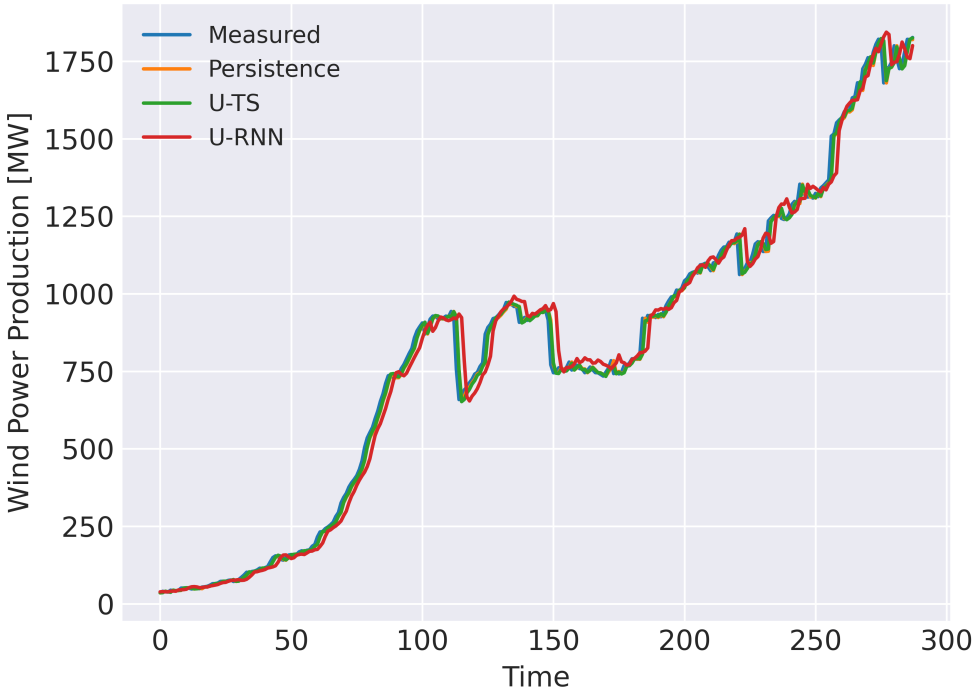
Forecasting Wind Power Production

Time Series Forecasting

Semester Project

MATTEK 9 - Group 6



Aalborg University
Mathematical Engineering

This project has been written in LaTeX with figures produced in Ti*k*Z and Matplotlib unless otherwise stated. Scripts have been made using Python 3.9. In case of stains, please note the washing guidance below.

AALBORG UNIVERSITY
STUDENT REPORT

Mathematical Engineering
Aalborg University
http://www.aau.dk

Title:
Forecasting Wind Power Production

Theme:
Time Series Forecasting

Project Period:
Fall Semester 2021

Project Group:
MAT-TEK 9 Group 6

Participants:
Andreas Anton Andersen
Martin Voigt Vejling
Morten Stig Kaaber

Supervisors:
Christophe Biscio
Petar Popovski
Tobias Kallehauge

Copies: 1

Number of Pages: 96

Date of Completion:
March 17, 2022

Abstract:

In this project, we implement different models for very short-term forecasting of the Danish wind power production. Furthermore, we check the generalisation abilities to longer forecasts of these models. In order to implement models for forecasting the wind power production the relevant theory of time series, recurrent neural networks (RNNs), and the empirical mode decomposition is presented. Using this theory, several methods for forecasting the wind power production are implemented. We implement both autoregressive moving average (ARMA) models, vector ARMA models, RNNs where each sub-grid in Denmark is modelled either separately or jointly, and a hybrid model which uses the empirical mode decomposition, RNNs, and ARMA models in conjunction. Through the experiments, we find that for very short-term predictions the ARMA model performs the best on average in terms of mean square error and normalised mean absolute error. Lastly, we check the models' ability to generalise to short-term and medium-term forecasts and we find that the RNN modelling the sub-grids separately generalises the best.

Preface

This project has been written in cooperation with Energinet which is the transmission system operator in Denmark.

The project has been made by a group of 3rd semester master students studying Mathematical Engineering at Aalborg University. The group thanks the supervisors Christophe Biscio, Tobias Kallehauge, and Petar Popovski for their supervision throughout the project period. Additionally, the group thanks Energinet for a good cooperation during the project period and would especially like to thank the contact person Lasse Diness Borup. Furthermore, the group thanks CLAAUDIA for making the AI-Cloud service available for GPU computing.

The references throughout the project have been handled with the alphabetical IEEE-method with specified page numbers of the respective source. Further information about the sources may be found in the bibliography.

The figures have been made using the TikZ package in LaTeX, and using the `matplotlib.pyplot` package in Python by the authors of the project unless otherwise stated. The scripts which have been used to obtain results in the project can be found in `https://github.com/DeTreMusketerer/Wind_Power_Forecasting-P9`.

Aalborg University, March 17, 2022

Andreas Anton Andersen
<aand17@student.aau.dk>

Martin Voigt Vejling
<mvejli17@student.aau.dk>

Morten Stig Kaaber
<mkaabe17@student.aau.dk>

Abbreviation List

ACF	Autocorrelation function
ARIMA	Autoregressive integrated moving average
ARIMAX	Autoregressive integrated moving average with exogenous variables
ARMA	Autoregressive moving average
ARMAX	Autoregressive moving average with exogenous variables
ARX	Autoregressive with exogenous variables
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BR	Balance Responsible
CSI	Cubic spline interpolation
EMD	Empirical mode decomposition
GRU	Gated recurrent unit
iid.	Independent and identically distributed
IMF	Intrinsic mode function
LBR	Load Balance Responsible
LMMSEE	Linear minimum mean square error estimator
LSTM	Long short-term memory
MMSEE	Minimum mean square error estimator
MSE	Mean square error

NEMO	Nominated Electricity Market Operator
NMAE	Normalised mean absolute error
NOIS	Nordic Operational Information System
NWP	Numerical weather prediction
PACF	Partial autocorrelation function
PBR	Production Balance Responsible
RNN	Recurrent neural network
s-ARIMAX	Seasonal autoregressive integrated moving average with exogenous variables
s-VARIMAX	Seasonal vector autoregressive integrated moving average with exogenous variables
s-VARMA	Seasonal vector autoregressive moving average
s-VARMAX	Seasonal vector autoregressive moving average with exogenous variables
SampEn	Sample entropy
TSO	Transmission System Operator
VAR	Vector autoregressive
VARIMA	Vector autoregressive integrated moving average
VARIMAX	Vector autoregressive integrated moving average with exogenous variables
VARMA	Vector autoregressive moving average
VARMAX	Vector autoregressive moving average with exogenous variables
VARX	Vector autoregressive with exogenous variables
VMA	Vector moving average
WSS	Weak sense stationary

Notation List

Miscellaneous	
$\mathbf{0}$	Zero matrix
$\boldsymbol{I}$	Identity matrix
C^n	n times continuously differentiable
j	Imaginary unit
$\lvert\boldsymbol{A}\rvert$	Determinant of a matrix $\boldsymbol{A}$
$\operatorname{sgn}(\cdot)$	Sign function
$\mathbf{1}$	Vector of ones
$L(\boldsymbol{\theta})$	Likelihood function
$\ell(\boldsymbol{\theta})$	Log-likelihood function
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
Time Series Analysis	
$\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$	Time series or stochastic process
$\boldsymbol{\mu}_{\boldsymbol{y}}$	Mean of $\boldsymbol{y}$
$\boldsymbol{\Gamma}(h)$	Cross-covariance matrix
$\boldsymbol{R}(h)$	Cross-correlation matrix
$\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$	Centred time series
$\{\boldsymbol{u}_t\}_{t\in\mathbb{Z}}$	Vector white noise process

$\boldsymbol{\Phi}_j$	VAR-parameters for time lag j
$\boldsymbol{\Psi}_j$	VMA-parameters for time lag j
B	Backward lag operator
$\boldsymbol{\Xi}$	Exogenous variable parameters
$\{\boldsymbol{z}_t\}_{t\in\mathbb{Z}}$	Exogenous variables
$\boldsymbol{\Theta}$	Matrix of parameters
$\bar{\boldsymbol{y}}_{t+\tau}^n$	Truncated τ-ahead prediction
$\bar{\boldsymbol{u}}_t^n$	Truncated prediction errors
∇^d	Differencing operator of order d
$\boldsymbol{\Sigma}_{\boldsymbol{u}}$	Covariance matrix of $\boldsymbol{u}_t$
$\hat{u}_t$	Estimate of u_t

State Space

$\boldsymbol{F}$	Transition matrix
$\boldsymbol{H}$	External variable parameter matrix
$\boldsymbol{x}_t$	State variables
$\boldsymbol{y}_t$	Observed variables
$\boldsymbol{A}$	Observation matrix
$\boldsymbol{K}_t$	Kalman Gain
$\boldsymbol{\varepsilon}_t$	Innovations
$\boldsymbol{\Sigma}_t$	Innovation covariance
$\boldsymbol{x}_t^{t-1}$	State forecast
$\boldsymbol{P}_t^{t-1}$	Conditional error covariance of the state forecast

Time Series Decomposition

$z_x(t)$	A complex valued function
$Z_x(\omega)$	The Fourier transform of $z_x(t)$
$\theta(t)$	Instantaneous phase
$\omega(t)$	Instantaneous frequency
p.v.	Cauchy principal value
$x(t)$	A real valued function
$\hat{x}(t)$	Hilbert transform of x
$\mathcal{H}(x)$	Hilbert transform of x
$\mathcal{F}(x)$	Fourier transform of x
L^p	Lebesgue space of order p
$\mathcal{F}^{-1}(X)$	Inverse Fourier transform of X
$\mathcal{H}^{-1}(\hat{x})$	Inverse Hilbert transform of $\hat{x}$
n_e	Number of extremas
z_c	Number of zero-crossings
e_t^u	Upper envelope of x_t
e_t^l	Lower envelope of x_t
m_t	Mean of the upper and lower envelope
P_i	A polynomial
$S(t)$	A spline
$h_t^{(n,k)}$	kth component of the sifting procedure after finding $n-1$ IMFs

$c_t^{(k)}$	kth IMF
$r_t^{(k)}$	Residual after finding k IMFs
α_t	Mode amplitude
ι_t	Evaluation sequence
D	Deviation
y_t^m	Block of length m
$d(y_i^m, y_j^m)$	Chebyshev distance between y_i^m and y_j^m
$B_i^m(r)$	Empirical probability that two sequences of length m are similar

Recurrent Neural Networks

$\tanh(\cdot)$	Hyperbolic tangent activation function
$\boldsymbol{h}_t$	Hidden layer value at time t
$\hat{\boldsymbol{y}}_t$	Output at time t
$\boldsymbol{b}_h$	Bias of $\boldsymbol{h}_t$
$\boldsymbol{W}_{hh}$	Weight matrix between $\boldsymbol{h}_{t-1}$ and $\boldsymbol{h}_t$
$\sigma(\cdot)$	Sigmoid activation function
$\boldsymbol{f}_t$	Forget gate
$\boldsymbol{i}_t$	Input gate
$\boldsymbol{o}_t$	Output gate
$\boldsymbol{s}_t$	LSTM/GRU state
$\boldsymbol{u}_t$	Update gate
$\boldsymbol{r}_t$	Reset gate

Experiments

$\{P_{l,t}\}_{t=1}^{n}$	Power data for sub-grid l
$T_{l,t+\tau}^{t}$	Temperature forecast at time t to time $t+\tau$ for sub-grid l
$v_{l,t+\tau}^{1,t}$	Wind speed forecast at 10 meters forecast at time t to time $t+\tau$ for sub-grid l
$v_{l,t+\tau}^{2,t}$	Wind speed forecast at 100 meters forecast at time t to time $t+\tau$ for sub-grid l
$w_{l,t+\tau}^{1,t}$	Wind direction forecast at 10 meters forecast at time t to time $t+\tau$ for sub-grid l
$w_{l,t+\tau}^{2,t}$	Wind direction forecast at 100 meters forecast at time t to time $t+\tau$ for sub-grid l
$\{\rho_{1,t}\}_{t=1}^{n}$	Down regulation for DK1
$\{\rho_{2,t}\}_{t=1}^{n}$	Down regulation for DK2
$\hat{P}_{l,t+\tau}^{t}$	Wind power forecast in sub-grid l at time t for time $t+\tau$

Contents

1. Problem Analysis

In recent years, the need of an energy transition from fossil fuels to renewable energy sources such as wind power, solar power, hydropower, biomass power etc. has increased due to fossil fuels' connection to climate change, ozone depletion, and air pollution. Sustainable energy can be used as a means to reach the goal of the Paris agreement which is to limit the increment of the average global surface temperature by less than 2 °C. In order for this to happen, renewable energy needs to make up two-thirds of the global energy demand by 2050. [Han+20, p. 1][Gie+19, pp. 38 & 43]

A source of renewable energy which has gained popularity due to its efficiency, relatively low costs, and environmental benefits is wind power and according to [Gie+19, pp. 41 & 43], wind power will play a significant role in the energy transition and will account for 24% of all energy from renewable energy sources when considering power, transport, heat, and other direct uses in 2050 if the goal from the Paris agreement should be reached.

In Denmark, the vast majority of renewable energy comes from wind power which is displayed in Fig. 1.1, showing the power production of renewable energy within the last year compared to the energy consumption. In 2020, the Danish electricity production generated 28.1 TWh from which 48% came from wind power [sta21a][sta21b]. Since the majority of the produced power in Denmark is wind power, this project will restrict its attention to wind power.

A disadvantage of wind power is that it varies due to its dependence on the weather making it unreliable and this makes it challenging for large-scale integration of wind power. These challenges introduce the need of a standby generator in case the supply does not match the demand, potential curtailment of wind power in case the supply is higher than the demand, and difficulties for choosing the location of wind farms. [Zha+19, p. 229]

A way to reduce this disadvantage is to use forecasting of the wind power for the wind farms in the power grid. Depending on the time horizon of the forecast, it can provide advantages for different applications. For very short-term, i.e. forecasts ranging from only a few minutes to 30 minutes, the applications gaining advantages are regulation actions, real-time grid operations, market clearing, and turbine control. For short-term, i.e. forecasts ranging from 30 minutes to 6 hours, advantages can be gained in load dispatch planning, and for medium-term forecasts, i.e. forecasts ranging from 6 hours to 1 day, the forecasts can be used for operational security in

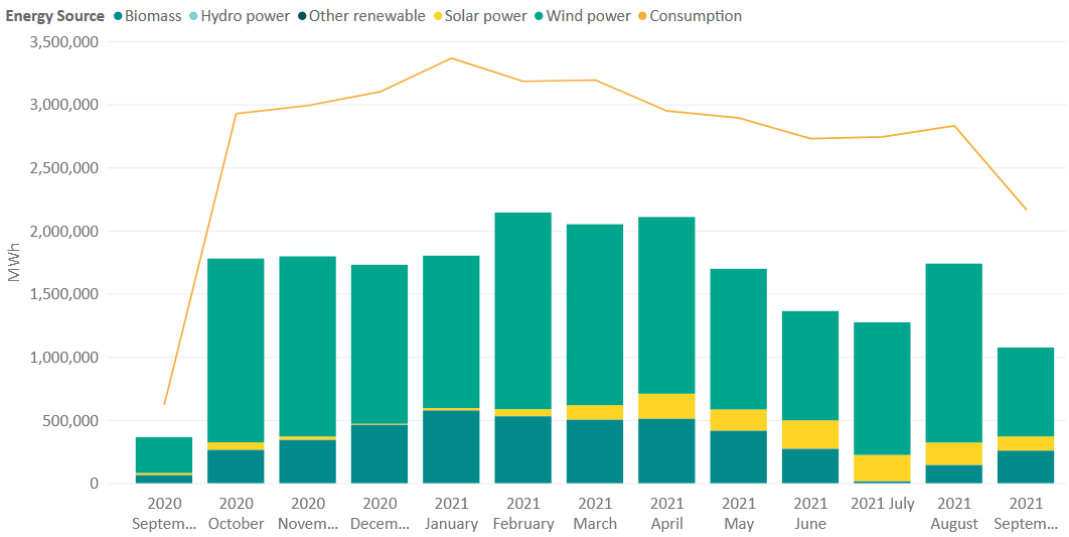


Figure 1.1: Danish power production from renewable energy sources in the period from the 13th of September 2020 to the 12th of September 2021. Taken from Energinet.

the electricity market, energy trading, and on-line and off-line generating decisions. Lastly, for long-term forecasts, i.e. forecasts more than 1 day in the future, advantages occur for reserve requirements, maintenance schedules, optimum operating cost, and operation management. [Han+20, p. 3]

In collaboration with the Danish company Energinet, this project aims to make a model for forecasting the wind power. The electricity market as well as Energinet's role in the electricity market is described in the following section where we also further elaborate on the time horizon of the forecasting needed by Energinet. Unless otherwise is stated, the following section is based on a presentation given by Energinet.

1.1 The Electricity Market

The Danish electricity market consists of different parties. The main parties are the Transmission System Operator (TSO), Nominated Electricity Market Operator (NEMO), and Balance Responsible (BR) players.

The TSO in Denmark is the company Energinet and the job of a TSO is to run the power grid and electricity market. Besides this, Energinet is also responsible for securing that the Danish population is supplied with electricity at all times. Hence, the main task of Energinet is to ensure that there is balance between the electricity consumption and generation in the electricity system. Furthermore, the power grid is owned by Energinet and Energinet is responsible for the electricity market being well-functioning with fair prices for both consumers and producers as well as promoting climate friendly energy solutions.

The NEMOs are power exchanges which fix the prices of electricity based on

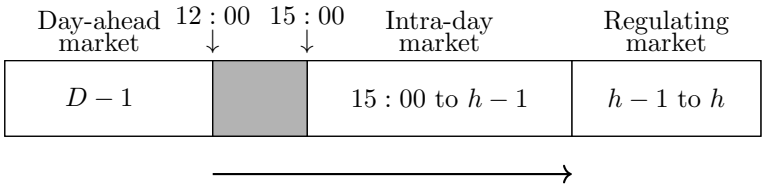


Figure 1.2: The timeline of the different phases in the electricity market until the delivery hour h and where D denotes the day of delivery. It should be noted that the regulating market is open during other hours than shown in the figure. This figure is inspired by a presentation given by Energinet.

supply and demand by the hour. In the market model for the electricity market, the electricity is generated from the production unit which is the cheapest such that the bid of the buyer is matched while congestion in the power grid is taken into account. The power exchange in Denmark is Nord Pool Spot which is owned by the TSOs in Denmark, Finland, Norway, Sweden, Estonia, Latvia, and Lithuania.

The BR players operate at Nord Pool Spot where they buy and sell electricity on behalf of electricity suppliers and plant owners for which they are balance responsible. The BR players can be divided into two groups, production BR (PBR) and load BR (LBR). A PBR can consist of multiple companies with multiple electricity plants and is responsible for the balance between the expected production of electricity and the actual production. On the other hand, an LBR is mostly concerned with electricity trading companies and is responsible for the balance between the expected electricity consumption and the actual electricity consumption of the customers of the companies for which they are responsible. [Ene; Ene19]

In the following, we will describe the timeline of the electricity market as well as further elaborate on the relation between the different parties in the market. The timeline of the electricity market is seen in Fig. 1.2.

In Nord Pool Spot, there are two main types of markets: The day-ahead market and the intra-day market. On the day before the delivery hour, the electricity is traded in the day-ahead market. Here, the BR players either sell or buy electricity for the following day. The deadline for bids is at 12:00 whereafter the prices are determined hourly in order to match the supply and demand. The BR players then report the expected electricity production and expected electricity consumption, respectively as well as their trading plans for the following 24 hours to Energinet. However, these plans can be altered until 45 minutes before the delivery hour on the intra-day market or bilateral trades between two parties.

The intra-day market opens at 15:00 the day before delivery hour and closes an hour before the delivery hour. Here the BR players trade such that they obtain balance. A BR might have an imbalance due to their power production not being as high as anticipated and therefore have sold more power than they can produce.

After the intra-day market closes, the TSO is the only one who can regulate the

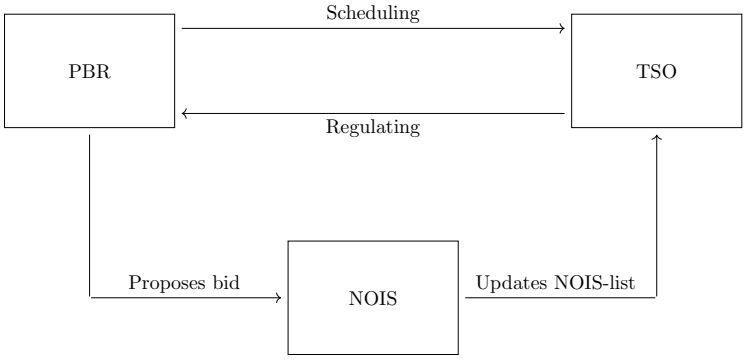


Figure 1.3: Block diagram depicting the relation between TSO, PBR, and NOIS. This figure is inspired by a presentation given by Energinet.

BR players which is done through the regulating market. The TSO can regulate the BR players at all times, however this is mainly done in the hour before delivery called the operational hour. [Ene19]

The Nordic regulating market is called Nordic Operational Information System (NOIS) and here the TSO regulates the BR players such that the supply matches the demand on delivery hour. The TSO can request that a BR player either decreases or increases its power production. If this is needed, the TSO requests that the BR players submit bids for decrement or increment of their power production. These requests can be made with only a 15 minutes notice and the bids are added to a list called the NOIS-list. This list can be updated until 45 minutes before delivery hour. The TSO will then choose the cheapest regulation on the list which ensures balance. The relation between the TSO, BR players, and NOIS is depicted in Fig. 1.3. [Ene20; Ene19]

In this project, we will after correspondence with Energinet focus on wind power forecasting during the operational hour. Therefore, we will focus on very short-term forecasting such a forecast is useful for Energinet in order to be able to obtain balance in the electricity market during the operational hour.

1.2 Problem

When determining the produced wind power of a single wind turbine, the theoretical relationship between wind speed v and wind power P can be used which is

$$P = \frac{1}{2}\rho_{\text{air}} A C v^3 \tag{1.1}$$

where ρ_{air} is the air density, A is the swept area, and C is a constant describing the physical properties of the wind turbine. The constant C is bounded by the

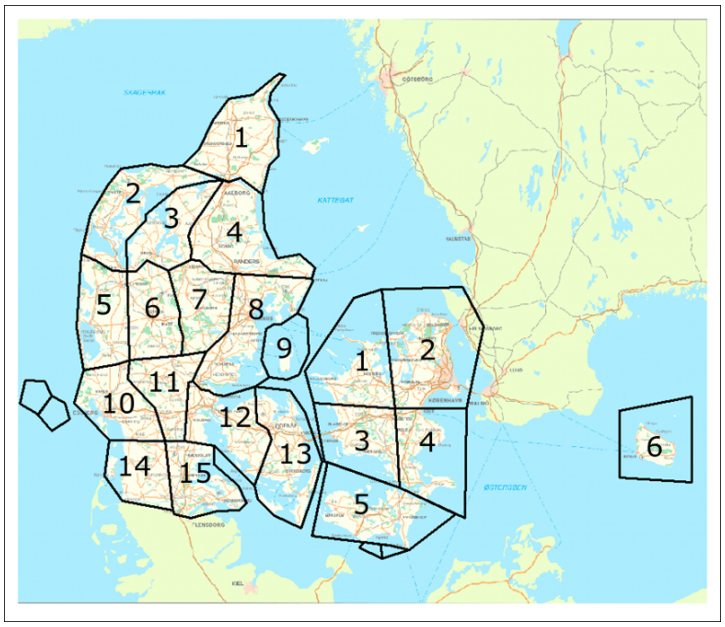


Figure 1.4: The sub-grids in DK1 and DK2. Taken from a presentation given by Energinet.

Betz limit $C \leq \frac{16}{27}$. However, (1.1) has its limitation for low and high wind speeds where the cubic relation between wind speed and wind power does not hold. [ZCA16, p. 12][Sin+21][Lyd+14, p. 453][Jai16, pp.11-23]

In practice, there are additional factors to consider for wind power forecasting, which allows for more accurate predictions than that of the theoretical power curve. Moreover, forecasting the wind power production of entire wind farms, rather than single wind turbines, further complicates the problem. [Han+20, p. 16][Som+10, p. 3]

The wind power from a wind turbine depends on weather dependent variables such as wind speed v, wind direction ω, temperature T, and air density ρ_{air} and on quantities such as the hub height and the surroundings of the wind turbine. [Han+20, pp. 2 & 16]

Before further describing the model, we will discuss the resolution of sub-grids available by Energinet. Energinet does not have access to information regarding the wind power production of every wind farm placed in the Danish power grid. Instead, they are divided into sub-grids where a single sub-grid consists of multiple wind mills. The Danish power grid is divided into two separate grids called DK1 and DK2. The different sub-grids in DK1 and DK2 are seen in Fig. 1.4 where the grid on the left side is DK1 and the grid on the right side is DK2. There are 15 sub-grids in DK1 and 6 sub-grids in DK2, i.e. a total of 21 sub-grids.

As mentioned, the weather influences the wind power and hence, when forecasting wind power, weather forecasts can be used. The weather forecasts which are used by Energinet has a resolution of 15×15 km and this weather forecasting grid is seen in

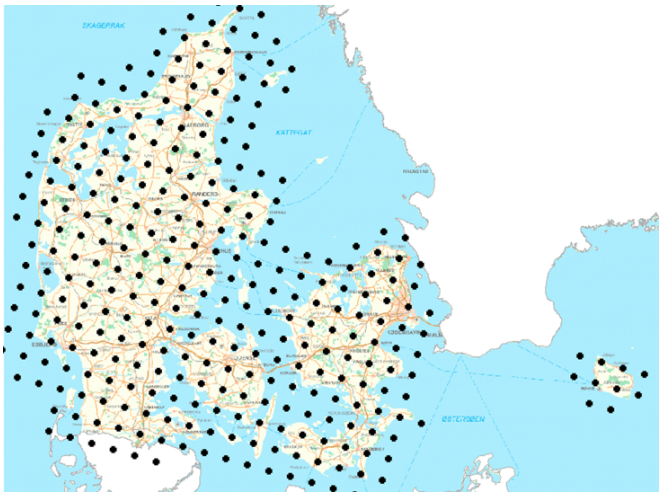


Figure 1.5: The 15×15 km weather forecasting grid in Denmark. Taken from a presentation given by Energinet.

Fig. 1.5.

If we consider the 21 sub-grids of DK1 and DK2 and let $l = 1, \ldots, 21$ denote a sub-grid, then we want to make a model which forecasts the wind power of each sub-grid at a given time t. We will now introduce the following notation for the lth sub-grid: Let $\{P_{l,t}\}_{t=1}^{n}$ be a time series of the wind power and let $\{\boldsymbol{z}_{l,t}\}_{t=1}^{n}$ be a multivariate time series containing exogenous variables herein weather forecasts. For the lth sub-grid, we can forecast the wind power at time $t+1$, $\hat{P}_{l,t+1}$, by some model

$$\hat{P}_{l,t+1} = f(P_{l,t}, P_{l,t-1} \ldots, P_{l,t+1-p}, \boldsymbol{z}_{l,t+1}; \boldsymbol{\theta}) \tag{1.2}$$

where $\boldsymbol{\theta}$ is the parameters of the model and p is a time lag describing how much prior information is used in the model. This process can then be used iteratively to make a forecast for time $t + \tau$

$$\hat{P}_{l,t+\tau} = f(\hat{P}_{l,t+\tau-1} \ldots, \hat{P}_{l,t+\tau-p}, \boldsymbol{z}_{l,t+1}; \boldsymbol{\theta}) \tag{1.3}$$

where $\hat{P}_{l,i} = P_{l,i}$ for $i \leq t$.

Different approaches have previously been used for wind power forecasting and in the following section, the main approaches will be introduced.

1.3 Methods

There are several methods for wind power forecasting and these methods can generally be categorised into four categories: physical methods, hybrid methods, statistical methods, and machine learning methods herein neural networks.

The physical methods use the physical characterisations of the wind turbines and the area in order to make the forecast. An example of a physical method is [FLW01]

where they make a 48 hours ahead prediction of the wind power for different regions in Germany. This is done based on an input consisting of an estimate of the wind speed at hub height and data describing the surrounding terrain.

The statistical methods are often based on time series which take historical numerical weather predictions (NWPs) and historical power production and develops a mathematical model to predict the wind power production. In [PS+09], an autoregressive integrated moving average (ARIMA) model is implemented. In [Liu+10], the historical data is decomposed using wavelets and then an improved time series model is implemented which consists of an ARIMA model which iteratively re-estimates the parameters. In [ZCA16], both wind speed and wind power are modelled concurrently by a vector autoregressive moving average (VARMA) model including the use of time varying coefficients, thresholding for modelling non-linearities, and assuming correlation in the noise. In [Nie+07], an autoregressive model using the wind speed as an exogenous variable designed for short-term forecasts. In [CD14], a generalisation of the method in [Nie+07] is implemented for medium-term forecasts including censoring and with both wind speed and wind direction as exogenous variables.

The neural network methods, as the statistical methods, use historical data to forecast the wind power production with various neural network architectures. In [Zha+19], a long short-term memory (LSTM) model is implemented. In [Tou+21], three different LSTM architectures are implemented and a test determining how often the networks need to be recalibrated and how much historical data needs to be used for training is made. In [Xia+17], a spectral decomposition of the data is made before applying it to an implemented wavelet neural network. In [Pen+20], wavelets are used to decompose the data before feeding it to a gated recurrent unit (GRU). In [BT07], fuzzy logic and recurrent neural networks are used for forecasting. In [Wu+20], the abilities of convolutional neural networks and LSTMs are combined for forecasting. Finally, [Zha+20] implements a Wasserstein generative adversarial network for forecasting. Additionally, the NWPs are clustered and given as a conditional input.

An example of a Hybrid method can be seen in [LDB21] where the wind speed is decomposed using the empirical mode decomposition (EMD). Then based on the sample entropy each part of the decomposition is modelled by either a neural network model or an ARMA model.

1.4 Problem Statement

We will focus on VARMA models and neural network models as well as a hybrid model inspired by [LDB21]. Using these models, we will answer the following problem statement.

What accuracy can be achieved for very short-term forecasts of the Danish wind power production and how do the forecasts generalise to short-term and medium-term forecasting?

The project consists of two parts with the first part laying the theoretical foundation of time series, the empirical mode decomposition, and neural networks, while the second part contains the application and results with these methods.

2. Time Series Analysis

In time series analysis, we are dealing with data which has a temporal dependency. Based on this data, the purpose is to do statistical modelling and perform inference for the data. From a theoretical perspective, the observed time series is assumed to follow some probabilistic model and thereby is drawn as a realisation of a so-called stochastic process. In this chapter, the classical methods as well as modern extensions used for time series analysis will be introduced. The treatment will be concise and focused on the purpose of this project which is to forecast wind power production. When using time series for wind power forecasting, it might be of interest to not only consider a single time series but instead multiple time series. Therefore, this chapter will focus on multivariate time series.

Consider a multivariate time series which is a realisation of a stochastic process $\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$. Note that we will denote both the stochastic process as well as the time series by $\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$. The reader is assumed to be familiar with the concept of stochastic processes. However, for a brief recap of definitions relevant for the reading of this chapter is introduced in Appendix A.1.

2.1 Vector Autoregressive Moving Average Model

In this section, we will introduce a class of multivariate linear models, called vector autoregressive moving average (VARMA), which is useful when modelling weak sense stationary (WSS) stochastic processes. When implementing VARMA models, the multivariate time series $\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$ is often centred around zero such that the centred process $\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$ is given as $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{\mu}_{\boldsymbol{y}}$ where $\boldsymbol{\mu}_{\boldsymbol{y}}$ is the mean of $\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$. Before proceeding to the definition of VARMA models, a particular class of stochastic processes called vector white noise processes are defined.

Definition 2.1 (Vector White Noise Process)
Let $\{\boldsymbol{u}_t\}_{t\in\mathbb{Z}}$ be a stochastic process. If $\boldsymbol{\mu}_{\boldsymbol{u}}(t) = \mathbf{0}$ and

$$\boldsymbol{\Gamma}_{\boldsymbol{u}}(h) = \boldsymbol{\Sigma}_{\boldsymbol{u}}\mathbb{1}[h = 0] \tag{2.1}$$

where $\boldsymbol{\Gamma}(h)$ is the cross-covariance matrix at time lag h defined in Definition A.5 and $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ is the covariance matrix of $\boldsymbol{u}_t$, then $\{\boldsymbol{u}_t\}_{t\in\mathbb{Z}}$ is a vector white noise process. [Box+16, p. 507]

Definition 2.2 (VARMA(p, q) Model)
Assume that $\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$ is a multivariate WSS time series with zero mean. Then a VARMA(p, q) model of dimension k is given by

$$\tilde{\boldsymbol{y}}_t = \sum_{j=1}^{p} \boldsymbol{\Phi}_j \tilde{\boldsymbol{y}}_{t-j} - \sum_{i=1}^{q} \boldsymbol{\Psi}_i \boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{2.2}$$

where $\boldsymbol{\Phi}_j, \boldsymbol{\Psi}_j \in \mathbb{R}^{k\times k}$. The first sum in Eq. (2.2) is the vector autoregressive VAR(p) part, the second sum in Eq. (2.2) is the vector moving average VMA(q) part, and $\{\boldsymbol{u}_t\}_{t\in\mathbb{Z}}$ is a vector white noise process. [Box+16, p. 527]

An alternative way of formulating Eq. (2.2) is by introducing parametric functions $\boldsymbol{\Phi}(B)$ and $\boldsymbol{\Psi}(B)$ where B is the backward lag operator defined as $B^j \boldsymbol{y}_t = \boldsymbol{y}_{t-j}$ for $j \in \mathbb{Z}$. Specifically, we can write

$$\boldsymbol{\Phi}(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.3}$$

where

$$\boldsymbol{\Phi}(B) = \boldsymbol{I} - \boldsymbol{\Phi}_1 B - \cdots - \boldsymbol{\Phi}_p B^p, \tag{2.4}$$
$$\boldsymbol{\Psi}(B) = \boldsymbol{I} - \boldsymbol{\Psi}_1 B - \cdots - \boldsymbol{\Psi}_q B^q. \tag{2.5}$$

The interpretation of a VARMA(p, q)-process as a linear filter with transfer function $\boldsymbol{\Phi}(B)^{-1}\boldsymbol{\Psi}(B)$ is now apparent from Eq. (2.3) as

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{\Phi}(B)^{-1}\boldsymbol{\Psi}(B)\boldsymbol{u}_t. \tag{2.6}$$

We can also express the VARMA(p, q)-process in terms of the original process $\{\boldsymbol{y}_t\}_{t\in\mathbb{Z}}$ as

$$\boldsymbol{\Phi}(B)\boldsymbol{y}_t = \boldsymbol{\mu}_0 + \boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.7}$$

where $\boldsymbol{\mu}_0 = (\boldsymbol{I} - \boldsymbol{\Phi}_1 - \cdots - \boldsymbol{\Phi}_p)\boldsymbol{\mu}_{\boldsymbol{y}}$.

A VARMA(p, q) model can be seen as a composition of a VAR(p) model and a VMA(q) model. These are defined as

$$\boldsymbol{\Phi}(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{u}_t, \tag{2.8}$$
$$\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.9}$$

for VAR(p) and VMA(q), respectively. In this context, the VAR(p)-process is a k-dimensional infinite impulse response filter and the VMA(q)-process is a k-dimensional finite impulse response filter given the input to the filter, $\boldsymbol{u}_t$. In Section 2.1.1, we will delve into the causality and invertibility properties of the two filters and introduce the Wold representation of VAR(p), VMA(q), and VARMA(p, q) models.

In the univariate case, i.e. $k = 1$, the VARMA(p, q) model is called an ARMA(p, q) model.

2.1.1 Causality and Invertibility

The Wold representation of VAR(p), VMA(q), and VARMA(p, q) are ones in which $\tilde{\boldsymbol{y}}_t$ is given as a linear combination of the past values of the noise process $\boldsymbol{u}_{t-j}$ and some parameters $\boldsymbol{\mathcal{A}}_j$, i.e.

$$\tilde{\boldsymbol{y}}_t = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{t-j}. \tag{2.10}$$

This representation is trivial for a VMA(q) model since then $\boldsymbol{\mathcal{A}}_0 = \boldsymbol{I}$, $\boldsymbol{\mathcal{A}}_j = \boldsymbol{\Psi}_j$ for $j = 1, \ldots, q$ and zero otherwise. However, for VAR(p) and VARMA(p, q) attaining the Wold representation requires inverting the filter $\boldsymbol{\Phi}(B)$.

Consider a VARMA(p, q) model, $\boldsymbol{\Phi}(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}(B)\boldsymbol{u}_t$. An alternative representation of the process is through the inverse filter

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{\Phi}^{-1}(B)\boldsymbol{\Psi}(B)\boldsymbol{u}_t = \boldsymbol{\mathcal{A}}(B)\boldsymbol{u}_t \tag{2.11}$$

where $\boldsymbol{\mathcal{A}}(B) = \boldsymbol{\Phi}^{-1}(B)\boldsymbol{\Psi}(B)$. Using the series expansion with undetermined coefficients, we can write [KW07, p. 41]

$$\boldsymbol{\mathcal{A}}(B) = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j B^j \tag{2.12}$$

where $\sum_{j=0}^{\infty} \|\boldsymbol{\mathcal{A}}_j\|_F < \infty$ and $\|\cdot\|_F$ denotes the Frobenius norm [Neu16, pp. 205 & 219]. Note that Eq. (2.11) is the Wold representation of the VARMA(p, q) model. The expression for the inverse filter in Eq. (2.12) only holds in case the roots of the polynomial $|\boldsymbol{\Phi}(z)|$ for $z \in \mathbb{C}$ lie outside the unit circle, i.e. if $|\boldsymbol{\Phi}(z)| = 0$, then $|z| > 1$ [SS17, p. 86]. In this case, the VARMA(p, q) model is said to be stable and causal. The same statement holds for VAR(p) models.

Another representation of a VARMA(p, q) model can also be considered, in which the noise process $\boldsymbol{u}_t$ is expressed by the past observations of $\tilde{\boldsymbol{y}}_t$. In this case, we are considering the filter

$$\boldsymbol{u}_t = \boldsymbol{\Psi}^{-1}(B)\boldsymbol{\Phi}(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{\mathcal{B}}(B)\tilde{\boldsymbol{y}}_t. \tag{2.13}$$

where $\boldsymbol{\mathcal{B}}(B) = \boldsymbol{\Psi}^{-1}(B)\boldsymbol{\Phi}(B)$. Once again using the series expansion with undetermined coefficients

$$\boldsymbol{\mathcal{B}}(B) = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{B}}_j B^j \tag{2.14}$$

where $\sum_{j=0}^{\infty} \|\boldsymbol{\mathcal{B}}_j\|_F < \infty$. As before, the expression in Eq. (2.14) for the filter $\boldsymbol{\mathcal{B}}(B)$ only holds in case the roots of the polynomial $|\boldsymbol{\Psi}(z)|$ for $z \in \mathbb{C}$ lie outside the unit circle [SS17, p. 86]. In this case, the VARMA(p, q) model is said to be invertible. The same statement holds for VMA(q) models. The results presented in this section is summarised in Proposition 2.3.

Proposition 2.3 (Causality and Invertibility of VARMA Models)
A VARMA(p,q) model is causal and stable if the roots of $|\boldsymbol{\Phi}(z)|$ lie outside of the unit circle. In this case, the VARMA(p,q) model can be expressed as

$$\tilde{\boldsymbol{y}}_t = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{t-j} \tag{2.15}$$

for $t \in \mathbb{Z}$ and where $\boldsymbol{\mathcal{A}}_j \in \mathbb{R}^{k\times k}$. Moreover, a VARMA$(p,q)$ model is invertible if the roots of $|\boldsymbol{\Psi}(z)|$ lie outside the unit circle in which case it can be expressed as

$$\boldsymbol{u}_t = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{B}}_j \tilde{\boldsymbol{y}}_{t-j} \tag{2.16}$$

for $t \in \mathbb{Z}$ and where $\boldsymbol{\mathcal{B}}_j \in \mathbb{R}^{k\times k}$. [SS17, pp. 280-281][Box+16, pp. 527-528]

In the simple case, we consider the roots of $\phi(z)$ and $\psi(z)$. If the roots of $\phi(z)$ lie outside the unit circle, the ARMA(p,q) model is causal and if the roots of $\psi(z)$ lie outside the unit circle, then the ARMA(p,q) model is invertible [SS17, pp. 85-86].

2.1.2 Issues of VARMA Models

In this section, we will delve into some of the issues arising for the VARMA(p,q) models but before doing so, we will introduce a result which will be useful later in this section.

Proposition 2.4
Let $\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$ be a causal VARMA(p,q)-process. Then the cross-covariance matrix $\boldsymbol{\Gamma}(h)$ for $h \geq 0$ is given by

$$\boldsymbol{\Gamma}(h) = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{\Sigma}_{\boldsymbol{u}} \boldsymbol{\mathcal{A}}_{j+h}^T \tag{2.17}$$

and $\boldsymbol{\Gamma}(-h) = \boldsymbol{\Gamma}(h)^T$. [SS17, pp. 280-281].

Proof.
First we will show Eq. (2.17) taking offset in the definition of the cross-covariance.

$$\begin{aligned}\boldsymbol{\Gamma}(h) &= \mathbb{E}[\tilde{\boldsymbol{y}}_t \tilde{\boldsymbol{y}}_{t+h}^T] \\ &= \mathbb{E}[(\sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{t-j})(\sum_{j=-h}^{\infty} \boldsymbol{\mathcal{A}}_{j+h} \boldsymbol{u}_{t-j})^T] \\ &= \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{\Sigma}_{\boldsymbol{u}} \boldsymbol{\mathcal{A}}_{j+h}^T. \end{aligned} \tag{2.18}$$

Next we will show $\boldsymbol{\Gamma}(-h) = \boldsymbol{\Gamma}(h)^T$

$$\begin{aligned}\boldsymbol{\Gamma}(-h) &= \mathbb{E}[\tilde{\boldsymbol{y}}_t \tilde{\boldsymbol{y}}_{t-h}^T] \\ &= \mathbb{E}[\tilde{\boldsymbol{y}}_{t+h} \tilde{\boldsymbol{y}}_t^T] \\ &= \mathbb{E}[(\sum_{j=-h}^{\infty} \boldsymbol{\mathcal{A}}_{j+h} \boldsymbol{u}_{t-j})(\sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{t-j})^T] \\ &= \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_{j+h} \boldsymbol{\Sigma}_{\boldsymbol{u}} \boldsymbol{\mathcal{A}}_j^T. \end{aligned} \tag{2.19}$$

■

Now we are ready to discuss the issues arising from the VARMA(p, q) model. One of these issues is the number of parameters in the model. Both the VAR and VMA part introduce $k \times k$ parameters and hence, the number of parameters increases rapidly when the model order increases. In addition to this, the parameter estimates in the obtained model can be highly correlated and together with the big number of parameters in the model, the results of the model may be difficult to interpret.

Furthermore, issues regarding uniqueness of the parameters, i.e. lack of identifiability of the model arise for the VARMA(p, q) models. Situations can occur where we have two different VARMA models $\boldsymbol{\Phi}(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}(B)\boldsymbol{u}_t$ and $\boldsymbol{\Phi}_*(B)\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}_*(B)\boldsymbol{u}_t$ for which

$$\boldsymbol{\mathcal{A}}(B) = \boldsymbol{\Phi}^{-1}(B)\boldsymbol{\Psi}(B) = \boldsymbol{\Phi}_*^{-1}(B)\boldsymbol{\Psi}_*(B). \tag{2.20}$$

Hence, by Proposition 2.4 these two VARMA models yield the same covariance matrix structure $\{\boldsymbol{\Gamma}(h)\}_{h\in\mathbb{Z}}$ and are hence the same process. In such situations, the two models are said to be exchangeable or observationally equivalent. The reason that two observationally equivalent VARMA(p, q) models can exist is that the filters $\boldsymbol{\Phi}(B)$ and $\boldsymbol{\Psi}(B)$ could share some common factor $\boldsymbol{U}(B)$, i.e.

$$\boldsymbol{\Phi}(B) = \boldsymbol{U}(B)\boldsymbol{\Phi}_*(B), \tag{2.21}$$

$$\boldsymbol{\Psi}(B) = \boldsymbol{U}(B)\boldsymbol{\Psi}_*(B) \tag{2.22}$$

where the orders of $\boldsymbol{\Phi}(B)$ and $\boldsymbol{\Psi}(B)$ are equal to the orders of $\boldsymbol{\Phi}_*(B)$ and $\boldsymbol{\Psi}_*(B)$, respectively. From this, we see that the common factor $\boldsymbol{U}(B)$ would cancel out

when determining $\boldsymbol{A}(B) = \boldsymbol{\Phi}^{-1}(B)\boldsymbol{\Psi}(B)$ which by Eq. (2.17) implies the same cross-covariance and thereby also lack of identifiability of the model.

If $\boldsymbol{\Phi}(B)$ and $\boldsymbol{\Psi}(B)$ are uniquely determined by either the covariance matrices $\boldsymbol{\Gamma}(h)$ in the case of a causal process or the $\boldsymbol{A}(B)$ filter, the VARMA(p, q) model is said to be identifiable. [Box+16, pp. 528-529]

2.1.3 Vector Autoregressive Integrated Moving Average

The VARMA-processes assumes stationary data. In this section, we will introduce an extension of the VARMA-processes for non-stationary data. This extension is the class of vector autoregressive integrated moving average (VARIMA)-processes.

The VARIMA model is obtained by introducing a differencing operator $\nabla^d \tilde{\boldsymbol{y}}_t = (1-B)^d \tilde{\boldsymbol{y}}_t$ where $d \geq 1$ is the order of differencing. The definition of VARIMA(p, d, q) models is given below.

Definition 2.5 (VARIMA(p, d, q) Model)
Assume that $\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$ is a multivariate time series with zero mean. Then a VARIMA(p, d, q) model is given by

$$\boldsymbol{\Phi}(B)\nabla^d \tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.23}$$

where the process

$$\nabla^d \tilde{\boldsymbol{y}}_t = (1 - B)^d \tilde{\boldsymbol{y}}_t \tag{2.24}$$

is VARMA [Box+16, ch. 4].

The purpose of differencing is to form a time series $\nabla^d \tilde{\boldsymbol{y}}_t$ which is stationary.

2.1.4 Seasonal VARIMA

In many applications, the strongest temporal dependency may not be on the most recent observations. For example when considering wind power production forecasting, we might expect a diurnal cycle which implies that the current wind power production is dependent on the wind power production the previous day. We might expect this behaviour due to the strong dependency between the wind power production and the wind speed and due to diurnality of wind speed [Bur+01, pp. 11-16]. This diurnal cycle expresses a certain seasonality. In Definition 2.6, an extension of the VARIMA model to include seasonality is defined.

Definition 2.6 (Multiplicative Seasonal VARIMA Models)
Assume that $\{\tilde{\boldsymbol{y}}_t\}_{t\in\mathbb{Z}}$ is a multivariate time series with zero mean. Then a multiplicative seasonal VARIMA (s-VARIMA) model with seasonal lag s, autoregressive

seasonal order p_s, moving average seasonal order q_s, and seasonal differencing order d_s is given as

$$\boldsymbol{\Phi}_s(B^s)\boldsymbol{\Phi}(B)\nabla^{d_s,d}\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}_s(B^s)\boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.25}$$

where $\boldsymbol{u}_t$ is a vector white noise process, $\nabla^{d_s,d}\tilde{\boldsymbol{y}}_t = (1-B^s)^{d_s}(1-B)^d\tilde{\boldsymbol{y}}_t$, and

$$\boldsymbol{\Phi}_s(B^s) = 1 - \boldsymbol{\Phi}_{s,1}B^s - \cdots - \boldsymbol{\Phi}_{s,p_s}B^{p_s s}, \tag{2.26}$$

$$\boldsymbol{\Psi}_s(B^s) = 1 - \boldsymbol{\Psi}_{s,1}B^s - \cdots - \boldsymbol{\Psi}_{s,q_s}B^{q_s s}. \tag{2.27}$$

The model is denoted s-VARIMA$(p,d,q)\times(p_s,d_s,q_s)_s$. [SS17, p. 150]

In the case of a s-VARIMA model, differencing can be done resulting in a process which follows a s-VARMA model. Therefore, we can consider a s-VARMA model without loss of generality. In the following, we will express the s-VARMA model in a way which mimics a usual VARMA model.

Consider the following difference equation formulation of Eq. (2.25) for $d_s = 0$

$$\begin{aligned}\tilde{\boldsymbol{y}}_t = &\sum_{j_s=1}^{p_s}\boldsymbol{\Phi}_{s,j_s}\tilde{\boldsymbol{y}}_{t-sj_s} + \sum_{j=1}^{p}\boldsymbol{\Phi}_j\tilde{\boldsymbol{y}}_{t-j} - \sum_{j_s=1}^{p_s}\sum_{j=1}^{p}\boldsymbol{\Phi}_{s,j_s}\boldsymbol{\Phi}_j\tilde{\boldsymbol{y}}_{t-(sj_s+j)}\\ &- \sum_{i_s=1}^{q_s}\boldsymbol{\Psi}_{s,i_s}\boldsymbol{u}_{t-si_s} - \sum_{i=1}^{q}\boldsymbol{\Psi}_i\boldsymbol{u}_{t-i} + \sum_{i_s=1}^{q_s}\sum_{i=1}^{q}\boldsymbol{\Psi}_{s,i_s}\boldsymbol{\Psi}_i\boldsymbol{u}_{t-(si_s+i)} + \boldsymbol{u}_t.\end{aligned} \tag{2.28}$$

Assume that $s > \max(p,q)$ and note that if this is not the case, then it is redundant to consider a s-VARMA model instead of a VARMA model. Define the index sets

$$\mathcal{S}_{\mathrm{AR}}(p,p_s) = \bigcup_{j_s=0}^{p_s}\{j^*\in\mathbb{N}\setminus\{0\} | sj_s \le j^* \le sj_s + p\}, \tag{2.29}$$

$$\mathcal{S}_{\mathrm{MA}}(q,q_s) = \bigcup_{i_s=0}^{q_s}\{i^*\in\mathbb{N}\setminus\{0\} | si_s \le i^* \le si_s + q\}. \tag{2.30}$$

The sets $\mathcal{S}_{\mathrm{AR}}(p,p_s)$ and $\mathcal{S}_{\mathrm{MA}}(q,q_s)$ will be referred to as the active autoregressive time lags and the active moving average time lags, respectively. Moreover, let $p^* = \max(\mathcal{S}_{\mathrm{AR}}(p,p_s))$ and $q^* = \max(\mathcal{S}_{\mathrm{MA}}(q,q_s))$ be the maximum autoregressive time lag and the maximum moving average time lag, respectively. In this case, Eq. (2.28) can be expressed as

$$\tilde{\boldsymbol{y}}_t = \sum_{j^*\in\mathcal{S}_{\mathrm{AR}}(p,p_s)}\boldsymbol{\Phi}^*_{j^*}\tilde{\boldsymbol{y}}_{t-j^*} - \sum_{i^*\in\mathcal{S}_{\mathrm{MA}}(q,q_s)}\boldsymbol{\Psi}^*_{i^*}\boldsymbol{u}_{t-i^*} + \boldsymbol{u}_t \tag{2.31}$$

where for $j^*\in\mathcal{S}_{\mathrm{AR}}(p,p_s)$

$$\boldsymbol{\Phi}^*_{j^*} = \begin{cases}\boldsymbol{\Phi}_{j^*} & j^* = 1,2,\ldots,p,\\ \boldsymbol{\Phi}_{s,\frac{j^*}{s}} & j^* = s,2s,\ldots,p_s s,\\ -\boldsymbol{\Phi}_{s,\frac{j^*-j}{s}}\boldsymbol{\Phi}_{j^*-sj_s} & \text{otherwise,}\end{cases} \tag{2.32}$$

where $j, j_s \in \mathbb{N}$ with $0 \leq j \leq p$ such that $j^* = p \cdot j_s + j$ and for $i^* \in \mathcal{S}_{\text{MA}}(q, q_s)$

$$\boldsymbol{\Psi}_{i^*}^* = \begin{cases} \boldsymbol{\Psi}_{i^*} & i^* = 1, 2, \ldots, q, \\ \boldsymbol{\Psi}_{s,\frac{i^*}{s}} & i^* = s, 2s, \ldots, q_s s, \\ -\boldsymbol{\Psi}_{s,\frac{i^*-i}{s}} \boldsymbol{\Psi}_{i^*-si_s} & \text{otherwise.} \end{cases} \tag{2.33}$$

where $i, i_s \in \mathbb{N}$ with $0 \leq i \leq q$ such that $i^* = q \cdot i_s + i$. The benefit of the formulation of the s-VARMA models as in Eq. (2.31) is that this form can be written on the form of a VARMA model where the only difference is that for the autoregressive time lags $1, \ldots, p^*$ which are not active the parameter matrix is the zero matrix. The same applies for the moving average time lags $1, \ldots, q^*$ which are not active.

2.1.5 VARMA Models with Exogenous Variables

When we consider a multivariate time series $\boldsymbol{y}_t$ with mean $\boldsymbol{\mu}_{\boldsymbol{y}}$, the VARMA$(p, q)$ model in Eq. (2.2) becomes

$$\boldsymbol{y}_t = \boldsymbol{\mu}_0 + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{i=1}^{q} \boldsymbol{\Psi}_i \boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{2.34}$$

where $\boldsymbol{\mu}_0 = (\boldsymbol{I} - \boldsymbol{\Phi}_1 - \cdots - \boldsymbol{\Phi}_p)\boldsymbol{\mu}_{\boldsymbol{y}}$. In practice, there may be exogenous variables which have an effect on the process. A method of including such variables in the model is by introducing a mean function $\boldsymbol{\mu}_t$ in place of $\boldsymbol{\mu}_0$ which is a parametric linear function of the exogenous variables $\boldsymbol{z}_t \in \mathbb{R}^r$. The model obtained in this case is called the VARMAX(p, q) model and is given as

$$\boldsymbol{y}_t = \boldsymbol{\Xi} \boldsymbol{z}_t + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{i=1}^{q} \boldsymbol{\Psi}_i \boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{2.35}$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{k \times r}$ is a parameter matrix [SS17, pp. 273 & 280].

We can also include exogenous variables in the VARIMA model in Section 2.1.3 and the s-VARIMA model in Section 2.1.4, resulting in the VARIMAX(p, d, q) and s-VARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ models, respectively.

The VARIMAX(p, d, q) model is given as

$$\boldsymbol{\Phi}(B)\nabla^d \boldsymbol{y}_t = \boldsymbol{\Xi}\nabla^d \boldsymbol{z}_t + \boldsymbol{\Psi}(B)\boldsymbol{u}_t \tag{2.36}$$

and the s-VARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ model is given as

$$\boldsymbol{\Phi}_s(B^s)\boldsymbol{\Phi}(B)\nabla^{d_s,d} \boldsymbol{y}_t = \boldsymbol{\Xi}\nabla^{d_s,d} \boldsymbol{z}_t + \boldsymbol{\Psi}_s(B^s)\boldsymbol{\Psi}(B)\boldsymbol{u}_t. \tag{2.37}$$

The motivation for introducing VARMA models with exogenous variables in this project is that the wind power production is highly dependent on the weather and when we model the wind power production as a time series, the weather can be included as an exogenous variable.

In the following section, we will delve into the estimation and forecasting aspects of the VARMA models.

2.2 Estimation and Forecasting for VARMA Models

The purpose of forecasting is to infer the future values of the time series given observations of the past and present. In the estimation phase, a VARMA(p, q) model is fitted to the data based on some optimum criterion. During forecasting, the inferred model is used for prediction. Let $\boldsymbol{y}_t$ be a time series observed for $t = 1, \ldots, n$ and let $\hat{\boldsymbol{y}}^n_{n+\tau}$ be a predictor of $\boldsymbol{y}_{n+\tau}$ using the previous n observations. Then $\hat{\boldsymbol{y}}^n_{n+\tau}$ is the minimum mean square error predictor of $\boldsymbol{y}_{n+\tau}$ given $\{\boldsymbol{y}_t\}^n_{t=1}$ if it is the predictor which minimises

$$\mathbb{E}\Big[\big(\boldsymbol{y}_{n+\tau} - \hat{\boldsymbol{y}}^n_{n+\tau}\big)\big(\boldsymbol{y}_{n+\tau} - \hat{\boldsymbol{y}}^n_{n+\tau}\big)^T\Big]. \tag{2.38}$$

It can be shown that the predictor which minimises the mean square error matrix is the conditional expectation [Box+16, pp. 131 & 535]

$$\boldsymbol{y}^n_{n+\tau} = \mathbb{E}[\boldsymbol{y}_{n+\tau}|\boldsymbol{y}_n, \ldots, \boldsymbol{y}_1]. \tag{2.39}$$

However, this does not provide a productive way of predicting $\boldsymbol{y}_{n+\tau}$. In the following sections, the likelihood for VARMA models will be considered and subsequently the truncated τ-ahead prediction equations for a VARMA(p, q) model will be introduced.

2.2.1 Estimation

The problem of estimating parameters in a VARMA(p, q) model is a non-linear optimisation problem and as such admits to the use of numerical methods [SS17, p. 115]. In this project, we will introduce maximum likelihood estimation with a Newton-like method described in Appendix B in order to estimate the parameters for VARMA models. This will be the content of Chapter 3. In this section, we will introduce a likelihood function assuming normality for the vector white noise process and outline a method for computing an initial parameter estimate.

Consider n observations $\{\boldsymbol{y}_t\}^n_{t=1}$ and let $\boldsymbol{\Theta} = (\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_p, \boldsymbol{\Psi}_1, \boldsymbol{\Psi}_q)$ be a matrix consisting of the parameters for the VARMA(p, q) model. The conditional likelihood given the past can be written as

$$L(\boldsymbol{\Theta}) = \prod_{t=1}^{n} f(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1) \tag{2.40}$$

where the mean vector of $\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1$ is $\boldsymbol{y}^{t-1}_t = \mathbb{E}[\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1]$ and the variance is $\boldsymbol{\Sigma}_t = \mathbb{E}[(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t)(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t)^T]$. Using the normality assumption for the vector white noise process, we obtain

$$L(\boldsymbol{\Theta}) = \prod_{t=1}^{n} \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}_t|}} \exp\bigg(-\frac{1}{2}\big(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t\big)^T\boldsymbol{\Sigma}^{-1}_t\big(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t\big)\bigg). \tag{2.41}$$

Now we take the logarithm of the likelihood function and remove the term that does not depend on the parameter [Box+16, pp. 532-533][SS17, p. 304]

$$\ell(\boldsymbol{\Theta}) = -\frac{1}{2}\sum_{t=1}^{n}\log\big(|\boldsymbol{\Sigma}_t|\big) - \frac{1}{2}\Big(\big(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t\big)^T\boldsymbol{\Sigma}^{-1}_t\big(\boldsymbol{y}_t - \boldsymbol{y}^{t-1}_t\big)\Big). \tag{2.42}$$

This is the log-likelihood function for VARMA models and in maximum likelihood estimation we are interested in maximising this function. As mentioned, details of how the likelihood can be maximised will be introduced in Chapter 3.

While the problem of estimating parameters in a VARMA(p, q) or a VARMAX(p, q) model is a non-linear optimisation problem, it is a linear optimisation problem for a VARX(p) model. Hence, for a VARX(p) model the conditional maximum likelihood methodology can be used. In this case, we might assume the noise process $\{\boldsymbol{u}_t\}_{t=1}^n$ is normally distributed such that the conditional distribution of $\boldsymbol{y}_t$ given the past p observations is

$$\mathcal{P}(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_{t-p}) \sim \mathcal{N}\big(\boldsymbol{\Xi}\boldsymbol{z}_t + \boldsymbol{\Phi}_j \sum_{j=1}^{p} \boldsymbol{y}_{t-j}, \boldsymbol{\Sigma}_u\big). \tag{2.43}$$

The conditional likelihood function which arises can then be maximised in closed form using standard results from multivariate linear regression. Specifically, the estimates of $\boldsymbol{\Phi}_j$ become the ordinary least squares estimate and the variance $\boldsymbol{\Sigma}_u$ can then be estimated using the profile likelihood.

Using the aforementioned result in conjunction with the invertible representation of the VARMAX(p, q) model, see Proposition 2.3, the estimation process of the parameters can be done following a sequential algorithm [KW07, sec. 2.3.2]. Consider for instance a VARMAX(p, q) model

$$\boldsymbol{y}_t = \boldsymbol{\Xi}\boldsymbol{z}_t + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{i=1}^{q} \boldsymbol{\Psi}_i \boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{2.44}$$

for $t = 1, \ldots, n$ where $\boldsymbol{y}_t \in \mathbb{R}^k$ are the endogenous variables, $\boldsymbol{u}_t$ is a k-dimensional vector white noise process, and $\boldsymbol{z}_t \in \mathbb{R}^r$ are the exogenous variables. First, since a VARMAX(p, q) model is equivalent to a VARX(∞) model, which is a consequence of invertibility, see Proposition 2.3, a VARX(m) model can be fitted to the data where $m > p$. This fit can be done using ordinary least squares which when assuming the vector white noise process is Gaussian coincides with maximum likelihood. Let $\boldsymbol{\theta}^{(\mathrm{AR})}$ denote the parameters for the VARX(m) model where $m > p$. The parameters can then be estimated as

$$\hat{\boldsymbol{\theta}}^{(\mathrm{AR})} = \big(\boldsymbol{X}^{(\mathrm{AR})^T}\boldsymbol{X}^{(\mathrm{AR})}\big)^{-1}\boldsymbol{X}^{(\mathrm{AR})^T}\boldsymbol{Y} \tag{2.45}$$

where

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_{p+1}^T \\ \boldsymbol{y}_{p+2}^T \\ \vdots \\ \boldsymbol{y}_n^T \end{bmatrix}, \quad \boldsymbol{X}^{(\mathrm{AR})} = \begin{bmatrix} \boldsymbol{y}_p^T & \cdots & \boldsymbol{y}_1^T & \boldsymbol{z}_{p+1}^T \\ \boldsymbol{y}_{p+1}^T & \cdots & \boldsymbol{y}_2^T & \boldsymbol{z}_{p+2}^T \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{y}_{n-1}^T & \cdots & \boldsymbol{y}_{n-p}^T & \boldsymbol{z}_n^T \end{bmatrix}. \tag{2.46}$$

Using this model, the noise process is estimated as

$$\hat{\boldsymbol{U}} = \boldsymbol{Y} - \boldsymbol{X}^{(\mathrm{AR})}\hat{\boldsymbol{\theta}}^{(\mathrm{AR})} \tag{2.47}$$

where $\hat{\boldsymbol{U}} = (\hat{\boldsymbol{u}}_{p+1}, \dots, \hat{\boldsymbol{u}}_n)^T$. When the noise process has been estimated, the parameters of the VARMAX(p, q) model can be estimated using the conditional distribution

$$\mathcal{P}(\boldsymbol{y}_t | \boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_{t-p}, \hat{\boldsymbol{u}}_{t-1}, \dots, \hat{\boldsymbol{u}}_{t-q}) \sim \mathcal{N}\big(\boldsymbol{\Xi}\boldsymbol{z}_t + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{i=1}^{q} \boldsymbol{\Psi}_i \hat{\boldsymbol{u}}_{t-i}, \boldsymbol{\Sigma}_u\big). \tag{2.48}$$

Using the same approach as maximising Eq. (2.43), let $M = \max(p, q)$ and $\boldsymbol{\theta}$ be the parameters the VARMAX(p, q) model and estimate these parameters as

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{Y} \tag{2.49}$$

where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{y}_M^T & \cdots & \boldsymbol{y}_{M+1-p}^T & \hat{\boldsymbol{u}}_M^T & \cdots & \hat{\boldsymbol{u}}_{M+1-q}^T & \boldsymbol{z}_{M+1}^T \\ \boldsymbol{y}_{M+1}^T & \cdots & \boldsymbol{y}_{M+2-p}^T & \hat{\boldsymbol{u}}_{M+1}^T & \cdots & \hat{\boldsymbol{u}}_{M+2-q}^T & \boldsymbol{z}_{M+2}^T \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{y}_{n-1}^T & \cdots & \boldsymbol{y}_{n-p}^T & \hat{\boldsymbol{u}}_{n-1}^T & \cdots & \hat{\boldsymbol{u}}_{n-q}^T & \boldsymbol{z}_n^T \end{bmatrix}. \tag{2.50}$$

This methodology can be applied as an initial parameter estimate which can then be improved upon using a numerical maximum likelihood algorithm.

Note that this method can be extended to the class of s-VARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ since these types of model can be formulated as VARMAX$(p*, q*)$ models as described in Sections 2.1.3 to 2.1.5.

2.2.2 Forecasting

Assume now that the parameters for a VARMA(p, q) model have been estimated. For VARMA models, considering the minimum mean square error predictor $\boldsymbol{y}_{n+\tau}^n = \mathbb{E}[\boldsymbol{y}_{n+\tau} | \boldsymbol{y}_n, \dots, \boldsymbol{y}_1]$ is problematic since $\mathbb{E}[\boldsymbol{u}_t | \boldsymbol{y}_n, \dots, \boldsymbol{y}_1] \neq \boldsymbol{0}$ for $t = 1, \dots, n$. Instead we can consider the expected value conditioned on the entire past

$$\bar{\boldsymbol{y}}_{n+\tau} = \mathbb{E}[\boldsymbol{y}_{n+\tau} | \boldsymbol{y}_n, \dots, \boldsymbol{y}_1, \boldsymbol{y}_0, \boldsymbol{y}_{-1}, \dots]. \tag{2.51}$$

This is naturally not equivalent to $\boldsymbol{y}_{n+\tau}^n$, however for large sample sizes it does provide a good approximation. [SS17, p. 107]

Assuming the VARMA model is causal, the process can be expressed as in Eq. (2.11)

$$\boldsymbol{y}_{n+\tau} = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{n+\tau-j}. \tag{2.52}$$

Now by the assumptions of causality and invertibility, see Eq. (2.13),

$$\bar{\boldsymbol{u}}_t = \mathbb{E}[\boldsymbol{u}_t | \boldsymbol{y}_n, \dots, \boldsymbol{y}_1, \boldsymbol{y}_0, \boldsymbol{y}_{-1}, \dots] = \begin{cases} \boldsymbol{0}, & n < t, \\ \boldsymbol{u}_t, & t \leq n. \end{cases} \tag{2.53}$$

This implies that [SS17, p. 107]

$$\bar{\boldsymbol{y}}_{n+\tau} = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{A}}_j \bar{\boldsymbol{u}}_{n+\tau-j} = \sum_{j=\tau}^{\infty} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{n+\tau-j}. \tag{2.54}$$

The error of this prediction is

$$\boldsymbol{y}_{n+\tau} - \bar{\boldsymbol{y}}_{n+\tau} = \sum_{j=0}^{\tau-1} \boldsymbol{\mathcal{A}}_j \boldsymbol{u}_{n+\tau-j} \tag{2.55}$$

which is a sum of the errors for which no information is known multiplied by the parameters. The mean square error of this prediction is then

$$\mathbb{E}\big[\big(\boldsymbol{y}_{n+\tau} - \bar{\boldsymbol{y}}_{n+\tau}\big)\big(\boldsymbol{y}_{n+\tau} - \bar{\boldsymbol{y}}_{n+\tau}\big)^T\big] = \sum_{j=0}^{\tau-1} \boldsymbol{\mathcal{A}}_j \boldsymbol{\Sigma}_{\boldsymbol{u}} \boldsymbol{\mathcal{A}} j^T \tag{2.56}$$

where $\boldsymbol{\mathcal{A}}_0 = \boldsymbol{I}$ [Box+16, pp. 535-536][SS17, p. 108].

Consider now the invertible representation of the VARMA model as in Eq. (2.13)

$$\boldsymbol{u}_{n+\tau} = \sum_{j=0}^{\infty} \boldsymbol{\mathcal{B}}_j \boldsymbol{y}_{n+\tau-j}. \tag{2.57}$$

Taking the conditional expectation given $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}, \dots$ yields

$$\boldsymbol{0} = \bar{\boldsymbol{y}}_{n+\tau} + \sum_{j=1}^{\infty} \boldsymbol{\mathcal{B}}_j \bar{\boldsymbol{y}}_{n+\tau-j} \tag{2.58}$$

such that

$$\bar{\boldsymbol{y}}_{n+\tau} = -\sum_{j=1}^{\infty} \boldsymbol{\mathcal{B}}_j \bar{\boldsymbol{y}}_{n+\tau-j}. \tag{2.59}$$

In practice, since $\boldsymbol{y}_t$ is only observed for $t = 1, \dots, n$, we need to use a truncated version of Eq. (2.59). The truncated τ-ahead prediction is then defined as

$$\begin{aligned} \bar{\boldsymbol{y}}^n_{n+\tau} &= -\sum_{j=1}^{\tau-1} \boldsymbol{\mathcal{B}}_j \bar{\boldsymbol{y}}_{n+\tau-j} - \sum_{j=\tau}^{n+\tau-1} \boldsymbol{\mathcal{B}}_j \bar{\boldsymbol{y}}_{n+\tau-j}, \\ &= -\sum_{j=1}^{\tau-1} \boldsymbol{\mathcal{B}}_j \bar{\boldsymbol{y}}_{n+\tau-j} - \sum_{j=\tau}^{n+\tau-1} \boldsymbol{\mathcal{B}}_j \boldsymbol{y}_{n+\tau-j}. \end{aligned} \tag{2.60}$$

It can be shown that the difference equations introduced in Proposition 2.7 allow computing the truncated forecasting for VARMA(p, q) models.

Proposition 2.7 (Truncated τ-Ahead Prediction)
The truncated τ-ahead prediction, denoted $\bar{\boldsymbol{y}}_{n+\tau}^n$, is computed using the difference equation formulation of the VARMA(p,q) model as [SS17, p. 109][Box+16, p. 535]

$$\bar{\boldsymbol{y}}_t^n = \begin{cases} \sum_{j=1}^{p} \boldsymbol{\Phi}_j \bar{\boldsymbol{y}}_{t-j}^n - \sum_{j=1}^{q} \boldsymbol{\Psi}_j \bar{\boldsymbol{u}}_{t-j}^n, & n+1 \leq t, \\ \boldsymbol{y}_t, & 1 \leq t \leq n, \\ \mathbf{0}, & t \leq 0, \end{cases} \tag{2.61}$$

and the truncated prediction errors are given as

$$\bar{\boldsymbol{u}}_t^n = \begin{cases} \bar{\boldsymbol{y}}_t^n - \sum_{j=1}^{p} \boldsymbol{\Phi}_j \bar{\boldsymbol{y}}_{t-j}^n + \sum_{j=1}^{q} \boldsymbol{\Psi}_j \bar{\boldsymbol{u}}_{t-j}^n, & 1 \leq t \leq n, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{2.62}$$

3. State Space s-VARMAX

The state space representation encapsulates a wide class of models and as such provides a unified methodology for a wide range of problems in time series analysis. In this chapter, the state space representation of the s-VARMAX model will be introduced. This formulation provides tools for performing Bayesian forecasting and inference for the time series model which include Kalman filtering and Newton-like methods for estimating the parameters.

Theory regarding unconstrained optimisation and Newton-like methods can be seen in Appendix B. Additionally, theory regarding the minimum mean square error estimator, the orthogonality principle, and the projection theorem for this estimator and its linear counterpart can be seen in Appendix A.2. The theory introduced in these appendices will be useful when reading this chapter.

Consider a s-VARMAX$(p, q) \times (p_s, q_s)_s$ model given as

$$\boldsymbol{y}_t = \boldsymbol{\Xi}\boldsymbol{z}_t + \sum_{j \in \mathcal{S}_{\mathrm{AR}}(p,p_s)} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{i \in \mathcal{S}_{\mathrm{MA}}(q,q_s)} \boldsymbol{\Psi}_i \boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{3.1}$$

where $\{\boldsymbol{u}_t\}_{t\in\mathbb{Z}}$ is a vector white noise process and $\mathcal{S}_{\mathrm{AR}}(p, p_s)$ and $\mathcal{S}_{\mathrm{MA}}(q, q_s)$ are the active autoregressive lags and active moving average lags, respectively, see Eq. (2.31). We begin by considering a particular formulation of state space models which is pertinent to s-VARMAX models, by defining the so-called state and observation equations. Firstly, the state equation is given as

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{G}\boldsymbol{u}_t \tag{3.2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{k_x}$ is the k_x-dimensional state vector, $\boldsymbol{F} \in \mathbb{R}^{k_x \times k_x}$ is the transition matrix, $\boldsymbol{H} \in \mathbb{R}^{k_x \times r}$ is the external variable parameter matrix, $\boldsymbol{z}_t \in \mathbb{R}^r$ are external variables, $\boldsymbol{G} \in \mathbb{R}^{k_x \times k}$ is a parameter matrix, and $\boldsymbol{u}_t \overset{iid.}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_u)$ is the vector white noise process. Secondly, the observation equation is given as

$$\boldsymbol{y}_t = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{u}_t \tag{3.3}$$

where $\boldsymbol{y}_t \in \mathbb{R}^k$, and $\boldsymbol{A} \in \mathbb{R}^{k \times k_x}$ is the observation matrix. This type of state space model is referred to as the linear Gaussian state space model [SS17, p. 290]. Moreover in this state space model, the error term in the state and observation equations are correlated.

3.1 State Space Formulation

A k-dimensional VARMAX(p, q) model, given as in Eq. (2.35)

$$\boldsymbol{y}_t = \boldsymbol{\Xi}\boldsymbol{z}_t + \sum_{j=1}^{p} \boldsymbol{\Phi}_j \boldsymbol{y}_{t-j} - \sum_{j=1}^{q} \boldsymbol{\Psi}_j \boldsymbol{u}_{t-j} + \boldsymbol{u}_t, \tag{3.4}$$

can be represented in state space form. Assume that $p \geq q$ and consider the following definitions

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{\Phi}_1 & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{\Phi}_2 & \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Phi}_{p-1} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \boldsymbol{\Phi}_p & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{pmatrix}, \ \boldsymbol{G} = \begin{pmatrix} \boldsymbol{\Phi}_1 - \boldsymbol{\Psi}_1 \\ \vdots \\ \boldsymbol{\Phi}_q - \boldsymbol{\Psi}_q \\ \boldsymbol{\Phi}_{q+1} \\ \vdots \\ \boldsymbol{\Phi}_p \end{pmatrix}, \ \boldsymbol{H} = \begin{pmatrix} \boldsymbol{\Xi} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix}, \ \boldsymbol{A} = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix}. \tag{3.5}$$

The state and observation equations resulting in the VARMAX(p, q) model in Eq. (3.4) are then

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{F}\boldsymbol{x}_t + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{G}\boldsymbol{u}_t, \\ \boldsymbol{y}_t &= \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{u}_t \end{aligned} \tag{3.6}$$

which we notice is exactly the state space model formulation from the beginning of the chapter.

We note that in the notation, $\boldsymbol{I}$ is the k-dimensional identity matrix and $\boldsymbol{0}$ is the $(k \times k)$-dimensional zero matrix such that the state vector, $\boldsymbol{x}_t$, is k_x-dimensional while the observation vector, $\boldsymbol{y}_t$, is k-dimensional. In case $q > p$, then simply let $\boldsymbol{\Phi}_{p+1}, \ldots, \boldsymbol{\Phi}_q = \boldsymbol{0}$ in which case Eq. (3.6) with Eq. (3.5) still applies. [SS17, p. 323]

Consider now the s-VARMAX$(p, q) \times (p_s, q_s)_s$ model in Eq. (3.1). This model can also be written in state space form using the same ideas as for a VARMAX(p, q) model. In this case, simply let p^* and q^* denote the maximum delay in the autoregressive part and the moving average part, respectively. If the delay $j \in \{1, \ldots, p^*\}$ or $i \in \{1, \ldots, q^*\}$ is not included in the autoregressive part or the moving average part of the s-VARMAX model, then let $\boldsymbol{\Phi}_j = \boldsymbol{0}$ or $\boldsymbol{\Psi}_i = \boldsymbol{0}$, respectively.

3.2 Kalman Filtering and Forecasting

In this section, it will be shown how to make forecasts given the state space model in Eq. (3.6). Consider the minimum mean square error prediction of the state

$$\boldsymbol{x}_t^{t-1} = \mathbb{E}[\boldsymbol{x}_t | \boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1]. \tag{3.7}$$

Assuming normality of the noise process $\{\boldsymbol{u}_t\}_{t \in \mathbb{Z}}$, the conditional error variance of this forecast is

$$\boldsymbol{P}_t^{t-1} = \mathbb{E}\big[(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})^T\big] \tag{3.8}$$

where the conditioning on the past in the expected value has been neglected from notation as a consequence of the orthogonality principle as discussed in Section 2.2. The normality assumption will be a pertinent part of deriving the Kalman filter. However, when the normality assumption is broken, the results will still yield the minimum mean square predictions among linear predictors. [SS17, p. 295]

Now the Kalman filter with correlated noise is stated and proved.

Proposition 3.1 (Kalman Filter with Correlated Noise)
Consider the state space model in Eq. (3.6), and let $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. Then it holds that

$$\boldsymbol{x}_1^0 = \mathbb{E}[\boldsymbol{x}_1] = \boldsymbol{F}\boldsymbol{\mu}_0 + \boldsymbol{H}\boldsymbol{z}_1, \tag{3.9}$$
$$\boldsymbol{P}_1^0 = \mathrm{Var}[\boldsymbol{x}_1] = \boldsymbol{F}\boldsymbol{\Sigma_0}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T. \tag{3.10}$$

Moreover, for $t = 1, \ldots, n$

$$\boldsymbol{x}_{t+1}^t = \boldsymbol{F}\boldsymbol{x}_t^{t-1} + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t, \tag{3.11}$$
$$\boldsymbol{P}_{t+1}^t = \boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T - \boldsymbol{K}_t\boldsymbol{\Sigma}_t\boldsymbol{K}_t^T \tag{3.12}$$

where

$$\boldsymbol{\varepsilon}_t = \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}, \tag{3.13}$$
$$\boldsymbol{\Sigma}_t = \boldsymbol{A}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{\Sigma}_u, \tag{3.14}$$
$$\boldsymbol{K}_t = \left(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\right)\boldsymbol{\Sigma}_t^{-1} \tag{3.15}$$

are referred to as the innovations, the covariance of the innovations, and the Kalman gain, respectively. Additionally,

$$\boldsymbol{x}_t^t = \boldsymbol{x}_t^{t-1} + \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t, \tag{3.16}$$
$$\boldsymbol{P}_t^t = \boldsymbol{P}_t^{t-1} - \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{\Sigma}_t^{-1}\boldsymbol{A}\boldsymbol{P}_t^{t-1} \tag{3.17}$$

are called the filter values. [SS17, p. 322]

Proof.
First the initialisation given in Eqs. (3.9) and (3.10) follows immediately from the definition of the state space model.

Now the filter values Eqs. (3.16) and (3.17) are derived as these will be used to subsequently derive Eqs. (3.11) and (3.12). This part of the proof will be based on using the projection theorem for jointly normally distributed random vectors, Theorem A.8. Specifically, we will be interested in the distribution of $(\boldsymbol{x}_t^T, \boldsymbol{\varepsilon}_t^T)^T | \boldsymbol{y}_{t-1}, \ldots, \boldsymbol{y}_1$. Firstly, $\mathrm{Cov}(\boldsymbol{\varepsilon}_t, \boldsymbol{y}_s) = \boldsymbol{0}$ for $s < t$ which by the normality

assumption implies that the innovations are independent of the past observations. Secondly, we have that

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}_t|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1) &\overset{(a)}{=} \mathrm{Cov}(\boldsymbol{x}_t, \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &\overset{(b)}{=} \mathrm{Cov}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1}, \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{u}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &\overset{(c)}{=} \mathrm{Cov}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1}, \boldsymbol{A}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1}) + \boldsymbol{u}_t|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &\overset{(d)}{=} \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T \end{aligned} \tag{3.18}$$

where (a) is by definition of $\boldsymbol{\varepsilon}_t$, (b) holds by the orthogonality principle Proposition A.9, (c) arises from a factorisation, and (d) uses the independence of $\boldsymbol{u}_t$ on the past observations as well as the definition of $\boldsymbol{P}_t^{t-1}$. Hence,

$$\begin{bmatrix}\boldsymbol{x}_t\\ \boldsymbol{\varepsilon}_t\end{bmatrix} \,\Big|\, \boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1 \sim \mathcal{N}\Bigg(\begin{bmatrix}\boldsymbol{x}_t^{t-1}\\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{P}_t^{t-1} & \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\\ \boldsymbol{A}\boldsymbol{P}_t^{t-1} & \boldsymbol{\Sigma}_t\end{bmatrix} \Bigg). \tag{3.19}$$

Since the innovations are defined as $\boldsymbol{\varepsilon}_t = \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}$,

$$\boldsymbol{x}_t^t = \mathbb{E}[\boldsymbol{x}_t|\boldsymbol{y}_t, \dots, \boldsymbol{y}_1] = \mathbb{E}[\boldsymbol{x}_t|\boldsymbol{\varepsilon}_t, \boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1]. \tag{3.20}$$

Then an application of Theorem A.8 yields

$$\boldsymbol{x}_t^t = \boldsymbol{x}_t^{t-1} + \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t. \tag{3.21}$$

Again using Theorem A.8, the covariance filter value can be computed as

$$\begin{aligned}\boldsymbol{P}_t^t &= \mathbb{E}\big[(\boldsymbol{x}_t - \boldsymbol{x}_t^t)(\boldsymbol{x}_t - \boldsymbol{x}_t^t)^T\big],\\ &= \mathrm{Var}(\boldsymbol{x}_t|\boldsymbol{y}_t, \boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &= \mathrm{Var}(\boldsymbol{x}_t|\boldsymbol{\varepsilon}_t, \boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &= \boldsymbol{P}_t^{t-1} - \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{\Sigma}_t^{-1}\boldsymbol{A}\boldsymbol{P}_t^{t-1}. \end{aligned} \tag{3.22}$$

This proves the filter values Eqs. (3.16) and (3.17). [SS17, pp. 296-297]

Next the filter updates Eqs. (3.11) and (3.12) are derived. Following direct computations

$$\boldsymbol{x}_{t+1}^t = \boldsymbol{F}\boldsymbol{x}_t^t + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{G}\mathbb{E}[\boldsymbol{u}_t|\boldsymbol{y}_t, \dots, \boldsymbol{y}_1]. \tag{3.23}$$

The definition of the noise in the state equation implies that $\boldsymbol{u}_t$ is not independent of $\boldsymbol{y}_t$, hence it is of interest to use the projection theorem to determine the dependency. This can be done by considering the jointly normally distributed random vectors $(\boldsymbol{u}_t^T, \boldsymbol{\varepsilon}_t^T)^T|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1$. The mean of the random vector is trivially zero due to independence of the noise on the past observations and the covariance is

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{u}_t, \boldsymbol{\varepsilon}_t|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1) &= \mathrm{Cov}(\boldsymbol{u}_t, \boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &= \mathrm{Cov}(\boldsymbol{u}_t, \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{u}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &= \mathrm{Cov}(\boldsymbol{u}_t, \boldsymbol{u}_t + \boldsymbol{A}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})|\boldsymbol{y}_{t-1}, \dots, \boldsymbol{y}_1),\\ &= \boldsymbol{\Sigma}_u. \end{aligned} \tag{3.24}$$

This leads to the joint distribution

$$\begin{bmatrix}\boldsymbol{u}_t\\ \boldsymbol{\varepsilon}_t\end{bmatrix} \Big|\ \boldsymbol{y}_{t-1},\dots,\boldsymbol{y}_1 \sim \mathcal{N}\Bigg(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_u & \boldsymbol{\Sigma}_u\\ \boldsymbol{\Sigma}_u & \boldsymbol{\Sigma}_t\end{bmatrix}\Bigg). \tag{3.25}$$

Now by the same argument as in Eq. (3.20), it holds that and by using the projection theorem

$$\begin{aligned}\mathbb{E}[\boldsymbol{u}_t|\boldsymbol{y}_t,\dots,\boldsymbol{y}_1] &= \mathbb{E}[\boldsymbol{u}_t|\boldsymbol{\varepsilon}_t,\boldsymbol{y}_{t-1},\dots,\boldsymbol{y}_1],\\ &= \boldsymbol{\Sigma}_u\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t.\end{aligned} \tag{3.26}$$

Hence, by Eq. (3.23)

$$\begin{aligned}\boldsymbol{x}_{t+1}^t &= \boldsymbol{F}\boldsymbol{x}_t^t + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{G}\mathbb{E}[\boldsymbol{u}_t|\boldsymbol{y}_t,\dots,\boldsymbol{y}_1],\\ &= \boldsymbol{F}\big(\boldsymbol{x}_t^{t-1} + \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t\big) + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t,\\ &= \boldsymbol{F}\boldsymbol{x}_t^{t-1} + \boldsymbol{H}\boldsymbol{z}_{t+1} + \big(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\big)\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t,\\ &= \boldsymbol{F}\boldsymbol{x}_t^{t-1} + \boldsymbol{H}\boldsymbol{z}_{t+1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t.\end{aligned} \tag{3.27}$$

Finally, the variance update follows by using the derived expression for the state prediction and the definition of the state Eqs. (3.6) and (3.27)

$$\begin{aligned}\boldsymbol{P}_{t+1}^t &= \mathbb{E}\big[(\boldsymbol{x}_{t+1} - \boldsymbol{x}_{t+1}^t)(\boldsymbol{x}_{t+1} - \boldsymbol{x}_{t+1}^t)^T\big],\\ &= \mathbb{E}\big[\big(\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1}) + \boldsymbol{G}\boldsymbol{u}_t - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\big)\big(\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1}) + \boldsymbol{G}\boldsymbol{u}_t - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\big)^T\big],\\ &= \mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})^T\boldsymbol{F}^T] + \mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{u}_t^T\boldsymbol{G}^T] + \mathbb{E}[\boldsymbol{K}_t\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T]\\ &\quad - \mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] - \mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T]^T\\ &\quad - \mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] - \mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T]^T\end{aligned} \tag{3.28}$$

where it has been used that $\boldsymbol{u}_t$ and $(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})$ are independent which holds since $\boldsymbol{u}_t$ is a vector white noise process. Consider each of the different terms in Eq. (3.28)

$$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})^T\boldsymbol{F}^T] = \boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{F}^T, \tag{3.29}$$

$$\mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{u}_t^T\boldsymbol{G}^T] = \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T, \tag{3.30}$$

$$\mathbb{E}[\boldsymbol{K}_t\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] = \boldsymbol{K}_t\boldsymbol{\Sigma}_t\boldsymbol{K}_t, \tag{3.31}$$

$$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] = \boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T\boldsymbol{K}_t^T, \tag{3.32}$$

$$\mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] = \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{K}_t^T. \tag{3.33}$$

Then using the definition of the Kalman gain Eq. (3.15), it follows that

$$\begin{aligned}\mathbb{E}[\boldsymbol{F}(\boldsymbol{x}_t - \boldsymbol{x}_t^{t-1})\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] + \mathbb{E}[\boldsymbol{G}\boldsymbol{u}_t\boldsymbol{\varepsilon}_t^T\boldsymbol{K}_t^T] &= (\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u)\boldsymbol{K}_t^T,\\ &= \boldsymbol{K}_t\boldsymbol{\Sigma}_t\boldsymbol{K}_t^T.\end{aligned} \tag{3.34}$$

Since $\boldsymbol{K}_t\boldsymbol{\Sigma}_t\boldsymbol{K}_t^T$ is a symmetric matrix, this implies that

$$\boldsymbol{P}_{t+1}^t = \boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T - \boldsymbol{K}_t\boldsymbol{\Sigma}_t\boldsymbol{K}_t^T \tag{3.35}$$

which concludes the proof. [SS17, pp. 296-297] ■

The result in Proposition 3.1 provides an iterative scheme, based on updating the beliefs as new datapoints are observed, for computing minimum mean square error one-step ahead forecasts for both the observations and the states in case the normality assumption is met. Moreover, the error variances for both the state and observation forecasts are also computed. If the normality assumption is not met, then the result is limited to the linear minimum mean square error results. [SS17, pp. 297-298]

In Corollary 3.2, the equations for performing τ-ahead forecasts where $\tau \geq 1$ with the state space model in Eq. (3.6) is given. This result is derived in a similar manner to Proposition 3.1, however the derivations are simpler and therefore the proof will be omitted.

Corollary 3.2 (τ-Ahead Forecast)
Consider the state-space model in Eq. (3.6). The minimum mean square error τ-ahead forecast for $\tau \geq 1$ of the state is

$$\boldsymbol{x}_{n+\tau}^{n} = \boldsymbol{F}\boldsymbol{x}_{n+\tau-1}^{n} + \boldsymbol{H}\boldsymbol{z}_{n+\tau} \tag{3.36}$$

with error covariance

$$\boldsymbol{P}_{n+\tau}^{n} = \boldsymbol{F}\boldsymbol{P}_{n+\tau-1}^{n}\boldsymbol{F}^{T} + \boldsymbol{G}\boldsymbol{\Sigma}_{u}\boldsymbol{G}^{T}. \tag{3.37}$$

The τ-ahead forecast of the observation is

$$\boldsymbol{y}_{n+\tau}^{n} = \boldsymbol{A}\boldsymbol{x}_{n+\tau}^{n} \tag{3.38}$$

with innovations given by

$$\boldsymbol{\varepsilon}_{n+\tau}^{n} = \boldsymbol{y}_{n+\tau} - \boldsymbol{y}_{n+\tau}^{n} \tag{3.39}$$

and the covariance of the innovations are

$$\boldsymbol{\Sigma}_{n+\tau}^{n} = \boldsymbol{A}\boldsymbol{P}_{n+\tau}^{n}\boldsymbol{A}^{T} + \boldsymbol{\Sigma}_{u}. \tag{3.40}$$

3.3 Maximum Likelihood Estimation

Let $\boldsymbol{\theta}$ be a vector consisting of the unknown parameters for the state space model in Eq. (3.6). Similarly to the log-likelihood for the class of VARMA models in Eq. (2.42), the log-likelihood for the state space model in Eq. (3.6) is

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\sum_{t=1}^{n}\log\left(|\boldsymbol{\Sigma}_{t}(\boldsymbol{\theta})|\right) - \frac{1}{2}\sum_{t=1}^{n}\boldsymbol{\varepsilon}_{t}(\boldsymbol{\theta})^{T}\boldsymbol{\Sigma}_{t}^{-1}(\boldsymbol{\theta})\boldsymbol{\varepsilon}_{t}(\boldsymbol{\theta}) \tag{3.41}$$

where $\boldsymbol{\Sigma}_{t}(\boldsymbol{\theta})$ and $\boldsymbol{\varepsilon}_{t}(\boldsymbol{\theta})$ are given in Eqs. (3.13) and (3.14) and the dependence on the parameters is explicitly denoted. To estimate the parameters a Newton-like method,

as described in Appendix B, can be used. In this context, a minimisation problem is considered which motivates the use of the negative log-likelihood as the objective function

$$J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) = \frac{1}{2}\sum_{t=1}^{n} \log\left(|\boldsymbol{\Sigma}_t(\boldsymbol{\theta})|\right) + \frac{1}{2}\sum_{t=1}^{n} \boldsymbol{\varepsilon}_t(\boldsymbol{\theta})^T \boldsymbol{\Sigma}_t^{-1}(\boldsymbol{\theta})\boldsymbol{\varepsilon}_t(\boldsymbol{\theta}). \tag{3.42}$$

The estimation procedure can now proceed by successively running Kalman filtering to compute the innovations and the innovation covariance followed by a Newton-step with $J(\boldsymbol{\theta})$ as the objective function. The method is initialised by an initial parameter estimate and is ended when a stopping criterion is met.

When using a Newton-like method to numerically optimise the objective function, the gradient and the inverse Hessian are needed. In Proposition 3.3, the gradient of Eq. (3.42) is derived. For convenience of notation, we will omit $\boldsymbol{\theta}$ and let $\partial_i f = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$.

Proposition 3.3 (Gradient of the Objective Function)
The partial derivative of the objective function Eq. (3.42) in the ith parameter is [SS17, p. 381]

$$g_i(\boldsymbol{\theta}) = \partial_i J = \sum_{t=1}^{n} \left(\boldsymbol{\varepsilon}_t^T \boldsymbol{\Sigma}_t^{-1} \partial_i \boldsymbol{\varepsilon}_t - \frac{1}{2}\boldsymbol{\varepsilon}_t^T \boldsymbol{\Sigma}_t^{-1} \partial_i \boldsymbol{\Sigma}_t \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\varepsilon}_t + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_t^{-1}\partial_i \boldsymbol{\Sigma}_t\right)\right) \tag{3.43}$$

where for $t = 1, \ldots, n$

$$\partial_i \boldsymbol{\varepsilon}_t = -\boldsymbol{A}\partial_i \boldsymbol{x}_t^{t-1}, \tag{3.44}$$
$$\partial_i \boldsymbol{\Sigma}_t = \boldsymbol{A}\partial_i \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \partial_i \boldsymbol{\Sigma}_u, \tag{3.45}$$
$$\partial_i \boldsymbol{K}_t = \left(\partial_i \boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{F}\partial_i \boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \partial_i \boldsymbol{G}\boldsymbol{\Sigma}_u + \boldsymbol{G}\partial_i \boldsymbol{\Sigma}_u - \boldsymbol{K}_t \partial_i \boldsymbol{\Sigma}_t\right)\boldsymbol{\Sigma}_t^{-1} \tag{3.46}$$

and for $t = 2, \ldots, n$

$$\partial_i \boldsymbol{x}_t^{t-1} = \partial_i \boldsymbol{F}\boldsymbol{x}_{t-1}^{t-2} + \boldsymbol{F}\partial_i \boldsymbol{x}_{t-1}^{t-2} + \partial_i \boldsymbol{H}\boldsymbol{z}_t + \partial_i \boldsymbol{K}_{t-1}\boldsymbol{\varepsilon}_{t-1} + \boldsymbol{K}_{t-1}\partial_i \boldsymbol{\varepsilon}_{t-1}, \tag{3.47}$$
$$\begin{aligned}\partial_i \boldsymbol{P}_t^{t-1} = {} & \partial_i \boldsymbol{F}\boldsymbol{P}_{t-1}^{t-2}\boldsymbol{F}^T + \boldsymbol{F}\partial_i \boldsymbol{P}_{t-1}^{t-2}\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{P}_{t-1}^{t-2}\partial_i \boldsymbol{F}^T \\ & + \partial_i \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\partial_i \boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\partial_i \boldsymbol{G}^T \\ & - \partial_i \boldsymbol{K}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{K}_{t-1}^T - \boldsymbol{K}_{t-1}\partial_i \boldsymbol{\Sigma}_{t-1}\boldsymbol{K}_{t-1}^T - \boldsymbol{K}_{t-1}\boldsymbol{\Sigma}_{t-1}\partial_i \boldsymbol{K}_{t-1}^T.\end{aligned} \tag{3.48}$$

To initialise the recursion, we have

$$\partial_i \boldsymbol{x}_1^0 = \partial_i \boldsymbol{F}\boldsymbol{\mu}_0 + \partial_i \boldsymbol{H}\boldsymbol{z}_1, \tag{3.49}$$
$$\partial_i \boldsymbol{P}_1^0 = \partial_i \boldsymbol{F}\boldsymbol{\Sigma}_0\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{\Sigma}_0\partial_i \boldsymbol{F}^T + \partial_i \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\partial_i \boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\partial_i \boldsymbol{G}^T. \tag{3.50}$$

Proof.
We begin the proof by deriving Eq. (3.43). By the product rule and the chain rule for partial differentiation

$$\begin{aligned} g_i(\boldsymbol{\theta}) &= \frac{1}{2}\sum_{t=1}^{n}\Big(2\boldsymbol{\varepsilon}_t^T\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\varepsilon}_t + \boldsymbol{\varepsilon}_t^T\partial_i\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t + \frac{1}{|\boldsymbol{\Sigma}_t|}\partial_i|\boldsymbol{\Sigma}_t|\Big), \\ &= \sum_{t=1}^{n}\Big(\boldsymbol{\varepsilon}_t^T\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\varepsilon}_t - \frac{1}{2}\boldsymbol{\varepsilon}_t^T\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\varepsilon}_t + \frac{1}{2}\mathrm{tr}\big(\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\Sigma}_t\big)\Big) \end{aligned} \tag{3.51}$$

where tr denotes the trace-operator and where we have used the matrix identities

$$\partial_i\boldsymbol{\Sigma}_t^{-1} = -\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_t^{-1}, \tag{3.52}$$

$$\partial_i|\boldsymbol{\Sigma}_t| = |\boldsymbol{\Sigma}_t|\mathrm{tr}\big(\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\Sigma}_t\big). \tag{3.53}$$

To derive Eqs. (3.44) to (3.50), the Kalman filter equations Eqs. (3.9) to (3.15) are used. Consider first the partial derivative of the innovations for $t = 1, \ldots, n$

$$\partial_i\boldsymbol{\varepsilon}_t = \partial_i\big(\boldsymbol{y}_t - \boldsymbol{A}\boldsymbol{x}_t^{t-1}\big) = -\boldsymbol{A}\partial_i\boldsymbol{x}_t^{t-1}. \tag{3.54}$$

The partial derivative of the innovation covariance for $t = 1, \ldots, n$ is

$$\partial_i\boldsymbol{\Sigma}_t = \partial_i\big(\boldsymbol{A}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{\Sigma}_u\big) = \boldsymbol{A}\partial_i\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \partial_i\boldsymbol{\Sigma}_u. \tag{3.55}$$

The partial derivative of the Kalman gain for $t = 1, \ldots, n$ is

$$\begin{aligned} \partial_i\boldsymbol{K}_t &= \partial_i\Big(\big(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\big)\boldsymbol{\Sigma}_t^{-1}\Big), \\ &= \partial_i\big(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\big)\boldsymbol{\Sigma}_t^{-1} + \big(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\big)\partial_i\boldsymbol{\Sigma}_t^{-1}, \\ &= \big(\partial_i\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{F}\partial_i\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \partial_i\boldsymbol{G}\boldsymbol{\Sigma}_u + \boldsymbol{G}\partial_i\boldsymbol{\Sigma}_u\big)\boldsymbol{\Sigma}_t^{-1} \\ &\quad - \big(\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\big)\boldsymbol{\Sigma}_t^{-1}\partial_i\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_t^{-1} \\ &= \big(\partial_i\boldsymbol{F}\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \boldsymbol{F}\partial_i\boldsymbol{P}_t^{t-1}\boldsymbol{A}^T + \partial_i\boldsymbol{G}\boldsymbol{\Sigma}_u + \boldsymbol{G}\partial_i\boldsymbol{\Sigma}_u - \boldsymbol{K}_t\partial_i\boldsymbol{\Sigma}_t\big)\boldsymbol{\Sigma}_t^{-1}. \end{aligned} \tag{3.56}$$

The partial derivative of the state forecast for $t = 2, \ldots, n$ is

$$\begin{aligned} \partial_i\boldsymbol{x}_t^{t-1} &= \partial_i\big(\boldsymbol{F}\boldsymbol{x}_{t-1}^{t-2} + \boldsymbol{H}\boldsymbol{z}_t + \boldsymbol{K}_{t-1}\boldsymbol{\varepsilon}_{t-1}\big), \\ &= \partial_i\boldsymbol{F}\boldsymbol{x}_{t-1}^{t-2} + \boldsymbol{F}\partial_i\boldsymbol{x}_{t-1}^{t-2} + \partial_i\boldsymbol{H}\boldsymbol{z}_t + \partial_i\boldsymbol{K}_{t-1}\boldsymbol{\varepsilon}_{t-1} + \boldsymbol{K}_{t-1}\partial_i\boldsymbol{\varepsilon}_{t-1} \end{aligned} \tag{3.57}$$

while for $t = 1$ it is

$$\partial_i\boldsymbol{x}_1^0 = \partial_i\big(\boldsymbol{F}\boldsymbol{\mu}_0 + \boldsymbol{H}\boldsymbol{z}_1\big) = \partial_i\boldsymbol{F}\boldsymbol{\mu}_0 + \partial_i\boldsymbol{H}\boldsymbol{z}_1. \tag{3.58}$$

The partial derivative of the state forecast covariance for $t = 2, \ldots, n$ is

$$\begin{aligned} \partial_i\boldsymbol{P}_t^{t-1} &= \partial_i\big(\boldsymbol{F}\boldsymbol{P}_{t-1}^{t-2}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T - \boldsymbol{K}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{K}_{t-1}^T\big), \\ &= \partial_i\boldsymbol{F}\boldsymbol{P}_{t-1}^{t-2}\boldsymbol{F}^T + \boldsymbol{F}\partial_i\boldsymbol{P}_{t-1}^{t-2}\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{P}_{t-1}^{t-2}\partial_i\boldsymbol{F}^T \\ &\quad + \partial_i\boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\partial_i\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\partial_i\boldsymbol{G}^T \\ &\quad - \partial_i\boldsymbol{K}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{K}_{t-1}^T - \boldsymbol{K}_{t-1}\partial_i\boldsymbol{\Sigma}_{t-1}\boldsymbol{K}_{t-1}^T - \boldsymbol{K}_{t-1}\boldsymbol{\Sigma}_{t-1}\partial_i\boldsymbol{K}_{t-1}^T \end{aligned} \tag{3.59}$$

while for $t = 1$ it is

$$\begin{aligned}\partial_i \boldsymbol{P}_1^0 &= \partial_i\big(\boldsymbol{F}\boldsymbol{\Sigma}_0\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T\big),\\ &= \partial_i\boldsymbol{F}\boldsymbol{\Sigma}_0\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{\Sigma}_0\partial_i\boldsymbol{F}^T + \partial_i\boldsymbol{G}\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\partial_i\boldsymbol{\Sigma}_u\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{\Sigma}_u\partial_i\boldsymbol{G}^T. \quad (3.60)\end{aligned}$$

This concludes the proof. ■

Broyden-Fletcher-Goldfarb-Shanno (BFGS) is a Newton-like method which uses the analytical expression for the gradient to iteratively estimate the inverse Hessian using Eq. (B.34). With the gradient and the expression for the inverse Hessian estimated using BFGS, a Newton-step can be computed as in Eq. (B.6). An algorithm outlining the parameter estimation procedure is given in Algorithm 1.

Algorithm 1 Parameter Estimation

Input: Endogenous variables $\boldsymbol{y}_t$, exogenous variables $\boldsymbol{z}_t$, and stopping criterion parameters.

1: Set initial parameter estimate $\boldsymbol{\theta}^{(0)}$. ▷ Eq. (2.49)
2: $j = 0$.
3: **while** Stopping criterion **is** False **do**
4: Compute $\boldsymbol{\varepsilon}_t(\boldsymbol{\theta}^{(j)})$ and $\boldsymbol{\Sigma}_t(\boldsymbol{\theta}^{(j)})$ using Kalman filtering. ▷ Proposition 3.1
5: Compute the gradient $\boldsymbol{g}(\boldsymbol{\theta}^{(j)})$. ▷ Proposition 3.3
6: Update the estimate of the inverse Hessian $\hat{\boldsymbol{H}}^{(j)}_{\text{BFGS}}$. ▷ Eq. (B.34)
7: Update the parameter estimate $\boldsymbol{\theta}^{(j+1)}$. ▷ Eqs. (B.4) and (B.6)
8: $j = j + 1$.
9: **end while**

Output: $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(j)}$.

4. Time Series Decomposition

Recall from Chapter 2, that a time series $\{y_t\}_{t\in\mathbb{Z}}$ is a realisation of a stochastic process. If this time series has been drawn from a WSS process, we can determine the energy-frequency distribution by taking the Fourier transform of the autocorrelation function. However, if the stochastic process is non-stationary, then the autocorrelation function changes over time resulting in an energy-frequency distribution which changes with respect to time. In this case, a short-time Fourier transform or a Wavelet transform could be used [VKG14, pp. 638-650], however these methods are still limited in their time-frequency resolution as stated in Heisenberg's uncertainty principle [Fol92, p. 232]. Instead another decomposition method which can be used for non-stationary time series is the empirical mode decomposition (EMD). This method operates purely in the time domain. The EMD decomposes the time series $\{y_t\}_{t\in\mathbb{Z}}$ to a collection of intrinsic mode functions (IMFs) and a residual. In this project, the EMD will be used to divide the signal sub components which will then be fed to two different models based on their complexity. Several articles have found improved results in their time series forecasting by using the EMD before applying their model, see for instance [LDB21; XHY19; SV13; Zhe+13].

4.1 Preliminaries

Before introducing the EMD, we will present the Hilbert transform, instantaneous frequency, and cubic spline interpolation (CSI).

4.1.1 The Hilbert Transform and Instantaneous Frequency

When working with a non-stationary signal, we cannot expect to observe a full wavelength over which we can determine a frequency since the energy-frequency distribution might vary between each time. Hence, in practice we will use the so-called instantaneous frequency which we will present in the following. However, in order to do so we will first define the notion of analytical signals.

Definition 4.1 (Analytical Signal)
Let $z_x : \mathbb{R} \to \mathbb{C}$ be a complex-valued function with Fourier transform $Z_x(\omega)$. Then $z_x(t)$ is called an analytical signal if $Z_x(\omega) = 0$ when $\omega < 0$. [Smi07]

If we consider an analytical signal $z_x : \mathbb{R} \to \mathbb{C}$, then the instantaneous phase is determined by

$$\theta(t) = \arg\{z_x(t)\}. \tag{4.1}$$

The instantaneous frequency is determined by calculating the temporal rate of the so-called instantaneous phase, i.e. [Hua+98, pp. 911-912]

$$\omega(t) = \frac{d\theta(t)}{dt}. \tag{4.2}$$

If we instead consider a function $x : \mathbb{R} \to \mathbb{C}$ which is not an analytical signal, then the instantaneous frequency may be negative and thereby have no physical meaning. Hence, it is of interest to create an analytical signal z_x from x without changing the distribution of the magnitude spectrum of the non-negative frequencies. One way of doing so is by using the Hilbert transform which is defined in Definition 4.3. Before proceeding to the definition of the Hilbert transform, the notion of Cauchy principle value must be introduced.

Definition 4.2 (Cauchy Principal Value)
Let $[a, b]$ be a real interval and let $x : [a, b] \to \mathbb{C}$. If x is unbounded near some interior point $c \in [a, b]$, the integral over $[a, b]$ does not always exist. However, if the two limits

$$\lim_{\varepsilon \to 0} \int_a^{c-\varepsilon} x(t)dt \quad \text{and} \quad \lim_{\varepsilon \to 0} \int_{c+\varepsilon}^{b} x(t)dt \tag{4.3}$$

exist, then their sum is called the improper integral of $x(t)$ over $[a, b]$ and is denoted as usual

$$\int_a^b x(t)dt. \tag{4.4}$$

However, even if these limits do not exist, then if the limit

$$\lim_{\varepsilon \to 0^+} \Big(\int_a^{c-\varepsilon} x(t)dt + \int_{c+\varepsilon}^{b} x(t)dt \Big) \tag{4.5}$$

exists, it is called the principal value integral of x on $[a, b]$ and it is denoted by [Joh12, p. 2]

$$p.v. \int_a^b x(t)dt. \tag{4.6}$$

Definition 4.3 (Hilbert Transform)
The Hilbert transform $\hat{x} : \mathbb{R} \to \mathbb{C}$ of a function $x : \mathbb{R} \to \mathbb{C}$ is defined for all t by

$$\hat{x}(t) = \mathcal{H}\{x\}(t) = x(t) * \frac{1}{\pi t} = \frac{1}{\pi} p.v. \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau \tag{4.7}$$

when the integral exists as a principle value [Joh12, p. 1].

In the Hilbert transform, the Cauchy principle value is used when $\tau = t$. The Hilbert transform can be seen as the response of a linear filter with impulse response $h(t) = \frac{1}{\pi t}$ to some signal $x(t)$. Using the Hilbert transform, we can obtain an analytical signal from a non-analytical signal x, without changing the distribution of the magnitude spectrum for the non-negative frequencies. This will be shown in the following.

The effect of applying the Hilbert transform can most easily be seen in the frequency domain. Therefore, we will define the Fourier transform.

Definition 4.4 (Fourier Transform)
Let $x : \mathbb{R} \to \mathbb{C}$ be a function in L^1. Then its Fourier transform is given by

$$X(\omega) = \mathcal{F}\{x\}(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt. \tag{4.8}$$

Furthermore, for $X(\omega) \in L^1$ the inverse Fourier transform is given by [Fol92, p. 213 & 218]

$$x(t) = \mathcal{F}^{-1}\{X\}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} dt. \tag{4.9}$$

Another version of this definition exists for $x \in L^2$, but this definition will not be presented in this project. For a presentation of the Fourier transform in L^2 see [Tes10, p. 187].

As was seen earlier, the Hilbert transform can be expressed as

$$\hat{x}(t) = h(t) * x(t). \tag{4.10}$$

The convolution theorem for the Fourier transform [Fol92, p. 214] then implies that

$$\hat{x}(t) = \mathcal{F}^{-1}\big\{H(\omega)X(\omega)\big\}. \tag{4.11}$$

The Fourier transform of $h(t) = \frac{1}{\pi t}$ is $H(\omega) = -j\text{sgn}(\omega)$ where $\text{sgn}(\omega)$ is the sign function [Joh12, pp. 10-11]

$$\text{sgn}(\omega) = \begin{cases} 1, & \omega > 0 \\ 0, & \omega = 0 \\ -1, & \omega < 0. \end{cases} \tag{4.12}$$

This implies that

$$\hat{x}(t) = \mathcal{F}^{-1}\big\{ -j\text{sgn}(\omega)X(\omega)\big\}. \tag{4.13}$$

Now we will define a function $Z_x(\omega)$ which is zero for all negative frequencies, $X(\omega)$ for $\omega = 0$, and $2X(\omega)$ for all positive frequencies

$$Z_x(\omega) = X(\omega) + \text{sgn}(\omega)X(\omega). \tag{4.14}$$

The inverse Fourier transform of $Z_x(\omega)$ can be found as

$$\begin{aligned} z_x(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} Z_x(\omega)e^{j\omega t}d\omega, \\ &= \frac{1}{2\pi}\Big(\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega + \int_{-\infty}^{\infty} \text{sgn}(\omega)X(\omega)e^{j\omega t}d\omega\Big), \\ &= x(t) + \frac{1}{2\pi}\int_{-\infty}^{\infty} \text{sgn}(\omega)X(\omega)e^{j\omega t}d\omega, \\ &= x(t) + j\frac{1}{2\pi}\int_{-\infty}^{\infty} -j\text{sgn}(\omega)X(\omega)e^{j\omega t}d\omega, \\ &= x(t) + j\hat{x}(t). \end{aligned} \tag{4.15}$$

Hence, $z_x(t)$ is an analytical signal derived from $x(t)$ using the Hilbert transform.

The Hilbert transform corresponds to a multiplication by $-j\text{sgn}(\omega)$ in the frequency domain as seen in Eq. (4.13). For positive frequencies, this is $-j = \exp(-j\pi/2)$, while for negative frequencies it is $j = \exp(j\pi/2)$. Hence, the Hilbert transform has the effect of shifting the phase of the negative frequency components of $x(t)$ by $\frac{\pi}{2}$ radians and the positive frequency components by $-\frac{\pi}{2}$ radians. Then $j\mathcal{H}\{x\}(t)$ has the effect of restoring the positive frequency components while shifting the negative components an additional $\frac{\pi}{2}$ radians. This results in a function with no negative frequency components.

Applying the Hilbert transformation twice, the positive and negative frequency components are shifted by π and $-\pi$ radians, respectively. This results in the negated function, i.e. $\mathcal{H}\{\mathcal{H}\{x\}\}(t) = -x(t)$. This motivates the following definition of the inverse Hilbert transform

Definition 4.5 (Inverse Hilbert Transform)
Given a Hilbert transformed function $\hat{x} : \mathbb{R} \to \mathbb{C}$, we can recover the function $x(t)$ by the inverse Hilbert transform given by

$$x(t) = -\mathcal{H}\{\hat{x}\}(t) = -\frac{1}{\pi}p.v.\int_{-\infty}^{\infty}\frac{\hat{x}(\tau)}{t-\tau}d\tau. \tag{4.16}$$

In Eq. (4.2), the instantaneous frequency is a single value for each time t. Thus, in order to obtain an instantaneous frequency it is required that at each time t the

frequency can be determined by a single frequency component. This requirement is called the narrowband requirement. [Hua+98, p. 912]

In general, another condition than the narrowband requirement have to be applied to a function in order to obtain a meaningful instantaneous frequency. Specifically, the function has to contain only positive frequency components [Hua+98, p. 913]. This means that the function has to be an analytical signal and this analytical signal can be obtained using the Hilbert transform.

In practice, we will be considering time series data which is discrete in time $\{y_t\}_{t=1}^n$. We note that the theory of the Hilbert transform and analytical signals has a natural extension to the discrete time domain [SS93, pp. 1867-1868]. The narrowband requirement can in practice be approximately ensured by imposing a global and a local requirement. The global requirement is that the number of extrema and zero crossings at most differs by one. Let

$$g_t = y_t - y_{t-1} \tag{4.17}$$

for $t = 2, \ldots, n$. Then the number of extrema is given by

$$n_e = \sum_{t=3}^{n} \mathbb{1}[g_t g_{t-1} < 0]. \tag{4.18}$$

Additionally, the number of zero crossings is given by

$$z_c = \sum_{t=2}^{n} \mathbb{1}[y_t y_{t-1} < 0]. \tag{4.19}$$

The global requirement is then

$$|n_e - z_c| \leq 1. \tag{4.20}$$

The local requirement is that the function is symmetric locally with respect to zero mean. For $t = 1, \ldots, n$ let f_t be the probability density function of y_t. If $\mathbb{E}[y_t] = 0$ and $f_t(-\delta) = f_t(\delta)$ for all $\delta \in \mathbb{R}$, then y_t is symmetric locally with respect to zero mean and thereby fulfils the local requirement. In practice, the probability density functions f_t are not known and hence this criterion cannot be checked. Instead, an approximation of this criterion is employed. This approximation of the local requirement is based on using so-called upper and lower envelopes. An upper envelope e_t^u of a time series y_t is a sequence which is greater than y_t except at the local maxima of y_t where the envelope is equal to the time series. Similarly, a lower envelope e_t^l is less than y_t except at the local minima of y_t where the envelope is equal to the time series. Thereby, the value of y_t at each point t is bounded by the value of the lower and upper envelopes. An example of upper and lower envelopes can be seen in Fig. 4.1b. The local requirement is imposed by requiring that

$$m_t = \left| \frac{e_t^u + e_t^l}{2} \right| \leq \varepsilon \tag{4.21}$$

for some $\varepsilon > 0$. The notion of how envelopes can be formed will be treated in further detail in Section 4.1.2.

The global requirement is a traditional narrowband requirement, while the local requirement is necessary to avoid unwanted instantaneous frequency content caused by asymmetric waveforms. [Hua+98, pp. 913-914]

An IMF is a type of function which approximately fulfils the narrowband requirements in the aforementioned sense.

Definition 4.6 (Intrinsic Mode Function)
Let $\{y_t\}_{t=1}^n$ be a discrete time series with n_e extrema and z_c zero crossing and let it be bounded by an upper envelope e_t^u and a lower envelope e_t^l, defined by the local maxima and minima of y_t, respectively. Then y_t is an IMF if

$$|n_e - z_c| \leq 1 \tag{4.22}$$

and if for $t = 1, \ldots, n$

$$m_t = \left|\frac{e_t^u + e_t^l}{2}\right| \leq \varepsilon \tag{4.23}$$

for some $\varepsilon > 0$.

An IMF can vary both in frequency and amplitude and can in fact be non-stationary. [Hua+98, p. 917]

4.1.2 Cubic Spline Interpolation

When determining an IMF, we have to create envelopes and one method for creating envelopes is CSI which is the method used in the EMD. Before introducing CSI, we will define a spline.

Definition 4.7 (Spline)
Consider the interval $[a, b]$ and let this interval be covered by k disjoint subintervals, i.e. $I_i = [t_{i-1}, t_i)$ for $i = 1, \ldots, k-1$ and an interval $I_k = [t_{k-1}, t_k]$ such that

$$[a, b] = \bigcup_{i=1}^{k} I_i \tag{4.24}$$

and $I_i \cap I_j = \emptyset$ for $i \neq j$. On each subinterval, define a polynomial $P_i : I_i \to \mathbb{R}$. A spline $S : [a, b] \to \mathbb{R}$ is then defined by [Sch07, p. 5]

$$S(t) = \begin{cases} P_1(t), & t_0 \leq t < t_1, \\ P_2(t), & t_1 \leq t < t_2, \\ \vdots & \vdots \\ P_k(t), & t_{k-1} \leq t \leq t_k. \end{cases} \tag{4.25}$$

In CSI, we consider a set of datapoints $\{(t_i, y_i)\}_{i=0}^{k}$ and create a spline, S, where each polynomial is of degree three, i.e.

$$P_i(t) = \alpha_i t^3 + \beta_i t^2 + \gamma_i t + \delta_i \tag{4.26}$$

for $i = 1, 2, \ldots, k$. The CSI has to conform to the following criteria:

i. S has to interpolate the datapoints, i.e. $S(t_i) = y_i$.

ii. $S \in C^2$ on the interval $[a, b]$.

When applying CSI, the coefficients of the polynomials in Eq. (4.26) should be determined. There is a total of $4k$ coefficients which are the four parameters for each of the k polynomials. Using criterion i., the domain of P_i is extended to include both endpoints t_{i-1} and t_i and thereby yielding the $2k$ equations

$$P_i(t_{i-1}) = y_{i-1}, \tag{4.27}$$

$$P_i(t_i) = y_i. \tag{4.28}$$

Furthermore, enforcing criterion ii. is done by requiring that the first and second order derivatives of P_i and P_{i+1} for $i = 1, 2 \ldots, k$ in the observed points are equal, i.e.

$$\frac{dP_i}{dt}(t_i) = \frac{dP_{i+1}}{dt}(t_i), \tag{4.29}$$

$$\frac{d^2P_i}{dt^2}(t_i) = \frac{d^2P_{i+1}}{dt^2}(t_i). \tag{4.30}$$

This gives us an additional $2(k-1)$ equations from which we can determine parameters. Thus, we have a total of $4k-2$ equations and are only two equations short of obtaining a linear system of equations with a unique solution. These last two equations are usually chosen using a boundary criteria. One such boundary criterion is

$$\frac{d^2P_1}{dt^2}(t_0) = 0, \tag{4.31}$$

$$\frac{d^2P_k}{dt^2}(t_k) = 0. \tag{4.32}$$

These are called the natural boundary conditions. This choice of boundary condition ensures that the CSI extends as a line outside the endpoints. Another boundary criterion is

$$\frac{dP_1}{dt}(t_0) = 0, \tag{4.33}$$

$$\frac{dP_k}{dt}(t_k) = 0 \tag{4.34}$$

which is called the clamped boundary conditions. These ensure that the slopes are zero at the end points. When we have chosen the boundary conditions, we have a set of $4k$ equations with $4k$ unknowns which we can solve as a system of linear equations to obtain a CSI. [YM21, pp. 76-77] [ML, pp. 1-5]

4.2 Empirical Mode Decomposition

Understanding the notion of instantaneous frequency, IMFs, and CSI, we are ready to introduce the EMD. Let $\{y_t\}_{t=1}^n$ be an observed discrete time series. Then by applying the EMD, the time series $\{y_t\}_{t=1}^n$ is decomposed into IMFs and a residual without making any assumptions regarding stationarity of $\{y_t\}_{t=1}^n$. Applying the EMD, the decomposition obtained remains in the time domain from which the original time series can be reconstructed.

The procedure of extracting the IMFs is called the sifting process and works as follows. Initially, identify all the extrema in the data. Then create an upper envelope by connecting the local maxima by a CSI described in Section 4.1.2. A lower envelope is created similarly using the local minima. Having done this, the upper and lower envelopes enclose the data. Subsequently, the mean of the envelopes, denoted $m_t^{(1,0)}$, is found and the difference between $m_t^{(1,0)}$ and y_t is the first component

$$h_t^{(1,0)} = y_t - m_t^{(1,0)}. \tag{4.35}$$

The sifting procedure is depicted in Fig. 4.1.

Ideally, $h_t^{(1,0)}$ should be an IMF. However, this is often not the case due to overshoots and undershoots of the envelopes compared to the data which can result in new extrema, shifts, or exaggerate existing extrema. Another problem is that the envelope mean may differ from the true mean. Thus, it is likely that $h_t^{(1,0)}$ is not an IMF and if this is the case, then the process is repeated and $h_t^{(1,0)}$ is treated as the data. This process is repeated k times until we obtain an $h_t^{(1,k)}$ as

$$h_t^{(1,k)} = h_t^{(1,k-1)} - m_t^{(1,k)} \tag{4.36}$$

which is an IMF and then $h_t^{(1,k)}$ is denoted as $c_t^{(1)}$. Next the first residual $r_t^{(1)}$ is computed as

$$r_t^{(1)} = y_t - c_t^{(1)}. \tag{4.37}$$

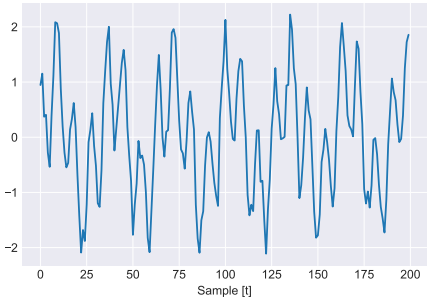
After the first iteration of the sifting procedure, $r_t^{(1)}$ is treated as the new data and the sifting process is applied to $r_t^{(1)}$ to obtain the second IMF $c_t^{(2)}$ and a new residual $r_t^{(2)}$ is then determined as

$$r_t^{(2)} = r_t^{(1)} - c_t^{(2)}. \tag{4.38}$$

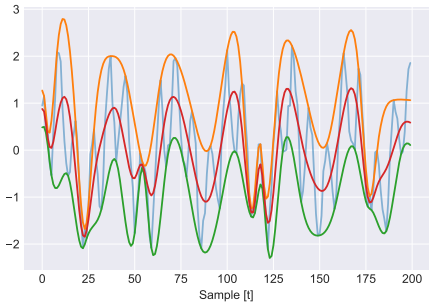
This process is continued until iteration $\nu \in \mathbb{N}$ when the residual $r_t^{(\nu)}$ becomes monotone, when $\max(r_t^{(\nu)}) - \min(r_t^{(\nu)})$ is less than some threshold, or when $\sum_{t=1}^n |r_t^{(\nu)}|$ is less than some threshold. Then, the signal has been decomposed as

$$y_t = \sum_{i=1}^{\nu} c_t^{(i)} + r_t^{(\nu)}. \tag{4.39}$$

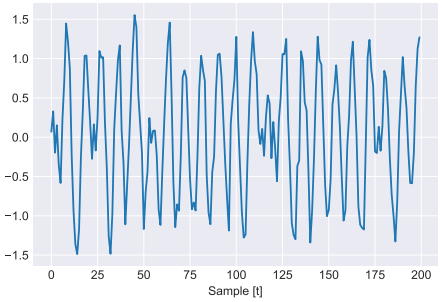
When implementing the EMD algorithm, stopping criteria are imposed on the sifting process. First of all, it is checked if h_t is an IMF as described in Definition 4.6. In



(a) The original signal defined by $y_t = \sin\left(t\frac{4\pi}{199}\right) + \cos\left(t\frac{14\pi}{199}\right) + 0.2\varepsilon_t$ with $\varepsilon_t \overset{iid.}{\sim} \mathcal{N}(0,1)$ for $t = 0, \ldots, 199$.



(b) An upper envelope (orange) and lower envelope (green) have been added alongside the mean of the envelopes (red).



(c) The first component $h_t^{(1,0)}$.

Figure 4.1: The sifting procedure.

conjunction with this, other stopping criteria are used. One such criterion is based on two thresholds κ_1 and κ_2. First the so-called mode amplitude is computed

$$\alpha_t = \frac{e_t^u - e_t^l}{2} \tag{4.40}$$

and then the evaluation sequence is computed as

$$\iota_t = \left|\frac{m_t}{\alpha_t}\right|. \tag{4.41}$$

The sifting process is iterated until $\iota_t < \kappa_1$ for some percentage of the discrete time points $(1 - \lambda)$ while $\iota_t < \kappa_2$ for the rest of the discrete time points. Typical values are $\lambda = 0.05$, $\kappa_1 = 0.05$, and $\kappa_2 = 10\kappa_1$. [RFG03, p. 3]

Consider the case of determining the ith IMF. Another stopping criterion for the sifting process is based on the deviation between two consecutive components normalised with respect to the previously computed component given as

$$D = \sum_{t=1}^{n} \left(\frac{\left|h_t^{(i,j-1)} - h_t^{(i,j)}\right|^2}{\left(h_t^{(i,j-1)}\right)^2} \right) \tag{4.42}$$

for $j = 1, 2, \ldots, k$ and this deviation is compared to some threshold which is usually set to 0.2. [Hua+98, p. 920]

The sifting procedure and the EMD are summarised in Algorithms 2 and 3, respectively. [Hua+98, pp. 917-923]

Algorithm 2 The Sifting Procedure

Input: Time series y_t, sifting stopping criteria.
$h_t = y_t$.
while sifting stopping criteria **are** False **do**
    Identify all extrema in h_t.
    Create upper and lower envelopes e_t^u and e_t^l of h_t.
    Compute the mean of the envelopes: $m_t = \frac{e_t^u + e_t^l}{2}$.
    Compute IMF candidate $h_t = h_t - m_t$.
end while
$c_t = h_t$.
$\rho_t = y_t - h_t$.
Output: IMF c_t, residual ρ_t.

Algorithm 3 Empirical Mode Decomposition

Input: Time series y_t, EMD stopping criteria, sifting stopping criteria.
IMFs = {}.
$r_t = y_t$.
while EMD stopping criteria **are** False **do**
    c_t, ρ_t = Algorithm 2(y_t, sifting stopping criteria).
    IMFs = IMFs $\cup$ c_t.
    $r_t = \rho_t$.
end while
Output: IMFs, r_t.

There are several difficulties in making a model which uses the EMD as a preprocessing tool such as so-called modal mixing and the boundary effect [LDB21]. Modal mixing occurs when an IMF has oscillations on widely different scales and is often caused by signal intermittency. The consequence of modal mixing is that the IMFs loose physical meaning [ZX19]. Boundary effects occur when the endpoints of the time series are not extrema. In this case, the decomposition will be distorted by the cubic spline interpolation using the endpoints [WZP10]. Variations of the EMD have been developed in an attempt to alleviate these issues one of which is the ensemble EMD [WH09].

Additionally, since the number of IMFs for a given time series is unknown, the number of IMFs for the wind power production for each of the sub-grids can vary and therefore it is not possible to implement a multivariate model. However, multivariate

extensions of the EMD have been developed, see e.g. [RM10]. Moreover, in terms of real time applications, the addition of new datapoints to the time series poses a challenge for the EMD. However, this can be alleviated by using a window [TH11]. In [SPC20], wind power production is forecasted using ensemble EMD and a variation hereof called complete ensemble EMD and compares the performance in a real time application setup to an offline setup.

4.3 Sample Entropy

When forecasting time series data, it is important for the precision of the forecast that the time series data is predictable. The term predictability refers to the existence of patterns within the time series which can be predicted. Other terms used in this context is randomness, uncertainty, and complexity. For instance, attempting to forecast time series data drawn from a white noise process is pointless and the best forecast which can be made is to predict zero. This is an example of a time series which is unpredictable, completely random, has high uncertainty, and high complexity. [DBM19]

In this section, a measure of this aforementioned complexity called the sample entropy (SampEn) is introduced. The purpose of introducing such a measure is that the complexity varies for the different components of the EMD of a time series. This can then be exploited for forecasting since it is known that neural network based models have achieved good results when working with high complexity time series while simpler models in time series analysis are able to model signals of lower complexity [LDB21]. Sample entropy can be used as a measure to distinguish between EMD components of low entropy and EMD components of high entropy.

Consider a discrete time series $\{y_t\}_{t=1}^n$ and let

$$y_i^m = \{y_i, y_{i+1}, \dots, y_{i+m-1}\}, \quad i \leq n - m + 1 \tag{4.43}$$

be a block of length m. The sample entropy is defined as the negative logarithm of the empirical conditional probability that two blocks of length $m+1$ are similar given that two blocks of length m are similar. Whether two blocks are similar is determined by the following metric.

Definition 4.8 (Chebyshev Distance)
Let $\{y_t\}_{t=1}^n$ be a time series and let y_i^m and y_j^m be two blocks of length $m < n$ given as in Eq. (4.43), consisting of real numbers. Then the Chebyshev distance is defined as

$$d(y_i^m, y_j^m) = \max_{k=0,\dots,m-1} |y_{i+k} - y_{j+k}| \tag{4.44}$$

for $i, j \leq n - m + 1$.

The Chebyshev distance returns the largest difference between two points in the blocks y_i^m and y_j^m. Two blocks are said to be similar if their Chebyshev distance is less than some tolerance $r > 0$.

Thus, in order to compute the sample entropy a pair (m, r) must be chosen as the input parameters. Intuitively, a low sample entropy implies that a signal is repetitive and predictive, while a high sample entropy indicates a low amount of repeated patterns and a high amount of randomness.

The empirical probability that a block y_j^m is similar to some block y_i^m can be computed as

$$B_i^m(r) = \frac{1}{n-m-1} \sum_{\substack{j=1 \\ j \neq i}}^{n-m} \mathbb{1}[d(y_j^m, y_i^m) < r]. \tag{4.45}$$

Here the sum counts how many blocks of length m are similar to a block y_i^m. This is normalised with respect to the number of blocks of length m. Summing over all i and normalising, we get the empirical probability that any two blocks of length m are similar within the tolerance r

$$B^m(r) = \frac{1}{n-m} \sum_{i=1}^{n-m} B_i^m(r). \tag{4.46}$$

Similarly, we can compute the empirical probability that two blocks of length $m+1$ are similar

$$B^{m+1}(r) = \frac{1}{n-m-1} \sum_{i=1}^{n-m-1} B_i^{m+1}(r). \tag{4.47}$$

By construction, we have $B^{m+1}(r) \leq B^m(r)$ and the sample entropy is defined as [DBM19, pp. 11-20]

$$\text{SampEn}(m, r, n)(y) = -\log\Big(\frac{B^{m+1}(r)}{B^m(r)}\Big). \tag{4.48}$$

A popular choice of the values m and r are $m = 2$ and $r = 0.2\sigma$, respectively, where σ is the sample standard deviation of the time series. [RM00]

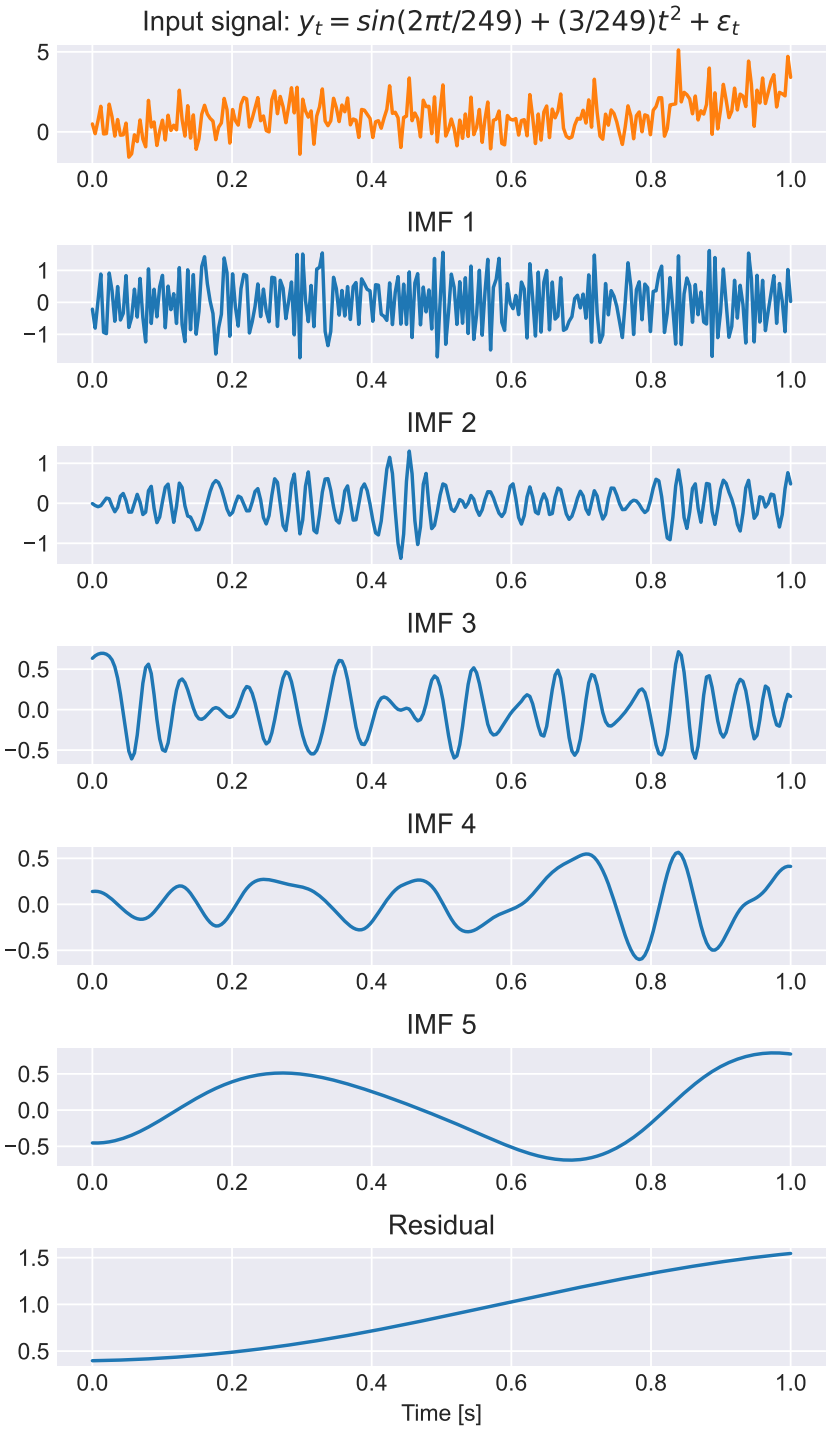


Figure 4.2: The results of applying the EMD procedure. The orange time series is the input signal and the blue time series are the IMFs and the residual.

We end this chapter by showing the results of applying the EMD procedure to the time series defined by

$$y_t = \cos\left(\frac{2\pi}{249}t\right) + \frac{3}{249}t^2 + \varepsilon_t \tag{4.49}$$

for $t = 0, \ldots, 249$ where $\varepsilon_t \overset{iid.}{\sim} \mathcal{N}(0, 1)$. The resulting IMFs and the residual can be seen in Fig. 4.2. We notice that the first IMFs have a higher complexity. Furthermore, we can see that the obtained IMFs are approximately symmetric with respect to zero mean. Finally, we can see the residual which shows the trend of the decomposed time series.

The sample entropy of the IMFs and the residual seen in Fig. 4.2 have been computed and can be seen in Fig. 4.3. We notice that the sample entropy decreases as the IMF number increases.

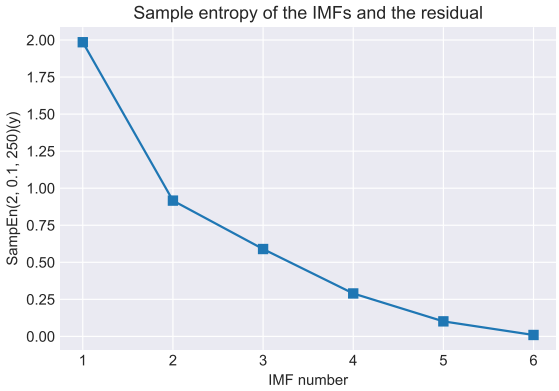


Figure 4.3: The sample entropy of the IMFs and residual seen in Fig. 4.2. The last point on the first axis is the residual.

5. Neural Networks

In other works related to forecasting of wind power production, neural networks, and often times architectures based on the recurrent neural network (RNN) variations long short-term memory (LSTM) networks or gated recurrent units (GRUs) have achieved good performance [Zha+19; Pen+20; Tou+21; LDB21]. Therefore, theory related to these architectures will be presented.

5.1 Recurrent Neural Networks

LSTMs and GRUs are two types of RNNs. RNNs have been shown to be good at modelling sequential data, making it a good candidate for wind power production forecasting [GBC16, p. 363]. The idea behind RNNs is to share parameters in different parts of a neural network model and use this to learn temporal dependencies. A typical RNN could for instance have the following forward propagation equations for an input $\boldsymbol{x}_t$ at time t

$$\boldsymbol{h}_t = \tanh(\boldsymbol{b}_h + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xh}\boldsymbol{x}_t), \tag{5.1}$$

$$\hat{\boldsymbol{y}}_t = \tanh(\boldsymbol{b}_y + \boldsymbol{W}_{hy}\boldsymbol{h}_t) \tag{5.2}$$

where tanh is the hyperbolic tangent activation function but could in practice be any squishing activation, $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ denotes the biases, $\boldsymbol{W}_{hh}$, $\boldsymbol{W}_{xh}$, and $\boldsymbol{W}_{hy}$ denotes the weight matrices, $\hat{\boldsymbol{y}}_t$ denotes the output at time t, and $\boldsymbol{h}_t$ denotes the hidden layer value at time t. [GBC16, pp. 370-371]

Simple RNNs have the problem that the gradient will often either vanish or explode when propagating the error many steps backwards in time and this is the problem the LSTM and GRU seek to solve [GBC16, pp. 390-392]. This problem have been handled by introducing a gate structure used to forget an old state after the information from this state has been used. [GBC16, p. 397]

However, before introducing the LSTM and the GRU the Hadamard product will be introduced as this will be used in the internal structure of these architecture types.

Definition 5.1 (Hadamard Product)
Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times n}$. Then the Hadamard product is defined by

$$(\boldsymbol{A} \odot \boldsymbol{B})_{i,j} = A_{i,j} B_{i,j}, \tag{5.3}$$

for $i = 1, 2 \ldots, m$ and $j = 1, 2, \ldots, n$.

5.1.1 Long Short-Term Memory

In LSTMs, the usual units of a neural network are replaced with so-called LSTM cells. The key aspects of the LSTM architecture are the gate structure and the internal self loop. The gate structure plays an important role in how the information is passed through the network. LSTMs have three different gates, i.e. the forget, input, and output gate.

Consider an input time series $\boldsymbol{x}_t$ observed at $t = 1, 2, \ldots, n$, then at time t the hidden layer gives output $\boldsymbol{h}_t$ and the three gates can be expressed by the following equations

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{hf} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{xf} \boldsymbol{x}_t + \boldsymbol{b}_f), \tag{5.4}$$
$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{hi} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{xi} \boldsymbol{x}_t + \boldsymbol{b}_i), \tag{5.5}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{ho} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{xo} \boldsymbol{x}_t + \boldsymbol{b}_o) \tag{5.6}$$

where $\boldsymbol{f}_t$, $\boldsymbol{i}_t$, and $\boldsymbol{o}_t$ are the forget gate, input gate, and output gate, respectively and where σ denotes the sigmoid activation function. In addition to the gates, the LSTM also has a state which is updated through a self loop

$$\boldsymbol{s}_t = \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{5.7}$$

where $\boldsymbol{g}_t$ is given by

$$\boldsymbol{g}_t = \tanh(\boldsymbol{W}_{hg} \boldsymbol{h}_{t-1} + \boldsymbol{b}_{hg} + \boldsymbol{W}_{xg} \boldsymbol{x}_t + \boldsymbol{b}_{xg}). \tag{5.8}$$

Afterwards, the hidden layer output of an LSTM cell is computed as

$$\boldsymbol{h}_t = \tanh(\boldsymbol{s}_t) \odot \boldsymbol{o}_t. \tag{5.9}$$

The architecture of an LSTM cell can be seen in Fig. 5.1. [GBC16, pp. 397-400]

The gate structure can be interpreted as follows. The forget gate determines the information which needs to be discarded from the state. The input gate determines the information which should be stored in the state. Finally, the output gate determines what will be outputted from the state to the hidden unit or if a cell should be shut down.

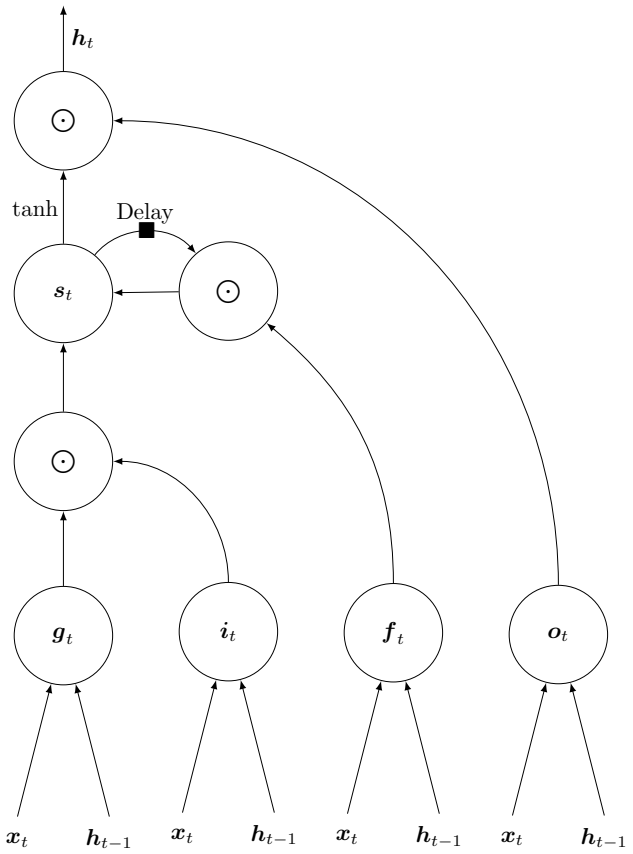


Figure 5.1: A block diagram of an LSTM cell. The gate variables $\boldsymbol{g}_t$, $\boldsymbol{i}_t$, $\boldsymbol{f}_t$ and $\boldsymbol{o}_t$ internally apply the weights, biases, and activation functions seen in the forward propagation equations. Figure inspired by [GBC16, fig. 10.16].

5.1.2 Gated Recurrent Unit

Another type of gated RNN is the GRU. The idea of the GRU is similar to that of the LSTM but with fewer gates. In the GRU, there are only an update gate and a reset gate. These gates are computed as usual

$$\boldsymbol{u}_t = \sigma(\boldsymbol{W}_{hu}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xu}\boldsymbol{x}_t + \boldsymbol{b}_u), \tag{5.10}$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{b}_r). \tag{5.11}$$

Using the reset gate, the state $\boldsymbol{s}_t$ is computed as

$$\boldsymbol{s}_t = \tanh\big(\boldsymbol{W}_{hs}\boldsymbol{h}_{t-1} \odot \boldsymbol{r}_t + \boldsymbol{W}_{xs}\boldsymbol{x}_t + \boldsymbol{b}_s\big). \tag{5.12}$$

The hidden unit is then updated as [GBC16, p. 400]

$$\boldsymbol{h}_t = \boldsymbol{u}_t \odot \boldsymbol{h}_{t-1} + (\boldsymbol{1} - \boldsymbol{u}_t) \odot \boldsymbol{s}_t. \tag{5.13}$$

The interpretation of the gates are that the update gate, $\boldsymbol{u}_t$, determines how much of $\boldsymbol{h}_{t-1}$ is transferred to $\boldsymbol{h}_t$ while the reset gate, $\boldsymbol{r}_t$, determines the influence which $\boldsymbol{h}_{t-1}$ has on a state, $\boldsymbol{s}_t$, to the hidden state $\boldsymbol{h}_t$.

6. Experimental Setup

On the basis of the theory introduced in Chapters 2 to 5, we are now ready to implement models to solve Eq. (1.3). Thus, in this chapter we will introduce the models which will be implemented as well as describe the experimental setup used to train and test the models. However, before proceeding to do so, the available data will be introduced.

6.1 Data Description

The data used in this project has been supplied by the Danish company Energinet which is the TSO in Denmark. The data covers all of Denmark and includes wind power production, numerical weather predictions (NWP), the installed capacity, and information regarding individual wind mills and wind farms. The data covers the time period from the 1st of January 2019 to the 5th of October 2021. In this chapter, some details regarding the supplied data will be discussed as this will influence model selection and implementation in this project.

Denmark is divided into two separate grids called DK1 and DK2. The DK1 grid covers Jutland and Funen, while DK2 covers Zealand, adjacent islands, and Bornholm. Moreover, each grid consists of a number of sub-grids with 15 sub-grids in DK1 and 6 sub-grids in DK2. Sub-grid $l = 1, \ldots, 15$ in DK1, is denoted as DK1-l and is also referred to as the lth sub-grid, while sub-grid $l = 1, \ldots, 6$ in DK2, is denoted as DK2-l and is also referred to as the $(l + 15)$th sub-grid. The map in Fig. 6.1 displays the wind mills in the Danish power grid, herein showing which wind mills belong to the different sub-grids.

The data which Energinet has available only includes a few measurements from wind mills and the actual wind power production data available is upscaled from these measurements for each sub-grid. Hence, the wind power production for the individual wind mills in the power grid is not available. The upscaling is done according to the installed capacity for each sub-grid and is referred to as SCADA-upscaling. This SCADA-upscaled wind power production data is available for each sub-grid in Denmark with 5 minute intervals. The power data for sub-grid l where $l = 1, \ldots, 21$ is denoted by the time series $\{P_{l,t}\}_{t=1}^{n}$ where n is the number of available data points. In Fig. 6.2a, the data for September 2021 is shown.

The NWPs are computed by weather forecasting services and given to Energinet.

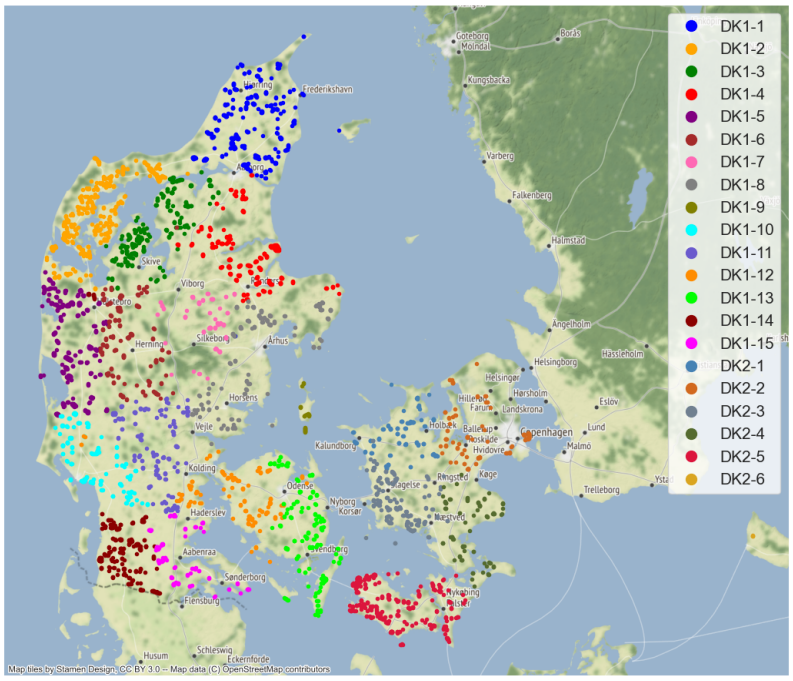
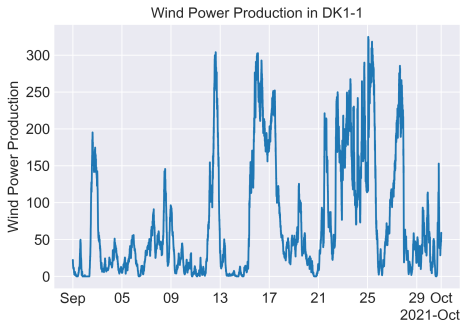
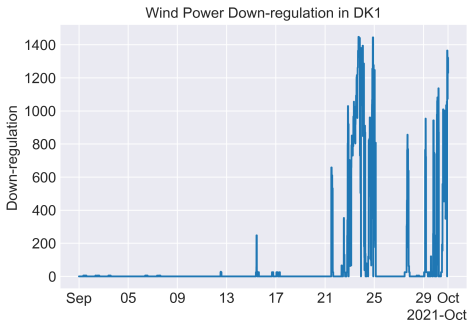


Figure 6.1: Map of wind mills in the Danish power grid.



(a) Wind power production in megawatts for sub-grid DK1-1 from 1st of September 2021 to 1st of October 2021.



(b) Down-regulation of the installed wind power production capacity in megawatts in grid DK1 from the 1st of September 2021 to the 1st of October 2021.

Figure 6.2: Data for September 2021.

The NWPs are updated every 3 hours and provide weather forecasts up to 54 hours ahead with 1 hour intervals on a 15×15 km grid. The NWPs which are available are the temperature at a height of 2 meters, the wind speed at heights of 10 and 100 meters, and the wind direction at heights of 10 and 100 meters.

When modelling the wind power production, we expect a dependency on the NWPs. Particularly, if we consider a single wind mill, we have a theoretical relation between the wind power and the NWPs which can be seen in Eq. (1.1) where the wind power can be modelled using the wind speed and the air density which in turn is dependent on the temperature. Additionally, in [CD14] it was found that the correlation between the wind power production and the wind speed is dependent on the wind direction. Since the wind power production history is only available on sub-grid level, we will make the simplification of assuming that the wind power in a given sub-grid is produced at one point in the area which we will refer to as the midpoint of the sub-grid. This midpoint is determined by computing the mean of the coordinates of each wind mill in a given sub-grid. Then when modelling the wind power production for a given sub-grid, we will restrict our attention to the NWP which is closest to the midpoint of that sub-grid. A map showing the computed midpoints of the sub-grids together with the NWP gridpoints is given in Fig. 6.3.

The temperature forecast at time t to time $t+\tau$ for sub-grid $l = 1, \ldots, 21$ is denoted as $T^t_{l,t+\tau}$. In a similar way, the wind speed forecast at 10 meters and 100 meters are denoted as $v^{1,t}_{l,t+\tau}$ and $v^{2,t}_{l,t+\tau}$, respectively. Finally, the wind direction forecast at 10 meters and 100 meters are denoted as $\omega^{1,t}_{l,t+\tau}$ and $\omega^{2,t}_{l,t+\tau}$, respectively.

Under certain weather conditions, e.g. when it is windy such that the power grid would be overloaded by the power production from the wind mills, the wind power production is regulated by forcing some wind mills to not produce power. Doing so changes the capacity at that time. However, the capacity change is only available at a grid level, i.e. the changes in DK1 and DK2 separately. Hence, we do not know how the wind power production is regulated in the individual sub-grids. Possible approaches for handling this down-regulation in the modelling of the wind power production could be to exclude periods in time in which a large amount of down-regulation is enforced. Another approach could be to use this information to map the wind power production data to an approximation of the production numbers which would have occurred in case of no down-regulation. However, how to compute this approximation is unclear. Instead, we will try to use the down-regulation as an exogenous variable in our models with the expectation that doing so will explain away some of the impact of the down-regulation. The down-regulations are given in 5 minute intervals and for grid DK1 and grid DK2 they will be denoted by the time series $\{\rho_{1,t}\}_{t=1}^n$ and $\{\rho_{2,t}\}_{t=1}^n$, respectively. In Fig. 6.2b, we see the wind power down-regulation for DK1 in September 2021.

Since the wind speed and the wind power production are strongly related, plotting these variables against each other can provide insight into what the relation is. This is called power curves. In Fig. 6.4, the wind power production for the sub-grids DK2-3 and DK1-8 are plotted against the wind speed at the heights of 10 meters and 100

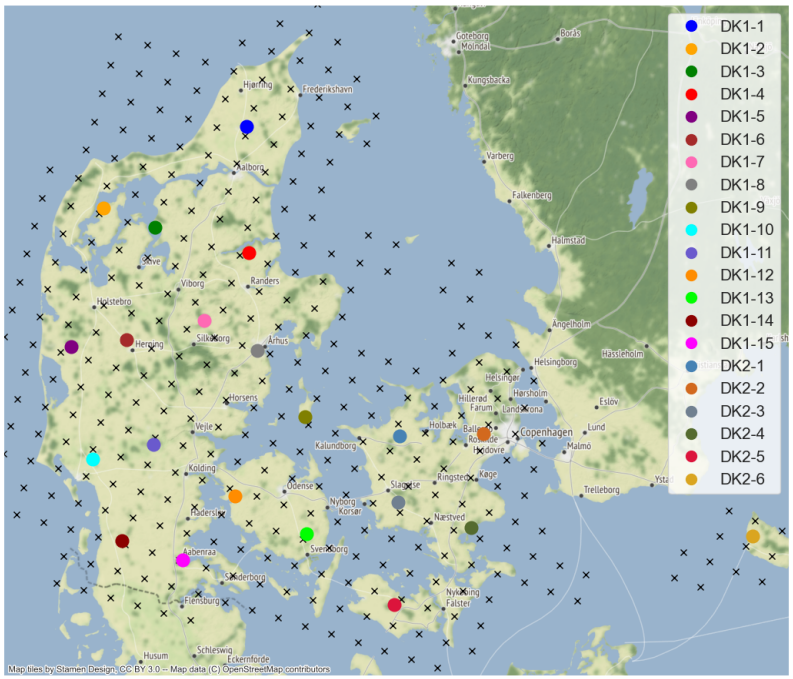


Figure 6.3: Sub-grid midpoints plotted together with the NWP gridpoints.

meters. These sub-grids have been chosen for plotting as the power curves by visual inspection have the least and the most uncertainty in the wind power based on the wind speed, respectively.

6.2 Selected Models

In this section, the different models used for our experiments are introduced. Before proceeding to do so, some considerations regarding the models will be given. This will give the motivation for our model choices.

In general, we are interested in a model which for a given sub-grid l can forecast the wind power production for time $t+\tau$ at time t using the wind power production at previous times, NWP data, and regulation data, i.e. the model

$$\hat{P}^t_{l,t+\tau} = f(P^t_{l,t-1}, \ldots, P^t_{l,t-p}, \boldsymbol{z}^t_{l,t+\tau}; \boldsymbol{\theta}) \tag{6.1}$$

where $\hat{P}^t_{l,t+\tau}$ denotes the wind power forecast in sub-grid l for time $t+\tau$ at time t, $P_{l,t}$ denotes the wind power production in sub-grid l at time t, $\boldsymbol{z}^t_{l,t+\tau}$ is a vector consisting of NWP data and regulation data for sub-grid l at time $t+\tau$ given time t and will be referred to as the exogenous variable, p is the amount of power history needed in the model, and f is the model we need to train. The exogenous variables will consist of a combination of transformations of the NWPs and regulation data and will be

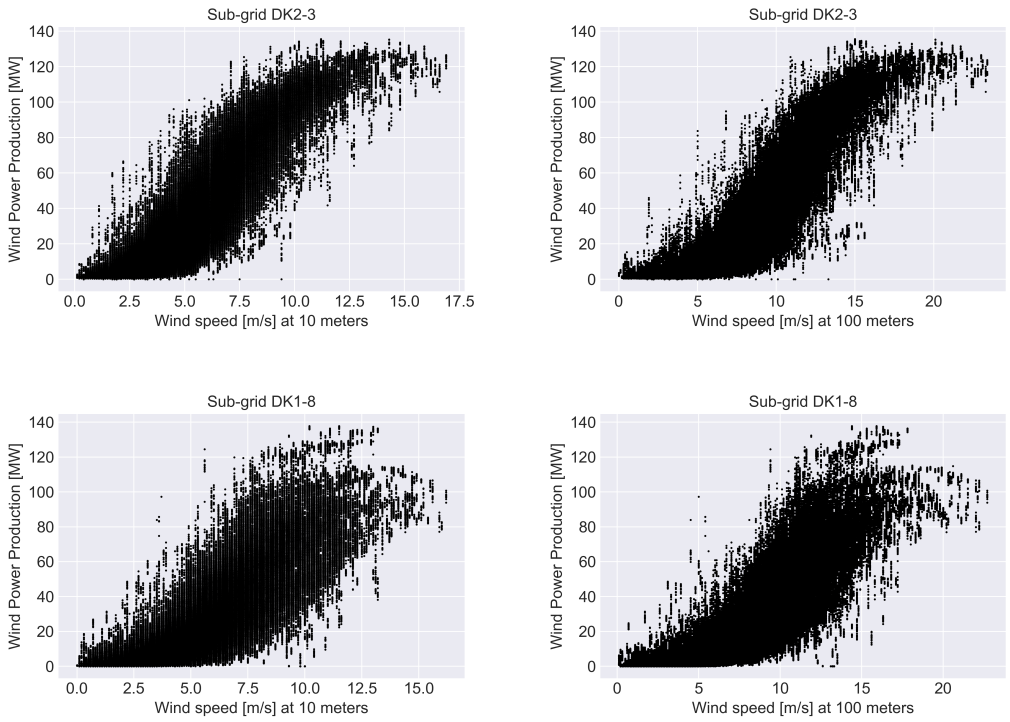


Figure 6.4: This figure shows the power curves for two of the sub-grids in relation to the wind speed at the heights 10 meters and 100 meters.

given as follows. Consider sub-grid l and let it be in grid i. The exogenous variable for sub-grid l at time $t+\tau$ given time t is

$$\begin{aligned} \boldsymbol{z}_{l,t+\tau}^{t} = \Big(&v_{l,t+\tau}^{1,t}, (v_{l,t+\tau}^{1,t})^2, (v_{l,t+\tau}^{1,t})^3, v_{l,t+\tau}^{2,t}, (v_{l,t+\tau}^{2,t})^2, (v_{l,t+\tau}^{2,t})^3, \\ &\cos\Big(\omega_{l,t+\tau}^{1,t}\Big), \sin\Big(\omega_{l,t+\tau}^{1,t}\Big), \cos\Big(\omega_{l,t+\tau}^{2,t}\Big), \sin\Big(\omega_{l,t+\tau}^{2,t}\Big), T_{l,t}^{t}, \rho_{i,t}\Big)^T. \end{aligned} \tag{6.2}$$

The reason we have chosen to consider the wind speed up to the power of three is that we have seen from Eq. (1.1) that the wind power is directly related to the wind speed cubed and since using a polynomial is also the approach in [CD14]. Furthermore, we have chosen the transformations of the wind direction such that the wind direction can be seen as a point on the unit circle to avoid the discontinuity in the representation of the direction at an angle of $0 = 2\pi$. Since we have exogenous variables for each sub-grid, we obtain a matrix $\boldsymbol{z}_{t+\tau}^{t} \in \mathbb{R}^{21\times 12}$.

In the literature, different models have been applied as described in Section 1.3 and each has its own advantages and disadvantages. Since weather conditions such as the wind speed do not change rapidly over shorter distances [Jai16, pp. 29-30], we suspect that there is a spatial correlation between the wind power of sub-grids which

we want to include in the model. This has also been seen in [BT07; ADS99] where the prediction accuracy is improved when spatial correlation in the wind power is taken into account. Therefore, we will not only consider temporal models but also spatio-temporal models.

In this project, we have been considering the theory of time series analysis in Chapters 2 and 3, specifically the class of s-VARIMAX models, and that of neural networks in Chapter 5, particularly LSTM and GRU neural networks. These types of models will be tested in this project and have been chosen since both types of models are able to include the spatial correlation in the wind power and since models of these types have been used in the literature [ZCA16; LDB21; PS+09; Zha+19; Tou+21; KS09; Liu+10].

Another aspect which should be taken into account for the models is the stationarity of the wind power data since this highly influences the suitability of different types of models. Particularly, the time series models introduced in Chapters 2 and 3 are well suited for stationary data. Furthermore, compared to neural networks time series models may lead to a lower computational complexity as in [PS+09]. However, the time series models do not work well for non-stationary data and even when performing differencing as described in Section 2.1.3 the data might not become stationary. In opposition to the time series models, neural networks work well for non-stationary data leading to high prediction accuracy. However, neural networks can be difficult to train to learn a mapping capturing the characteristics of the data well and they can be subject to problems during optimisation in terms of ending in a local minimum [LDB21][GBC16, ch. 8]. When dealing with non-stationary data, it can be advantageous to decompose the time series, e.g. using the EMD introduced in Chapter 4, into components that can be forecasted separately with less error [LDB21].

In the following, a detailed description of each of the models considered in this project will be given.

6.2.1 Persistence

The simplest method implemented is the persistence method and the purpose of this method is to be a baseline for the other models. A persistence method is a non-parametric method and does not require training in opposition to the other methods used in this project. In a persistence method, the forecast is chosen as the most recent observations. Hence, the τ-ahead forecast at time t for sub-grid $l = 1, \dots, 21$ is

$$\hat{P}^t_{l,t+\tau} = P_{l,t}. \tag{6.3}$$

We expect the persistence method to be decent for very short term forecast, however the performance should decrease as τ increases.

6.2.2 s-ARIMAX

The next model which we will consider is the class of s-ARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ models. These are time series models well suited for stationary data and which only

model the temporal dependency in the data. This type of model is defined as

$$\nabla^{d_s,d} P_{l,t} = \boldsymbol{\Xi}_l^T \boldsymbol{z}_{l,t} + \sum_{j \in \mathcal{S}_{\mathrm{AR}}(p,p_s)} \phi_{l,j} \nabla^{d_s,d} P_{l,t-j} - \sum_{i \in \mathcal{S}_{\mathrm{MA}}(q,q_s)} \psi_{l,i} u_{l,t-i} + u_{l,t} \tag{6.4}$$

for $t = 1, \ldots, n$ where $\nabla^{d_s,d} = (1 - B^s)^{d_s}(1 - B)^d$ is the differencing operation, $P_{l,t}$ is the wind power production for sub-grid $l \in \{1, \ldots, 21\}$, $u_{l,t}$ is a white noise process, $\phi_{l,j}$ are the autoregressive parameters, $\psi_{l,i}$ are the moving average parameters, $\mathcal{S}_{\mathrm{AR}}(p, p_s)$ and $\mathcal{S}_{\mathrm{MA}}(q, q_s)$ are given as in Eq. (2.29), and the exogenous variables are

$$\boldsymbol{z}_{l,t} = \begin{bmatrix} 1 \\ \nabla^{d_s,d} \boldsymbol{z}_{l,t}^t \end{bmatrix} \tag{6.5}$$

where $\boldsymbol{z}_{l,t}^t$ are defined as in Eq. (6.2). The parameters are estimated as in Eq. (2.49).

This model uses the tools introduced in Chapter 2 to form a model which models the temporal dependencies in the wind power production. The estimation procedure computes the maximum likelihood estimate in case $q = q_s = 0$ and the noise process u_t is Gaussian. Otherwise, it serves as a simple and fast way of fitting the model to minimise the squared error. The estimation algorithm introduced in Algorithm 1 was not used due to time and software limitations.

The hyperparameters for this model are the amount of power history used in form of p and p_s, the usage of moving average terms in form of q and q_s, the orders of differencing in form of d and d_s, and finally the seasonal lag s.

6.2.3 Univariate RNN

This model uses a LSTM or GRU neural network for each sub-grid in the dataset. In the following, the forward propagation equations for the neural network are given. Let the input to the neural network for sub-grid l at time t be $\boldsymbol{x}_{l,t} = (P_{l,t-1}, \ldots, P_{l,t-p}, \boldsymbol{z}_{l,t}^{t-1})^T$ where p is the amount of used power history. For the LSTM model, the first hidden layer is a LSTM layer which is computed as in Eq. (5.9) and is denoted LSTM(). This leads to the first hidden state

$$\boldsymbol{h}_t^{(1,l)} = \mathrm{LSTM}(\boldsymbol{x}_{l,t}, \boldsymbol{h}_{t-1}^{(1,l)}, \boldsymbol{s}_{t-1}^{(1,l)}) \tag{6.6}$$

where $\boldsymbol{h}_{t-1}^{(1,l)}$ and $\boldsymbol{s}_{t-1}^{(1,l)}$ are the hidden state and the cell state for the previous datapoint, respectively. The hidden state for LSTM layer $i = 2, 3$ is computed as

$$\boldsymbol{h}_t^{(i,l)} = \mathrm{LSTM}(\boldsymbol{h}_t^{(i-1,l)}, \boldsymbol{h}_{t-1}^{(i,l)}, \boldsymbol{s}_{t-1}^{(i,l)}). \tag{6.7}$$

Finally, the hidden state $\boldsymbol{h}_t^{(3,l)}$ is passed to a fully connected layer yielding the output of the neural network as

$$\hat{P}_{l,t}^{t-1} = \boldsymbol{W} \boldsymbol{h}_t^{(3,l)} + \boldsymbol{b} \tag{6.8}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and the bias vector for the output layer, respectively.

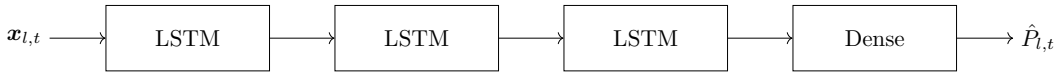


Figure 6.5: Block diagram of the univariate LSTM model.

A block diagram of the LSTM model is depicted in Fig. 6.5. The model for the GRU has a similar design but with GRUs instead of LSTMs and the hidden state equations are also determined similarly but computed using Eq. (5.13).

A neural network using the same set of hyperparameters is trained for each sub-grid in the data. In the following, the learning procedure for the neural network models is discussed. When implementing a machine learning model, the goal is to reduce the error on the test data by fitting a model on the training data. If the training error is large, the model is said to be under-fitted, whereas if the gap between test error and testing error is large, the model is said to be over-fitted. When fitting a neural network architecture to data, some choices have to made as to which optimiser and which regulation methods to use and in addition to this a number of hyperparameters must be chosen in order to obtain a model which fits the training data well and generalises well to the test data.

In terms of regularisation, we have chosen to use early stopping as well as dropout. Early stopping is a method which is used to evaluate when the model training should stop [GBC16, pp. 239-245]. This works by letting the training run until a loss calculated on a validation set has not been improved for c iterations. Thus, early stopping is a method used to stop the model from over-fitting on the training data. In practice, we use $c = 100$ and we evaluate the validation loss after every 100th parameter update. Secondly, we use dropout in the hidden layers. Dropout works by removing a random selection of hidden units in the network in each iteration of training [GBC16, pp. 251-261]. The amount of hidden units removed is determined by a dropout rate which is another hyperparameter that will have to be chosen. These regularisation methods have been chosen as [GBC16, p. 413] states that early stopping should be used almost universally and that dropout is an excellent regulariser which is easily implemented.

The neural network is trained using the MSE loss function Eq. (6.16). The optimisation method used to train the neural networks is the root mean square propagation algorithm with an exponential learning rate decay [GBC16, pp. 299-300].

When implementing a neural network model, the following hyperparameters will be fitted: the learning rate, the learning rate decay, the batch size, the number of layers, the number of hidden units in each layer, the dropout rate, and the amount of power history to use p. [GBC16, pp. 286-287]

6.2.4 s-VARIMAX

The univariate time series model described in Section 6.2.2 can be extended to the multivariate case by modelling the wind power production for all the sub-grids simultaneously. In this way both temporal and spatial dependencies can be taken into

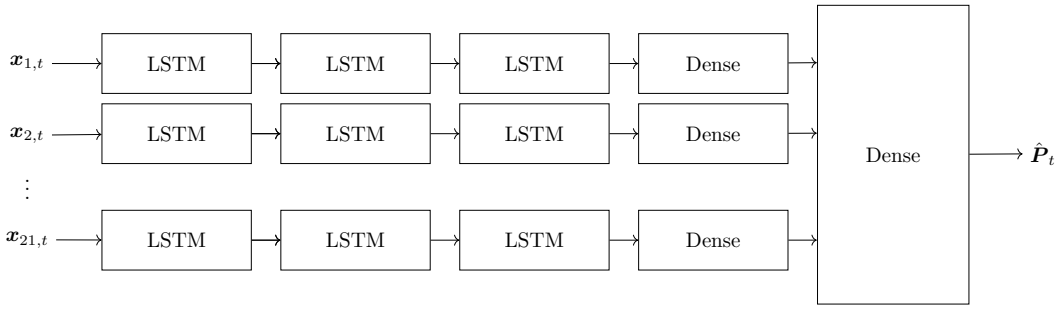


Figure 6.6: Block diagram of the multivariate LSTM model.

account, however this also means that the complexity of the model is higher. This motivates 21-dimensional s-VARIMAX$(p,d,q)\times(p_s,d_s,q_s)_s$ models of the type

$$\nabla^{d_s,d}\boldsymbol{P}_t = \begin{bmatrix} \boldsymbol{\Xi}_1^T\boldsymbol{z}_{1,t} \\ \vdots \\ \boldsymbol{\Xi}_k^T\boldsymbol{z}_{21,t} \end{bmatrix} + \sum_{j\in\mathcal{S}_{\mathrm{AR}}(p,p_s)} \boldsymbol{\Phi}_j\nabla^{d_s,d}\boldsymbol{P}_{t-j} - \sum_{i\in\mathcal{S}_{\mathrm{MA}}(q,q_s)} \boldsymbol{\Psi}_i\boldsymbol{u}_{t-i} + \boldsymbol{u}_t \tag{6.9}$$

for $t=1,\ldots,n$ where $\nabla^{d_s,d}=(1-B^s)^{d_s}(1-B)^d$ is the differencing operation, $\boldsymbol{P}_t\in\mathbb{R}^{21}$ is the wind power production, $\boldsymbol{u}_t$ is a k-dimensional vector white noise process, $\mathcal{S}_{\mathrm{AR}}(p,p_s)$ and $\mathcal{S}_{\mathrm{MA}}(q,q_s)$ are given as in Eq. (2.29), and $\boldsymbol{z}_{l,t}$ are the exogenous variables as in Eq. (6.5). Estimation of the parameters are done as in Eq. (2.49) and the hyperparameters are the same as for the univariate case Section 6.2.2.

6.2.5 Multivariate RNN

In this section, we will present the multivariate RNN which consists of either a LSTM or GRU neural network. In the multivariate LSTM, the input for each sub-grid is first given each to a separate LSTM and the output of these are joined before being applied to a unified fully connected layer. The purpose of this unified fully connected layer is to try to determine the spatial properties of the data. A block diagram of the multivariate LSTM can be seen in Fig. 6.6. In terms of fitting these neural networks to the data, the same methods as stated in Section 6.2.3 are used and the hyperparameters are the same. In the following, the forward propagation equations will be given. Let $\boldsymbol{x}_{l,t}=(P_{l,t-1},\ldots,P_{l,t-p},\boldsymbol{z}_{l,t}^T)^T$ where p is the amount of power history used be the input to the neural network for sub-grid l at time t. The first part of the neural network consists of a series of LSTM layers in parallel. This leads to the first hidden state for sub-grid l

$$\boldsymbol{h}_t^{(1,l)} = \mathrm{LSTM}(\boldsymbol{x}_{l,t},\boldsymbol{h}_{t-1}^{(1,l)},\boldsymbol{s}_{t-1}^{(1,l)}) \tag{6.10}$$

where $\boldsymbol{h}_{t-1}^{(1,l)}$ and $\boldsymbol{s}_{t-1}^{(1,l)}$ are the hidden state and the cell state for the previous datapoint, respectively. The hidden state for LSTM layer $i=2,3$ for sub-grid l is computed as

$$\boldsymbol{h}_t^{(i,l)} = \mathrm{LSTM}(\boldsymbol{h}_t^{(i-1,l)},\boldsymbol{h}_{t-1}^{(i,l)},\boldsymbol{s}_{t-1}^{(i,l)}). \tag{6.11}$$

Next the hidden states $\boldsymbol{h}_t^{(3,l)}$ are passed to a fully connected layer

$$h_t^{(4,l)} = \boldsymbol{W}_l \boldsymbol{h}_t^{(3,l)} + \boldsymbol{b}_l \tag{6.12}$$

where $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ are the weight matrix and the bias vector for the fully connected layer for sub-grid l, respectively and $h_t^{(4,l)} \in \mathbb{R}$. The hidden states for each sub-grid are then combined to a single hidden state as

$$\boldsymbol{h}_t^{(4)} = \begin{pmatrix} h_t^{(4,1)} \\ \vdots \\ h_t^{(4,21)} \end{pmatrix}. \tag{6.13}$$

Subsequently, the hidden state $\boldsymbol{h}_t^{(4)}$ is passed to the output by a fully connected mapping as

$$\hat{\boldsymbol{P}}_t^{t-1} = \boldsymbol{W} \boldsymbol{h}_t^{(4)} + \boldsymbol{b} \tag{6.14}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and the bias vector for the output layer, respectively. As in Section 6.2.3, the LSTM and GRU are interchangeable.

6.2.6 EMD-LSTM-ARMA

As previously mentioned in Section 6.2, the stationarity of the wind power production time series influences which models are best suited for forecasting. Particularly, we expect that the s-ARIMAX and s-VARIMAX models are better in case of stationarity, while the neural networks could do better if the data is non-stationary. As will be seen in Chapter 7, the wind power production is indeed non-stationary and simply attempting to use differencing does not yield stationary data.

In [LDB21; Zhe+13; LW08], the EMD has been used to forecast wind speed and wind power production. The EMD was introduced in Chapter 4. Similarly to [LDB21] who forecasts wind speed using the EMD in conjunction with ARMA models and LSTM/GRU neural networks, we will implement an EMD-LSTM-ARMA model for wind power production forecasting. A block diagram of the model is shown in Fig. 6.7. The EMD is used to reduce the complexity and non-stationarity of the wind power. Then the high entropy parts are modelled by LSTM neural networks and the low entropy parts are modelled by the ARMA time series model such that we take advantage of the strengths of both the ARMA models and the LSTM neural networks.

The physical interpretation and the influence of the exogenous variables on each of the IMFs and the residual is not clear, hence the exogenous variables will not be included as input for this model. Additionally, the model will be univariate as was also discussed in Section 4.2.

Let $\{c_{l,t}^{(i)}\}_{t=1}^n$ be the ith IMF for $i = 1, \ldots, \nu - 1$ and the residual for $i = \nu$ of the wind power production time series for sub-grid l, i.e. $\{P_{l,t}\}_{t=1}^n$. For the wind power production data for all the sub-grids, the number of IMFs including the residual is

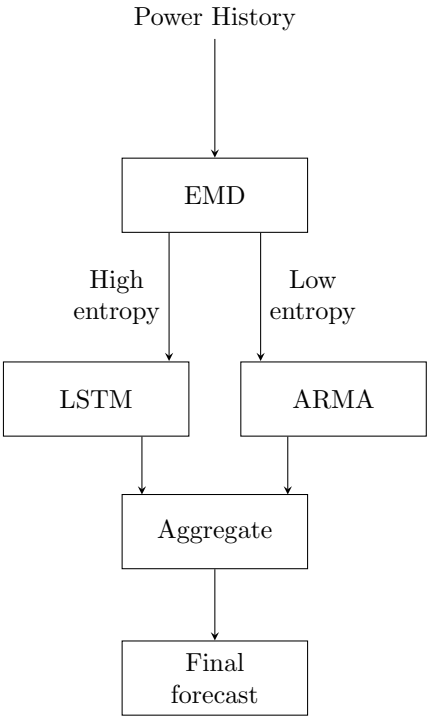


Figure 6.7: Block diagram of the EMD-LSTM-ARMA model.

in the set $\nu \in 19, 20, 21, 22$. Moreover, let the sample entropy of each part of the decomposition be given as $\text{SampEN}_i = \text{SampEn}(2, 0.2, n)(c_{l,t}^{(i)})$, see Eq. (4.48), and let Ω denote the threshold separating low entropy from high entropy. If $\text{SampEN}_i \geq \Omega$, then $\{c_{l,t}^{(i)}\}_{t=1}^{n}$ is forecasted using an LSTM network with forward propagation equations, learning algorithm, and hyperparameters as the LSTM network in Section 6.2.3. Otherwise if $\text{SampEN}_i < \Omega$, then $\{c_{l,t}^{(i)}\}_{t=1}^{n}$ is forecasted using an ARMA(p, q) model which is a simplification of the s-ARIMAX model in Section 6.2.2.

When forecasting, the trained models for each IMF and the residual are used to forecast the respective IMF or residual and the forecast of the wind power production is computed by aggregating the results for each of the IMFs and the residual as

$$\hat{P}_{l,t+\tau} = \sum_{i=1}^{\nu} \hat{c}_{l,t+\tau}^{(i),t} \tag{6.15}$$

where $\hat{c}_{l,t+\tau}^{(i),t}$ is the forecast of $c_{l,t+\tau}^{(i)}$.

6.2.7 Overview

To summarise, the models implemented in this project are listed in Table 6.1. Here it is indicated what the input to each of the models are and what the output is. Note

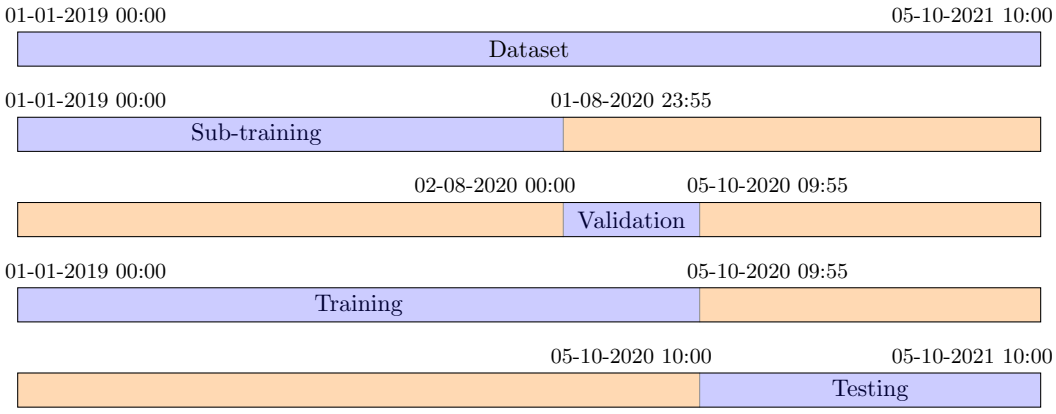


Figure 6.8: Partitioning of the dataset into the four datasets.

that for simplicity of notation, we have assumed that the time series models ARIMAX and VARIMAX have no seasonal component. In Chapter 7, hyperparameters will be

Model	Input	Output
Persistence	$P_{l,t-1}$	$\hat{P}_{l,t}$
ARIMAX	$P_{l,t-1}, \ldots, P_{l,t-p}$ and $\boldsymbol{z}_{l,t}^t$	$\hat{P}_{l,t}$
Univariate RNN	$P_{l,t-1}, \ldots, P_{l,t-p}$ and $\boldsymbol{z}_{l,t}^t$	$\hat{P}_{l,t}$
VARIMAX	$\boldsymbol{P}_{t-1}, \ldots, \boldsymbol{P}_{t-p}$ and $\boldsymbol{z}_t^t$	$\hat{\boldsymbol{P}}_t$
Multivariate RNN	$\boldsymbol{P}_{t-1}, \ldots, \boldsymbol{P}_{t-p}$ and $\boldsymbol{z}_t^t$	$\hat{\boldsymbol{P}}_t$
EMD-LSTM-ARMA	$P_{l,t-1}, \ldots, P_{l,t-p}$	$\hat{P}_{l,t}$

Table 6.1: Input and output of each of the selected models.

selected, and results comparing the models will be shown.

6.3 Supervised Learning Setup

In this project, we want to obtain models described by Eq. (6.1) for 5 minute ahead wind power forecasting, i.e. $\tau = 1$. In order to obtain such models, we will use the provided data from Energinet. As mentioned in Section 6.1, the data consists of the wind power production, NWP data, and regulation data from 1st of January 2019 at 00:00 to 5th of October 2021 at 10:00. The data for the wind power production and the regulation data consists of n samples. In this section, we will describe how the provided data will be used to train, select hyperparameters, and evaluate the models described in Section 6.2.

The dataset will be divided into two sets, i.e. a training set and a test set. In order to determine hyperparameters, the training set will be further divided into a sub-training set and a validation set. In the following, each of the sub-datasets will be described. This dataset partitioning can be seen in Fig. 6.8.

6.3.1 Training

The training set consists of data from 1st of January 2019 at 00:00 to 5th of October 2020 at 09:55 as seen in Fig. 6.8. The wind power measurements in the training set are denoted $\{\boldsymbol{P}_t\}_{t=1}^{n_{\text{train}}}$ and p denotes the number of previous wind power samples used in the model.

We train both time series and neural networks and these are trained differently. For the time series, the least squares method is used for training as described in Sections 6.2.2 and 6.2.4. For the RNN models, the mean squared error (MSE) is used as loss function in the training of the network. The MSE is calculated for a batch of some batch size. Let the index set for the ith batch of training be given as $B_i = \{i \cdot \text{batch size} + j : j = 0, \ldots, \text{batch size} - 1\}$. The MSE of batch i for the univariate case is then given by

$$\text{MSE}_l = \frac{1}{\text{batch size}} \sum_{t \in B_i} \left(P_{l,t+1} - \hat{P}_{l,t+1}^t\right)^2 \tag{6.16}$$

for $l = 1, \ldots, 21$. The MSE of batch i for the multivariate case is

$$\text{MSE} = \frac{1}{\text{batch size}} \sum_{t \in B_i} \frac{\left(\boldsymbol{P}_{t+1} - \hat{\boldsymbol{P}}_{t+1}^t\right)^T \left(\boldsymbol{P}_{t+1} - \hat{\boldsymbol{P}}_{t+1}^t\right)}{21}. \tag{6.17}$$

6.3.2 Testing

When the selected models have been trained, we want to evaluate the performance of the models using two evaluation metrics which are the MSE and the normalised mean absolute error (NMAE).

As seen in Fig. 6.8, the test set consists of the data from 5th of October 2020 at 10:00 to 5th October 2021 at 10:00 with n_{test} wind power measurements. The wind power measurements of all sub-grids in the test set are denoted $\{\boldsymbol{P}_t\}_{t=n_{\text{train}}+1}^{n}$.

The average MSE, i.e. the MSE averaged over the sub-grids, in the testing phase is computed by

$$\text{MSE}_{\text{average}} = \frac{1}{21} \sum_{l=1}^{21} \text{MSE}_l \tag{6.18}$$

where

$$\text{MSE}_l = \frac{1}{n_{\text{test}} - 1 - p} \sum_{t=n_{\text{train}}+1+p}^{n-1} \left(P_{l,t+1} - \hat{P}_{l,t+1}^t\right)^2 \tag{6.19}$$

for $l = 1, \ldots, 21$. As mentioned, we will also use NMAE to evaluate the selected models. The average NMAE is given by

$$\text{NMAE}_{\text{average}} = \frac{1}{21} \sum_{l=1}^{21} \text{NMAE}_l \tag{6.20}$$

where

$$\text{NMAE}_l = \frac{1}{n_{\text{test}} - 1 - p} \sum_{t=n_{\text{train}}+1+p}^{n-1} \frac{|P_{l,t+1} - \hat{P}^t_{l,t+1}|}{P_{l,\max}} \tag{6.21}$$

for $l = 1, \ldots, 21$ and where $P_{l,\max} = \max_{t=1,\ldots,n_{\text{train}}} P_{l,t}$.

In addition to evaluating the model on 5 minute ahead forecasting, we also examine how the performance of the selected models trained for 5 minute ahead forecasting evolves for longer forecasts. This will be done by iteratively using the models trained for 5 minute forecasts where the oldest observation is removed and the newest prediction is used instead, i.e.

$$\hat{P}^t_{l,t+\tau} = f(\hat{P}^t_{l,t+\tau-1}, \ldots, \hat{P}^t_{l,t+\tau-p}, \boldsymbol{z}^t_{l,t+\tau}; \boldsymbol{\theta}) \tag{6.22}$$

where $\hat{P}^t_{l,i} = P_{l,i}$ for $1 \leq i \leq t$. Notice that as we forecast further into the future, the NWP forecast will increase in uncertainty as the NWP is forecasted at time t to time $t + \tau$.

6.3.3 Validation

In order to determine the hyperparameters of the selected models, we will use holdout cross-validation. Hence, the training set will be divided into a sub-training set and a validation set. As seen in Fig. 6.8, the sub-training set consists of the data from 1st of January 2019 at 00:00 to 1st of August 2020 at 23:55 and the validation set consists of the data from 2nd of August 2020 at 00:00 to 5th of October 2020 at 09:55 where the number of wind power measurements is n_{subtrain} and $n_{\text{validation}}$ for the two sets, respectively. The wind power measurements in the sub-training set and validation set are $\{\boldsymbol{P}_t\}_{t=1}^{n_{\text{subtrain}}}$ and $\{\boldsymbol{P}_t\}_{t=n_{\text{subtrain}}+1}^{n_{\text{train}}}$, respectively.

The selected models will be trained on the sub-training set as in Section 6.3.1 where n_{train} is replaced with n_{subtrain} and tested on the validation set as in Section 6.3.2 using only the MSE and not the NMAE.

When the hyperparameters have been determined, the sub-training set and the validation set will be merged again and the models will be trained on the entire training set.

7. Experiments

In this chapter, experimental results obtained with the models introduced in Section 6.2 will be shown. This will include model selection as well as results on the test data.

7.1 Model Selection

Model selection refers to the process of determining hyperparameter settings for each of the models introduced in Section 6.2. The model selection will be based partly on an analysis of the data but mainly on using cross-validation to compare the performance of different hyperparameter settings in terms of MSE on the validation set. Due to time limitations and for simplicity, we will assume that one model fits all the sub-grids. This entails that during the validation phase, the hyperparameter settings which on average performs best for all the sub-grids are deemed to be the best choice of hyperparameters. Moreover, due to time limitations, some preliminary testing with the models using neural networks have been done only for individual sub-grids, particularly sub-grid DK1-1. Generally, the hyperparameter settings for each model which has the best performance in terms of MSE on the validation set are chosen and will then subsequently be used for evaluation on the test set.

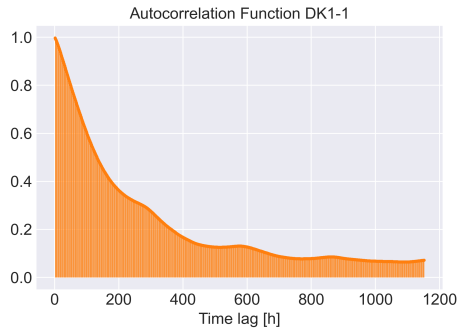
For the time series models described in Sections 6.2.2 and 6.2.4, the autocorrelation function (ACF) and partial autocorrelation function (PACF), defined in Appendix A.1, are used to determine which values of the hyperparameters we will consider during cross-validation. The hyperparameters of the s-ARIMAX and s-VARIMAX models are chosen in Section 7.1.1 and Section 7.1.2, respectively.

For the univariate and multivariate RNN described in Sections 6.2.3 and 6.2.5, the hyperparameters are manually adjusted and compared using cross-validation and this is done in Section 7.1.3 and Section 7.1.4, respectively.

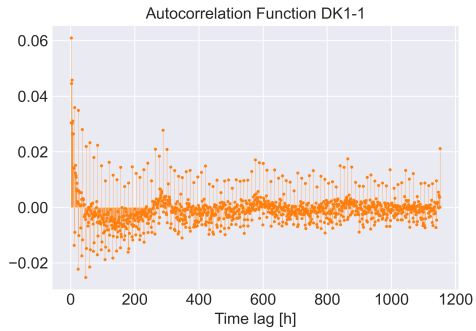
Finally, for the EMD-LSTM-ARMA model described in Section 6.2.6 the parameters for the LSTM and ARMA are fitted as for their respective univariate cases. The new hyperparameter, i.e. the threshold deciding whether an IMF is forecasted using a time series or a neural network is determined by a combination of an inspection of the ACF, the PACF, and cross-validation. The hyperparameters are chosen in Section 7.1.5.

7.1.1 s-ARIMAX

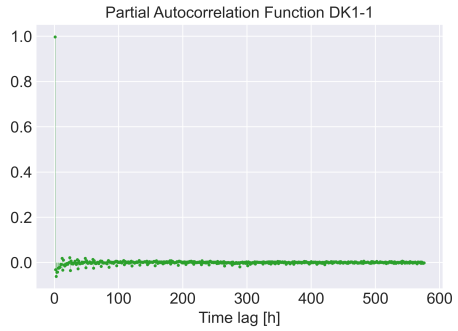
In the following, a brief analysis of the data will be given to motivate different hyperparameter settings. The hyperparameters of the s-ARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ model are p, d, q, p_s, d_s, q_s, and s. The analysis of the data will be based on computing some moments of the data, specifically the mean, the variance, and the ACF as well as the PACF. First the orders of differencing, d and d_s, will be determined. Subsequently, the ACF and the PACF will be used to determine candidate models in terms of the choice of p, q, p_s, q_s, and s [SS17, sec. 3.7 & 3.9].



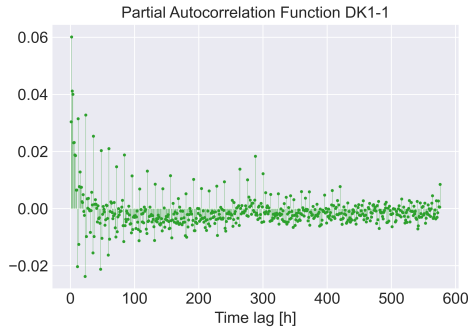
(a) Autocorrelation without differencing.



(b) Autocorrelation after first order differencing.



(c) Partial autocorrelation without differencing.



(d) Partial autocorrelation after first order differencing.

Figure 7.1: Autocorrelation and partial autocorrelation for sub-grid DK1-1 for the time period January 1st 2019 to October 10th 2021.

We begin by considering one sub-grid at a time and using the assumption that the same hyperparameter settings of a s-ARIMAX$(p, d, q) \times (p_s, d_s, q_s)_s$ can be used for all the individual sub-grids. The first step is to determine if differencing is needed. For this purpose, the ACF has been plotted, see Fig. 7.1a. It is seen that the ACF does not decay to zero fast as the time lag h increases. This is an indication of non-stationarity [SS17, p. 136]. Moreover, another indication of non-stationarity is if the ACF for disjoint partitions of the dataset are different and the sample mean and sample variance vary between the partitions. This is observed for the data, however has been left out for the sake of brevity. From the PACF in Fig. 7.1c, a persistency

effect is noticed as the partial autocorrelation at time lag 1 is close to one and could indicate that an autoregressive model of low order without differencing could be used for forecasting.

Based on the preceding results, we are inclined to attempt differencing the data. First we try a first order difference and as such we define

$$\nabla^1 P_{l,t} = P_{l,t} - P_{l,t-1} \tag{7.1}$$

for $t = 2, \ldots, n$ where $\{P_{l,t}\}_{t=1}^{n}$ is the wind power production time series for sub-grid l. This time series represents the rate of change in the wind power production at 5 minute intervals. The resulting ACF for sub-grid DK1-1 is shown in Fig. 7.1b. In contrast with the original time series, the ACF of the differenced time series on different partitions of the data are more similar by visual inspection. Moreover, the sample mean and variance for different partitions are closer for the differenced time series than for the original time series. In general, we deem that the data is closer to stationarity after first order differencing than the original time series.

Attempting, second-order differencing and seasonal differencing, it seems that a dependency arises when there is none. If e.g. we do seasonal differencing, a peak in the ACF is seen at time lag s no matter which seasonal lag s is used. This suggests over-differencing and therefore we deem that first-order differencing is better [SS17, p. 135].

Based on the preceding discussion, we will be considering models without differencing and models using first-order differencing. The next step is to determine the orders p, and q and potentially also p_s, q_s, and s. As mentioned, without differencing an autoregressive model of low order seems to be a possibility based on Figs. 7.1a and 7.1c. When using differencing, a diurnal effect seems to be present in both the ACF and the PACF, see Figs. 7.1b and 7.1d. We deem that based on the ACF and the PACF it could potentially be beneficial to use up to two days history in both the autoregressive part and the moving average part, however it is unclear exactly which orders would be best. Instead cross-validation will be used to find the orders yielding the best performance in terms of MSE on the validation set.

In Table 7.1, a wide variety of models with different hyperparameter settings are fitted to the sub-training set and the performance in terms of MSE is evaluated on the validation set. It is seen that all the fitted models have comparable performance in terms of MSE on the validation set. The model with the best performance in terms of MSE on the validation set is the ARIMAX$(12, 1, 0)$ model, i.e. a purely autoregressive model with first order differencing using one hour of wind power production history.

7.1.2 s-VARIMAX

The hyperparameters of the s-VARIMAX model is the same as those of the s-ARIMAX model. The experience gained in Section 7.1.1 can as such be used when selecting the hyperparameters of the multivariate time series model in this section. In Table 7.2, some results from training on the sub-training set and evaluating the performance in

Specifications	Average MSE
ARX(1)	19.48
ARX(576)	19.79
ARIMAX$(1, 1, 0)$	19.52
ARIMAX$(12, 1, 0)$	**19.43**
ARIMAX$(576, 1, 0)$	19.81
ARIMAX(72, 1, 12)	19.44
ARIMAX$(576, 1, 576)$	19.80
s-ARIMAX$(36, 1, 0) \times (2, 0, 0)_{288}$	19.74

Table 7.1: Average MSE for the validation data.

terms of MSE on the validation set can be seen. It is seen that the model with the best performance in terms of MSE on the validation set is the VARIMAX$(1, 1, 0)$ model, however the performance of this model is still worse than the ARIMAX$(12, 1, 0)$ model. Moreover, we notice that without differencing the performance in terms of MSE on the validation set is notably worse and the same can be said when using higher orders. The results shown in Table 7.2 are only a subset of the experiments which have been conducted in relation to this project.

Specifications	Average MSE
VARX(1)	55.49
VARIMAX(1, 1, 0)	**21.65**
VARIMAX(12, 1, 0)	26.16
VARIMAX(72, 1, 0)	39.87
VARIMAX$(72, 1, 12)$	53.27
s-VARIMAX$(36, 1, 0) \times (2, 0, 0)_{288}$	40.58

Table 7.2: Average MSE for the validation data.

7.1.3 Univariate RNN

In order to determine which hyperparameters are the best, manual tuning has been used. We have tested several different LSTM and GRU models with different batch sizes, learning rates, hidden units, number of layers, learning rate decays, and dropout rates. After the preliminary tests, we have found that similar hyperparameters for the GRU and the LSTM yield the best results. The learning rate has been fitted to $\eta = 1.2\text{e}{-04}$ for the LSTM neural networks and $\eta = 1.1\text{e}{-04}$ for the GRU neural networks. Additionally, all models use exponential learning rate decay of 0.7 and a batch size of 32. Furthermore, we find the best performance with a hidden unit dropout rate of 0.1. Finally, in terms of hidden units the best performance has been found using 3 recurrent layers with 512 units in each and a dense output layer. Tests for LSTM neural networks with different lengths of power production history can

be seen in Table 7.3. Additionally, a GRU neural network is fitted for the power production history length which resulted in the best performance.

Type	p	Univariate RNN	Multivariate RNN	EMD
LSTM	576	99.79	578.6	342.5
LSTM	288	47.81	622.7	133.6
LSTM	12	**19.47**	**448.7**	**14.63**
GRU	12	21.06	464.8	-

Table 7.3: Comparison of the neural network models for different amounts of wind power production history using LSTM networks. For the univariate and multivariate RNNs, the average MSE on the validation set is shown, while for the EMD-LSTM-ARMA model the MSE on sub-grid DK1-1 is shown. Moreover, for the univariate and multivariate RNNs a GRU network using one hour of wind power production history is also shown.

A learning curve for the univariate LSTM with $p = 12$ on sub-grid DK1-1 can be seen in Fig. 7.2. Due to the temporal dependency, the data is not shuffled. This results in the periodicity seen in the training loss. We see that the validation error decreases quickly at first and then starts to saturate before the procedure is terminated by the early stopping regularisation.

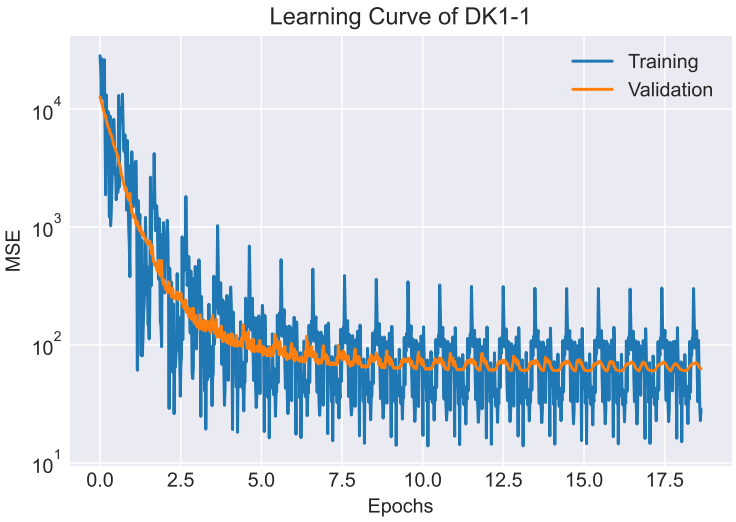


Figure 7.2: Learning curve for the univariate LSTM network with $p = 12$ on DK1-1.

Seeing as the LSTM neural network trained on one hour of power production history performs the best in terms of average MSE on the validation set, this is the model which will be used for testing.

7.1.4 Multivariate RNN

For the multivariate RNN, we have experimented with several different architectures and hyperparameters. We found the best performance using the architecture seen in Fig. 6.6 where the parallel layers are designed as in Section 7.1.3. However, for the multivariate RNN we have used a batch size of 128 and a learning rate of $\eta = 3.2\text{e}{-04}$ for the LSTM models and a learning rate of $\eta = 3.1\text{e}{-04}$ for the GRU models. The multivariate RNN architecture is very similar to the design of the parallel RNN where we have added a single multivariate dense layer the purpose of which is to include the spatial correlations. The performance of the multivariate model for different amounts of power production history can be seen in Table 7.3.

As in the univariate case, the LSTM neural network trained on one hour of power production history performs the best in terms of average MSE on the validation set. Thus, this model will be used for testing.

7.1.5 EMD-LSTM-ARMA

In this subsection, results from experiments with the model introduced in Section 6.2.6 is given. The EMD is computed for the wind power production data for each sub-grid and is then split into a training set, a validation set, a sub-training set, and a test set as in Section 6.3. The sample entropy is then computed for each IMF in each sub-grid. This is shown for sub-grid DK1-1 in Fig. 7.3. For the high entropy IMFs, an LSTM neural network is used to do forecasting, while for the low entropy IMFs and the residual, an ARMA(p, q) model is used. A complication arises with hyperparameter choices since now we are modelling each IMF for each sub-grid in parallel. For the low entropy IMFs, we will restrict our attention to using the same hyperparameter settings for all of the IMFs and the residual. However, the choice of hyperparameters for the ARMA(p, q) model can still be guided by an analysis of the IMFs and the residual. Moreover, for the high entropy IMFs we will also restrict our attention to using the same hyperparameter settings. This choice of hyperparameter settings for the neural network model will be made based on the validation results as for the univariate LSTM models.

Consider the EMD for sub-grid DK1-1. In Fig. 7.4, the ACF and the PACF of IMF number 18 is shown. The ACF and PACF indicates an AR(1) model is useful for forecasting this IMF [SS17, pp. 96-99]. Similar ACFs and PACFs are seen for IMFs number 17 and 19 as well as the residual. The same can also be seen for the last few IMFs and the residual for other sub-grids.

During preliminary testing, we observed that if the sample entropy threshold Ω was not sufficiently low, then an autoregressive model fitted on the first IMF below the threshold could yield poor results. Based on the preceding discussion and the preliminary testing, we have chosen the sample entropy threshold as $\Omega = 1\text{e}{-03}$ with an AR(1) model being used for the low entropy IMFs and the residual. Moreover, the batch size has been fixed at 64, the dropout rate in the hidden layers at 0.2, the learning rate to the interval $\eta \in [8\text{e}{-05}, 2\text{e}{-04}]$ depending on the amount of wind

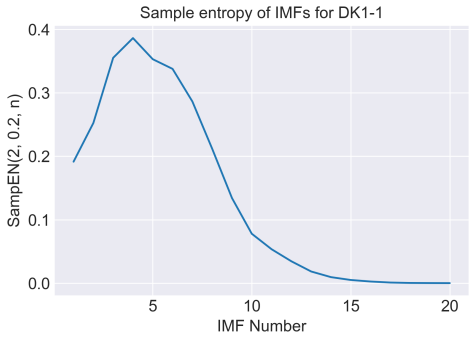
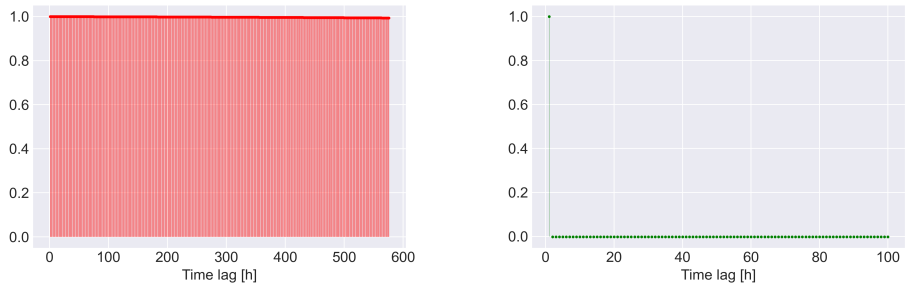


Figure 7.3: Sample entropy for sub-grid DK1-1.



(a) ACF. **(b)** PACF.

Figure 7.4: ACF and PACF of IMF number 18 for sub-grid DK1-1.

power production history, and the exponential learning rate decay at 0.7. As in the case of the univariate RNN, we have three LSTM layers with 512 hidden units in each and a dense layer which maps to a single output. In Table 7.3, the performance in terms of MSE on the validation set for different amount of power production history is shown. These experiments indicate that using one hour of wind power production history in the neural networks yields the best results and as such this model will be used for testing.

7.2 Results

In this section, we will present the results for each of the selected models. The selected models have been evaluated in accordance to Section 6.3.2 and the performance in terms of MSE and NMAE of each of the models is seen in Tables 7.4 and 7.5, respectively. It is seen that it differs between the sub-grids which model has the best performance. As a general pattern, we see that the ARIMAX$(12, 1, 0)$ model has the best performance in terms of MSE and NMAE for the most sub-grids and the best performance on average in terms of MSE and NMAE. Moreover, as a general trend we notice that ARIMAX$(12, 1, 0)$ performs slightly better than the persistence model

on each sub-grid.

We notice that the remaining models can achieve a good performance on individual sub-grids and for instance the s-VARIMAX$(1, 1, 0)$ has the best performance in terms of MSE on some of the first sub-grids, the EMD-LSTM-ARMA has a very good performance on DK1-7, and the univariate LSTM model also performs the best on DK1-9, DK2-5, and DK2-6 in terms of MSE. This shows the weakness of the one model fits all sub-grids methodology as the models using neural networks have a high variation in performance for different sub-grids. Notice for instance the performance in terms of MSE and NMAE of the EMD-LSTM-ARMA model for sub-grid DK1-7. As such, an improvement is expected if several models were fitted. One could expect the best results by individually selecting hyperparameters for each of the 21 sub-grids. However, that may not be needed as we see a single model is capable of performing well on several sub-grids. Such a model is the univariate LSTM model which performs well in terms of MSE on DK1-9, DK2-5, and DK2-6.

Additionally, a tendency is seen for the persistence method and the ARIMAX$(12, 1, 0)$ model to perform comparatively better in terms of NMAE than MSE than the other models which is evident since the ARIMAX$(12, 1, 0)$ model is best in terms of NMAE for more sub-grids than in terms of MSE. This may indicate that these methods are less sensitive to outliers in the data.

For the EMD-LSTM-ARMA model, we have applied a methodology assuming that one LSTM model fits all IMFs with SampEN above Ω for all sub-grids and one ARMA model fits all IMFs with SampEN below Ω for all sub-grids. This does not seem to be the optimum way of fitting the models since significant variations in performance can be observed both between the different sub-grids and between different IMFs in each sub-grid, the first of which is evident from Tables 7.4 and 7.5. A general tendency is noticed that the EMD-LSTM-ARMA model is significantly better in terms of MSE on the validation set compared to the test set. This is seen by comparing Tables 7.3 and 7.4. Specifically, the average MSE on the validation set with the EMD-LSTM-ARMA model is 5.512, while for the test set it is 2.040e+04. One consideration is that recalibrating the model as new data is received could be beneficial. Another consideration is that the model could possibly be improved if it was trained only on the past few months of data for each time of evaluation. These considerations are also interesting regarding the other parametric models in this project. [Tou+21; DCS07]

In Fig. 7.5, the performance of τ-ahead forecasts in terms of average MSE and average NMAE for the models used for testing with exception of the EMD-LSTM-ARMA model for $\tau = 1, \ldots, 288$ is shown. It is seen that the univariate LSTM model is the best in terms of MSE and NMAE with regards to generalising to other time horizons of forecasting than 5 minutes while persistence is the worst in terms of MSE and the VARIMAX$(1, 1, 0)$ is the worst in terms of NMAE. Alternatively, models could have been trained for specific time horizons to avoid problems of generalisation.

We have excluded the EMD-LSTM-ARMA model from Fig. 7.5 since the performance of this model in terms of average MSE and average NMAE comparably to the other methods is significantly worse which is mainly due to the poor performance in

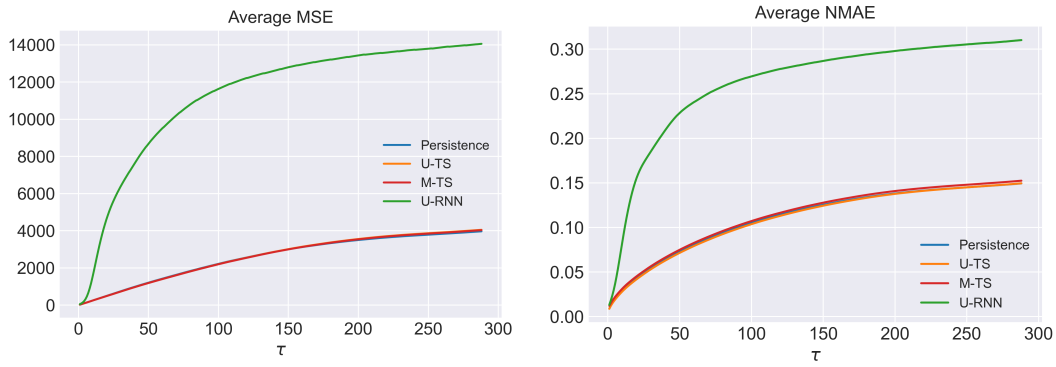
Sub-grid	Per.	U-TS	U-RNN	M-TS	M-RNN	EMD
DK1-1	39.47	38.67	93.57	38.44	574.1	**11.23**
DK1-2	59.71	59.24	138.9	**59.01**	543.5	86.52
DK1-3	14.10	**13.87**	27.60	16.11	98.95	23.04
DK1-4	22.22	21.98	29.36	21.95	685.1	**6.305**
DK1-5	163.4	156.7	402.3	157.6	2016	**30.77**
DK1-6	83.25	81.77	115.4	79.15	294.3	**18.83**
DK1-7	1.250	1.174	1.079	4.772	23.15	**0.1845**
DK1-8	2.366	2.178	5.281	7.298	43.71	**1.186**
DK1-9	1.302	1.288	1.165	10.13	29.76	**0.4272**
DK1-10	11.25	11.14	38.95	12.01	174.9	**3.400**
DK1-11	7.173	6.921	34.72	7.296	55.33	**4.545**
DK1-12	2.109	2.024	7.868	10.54	35.89	**0.3416**
DK1-13	6.473	6.175	13.50	15.99	96.13	**1.794**
DK1-14	9.752	9.569	41.49	13.94	135.1	**3.675**
DK1-15	1.447	1.344	10.33	22.51	30.82	**0.5480**
DK2-1	4.296	4.149	4.657	4.192	78.27	**1.238**
DK2-2	3.552	3.475	17.83	3.596	132.7	**1.226**
DK2-3	4.666	4.478	11.08	5.693	97.11	**1.059**
DK2-4	1.924	**1.857**	3.901	2.871	31.82	8.900
DK2-5	38.57	37.92	21.26	38.15	733.9	**8.960**
DK2-6	0.7983	0.7452	**0.7241**	1.760	15.53	2.303
Average	22.81	22.22	48.62	25.38	282.2	**10.31**

Table 7.4: MSE on the test set. Persistence is abbreviated Per., the univariate time series model ARIMAX$(36, 1, 0)$ is abbreviated U-TS, the univariate LSTM is abbreviated U-RNN, the multivariate time series model s-VARIMAX$(1, 1, 0)$ is abbreviated M-TS, the multivariate LSTM is abbreviated M-RNN, and finally the EMD-LSTM-ARMA model is abbreviated EMD. The lowest MSE in each row is written with boldface. DISCLAIMER!: The table have been changed since the explanatory text around them was written

terms of MSE and NMAE on sub-grid DK2-4. This in turn is mainly caused by poor performance for IMF14 in sub-grid DK2-4 for which the one-step ahead forecast MSE is 4.082e+05 on the test set. This is in contrast with the one-step ahead validation MSE for IMF14 on sub-grid DK2-4 which is only 4.128.

Sub-grid	Per.	U-TS	U-RNN	M-TS	M-RNN	EMD
DK1-1	8.34e−03	8.24e−03	2.11e−02	8.31e−03	5.20e−02	**4.02e−03**
DK1-2	1.00e−02	9.93e−03	2.10e−02	9.96e−03	4.16e−02	**8.67e−03**
DK1-3	7.13e−03	7.03e−03	1.40e−02	7.70e−03	2.51e−02	**6.99e−03**
DK1-4	7.28e−03	7.20e−03	1.15e−02	7.24e−03	5.81e−02	**3.26e−03**
DK1-5	5.17e−03	5.11e−03	1.42e−02	5.13e−03	3.05e−02	**2.45e−03**
DK1-6	1.04e−02	1.04e−02	1.96e−02	1.03e−02	3.30e−02	**5.14e−03**
DK1-7	7.56e−03	7.33e−03	9.71e−03	1.32e−02	4.14e−02	**2.94e−03**
DK1-8	6.71e−03	6.48e−03	1.09e−02	1.08e−02	3.73e−02	**3.88e−03**
DK1-9	9.82e−03	9.80e−03	1.18e−02	2.37e−02	6.09e−02	**4.97e−03**
DK1-10	7.93e−03	7.90e−03	2.11e−02	8.30e−03	4.23e−02	**3.98e−03**
DK1-11	8.58e−03	8.49e−03	2.54e−02	8.91e−03	3.38e−02	**4.62e−03**
DK1-12	9.11e−03	8.92e−03	2.41e−02	1.77e−02	4.72e−02	**3.59e−03**
DK1-13	8.27e−03	8.12e−03	1.61e−02	1.22e−02	3.93e−02	**3.70e−03**
DK1-14	7.80e−03	7.72e−03	2.15e−02	9.25e−03	3.62e−02	**3.96e−03**
DK1-15	6.33e−03	6.15e−03	2.45e−02	2.14e−02	3.80e−02	**3.14e−03**
DK2-1	1.02e−02	1.01e−02	1.35e−02	1.02e−02	4.89e−02	**4.34e−03**
DK2-2	1.29e−02	1.28e−02	3.93e−02	1.32e−02	9.77e−02	**6.08e−03**
DK2-3	9.39e−03	9.23e−03	1.76e−02	1.06e−02	5.11e−02	**4.05e−03**
DK2-4	9.73e−03	**9.54e−03**	2.18e−02	1.21e−02	5.83e−02	1.23e−02
DK2-5	9.97e−03	9.87e−03	8.87e−03	9.96e−03	6.94e−02	**4.98e−03**
DK2-6	1.47e−02	1.42e−02	1.88e−02	2.11e−02	8.75e−02	**1.01e−02**
Average	8.92e−03	8.79e−03	1.84e−02	1.20e−02	4.90e−02	**5.10e−03**

Table 7.5: NMAE on the test set. Persistence is abbreviated Per., the univariate time series model ARIMAX$(12, 1, 0)$ is abbreviated U-TS, the univariate LSTM is abbreviated U-RNN, the multivariate time series model VARIMAX$(1, 1, 0)$ is abbreviated M-TS, the multivariate LSTM is abbreviated M-RNN, and finally the EMD-LSTM-ARMA model is abbreviated EMD. The lowest NMAE in each row is written in boldface. DISCLAIMER!: The table have been changed since the explanatory text around them was written



(a) Average MSE.

(b) Average NMAE.

Figure 7.5: Performance of τ-ahead forecast for $\tau = 1, \ldots, 288$ in terms of average MSE and average NMAE for the models used for testing with exception of the EMD-LSTM-ARMA model. DISCLAIMER!: The figures have been changed since the explanatory text around them was written

In Fig. 7.6, an example of a one-step ahead forecast on one day in the test set is given for the persistence method, the ARIMAX$(12, 1, 0)$ model, and the univariate LSTM model. It is seen that the forecasts follow the measured data closely, however a small tendency is seen for the univariate LSTM model to be worse than the other two forecast methods.

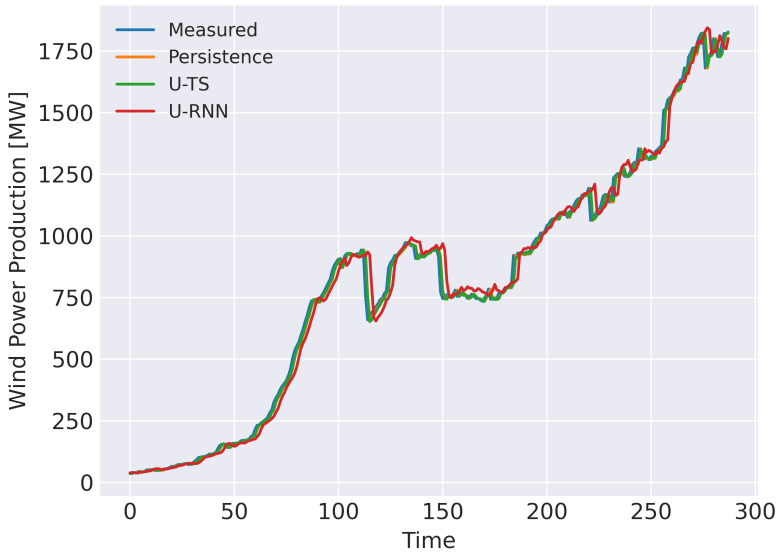


Figure 7.6: One-step ahead forecast for 1 day in the test set comparing the measured data to the forecast of the persistence method, the ARIMAX$(12, 1, 0)$ abbreviated U-TS, and the univariate LSTM model abbreviated U-RNN.

8. Conclusion

The goal of this project has been to forecast the Danish wind power production and we sought to do this by answering the question:

What accuracy can be achieved for very short-term forecasts of the Danish wind power production and how do the forecasts generalise to short-term and medium-term forecasting?

In order to implement models which could be used to answer this question, we investigated the theory of time series analysis, recurrent neural networks (RNNs), and the empirical mode decomposition (EMD).

The theory has been used to implement different models for forecasting the wind power production in Denmark. We have implemented both autoregressive moving average (ARMA) and vector ARMA models as well as modelling each sub-grid in Denmark both separately and jointly using RNNs. Finally, we have implemented an EMD model where the wind power production data was decomposed into intrinsic mode functions (IMFs) using the EMD and then used either an ARMA model or a RNN model to forecast the IMFs. All of these models have been compared to each other as well as a persistence method which was used as a baseline.

The models have been optimised for a one-step ahead prediction, i.e. a 5 minute forecast. Looking at the 5 minute forecast, we have seen that on average the ARMA model has the best performance in terms of MSE and NMAE. Generally, we can conclude that including spatial information in both the ARMA and RNNs do not improve the results and similarly we do not see improved results using the EMD.

We have used a one model fits all approach where the model hyperparameters were set to the same for each sub-grid. Additionally, for the EMD method, one hyperparameter setting has been used for the univariate RNN and one for the ARMA model, i.e. a total of two models for between 19 and 22 different IMFs including the residual per sub-grid.

Generally, we have found that a single neural network model could not forecast the

wind power production for every sub-grid. This could be seen by the high variation in the performance of the neural network models on different sub-grids. This problem also extends to the IMFs for the EMD-LSTM-ARMA model where the performance for each IMF has a high variation.

In terms of generalisation, we have seen the best generalisation ability to longer forecast horizons using the RNN that models the sub-grids separately. This model has achieved a relatively flat MSE and NMAE curve until it reached a 20 hour forecast after which the performance of the model decayed. However, this RNN model still has the best performance of the models after the 20 hour mark. It is also seen for the RNN model, which models the sub-grids jointly, that it generalises better to longer forecast horizons than some of the other models, however the performance is worse than for the RNN model which models the sub-grids separately.

9. Future Work

In this project, we have considered different models for 5 minute ahead wind power forecasting of the Danish power grid.

In the treatment of the available NWP data, the midpoint of each sub-grid was chosen as the NWP for the entire sub-grid. However, improvements in the models could perhaps be obtained by including either all the NWP data in each sub-grid or by use of a weighted average of the NWP data with respect to the number of wind mills close to the point for which the NWP data is computed. In addition, studying the significance and effects of the NWP data could be of interest in extension to the work presented in this project.

Additionally, interpolation could be used on the NWP data such that we would get a corresponding NWP for each wind power sample which would then be given to the models instead of using the most recent NWP. Since the NWPs do not change instantly between each sample but instead change continually between each sample, using interpolation on the NWPs might yield an improved performance of the models.

One of the main assumptions used when fitting the models in this project is the one model fits all assumption, i.e. for all sub-grids the same hyperparameters are used. However, from the results in Section 7.2, we see that when using neural network models better results can be obtained by fitting several different models. Hence, we suspect that improved performances of the models could be gained by either fitting 21 different models, i.e. a different model for each sub-grid, or by clustering the sub-grids based on some criterion such as the number of wind mills in the sub-grid or some characteristics of the surroundings in sub-grid and then fit a different model for each of the clusters.

An investigation regarding how often the models should be recalibrated is of interest if the models are to be used in practice. Furthermore, including a recalibration for the models might improve the performance of the models during testing. Additionally, it is of interest to investigate what part of the historical data should be used to train the models to perform best during testing.

In this project, we have trained models for 5 minute ahead forecasting and examined how well these generalise to longer forecasts and in relation to this, it could be interesting to see how well these models would perform if they had been trained for different time horizons such as 1 or 6 hours.

A. Mathematical Preliminaries

In this appendix, mathematical preliminaries required for reading Chapters 2 and 3 are given. The appendix begins by defining the first and second order moments of stochastic processes and a stationarity property called weak sense stationarity (WSS) is introduced. Subsequently, properties of the minimum mean square error estimator are given. This will include the projection theorem for normally distributed variables, the orthogonality principle, and its relation to the minimum mean square error estimator and the linear minimum mean square error estimator.

A.1 First and Second Order Moments

Consider a multivariate stochastic process $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ where $\boldsymbol{x}_t = (x_{1,t}, \dots, x_{k,t})^T$ is in $\mathbb{R}^k$ for $t \in \mathbb{Z}$.

Definition A.1 (Mean Function)
The mean sequence of $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$, denoted $\boldsymbol{\mu}_{\boldsymbol{x}}(t)$, is defined as

$$\boldsymbol{\mu}_{\boldsymbol{x}}(t) = \mathbb{E}[\boldsymbol{x}_t] \tag{A.1}$$

for $t \in \mathbb{Z}$. [SS17, p. 15]

Definition A.2 (Cross-Covariance Function)
The cross-covariance function of $\{x_{i,t}\}_{t\in\mathbb{Z}}$ and $\{x_{j,t}\}_{t\in\mathbb{Z}}$ for $i, j = 1, \dots, k$ is defined as

$$\gamma_{x_i,x_j}(t,s) = \mathrm{Cov}(x_{i,t}, x_{j,s}) = \mathbb{E}\Big[\big(x_{i,t} - \mu_{x_i}(t)\big)\big(x_{j,s} - \mu_{x_j}(s)\big)\Big] \tag{A.2}$$

for $t, s \in \mathbb{Z}$. [SS17, p. 16]

Definition A.3 (Cross-Correlation Function)
The cross-correlation function of $\{x_{i,t}\}_{t\in\mathbb{Z}}$ and $\{x_{j,t}\}_{t\in\mathbb{Z}}$ for $i,j=1,\ldots,k$ is defined as

$$r_{x_i,x_j}(t,s)=\mathbb{E}\big[x_{i,t}x_{j,s}\big]=\frac{\gamma_{x_i,x_j}(t,s)}{\sqrt{\gamma_{x_i,x_j}(t,t)\gamma_{x_i,x_j}(s,s)}} \tag{A.3}$$

for $t,s\in\mathbb{Z}$. [SS17, p. 18]

For a one-dimensional stochastic process, the cross-correlation function simplifies to the so-called autocorrelation function (ACF).

Definition A.4 (Weak Sense Stationary Stochastic Process)
If the mean and covariance functions for a k-dimensional stochastic process $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ can be expressed as

$$\boldsymbol{\mu}_{\boldsymbol{x}}(t)=\boldsymbol{\mu}_{\boldsymbol{x}},\quad t\in\mathbb{Z}, \tag{A.4}$$

$$\gamma_{x_i,x_j}(t,s)=g_{x_i,x_j}(|s-t|),\quad i,j=1,\ldots,k,\quad t,s\in\mathbb{Z}, \tag{A.5}$$

respectively where $\boldsymbol{\mu}_{\boldsymbol{x}}\in\mathbb{R}^k$ is a constant and $g_{x_i,x_j}:\mathbb{Z}\to\mathbb{R}$, then $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ is a WSS stochastic process. [SS17, p. 20]

For WSS processes, the cross-covariance and cross-correlation functions are fully described as a function of the time lag, $|s-t|$, and hence for WSS processes, we have the following definitions.

Definition A.5 (Cross-Covariance Matrix)
Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be a k-dimensional WSS stochastic process with mean vector $\boldsymbol{\mu}_{\boldsymbol{x}}=(\mu_{x_1},\ldots,\mu_{x_k})^T$. The cross-covariance at time lag h between elements $\{x_{i,t}\}_{t\in\mathbb{Z}}$ and $\{x_{j,t}\}_{t\in\mathbb{Z}}$ is given by

$$\gamma_{x_i,x_j}(h)=\mathbb{E}[(x_{i,t}-\mu_{x_i})(x_{j,t+h}-\mu_{x_j})],\quad i,j=1,\ldots,k, \tag{A.6}$$

yielding the cross-covariance matrix at time lag h [Box+16, p. 506]

$$\boldsymbol{\Gamma}(h)=\begin{bmatrix}\gamma_{x_1,x_1}(h) & \gamma_{x_1,x_2}(h) & \ldots & \gamma_{x_1,x_k}(h)\\ \gamma_{x_2,x_1}(h) & \gamma_{x_2,x_2}(h) & \ldots & \gamma_{x_2,x_k}(h)\\ \vdots & \vdots & \ddots & \vdots\\ \gamma_{x_k,x_1}(h) & \gamma_{x_k,x_2}(h) & \ldots & \gamma_{x_k,x_k}(h)\end{bmatrix}. \tag{A.7}$$

Definition A.6 (Cross-Correlation Matrix)
Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be a k-dimensional WSS stochastic process with cross-covariance function γ_{x_i,x_j}. The cross-correlation between $x_{i,t}$ and $x_{j,t}$ at time lag h is given by

$$r_{x_i,x_j}(h) = \mathbb{E}[x_{i,t}x_{j,t+h}] = \frac{\gamma_{x_i,x_j}(h)}{\sqrt{\gamma_{x_i,x_i}(0)\gamma_{x_j,x_j}(0)}}, \quad i,j = 1,\dots,k, \tag{A.8}$$

yielding the cross-correlation matrix at time lag h

$$\boldsymbol{R}(h) = \boldsymbol{V}^{-\frac{1}{2}}\boldsymbol{\Gamma}(h)\boldsymbol{V}^{-\frac{1}{2}} \tag{A.9}$$

where $\boldsymbol{V}^{-\frac{1}{2}} = \operatorname{diag}\big(\gamma_{x_1,x_1}(0)^{-\frac{1}{2}},\dots,\gamma_{x_k,x_k}(0)^{-\frac{1}{2}}\big)$. [Box+16, pp. 506-507]

Another function which is useful for analysis of time series data sampled from a WSS stochastic process is the partial autocorrelation function (PACF). The definition is given in Definition A.7.

Definition A.7 (Partial Autocorrelation Function)
The PACF of a zero mean WSS stochastic process $\{x_t\}_{t\in\mathbb{Z}}$ at time lag 1 is defined as

$$\rho(1) = \mathbb{E}[x_{t+1}x_t], \tag{A.10}$$

and for time lag $h = 2,\dots$ it is defined as

$$\rho(h) = \mathbb{E}[(x_{t+h} - \hat{x}_{t+h})(x_t - \hat{x}_t)] \tag{A.11}$$

where

$$\hat{x}_{t+h} = \sum_{j=1}^{h-1}\beta_j x_{t+h-j}, \quad \text{and} \quad \hat{x}_t = \sum_{j=1}^{h-1}\beta_j x_{t+j} \tag{A.12}$$

are the linear regressions of $\{x_{t+1},\dots,x_{t+h-1}\}$ onto x_{t+h} and x_t, respectively. [SS17, p. 97]

As can be seen from Definition A.7, the PACF of a WSS stochastic process $\{x_t\}_{t\in\mathbb{Z}}$ at time lag h is the correlation between x_{t+h} and x_t with the linear dependency on the variables between them, i.e. $x_{t+1},\dots,x_{t+h-1}$, removed. [SS17, p. 97]

A.2 Minimum Mean Square Error Estimator

Let $\boldsymbol{x}$ be an n-dimensional random vector and let $\boldsymbol{y}$ be a k-dimensional random vector. The minimum mean square error estimator (MMSEE) of $\boldsymbol{y}$ given $\boldsymbol{x}$ is the estimator $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}(\boldsymbol{x})$ which minimises the mean squared error, i.e. $\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2]$. The minimum mean square error estimator is the conditional expectation, i.e. [Kay93, pp. 312-313]

$$\hat{y}_{\text{MMSEE}} = \mathbb{E}[\boldsymbol{y}|\boldsymbol{x}]. \tag{A.13}$$

The MMSEE is impractical since it relies on prior knowledge of the distribution of $\boldsymbol{y}|\boldsymbol{x}$. Instead, a linear estimator which minimises the mean square error can be used. The estimator which minimises the mean square error among all affine estimates, i.e. of the form $\boldsymbol{y} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$, is called the linear minimum mean square error estimator (LMMSEE) and is given as [Kay93, pp. 380-382]

$$\hat{y}_{\text{LMMSEE}} = \boldsymbol{\mu}_{\boldsymbol{y}} + \boldsymbol{\Sigma}_{\boldsymbol{yx}} \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} \big(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}\big) \tag{A.14}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ is the covariance matrix of $\boldsymbol{x}$ and $\boldsymbol{\Sigma}_{\boldsymbol{yx}}$ is the cross-covariance matrix between $\boldsymbol{y}$ and $\boldsymbol{x}$. In the Gaussian case, the distribution of $\boldsymbol{y}|\boldsymbol{x}$ is given in Theorem A.8.

Theorem A.8 (Projection Theorem for Gaussian Variables)
Let $(\boldsymbol{y}^T, \boldsymbol{x}^T)^T$ be a normally distributed random vector with mean vector and covariance matrix

$$\begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{y}} \\ \boldsymbol{\mu}_{\boldsymbol{x}} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{x}} & \boldsymbol{\Sigma}_{\boldsymbol{yx}} \\ \boldsymbol{\Sigma}_{\boldsymbol{xy}} & \boldsymbol{\Sigma}_{\boldsymbol{y}} \end{bmatrix}, \tag{A.15}$$

respectively. Then $\boldsymbol{y}|\boldsymbol{x}$ is normally distributed with mean

$$\mathbb{E}[\boldsymbol{y}|\boldsymbol{x}] = \boldsymbol{\mu}_{\boldsymbol{y}} + \boldsymbol{\Sigma}_{\boldsymbol{yx}} \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}) \tag{A.16}$$

and covariance

$$\text{Var}[\boldsymbol{y}|\boldsymbol{x}] = \boldsymbol{\Sigma}_{\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{yx}} \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{yx}}^{T}. \tag{A.17}$$

Furthermore, the error $(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}|\boldsymbol{x}])$ and $\boldsymbol{x}$ are independent, i.e. the error is orthogonal to the data $\boldsymbol{x}$. [MT10, p. 45]

For proof see [Kay93, pp. 323-325]. This result shows that for normally distributed random variables the MMSEE is linear in the data and as such the MMSEE and the LMMSEE coincide.

The orthogonality statement in Theorem A.8 can be generalised. In fact, it holds that $(\boldsymbol{y} - \mathbb{E}[\boldsymbol{y}|\boldsymbol{x}])$ is independent of any function of the data $\boldsymbol{x}$. If the normality assumption is dropped, then the function is limited to the affine functions. These statements are formalised in Propositions A.9 and A.10. [SS17, p. 494-496]

Proposition A.9 (Orthogonality Principle for MMSEE)
Let $\boldsymbol{y}$ and $\boldsymbol{x}$ be random vectors. The MMSEE $\hat{\boldsymbol{y}} = \mathbb{E}[\boldsymbol{y}|\boldsymbol{x}]$ of $\boldsymbol{y}$ given $\boldsymbol{x}$ fulfils the orthogonality principle for any function $f : \mathbb{R}^n \to \mathbb{R}$, i.e.

$$\mathbb{E}\big[(\boldsymbol{y} - \hat{\boldsymbol{y}})f(\boldsymbol{x})\big] = \mathbf{0}. \tag{A.18}$$

Proposition A.10 (Orthogonality Principle for LMMSEE)
Let $\boldsymbol{y}$ and $\boldsymbol{x}$ be random vectors. The LMMSEE $\hat{\boldsymbol{y}} = \boldsymbol{\mu}_{\boldsymbol{y}} + \boldsymbol{\Sigma}_{\boldsymbol{yx}}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\big(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}\big)$ of $\boldsymbol{y}$ given $\boldsymbol{x}$ fulfils the orthogonality principle for any affine function $f : \mathbb{R}^n \to \mathbb{R}$, i.e.

$$\mathbb{E}\Big[\big(\boldsymbol{y} - \hat{\boldsymbol{y}}\big)f(\boldsymbol{x})\Big] = 0. \tag{A.19}$$

where $f(\boldsymbol{x}) = f_0 + \boldsymbol{f}^T\boldsymbol{x}$ for $f_0 \in \mathbb{R}$ and $\boldsymbol{f}^T \in \mathbb{R}^n$.

B. Unconstrained Optimisation

In this appendix, descent methods and newton like methods of unconstrained optimisation are introduced. Unconstrained optimisation refers to problems of the form

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^k} \quad J(\boldsymbol{\theta}) \tag{B.1}$$

where $J : \mathbb{R}^k \to \mathbb{R}$ is the objective function and $\boldsymbol{\theta} \in \mathbb{R}^k$ is the parameters to be optimised. The objective is then to determine the parameter vector $\boldsymbol{\theta}$ which minimises the objective function. Let us assume that J is a nonlinear function in the parameters $\boldsymbol{\theta}$, then methods can be employed which are based on updating the parameter estimates iteratively by computing a step for the parameters at each iteration. This leads to descent methods which will be introduced in Appendix B.1. Subsequently, we will discuss Newton-like methods in Appendix B.2 herein Newton's method, and quasi-Newton methods.

B.1 Descent Methods

Let $\boldsymbol{\theta} \in \Theta^k$ denote the parameters of a parametric model with parameter space Θ^k which is assumed to be all of $\mathbb{R}^k$ in unconstrained optimisation. Then an update or a step is given as

$$\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} + \boldsymbol{s}^{(j)} \tag{B.2}$$

where $\boldsymbol{\theta}^{(j)}$ is the parameter estimate at iteration j and $\boldsymbol{s}^{(j)} = \boldsymbol{s}(\boldsymbol{\theta}^{(j)})$ is the step at iteration j. Given an objective function $J : \Theta^k \to \mathbb{R}$ which should be minimised, the so-called descent methods attempts to solve this optimisation problem by designing steps in which

$$J(\boldsymbol{\theta}^{(j+1)}) < J(\boldsymbol{\theta}^{(j)}) \tag{B.3}$$

except when $\boldsymbol{\theta}^{(j)}$ minimises J. A step direction which fulfils this criterion is called a descent direction for J at $\boldsymbol{\theta}^{(j)}$. [BV04, p. 463]

In practice, the step in Eq. (B.2) is scaled in an attempt to minimise the objective function in the step direction, i.e.

$$\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} + \lambda^{(j)}\boldsymbol{s}^{(j)} \tag{B.4}$$

where $\lambda^{(j)} > 0$. The parameter $\lambda^{(j)}$ is called the step size and it can either be fixed as a constant or found using line search. For exact line search, the step size is found as

$$\lambda = \arg\min_{a \geq 0} J(\boldsymbol{\theta} + a\boldsymbol{s}). \tag{B.5}$$

However, this is typically not feasible and instead inexact line search methods are used, examples of which are backtracking line search and Wolfe line search [BV04, pp. 463-465][Fle87, p. 27].

B.2 Newton-like Methods

In this section, Newton's method and variations thereof are introduced. Newton-like methods are descent methods which are motivated by a second order approximation of the objective function. For Newton's method, the step is given as

$$\boldsymbol{s}^{(j)} = -\boldsymbol{G}^{(j)^{-1}}\boldsymbol{g}^{(j)} \tag{B.6}$$

where $J \in C^2$ is the objective function, $\boldsymbol{g}^{(j)}$ and $\boldsymbol{G}^{(j)}$ is the gradient vector and the Hessian matrix of J at the point $\boldsymbol{\theta}^{(j)}$, respectively. The method can be motivated by considering the second order Taylor expansion of J around the point $\boldsymbol{\theta}$

$$\hat{J}(\boldsymbol{\theta} + \boldsymbol{s}) = J(\boldsymbol{\theta}) + \boldsymbol{g}^T\boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T\boldsymbol{G}\boldsymbol{s} \tag{B.7}$$

where $\hat{J}$ is the Taylor approximation, the error of which is the remainder term as specified in Taylor's theorem [TAK18, p. 28]. Using standard vector calculus, Eq. (B.7) is minimised with respect to $\boldsymbol{s}$ by the step in Eq. (B.6). This means that in Newton's method the step which is used, is the one which minimises a second order approximation of the likelihood function at the point of the previous parameter estimate. [BV04, p. 484] [SS17, pp. 119-120]

Moreover, let us assume that the Hessian matrix is positive definite. In this case, it follows that

$$\boldsymbol{g}^{(j)^T}\boldsymbol{s}^{(j)} = -\boldsymbol{g}^{(j)^T}\boldsymbol{G}^{(j)^{-1}}\boldsymbol{g}^{(j)} > 0 \tag{B.8}$$

unless $\boldsymbol{g}^{(j)} = \boldsymbol{0}$ [BV04, p. 484]. The positive definiteness of the Hessian matrix implies that the objective function is strictly convex and that the Newton step is a descent direction. [BV04, p. 463]

B.2.1 Gauss-Newton

The Gauss-Newton method can be motivated through two different arguments. Firstly, it can be derived by considering a residual sum of squares objective function and then using a first order Taylor expansion. Secondly, it can be interpreted as an approximation of the Newton-Rhapson method.

We begin the treatment of the first interpretation by considering the residual sum of squares objective function

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{t=1}^{n} u_t^2(\boldsymbol{\theta}) \tag{B.9}$$

where u_t is the residual for the t'th datapoint. Using Gauss-Newton, u_t is linearised using a first order Taylor approximation

$$\hat{u}_t(\boldsymbol{\theta} + \boldsymbol{s}) := u_t(\boldsymbol{\theta}) + \boldsymbol{\nabla} u_t(\boldsymbol{\theta})^T \boldsymbol{s} \tag{B.10}$$

and thereby

$$\hat{J}(\boldsymbol{\theta} + \boldsymbol{s}) := \sum_{t=1}^{n} \big(u_t(\boldsymbol{\theta}) + \boldsymbol{\nabla} u_t(\boldsymbol{\theta})^T \boldsymbol{s}\big)^2. \tag{B.11}$$

By the result from ordinary least squares, the value of $\boldsymbol{s}$ which minimises Eq. (B.11) given $\boldsymbol{\theta} = \boldsymbol{\theta}^{(j)}$ is

$$\boldsymbol{s}^{(j+1)} = -\Big(\sum_{t=1}^{n} \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)}) \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)})^T\Big)^{-1} \Big(\sum_{t=1}^{n} \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)}) u_t(\boldsymbol{\theta}^{(j)})\Big) \tag{B.12}$$

which is the Gauss-Newton step in the case of the residual sum of squares objective function. [Box+16, pp. 226-227][BV04, p. 520]

The second interpretation arises by replacing the negative of the Hessian matrix in Eq. (B.6) by the Fisher information matrix [MT10, pp. 18-19]

$$\boldsymbol{M} = \mathbb{E}[-\boldsymbol{G}] = \mathbb{E}[\boldsymbol{g}\boldsymbol{g}^T] \tag{B.13}$$

where the second equality holds assuming differentiation and integration can be interchanged. The Gauss-Newton step is then

$$\boldsymbol{s}^{(j+1)} = {\boldsymbol{M}^{(j)}}^{-1} \boldsymbol{g}^{(j)}. \tag{B.14}$$

In practice, the Fisher information matrix is generally estimated from the sample [GM74]

$$\hat{\boldsymbol{M}}^{(j)} = \boldsymbol{g}^{(j)} \boldsymbol{g}^{(j)^T}. \tag{B.15}$$

In example B.1, an application of the Gauss-Newton method to parameter estimation for an ARMA(p, q) model is considered.

Example B.1 (Gauss-Newton for ARMA(p, q) models)
Consider the ARMA(p, q) model as in Eq. (2.2)

$$x_t = \sum_{j=1}^{p} \phi_j x_{t-j} - \sum_{i=1}^{q} \psi_i u_{t-i} + u_t \tag{B.16}$$

where it is assumed that $\mathbb{E}[x_t] = 0$. Rearranging to write the difference equation in terms of the noise process yields

$$u_t(\boldsymbol{\theta}) = x_t - \sum_{j=1}^{p} \phi_j x_{t-j} + \sum_{i=1}^{q} \psi_i u_{t-i}(\boldsymbol{\theta}) \tag{B.17}$$

where $\boldsymbol{\theta} = (\phi_1, \ldots, \phi_p, \psi_1, \ldots, \psi_q)^T$. In this example, the objective function in consideration will be the residual sum of squares. Specifically, we will consider an approximation of the residual sum of squares in which we condition on $\{x_1, \ldots, x_p\}$ and set $u_p = u_{p-1} = \cdots = u_{1-q} = 0$. This approximation is called the conditional least squares and is given as

$$S_c(\boldsymbol{\theta}) = \frac{1}{2} \sum_{t=p+1}^{n} u_t^2(\boldsymbol{\theta}). \tag{B.18}$$

As derived in Eq. (B.12), the Gauss-Newton step in this case is then

$$\boldsymbol{s}^{(j+1)} = -\Big(\sum_{t=p+1}^{n} \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)}) \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)})^T \Big)^{-1} \Big(\sum_{t=p+1}^{n} \boldsymbol{\nabla} u_t(\boldsymbol{\theta}^{(j)}) u_t(\boldsymbol{\theta}^{(j)}) \Big) \tag{B.19}$$

where each of the partial derivatives in Eq. (B.19) can be derived using Eq. (B.17) and the result is

$$\frac{\partial u_t}{\partial \phi_j} = -x_{t-j}, \tag{B.20}$$

$$\frac{\partial u_t}{\partial \psi_i} = u_{t-i} + \psi_i \frac{\partial u_{t-i}}{\partial \psi_i}. \tag{B.21}$$

This allows us to compute the Gauss-Newton step in Eq. (B.14). [SS17, pp. 120-122]

B.2.2 Quasi-Newton

While Newton's method provides an effective descent method with desirable properties given the assumption of positive definiteness of the Hessian, it is restrictive since it requires a closed form solution of the Hessian matrix. Quasi-Newton methods attempts to alleviate this issue by numerically approximating the inverse of the Hessian matrix. [Fle87, pp. 44-49]

Let us assume that an objective function $J : \mathbb{R}^k \to \mathbb{R}$ and its gradient vector, denoted $\boldsymbol{g}$, is known. Moreover, as before let $\boldsymbol{G}$ denote the Hessian matrix and let $\boldsymbol{H}$ denote the inverse of the Hessian matrix. The purpose now is to use J and $\boldsymbol{g}$ to design an approximation $\hat{\boldsymbol{H}}$ to use instead of the inverse of the Hessian matrix in Eq. (B.6). Different methods for this approximation have been designed each holding

different properties. Some desirable properties of the designed matrix is symmetry and positive definiteness. These properties are desirable since the Hessian matrix is a symmetric matrix and positive definiteness implies a descent direction. Additionally, low computational complexity is preferred and ultimately the purpose is to achieve fast global convergence. [Fle87, pp. 49-57]

An important class of quasi-Newton methods are methods in which the approximation of $\boldsymbol{H}$ is formed as a symmetric and positive definite matrix which is updated from iteration to iteration. Designing a method then amounts to designing an updating formula for the approximation of $\boldsymbol{H}$. This will be done by attempting to use information gained about the second derivatives from the previous iteration. Before introducing updating formula, we define the differences for the jth iteration

$$\boldsymbol{\delta}^{(j)} = \lambda^{(j)}\boldsymbol{s}^{(j)} = \boldsymbol{\theta}^{(j+1)} - \boldsymbol{\theta}^{(j)}, \tag{B.22}$$

$$\boldsymbol{\kappa}^{(j)} = \boldsymbol{g}^{(j+1)} - \boldsymbol{g}^{(j)}. \tag{B.23}$$

Using a first order Taylor expansion of $\boldsymbol{g}$ at the point $\boldsymbol{\theta}^{(j)}$ and step $\boldsymbol{\delta}^{(j)}$ yields

$$\boldsymbol{\kappa}^{(j)} = \boldsymbol{G}^{(j)}\boldsymbol{\delta}^{(j)} + R \tag{B.24}$$

where R is the remainder term in Taylor's theorem [TAK18, p. 28]. This result leads to the so-called quasi-Newton condition which is a condition used for the update of the approximation of the Hessian so as to fulfil Eq. (B.24), i.e.

$$\hat{\boldsymbol{H}}^{(j+1)}\boldsymbol{\kappa}^{(j)} = \boldsymbol{\delta}^{(j)}. \tag{B.25}$$

Given the condition Eq. (B.25), an update formula for $\hat{\boldsymbol{H}}$ can be defined by e.g. using a rank one or a rank two update. [Fle87, pp. 50-51]

Consider a rank one update

$$\hat{\boldsymbol{H}}^{(j+1)} = \hat{\boldsymbol{H}}^{(j)} + a\boldsymbol{w}\boldsymbol{w}^T \tag{B.26}$$

such that the quasi-Newton condition implies

$$\boldsymbol{\delta}^{(j)} = \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)} + a\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{\kappa}^{(j)}. \tag{B.27}$$

It follows that $\boldsymbol{w}$ must be proportional to $\boldsymbol{\delta}^{(j)} - \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}$, thus we let $\boldsymbol{w} = \boldsymbol{\delta}^{(j)} - \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}$. By Eq. (B.27), this definition of $\boldsymbol{w}$ implies that a must be defined as to satisfy the condition $a\boldsymbol{w}^T\boldsymbol{\kappa}^{(j)} = 1$. Hence, the rank one update is

$$\hat{\boldsymbol{H}}^{(j+1)} = \hat{\boldsymbol{H}}^{(j)} + \frac{(\boldsymbol{\delta}^{(j)} - \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)})(\boldsymbol{\delta}^{(j)} - \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)})^T}{(\boldsymbol{\delta}^{(j)} - \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)})^T\boldsymbol{\kappa}^{(j)}}. \tag{B.28}$$

Two disadvantages of the rank one update is that the positive definiteness is not guaranteed and the denominator may become zero yielding an undefined update. [Fle87, pp. 51-53]

Consider instead a rank two update

$$\hat{\boldsymbol{H}}^{(j+1)} = \hat{\boldsymbol{H}}^{(j)} + a\boldsymbol{w}\boldsymbol{w}^T + b\boldsymbol{v}\boldsymbol{v}^T \tag{B.29}$$

such that the quasi-Newton condition implies

$$\boldsymbol{\delta}^{(j)} = \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)} + a\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{\kappa}^{(j)} + b\boldsymbol{v}\boldsymbol{v}^T\boldsymbol{\kappa}^{(j)}. \tag{B.30}$$

In contrary to the rank one update, the vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ are no longer uniquely defined. For the so-called David-Fletcher-Powell formula, the choices are $\boldsymbol{w} = \boldsymbol{\delta}^{(j)}$ and $\boldsymbol{v} = \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}$. With these choices, we must have that $a\boldsymbol{w}^T\boldsymbol{\kappa}^{(j)} = 1$ and $b\boldsymbol{v}^T\boldsymbol{\kappa}^{(j)} = -1$ for Eq. (B.30) to hold. Solving for the factors a and b yields the David-Fletcher-Powell update formula

$$\hat{\boldsymbol{H}}_{DFP}^{(j+1)} = \hat{\boldsymbol{H}}^{(j)} + \frac{\boldsymbol{\delta}^{(j)}\boldsymbol{\delta}^{(j)^T}}{\boldsymbol{\delta}^{(j)^T}\boldsymbol{\kappa}^{(j)}} - \frac{\hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}\boldsymbol{\kappa}^{(j)^T}\hat{\boldsymbol{H}}^{(j)}}{\boldsymbol{\kappa}^{(j)^T}\hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}} \tag{B.31}$$

where the symmetry of $\hat{\boldsymbol{H}}^{(j)}$ have been used. The method has been found to work well in practice. [Fle87, pp. 53-54]

Another method called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) is also based on a rank two update, however in this case the rank two update is applied to the Hessian matrix directly. The quasi-Newton condition is in this case

$$\boldsymbol{\kappa}^{(j)} = \hat{\boldsymbol{G}}^{(j+1)}\boldsymbol{\delta}^{(j)}. \tag{B.32}$$

By the similarity of Eqs. (B.25) and (B.32), the BFGS update formula for the Hessian matrix is

$$\hat{\boldsymbol{G}}_{BFGS}^{(j+1)} = \hat{\boldsymbol{G}}^{(j)} + \frac{\boldsymbol{\kappa}^{(j)}\boldsymbol{\kappa}^{(j)^T}}{\boldsymbol{\kappa}^{(j)^T}\boldsymbol{\delta}^{(j)}} - \frac{\hat{\boldsymbol{G}}^{(j)}\boldsymbol{\delta}^{(j)}\boldsymbol{\delta}^{(j)^T}\hat{\boldsymbol{G}}^{(j)}}{\boldsymbol{\delta}^{(j)^T}\hat{\boldsymbol{G}}^{(j)}\boldsymbol{\delta}^{(j)}}. \tag{B.33}$$

The BFGS update formula for the inverse Hessian matrix is then found using the relation $\hat{\boldsymbol{G}}_{BFGS}^{(j+1)}\hat{\boldsymbol{H}}_{BFGS}^{(j+1)} = \boldsymbol{I}$. The result is

$$\hat{\boldsymbol{H}}_{BFGS}^{(j+1)} = \hat{\boldsymbol{H}}^{(j)} + \Big(1 + \frac{\boldsymbol{\kappa}^{(j)^T}\hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}}{\boldsymbol{\delta}^{(j)^T}\boldsymbol{\kappa}^{(j)}}\Big)\frac{\boldsymbol{\delta}^{(j)}\boldsymbol{\delta}^{(j)^T}}{\boldsymbol{\delta}^{(j)^T}\boldsymbol{\kappa}^{(j)}} - \Big(\frac{\boldsymbol{\delta}^{(j)}\boldsymbol{\kappa}^{(j)^T}\hat{\boldsymbol{H}}^{(j)} + \hat{\boldsymbol{H}}^{(j)}\boldsymbol{\kappa}^{(j)}\boldsymbol{\delta}^{(j)^T}}{\boldsymbol{\delta}^{(j)^T}\boldsymbol{\kappa}^{(j)}}\Big). \tag{B.34}$$

The BFGS update is encouraged in place of the David-Fletcher-Powell update, since both theoretically and practically the BFGS algorithm has shown better results in conjunction with inexact line searches instead of exact line searches. [Fle87, pp. 55-56]

Bibliography

[ADS99] M. Alexiadis, P. Dokopoulos, and H. Sahsamanoglou, "Wind speed and power forecasting based on spatial correlation models," *IEEE Transactions on Energy Conversion*, vol. 14, no. 3, pp. 836–842, 1999. DOI: 10.1109/60.790962.

[Box+16] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. John Wiley and Sons Inc., 2016.

[BT07] T. G. Barbounis and I. B. Theocharis, "Locally recurrent neural networks for wind speed prediction using spatial correlation," *Inf. Sci.*, vol. 177, pp. 5775–5797, 2007.

[Bur+01] T. Burton, N. Jenkins, D. Sharpe, and E. Bossanyi, *Wind energy handbook*. J. Wiley, 2001, ISBN: 0471489972.

[BV04] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004, ISBN: 0521833787.

[CD14] C. Croonenbroeck and C. M. Dahl, "Accurate medium-term wind power forecasting in a censored classification framework," Tech. Rep., 2014.

[DBM19] A. Delgado-Bonal and A. Marshak, "Approximate entropy and sample entropy: A comprehensive tutorial," *Entropy (Basel, Switzerland)*, vol. 21, no. 6, pp. 541–, 2019.

[DCS07] M. Duran, D. Cros, and J. Santos, "Short-term wind power forecast based on arx models," *Journal of Energy Engineering*, vol. 133, Sep. 2007. DOI: 10.1061/(ASCE)0733-9402(2007)133:3(172).

[Ene] Energinet. (). "Roles and responsibilities." https://energinet.dk/El/Elmarkedet/Roller-paa-elmarkedet (visited: 05/10-2021).

[Ene19] ——, *Introduktion til elmarkedet - kort introduktion til engros- og detailmarkedet*, 2019.

[Ene20] ——, *Introduktion til systemydelser*, 2020.

[Fle87] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. John Wiley & Sons, 1987.

[FLW01] U. Focken, M. Lange, and H.-P. Waldl, "Previento - a wind power prediction system with an innovative upscaling algorithm," *EWEC*, 2001.

[Fol92] G. B. Folland, *Fourier analysis and its applications*, ser. The Sally series. American Mathematical Society, 1992, ISBN: 9780821847909.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, eng, ser. Adaptive computation and machine learning. The MIT Press, 2016, ISBN: 9780262035613.

[Gie+19] D. Gielen, F. Boshell, D. Saygin, M. D. Bazilian, and N. Wagner, "The role of renewable energy in the global energy transformation," *Energy Strategy Reviews 2019, 24, 38-50*, 2019. DOI: http://dx.doi.org/10.3390/en13153764.

[GM74] N. Gupta and R. Mehra, "Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 774–783, 1974. DOI: 10.1109/TAC.1974.1100714.

[Han+20] S. Hanifi, X. Liu, Z. Lin, and S. Lotfian, "A critical review of wind power forecasting methods—past, present and future," *Energies 2020, 13, 3764*, 2020. DOI: http://dx.doi.org/10.3390/en13153764.

[Hua+98] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Snin, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," vol. 454, pp. 903–995, 1998, ISSN: 1364-5021. DOI: 10.1098/rspa.1998.0193.

[Jai16] P. Jain, *Wind Energy Engineering*, 2nd ed. McGraw-Hill Education, 2016, ISBN: 978-0-07-184384-3.

[Joh12] M. Johansson, "The hilbert transform," M.S. thesis, Växjö University, 2012.

[Kay93] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, 1st ed. Prentice Hall PTR, 1993.

[KS09] R. G. Kavasseri and K. Seetharaman, "Day-ahead wind speed forecasting using f-arima models," *Renewable Energy*, vol. 34, no. 5, pp. 1388–1393, 2009, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2008.09.006.

[KW07] G. Kirchgässner and J. Wolters, *Introduction to Modern Time Series Analysis*. Springer-Verlag Berlin Heidelberg, 2007, ISBN: 978-3-540-73291-4.

[LDB21] M.-D. Liu, L. Ding, and Y.-L. Bai, "Application of hybrid model based on empirical mode decomposition, novel recurrent neural networks and the arima to wind speed prediction," *Elsevier - Energy Conversion and Management*, 2021. DOI: https://doi.org/10.1016/j.enconman.2021.113917.

[Liu+10] H. Liu, H.-Q. Tian, C. Chen, and Y. fei Li, "A hybrid statistical method to predict wind speed and wind power," *Elsevier, Renewable Energy*, 2010. DOI: https://doi.org/10.1016/j.renene.2009.12.011.

[LW08] R. Li and Y. Wang, "Short-term wind speed forecasting for wind farm based on empirical mode decomposition," International Conference on Electrical Machines and Systems, 2008, pp. 2521–2525.

[Lyd+14] M. Lydia, S. S. Kumar, A. I. Selvakumar, and G. E. Prem Kumar, "A comprehensive review on wind turbine power curve modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 452–460, 2014, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.10.030.

[ML] S. Mckinley and M. Levine, *Cubic spline interpolation*, https://www.rajgunesh.com/resources/downloads/numerical/cubicsplineinterpol.pdf (Visited 16/12 2021).

[MT10] H. Madsen and P. Thyregod, *Introduction to general and generalized linear models*, 1st ed. CRC press, 2010. DOI: 10.1201/9781439891148.

[Neu16] K. Neusser, *Time Series Econometrics*, 1st ed. Springer, 2016, ISBN: 978-3-319-32861-4.

[Nie+07] H. A. Nielsen, P. Pinson, L. E. Christiansen, T. S. Nielsen, H. Madsen, J. Badger, G. Giebel, and H. F. Ravn, "Improvement and automation of tools for short term wind power forecasting," European Wind Energy Conference, 2007.

[Pen+20] Z. Penga, S. Pengb, L. Fuc, B. Luc, J. Tanga, K. Wangd, and W. Lia, "A novel deep learning ensemble model with data denoising for short-term wind speed forecasting," *Elsevier - Energy Conversion and Management*, 2020. DOI: https://doi.org/10.1016/j.enconman.2020.112524.

[PS+09] J. Palomares-Salas, J. J. de la Rosa, J. Ramiro-Leo, J. Melgar, A. Agüera-Pérez, and A. Moreno-Munoz, "Arima vs. neural networks for wind speed forecasting," Computational Intelligence for Measurement Systems and Applications, May 2009, pp. 129–133, ISBN: 978-1-4244-3819-8. DOI: 10.1109/CIMSA.2009.5069932.

[RFG03] G. Rilling, P. Flandrin, and P. Gonçalves, "On empirical mode decomposition and its algorithms," *Proceedings of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing NSIP-03*, vol. 3, Jun. 2003.

[RM00] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 278, no. 6, H2039–H2049, 2000. DOI: 10.1152/ajpheart.2000.278.6.H2039.

[RM10] N. Rehman and D. Mandic, "Multivariate empirical mode decomposition," *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 466, pp. 1291–1302, May 2010. DOI: 10.1098/rspa.2009.0502.

[Sch07] L. L. Schumaker, *Spline functions : basic theory*, 3rd ed., ser. Cambridge mathematical library. Cambridge University Press, 2007, ISBN: 1-107-17377-9.

[Sin+21] U. Singh, M. Rizwan, M. Alaraj, and I. Alsaidan, "A machine learning-based gradient boosting regression approach for wind power production forecasting: A step towards smart grid environments," *Energies 2021, 14, 5196*, 2021. DOI: https://doi.org/10.3390/en14165196.

[Smi07] J. O. Smith III, *Mathematics of the discrete fourier transform (DFT) with audio applications*, 2nd ed. Center for Computer Research in Music and Acoustics (CCRMA), 2007.

[Som+10] S. Soman, H. Zareipour, O. Malik, and P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons," 2010. DOI: 10.1109/NAPS.2010.5619586.

[SPC20] N. Safari, G. Price, and C. Y. Chung, *Analysis of empirical mode decomposition-based load and renewable time series forecasting*, 2020. arXiv: 2011.11410.

[SS17] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications - With R Examples*, 4th ed. Springer, 2017.

[SS93] M. Sun and R. Sclabassi, "Discrete-time instantaneous frequency and its computation," *IEEE Transactions on Signal Processing*, vol. 41, no. 5, pp. 1867–1880, 1993. DOI: 10.1109/78.215305.

[sta21a] statista, *Electricity generation in denmark from 2001 to 2020*, https://www.statista.com/statistics/450409/denmark-electricity-generation/ (visited: 14/09-2021), 2021.

[sta21b] ——, *Share of wind power coverage in denmark from 2009 to 2020*, https://www.statista.com/statistics/991055/share-of-wind-energy-coverage-in-denmark/ (visited: 14/09-2021), 2021.

[SV13] T. Sivanagaraja and K. Veluvolu, "A hybrid approach for short-term forecasting of wind speed," *TheScientificWorldJournal*, vol. 2013, p. 548 370, Dec. 2013. DOI: 10.1155/2013/548370.

[TAK18] P. Turner, T. Arildsen, and K. Kavanagh, *Applied Scientific Computing: With Python*, ser. Texts in Computer Science. Springer, 2018, ISBN: 978-3-319-89574-1. DOI: 10.1007/978-3-319-89575-8.

[Tes10] G. Teschl, *Topics in Real Analysis*, 1st ed. American Mathematical Society Providence, 2010.

[TH11] P. Trnka and M. Hofreiter, "The empirical mode decomposition in real-time," *Proceedings of the 18th International Conference on Process Control*, Jan. 2011.

[Tou+21] J.-F. Toubeau, P.-D. Dapoz, J. Bottieau, A. Wautier, Z. D. Grève, and F. Vallée, "Recalibration of recurrent neural networks for short-term wind power forecasting," *Elsevier, Electric Power Systems Research*, 2021. DOI: https://doi.org/10.1016/j.epsr.2020.106639.

[VKG14] M. Vetterli, J. Kovacevic, and V. K. Goyal, *Foundations of Signal Processing*. Cambridge Univ. Press, 2014, ISBN: 9781107038608 110703860X 9781107662223 1107662222.

[WH09] Z. Wu and N. Huang, "Ensemble empirical mode decomposition: A noise-assisted data analysis method," *Advances in Adaptive Data Analysis*, vol. 1, pp. 1–41, Jan. 2009. DOI: 10.1142/S1793536909000047.

[Wu+20] Q. Wu, F. Guan, C. Lv, and Y. Huang, "Ultra-short-term multi-step wind power forecasting based on cnn-lstm," *IET Renewable Power Generation*, 2020. DOI: 10.1049/rpg2.12085.

[WZP10] J. Wang, X.-F. Zhong, and X.-Y. Peng, "Deal with boundary effect of emd method based on improved characteristic wave extending," 3rd International Symposium on Systems, Control in Aeronautics, and Astronautics, Jun. 2010. DOI: 10.1109/ISSCAA.2010.5632875.

[XHY19] W. Xu, H. Hu, and W. Yang, "Energy time series forecasting based on empirical mode decomposition and frbf-ar model," *IEEE Access*, vol. 7, pp. 36 540–36 548, 2019. DOI: 10.1109/ACCESS.2019.2902510.

[Xia+17] L. Xiao, W. Shao, M. Yu, J. Ma, and C. Jin, "Research and application of a hybrid wavelet neural network model with the improved cuckoo search algorithm for electrical power system forecasting," *Elsevier - Applied Energy*, 2017. DOI: http://dx.doi.org/10.1016/j.apenergy.2017.04.039.

[YM21] T. Young and M. J. Mohlenkamp, *Lecture 19 polynomial and spline interpolation*, http://www.ohiouniversityfaculty.com/youngt/IntNumMeth/, 2021.

[ZCA16] F. Ziel, C. Croonenbroeck, and D. Ambach, "Forecasting wind power – modeling periodic and non-linear effects under conditional heteroscedasticity," *Applied Energy*, vol. 177, 285–297, Sep. 2016, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2016.05.111.

[Zha+19] J. Zhang, J. Yan, D. Infield, Y. Liu, and F. sang Lien, "Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and gaussian mixture model," *Elsevier, Renewable Energy*, 2019. DOI: https://doi.org/10.1016/j.apenergy.2019.03.044.

[Zha+20] Y. Zhang, Q. Ai, F. Xiao, R. Hao, and T. Lu, "Typical wind power scenario generation for multiple wind farms using conditional improved wasserstein generative adversarial network," *International Journal of Electrical Power & Energy Systems*, vol. 114, p. 105 388, 2020, ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2019.105388.

[Zhe+13] Z. Zheng, Y. Y. Chen, X. Zhou, M. Huo, B. Zhao, and M. Y. Guo, "Short-term wind power forecasting using empirical mode decomposition and rbfnn," *International Journal of Smart Grid and Clean Energy*, vol. 2, pp. 192–199, 2013.

[ZX19] Y. Zheng and G. Xu, "Quantifying mode mixing and leakage in multivariate empirical mode decomposition and application in motor imagery–based brain-computer interface system," *Medical & Biological Engineering & Computing*, vol. 57, pp. 1–15, Feb. 2019. DOI: 10.1007/s11517-019-01960-9.