



Department of  
Materials and Environmental Technology

## Scenario Generation for Wind Power Using Ramping Behaviour Analysis (RBA)

Tuuleenergia stsenaariumide genereerimine RBA (Nõlv Käitumise  
Analüüsi) Abil

MASTER THESIS

Student: Esin Ören

Student code: KAYM156318

Supervisor: Sambeet Mishra, Early Stage Researcher

Co-supervisor: Prof. Ivo Palu

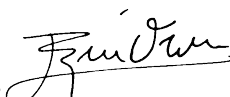
Tallinn, 2018

## AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.

No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

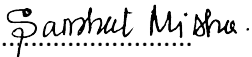
"30" May 2018.

Author: 

/signature /

This is in accordance with terms and requirements.

"30" May 2018.

Supervisor: 

/signature/

"....." ..... 201....

Co-supervisor: .....

/signature/

Accepted for defence

"....." .....201... .

Chairman of theses defence commission: .....

/name and signature/

## THESIS TASK

Student: Esin Ören, ID:1563181  
Study programme: Materials and Processes for Sustainable Energetics, KAYM  
Main speciality: Processes for Sustainable Energetics  
Supervisor(s): Sambeet Mishra, Early Stage Researcher, 620 3759  
Prof. Ivo Palu, 620 3752  
Consultants: N/A

### Thesis topic:

(in English) Scenario Generation for Wind Power Using Ramping Behaviour Analysis (RBA)

(in Estonian) Tuuleenergia Stsenariumide Genereerimine RBA (Nõlv Käitumise Analüüsi) Abil

### Thesis main objectives:

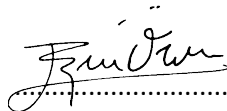
1. Preprocessing of the data
2. Identification of events using new RBA with persistence
3. Generation of forecast
4. Generation of scenarios

### Thesis tasks and time schedule:

No	Task description	Deadline
1.	Literature review	25.02.2018
2.	Preprocessing of the data	9.03.2018
3.	Event detection	23.03.2018
4.	Scenario generation	20.05.2018

Language: English Deadline for submission of thesis: 30.05.2018

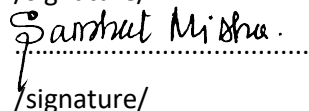
Student: Esin Ören



“ 30 ” May 2018.a

/signature/

Supervisor: Sambeet Mishra



“ 30 ” May 2018.a

/signature/

Co-supervisor: Phd. Ivo Palu

.....

“.....”.....201.....a

/signature/

*To find the secrets of the universe,  
think in terms of energy, frequency and vibration.*

---

***Nikola Tesla***

# CONTENTS

PREFACE .....	7
List of abbreviations.....	8
List of symbols.....	9
1 INTRODUCTION .....	10
1.1 Task definition and goal .....	10
1.2 Literature review.....	11
1.3 Outline.....	12
2 Ramp Events .....	13
2.1 Preprocessing of the data .....	14
2.1.1 Smoothing with B-splines.....	14
2.1.2 Spectral density analysis .....	16
2.2 Ramp event definition.....	19
2.2.1 Ramp events after preprocessing the data .....	21
2.3 Ramp events with Rainflow analysis.....	22
2.4 Persistence analysis.....	23
3 Scenario generation.....	25
3.1 Artificial neural networks(ANNs) .....	25
3.1.1 Basics with ANNs .....	25
3.1.2 Neural network architecture.....	27
3.2 Recurrent neural networks (RNNs) .....	30
3.2.1 Forecasting with Long-short term memory (LSTM) .....	30
3.3 Forecasting with Generative adversarial network (GAN) .....	33
3.4 Scenarios with Monte Carlo Markov Chains(MCMC).....	36
3.4.1 Markov Chains.....	36
3.4.2 Spatial Markov.....	36
4 Conclusions.....	38
4.1 Future work.....	38
4.2 Summary .....	38
5 BIBLIOGRAPHY.....	40

## TABLE OF FIGURES

Figure 2.1 Wind park power output (capacity factor) data .....	13
Figure 2.2 Cubic B-spline interpolation of the data with different smoothness factor .....	15
Figure 2.3 Dataset filtered with Haar wavelet .....	16
Figure 2.4 Blackman window and its frequency response from (Scipy.org).....	17
Figure 2.5 Power spectral density (Welch-periodogram)of $x[n]$ after interpolation with cubic splines, with the whole data as the time window using Blackman window.....	18
Figure 2.6 Filtered data with 0.2 Hz cut-off frequency .....	18
Figure 2.7 Ramp events extracted from a small sample of the data with 0.04 T .....	20
Figure 2.8 Ramp events extracted from the small sample of the data with 0.08 T.....	20
Figure 2.9 Ramp events extracted from the small sample of the data with 0.12 T.....	21
Figure 2.10 Events extracted after preprocessing our data.....	22
Figure 3.1 The representation of perceptron (Nielsen, 2015). .....	25
Figure 3.2 Multilayer perceptrons .....	28
Figure 3.3 One cell LSTM memory block (Graves et al., 2009) .....	30
Figure 3.4 Forecast with LSTM for time ranges .....	31
Figure 3.5 Forecast with LSTM for changes in amplitude .....	32
Figure 3.6 Forecast with LSTM for the angle of events.....	32
Figure 3.7 GAN working principle (Chen et al., 2018) .....	33
Figure 3.8 Distribution functions of the generator(blue) and the discriminator(orange) .....	34
Figure 3.9 Discriminator loss.....	35
Figure 3.10 Representation of areas defined for each turbine in the shapefile used in spatial analysis with corresponding numbers .....	36
Figure 3.11 Neighbouring relationships based on weights with Queen principal .....	37

## PREFACE

This thesis work was conducted in collaboration with the Department of Electrical Power Engineering and Mechatronics, Tallinn University of Technology. It is dedicated to computer vision and aims to bring more application of machine learning to sustainable energy technologies.

The author would like to thank supervisor Sambheet Mishra for all the assistance and inspiration to the work, and her family who believed in her and supported her whenever and wherever.

**Keywords:** *Renewable integration, scenario generation, generative models, deep learning*

## LIST OF ABBREVIATIONS

<i>ANN</i>	Artificial Neural Network
<i>CNN</i>	Convolutional Neural Network
<i>DTFT</i>	Discrete-Time Fourier Transform
<i>GAN</i>	Generative Adversarial Network
<i>MLP</i>	Multilayer Perceptron
<i>RNN</i>	Recurrent Neural Network
<i>SDE</i>	Spectral Density Estimation
<i>U.S.NRC</i>	United States Nuclear Regulatory Commission
<i>LSTM</i>	Long Short Term Memory



## LIST OF SYMBOLS

$C_f$	Capacity factor
$P_{gen_{net}}$	Net generated power
$P_{rated}$	Rated power
$s(x)$	Spline function
$c_i$	Spline coefficients
$B^k(x)$	Basis spline polynomials on the order of k
$x[n]$	Discrete time series
$X(e^{j\omega})$	Fourier series of $x[n]$
$\omega$	Frequency variable
$t$	Time
$\Delta t$	Time interval
$w(t)$	Power output at t
$\Delta w$	Ramp event
$\Delta w_s$	Significant ramp event
$T$	Threshold

# 1 INTRODUCTION

Renewable sources, in general, have been growing in the last decade becoming a more and more important component of the energy supply in many areas around the world. The reasons behind this growth are the growing demand, increasing fossil fuel prices and the necessity to reduce the greenhouse gas emissions. It can easily be predicted that these reasons will not disappear in the upcoming years even with intentions to increase energy efficiency and decrease fossil-based energy production (European Commission, n.d.). All this makes wind energy very attractive as an alternative, which is expected to grow in the years to come.

Electricity is an instantaneous commodity that is consumed as and when produced. Though efficient technologies exist to store energy, the current technologies can only store a limited amount. Hence, the essential principle of power system management is to ensure the balance between the supply and demand sides at all grid points and at all times. Conventionally, this is achieved by power stations to provide the power whenever and as much as the consumers need electricity.

However, the wind is a natural and renewable energy source. Wind energy is produced by the wind, thus inheriting the stochastic nature of wind.

When there is a small penetration of wind into the power systems, the uncertain behaviour of the wind power generation is treated as just another uncertainty on the demand side, and the conventional power stations cover for this variability which requires additional energy and reduces the environmental benefits.

Forecasting is one of the many possible solutions to this problem, as well as an interconnected grid, energy storage technologies, demand-side management such as electric vehicles. Forecasting aims to model the uncertainties inherited by the grid through wind power production and thus are a necessary and cost-effective element for the optimal integration of wind power into energy systems. However, forecasting is never accurate and literature suggests providing bounds for the forecast or confidence intervals.

## 1.1 Task definition and goal

The objective of this study is to propose novel indicators using methods to quantify ramp events from temporal wind production along with features of these events, generate forecasts using

Artificial Neural Networks(ANNs) the multivariate and spatiotemporal characteristics derived from Markov chains to generate probabilistic scenarios.

The goal is to generate realistic sets of scenarios depending on previous wind power generations, creating a set of tools that could contribute to advancing the limits of wind power forecasting. The purpose of this study is to analyse the underlying patterns in wind power production looking from a zoomed out perspective. Also, the indicators explain the characteristics of an event. Instead of forecasting the raw data, we generate scenarios for the significant events.

## 1.2 Literature review

There are many studies in the literature on characteristics of wind power, correlations of wind and wind power, forecasting wind and wind power output, scenario generation. (Bianco et al., 2016) proposed a model to forecast ramp events as well. They modelled observed wind speeds into forecast models and converted this into power forecasts with the help of the power curve of the wind turbines. It suggests that the same method could be implemented for solar power plants. Another noteworthy one is a model-free forecast generation implemented with Generative Adversarial Networks(GAN)(Chen, Wang, Kirschen, & Zhang, 2018). GANs have two deconvolutional, one that starts ou generating random data and the out discriminates whether its input is coming from the generator or historical data. These two neural networks play a Nashville game while giving feedback to each other, both getting better over time until the generator generates data that is almost like a forecast so that the discriminator can not discriminate anymore. (Ming-jian Cui et al., 2015) proposed a probabilistic forecasting method, utilising a Neural Network(NN) to generate possible future scenarios, employing an objective function based on cumulative distribution functions and autocorrelation functions to train the NN, primarily teaching it their distribution. Again another (Karatepe & Corscadden, 2013) proposed a model to synthesised wind speed scenarios based on statistical parameters of wind and Markov chains. In contrast, (Kaut, 2014) proposes a new heuristic to generate scenarios that use copulas instead of common correlation functions. (Mishra, Leinakse, & Palu, 2017) introduced the terminology for identification of ramp events, ramping behaviour analysis(RBA), which comprises the perspective used in this study. They also filtered and extracted events, and clustered them into groups. More studies exist on identifying ramp events (Mingjian Cui et al., 2016), (Bossavy, Girard, & Kariniotakis, 2013), (Bossavy, Girard, Kariniotakis, & Antipolis, 2013).

## 1.3 Outline

In the first chapter, the main idea is introduced, the motivation behind this thesis is explained. Thesis' task definition is clarified, and the goal is specified as well. Overviewed of the literature, which has already become a milestone in this field, and previous researcher related to this thesis task. In the overview part, we explained the essential concepts and methods which are related to the topic. Those will be compared with this thesis work to improve the result.

Based on the defined tasks in the first chapter and drawn boundaries, we will search for the most efficient methods to apply in the second chapter to our data after introducing our data. That is used for further analysis are explained, and methods are built up as we go further. At the very end of this chapter, we will have a set of algorithms applied to our data, and that will enable us to progress within the next chapters.

Since the chosen solution is already too wide to explain in the third chapter, explanation of the specific solution will take place in the third chapter with its boundaries and specific parameters. Implementation of the solution is going to be another task too. The result of this chapter is going to get the finalised algorithm to progress with experimentation section.

In the fourth, experimentation chapter, our chosen type of Neural Networks(NN), a GAN (Chen et al., 2018) and a common Long Short Term Memory(LSTM) will be used to develop our forecasts then using a multivariate copula simulation, a set of possible scenarios will be generated. Testing the finally implemented solution is going to be a task at the end of this chapter, and of course, evaluation is a must in this section. "What could be better?" and "How to make it more efficient?" are going to be answered in this part of the research.

In the concluding chapter, future works will be mentioned based on what could be done differently in this thesis work. Moreover, of course, the summary is another must at the very the end of this thesis.

All computational work done is made available through GitHub with the standard academic licence. Toolboxes used are Numpy, Scipy, Pysal, Tensorflow and Keras.

## 2 RAMP EVENTS

The data for this analysis is obtained from Paldiski wind farm(Nelja Energia) located in Harju County, on the territory of the town Paldiski. It contains the power output of seventeen wind turbines, each of which has a capacity of 2,5MW. It has ten-minute resolution and starts from September ending at the end of December 2013. All data points coincide with each other date and time-wise for each turbine. Figure 2.1 represents the wind park power generation for September 2013 including all turbines. It shows that, even though the data has quite a lot fluctuation, there seems to be a time-wise correlation between turbines.

Firstly, it is translated into capacity factor values before any method applied. Definition of the capacity factor denoted in Equation (2.1) is the ratio of the net power generated to the optimum power that could have been generated at continuous full-power operation during a period of the time (U.S.NRC, 2015). Therefore values after translation range between one and zero according to how much potential generation achieved.

$$Cf = P_{gen_{net}}/P_{rated} \tag{2.1}$$

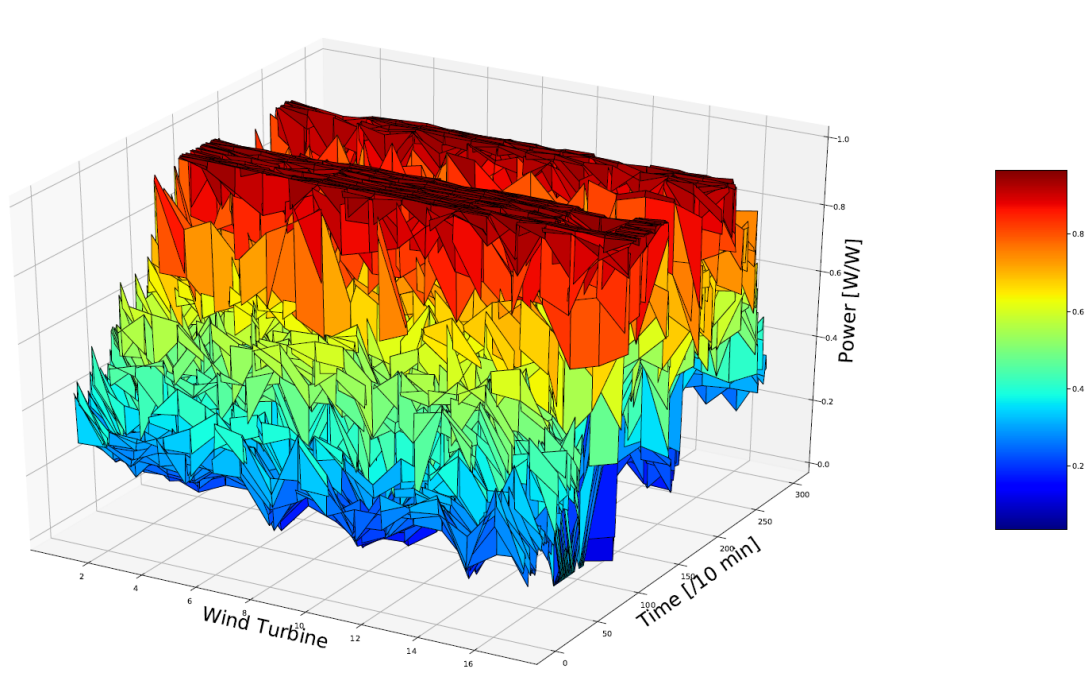


Figure 2.1 Wind park power output (capacity factor) data

## 2.1 Preprocessing of the data

We presume that the data includes an unknown degree of noise looking at its erratic nature with many minor fluctuations. We are interested in significant changes rather than smaller, so we propose a chain of methods to apply to the data.

### 2.1.1 Smoothing with B-splines

Splines are piecewise polynomial curves that are differentiable up to a prescribed order. The curve  $s(x)$  is a spline of degree  $k-1$  (or a spline of order  $k$ ) with knots  $t_0, \dots, t_m$ , where  $t_i \leq t_{i+1}$  and  $t_i \leq t_{i+k}$  for all possible  $i$ , if  $s(x)$  is  $k-r-1$  times differentiable at any  $r$ -fold knot, and  $s(x)$  is a polynomial of degree  $\leq k$  over each knot interval  $[t_i, t_{i+1}]$ , for  $i=0, \dots, m-1$ .

A spline of order  $k$   $s(x)$  is represented as an affine combination of coefficients  $c_i$ , with  $B_{j,k}^k$  as the basis spline functions.

$$s(x) = \sum c_i B_i^k(x) \quad (2.2)$$

When the knot sequence  $t_i$  is  $t$ , biinfinite and strictly increasing in sequence, which means  $t_i \leq t_{i+1}$  for all  $i$ ,

$$B_i^0(x) = 1, \text{ if } t_i \leq x < t_{i+1}, 0 \text{ otherwise.} \quad (2.3)$$

$$B_i^k(x) = \alpha_i^{k-1} B_i^{k-1}(x) + (1 - \alpha_{i+1}^{k-1}) B_{i+1}^{k-1}(x) \quad (2.4)$$

where

$$\alpha_i^{k-1} = \frac{x - t_i}{t_{i+k} - t_i} \quad (2.5)$$

is the local parameter with respect to the support of  $B_i^{k-1}(x)$ . (Prautzsch, Boehm, & Paluszny, 2002).

When we introduce a smoothing condition to the interpolation, there exists a trade-off between closeness and smoothness of the fit, which after a certain point the interpolation turns into a simple least mean square regression.

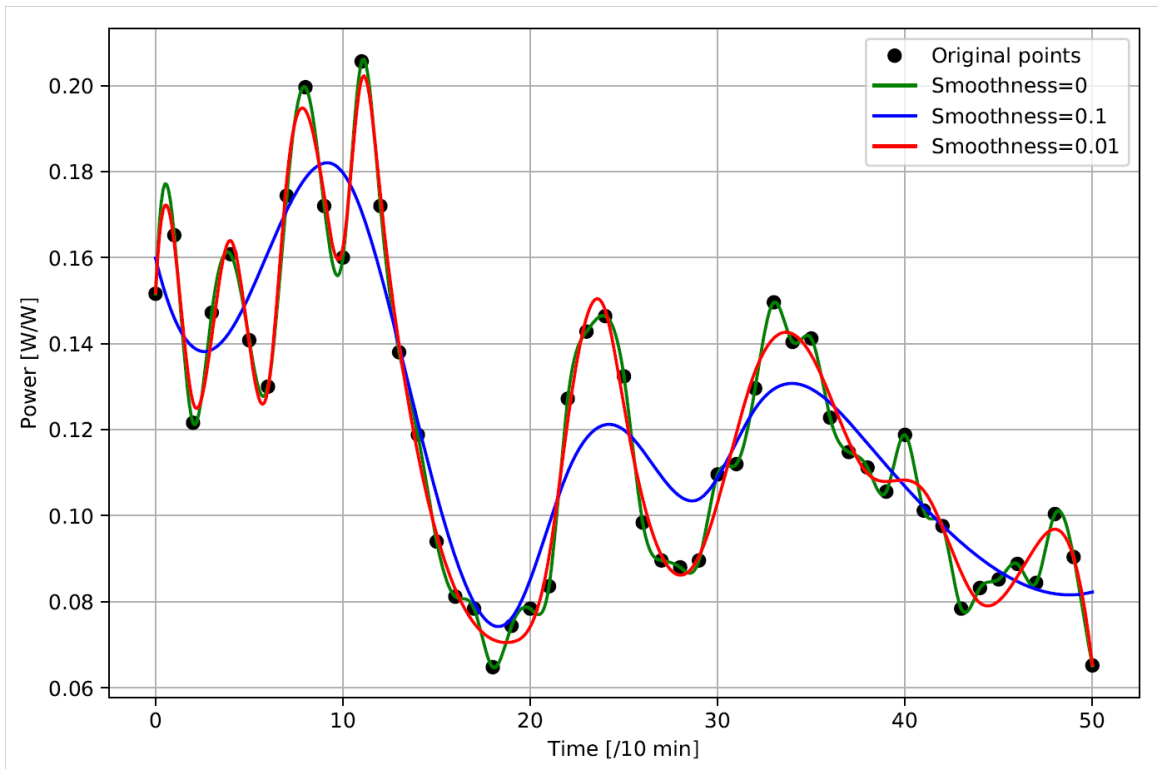


Figure 2.2 Cubic B-spline interpolation of the data with different smoothness factor

The structure of the algorithm we apply takes three arguments, the data, smoothness factor and the degree of the spline which are set to a default of 0.01 and 3, respectively. We choose to implement a cubic spline with the smoothness of 0.01 to be able to preserve the behaviour of the curve while smoothing out minor fluctuations.

There is another option that would do this task, wavelets. Wavelets are quite popular in signal processing and compressing because they can represent localisations in time and frequency contrary to the traditional Fourier. Figure 2.3 that was plotted using a Haar mother wavelet which is a rectangular filter shows that Fourier transform is much more useful for our case.

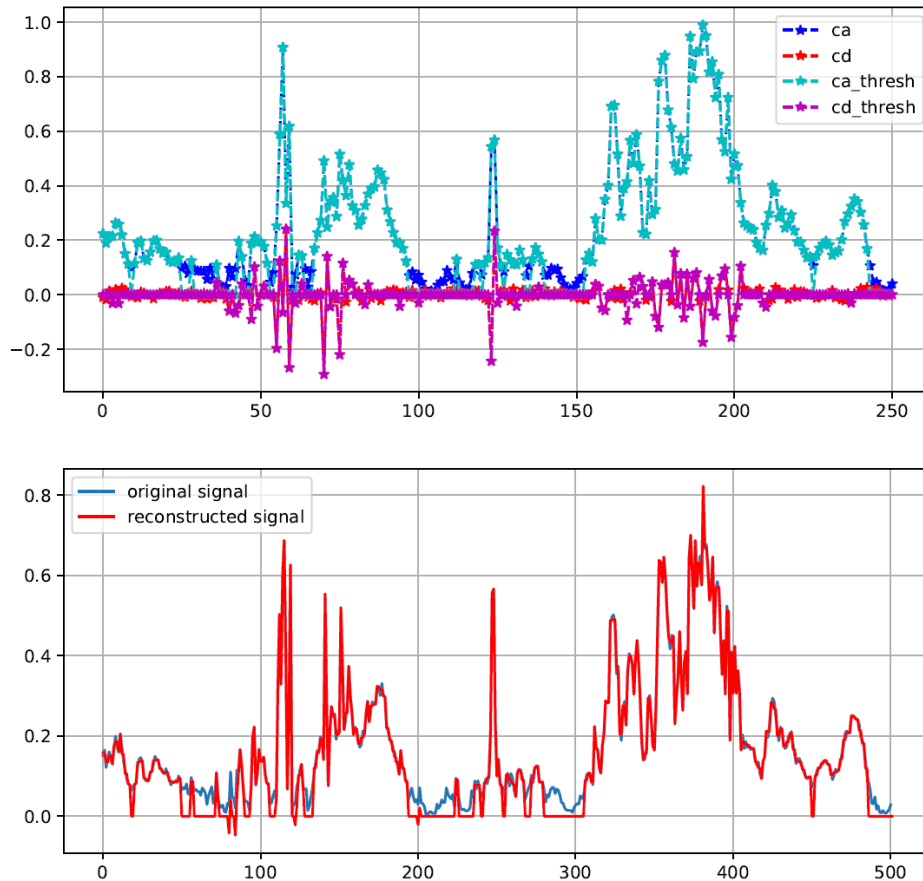


Figure 2.3 Dataset filtered with Haar wavelet

## 2.1.2 Spectral density analysis

In electronics, control theory and statistics, the frequency domain means the analysis of mathematical functions or signals in reference to frequency rather than time. A given function or signal can be transformed to the time domain from the frequency domain, and vice versa with mathematical operators called transforms. In this study, we focus on Fourier transforms.

The Fourier transform of a function contains all the information about the original signal, and with this information, it is possible to reconstruct the function entirely by an inverse Fourier transform. This information includes amplitude and phase of each frequency present in the function.

Usually, there is no actual signal available, like it is in our case, but the discrete-time sequence of samples. The Fourier transform of a discrete-time signal  $x[n]$ ,  $n=0,\dots,N$  is called the discrete-time Fourier transform (DTFT), which provides a mathematical approximation of the full integral solution, and yields a periodic frequency spectrum. The DTFT of the sequence  $x[n]$  denoted in



Equation (2.6) is a function of a continuous frequency variable  $\omega$  and  $X(e^{j\omega})$  and is always periodic with period  $2\pi$ . (McClellan, Schafer, & Yoder, 2003)

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \quad (2.6)$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega \quad (2.7)$$

Equation (2.7) represents the inverse DTFT of  $x[n]$  (McClellan et al., 2003).

Discrete Fourier Transform(DFT) can be obtained from the DTFT if we evaluate Equation (2.6) at a discrete set of equally spaced frequencies.

To be able to determine the spectrum of a sampled signal correctly using DFT, we have to select a finite number of samples for Fourier analysis. Then, it is possible to represent this selection as the multiplication of  $x[n]$  by another sequence  $w[n]$  which is called a window.

In this study, the window we choose is a Blackman window plotted in Figure 2.4. Its time-domain representation is denoted in Equation (2.8).

$$w(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right) \quad 0 \leq n \leq M-1 \quad (2.8)$$

Where  $N$  – the length of the Blackman window

$M = N/2$ , if  $N$  is even,  $(N+1)/2$  if  $N$  is odd (Agarwal, Singh, & Pandey, 2014)

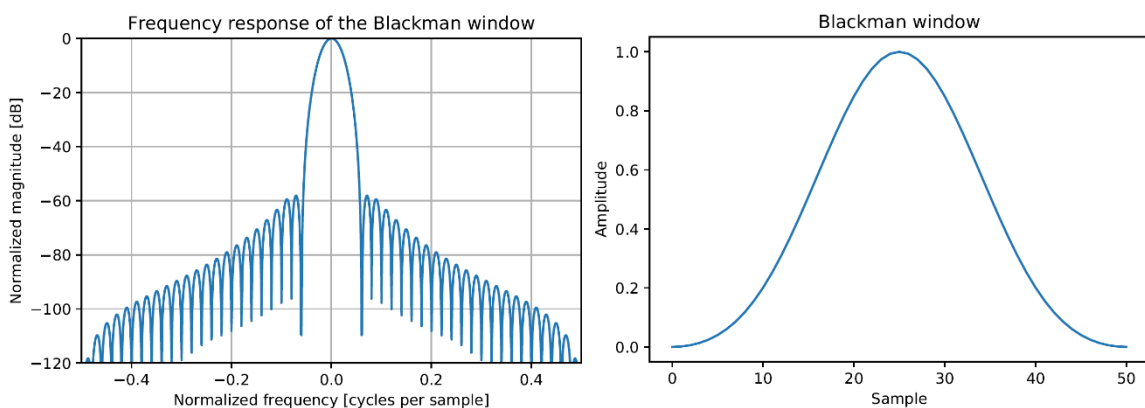


Figure 2.4 Blackman window and its frequency response from (Scipy.org)

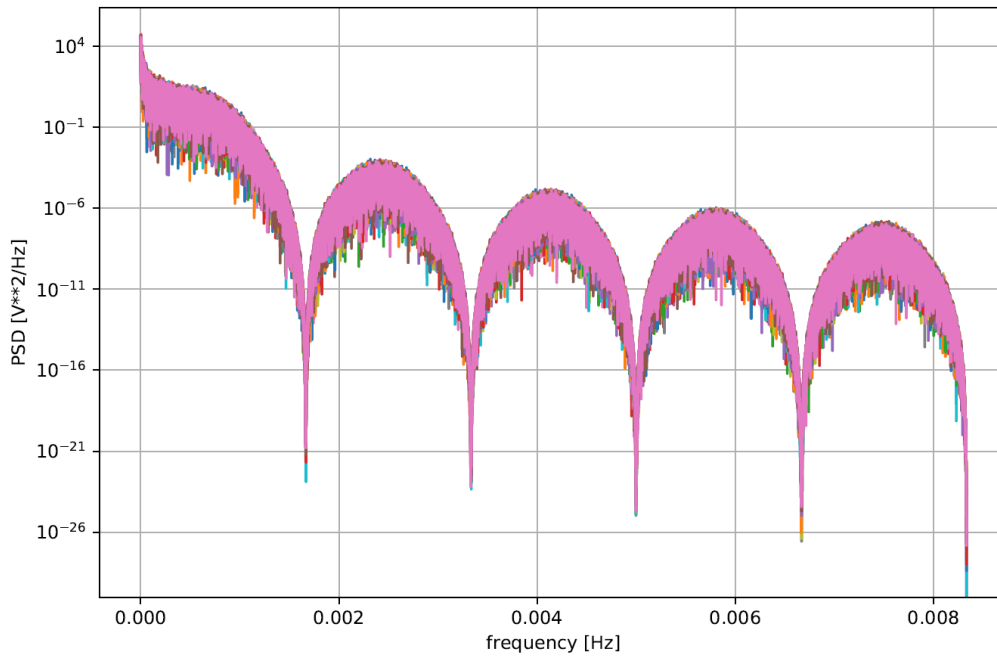


Figure 2.5 Power spectral density (Welch-periodogram) of  $x[n]$  after interpolation with cubic splines, with the whole data as the time window using Blackman window

After inspecting Figure 2.5 Power spectral density (Welch-periodogram) of  $x[n]$  after interpolation with cubic splines, with the whole data as the time window using Blackman window Figure 2.5, we choose the cut-off frequency  $f_s$  as 0.2 Hz. A sample of the filtered signal with Blackman and its original can be seen in Figure 2.6.

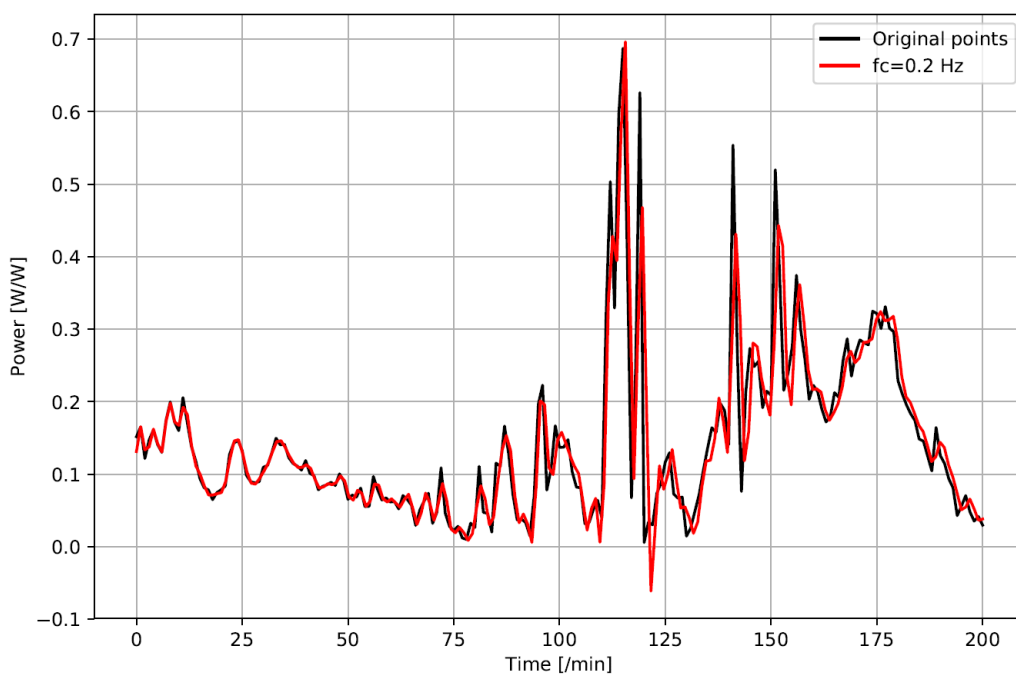


Figure 2.6 Filtered data with 0.2 Hz cut-off frequency

## 2.2 Ramp event definition

When discrete time points are denoted as  $t$  and the time point after  $t$  as  $t+\Delta t$ , the wind output are represented as  $w(t)$  and  $w(t + \Delta t)$ , respectively, and the difference between two consecutive wind power outputs is defined as a ramp,  $\Delta w$ ;

$$\Delta w = w(t + \Delta t) - w(t) \quad (2.9)$$

A positive value of  $\Delta w$  is an increase in the wind power output which will be identified as an “up ramp”, while a negative value of  $\Delta w$  stands for a decrease in the wind power output, hence a “down ramp”.

A ramp event  $\Delta w_s$  is defined as an event where a significant change in power production happens in a time period  $\Delta t$ . The significance comes from the parameter  $T$ , which stands for an adjustable threshold to neglect  $\Delta w$  values that are smaller than  $T$ .

$$\Delta w_s = \Delta w, \text{ if } \Delta w > T \quad (2.10)$$

When there are two or more consequential  $\Delta w_s$ , they get connected together and counted as one, even if there might be neglected  $\Delta w$  in between.

Values needed for further analysis are  $\alpha$  which is the angle between the time interval and the change in amplitude, and the mean for every significant event.

$$\text{mean}(\Delta w_s) = [w_s(t) + w_s(t + \Delta t)]/2 \quad (2.11)$$

$$\alpha(\Delta w_s) = \arctan(\Delta w_s, \Delta t) \quad (2.12)$$

---

**Algorithm I: Event extraction**

---

**Inputs:**  $T, w(t), t=1, \dots, N$

$\Delta w = w(t + \Delta t) - w(t)$

**For**  $i \leftarrow 1$  **to**  $\text{length}(\Delta w)$  **do:**

**If**  $\text{sign}(\Delta w[i]) = \text{sign}(\Delta w[i + 1])$ :

        concatenate( $\Delta w$ )

**For**  $i \leftarrow 1$  **to**  $\text{length}(\Delta w)$  **do:**

**If**  $\Delta w > T$ :

$\Delta w_s \leftarrow \Delta w$

---

---

```

For  $i \leftarrow 1$  to length( $\Delta w_s$ ) do:
  If sign( $\Delta w_s[i]$ ) = sign( $\Delta w_s[i + 1]$ ):
    concatenate( $\Delta w_s$ )
Return  $w_s(t)$ ,  $w_s(t + \Delta t)$ ,  $t$ ,  $t + \Delta t$ ,  $\Delta w_s$ ,  $\alpha(\Delta w_s)$ , mean( $\Delta w_s$ )
End

```

---

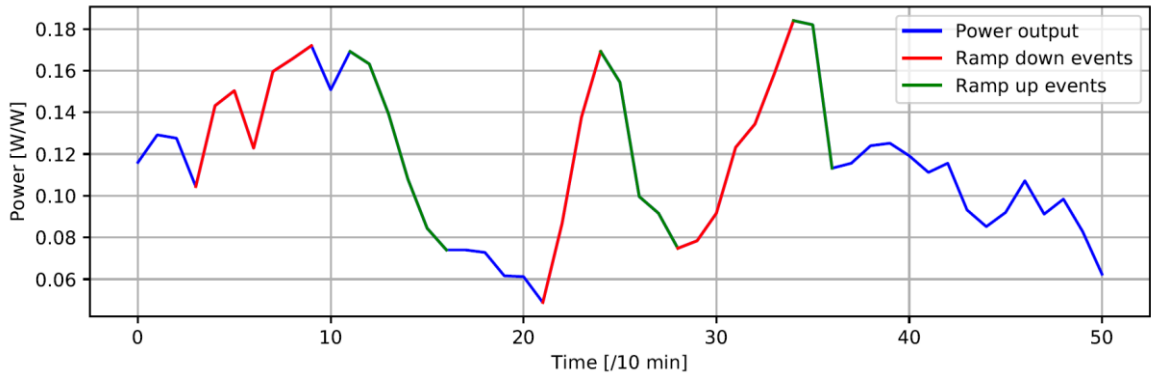


Figure 2.7 Ramp events extracted from a small sample of the data with 0.04 T

Table 2.1 Ramp events in Figure 2.7

Event	$w_s(t)$	$\Delta w_s(t + \Delta t)$	$t$	$t + \Delta t$	$\Delta w_s(t)$	$\Delta t$	$\alpha(\Delta w_s)$	mean( $\Delta w_s$ )
1	0.104	0.172	3	9	0.067	6	6.428	0.138
2	0.12	0.172	6	9	0.049	3	9.313	0.14
3	0.1692	0.074	11	16	-0.095	5	-10.780	0.121
4	0.048	0.169	21	24	0.120	3	21.867	0.109
5	0.169	0.074	24	28	-0.094	4	-13.278	0.122
6	0.074	0.184	28	34	0.109	6	10.314	0.129
7	0.184	0.113	34	36	-0.070	2	-19.494	0.148

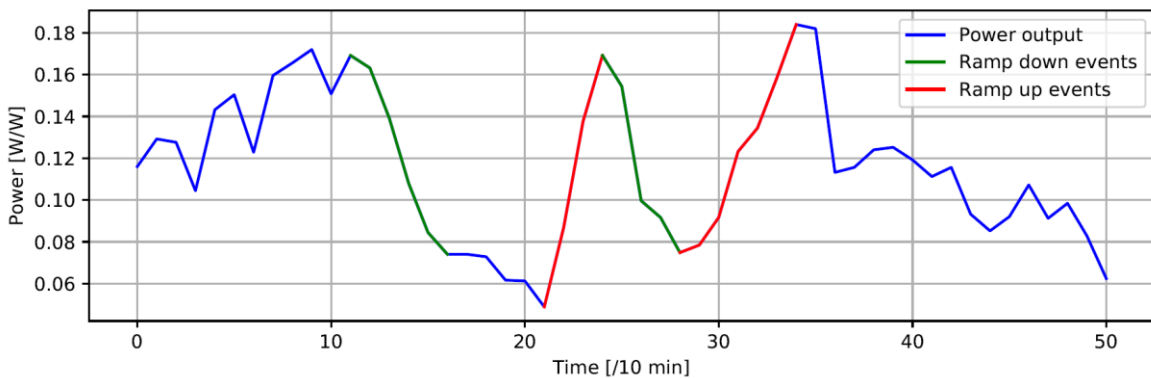


Figure 2.8 Ramp events extracted from the small sample of the data with 0.08 T

Table 2.2 Ramp events in Figure 2.8

Event	$w_s(t)$	$\Delta w_s(t + \Delta t)$	$t$	$t + \Delta t$	$\Delta w_s(t)$	$\Delta t$	$\alpha(\Delta w_s)$	$\text{mean}(\Delta w_s)$
1	0.169	0.074	11	16	-0.095	5	-10.780	0.121
2	0.048	0.169	21	24	0.120	3	21.867	0.109
3	0.169	0.07	24	28	-0.094	4	-13.278	0.122

The optimum  $T$  for meaningful event extraction is unknown, but we use the assumption of 10% of the nominal capacity, i.e.  $0.01 T$  as the threshold value. Given that the wind turbines not necessarily operate on full capacity in general, an alternative would be to cluster the input dataset to find the peak point that occurs the highest time, and this peak may be used as the nominal power.

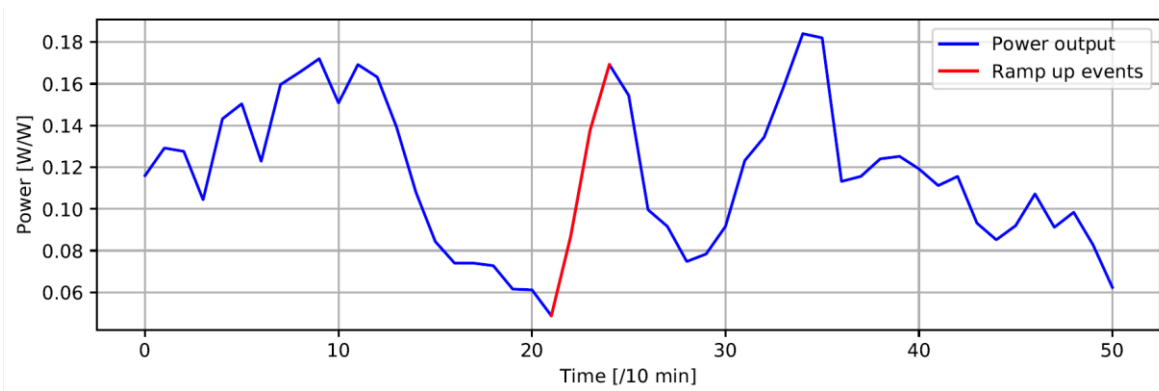


Figure 2.9 Ramp events extracted from the small sample of the data with  $0.12 T$

After extracting events with different  $T$  values from data points of the same size, we observe that the bigger the  $T$ , the fewer events there is.

Table 2.3 Ramp events in Figure 2.9

Event	$w_s(t)$	$\Delta w_s(t + \Delta t)$	$T$	$t + \Delta t$	$\Delta w_s$	$\Delta t$	$\alpha(\Delta w_s)$	$\text{mean}(\Delta w_s)$
1	0.048	0.169	21	24	0.120	3	21.867	0.109

## 2.2.1 Ramp events after preprocessing the data

We decided  $T$  to be 0.1 after several trials and concluded that it preserves the events best to our interest. It can be seen in Figure 2.1 that preprocessing alters the peak points slightly, most of the meaning in the data is preserved.

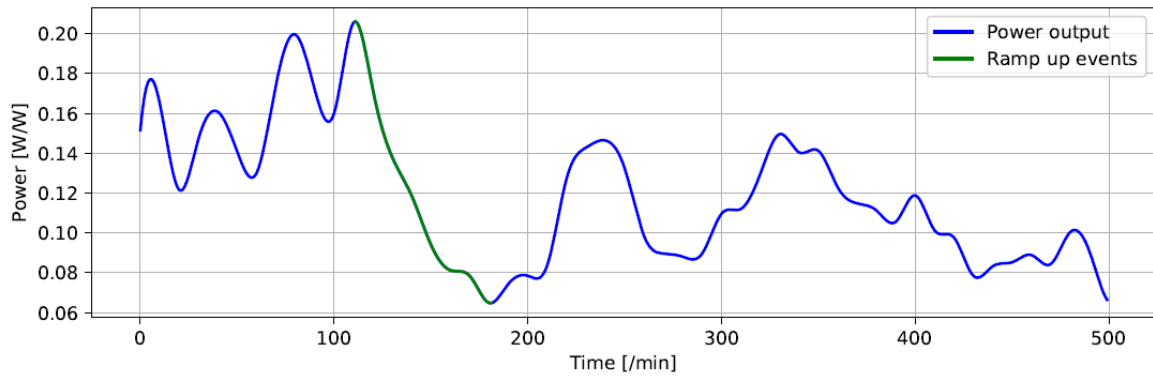


Figure 2.10 Events extracted after preprocessing our data

### 2.3 Ramp events with Rainflow analysis

Rainflow counting algorithm (Downing & Socie, 1982) was developed to be used in the analysis of fatigue data in order to reduce a spectrum of varying stress into a set of simple stress reversals. The input to the algorithm is a simple series of peaks and valleys (troughs), i.e., local maxima and minima, that form hysteresis loops. Closed loops are full cycles, and unclosed loops are half cycles. The algorithm uses a change in slope as an indicator that the time series is going through a peak or valley. Only the magnitude of the peak or valley is then entered into the Rainflow counting algorithm.

We introduced this algorithm to extract ramp events as an alternative method, with modifications to extract the starting and ending points of ramp events hence the time range, the starting and ending power of the events hence the amplitude, the angle of the event, and the cycle with some modification. It was created with the help of (Jennifer Rinker, n.d.).

Table 2.4 Details of Rainflow cycles extracted from the small sample of the data

Event	$w_s(t)$	$\Delta w_s(t + \Delta t)$	t	t + $\Delta t$	$\Delta w_s(t)$	$\Delta t$	$mean(w_s(t))$	Cycle	$\alpha(\Delta w_s)$
1	0.152	0.165	0	1	0.013	1	0.158	0.5	0.779
2	0.165	0.12	1	2	-0.043	1	0.143	0.5	-2.496
3	0.161	0.13	4	6	-0.030	2	0.145	1	-0.882
4	0.199	0.16	8	10	-0.039	2	0.179	1	-1.134
5	0.121	0.20	2	11	0.084	9	0.163	0.5	0.534
6	0.205	0.06	11	18	-0.140	7	0.135	0.5	-1.152
7	0.064	0.146	18	24	0.081	6	0.105	0.5	0.779
8	0.146	0.088	24	28	-0.058	4	0.117	0.5	-0.83
9	0.088	0.109	28	30	0.0216	2	0.098	0.5	0.618

## 2.4 Persistence analysis

Persistence of an event is defined here as how repetitive that event is regarding one of its parameters. Persistence values are calculated for parameters  $\Delta w_s$ ,  $\Delta t$ ,  $\text{mean}(\Delta w_s)$  and  $\alpha(\Delta w_s)$  are calculated from the extracted events, individually with their corresponding ranges. Bins are linearly spaced between the corresponding ranges to the defined number of bins which is set to a default of 100. Bins divide the whole range for the parameter into equivalent ranges. Then if the value equals to the bin value or is in between with the consequential bin, then the persistence value is incremented one. Once all the events are visited, the accumulated persistence values are found.

In short, the persistence value for one parameter of one event is the number of events that has the value for that parameter between the same two bins.

Table 2.5 Ranges of bins of the corresponding event parameters

Event parameter	begins	ends
$\Delta w_s$	-1	1
$\Delta t$	1	$\max(\Delta t)$
$\text{mean}(\Delta w_s)$	$\min(\text{mean}(\Delta w_s))$	$\max(\text{mean}(\Delta w_s))$
$\alpha(\Delta w_s)$	$-90^\circ$	$90^\circ$

---

### Algorithm III: Persistence

---

**Inputs:** X, bins

**For**  $i \leftarrow 0$  **to**  $\text{length}(\text{bins})$  **do:**

**For**  $i \leftarrow 0$  **to**  $\text{length}(\text{array})$  **do:**

**If**  $\text{bins}[i] \leq X < \text{bins}[i+1]$ :

persistence +1

**For**  $i \leftarrow 0$  **to**  $\text{length}(\text{bins})$  **do:**

**For**  $i \leftarrow 0$  **to**  $\text{length}(\text{array})$  **do:**

**If**  $\text{bins}[i] \leq X < \text{bins}[i+1]$ :

$P[X] = \text{persistence}$

**End**

---

To be able to get consistent results for spatial analysis, all turbine outputs are concatenated, processed with ramp extraction method and Rainflow Analysis; then persistence values are parted accordingly to corresponding turbines.

Table 2.6 Persistence values for events in Table 2.1

Event	$\Delta w_s$	$\Delta t$	$\alpha(\Delta w_s)$	$\text{mean}(\Delta w_s)$	$P(\Delta w_s)$	$P(\Delta t)$	$P(\alpha(\Delta w_s))$	$P(\text{mean}(\Delta w_s))$
1	-0.044	1	-23.557	0.143	3058	24948	1467	848
2	0.076	5	8.600	0.167	3934	2033	498	853
3	0.046	1	24.512	0.182	3047	24948	1527	926
4	-0.141	7	-11.373	0.135	2211	388	706	852
5	0.082	6	7.744	0.105	3934	884	353	676
6	-0.058	4	-8.306	0.117	4735	4581	590	759

Table 2.7 Persistence values for events in Table 2.4 Details of Rainflow cycles extracted from the small sample of the data

Event	$\Delta w_s(t)$	$\Delta t$	$\alpha(\Delta w_s)$	$\text{mean}(\Delta w_s)$	Cycle	$P(\Delta w_s)$	$P(\Delta t)$	$P(\alpha(\Delta w_s))$	$P(\text{mean}(\Delta w_s))$	Total Cycle
1	-0.0436	1	-23.557	0.1434	0.5	3058	24948	1467	848	10.5
2	0.0756	5	8.597	0.1678	0.5	3934	2033	498	853	1
3	0.0456	1	24.5129	0.1828	1	3047	24948	1527	926	1.5
4	-0.1408	7	-11.372	0.1352	1	2211	388	706	852	1
5	0.0816	6	7.74471	0.1056	0.5	3934	884	353	676	0.5
6	-0.0584	4	-8.3065	0.1172	0.5	4735	4581	590	759	2



## 3 SCENARIO GENERATION

In this section, we will focus on the generation of forecasts, probabilistic models and scenario generation depending on them.

### 3.1 Artificial neural networks(ANNs)

In the first section, the primary explanation about ANNs will take place. In this section, the basics of ANN will be introduced. The contribution of ANN to this work will be evaluated.

Starting from the second section, recurrent neural networks, capsule networks and generative adversarial networks will be briefly explained for forecasting, and the contribution of these architectures to this work will be explained.

#### 3.1.1 Basics with ANNs

To be able to explain what an ANN is, we should introduce some concepts first. Then we will progress towards ANN. As the most basic definition, ANNs are biologically inspired networks that mimic the human brain. The smallest sub-unit in the human brain is a neuron. In the ANN terminology, those are called as perceptrons. In the following sub-section the smallest sub-units of networks will be explained and after that will have a better understanding of how ANN works and why those are powerful and robust structures.

#### Perceptrons

Perceptron (as shown in Figure 3.1) is one of the artificial neuron models as already mentioned. In this part, the perceptron will be explained as an idea, and the mathematics behind will be studied.

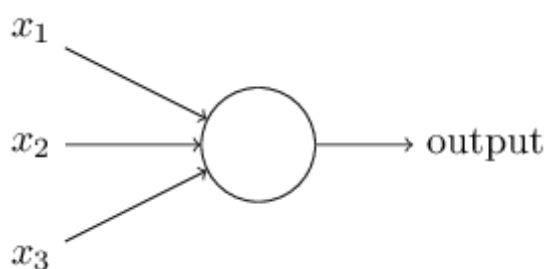


Figure 3.1 The representation of perceptron (Nielsen, 2015).

The working mechanism of a perceptron is quite simple, it takes inputs and produces output. Input and output are in binary form. Weights  $w_i$  is used to express the output. Sum of the multiplied inputs by their weights give a result to determine the output as explained in Equation (3.1) (Nielsen, 2015). We can see that perceptron is able to make a decision based on our defined threshold. If the result is bigger than the threshold, then the perceptron gives out a 1.

$$output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq threshold \\ 1 & \text{if } \sum_j w_j x_j > threshold \end{cases} \quad (3.1)$$

$$w \cdot x = \sum_j w_j x_j \quad (3.2)$$

$$output = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \quad (3.3)$$

Where  $w$  is a vectorial representation of the weight,

$x$  is a vectorial representation of the input,

$b$  is bias and  $b \equiv -threshold$ .

Perceptrons can be used as an elementary function. For instance, if there is a perceptron which has two inputs and one output. Both inputs has one value, the given weights are -2, and the bias value is 3. The output value is going to be  $(-2) * 1 + (-2) * 1 + 3 = -1$ . Since -1 is smaller than 0 output value of the perceptron is going to be 0. In this case, our perceptron behaves as NAND logic gate. It is also possible to program the perceptrons as AND, OR gate. NAND gates are used universally for computations. Hence a perceptron can be a universal computation unit(Nielsen, 2015).

## Sigmoid Neurons

In most cases, we want the change in the output of the network to be in accordance with the change in the weights or bias. The problem with perceptrons is that when there is a change in a weight or bias, it might affect the entire result, even to a point where correct outputs become incorrect. There is another model of neuron that overcomes this problem and allows us to change weights and bias effectively so that its output changes but does not cause dramatic changes in entire output. They function similarly with perceptrons. This small change in the behaviour of the neuron makes quite a difference in the network, gives it a chance to learn better.

Let's assume that presented neuron in Figure 3.1 is sigmoid neuron. Sigmoid neuron also has inputs, weights, bias, and output. The only thing is that the input of the sigmoid neuron can get any values in a range of 0 to 1. Since the input not binary, the output gets calculated by using Equation (3.4) by using a sigmoid function which is given in Equation (3.5).

$$output = \sigma(w \cdot x + b) \tag{3.4}$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}} \tag{3.5}$$

If we use Equation (3.5) in Equation(3.4), the Equation (3.6) will give the output of a sigmoid neuron.

$$output = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)} \tag{3.6}$$

### 3.1.2 Neural network architecture

Perceptrons can be connected as layers to each other as shown in Figure 3.2. The leftmost layer is called input layer that corresponds to inputs. In the middle, we have a hidden layer. The reason for calling it as the hidden layer is because while the information just passes on it, there is no “real world” meaning for humans to comprehend. There might be several hidden layers depending on the application. The rightmost layer is output layer which makes decisions. This concept is called multilayer perceptrons (MLPs).

This concept takes the input from the first layer and passes on to the hidden layer, and finally to the output layer which makes the decision as mentioned. This kind of straight network is called a feed-forward network. There is no connection between neurons in the same layer, and the information always goes straight. However, some networks have neurons which are connected to each other in the same layers, and those are called as recurrent neural networks (RNNs). Since in our task we use RNN type network, it is explained in the next section.

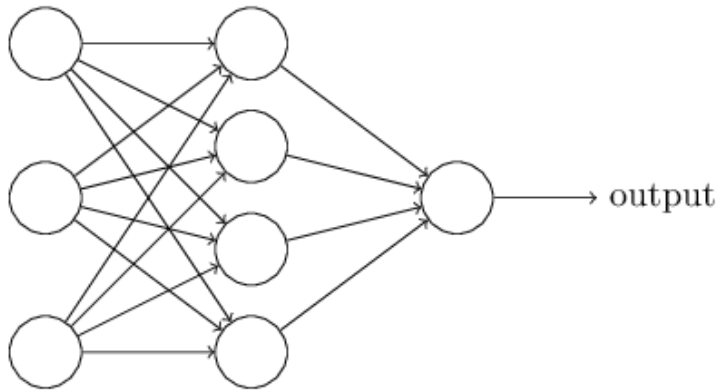


Figure 3.2 Multilayer perceptrons

Assume that we have an input in a 25-by-25 matrix. Then the hidden layer consists of  $25 \times 25 = 625$  neurons. If we have complex tasks, then we can use more hidden layers. If there are many neurons in a layer, it makes the network memory greater, but if there are many layers, then it makes the network more powerful.

### Gradient descent

Gradient descent is an optimisation algorithm used to find the values of parameters (coefficients) of a function ( $f$ ) that minimises a cost function (cost). Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimisation algorithm.

Gradient descent is an optimisation algorithm that used to find the parameters that are not linear. Gradient descent aims to minimise the cost function. To achieve that, it finds values of parameters of a function  $y$ . With the help of this function, the cost function gets minimised. After minimisation of the cost function, weights and biases of the network get updated and generate a better pattern. Therefore gradient descent is the essential parameter in a network. Let's say that  $y(x)$  the desired output of a network where  $x$  represents inputs of the network. The cost function can be written as shown in Equation (3.7).

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2 \quad (3.7)$$

Where  $w$  represents all the weights in network,

$b$  all biases,

n total number of inputs,

x is the input vector of the network,

a is the output vector of the network,

C is the quadratic cost function of mean squared error (MSE).

Since  $a = w \cdot x + b$ , the output value of the network depends on the weight and bias factor on the independent value of the input, if we find the correct values of weight and bias parameters, our output value of the network will be equal to the input value. Once we get this stage, see Equation (3.7),  $\|y(x) - a\|$  part of the function will be zero, but in reality, it is hardly possible to get it equal to zero. Therefore, cost function tries to minimize it. During this training phase, updating the weights and biases with the reaction of the algorithm helps to make the cost function get closer to zero. When the cost function is closer to zero, then the system will take the weights and biases as learned parameters and use for the test data or predict/classify the inputs (Nielsen, 2015).

For the further information about how gradient works let's keep the function simple and call it as  $v$  and cost function  $C(v)$  instead of  $C(w, b)$ . Assume that  $v$  changes small amount  $\Delta v$ , in the  $v$  direction. This small change will affect the cost function as shown in Equation (3.8), and gradient of cost function can be expressed as shown in Equation (3.9).

$$\Delta C \approx \frac{\partial C}{\partial v} \Delta v \quad (3.8)$$

$$\nabla C \equiv \left( \frac{\partial C}{\partial v} \Delta v \right)^T \quad (3.9)$$

Where  $\nabla C$  is the gradient vector,

T is the transpose operation.

$\Delta C$  can be written as shown in Equation (3.10).

$$\Delta C \approx \nabla C \cdot \Delta v \quad (3.10)$$

## 3.2 Recurrent neural networks (RNNs)

RNNs might be less potent than feed-forward networks but they are applicable in some cases which require communication between each neuron as it is in our task.

### 3.2.1 Forecasting with Long-short term memory (LSTM)

LSTM is a specified RNN architecture, and it is one of the most applicable networks for forecast time series. RNN is possibly the closest network that mimics the human brain. As it can be understood from the term “long-short”, the aim of this network is to model temporal sequences and their long-term dependencies. LSTM uses stochastic gradient descent as an optimisation tool. The reason behind RNN outperforms than feed-forward networks is it contains cyclic connections. Thanks to these connections it is easier to model sequence data (Beaufays, Sak, & Senior, 2014).

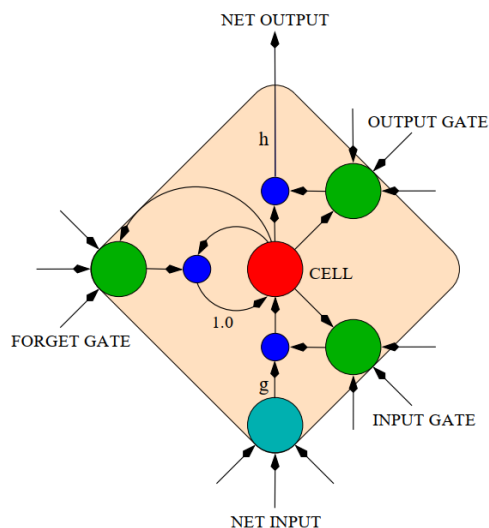


Figure 3.3 One cell LSTM memory block (Graves et al., 2009)

An LSTM type hidden layer has subnets that are recurrently connected as shown in Figure 3.3. These subnets have a set of cells whose activation is controlled by an input gate, forget gate and output gate. Gates affect the cells ability to store and access information for a long term. What it means is that when an input is absent, the input gate stays closed, and there is no overwriting on the activation cell. Cell activation is available as long as output gate stays open, and forget gate switches on and off the recurrent connection of the cell (Graves et al., 2009).

The architecture of LSTM that we used in this thesis work is deep LSTM (DLSTM) which has two LSTM in the architecture. As it can be seen in representation below, each LSTM forwards its output to next step and takes it as input to itself too.

$INPUT \rightarrow LSTM^{\wedge} \rightarrow LSTM^{\wedge} \rightarrow OUTPUT$

As an architecture, this can be considered as a feed-forward network. Within this configuration, networks can learn parameters from input at different time scales (Hermans & Schrauwen, 2013). Keras (Keras) is used for this task.

(Pascanu et al., 2013)

We implemented a simple LSTM network with two LSTM layers and a fully connected output layer. The outputs in Figure 3.4, Figure 3.5 and Figure 3.6 shows forecasts for  $\Delta w_s$ ,  $\Delta t$  and  $\alpha(\Delta w_s)$  which are the most important features of events. The forecasts are following the real inputs behaviour which we concluded that this structure was satisfactory for this work.

Many adjustments can be made for this network. The number of iterations, the number of epochs, the number of layers and the neuron size in the layers, batch size, a different kind of activation function and a different kind of an optimiser, adding a classifier that would make it a supervised machine learning structure are few of the many.

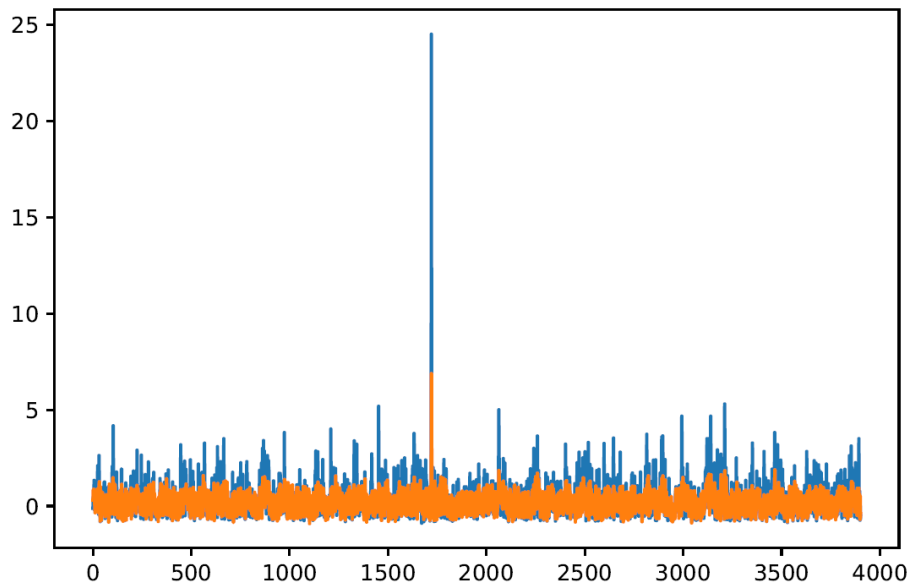


Figure 3.4 Forecast with LSTM for time ranges

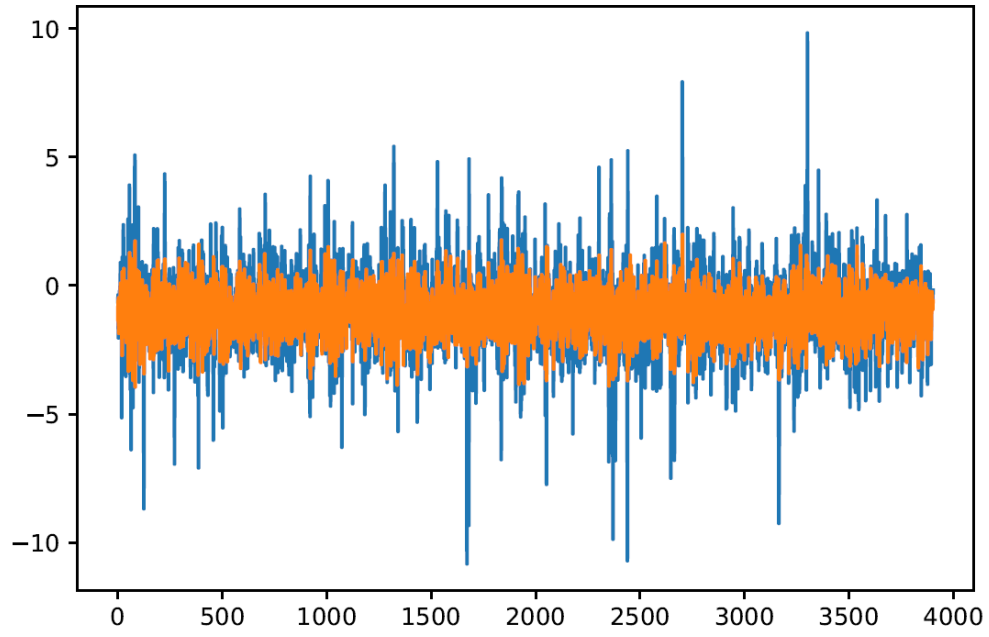


Figure 3.5 Forecast with LSTM for changes in amplitude

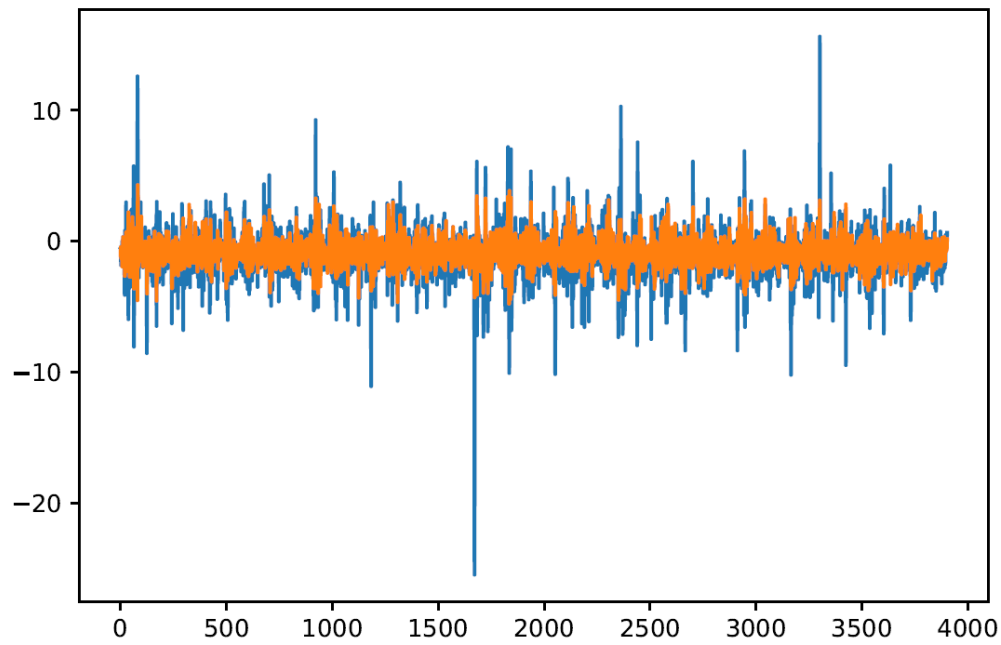


Figure 3.6 Forecast with LSTM for the angle of events



### 3.3 Forecasting with Generative adversarial network (GAN)

The generative adversarial network consists of two networks, generative one and discriminator one. These two networks compete, one tries to overcome the other one. Therefore, it is called adversarial. It is proposed by (Goodfellow et al., 2014).

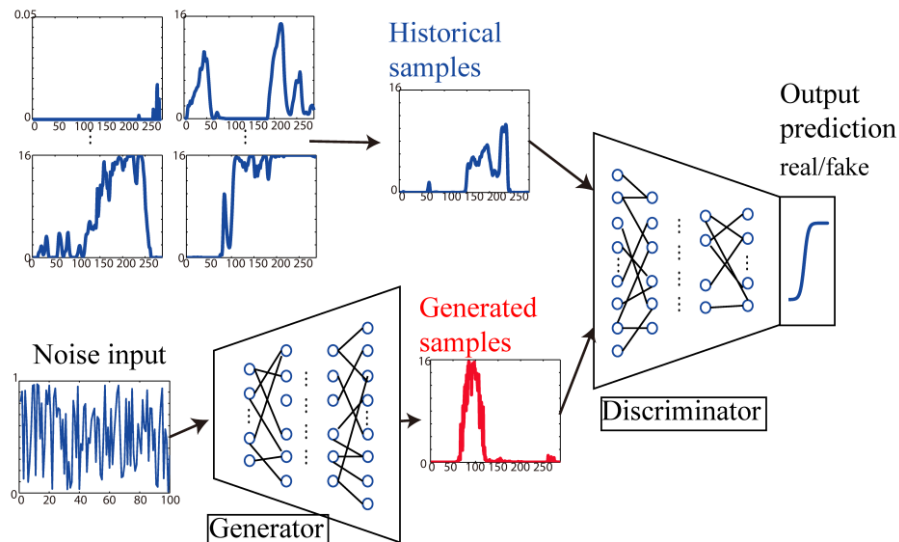


Figure 3.7 GAN working principle (Chen et al., 2018)

According to (Goodfellow et al., 2014) GAN's generator model uses an adversarial process. Both networks get trained at the same time. Data distribution is captured by the generator model, and the discriminator model estimates the data whether it came from the generator model or the historical data. Training procedure of the generator network aims to maximise the probability of discriminator to make a mistake.

Backpropagation (Rumelhart, Hinton, & Williams, 1986) method can be used while training the entire system. Dropout (Srivastava et al., 2014) took place to prevent overfitting and to memorise the data in the network.

In this network, the generator produces as good made up data as it can, then sends it to the discriminator. While generator expects discriminator to accept it, discriminator defines the data either it comes from historical data or discriminator. In this game, if discriminator does not accept the data which comes from the generator, then the generator gets feedback to produce better fake data. This competition allows both networks to readjust themselves. At one point, it gets harder for the discriminator to predict, and the generator to generate better. In this stage, we get a model which is very close the real dataset which discriminator does not understand if it is real or fake.

Since this network is compatible with many artificial intelligence tasks and scenario generation is one of these tasks, it makes this system applicable to our task. It helps to create a model which can apply scenario generation. Therefore, this network will be using for the scenario generation task.

Both networks use a type of ANN which is called multilayer perceptron (MLP). This concept is explained in the previous section. Both networks uses backpropagation and dropout methods while training. For the sample which comes from the generative model is used forward propagation (Goodfellow et al., 2014).

After training the network for 500 iterations which are quite low for a network to converge, the discriminator loss (Figure 3.9) diverges from a meaningful value. Even though it can be seen in Figure 3.8 that the probabilistic distribution functions of generated and real samples seem close, generated values were too far away from the real values. Either this network needs to be deeper with more layers which is a complicated task, or it is not suitable for our data.

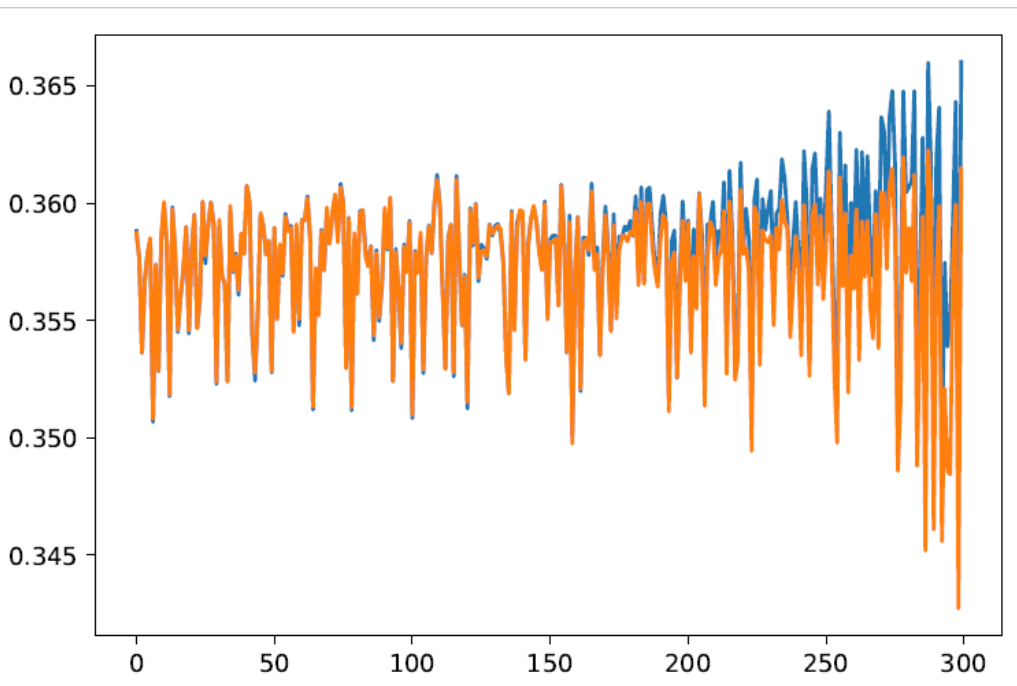


Figure 3.8 Distribution functions of the generator(blue) and the discriminator(orange)

---

**Algorithm 1** GAN minibatch stochastic gradient descent training

---

**Input:**  $k$  is a hyperparameter which identifies the number of steps that will be applied to the discriminator, and it is defined as  $k=1$ .

**for** number of training iterations **do**

**for**  $k$  steps **do**

        Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ ,

        Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating a distribution  $p_{data}(x)$ .

        Update the discriminator by increasing its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$$

**end for**

    Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ ,

    Update the generator by decreasing its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)})))$$

**end for**

---

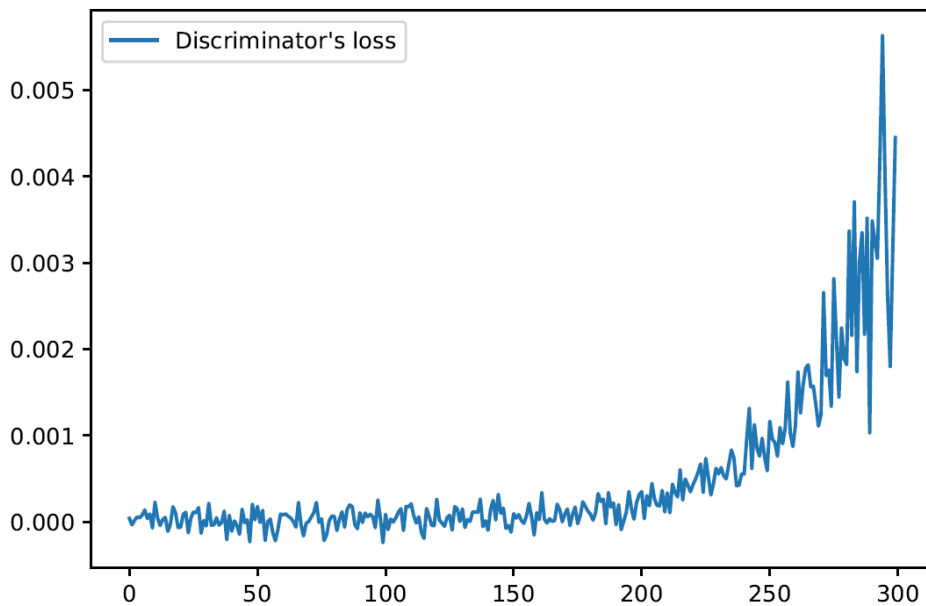


Figure 3.9 Discriminator loss

## 3.4 Scenarios with Monte Carlo Markov Chains(MCMC)

### 3.4.1 Markov Chains

A first-order Markov chain is the realization of the stochastic process in the discrete data  $x$  where every discrete point is attained to a discrete state value  $S=1,\dots,m$ . Here, the state of the process depends only on the previous state and a conditional probability.

### 3.4.2 Spatial Markov

A shapefile in Figure 3.10 is made with 17 random turbines chosen from Paldiski wind farm using the open source geographic information system, QGIS. A commonly-used type of weights is Queen-contiguity weights, which reflects adjacency relationships as a binary indicator variable denoting whether or not a polygon shares an edge or a vertex with another polygon. These weights are symmetric. The neighbouring relationships are shown in Figure 3.11.

Spatial Markov can show the correlations between the features of the events we extracted. Most importantly, it can help show the correlations between turbine locations and can be used for long and short-term probabilistic scenario generations for different turbines. We implemented LISA Markov which is based on Moran's  $I$  and show the transition probabilities while placing each areas behaviour into quantiles based on whether they behave similarly to their neighbours or not, and a basic spatial Markov which can show the relationship between the global and regional relationships of transition dynamics(Pysal, n.d.).

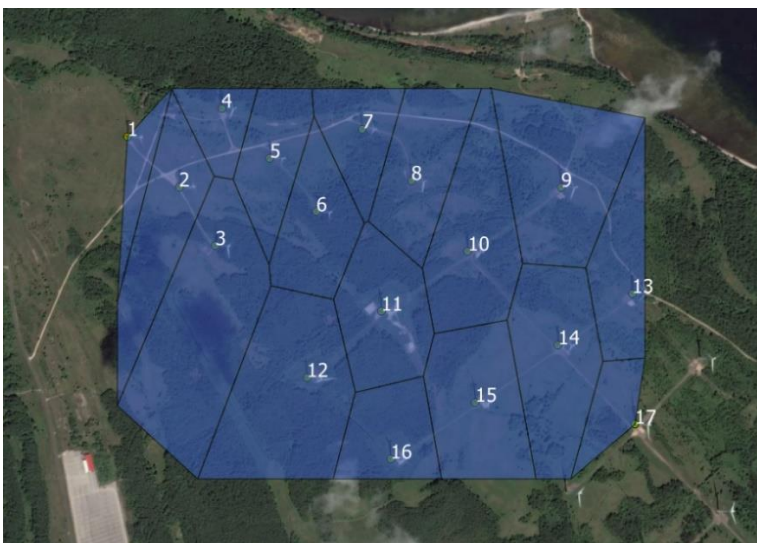


Figure 3.10 Representation of areas defined for each turbine in the shapefile used in spatial analysis with corresponding numbers

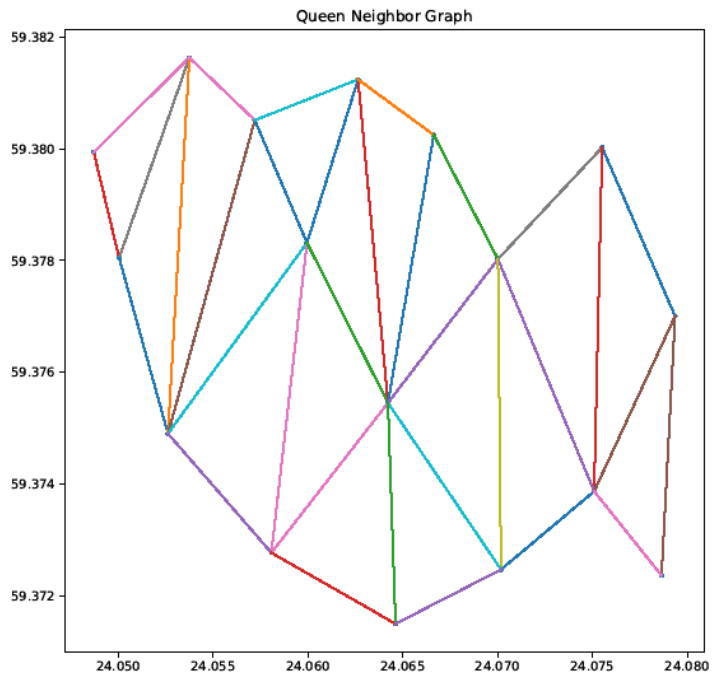


Figure 3.11 Neighbouring relationships based on weights with Queen principal

## 4 CONCLUSIONS

We have improved the wind ramping behaviour analysis by introducing new indicators for events. We introduced forecasting events with extracted indicators rather than raw data. The advantage here is that there is more meaning in the forecasts we generate from the point of significant ramp events rather than small fluctuations in the generation, which was our thesis goal.

### 4.1 Future work

As a possible future implementation, Capsule networks could be added to the structure of GAN to make it able to converge with a smaller amount of data. Also, features can be clustered in a multidimensional way and scenarios can be generated using a GAN architecture that includes a classifier discriminator that would draw its classes from these clusters.

### 4.2 Summary

In the first chapter, we introduced the main idea and the motivation behind this thesis work. Thesis' task definition is clarified, and the goal is specified as well. Overviewed of the literature, which has already become a milestone in this field, and previous researcher related to this thesis task. In the overview part, we explained the essential concepts and methods which are related to the topic.

Based on the defined tasks in the first chapter and drawn boundaries, after introducing our data, we filtered it from smaller fluctuations using cubic B-spline polynomials and Fourier transform. After smoothing our data, we defined what a significant ramp event is and what the features we will extract. We also introduced an alternative method that can be used for event extraction which is called Rainflow counting that is typically used for fatigue analysis. These extracted features are used for further analysis which was explained and methods were built up as we went further. After event extraction, we defined a new indicator called persistence, and we calculated them according to the event parameters. At the very end of this chapter, we had a set of algorithms applied to our data, and that enabled us to progress within the next chapters.

Explanation of the concept, NNs took place in the third chapter with its boundaries and specific parameters. The result of this chapter is going to get the finalised algorithm to progress with experimentation section.

In the fourth chapter, our chosen type of Neural Networks(NN), a GAN (Chen et al., 2018) and a common Long Short Term Memory(LSTM) were tested to develop our forecast. LSTM model performed better than GAN for now. After generation of forecasts, we proposed to use a multivariate MCMC, to generate a set of possible scenarios.

In the concluding chapter, future works will be mentioned based on what more could be done within the scope of this thesis work. "What could be better?" and "How to make it more efficient?" are going to be answered in this part of the research.

Moreover, of course, the summary is another must at the very the end of this thesis.

## 5 BIBLIOGRAPHY

Agarwal, P., Singh, S. P., & Pandey, V. kumar. (2014). Mathematical analysis of blackman window function in fractional Fourier transform domain. In *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)* (pp. 120–125). <https://doi.org/10.1109/MedCom.2014.7005988>

Beaufays, F., Sak, H., & Senior, A. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has. *Interspeech*, (September), 338–342. <https://doi.org/arXiv:1402.1128>

Bianco, L., Djalalova, I. V., Wilczak, J. M., Cline, J., Calvert, S., Konopleva-Akish, E., ... Freedman, J. (2016). A Wind Energy Ramp Tool and Metric for Measuring the Skill of Numerical Weather Prediction Models. *Weather and Forecasting*, 31(4), 1137–1156. <https://doi.org/10.1175/WAF-D-15-0144.1>

Bossavy, A., Girard, R., & Kariniotakis, G. (2013). Forecasting ramps of wind power production with numerical weather prediction ensembles. *Wind Energy*, 16(1), 51–63.

Bossavy, A., Girard, R., Kariniotakis, G., & Antipolis, S. (2013). A novel methodology for comparison of different wind power ramp characterization approaches. In *EWEA 2013 - European Wind Energy Association annual event* (pp. 4–7). <https://doi.org//10.1002/we.526>

Chen, Y., Wang, Y., Kirschen, D., & Zhang, B. (2018). Model-Free Renewable Scenario Generation Using Generative Adversarial Networks. *IEEE Transactions on Power Systems*, 33(3), 3265–3275. <https://doi.org/10.1109/TPWRS.2018.2794541>

Cui, M., Ke, D., Sun, Y., Gan, D., Zhang, J., & Hodge, B. (2015). A scenario generation method for wind power ramp events forecasting. In *2015 IEEE Power & Energy Society General Meeting* (pp. 1–5). <https://doi.org/10.1109/PESGM.2015.7285818>

Cui, M., Zhang, J., Florita, A. R., Hodge, B. M., Ke, D., & Sun, Y. (2016). An optimized swinging door algorithm for identifying wind ramping events. *IEEE Transactions on Sustainable Energy*, 7(1), 150–162. <https://doi.org/10.1109/TSTE.2015.2477244>

Downing, S., & Socie, D. F. (1982). Simple rainflow counting algorithms. *International Journal of Fatigue*, 4(1), 31--40. [https://doi.org/10.1016/0142-1123\(82\)90018-4](https://doi.org/10.1016/0142-1123(82)90018-4)



European Commission. (n.d.). 2020 Energy strategy. Retrieved May 25, 2018, from <https://ec.europa.eu/energy/en/topics/energy-strategy-and-energy-union/2020-energy-strategy>

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Networks, 1–9. <https://doi.org/10.1001/jamainternmed.2016.8245>

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), 855–868. <https://doi.org/10.1109/TPAMI.2008.137>

Hermans, M., & Schrauwen, B. (2013). Training and Analyzing Deep Recurrent Neural Networks. *Nips*, 190–198.

Jennifer Rinker. (n.d.). Rainflow cycle counting. Retrieved February 10, 2018, from <https://gist.github.com/jennirinker/688a917ccb7a9c14e78f>

Karatepe, S., & Corscadden, K. W. (2013). Wind Speed Estimation: Incorporating Seasonal Data Using Markov Chain Models. *ISRN Renewable Energy*, 2013, 1–9. <https://doi.org/10.1155/2013/657437>

Kaut, M. (2014). A copula-based heuristic for scenario generation. *Computational Management Science*, 11(4), 503–516. <https://doi.org/10.1007/s10287-013-0184-4>

Keras. (n.d.).

McClellan, J. H., Schafer, R. W., & Yoder, M. A. (2003). *Signal Processing First*.

Mishra, S., Leinakse, M., & Palu, I. (2017). Wind power variation identification using ramping behavior analysis. In *Energy Procedia* (Vol. 141, pp. 565–571). Elsevier B.V. <https://doi.org/10.1016/j.egypro.2017.11.075>

Nelja Energia. (n.d.). Paldiski wind farm. Retrieved May 7, 2018, from <https://www.4energia.ee/en/projects/pakri-wind-park>

Nielsen, A. M. (2015). *Neural Networks and Deep Learning*. Determination Press.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to Construct Deep Recurrent Neural Networks, 1–13.

Prautzsch, H., Boehm, W., & Paluszny, M. (2002). Bézier and B-Spline Techniques. <https://doi.org/10.1007/978-3-662-04919-8>

Pysal. (n.d.). Spatial Dynamics. Retrieved May 20, 2018, from <http://pysal.readthedocs.io/en/latest/users/tutorials/dynamics.html>

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533.

Scipy.org. (n.d.). Blackman window. Retrieved May 25, 2018, from <https://docs.scipy.org/doc/scipy-0.16.1/reference/generated/scipy.signal.blackman.html>

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929–1958. <https://doi.org/10.1214/12-AOS1000>

U.S.NRC. (2015). Capacity factor. Retrieved May 7, 2018, from <http://www.nrc.gov/reading-rm/basic-ref/glossary/capacity-factor-net.html>