal is zero as previously mentioned. There is a mistake happening somewhere in the training, but it has not yet been resolved. Therefore to make a fair comparison we are considering the specified quantiles.

Focusing on Figure 4.4a, we see that most of the time the FFNN ensembles are the broadest. The 90% quantile captures more than 90% of the data, actually it captures 94.5% of the data in this example. In general it captures 91.4% of the data over the entire test set. The original ensemble captures 90.3% of the data. So in this case it seems that the training set have had too many extreme cases, where the FFNN needed to adjust its ensembles wider, to capture the wider distribution that was present in the training set. However, we also note that over the entire set the 90% quantile of the original ensembles only cover 83.8% of the data. This is just to say that the exercise is not easy. We will of course not compare every single quantile like this; we will later present the general results measured with QSS, Variogram and CRPS measures. In general it seems, actually across the subplots Figure 4.4(a-d), that the FFNN is in general the wider of the two, comparing it with the LSTM. We can calculate the integral between the two and see what happens.

Let $q_{0.1}, q_{0.9}$ denote the quantiles from the corrected ensembles. Let T^{test} be the index time set for the test set, we then have that the quantile gap, G , for DK1 Onshore wind power production is

$$G_{FFNN} = \frac{1}{|T^{test}|} \int_{T^{test}} (q_{0.9}(t) - q_{0.1}(t)) dt = 656.5, \quad (4.1)$$

and

$$G_{LSTM} = \frac{1}{|T^{test}|} \int_{T^{test}} (q_{0.9}(t) - q_{0.1}(t)) dt = 643.2. \quad (4.2)$$

This becomes even clearer if we plot the 10th and 90th percentile next to each other, as seen in Appendix, Figure A.4. Note, since our timepoints are discrete, this is evaluated as a sum.

Now, moving on to Figure 4.4b, we see that both original and corrected ensembles have problems with capturing the maximum output from the offshore wind production. We do not see this in Figure 4.4d, for DK2 offshore wind. After having been in contact with EnergiDataService the reason is clear. There have been a new installation of "Havmølleparken Vesterhav Syd", which is another wind farm that have been added to the danish offshore wind power production. As described in Section 2 we try to

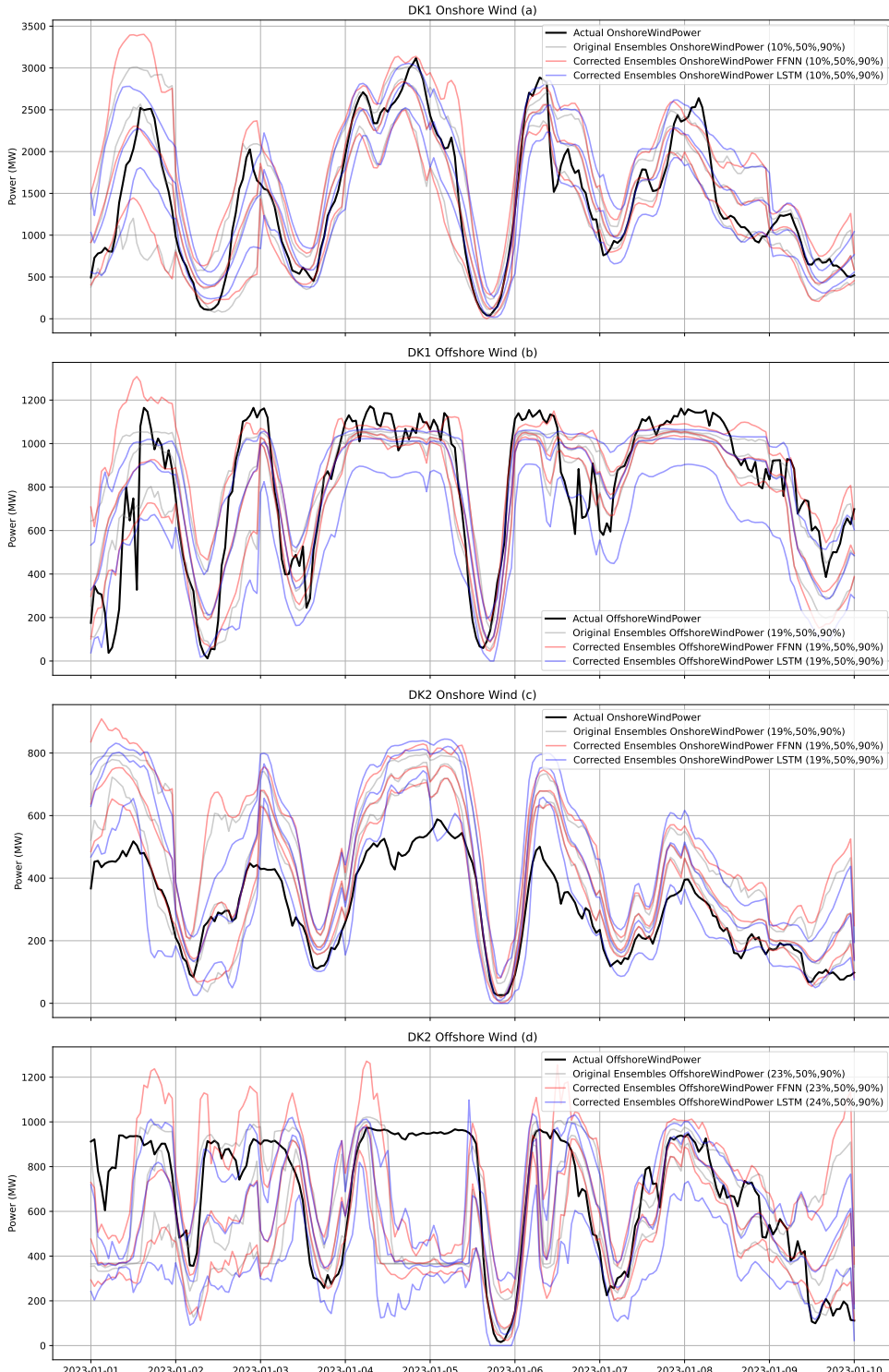


Figure 4.4. Examples of our corrected ensembles from our two different neural networks (FFNN and LSTM). Note, some quantiles are different from the usual (10/50/90)-quantile, since these quantiles was zero.

combat this change by ensuring that the maximum is the same in the training and test set. But we still end up with a bit of a skew in the distribution of the data in the test versus training set, see Figure 4.5.

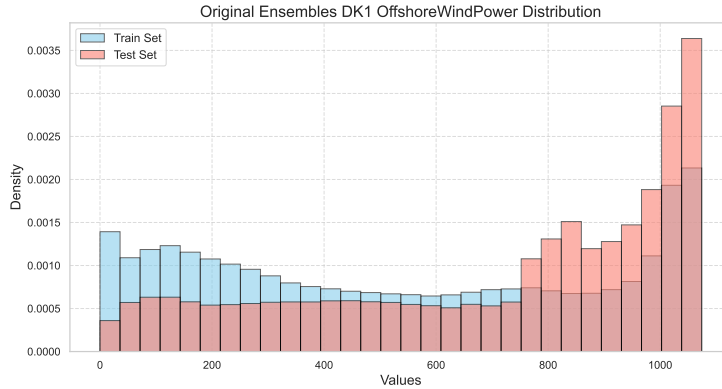


Figure 4.5. The change in the distribution of the Offshore Wind Power in DK1. "Train set" refers to the training set.

The actuals evaluation metrics for all of the different model and regions will be presented in Figure 4.6. This corresponds to Probe (a) from Figure 4.1. Note, that these are normalized values, but the actual values can be found in Appendix, Table A.3. For both the Variogram and QSS the LSTM model is much better than the FFNN model. However, for the CRPS section the LSTM model performs poorly on the DK1 Onshore Wind Power. If this was not the case we would prefer the LSTM over the FFNN architecture always. So, even though we thought that the LSTM network would perform better on all measures, this is not the case; at least not when only using the neural network to update the basis functions. Let us see what happens when we add more steps to our corrective procedure.

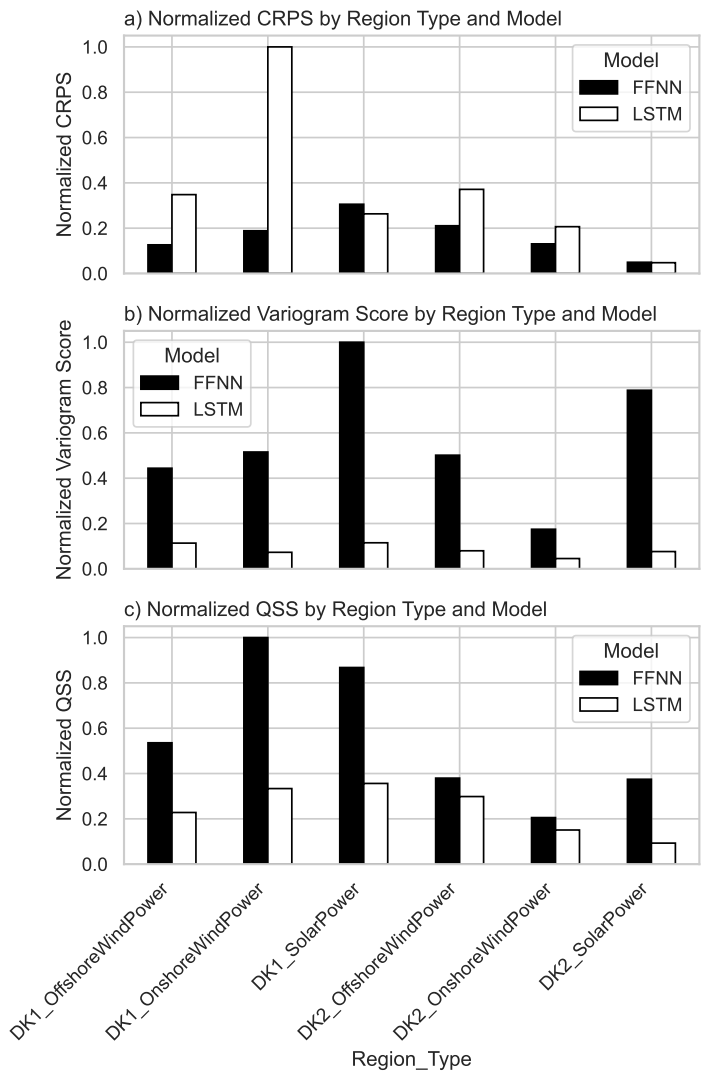


Figure 4.6. Normalized CRPS, Variogram and Quantile Score (QSS) from the two neural networks (FFNN and LSTM) for all available data sources. This corresponds to Probe (a).

4.2 Time-Adaptive Quantile Regression

In this section we will compare the output from our implementation of the Time-adaptive Quantile Regression (TAQR) algorithm when we use the the original ensembles versus using the corrected ensembles as basis functions for our predictions. This corresponds to Probe (b) from Figure 4.1.

We will be comparing using the usual quantile scoring metrics as described in Section 3.4.

When we have selected 75% of our data for training, and given the model the appropriate lags we have approximately 2800 hours worth of data in our test set i.e. our test set consist of 2800 50-dimensional corrected forecasts in the FFNN case.

DK1 Offshore Wind	Corrected	Original
CRPS	88.9043	89.8433
Variogram	1.1566	0.9267
Mean QSS	0.1253	0.0638
DK1 Onshore Wind		
CRPS	132.62	151.32
Variogram	0.44	0.43
Mean QSS	0.51	0.74
DK2 Offshore Wind		
CRPS	158.58	164.99
Variogram	0.64	0.54
Mean QSS	0.71	0.78
DK2 Onshore Wind		
CRPS	31.70	32.58
Variogram	0.06	0.05
Mean QSS	0.07	0.10

Table 4.4. TAQR Prediction Scores for FFNN Corrected vs Original Ensembles (Probe (b)).

We observe in Table 4.4 that the CRPS score always is better across the different datasets. In 3 out of 4 cases the mean QSS is also better. The Variogram score is however worse in all cases of the TAQR forecasts made with the corrected ensembles. The results for the LSTM network can be seen in Appendix, Table A.7 and A.8. We observe poor performance in the LSTM scores for Probe (b), due to the zero ensembles.

We know from Section 3.4 that the CRPS measures the overall probabilistic forecast

accuracy and is less sensitive to correlation structure, while the Variogram score is designed to detect misspecified correlation structures.

The degradation in the Variogram score for the "Corrected" indicates that the temporal focus of our algorithm might come at an expense of not properly modelling spatial dependencies and correlations. It makes sense, that altering the ensembles would destroy their correlation structure somewhat, at least not improve it, since that is not target of neither the neural networks or the TAQR algorithm.

The improvement in CRPS score make sense. Our correction method moves the ensembles closer to where the actual observations should have been. Since the CRPS score rewards observations close to the median, then we would expect that the TAQR predictions made based on the corrected ensembles had a better CRPS score.

When correcting the ensembles, our loss function is essentially the multidimensional mean QSS. Therefore it also make sense that we in 3 out of 4 cases also see an improvement here. The reason we are especially struggling with the DK1 Offshore Wind power production is explained earlier, but the main reason is contributed to the addition of another wind park, which increased the power production suddenly in the dataset.

In Figure 4.7 and 4.8 we have illustrated the differences. We have plotted the TAQR 10th-, 50th- and 90th-percentile. *This should not be understood as the ensemble quantiles*, but instead what the TAQR algorithm predicts the observation is within a certain "confidence interval" given the data that is in the basis at any given point. This might be clearer if we give a somewhat different context. Consider the illustrative Example 4.1.

Example 4.1. *Assume that we have three explanatory variables X_1, X_2 and X_3 ; rain-fall, solar-radiation and time-of-day. We want to say something about our variable Y , which is the amount of umbrella use. Using our TAQR algorithm, we estimate adaptive parameters to best infer what Y should be. Our confidence interval for our TAQR algorithm can only in this example make sense if it regards the amount of umbrella use, since none of the explanatory variables even have the same unit!*

If we apply Example 4.1 to our definition, we would have that our explanatory variables where the ensembles, but we are still only trying to predict Y , the actual observations. Thus, we should not consider quantiles the same in this case, as we do when we correct our ensembles.

In Figure 4.7 we notice specifically that the Predictions from Corrected Ensembles (PCE) in some cases, and especially around January and February of 2023 are close to the true actuals, while the Predictions from Original Ensembles (POE) are a bit too low. This is one reason for the better CRPS scores, where observations closer to

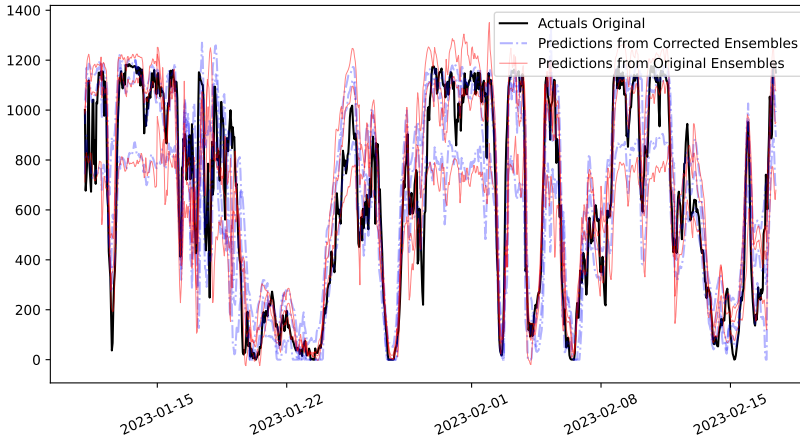


Figure 4.7. TAQR Predictions made based off of corrected and original ensembles for DK1 Offshore Wind power production. We have plotted the TAQR 10th-, 50th- and 90th-percentile. These are not labeled individually as they (mostly) do not cross.

the median is rewarded. Furthermore we also see the reverse for the 90th percentile, where the POE are too high.

In Figure 4.8 we do see some inexplicable spikes, that stem from the corrected ensembles. We see the spikes around 16th to 20th of January in 2023 as well as the 4th of February in 2023. This of course is not what we aim for. Other than these period we see the same things we saw in Figure 4.7 where PCE are closer to the actual data. Ideas for fixing this include adding a penalty in the objective function, see Section 5.3.

We will in Section 5 discuss how we may use ensemble methods to further improve our forecast.

4.2.1 ACF, PACF and QQ Plots

To quantify the results further we present ACF, PACF and QQ plots for the residuals regarding the median prediction from the TAQR algorithm based on the corrected ensembles (Probe (b)).

Note, the solar residuals in Figure 4.9(c) are the most normal; the wind seems to have fat tails, which is also evident from Figure 2.3 (especially offshore wind). We

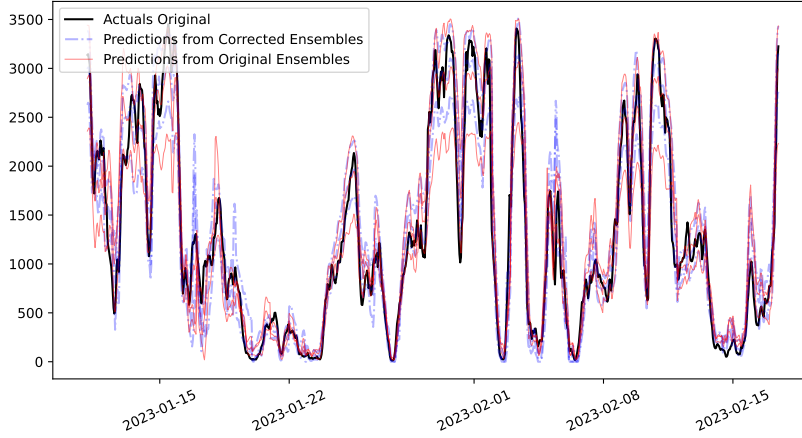
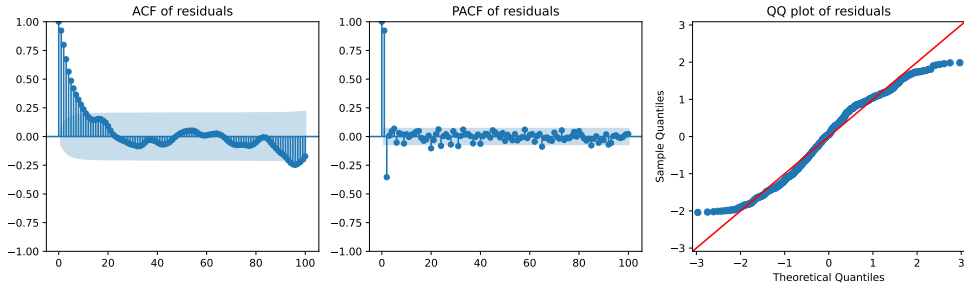


Figure 4.8. TAQR Predictions made based off of corrected and original ensembles for DK1 Onshore Wind power production. We have plotted the TAQR 10th-, 50th- and 90th-percentile. These are not labeled individually as they (mostly) do not cross.

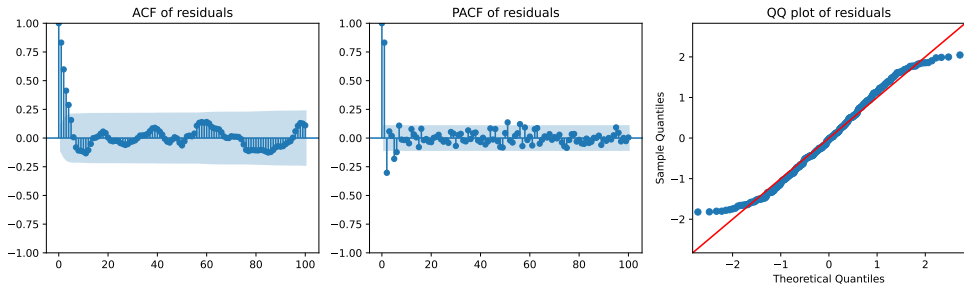
attribute the normality of the solar power residuals to the lagged values in the LSTM network, which captures the seasonality.

The presented results in Figure 4.9 is very similar for DK2, and it therefore not shown here. We also note that these are the corrected ensembles from the LSTM model. The only difference to the FFNN model, is that the wind residuals seems to fit the QQ normal quantiles better than the LSTM wind residuals do. We also see that the ACF show significant lags until around lag 5-10 for the different areas. The seasonality in the solar residuals is of higher frequency than the actual data ACF in Figure 2.5 and it dies off (becomes insignificant) after approximately 24 hours. In general the ACF shows exponential decay and the PACF has a cut of at 3, 3 and 4 or 6 lags indicating autoregressive models of the respective order for the Offshore wind, Onshore wind and solar power in the residuals stemming from the median of the TP-CE (LSTM).

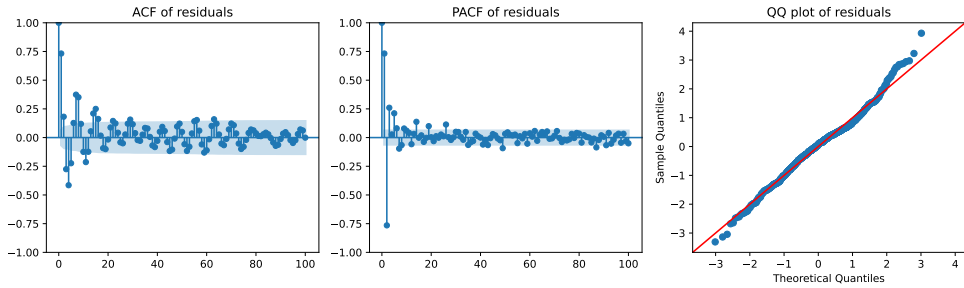
Summary 4.2.1. *The ultimate thing that we have shown so far is that we still need to fix the correlation structure in the ensembles, such that we can get even better TAQR predictions, hence we will now present our SDE corrected ensembles, applying the theory from Section 3.7.*



((a)) DK1 Offshore Wind (CE from LSTM)



((b)) DK1 Onshore Wind (CE from LSTM)



((c)) DK1 Solar (CE from LSTM)

Figure 4.9. ACF, PACF and QQ plot of residuals for the DK1 area. The DK2 area is similar and not shown.

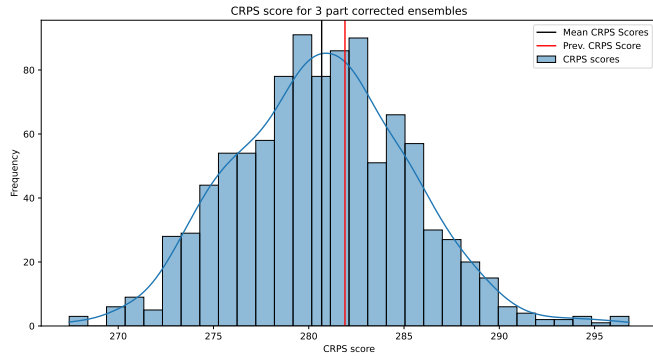
4.3 SDE Corrected Ensembles

When applying the SDE theory it is clear that when we need to simulate our data, it is based on randomness. This is why we here present our results as histograms from 1000 simulations and consider the mean value of the score from these 1000

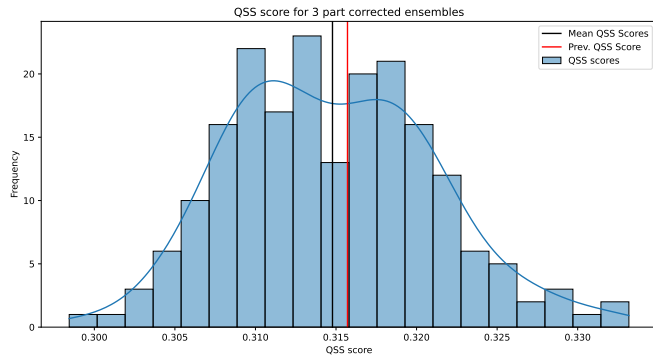
simulations¹.

Correcting the ensembles with the theory presented in Section 3.7 allow us start by presenting results corresponding to Probe (c) from Figure 4.1. We here present the results from three selected quantiles, namely the 25th-, 50th- and 75th-percentile. The scores from the simulations are collected in 30 bins shown in the histogram in Figure 4.10 and the mean value is shown in Table 4.5. Here, "Previous" corresponds to the score the already corrected ensembles (CE) scores for the specified quantiles. The variogram score is dramatically improved, while the CRPS score see a small improvement and for all pragmatic use cases the QSS score stays the same.

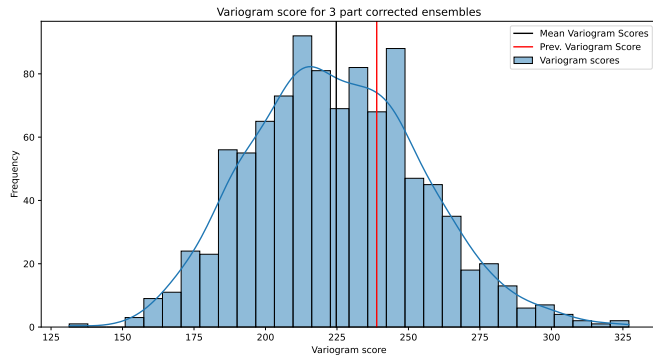
¹In the very final days of this thesis we recalled that instead of simulating, we could have used a closed form expression for the mean value through time, namely $\bar{x}_t = \mathbf{E}X_t = e^{-\lambda t}\mathbf{E}X_0$, and the variance $\mathbf{V}X_t = \Sigma(t) = \frac{\sigma^2}{2\lambda} + \left(\Sigma(0) - \frac{\sigma^2}{2\lambda}\right)e^{-2\lambda t}$. There was simply not enough time to redo all the results, and thus we will have to make do with the simulations.



((a)) CRPS score (black is the mean value) for the SDE-CE vs. CE (red is the mean value).



((b)) QSS score (black is the mean value) for the SDE-CE vs. CE (red is the mean value).



((c)) Variogram score (black is the mean value) for the SDE-CE vs. CE (red is the mean value).

Figure 4.10. CRPS, QSS and Variogram score (black is the mean value) for the SDE-CE vs. CE (red is the mean value).

The plots in Figure 4.10 presented is for DK1 Offshore wind power. In Table 4.6 we have done this exercise for all the possible combinations of region and type of network.

For DK1 offshore wind (LSTM) we see an improvement in all of the skill scores. This indicates that for this specific area and type we would always apply the SDE correction, no matter what the application may be. This is rarely the case. The SDE correction is supposed to fix the Variogram score. It does not always do that. In fact it sometimes worsens it quite a bit, e.g. the skill score on the DK2 Solar power (FFNN) ensembles is 3.5224, indicating a huge worsening in the Variogram score. It seems odd that the variogram score for DK2 solar power (FFNN) would perform so poorly, while the DK1 solar power (FFNN) achieves an almost 20% better performance in the Variogram score. As we saw in e.g. Figure 4.7 there are some undesirable peaks in the corrected ensembles from the FFNN models sometimes, that may very well translate to a poor fit for the SDE and therefore a worsening of the Variogram score.

	Mean SDE Corrected	Previous
Variogram	42.7720	45.0940
CRPS	313.0781	314.2345
QSS	1.4339	1.4312

Table 4.5. Comparison of scores for SDE corrected and previous ensembles

DK1 OffshoreWindPower FFNN	Skill Score
Variogram	1.0898
CRPS	0.9957
QSS	1.6720
DK1 OffshoreWindPower LSTM	Skill Score
Variogram	0.9592
CRPS	0.9954
QSS	0.9988
DK1 OnshoreWindPower FFNN	Skill Score
Variogram	0.9646
CRPS	1.0119
QSS	1.3282
DK1 OnshoreWindPower LSTM	Skill Score
Variogram	2.4691
CRPS	0.9880
QSS	0.9996
DK1 SolarPower FFNN	Skill Score
Variogram	0.8161
CRPS	0.9703
QSS	1.0284
DK2 OffshoreWind Power FFNN	Skill Score
Variogram	1.1011
CRPS	0.9945
QSS	1.6640
DK2 OffshoreWind Power LSTM	Skill Score
Variogram	1.1850
CRPS	1.0013
QSS	0.9985
DK2 SolarPower FFNN	Skill Score
Variogram	3.5224
CRPS	1.0600
QSS	1.0512

Table 4.6. Comparison of skill scores for SDE corrected and previous ensembles

We will not comment further on these results for now, but will move on to where this framework really shows its power, i.e. when we consider the results from Probe (d). In the following section we compare the TAQR prediction made on the original ensembles vs. the SDE corrected ensembles.

4.3.1 TAQR and SDE

For five selected quantiles (0.1, 0.25, 0.5, 0.75, 0.9) we here present the usual three scores. Again, in Figure 4.11 present the DK1 Offshore Wind power production. For all other areas and type of renewable energy, we refer to the Appendix, Figure A.7 to A.10. We have collected the results for the CRPS and Variogram in Table 4.7. The QSS is not shown, since it is effective not changing. Here, we see that in almost all cases the TAQR predictions based on the SDE-corrected ensembles performs best. We see as much as 66% improvement in the Variogram score and more than 44% improvement in the CRPS, while still having a better quantile score. For a plot on how the TAQR predictions actually could look like, we refer to the Appendix, Figure A.11.

For completeness we have included sharpness, reliability, IQR results from the TAQR-SDE-CE (Probe (d)) and TAQR-OE (Probe (b)) in Appendix, Section A.3.1, Table A.5 and A.6.

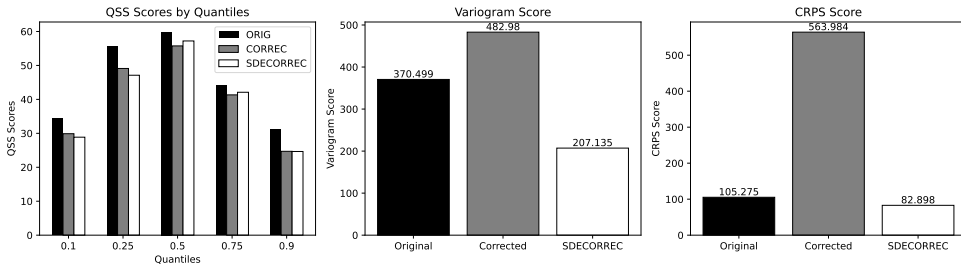


Figure 4.11. CRPS, Variogram and Quantile score (QSS) for five selected quantiles (0.1, 0.25, 0.5, 0.75, 0.9) "Orig" corresponds to the original ensembles. "Correc" corresponds to the prev. corrected FFNN ensembles and "SDECORREC" corresponds to the SDE correction applied to these (FFNN) corrected ensembles; probe (d). DK1 Offshore Wind power production.

In theory, we should focus on improving the variogram score with SDEs only when necessary. As Sørensen et al. (2023) mentions, in electricity markets where each time step is handled separately, quantile forecasts or single-horizon probabilistic forecasts suffice. These are optimal for dealing with asymmetric cost functions as they provide reliable single-horizon conditional density estimates, making CRPS ideal. However, these forecasts lack information about the temporal development of future power production. When storage capacity, start-up time, or costs are considered, advanced forecasts and the Variogram Score become crucial.

Table 4.6 indicates that for DK1 Solar Power (FFNN), both the Variogram and

Area and Type	Metric	Original	Corrected	SDE Corrected
DK1 Offshore Wind Power	Variogram	370.499	482.980	207.135
	CRPS	105.275	563.984	82.898
DK2 Offshore Wind Power	Variogram	2134.935	2543.738	725.312
	CRPS	181.184	179.294	160.566
DK1 Solar Power	Variogram	266.522	445.078	390.851
	CRPS	165.401	171.352	70.56
DK1 Onshore Wind Power	Variogram	254.995	234.084	292.066
	CRPS	191.442	178.964	166.366
DK2 Solar Power	Variogram	161.994	157.345	156.670
	CRPS	65.030	65.153	45.296

Table 4.7. Evaluating TAQR predictions based off of three different types of ensembles in our corrective procedure. The corrected ensembles are from the FFNN.

CRPS scores improve. For DK1 Offshore Wind Power (LSTM), all metrics improve, suggesting that the full three-step corrective procedure should always be applied to these ensembles. This demonstrates that although improving the variogram score may not always be necessary, in certain contexts like these, applying the SDE corrective procedure consistently enhances forecast accuracy.

4.4 Python- Implementation and Package

This section will give a brief overview of the Python package we plan to launch following this thesis defense.

This Python package will include everything presented in Section 3 and the code available to reproduce all results and plots in Section 4. A small thing to comment on here, is that for everything done in the Python-package a flag in the function header input is added, say `should_plot = True`, to allow for custom plotting options etc.

The entire code library used for this thesis is available at Github: <https://github.com/bast0320/S194357-MSC-Thesis> or see (Jørgensen, 2024b). The filenames we will be referring here can be found in the Github repository.

For the TAQR algorithm presented in Section 3.2 and in Algorithm 1 and 2 the main functions can be found in `function_for_TAQR.py`.

For training and testing of the neural networks as well as correcting the ensembles we refer to one of our main files, namely `pipeline_start_to_finish.py`, where we only need to provide the name of the area (DK1 or DK2) and the type of renewable energy we are interested in, as well as a specification of the neural network we are interested in training (FFNN or LSTM). In further releases of the Python package more types of neural networks will be implemented, please see Section 5.3 for a discussion of this.

The last step of our three step corrective procedure is fitting SDEs to the residuals if need be. The file `function_for_SDE_correction.py` includes functions for calculating the likelihood for the SDE parameters from Eq. (3.33), simulating the SDE with the Euler-Maruyama scheme from Eq. (3.34) and much more. Everything regarding the SDE-step as presented in Section 3.7 can be found in this file.

As mentioned in Section 3 we need an initial solution for the starting point on the simplex polyhedron. We needed to use the `subprocess` module in Python to be able to call R-scripts from within function. Exactly the solution we provide to do this was not seen anywhere else online, which is why the author wrote a small Medium article to cover this approach. The article can be seen in (Jørgensen, 2024a).

To get a feel for the size of the Python package we have included an overview of the number of lines in the code. See Appendix, Section A.1, Table A.1. Thus, given the extent of the code, we will not be showing code snippets in the appendix, and will refer to the algorithms presented throughout this thesis. For transparency, the code is at this moment not finalized, properly commented or organized.

Summary 4.4. *Regarding the corrected ensembles (Probe (a)), the LSTM ensembles*

are much better based on Quantile score and Variogram score. However, for DK1 onshore wind power, the CRPS score is quite poor for the LSTM model.

When applying the TAQR algorithm to the corrected ensembles the CRPS score is always better on the corrected ensembles, however the Variogram score is not (Probe (b)), which is why we introduce the SDE-corrected ensembles. At Probe (c), the results were inconclusive and we thus moved on to Probe (d):

Our three step corrective procedure, when applied in full, really do outshine almost any other combination of forecasts that we were able to provide. First correcting the ensembles with the neural network, fixing the correlation structure again and then making TAQR predictions for the actual value gives good results measured by the QSS, Variogram and CRPS.

CHAPTER 5

Discussion

In the beginning, this thesis were investigating the well known Vector Error Correction Models (VECM) on data from Germany and The Netherlands, following ideas from (Johansen, 1985). The forecasts in this data was of such bad quality, that no meaningful models could be build, and thus we abandoned the idea of using data from Germany and the Netherlands. The data we used for Germany and The Netherlands came from the ENTSO-E's transparency platform (ENTSO-E, 2024).

Therefore, we will here only discuss results from Danish data from Energinet and the forecast provider for the ensembles here.

We will repeat an important point here.

In general, when handling each time step separately, as is common in electricity markets, quantile forecasts or single-horizon probabilistic forecasts of power production are adequate. For scenarios involving asymmetric cost functions, reliable estimates of the single-horizon conditional density are necessary, making the use of CRPS potentially optimal. However, these forecasts lack information about the temporal evolution of future power production. Therefore, if factors such as storage capacity, start-up time, or costs are considered, more sophisticated forecasting and evaluation methods are required. In such instances, evaluating forecasts using the Variogram score becomes crucial (Sørensen et al., 2023). In this thesis we will take a pragmatic approach to this and evaluate both CRPS and Variogram score with great importance, since the application of this thesis may be single-horizon forecasts or it may be that we want to include factors such as storage capacity etc and therefore need the Variogram score as well.

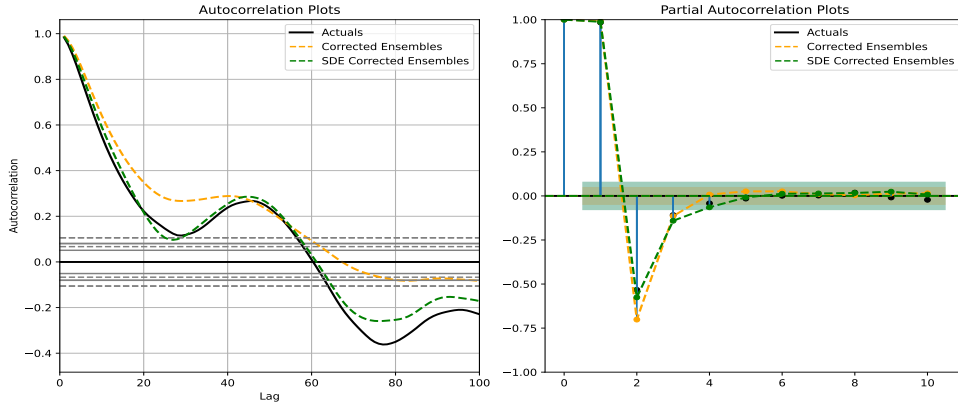
5.1 Comment on Results

Actually, it makes sense that the variogram scores deteriorates. The reason is that by updating our ensembles, we are effectively braking some of the "built-in" autocorrelation that exists in the ensembles. By breaking this, we will of course see worse results

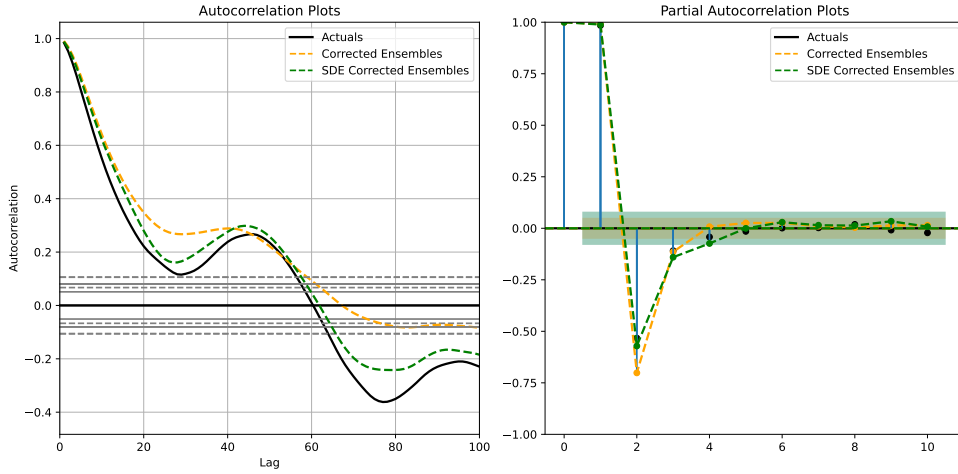
in the Variogram score, which essentially only measures the correlation structure.

In the section we will consider the autocorrelation for the corrected ensembles, and see clearly how the theory on SDE correction from Section 3.7 improves the correlation structure.

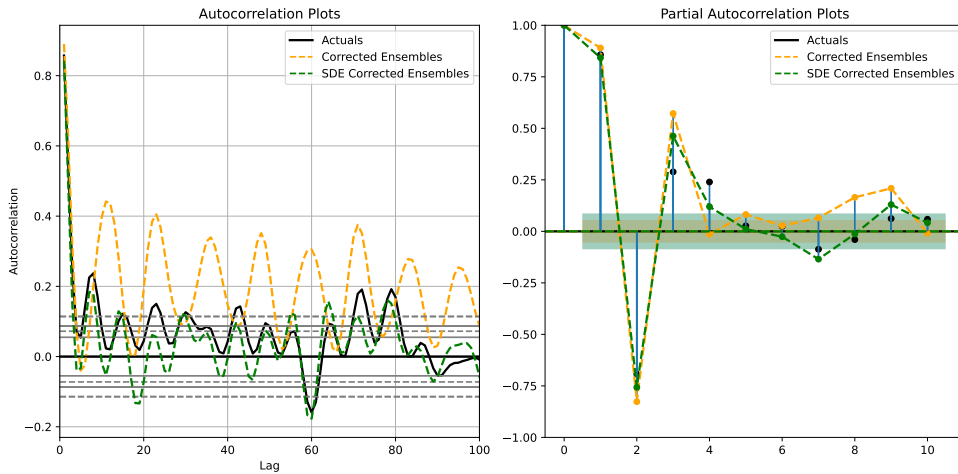
Consider Figure 5.1. The remaining plots are in Appendix, Figure A.13 and A.14. We see a clear improvement across both the ACF and PACF for the SDE corrected ensembles. This is impressive considering we used the simple Ornstein-Uhlenbeck process. We move the ACF plot for DK2 Offshore wind and DK1 Onshore wind drastically closer to the actuals ACF. Furthermore, the SDE corrected ensembles have an ACF for the DK2 Solar power that is much closer in resemblance of the actual. The corrected ensembles for DK2 solar simply has too much autocorrelation for too long.



((a)) DK2 Offshore Wind



((b)) DK1 Onshore Wind



((c)) DK2 Solar

Figure 5.1. ACF and PACF comparing the ensembles for the different ensembles. We clearly see that the SDE corrected ensembles improve the correlation structure.

5.2 Validation of Ensembles by Trading Strategy

Participants will secure a position in the balancing market, as Energinet ensures overall market equilibrium. Typically, producers are not inclined to speculate on the price differentials between the day-ahead and balancing markets. The primary objective is risk minimization, given the potential for extreme price fluctuations in both directions within the balancing market. In Denmark, this risk is asymmetrical; prices can escalate significantly but rarely turn substantially negative, see Figure A.12.

5.2.1 Case study 1: Own Wind Power Plant - Fixed Risk

Setting the scene.

Imagine you are the proud owner of a wind power plant, that produces all of DK1 onshore wind power. You have 50 ensemble forecast for the day ahead, and the median is what you consider most likely to happen. Due to regulatory instances/risk regulations/over- or under-promises etc. you must trade on the 33%-quantile. That means that according to your forecast you are selling less power every day, then what you actual think is going to happen, so that you are sure you can deliver on your promises. We do not consider the intraday market here, since that is continuous trading and that would require us to consider VWAP prices, and this data is not easily accessed.

In the day-ahead market you one day predict that the 33%-quantile = 100MWh. The price you can sell to in the day-ahead market is 15€/MWh. We assume the balancing price is lower than the day-ahead market, this is true approximately 64% of the time. Even though, as previously mentioned the primary objective is risk minimization we will make the assumption here that the price in the balancing market is 10€/MWh

Now, two things can happen.

1. You are right, and you collect the 1500€, no sweat.

2. You are not right.

- 2a. Your wind farm produced less than the 33%-quantile, so you must pay in the balancing market, to be able to fulfill on your promise of the 100MWh. Turns out your windfarm produced only 75MWh, so you need to buy 25MWh in the balancing market. This is less expensive at 10€/MW. You only made $15 \cdot 75 - 25 \cdot 10 = 875\text{€}$.

- 2b. Your windfarm produced more than the 33%-quantile, say 125MWh, so you must sell the additional 25MWh in the balancing market at 10€/MWh. Instead of the possible $125 \cdot 15 = 1875\text{€}$, you only made $100 \cdot 15 + 25 \cdot 10 = 1750\text{€}$.

So, of course it is better to be precisely correct, but here we assumed we would be on the safer side and be able to deliver and on average we here hope that we can sell to a high imbalance price. If we do this exercise for all quantiles, instead of only the 33%-quantile. Note, the assumptions of why we do this breaks if we do it for quantiles above 50%, but we continue to do so for illustrative purposes. Figure 5.2 is what we end up with. The remaining figures can be seen in Appendix, Figure A.15 to A.22. To keep the thesis concise, no pseudo code will be provided for Casestudy 1. And if we sum over all quantiles, we get the following results shown in Table 5.1.

Model	Corrected	Original	SDE Corrected
DK1 Onshore Wind Power FFNN	17557864.72	17558311.99	20755813.80
DK1 Onshore Wind Power LSTM	17378332.85	17558311.99	19741976.75
DK1 Offshore Wind Power LSTM	9619370.97	9722929.86	9627499.75
DK1 Solar Power FFNN	1419642.35	2728229.26	1411748.63
DK1 Solar Power LSTM	2866043.70	2728229.26	1429837.77
DK2 Offshore Wind Power FFNN	6538132.54	6510622.61	5438854.69
DK2 Offshore Wind Power LSTM	6571776.72	6510622.61	5044915.96
DK2 Solar Power FFNN	958146.61	958183.94	586543.16
DK2 Solar Power LSTM	961117.11	958183.94	589949.08

Table 5.1. Sum of Mean Profits for Different Models and Types

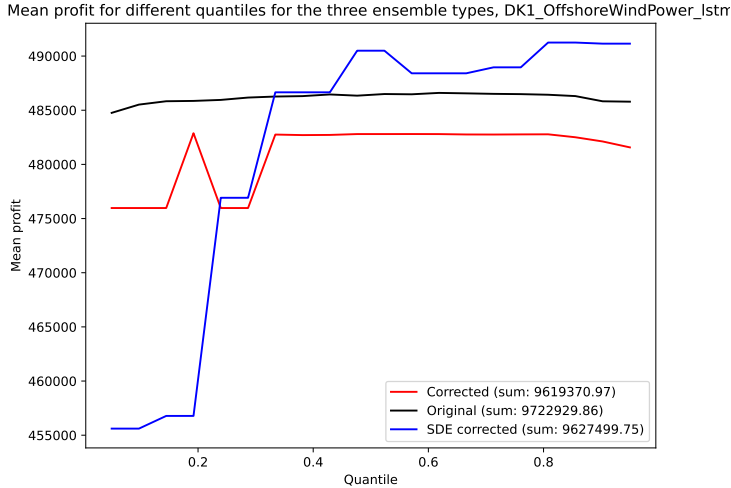


Figure 5.2. Trading according to case study 1 for DK1 Offshore Windpower LSTM ensembles. Comparing original, corrected and SDE corrected. We have plotted the mean over all time steps that we trade

In Table 5.2 the results are indexed to value 100 for the original ensembles. It is only in one case, DK1 offshore wind power LSTM, that the original ensembles are better for this casestudy.

	Model	Corrected	Original	SDE Corrected
0	DK1 Onshore Wind Power FFNN	100.00	100	118.21
1	DK1 Onshore Wind Power LSTM	98.97	100	112.44
2	DK1 Offshore Wind Power LSTM	98.93	100	99.02
3	DK1 Solar Power FFNN	52.04	100	51.75
4	DK1 Solar Power LSTM	105.05	100	52.41
5	DK2 Offshore Wind Power FFNN	100.42	100	83.54
6	DK2 Offshore Wind Power LSTM	100.94	100	77.49
7	DK2 Solar Power FFNN	100.00	100	61.21
8	DK2 Solar Power LSTM	100.31	100	61.57

Table 5.2. Sum of Mean Profits for Different Models and Types, index 100 for the original ensembles

In Figure 5.3 we have plotted the profit distribution for the 20 quantiles we focus on here. We are here comparing what happens if we trade based on the original or corrected ensembles. The plot is not perfectly illustrative, since the changes are minuscule in the broader scheme of things. We see the biggest changes around the median, where the corrected profits are biased slightly lower. As the quantiles become closer to one we see higher volatility in the trading profits (wider histograms) for the produced profit. This makes sense giving the asymmetric risk in the balancing market, as well as higher fluctuations on the upper quantiles. If you, for example, have predicted a high production on the 90% quantile, but you do not deliver, you have to buy a lot back in the balancing market, and thereby reducing profits or causing losses.

Note that the trading period is far from significant in this case study. The results are included primarily for transparency and should be taken with a grain of salt.

5.2.2 Case study 2: Energy-Trader

Even though we have established that you in all practical application will get a fill in the balancing market (at a reasonable price), it still depends on the amount of course. We will here assume that getting rid of your excess amount in the balancing market is doable, but on average not profitable, so we will try to make the most accurate prediction of the actual production one day ahead.

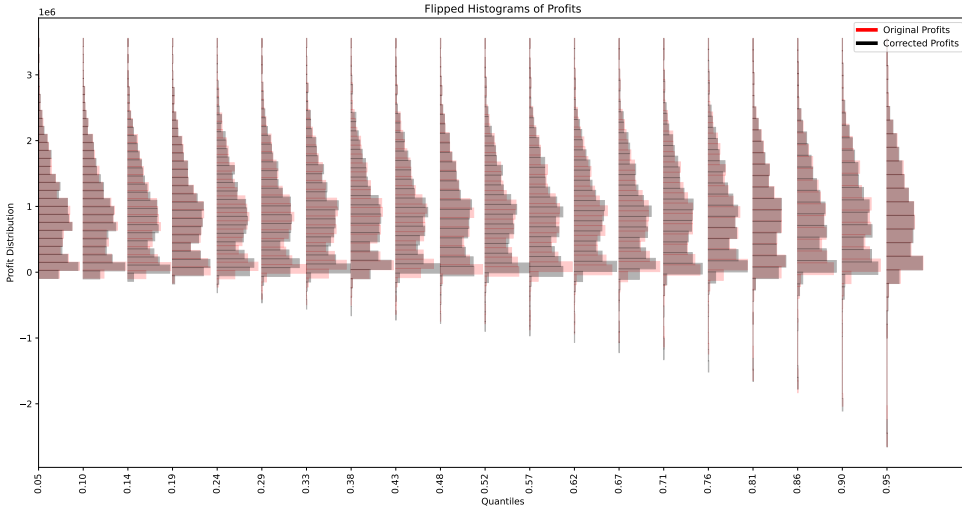


Figure 5.3. Histograms of profits for Case study 1, showing how the distribution of profits is for 20 selected quantiles.

We note, that the amount of data that we trade on here is only around 1.3 months worth of data or approximately 940 hours, which is nowhere close to significant. You would normally require at least a years worth of data. We will however continue with this framework. This also means that it is ready for deployment, once more data becomes available. The results will be briefly commented on, but they should not affect any trading decisions yet.

We will focus on DK1 here. Another big assumption is what we assume that the wind (onshore + offshore) power production is the only driver of the day-ahead spot price as well as the imbalance price. To justify this assumption we note that we are in the start of 2024, (January/February), where the sun does not shine a lot in Denmark. The average solar power production is $\sim 47\text{MW}$, while the wind is $\sim 2612\text{MW}$, thus a ~ 55 factor difference.

To predict the actual production tomorrow, we know from Section 4, that this is what our TP-SDE-CE is best at this, and we will leverage this.

In this case study, we will assume that we are an energy trader. If our median of TP-SDE-CE forecasts are higher than the original projections median, we can infer an increase in supply. According to the basic laws of supply and demand, this increased supply is expected to result in lower prices in the spot market tomorrow.

Assuming that the supply-demand laws holds, the spot price tomorrow will be lower

than initially forecasted, we would therefore sell electricity in the day-ahead market and buy it back at the lower spot price tomorrow. This strategy effectively shorts the "energy/electricity" price, and we aim to investigate its profitability.

Conversely, if we forecast lower production, indicating a decrease in supply, we can long the energy price by buying in the day-ahead market and selling at the potentially higher spot price tomorrow. This investigation seeks to determine if this strategy is also profitable.

We will not be estimating the supply and demand curves, but rather we will assume that their effects hold and trade on these beliefs. Furthermore, what we refer to as "spot price" here, is actually the balancing market, since this is where we have data available, and not the intraday market¹, as one energy trader probably would use, since the balancing market is hard to predict and with huge asymmetric volatility.

The trading strategy is presented in pseudocode in Algorithm 5. The main concepts have already been explained, and we just note that the main logic happens in line (8-15). The "DataFrame" command, is used so that we can use `pandas` data-frame environment to ensure every time-index matches up.

Employing this strategy will give the total bank results seen in Figure 5.4. We note that this trading strategy is trading only 1 MWh.

Using the bank roll as a measure of correcting ensembles is very interesting, but will be left for Section 5.3 (Future work).

Having conducted some case-studies where we apply the theory and discuss the results, we will now finish this section, with a section on Future work and adding some perspectives on the topics presented in this thesis.

5.3 Future Work

This section will serve as an inspiration for further work on this thesis, while we also comment on what we did not get to, due to time constraints.

¹We have been given access to NordPool's FTP server with the data for intraday trades and orders, but estimating years worth of intraday VWAP prices is simply not viable giving the time constraints of this thesis and the size of the datasets (approximately 368GB for the intraday orders over a four year period and estimated 7.3GB of data for the intraday trades).

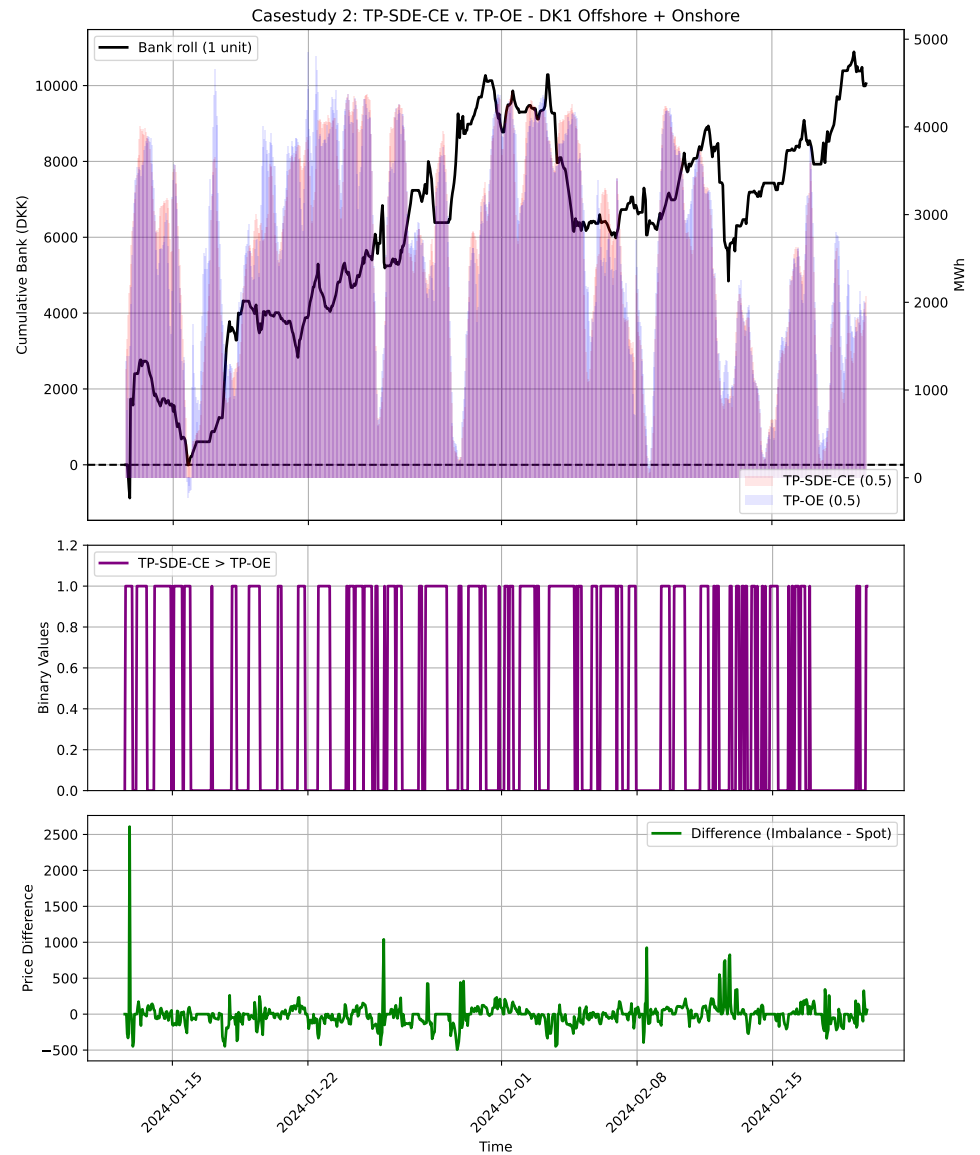


Figure 5.4. Trading strategy from Algorithm 5 applied to DK1 offshore + onshore wind power (LSTM).

Algorithm 5 Trading Strategy

Require: E : Summed (Off+Onshore) Ensembles, A : Actuals, E_{SDE} : Summed (Off+Onshore) SDE Ensembles, P : Day-ahead Spot Prices, P_I : Imbalance Prices

- 1: Let $Q = \{0.1, 0.25, 0.5, 0.75, 0.9\}$ be the set of quantiles
- 2: $(Y_{ORIG}, A_{ORIG}) \leftarrow \text{QuantilePrediction}(E, A, Q)$
- 3: $(Y_{SDE}, A_{SDE}) \leftarrow \text{QuantilePrediction}(E_{SDE}, A, Q)$
- 4: Let I be the index set of trading hours
- 5: $\hat{Y}_{SDE} \leftarrow \text{DataFrame}(Y_{SDE}, I)$
- 6: $\hat{Y}_{ORIG} \leftarrow \text{DataFrame}(Y_{ORIG}, I)$
- 7: Initialize $B \leftarrow \{\}$ as the set of bank values
- 8: **for** $i \in I$ **do**
- 9: **if** $\hat{Y}_{SDE,0.5}[i] > \hat{Y}_{ORIG,0.5}[i]$ **then**
- 10: $b_i \leftarrow -P_S[i] + P_I[i]$
- 11: **else**
- 12: $b_i \leftarrow P_S[i] - P_I[i]$
- 13: **end if**
- 14: $B \leftarrow B \cup \{b_i\}$
- 15: **end for**
- 16: $B_{total} \leftarrow \sum_{i \in I} b_i$, where $b_i = B[i]$
- 17: **Return** B_{total}

5.3.1 Going from Offline to Online Implementation

The whole idea of this 3-step corrective procedure is to let the data update the model itself i.e. *data-driven*. This is a key-point! The vision for the framework developed in this thesis is that it should run in an *online* implementation and all parameter are continuously adapting to the current environment. Therefore if we sometimes have spoken of data-driven neural network it should be seen in the context of the online implementation, where we in this section will introduce actual data-driven neural network types.

Neural networks weights.

With the neural networks we currently are using they need re-training now and again. We will here just mention that in the previous four years there have been a huge development in letting the weights be "liquid" i.e. able to change over time, and thus named Liquid Neural Networks (LNN). Instead of describing the system dynamics with non-linearities (as weights and activation functions), they instead "construct networks of linear first-order dynamical systems modulated via nonlinear interlinked gates", from the original paper, (Hasani et al., 2020). The outputs of these LNNs is computed by numerical differential equation solvers on a running basis, always ensuring the optimal "weights" for the given environment.

SDE parameters.

The SDE parameters we fit in this thesis is also only done once due to time constraints. But the goal of having SDEs is that Chen et al. (2018) and Särkkä (2006) show that it is possible to do a recursive and adaptive parameter estimation. This should also be implemented.

TAQR algorithm.

The inspiration to the online implementation came from the fast, time-adaptive implementation of quantile regression. This online implementation will serve a key role in the final online implementation.

5.3.2 Ideas for Improvement

In the beginning of 2022 the successor to N-BEATS was presented: Neural Hierarchical interpolation for Time Series (N-HITS). N-BEATS was famous, since it was the first deep learning model that finally succeeded in beating traditional statistics frameworks for time series forecasting. N-BEATS stands for Neural Basis Expansion Analysis for Interpretable Time Series, and the original paper is (Oreshkin et al., 2019). The Neural Basis Expansion part of N-BEATS is what really make this suitable for this thesis. N-HITS improved accuracy and lowered computational costs compared to N-BEATS, see the original paper (Challu et al., 2022). N-HITS does this by using hierarchical reconciliation across different time frequencies, so that it faster can learn long and short term dependencies of the time series.

Incorporating these models in a "liquid" framework, seem like the ultimate setup that this thesis could benefit from.

Considering the huge volatile shift in energy prices through 2022, see Figure 5.5. This leads us to believe that were something like that, albeit unlikely, to happen again, we would like the model to detect regime changes. We could incorporate Hidden Markov Models or e.g. non-linear SETAR models.

Since our TAQR algorithm excels in predicting the actuals values, a big thing that would be very interesting to investigate is a *point-forecast-driven logistic SDE* as introduced in (Sørensen et al., 2023). Hopefully this could improve the scores further.

A natural way to go from here, given many models that forecasts the same thing, would be to consider ensemble machine learning models, like random forest models. A training measure could be our own weighted score e.g. $S = 0.2 \cdot \text{CSS} + 0.4 \cdot \text{CRPS} + 0.4 \cdot \text{Variogram}$.

In Figure 4.8 and especially 4.7 we saw sudden spikes when we did predictions from

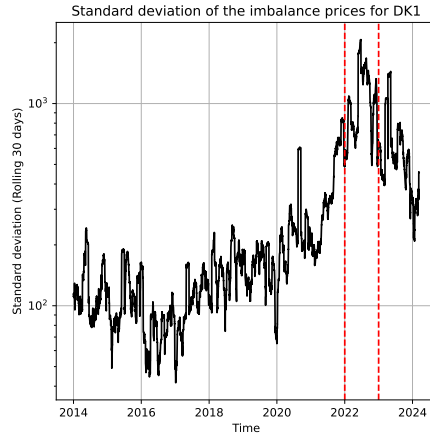


Figure 5.5. Rolling 30 day standard deviation for the imbalance price in DKK for DK1. Red lines indicate start and end of 2022.

the TAQR algorithm. To alleviate this problem we propose to introduce regularization in the LP formulation of the TAQR problem, penalizing observations with huge differences from one observation to the next.

The non-linear quantile regression that was introduced in Section 3.1.2, was introduced since we in the beginning believed that we could transform our basis functions into splines in a smart way in an online implementation. We realized that given the shape of the ensembles, we already had enough to describe the quantile regression linearly, based on the 50 quantiles, and thus we did not see the need to incorporate the non-linear quantile regression, even though it might have improved performance, as it drastically did in (J. K. Møller, 2006).

Introducing a more specific loss function could be very interesting, such as the bank roll from case study 2 or the score, S , just introduced. This was not done here, since we rely on the gradient estimation through backpropagation in the TensorFlow framework for faster training. The other "loss-functions" are not differentiable, and thus needs gradient free optimizations technique, which in general performs worse and slower.

We will just briefly mention here that more thought should be given to how we handle the design matrix in the updating procedure of the simplex algorithm. However, since the algorithm succeeded anyway, due to time constraints, no ideas have been developed yet. In some cases of the previous case studies especially in case study 2, we did not have the ability to produce TAQR estimates for some of the areas and neural

network combinations, due to singularity of $\mathbf{X}(h)$. We do not have a sure-safe way of fixing this yet. What we experienced was that many ensembles, when corrected by the neural network framework, were collinear in the basis. It could also be the case that some of them were simply 0 for all time-steps. Neither of these effects are to be desired. One work around used in this thesis, was that if this happen, i.e. that the basis was singular, and therefore could not be inverted, was that we selected every other non-zero ensembles instead. Since we had in the range of 20-50 ensembles to choose from this alleviated most problems for the specified case studies and results in this thesis.

Lastly, we will mention, that the three-step corrective procedure really can be applied in a lot of different fields. This thesis chose the application presented here; error-correction of forecasts in renewable energy, however there is nothing in the way for us to apply the theory to other areas of the forecasting field.

Due to time constraints a simulation study was not carried out in this thesis.

CHAPTER 6

Conclusion

The primary objective of this thesis was to leverage time-adaptive, data-driven neural networks and quantile regression to improve day-ahead forecasts for wind and solar power production using a model-free approach. Additionally, we aimed to assess the profitability of energy-trading strategies under specific assumptions.

We developed a three-step corrective procedure using both Feed Forward and Long Short-Term Memory neural networks to data-drivenly update and find optimal basis function from our forecasted energy production ensembles. When considering aspects that relied on proper estimation of the correlation structure, we improved the ensembles' Variogram score by fitting an Ornstein-Uhlenbeck process to the residuals and simulating from it to obtain our SDE-corrected ensembles.

For predicting the actual value of energy production, we deployed our Python version of a Time Adaptive Quantile Regression algorithm, which provided significantly better results than, for instance, the median from the original ensembles. The results demonstrate the effectiveness of the NN-SDE-TAQR combination in enhancing forecast accuracy for renewable energy resources.

In the two case studies, we evaluated the implications of updating our ensembles under different scenarios. While theoretically improved quantiles and predictions sometimes translated to profitable trading, the limited sample size prevented the results from being statistically significant. This limitation arose from the nature of time series data and the extensive data requirements for training neural networks.

The methodology employed in this research, particularly the use of Python for data-driven basis corrections and SDE simulations, proved to be robust and reproducible. Challenges such as handling singularity in the basis function matrix, $\mathbf{X}$, and collinearity in ensembles were encountered and addressed through specific workarounds.

This thesis contributes to the renewable energy forecasting field by introducing a set of new methodologies that combines neural networks, stochastic differential equations, and time-adaptive quantile regression. This synergistic method sets creates better forecasts as measured by CRPS, Variogram and Quantile score.

Future research should explore the integration of additional types of neural networks and the application of this methodology to larger datasets to verify the profitability of the trading strategies presented in the case studies. Collaborations with industry partners could provide more extensive data and practical insights, further validating the approach.

In conclusion, this research has demonstrated the potential of advanced error correction techniques in improving energy forecast accuracy. Through conducting this research it has been rewarding to provide valuable insights into the complexities of energy forecasting. The innovative methods developed here contributes to more reliable and efficient renewable energy systems and supports global sustainability goals.

Finally, it is with great excitement that we later will publish this Python package for the entire field of forecasters and practitioners.

Bibliography

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous distributed systems [Accessed: 2024-06-13].
- Alhamzawi, R., Yu, K., & Benoit, D. F. (2011). *Bayesian adaptive lasso quantile regression* (Working Paper No. 2011/728). Faculty of Economics and Business Administration, Ghent University. Ghent, Belgium. <https://www.ugent.be/eb/en/research/working-papers>
- Alitasb, G. K., & Salau, A. O. (2024). Multiple-input multiple-output radial basis function neural network modeling and model predictive control of a biomass boiler. *Energy Reports*, 11, 442–451. <https://doi.org/https://doi.org/10.1016/j.egy.2023.11.063>
- Avis, D., & Chvátal, V. (1978). Notes on bland's pivoting rule. In M. L. Balinski & A. J. Hoffman (Eds.), *Polyhedral combinatorics: Dedicated to the memory of d.r. fulkerson* (pp. 24–34). Springer Berlin Heidelberg. <https://doi.org/10.1007/BFb0121192>
- Bjerregård, M. B., Møller, J. K., & Madsen, H. (2021). An introduction to multivariate probabilistic forecast evaluation. *Energy and AI*, 4, 100058. <https://doi.org/https://doi.org/10.1016/j.egyai.2021.100058>
- Bremnes, J. B. (2020). Ensemble postprocessing using quantile function regression based on neural networks and bernstein polynomials. *Monthly Weather Review*, 148(1), 403–414. <https://doi.org/10.1175/MWR-D-19-0227.1>
- Broughton, R. (2024). Flowchart shapes and description [Accessed: 2024-06-30]. <https://www.quality-assurance-solutions.com/flowchart-shapes.html>
- Cannon, A. J. (2011). Quantile regression neural networks: Implementation in r and application to precipitation downscaling. *Computers & Geosciences*, 37(9), 1277–1284. <https://doi.org/10.1016/j.cageo.2010.07.005>
- Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler, M., & Dubrawski, A. (2022). N-hits: Neural hierarchical interpolation for time series forecasting. *CoRR*, abs/2201.12886. <https://arxiv.org/abs/2201.12886>
- Chen, M., Chow, S.-M., & Hunter, M. D. (2018). Stochastic differential equation models with time-varying parameters. In K. van Montfort, J. H. L. Oud, &

- M. C. Voelkle (Eds.), *Continuous time modeling in the behavioral and related sciences* (pp. 205–238). Springer International Publishing. https://doi.org/10.1007/978-3-319-77219-6_9
- Chhabra, J. (2017). Understanding lstm in tensorflow (mnist dataset) [Accessed: 2024-07-08].
- Clark, E. (2024). Eoltech study finds high predictability in wind power production [Accessed: 2024-07-03]. *Energy Theory*. <https://energytheory.com/eoltech-study-finds-high-predictability-in-wind-power-production/>
- ECMWF. (2023). Ens - ensemble forecasts [Accessed: 2024-07-05]. <https://confluence.ecmwf.int/display/FUG/Section+2.1.2.1+ENS+-+Ensemble+Forecasts>
- ECMWF. (2024). Ecmwf: European centre for medium-range weather forecasts [Accessed: 2024-07-01].
- Energidataservice. (2024). Energidataservice: The danish energy data service [Accessed: 2024-07-01].
- Energinet. (2023). Oversigt over systemydelser [Accessed: 2024-07-05]. <https://energinet.dk/el/systemydelser/introduktion-til-systemydelser/oversigt-over-systemydelser/>
- ENTSO-E. (2024). Entso-e: European network of transmission system operators for electricity [Accessed: 2024-07-01].
- Fakhoury, D., Fakhoury, E., & Speleers, H. (2022). Exspline: An interpretable and expressive spline-based neural network. *Neural Networks*, 152, 332–346. <https://doi.org/https://doi.org/10.1016/j.neunet.2022.04.029>
- Gade, P. A. V., Bindner, H. W., & Kazempour, J. (2024). Leveraging p90 requirement: Flexible resources bidding in nordic ancillary service markets. <https://arxiv.org/abs/2404.12807>
- Giles, D. E., Koenker, R., Chernozhukov, V., He, H., & Peng, L. (2018). Review of: Roger koenker, victor chernozhukov, huming he and limin peng (2017): Handbook of quantile regression. *Statistical Papers*, 59(2), 849–850. <https://doi.org/10.1007/s00362-018-1004-2>
- Hasani, R. M., Lechner, M., Amini, A., Rus, D., & Grosu, R. (2020). Liquid time-constant networks. *CoRR*, abs/2006.04439. <https://arxiv.org/abs/2006.04439>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9, 1735–80. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Hubert, M., Pison, G., Struyf, A., & Aelst, S. (Eds.). (2004). *Theory and applications of recent robust methods* (1st ed.) [Springer Basel AG 2004, eBook ISBN: 978-3-0348-7958-3, Published: 06 December 2012, Softcover ISBN: 978-3-0348-9636-8, Published: 05 November 2012]. Birkhäuser Basel. <https://doi.org/10.1007/978-3-0348-7958-3>
- Institute for Energy Economics and Financial Analysis. (2023). *Renewable energy sector in the philippines is poised to grow faster as investor interest rises*. <https://ieefa.org/articles/renewable-energy-sector-philippines-poised-grow-faster-investor-interest-rises>
- International Renewable Energy Agency (IRENA). (2020). *Advanced forecasting of variable renewable power generation: Innovation landscape brief*. International

- Renewable Energy Agency. https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Jul/IRENA_Advanced_weather_forecasting_2020.pdf
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*. <http://arxiv.org/abs/1502.03167>
- Iversen, E. B., Morales, J. M., Møller, J. K., & Madsen, H. (2016). Short-term probabilistic forecasting of wind speed using stochastic differential equations. *International Journal of Forecasting*, *32*(3), 981–990. <https://doi.org/https://doi.org/10.1016/j.ijforecast.2015.03.001>
- Iversen, E. B., Morales, J. M., Møller, J. K., Trombe, P.-J., & Madsen, H. (2017). Leveraging stochastic differential equations for probabilistic forecasting of wind power using a dynamic power curve. *Wind Energy*, *20*(1), 33–44. <https://doi.org/https://doi.org/10.1002/we.1988>
- Iversen, J., Morales González, J., Møller, J., Trombe, P.-J., & Madsen, H. (2017). Leveraging stochastic differential equations for probabilistic forecasting of wind power using a dynamic power curve. *Wind Energy*, *20*(1), 33–44. <https://doi.org/10.1002/we.1988>
- Johansen, S. (1985). *The mathematical structure of error correction models* [Preprint No. 6]. Institute of Mathematical Statistics, University of Copenhagen.
- Jørgensen, B. (2024a). Best way to run r-scripts from python in 2024 [Accessed: 2024-06-11]. <https://medium.com/p/b31849864205>
- Jørgensen, B. (2024b). Github repository s194357-msc-thesis [Accessed: 2024-07-05]. <https://github.com/bast0320/S194357-MSc-Thesis>
- Kandpal, V., Jaswal, A., Santibanez Gonzalez, E. D. R., & Agarwal, N. (2024). The economics of sustainable energy transition and the circular economy. In *Sustainable energy transition: Circular economy and sustainable financing for environmental, social and governance (esg) practices* (pp. 53–87). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-52943-6_2
- Karunasingha, D. S. K. (2022). Root mean square error or mean absolute error? use their ratio as well. *Information Sciences*, *585*, 609–629. <https://doi.org/https://doi.org/10.1016/j.ins.2021.11.036>
- Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.
- Koenker, R. (2005). *Quantile regression*. Cambridge University Press.
- Koenker, R., & Bassett Jr., G. (1978). Regression quantiles. *Econometrica*, *46*(1), 33–50. <https://doi.org/10.2307/1913643>
- Kolkowska, N. (2023). Grid integration of renewable energy. *Sustainable Review*. <https://sustainablereview.com/grid-integration-of-renewable-energy/>
- Lauret, P., David, M., & Pedro, H. (2017). Probabilistic Solar Forecasting Using Quantile Regression Models. *Energies*, *10*(10). <https://doi.org/10.3390/en10101591>
- Löwe, R., Mikkelsen, P. S., & Madsen, H. (2014). Stochastic rainfall-runoff forecasting: Parameter estimation, multi-step prediction, and evaluation of overflow

- risk. *Stochastic Environmental Research and Risk Assessment*, 28(3), 505–516. <https://doi.org/10.1007/s00477-013-0768-0>
- Lunde, G. A., Damm, E. V., Gade, P. A. V., & Kazempour, J. (2024). Aggregator of electric vehicles bidding in nordic fcr-d markets: A chance-constrained program.
- Madsen, H. (2007). *Time series analysis*. Chapman Hall.
- McKinsey. (2020). Integrating renewable energy sources into grids. *McKinsey*. <https://www.mckinsey.com/business-functions/sustainability/our-insights/integrating-renewable-energy-sources-into-grids>
- Ministry of Foreign Affairs of Denmark. (2024). New climate action strategy [Accessed: 2024-07-05].
- Møller, J. K. (2006). *Modeling of uncertainty in wind energy forecast* [DTU supervisors: Henrik Madsen, hmad@dtu.dk, DTU Compute, and Henrik Aalborg Nielsen]. <http://www.compute.dtu.dk/english>
- Møller, J. (2011). *Stochastic state space modelling of nonlinear systems - with application to marine ecosystems* (Publication No. 246) [Doctoral dissertation]. Technical University of Denmark.
- Møller, J., Nielsen, H., & Madsen, H. (2006). *Algorithms for adaptive quantile regression - and a matlab implementation*.
- Møller, J. K., Zugno, M., & Madsen, H. (2016). Probabilistic forecasts of wind power generation by stochastic differential equation models. *Journal of Forecasting*, 35(3), 189–205. <https://doi.org/https://doi.org/10.1002/for.2367>
- Nielsen, H. A., Madsen, H., & Torben, S. (2006). Using quantile regression to extend an existing wind power forecasting system with probabilistic forecasts. *Wind Energy*, 9(1-2), 95–108. <https://doi.org/https://doi.org/10.1002/we.180>
- Nielsen, H. A., Nielsen, T. S., Madsen, H., Giebel, G., Badger, J., Landberg, L., Sattler, K., Voulund, L., & Tofting, J. (2006). From wind ensembles to probabilistic information about future wind power production – results from an actual application. *2006 International Conference on Probabilistic Methods Applied to Power Systems*, 1–8. <https://doi.org/10.1109/PMAPS.2006.360289>
- Nielsen, M. (2015). Neural networks and deep learning [Accessed: 2024-06-13].
- Nord Pool Group. (2023a). Day-ahead market [Accessed: 2024-07-05]. <https://www.nordpoolgroup.com/en/the-power-market/Day-ahead-market/>
- Nord Pool Group. (2023b). Intraday trading [Accessed: 2024-07-05]. <https://www.nordpoolgroup.com/en/trading/intraday-trading/>
- Nord Pool Group. (2023c). Manual frequency restoration reserve [Accessed: 2024-07-05]. <https://data.nordpoolgroup.com/power-system/manual-frequency-restoration-reserve?deliveryDate=latest&deliveryAreas=&viewMode=All>
- Nounou, M. N., & Bakshi, B. R. (2000). Chapter 5 - multiscale methods for denoising and compression. In B. Walczak (Ed.), *Wavelets in chemistry* (pp. 119–150, Vol. 22). Elsevier. [https://doi.org/https://doi.org/10.1016/S0922-3487\(00\)80030-1](https://doi.org/https://doi.org/10.1016/S0922-3487(00)80030-1)
- Øksendal, B. (2003). *Stochastic differential equations: An introduction with applications* (Sixth). Springer.

- OpenAI. (2024). Openai: Artificial intelligence research and deployment [Accessed: 2024-07-05].
- Oreshkin, B. N., Carпов, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: neural basis expansion analysis for interpretable time series forecasting. *CoRR, abs/1905.10437*. <http://arxiv.org/abs/1905.10437>
- Rajawat, A. S., Mohammed, O., Shaw, R. N., & Ghosh, A. (2022). Chapter six - renewable energy system for industrial internet of things model using fusion-ai. In R. N. Shaw, A. Ghosh, S. Mekhilef, & V. E. Balas (Eds.), *Applications of ai and iot in renewable energy* (pp. 107–128). Academic Press. <https://doi.org/https://doi.org/10.1016/B978-0-323-91699-8.00006-1>
- Särkkä, S. (2006, April). *Recursive bayesian estimation in discrete-continuous filtering problems* [Doctoral dissertation, Helsinki University of Technology]. <https://urn.fi/urn:nbn:fi:tkk-006778>
- Scheuerer, M., & Hamill, T. M. (2015). Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities. *Monthly Weather Review*, 143(4), 1321–1334. <https://doi.org/10.1175/MWR-D-14-00269.1>
- Sinkin, D. (2024). Long short-term memory (lstm) [Accessed: 2024-07-08].
- Smilkov, D., & Carter, S. (2023). A neural network playground [Accessed: 2024-06-30].
- Sørensen, M. L., Nystrup, P., Bjerregård, M. B., Møller, J. K., Bacher, P., & Madsen, H. (2023). Recent developments in multivariate wind and solar power forecasting. *WIREs Energy and Environment*, 12(2), e465. <https://doi.org/https://doi.org/10.1002/wene.465>
- Stavrev, S., & Ginchev, D. (2024). Reinforcement learning techniques in optimizing energy systems. *Electronics*, 13(8). <https://doi.org/10.3390/electronics13081459>
- Stigler, S. M. (1984). Studies in the history of probability and statistics XL Boscovich, Simpson and a 1760 manuscript note on fitting a linear relation. *Biometrika*, 71(3), 615–620. <https://doi.org/10.1093/biomet/71.3.615>
- Thygesen, U. (2023, April). *Stochastic differential equations for science and engineering*. <https://doi.org/10.1201/9781003277569>
- Valdivieso, L., Schoutens, W., & Tuerlinckx, F. (2009). Maximum likelihood estimation in processes of ornstein-uhlenbeck type. *Statistical Inference for Stochastic Processes*, 12(1), 1–19. <https://doi.org/10.1007/s11203-008-9021-8>
- Van de Geer, S. A. (2003). Adaptive quantile regression. In M. G. Akritas & D. N. Politis (Eds.), *Recent advances and trends in nonparametric statistics* (pp. 235–248). Elsevier Science B.V.
- Vuckovic, A., Popović, D., & Radivojevic, V. (2002, October). Artificial neural network for detecting drowsiness from eeg recordings [ISBN: 0-7803-7593-9]. <https://doi.org/10.1109/NEUREL.2002.1057990>

Appendices

APPENDIX **A**

A.1 Code Overview

File Name	Lines of Code
/backup_crps_variogram.py	106
/data_description.py	51
/fix_data.py	120
/function_for_SDE_correction.py	551
/functions_for_TAQR.py	1068
/helper_functions.py	162
/multimodel_quantile_loss_network.py (abbrv: mqls)	130
/mqls_working.py	178
/mqls_TDS_article.py	532
/mqls_TDS_article_functions.py	1058
/pipeline_start_to_finish.py	872
/test.py	102
/test2.py	17
/test3_SDE_heatmap_plotting.py	93
/test4_polyhedra.py	129
/test5_simulating_SDE_plot.py	54
/test6_polyhedra_v3.5.py	141
/test_7_latex_table.py	38
Total (.py)	7009
/SDE_correction_for_TAQR.ipynb	754
/SDE_ensembles.ipynb	305
/SDE_on_residuals.ipynb	878
/TAQR_toy_example.ipynb	290
/casestudy2/casestudy_2_energy_trader.ipynb	338
/comparing_TAQR_orig_vs_corr_ensembles_section4.3_plots.ipynb	299
/data_stats_for_chap_2.ipynb	171
/example_for_TAQR.ipynb	76
/generating_plots_for_results_section.ipynb	516
/looking_at_the_models.ipynb	39
/pipeline_flowchart.ipynb	288
/trading_on_quantiles.ipynb	564
/Quantile Loss.ipynb	152
/N_BEATS_NHITS_NN.ipynb	20
Total (.ipynb)	4690

Table A.1. Number of Lines of Code in MscThesisCode_NN (Total: 11699)

To calculate the lines of code in our .py-files, we used the command line `git ls-files | grep '\.py' | xargs wc -l` and for .ipynb-files we used the code in the file `loc.py`.

A.2 Methods & Theory

A.2.1 Proof for Quantile Regression

To show that the τ th quantile of a real-valued random variable Y minimizes the expected loss defined by $L_\tau(m) = m(\tau - \mathbb{I}_{(m < 0)})$, The expected loss when comparing Y to a candidate quantile $f(x)$ is given by

$$\mathbb{E}(L_\tau(Y - f(x))).$$

We can write the expected loss as

$$\mathbb{E}(L_\tau(Y - f(x))) = \mathbb{E}((Y - f(x))(\tau - \mathbb{I}_{(Y < f(x))})).$$

Breaking this into two parts yields

$$\mathbb{E}(L_\tau(Y - f(x))) = (\tau - 1)\mathbb{E}[(Y - f(x))\mathbb{I}_{(Y < f(x))}] + \tau\mathbb{E}[(Y - f(x))\mathbb{I}_{(Y \geq f(x))}].$$

Using the cumulative distribution function $F_Y(y)$, we can express the expected loss as

$$\mathbb{E}(L_\tau(Y - f(x))) = (\tau - 1) \int_{-\infty}^{f(x)} (y - f(x)) dF_Y(y) + \tau \int_{f(x)}^{\infty} (y - f(x)) dF_Y(y).$$

Differentiating the expected loss function with respect to $f(x)$ gives

$$\frac{d}{df(x)} \mathbb{E}(L_\tau(Y - f(x))) = \frac{d}{df(x)} \left[(\tau - 1) \int_{-\infty}^{f(x)} (y - f(x)) dF_Y(y) + \tau \int_{f(x)}^{\infty} (y - f(x)) dF_Y(y) \right].$$

Using Leibniz's integral rule, which allows us to differentiate under the integral sign, we have for the first integral,

$$\begin{aligned} \frac{d}{df(x)} \left[(\tau - 1) \int_{-\infty}^{f(x)} (y - f(x)) dF_Y(y) \right] &= (\tau - 1) \left[\int_{-\infty}^{f(x)} \frac{\partial}{\partial f(x)} (y - f(x)) dF_Y(y) \right. \\ &\quad \left. + (y - f(x)) \Big|_{y=f(x)} f_Y(f(x)) \right]. \end{aligned}$$

The term $\frac{\partial}{\partial f(x)}(y-f(x)) = -1$, and evaluating at $y = f(x)$ gives $(f(x)-f(x))f_Y(f(x)) = 0$:

$$= (\tau - 1) \left[\int_{-\infty}^f (x)(-1)dF_Y(y) \right] = (\tau - 1) [-F_Y(f(x))] = (1 - \tau)F_Y(f(x)).$$

Similarly, for the second integral,

$$\begin{aligned} \frac{d}{df(x)} \left[\tau \int_{f(x)}^{\infty} (y - f(x))dF_Y(y) \right] &= \tau \left[\int_{f(x)}^{\infty} \frac{\partial}{\partial f(x)}(y - f(x))dF_Y(y) + (y - f(x)) \Big|_{y=f(x)} (-f_Y(f(x))) \right] \\ &= \tau \left[\int_u^{\infty} (-1)dF_Y(y) \right] = \tau [-(1 - F_Y(f(x)))] = -\tau(1 - F_Y(f(x))). \end{aligned}$$

We set the derivative to 0 and solve for $f(x)$,

$$(1 - \tau)F_Y(f(x)) - \tau(1 - F_Y(f(x))) = 0.$$

Simplifying, we get,

$$(1 - \tau)F_Y(f(x)) = \tau(1 - F_Y(f(x))),$$

$$(1 - \tau)F_Y(f(x)) = \tau - \tau F_Y(f(x)),$$

$$F_Y(f(x)) = \tau.$$

Hence, $f(x) = F_Y^{-1}(\tau)$.

Therefore, the τ th quantile $q_Y(\tau)$ of Y is the value that minimizes the expected loss $\mathbb{E}(L_\tau(Y - f(x)))$:

$$q_Y(\tau) = \arg \min_u \mathbb{E}(L_\tau(Y - f(x))).$$

$$\min \quad \tau \mathbf{1}^T \mathbf{r}^+ + (1 - \tau) \mathbf{1}^T \mathbf{r}^- \tag{A.1}$$

$$\text{s.t.} \quad \mathbf{X}\beta + \mathbf{r}^+ - \mathbf{r}^- = \mathbf{y}, \tag{A.2}$$

$$(\mathbf{r}^+, \mathbf{r}^-) \in \mathbb{R}_+^{2n}, \tag{A.3}$$

$$\beta \in \mathbb{R}^K. \tag{A.4}$$

A.2.2 Equations for Simplex Algorithm

These calculations are presented here for completeness, to be read alongside Algorithm 1.

Step 1.

We only need the first K elements of $\mathbf{g}$, due the structure of $\mathbf{C}$. These are

$$\mathbf{g}(h) = -\mathbf{X}(h)^{-T} \mathbf{X}(\bar{h})^T \mathbf{P} \rho_\tau(\text{sign}(\mathbf{r}(\bar{h}))). \quad (\text{A.5})$$

Now,

$$\mathbf{d} = \begin{bmatrix} \tau \mathbf{e}_K - \mathbf{g}(h) \\ (1 - \tau) \mathbf{e}_K + \mathbf{g}(h) \end{bmatrix} \quad (\text{A.6})$$

Step 2.

Use the formula provided in Algorithm 1, then set

$$\tilde{s}_i = \begin{cases} \tilde{s}_i & \text{if } \tilde{s}_i \leq K \\ \tilde{s}_i - K & \text{if } \tilde{s}_i > K \end{cases}$$

and $\mathbf{h} = \mathbf{B}_{:, \tilde{s}}^{-1} \mathbf{P}_s$.

Step 3.

If it happens that α comes back as ∞ , which in theory should not be possible, since the problem is naturally bounded, the algorithm goes straight to the updating procedure.

A.2.3 Equations for Updating Step in Simplex Algorithm

Step 3.

Letting l be the row that is leaving the design matrix, then let $\Omega = \{1, 2, \dots, N\}$, where N is the number of observations in the design matrix, $\mathbf{X}$. Let the updated residual be $r_{t+1} = y_{t+1} - \mathbf{x}_{t+1}^T \hat{\beta}_t$, then to complete Step 3 in Algorithm 2,

$$\begin{aligned} \mathbf{X}^{(t+1)} &= \begin{bmatrix} \mathbf{X}_{\Omega \setminus l}^{(t)} \\ \mathbf{x}_{t+1} \end{bmatrix} \\ \mathbf{P}^{(t+1)} &= \begin{bmatrix} \mathbf{P}_{\Omega \setminus l}^{(t)} \\ \text{sign}(r_{t+1}) \end{bmatrix} \\ \mathbf{c}_{\mathcal{B}}^{(t+1)} &= \begin{bmatrix} \mathbf{c}_{\mathcal{B}; \Omega \setminus l}^{(t)} \\ \rho_\tau(r_{t+1}) \end{bmatrix}. \end{aligned} \quad (\text{A.7})$$

Finally, we update the index set h as

$$h_j^{t+1} = \begin{cases} h_j^t & \text{if } h_j < l \\ h_j^t - 1 & \text{if } h_j > l \end{cases} . \quad (\text{A.8})$$

We refer to (J. K. Møller, 2006) for a more detailed explanation and proof for some of these statements.

A.2.4 Illustration of Simplex

We aim to solve the following linear programming problem using the Simplex algorithm. The steps up to the final solution can be seen in Figure A.1. We provided the algorithm with the starting point $(x_1, x_2) = (0, -2)$. It is a feasible starting point, but it not on the edge of the vertices in the constraint polyhedron, which has its acceptable area to the left of the red, orange, green and blue constraint.

$$\begin{array}{ll} \text{Maximize} & 3x_1 + 4x_2 \\ & 2x_1 + x_2 \leq 8, \\ & x_1 + 3x_2 \leq 12, \\ \text{subject to} & x_1 - x_2 \leq 3, \\ & -x_1 + 2x_2 \leq 6, \\ & x_1, x_2 \geq 0. \end{array}$$

To solve this problem, we convert it to a standard form by adding slack variables s_1, s_2, s_3, s_4 :

$$\begin{aligned} 2x_1 + x_2 + s_1 &= 8, \\ x_1 + 3x_2 + s_2 &= 12, \\ x_1 - x_2 + s_3 &= 3, \\ -x_1 + 2x_2 + s_4 &= 6, \\ x_1, x_2, s_1, s_2, s_3, s_4 &\geq 0. \end{aligned}$$

The objective function becomes:

$$\text{Maximize } 3x_1 + 4x_2 + 0s_1 + 0s_2 + 0s_3 + 0s_4.$$

Initial Basic Feasible Solution

We choose an initial basic feasible solution:

$$\begin{aligned}x_1 &= 0, \\x_2 &= -2.\end{aligned}$$

The corresponding slack variables are:

$$\begin{aligned}s_1 &= 8 - (2 \cdot 0 + 1 \cdot (-2)) = 10, \\s_2 &= 12 - (1 \cdot 0 + 3 \cdot (-2)) = 18, \\s_3 &= 3 - (1 \cdot 0 - 1 \cdot (-2)) = 5, \\s_4 &= 6 - (-1 \cdot 0 + 2 \cdot (-2)) = 2.\end{aligned}$$

Thus, the initial solution vector is:

$$\mathbf{x} = [0 \quad -2 \quad 10 \quad 18 \quad 5 \quad 2]^T.$$

The initial basic variables are s_1, s_2, s_3, s_4 .

Simplex Algorithm Steps

The steps of the Simplex algorithm involve:

1. Identifying the entering variable.
2. Identifying the leaving variable.
3. Updating the solution.

This process is repeated until the optimal solution is found or it is determined that the problem is unbounded.

To download the .gif-file showing the steps, see https://www.dropbox.com/scl/fi/owonprnb3v9zlwkyt8zs/simplex_animation.gif?rlkey=80agwg5yto1sexh0r2i7qpxwp&dl=0.

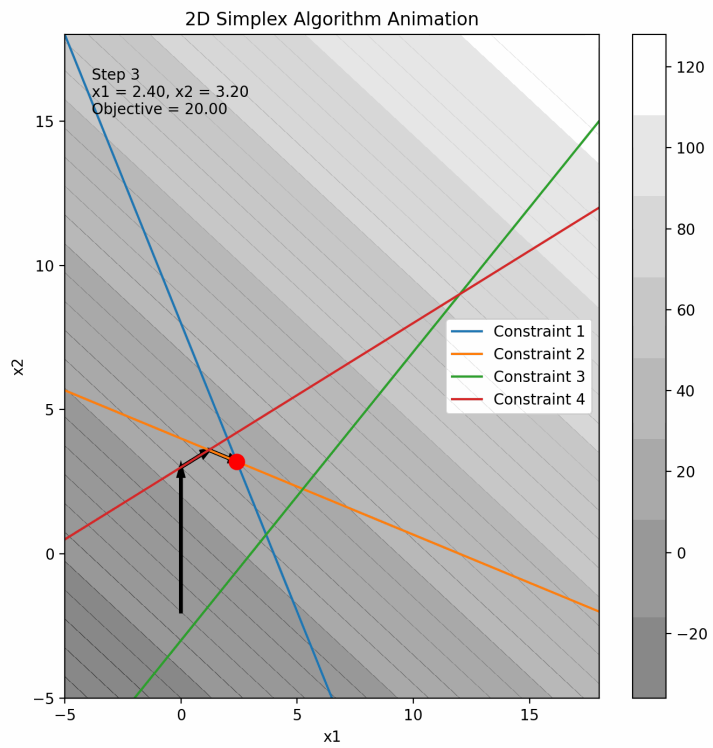


Figure A.1. The full steps from the simplex steps.

A.2.5 SDE Parameters

	0.1	0.25	0.5	0.75	0.9
DK1 OffshoreWindPower LSTM	[0.01, 103.28]	[0.03, 108.19]	[0.04, 110.86]	[0.04, 109.95]	[0.04, 108.90]
DK1 OnshoreWindPower LSTM	[0.02, 189.40]	[0.02, 189.56]	[0.02, 189.79]	[0.02, 192.40]	[0.02, 194.02]
DK2 OffshoreWindPower LSTM	[0.01, 69.72]	[0.01, 84.39]	[0.03, 99.06]	[0.03, 101.90]	[0.03, 101.24]
DK1 OnshoreWindPower FFNN	[0.05, 117.55]	[0.09, 114.50]	[0.10, 114.75]	[0.05, 121.00]	[0.03, 134.26]
DK2 OffshoreWindPower FFNN	[0.04, 111.67]	[0.03, 88.16]	[0.05, 83.86]	[0.06, 88.52]	[0.06, 101.67]
DK1 SolarPower FFNN	[0.11, 192.35]	[0.11, 197.63]	[0.12, 205.12]	[0.13, 214.35]	[0.14, 225.41]
DK2 SolarPower FFNN	[0.15, 54.42]	[0.17, 52.80]	[0.20, 51.78]	[0.21, 51.55]	[0.20, 52.05]
DK1 SolarPower LSTM	[0.05, 163.57]	[0.08, 177.19]	[0.12, 201.46]	[0.13, 213.67]	[0.14, 223.23]
DK2 SolarPower LSTM	[0.11, 56.31]	[0.16, 52.53]	[0.19, 51.47]	[0.21, 51.67]	[0.21, 52.76]

Table A.2. Rounding of each parameter for SDE residuals to 2 decimal points for $[\lambda, \sigma]$ parameters.

Normal vs. fitted t-distribution.
T-distribution parameters: (2.206, -0.061, 26.492).

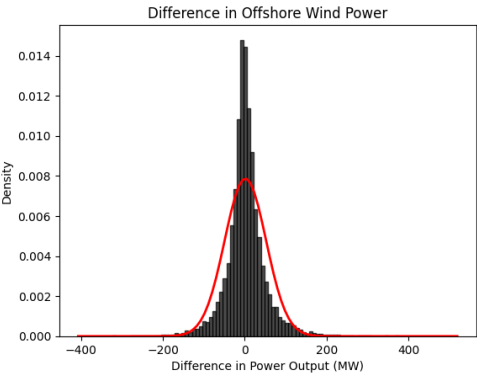


Figure A.2. Fitted normal to pairwise difference

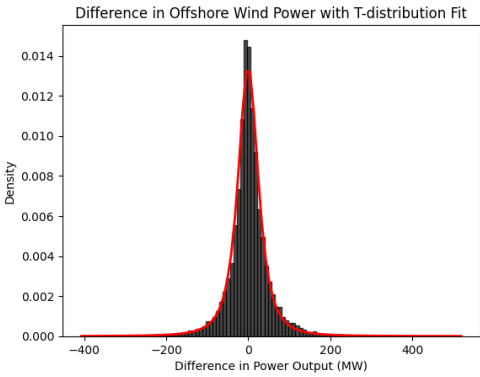


Figure A.3. Fitted t-distribution to pairwise difference

A.3 Results

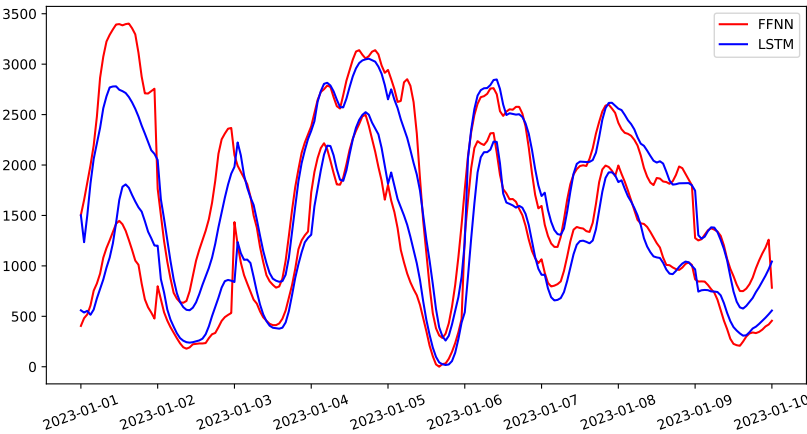


Figure A.4. Gap comparison between the FFNN and LSTM network for the 10th and 90th percentile.

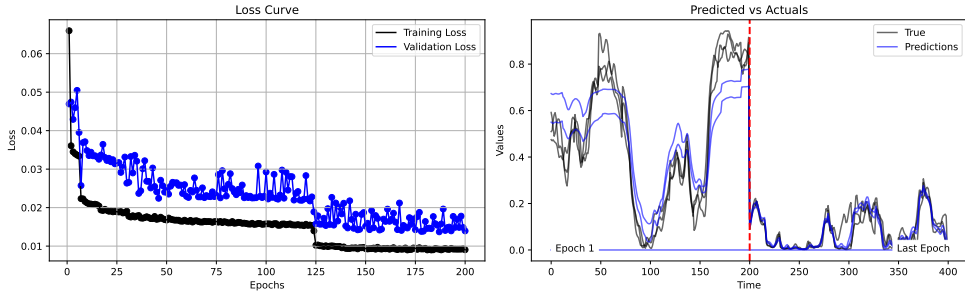
Region	Type	Model	CRPS	Variogram Score	QSS
DK1	OnshoreWindPower	LSTM	770.420000	14296.580000	3.200000
DK1	OnshoreWindPower	FFNN	145.320000	101093.600000	9.600000
DK2	OnshoreWindPower	LSTM	159.180000	8908.350000	1.450000
DK2	OnshoreWindPower	FFNN	100.800000	34256.860000	1.970000
DK1	OffshoreWindPower	LSTM	268.180000	22254.130000	2.190000
DK1	OffshoreWindPower	FFNN	97.480000	87058.680000	5.140000
DK2	OffshoreWindPower	LSTM	286.060000	15619.700000	2.860000
DK2	OffshoreWindPower	FFNN	162.350000	98338.520000	3.650000
DK1	SolarPower	LSTM	202.860000	22543.970000	3.420000
DK1	SolarPower	FFNN	235.470000	196077.050000	8.330000
DK2	SolarPower	LSTM	36.740000	14948.000000	0.890000
DK2	SolarPower	FFNN	38.240000	154533.120000	3.600000

Table A.3. All the actuals values rounded to two decimals places for Figure 4.6.

Region	Type	Model	CRPS	Variogram Score	QSS
DK1	OnshoreWindPower	LSTM	770.420000	14296.580000	3.200000
DK1	OnshoreWindPower	FFNN	145.320000	101093.600000	9.600000
DK2	OnshoreWindPower	LSTM	159.180000	8908.350000	1.450000
DK2	OnshoreWindPower	FFNN	100.800000	34256.860000	1.970000
DK1	OffshoreWindPower	LSTM	268.180000	22254.130000	2.190000
DK1	OffshoreWindPower	FFNN	97.480000	87058.680000	5.140000
DK2	OffshoreWindPower	LSTM	286.060000	15619.700000	2.860000
DK2	OffshoreWindPower	FFNN	162.350000	98338.520000	3.650000
DK1	SolarPower	LSTM	202.860000	22543.970000	3.420000
DK1	SolarPower	FFNN	235.470000	196077.050000	8.330000
DK2	SolarPower	LSTM	36.740000	14948.000000	0.890000
DK2	SolarPower	FFNN	38.240000	154533.120000	3.600000

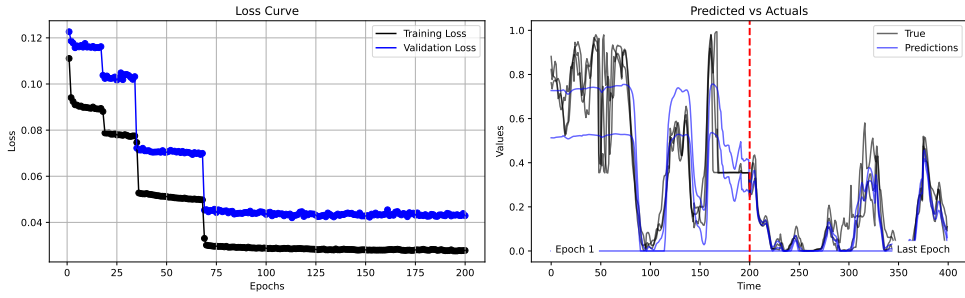
Table A.4. All the actuals values rounded to two decimals places for Figure 4.6.

Model Training Analysis



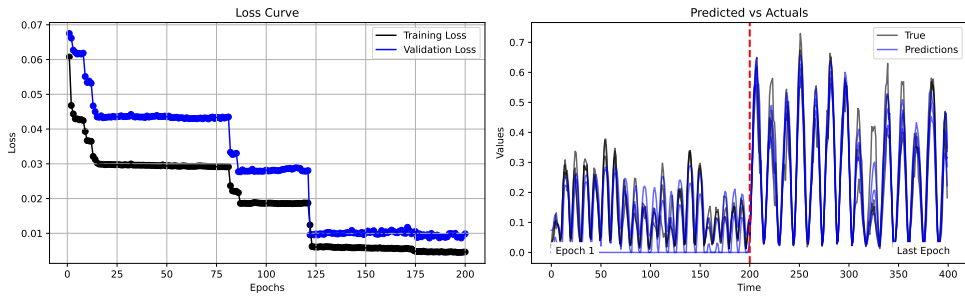
((a)) Onshore Wind Power

Model Training Analysis



((b)) Offshore Wind Power

Model Training Analysis



((c)) Solar Power

Figure A.5. Model Training Analysis for the three different power sources for DK2.

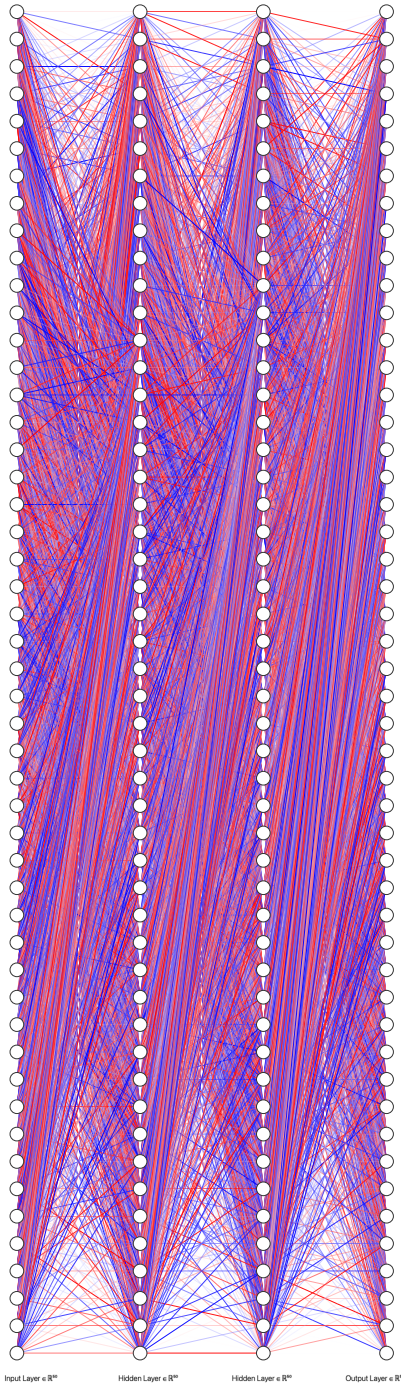


Figure A.6. A full overview of Model 3 from the FFNN Model selection process.

A.3.1 Additional Statistics for Corrected Quantiles

Model number for Table A.5 and A.6:

- 1: DK2 SolarPower
- 2: DK1 OnshoreWindPower LSTM
- 3: DK1 OffshoreWindPower LSTM
- 4: DK1 OnshoreWindPower FFNN
- 5: DK1 SolarPower FFNN
- 6: DK1 SolarPower LSTM
- 7 : DK2 OffshoreWindPower LSTM
- 8: DK2 OffshoreWindPower FFNN
- 9: DK2 SolarPower FFNN
- 10: DK2 SolarPower LSTM

Across the board we see improvements! We will not comment more on this, since our focus is on the CRPS, Variogram, and QSS score.

Model	Reliability (10, 25, 50, 75, 90)%	$IQR < 0$ (%)	Num. of Obs.	$E(IQR IQR < 0)$	min(IQR)	$E(IQR IQR > 0)$
1	0.19 , 0.28 , 0.39 , 0.50 , 0.61	0.65	307	-0.32	-0.48	47.68
2	0.17 , 0.30 , 0.49 , 0.74 , 0.83	0.00	921	nan	28.51	443.05
3	0.14 , 0.27 , 0.49 , 0.69 , 0.83	0.54	921	-10.19	-11.42	219.32
4	0.19 , 0.31 , 0.48 , 0.71 , 0.86	1.26	398	-18.77	-42.92	299.17
5	0.06 , 0.20 , 0.37 , 0.59 , 0.77	3.66	328	-7.6	-30.97	111.95
6	0.06 , 0.17 , 0.33 , 0.57 , 0.79	7.32	328	-24.35	-62.76	109.66
7	0.16 , 0.25 , 0.42 , 0.67 , 0.80	4.34	921	-39.46	-135.5	291.46
8	0.24 , 0.43 , 0.58 , 0.70 , 0.86	6.28	398	-57.07	-393.34	238.47
9	0.19 , 0.30 , 0.43 , 0.52 , 0.59	0.65	307	-1.95	-3.17	46.89
10	0.19 , 0.28 , 0.39 , 0.50 , 0.61	0.65	307	-0.32	-0.48	47.68

Table A.5. Comparison of Results for TAQR-SDE-CE prediction vs. actuals.

Model	Reliability (10, 25, 50, 75, 90)%	$IQR < 0$ (%)	Num. of Obs.	$E(IQR IQR < 0)$	min(IQR)	$E(IQR IQR > 0)$
1	0.33, 0.43, 0.51, 0.58, 0.66	2.28	1798	-21.26	-167.15	68.99
2	0.22, 0.35, 0.50, 0.66, 0.76	5.01	1798	-106.97	-825.08	401.8
3	0.17, 0.27, 0.43, 0.66, 0.81	3.62	1798	-73.09	-376.63	282.56
4	0.33, 0.44, 0.51, 0.57, 0.65	3.45	1798	-34.71	-152.0	146.73
5	0.15, 0.26, 0.39, 0.53, 0.68	5.73	1798	-15.18	-100.79	56.62
6	0.18, 0.26, 0.36, 0.55, 0.74	7.45	1798	-49.07	-220.08	251.73
7	0.33, 0.43, 0.51, 0.58, 0.66	2.28	1798	-21.26	-167.15	68.99
8	0.22, 0.35, 0.50, 0.66, 0.76	5.01	1798	-106.97	-825.08	401.8
9	0.17, 0.27, 0.43, 0.66, 0.81	3.62	1798	-73.09	-376.63	282.56
10	0.33, 0.44, 0.51, 0.57, 0.65	3.45	1798	-34.71	-152.0	146.73

Table A.6. Comparison of Results for TAQR-OE prediction vs. actuals.

A.3.2 Probe (b) LSTM

	Corrected	Original
CRPS	548.86	183.72
Variogram	0.11	0.21
Mean QSS	1.28	0.34

Table A.7. TAQR Prediction Scores for DK1 OnshoreWindPower with LSTM Corrected vs Original Ensembles. We observe poor performance by the LSTM network here.

	Corrected	Original
CRPS	242.18	108.72
Variogram	0.17	0.16
Mean QSS	0.22	0.22

Table A.8. TAQR Prediction Scores for DK1 OffshoreWindPower with LSTM Corrected vs Original Ensembles. We observe poor performance by the LSTM network here.

A.3.3 Further Statistics on TAQR Made from SDE-CE

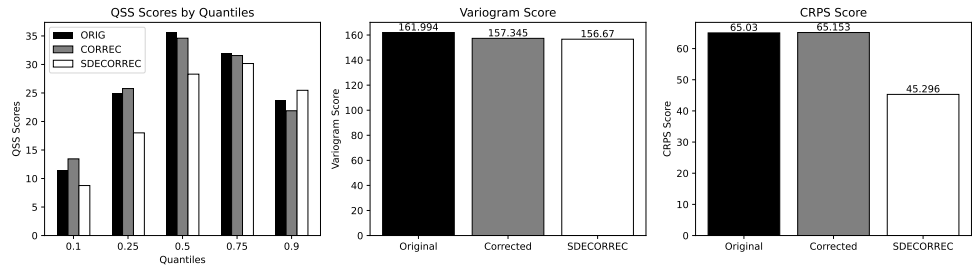


Figure A.7. CRPS, Variogram and QSS score for selected quantiles for the TAQR prediction based on the SDE-CE for DK2 Solar Power from the FFNN model.

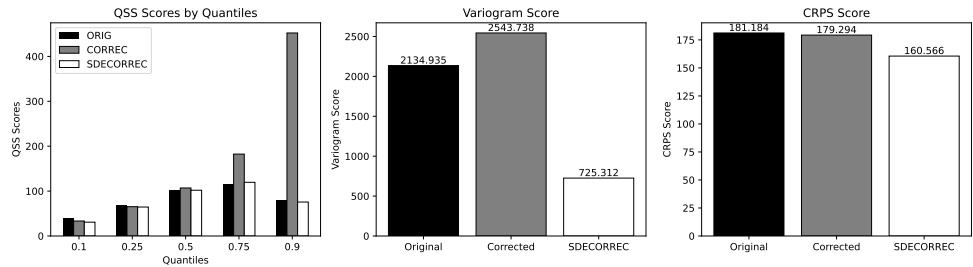


Figure A.8. CRPS, Variogram and QSS score for selected quantiles for the TAQR prediction based on the SDE-CE for DK2 Offshore Wind Power from the FFNN model.

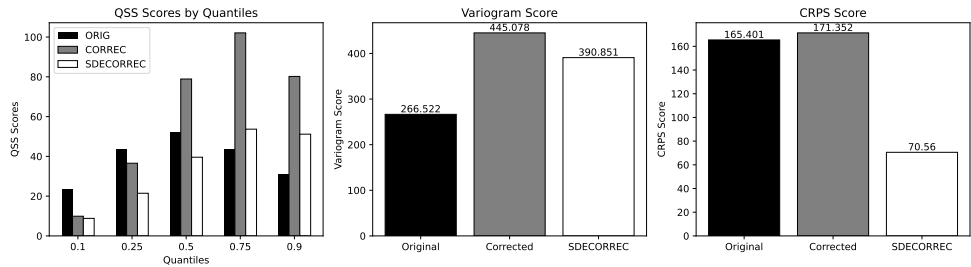


Figure A.9. CRPS, Variogram and QSS score for selected quantiles for the TAQR prediction based on the SDE-CE for DK1 Solar Power from the FFNN model.

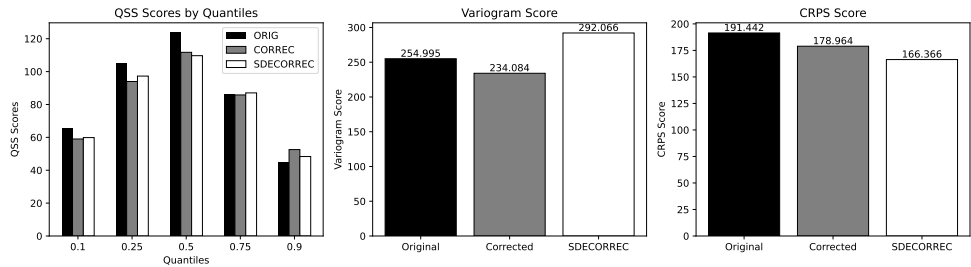


Figure A.10. CRPS, Variogram and QSS score for selected quantiles for the TAQR prediction based on the SDE-CE for DK2 Onshore Wind Power from the FFNN model.

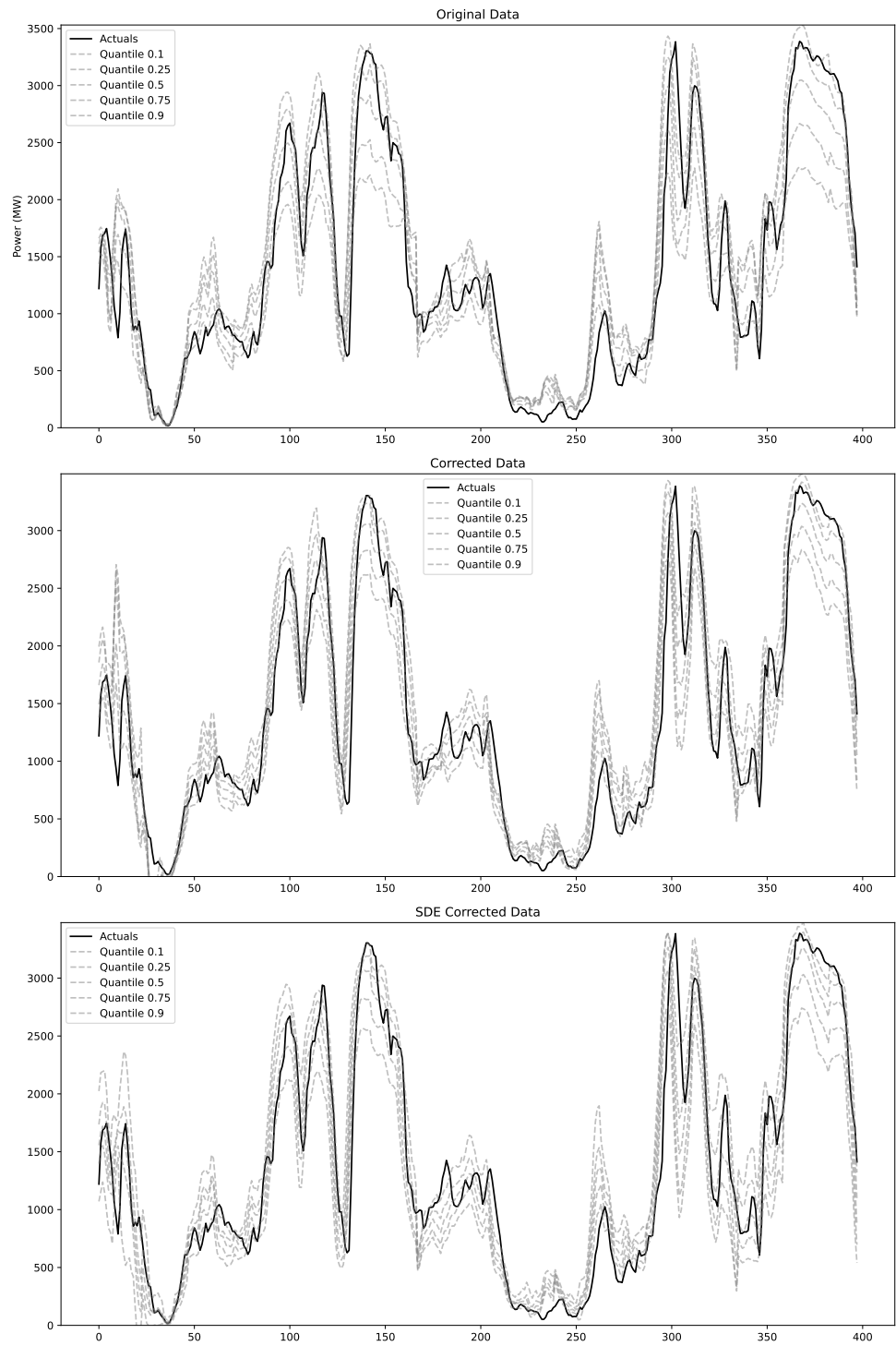


Figure A.11. Example of how the TAQR predictions look based ensembles, corrected ensembles, SDE-CE respectively.

A.4 Discussion

A.4.1 Casestudy 1

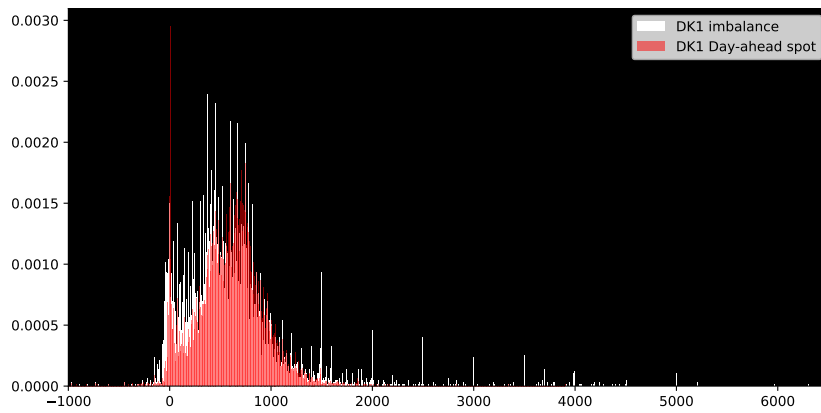


Figure A.12. Density histogram for the imbalance and day ahead prices in DK1.

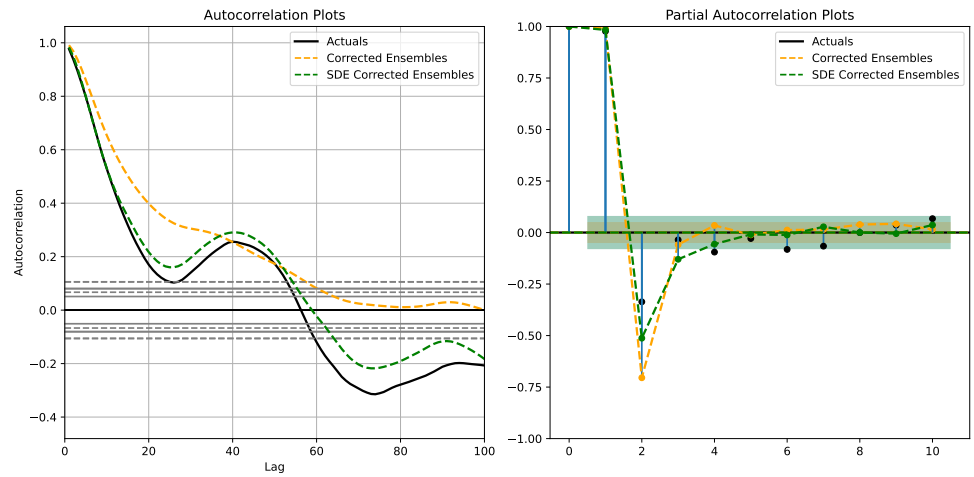


Figure A.13

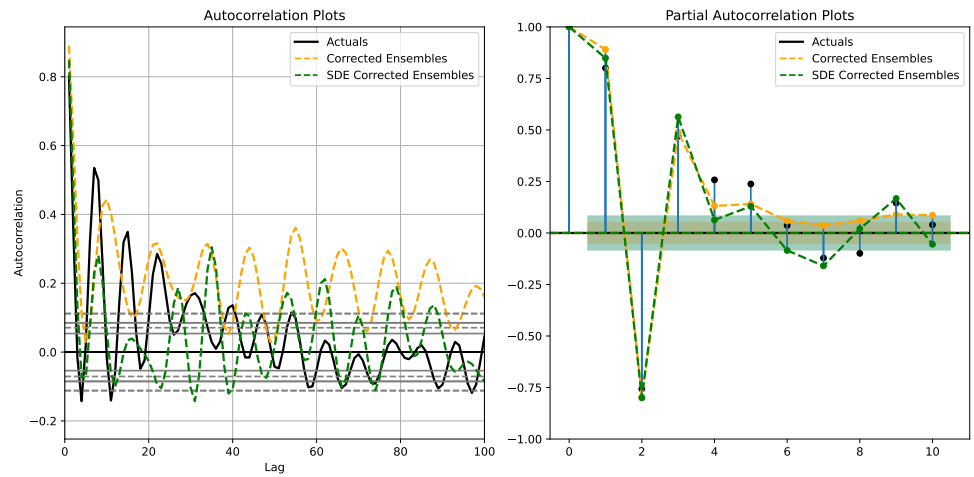


Figure A.14

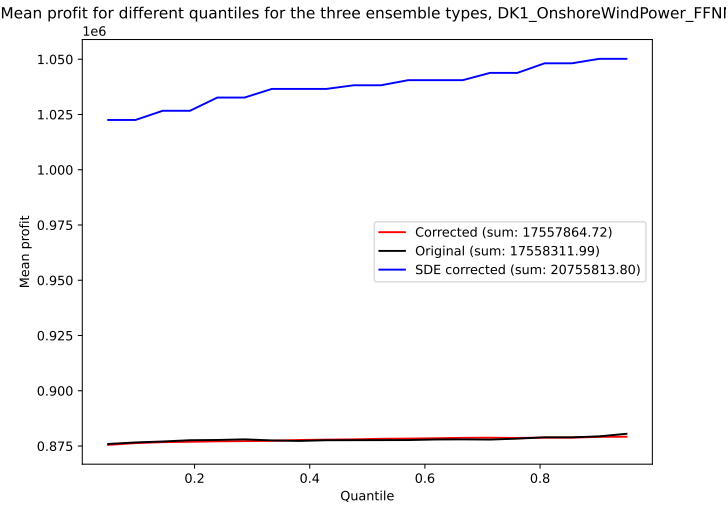


Figure A.15

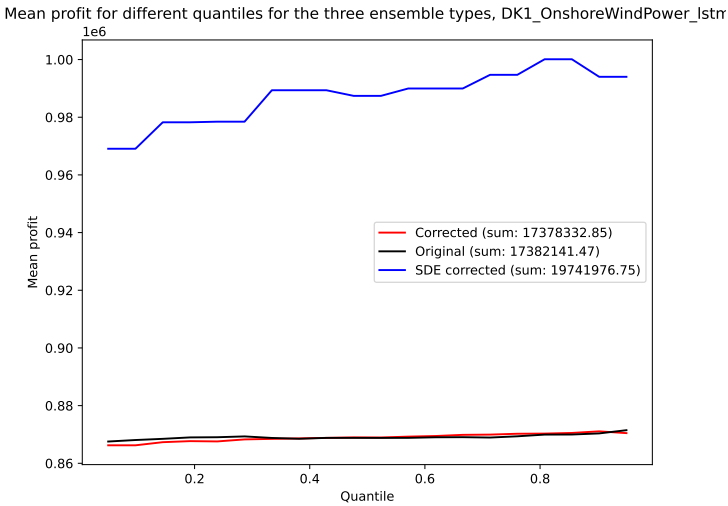


Figure A.16

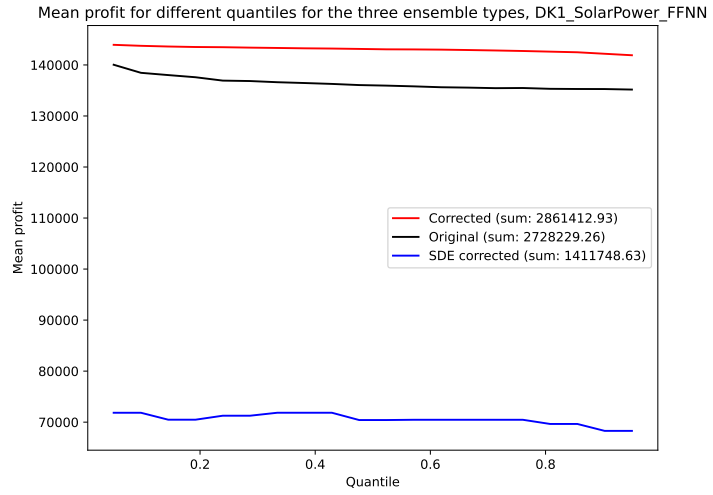


Figure A.17

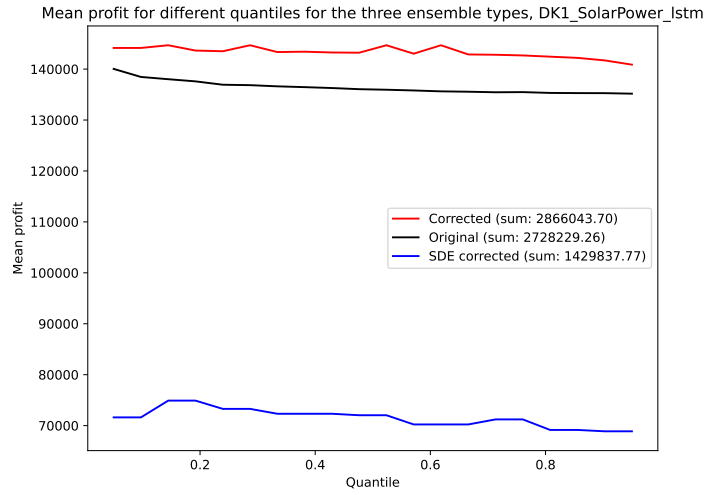


Figure A.18

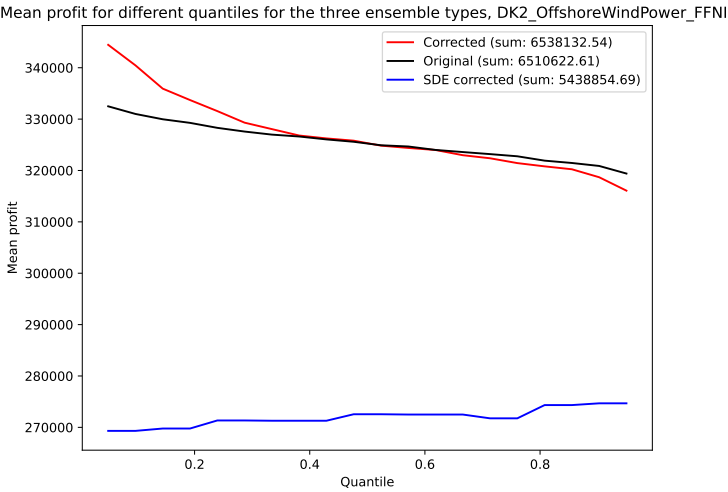


Figure A.19

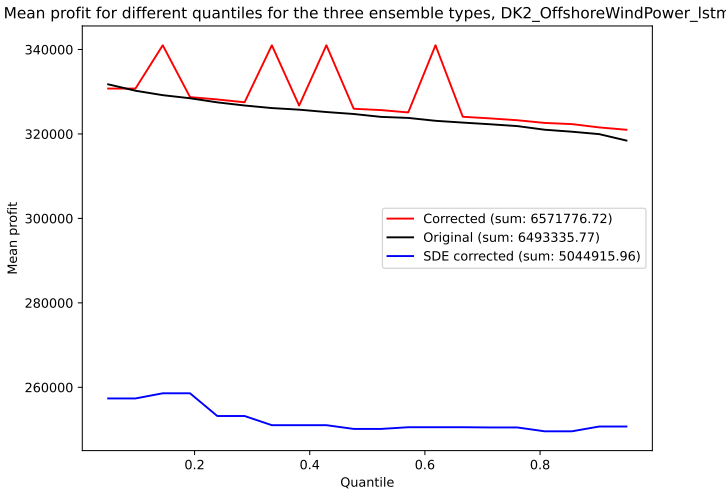


Figure A.20

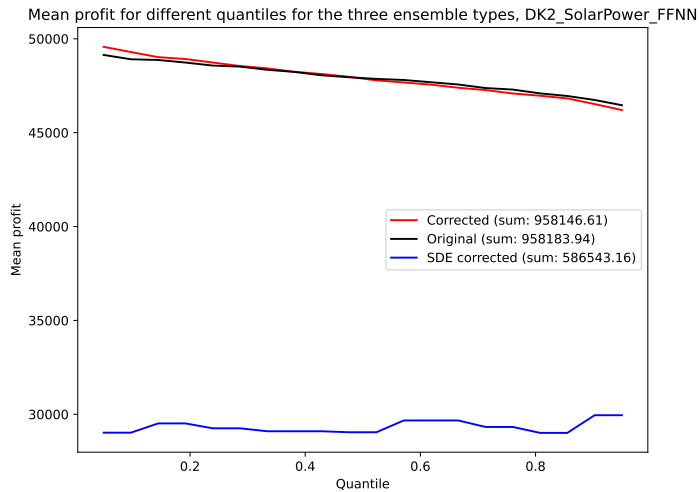


Figure A.21

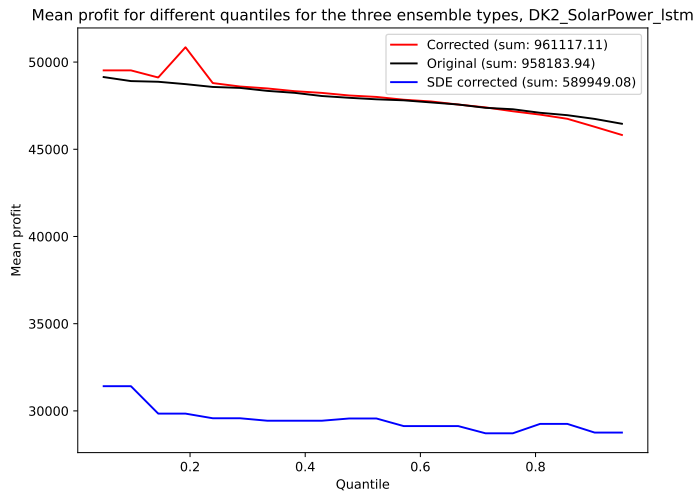


Figure A.22