The present work was submitted to the LuFG Theory of Hybrid Systems

WindProof: A Validation Framework for Wind Farm Simulations

WindProof: Ein Validierungs-Framework für Windpark-Simulationen

Bachelor of Science Thesis

November 4, 2024

Presented by Vorgelegt von	Arne Leon Matrikelnummer: 422205 arne.leon@rwth-aachen.de
First examiner Erstprüfer	Univ.-Prof. Dr. rer. nat. Dr. h.c. Erika Ábrahám LuFG Theory of Hybrid Systems RWTH Aachen University
Second examiner Zweitprüfer	Apl. Prof. Dr. Ralf Schelenz Lehrstuhl und Institut für Maschinenelemente und Systementwicklung RWTH Aachen University
Supervisor Betreuer	Nicolai Radke LuFG Theory of Hybrid Systems RWTH Aachen University

Eidesstattliche Versicherung
Declaration of Academic Integrity

Name, Vorname/Last Name, First Name

Matrikelnummer (freiwillige Angabe)
Student ID Number (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare under penalty of perjury that I have completed the present paper/bachelor's thesis/master's thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt; dies umfasst insbesondere auch Software und Dienste zur Sprach-, Text- und Medienproduktion. Ich erkläre, dass für den Fall, dass die Arbeit in unterschiedlichen Formen eingereicht wird (z.B. elektronisch, gedruckt, geplottet, auf einem Datenträger) alle eingereichten Versionen vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without unauthorized assistance from third parties (in particular academic ghostwriting**.** I have not used any other sources or aids than those indicated; this includes in particular software and services for language, text, and media production. In the event that the work is submitted in different formats (e.g. electronically, printed, plotted, on a data carrier), I declare that all the submitted versions are fully identical. I have not previously submitted this work, either in the same or a similar form to an examination body.

Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen/Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
§ 156 StGB (German Criminal Code): False Unsworn Declarations
Whosoever before a public authority competent to administer unsworn declarations (including Declarations of Academic Integrity) falsely submits such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment for a term not exceeding three years or to a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
§ 161 StGB (German Criminal Code): False Unsworn Declarations Due to Negligence
(1) If an individual commits one of the offenses listed in §§ 154 to 156 due to negligence, they are liable to imprisonment for a term not exceeding one year or to a fine.
(2) The offender shall be exempt from liability if they correct their false testimony in time. The provisions of § 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date

Unterschrift/Signature

Abstract

As the need for renewable energy rises, so does the need for proper planning and optimization. A renewable energy farm, such as a wind farm, can be simulated to ease and accelerate these processes. Several simulation tools that work with multiple wind models are already available. But, of course, the simulation results require high accuracy for such high-stakes calculations. To validate the tools, this thesis presents WindProof. WindProof is a general framework that enables a user to empirically examine wind farm simulations by only implementing the access point of that tool into WindProof. Then, WindProof tests the tool on a broad range of test cases and compares simulation results between tools. This process will either validate the tool or support the troubleshooting process of the found differences.

Acknowledgements

I would like to express my heartfelt gratitude to Prof. Dr. Dr. h.c. Erika Ábrahám for making this thesis possible and serving as the first examiner. I also extend my thanks to Prof. Dr. Ralf Schelenz for acting as the second examiner.

I am especially grateful to Nicolai Radke for his guidance and motivation throughout this thesis journey and for allowing me to take it on. I want to thank you also for being understanding of the hurdles encountered on the way. The journey could not have even begun without you.

I would also like to thank Patrick de Smet for his invaluable starting tips and engaging conversations about train rides.

Next, I want to acknowledge my big brother, Philip, for his moral support, helpful advice for investing the time to proofread this thesis several times, and for powering through the bad first draft.

Lastly, I must express my profound appreciation to my girlfriend, Ewa. You stood by my side through this process, proofread for me, and withstood my bad mood when nothing seemed to work. Your unwavering support has been essential to my progress, and I could not have come this far without you.

Contents

Introduction

Wind energy production is on the rise worldwide. As of 2022, the worldwide production of wind energy has risen to 2.098 petawatt hours. This amount is even more impressive because compared to only five years earlier, the production has risen by more than 85% [16].

The number of wind turbines in Germany grows yearly, with 745 installed just last year [1]. With that, wind power became Germany's most significant source of electricity, with 31% of total energy production [2]. To conclude, wind power is an essential player in renewable energies.

It follows that the development of new wind farms is significant. These wind farms should also be as efficient as possible to ensure maximal power output. Once a suitable site has been chosen, one of the available optimization algorithms is used [18]. These algorithms fine-tune the turbine positions within the site to maximize the Annual Energy Production (AEP). To optimize the turbine layout, it follows that the algorithm needs to calculate the AEP of a concrete layout.

Typically, wind farm simulations are used to fulfill this purpose. These wind farm simulations use different wind models as a basis for their calculations [6]. One approach is the modeling of the wind as a fluid using Computational Fluid Dynamics (CDF) models based on Large Eddy Simulations (LES) [17] or Reynolds-averaged Navier-Stokes (RANS) equations. However, these models are untypical for optimizing wind farms due to their high computational intensity compared to other methods. The second approach is using analytical models, such as the Gaussian wake model [5, 12] or the Jensen wake model [7, 9], which are based on observations and correlations of empirical data [6]. The Jensen wake model will be the focus of this thesis.

Numerous papers validate and compare analytical models and CDF models with measured wind farm data [4, 8, 15]. While this research is essential for ensuring correct wind farm models, there is another step to an accurate simulation. This step is the implementation of the wind farm models, which can introduce many errors. However, to our knowledge, no paper compares and validates the implementation of these wind farm simulation models.

This thesis addresses this gap. However, it would be shortsighted to only compute and compare the results of a handful of wind farm simulation software, especially given the amount of recent research in this area. With these developments in mind, this thesis presents WindProof, a general framework for validating wind farm simulation software. WindProof's goal is to allow for accessible empirical validation of several wind farm simulation tools. WindProof's general design allows more straightforward extensions of new simulation software, wake models, and test cases. The validation is based on a library of test cases WindProof provides and many randomly generated test cases, which point out and isolate errors. This information then allows for a systematic error search, identifying any error introduced during implementation.

The simulation tools considered for the comparison are: `PyWake`, an open-source wind farm simulation tool from the DTU Wind, Technical University of Denmark [13]. `WindFarm3D`, a wind farm planning tool developed by LuFG Theory of Hybrid Systems

at RWTH University [19]. Moreover, `WindPro`, a commercial wind farm simulation tool from EMD International [3]. Alas, gaining sufficient access to `WindPro` through its Python API was impossible. More details can be found in Section 4.1. Unfortunately, this thesis could only consider tools with a Python access point, as WindProof is written in Python.

▶ Outline

Section 1 will reiterate the foundations needed for WindProof and describe the Jensen wake model used in this thesis in detail. Section 2 will then describe the design of WindProof and note implementation details that align WindProof with its design goals. In Section 3, WindProof's library of test cases is presented. As an empirical validation framework, a significant focus lies on this section since WindProof's success is directly tied to the quality of the test cases. The thesis will also mention how and what part of the calculations each test case covers. Section 4 will then use WindProof to compare two wind farm simulation tools and identify any difference in their implementations.

1 Preliminaries

The preliminaries will make definitions of the mathematical concepts used in this thesis as well as the Jensen wake model.

1.1 Mathematical Foundations

This first subsection will define all the mathematics needed.

1.1.1 Probability Distributions

Let *dist* be a probability distribution and X be a random variable. When X is distributed like *dist*, it is noted as $X \sim dist$. $X \sim dist$ is then defined by the point density function, abbreviated to PDF, of *dist* being $f^X(x)$. $f^X(x)$ describes how likely the value of X falls into a small area around x; please note that the PDF does not represent the probability of X taking the value x, which would be $P(X = x)$. Instead, the probability of X can be calculated assuming a value within an interval, e.g., $P(2 \leq X \leq 5)$, which denotes the probability of X having a value between 2 and 5.
A distribution function or cumulative density function $F^X(x)$, abbreviated to CDF, is needed to calculate this probability. The value of $F^X(x)$ tells you how likely it is that X assumes a value less or equal to x, so $F^X(x) = P(X \leq x)$. The CDF is calculated based on the PDF. As the name suggests, one sums all the probabilities of all values x or less in the discrete case. In the continuous case, $F^X(x) = \int_{-\infty}^{x} f^X(x)dx$ defines the CDF.

1.1.2 Squared Sum

Let $x_1, \ldots, x_n \in \mathbb{R}$, then the squared sum of $x_1, \ldots, x_n$ is $x_{ss} = \sqrt{\sum_{i=1}^{n} x_i^2}$.

1.1.3 Measure & Negligible Errors

As a measurement of accuracy, this thesis will take the relative percentage difference in AEP of the two simulation tools, based on [11]:

$$\text{diff}(v_1, v_2) = \begin{cases} 0, & v_1 = v_2 = 0, \\ 100 \cdot \frac{|v_1 - v_2|}{(v_1 + v_2)}, & otherwise. \end{cases} \tag{1.1}$$

In this thesis, this difference measure is interpreted the following way: $\text{diff}(v_1, v_2) = 0$ states that the results are identical, $\text{diff}(v_1, v_2) = 100$ means that one result is zero and the other has a non-zero value. In general a lower $\text{diff}(v_1, v_2)$ is desirable.
Floating point errors often occur in the evaluation; therefore, any error $e < 10^{-10}$ is negligible.

1.2 Wind Data Foundations

This section will define structures for describing and modeling wind behavior.

1.2.1 Weibull Distribution

The Weibull distribution is defined by two parameters, k and λ, and the PDF:

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{1.2}$$

It follows that the CDF of Weib(k, λ) is defined by:

$$F(x) = \begin{cases} 1 - e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{1.3}$$

The λ parameter controls the shift of the curve. By increasing it, the PDF's curve flattens.
The k parameter controls the shape of the curve. By increasing it, the curve of the PDF forms a steeper but thinner spike. Figure 1 depicts the PDF curves of some Weibull distributions.

1.2.2 Distribution Vector

In this thesis, a distribution vector denotes the relative probability of each wind speed occurring. Since the space of wind speeds is continuous, binning is applied to discretize the wind speeds.

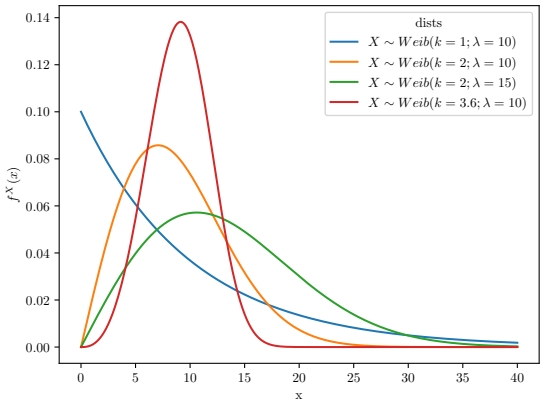


Figure 1: Point density functions of different Weibull distributions

1.2.3 Surface Roughness

Another value needed later is surface roughness, which describes how rugged a terrain is. It is quantified by the mean squared error of the terrain's elevation relative to a plane, minimizing this error.

1.2.4 Windroses

A windrose is a widespread way to describe the local wind at a site. Since the angle of the incoming wind is continuous, a windrose bins the angles of the incoming wind. Therefore, the windrose separates the surrounding area into several evenly distributed wind sectors. For example, if a windrose has 12 sectors, each of those sectors would have a width of 30°. The wind coming from a sector is combined and then assumed to originate from the center line of its corresponding sector. This thesis always places the first wind sector so the center line points to 0° north.
The windrose holds a different wind distribution for each sector, represented by either a distribution vector or a Weibull distribution.
Additionally, each sector $1, \ldots, n$ also has a sector probability p_i, which denotes the relative frequency of the wind coming from the i-th sector. Since the sector probabilities are relative, it holds true that $\sum_{i=1}^{n} p_i = 1$.

1.2.5 Wind Turbines

Each wind turbine has a *hub height*, the height of the turbine hub, the part of a wind turbine where the rotor blades are mounted and the generator is located. Additionally, each turbine has a *rotor diameter*.
Each turbine T also has performance data in the form of three curves provided by the manufacturer after testing: The power-, c_t-, and c_p-curve. Figure 2 displays the curves of the Vestas V112-3.45 turbine.

The *power-curve* T_p of a turbine defines the hourly power output of a turbine in terms of wind speed. The c_t-*curve* T_{ct} defines the thrust coefficient of the turbine for each wind speed. This value is the percentage of loss in wind thrust when powering the rotor blades. The third curve, the c_p-*curve* T_{cp}; its value, the power coefficient, depicts how much wind potential energy is converted into rotational energy at each wind speed.

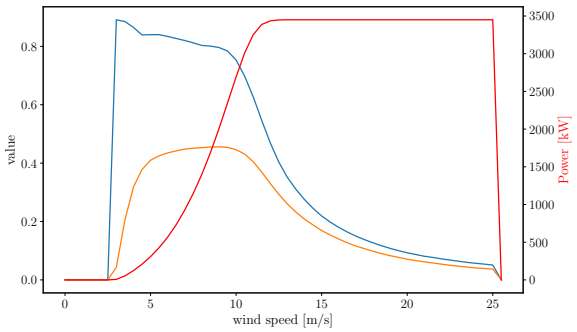


Figure 2: Power-curve (red), c_t-curve (blue) and c_p-curve(yellow) of the Vestas V112-3.45 turbine

All curves are continuous, but values are only supplied at specific anchor points; therefore, the values are interpolated linearly between anchors. The three curves implicitly define the last two defining values of a turbine. These are the *cut-in* and *cut-out* speeds, the lower and upper limits of the wind turbine's operative wind speed interval. Outside

of the interval defined by the *cut-in* and *cut-out* speeds, the turbine is not producing electricity. Therefore, all three curves have a value of 0 outside of this interval.

1.2.6 AEP Calculation

AEP itself stands for Annual Energy Production, and, as the name suggests, this is the energy output of a wind farm or turbine over a full year. In this thesis, the calculation assumes that one year has exactly 365 days, meaning any leap days are ignored.
To calculate the AEP of a stand-alone turbine T using a windrose with n sectors and wind speed distributions $X_1, \dots X_n$, the following formula is used:

$$(24 \cdot 365) \cdot \sum_{i=1}^{n} p_i \cdot \sum_{v \in bins} T_p(v) \cdot P(X_i = v)$$

where $bins$ is the set of median speeds of each wind speed bin used.

1.2.7 Logarithmic Wind Shear

Accurate wind measurements are required to correctly calculate a turbine's AEP. However, these measurements must be at the turbine's hub height; otherwise, the wind speed may differ. Unfortunately, wind measurement stations are rarely $100m$ above ground level, so a method is needed to approximate the wind speed at a different height. The logarithmic wind shear is used to do this, abbreviated to log shear. Using the surface roughness, the log shear approximates the wind speed at the target height using the following formula:

$$v(t) = u(h) \frac{\ln \frac{t-d}{z_0}}{\ln \frac{h-d}{z_0}} \tag{1.4}$$

where $v(x)$ is the wind speed at height x, t is the target height, h is the measurement height and z_0 is the surface roughness. In this thesis, the zero plane displacement d is always assumed to be zero.

1.2.8 Downwind and Crosswind Distance

To correctly refer to relevant distances between the turbines, the downwind and crosswind distances need to be defined. As seen in Figure 3, the Downwind distance from Turbine 1 to Turbine 2 is the part of the distance in the wind direction, while the Crosswind distance is the part of the distance perpendicular to the wind direction.

1.3 Jensen Wake Model

The AEP calculation for a stand-alone turbine is relatively simple, but AEP calculations for wind farms are the goal. The problem with the AEP calculation described earlier is that the influence of the turbines upon each other is not considered. This influence is called a *wake*; the turbulence created in the wind by the wind powering the turbine. This

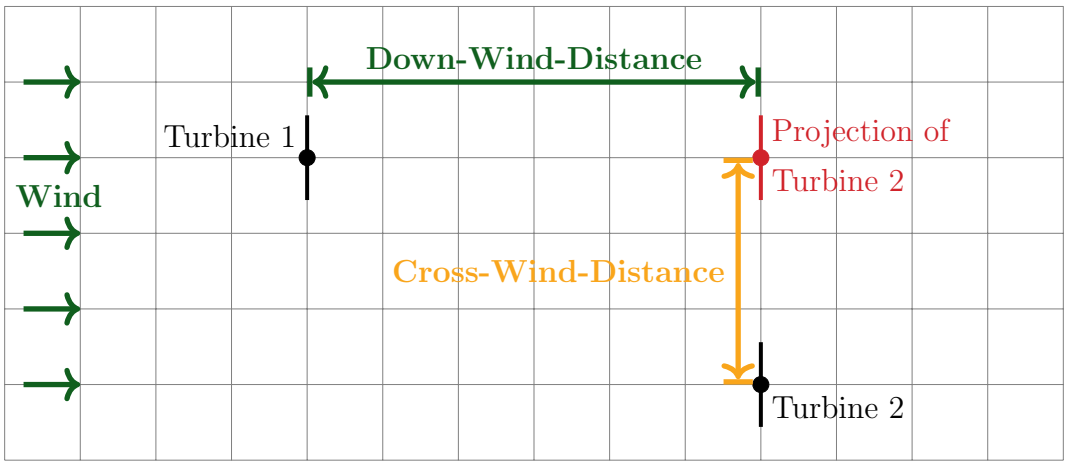


Figure 3: Diagram of a 2 turbine set-up outlining the down-wind and cross-wind distances between them

thesis uses the Jensen wake model [7], originally created by N. O. Jensen and improved by Katic et. al. [9], to calculate the wake of the turbines. To calculate the reduced wind speed u, from the free-flow wind speed u_0 with a velocity deficit δ_1, the following formula is used:

$$v = u_0 \cdot (1 - \delta_1) = u_0 - u_0 \cdot \delta_1 \tag{1.5}$$

1.3.1 Initial Velocity Loss

This leaves the question of how strong the velocity deficit is. The Jensen model provides a formula to calculate the initial velocity deficit δ_1 at the turbine causing it:

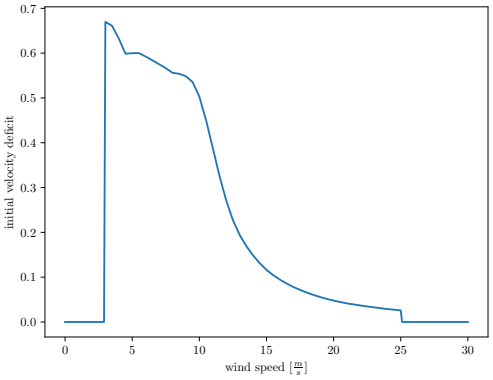
$$\delta_1 = 1 - \sqrt{1 - T_{ct}(u_0)} \tag{1.6}$$

where u_0 is the wind speed at the turbine. An example graph is given in Figure 4a

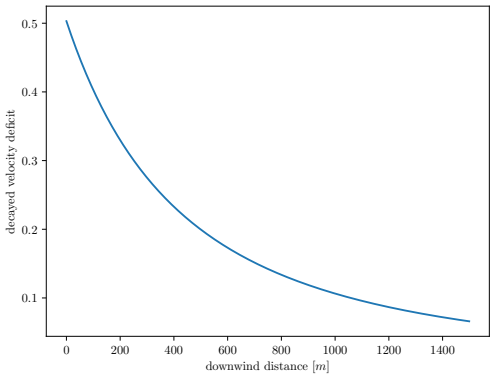
1.3.2 Wake Decay

The Jensen wake model models the wake of a turbine as a (conical) frustum created in wind direction, with the turbine's rotor area as its smaller face, as seen in Figure 5. The frustum would be cylinder-based in an environment with perfectly smooth terrain, meaning a surface roughness of $z_0 = 0m$. But in most cases, the terrain is rough; in this case, the frustum is of conical nature. Due to the momentum conservation, the velocity deficit is getting weaker as the conical wake frustum widens. This phenomenon is called the *wake decay*.
The Jensen model uses k, the wake decay factor, as a linear factor by which the wake widens. k can be measured, but in this thesis, is approximated at each turbine by:

$$k = \frac{0.5}{\ln \frac{z}{z_0}} \tag{1.7}$$



(a) Graph of initial velocity deficit.



(b) Graph of decayed velocity deficit with wind speed of $10\mathrm{m\,s^{-1}}$ and surface roughness $z_0 = 0.05m$

Figure 4: Both graphs are calculated using a Vestas V112-3.45 turbine.

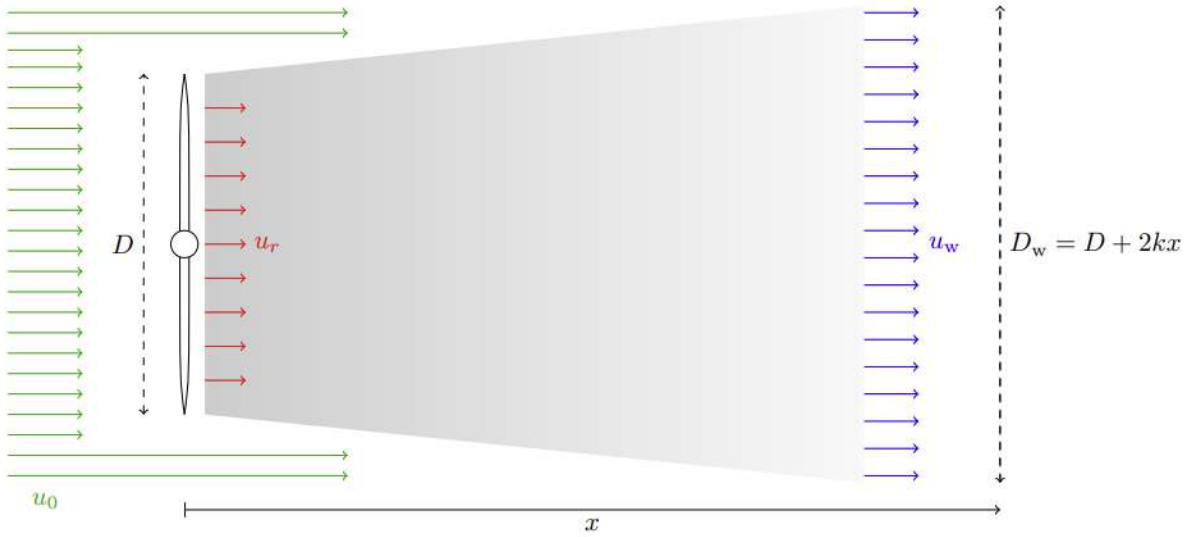


Figure 5: Top-down view of a turbine and its wake as described in the Jensen Wake model. u_0 is the free-flow wind speed, u_r the initially reduced wind speed and u_w the reduced wind speed at downwind distance x. D is the rotor diameter and D_w the wake diameter with wake decay factor k. Taken from [10].

Where z is the turbine's hub height and z_0 is the surface roughness at the site.
The Jensen model also provides a formula for calculating the wake radius $r_{wake}(x)$, x being the down-wind distance to the wake-producing turbine; it is:

$$r_{wake}(x) = r + x \cdot k \tag{1.8}$$

Where r is the rotor radius of the turbine. Using Formula 1.8 in conjunction with the energy produced by the wind, you get the formula for the wake decay at distance x, which is:

$$w_{decay}(x) = \left(\frac{r}{r_{wake}(x)}\right)^2 = \left(\frac{r}{r + x \cdot k}\right)^2 \tag{1.9}$$

In conclusion, to calculate the velocity deficit $\delta(x)$ at a down-wind distance x, the Jensen model multiplies Formaluas 1.6 and 1.9; therefore:

$$\delta(x) = \left(1 - \sqrt{1 - T_{ct}(u_0)}\right) \cdot w_{decay}(x) = \left(1 - \sqrt{1 - T_{ct}(u_0)}\right) \left(\frac{r}{r + x \cdot k}\right)^2 \tag{1.10}$$

Figure 4b displays a graph of a decayed velocity deficit dependent on the downwind distance.

1.3.3 Turbine Coverage

The *wake coverage* is the next part of the Jensen deficit model. There could be the case that a turbine is not fully, but only partially, in the wake of another turbine. In this case, the Jensen model calculates the intersection share regarding the rotor area and weakens the wake's impact accordingly. To calculate the relative area of this intersection, the following formula is used:

$$\beta = \frac{A_{int}}{A_{turb}} \tag{1.11}$$

Where A_{int} is the area of the intersection and A_{turb} is the turbine's rotor area, which is defined by the equation:

$$A_{turb} = \pi \cdot r^2 \tag{1.12}$$

Where r is the rotor radius. The area of the circle intersection is more complicated to calculate and is dependent on the cross-wind distance d between the wake-producing and receiving turbine and the wake radius r_{wake} at the down-wind distance of the two turbines; the equation is:

$$\begin{aligned} A_{int} = {} & r^2 \arccos \frac{d^2 + r^2 + r_{wake}^2}{2dr} + r_{wake}^2 \arccos \frac{d^2 + r_{wake}^2 - r^2}{2dr_{wake}} \\ & - \frac{1}{2}\sqrt{(-d + r + r_{wake})(d + r - r_{wake})(d - r + r_{wake})(d + r + r_{wake})} \end{aligned} \tag{1.13}$$

The first part of the formula calculates the area of the circle segment between the two intersection points of the "rotor-circle". The second part of the formula analogly calculates the affected circle segment of the "wake-circle". But now, the calculated area

has a lot of overlapping sections. The last part of the formula subtracts the area of the kite that spans between the circle center points and the intersection points. What is left is the area of each circle segment beyond the line between the intersection points; these two remaining areas form the lens, that is, the circle intersection area.

1.3.4 Wake Intersections

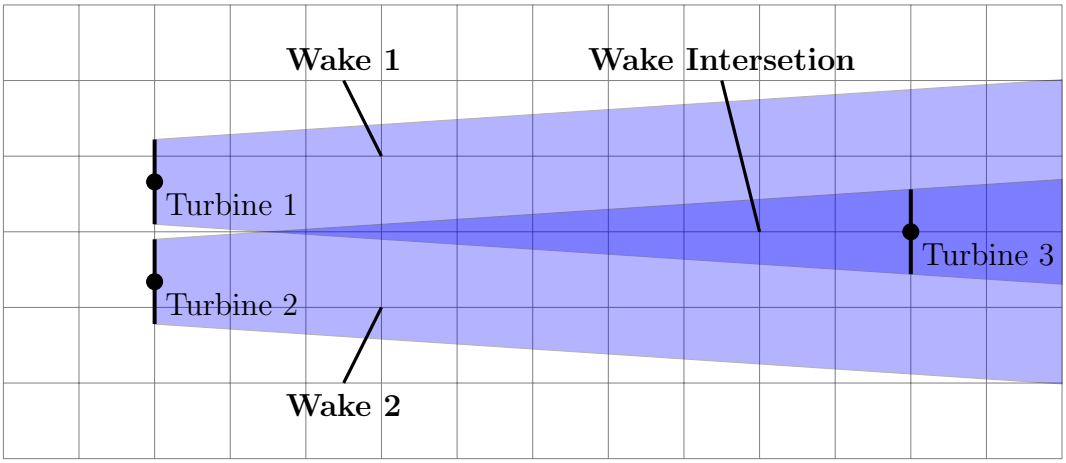


Figure 6: Sketch of Turbine 3 residing the wake intersection of turbines 1 and 2

When there are three or more turbines on a site, a turbine might be in the wake of two other turbines simultaneously. This is a *Wake Intersection*, depicted in Figure 6. Turbine 3 is located within the dark blue intersection of Wake 1 and Wake 2, where they overlap. To calculate the total velocity deficit in this intersection, the Jensen wake model uses a superposition model to resolve the different overlapping deficits. In this thesis, the squared sum is used as the superposition model. Meaning if a turbine would be affected by two velocity deficits δ_1, δ_2, then the resulting velocity deficit of the intersection is $\delta_{total} = \sqrt{\delta_1^2 + \delta_2^2}$.

1.3.5 Speed Adaption Factor

The scenario is that three wind turbines are placed in a row, and the current wind blows along that row. Now, let us calculate the wake produced by the second turbine upon the third turbine. But what wind speed value should be used for equation 1.6? Since the first turbine also projects a wake upon the second turbine, the wind speed at the second turbine is not the free-flow wind speed. It follows that the resulting velocity deficit of the second turbine is relative to the incident wind speed at the second turbine. However, since the velocity deficits are later applied to the free flow wind speed. So, in accordance with [14], a factor is added to reflect the lower incident wind at the second turbine u_2 relative to the free flow wind speed u_0, which is $\frac{u_0}{u_2}$.

1.3.6 Wake Dependencies

The scenario again involves three turbines in a row, with the wind blowing along this row. Under the directions of the Jensen Wake Model, the wake intersection of the first and second turbines should be calculated to investigate the velocity deficits acting on the third turbine. But logically speaking, the wake of the first turbine is already included in the wake of the second turbine, so it follows that this should not be a wake intersection. So only the wake of the second turbine affects the third. However, to the current knowledge, no simulation tool correctly resolves these wake dependencies.

2 WindProof's Structure & Implementation

This section will review how WindProof is structured and which assumptions were made when designing it. Please note that the classes in the UML diagrams may have been stripped of unnecessary details or were abbreviated for clarity.

2.1 Motivation

The motivation behind WindProof was to create a framework to compare wind farm simulations. However, the goal was not just to compare two tools but to create a general validation framework. Therefore, this framework should not be restricted to specific calculation types, output values, input types, or software. Following this principle, many of its classes will be abstract, allowing for the easy addition of new tools or calculation types. Figure 7 depicts the general structure of WindProof and an example workflow, which will be elaborated in the next sections.

2.2 Tools

This sector begins with the main part of WindProof, the tool class. Its basic function is displayed in Figure 8. It is an abstract class whose subclasses hold the simulation tools used for the comparisons.
In our case, a simulation tool is a simulation software performing a specific calculation. Assuming there exists a simulation software S, then S calculating the AEP and S calculating the Turbine Noise would be two different tools in WindProof. Each tool receives two inputs, a `Settings` and a `Scenario` object, and produces an output, in this thesis, the AEP. The `Settings` object contains everything that is not case specific, e.g., which interpolation methods and wake intersection models are used in the simulation. Each tool used for Simulation needs to be a subclass of the `Tool` class. The `Tool` class holds the default settings for a tool, some utility methods, and other constant data. It also asks every sub-class to implement the `_tool_calc` method, which prepares and calls the simulations software's calculations in the specific tool implementation.

▶ Implementation

The `Tool` class is abstract because the `_tool_calc` method is not implemented; this method is only implemented in the subclasses, meaning the calculation call of the actual software. Additionally, `_tool_clac` is "private" since a reference to the tool's default settings is needed to call it properly, which are themselves private. Due to the circumstances, `_tool_clac` needs to be called by another method, the `calc_scenario_single` method. Another advantage is that `calc_scenario_single` is not dependent on the simulation software; therefore, it is fully implemented. In a similar fashion,
`calc_scenario_single` is called by `calc_scenario_group`.

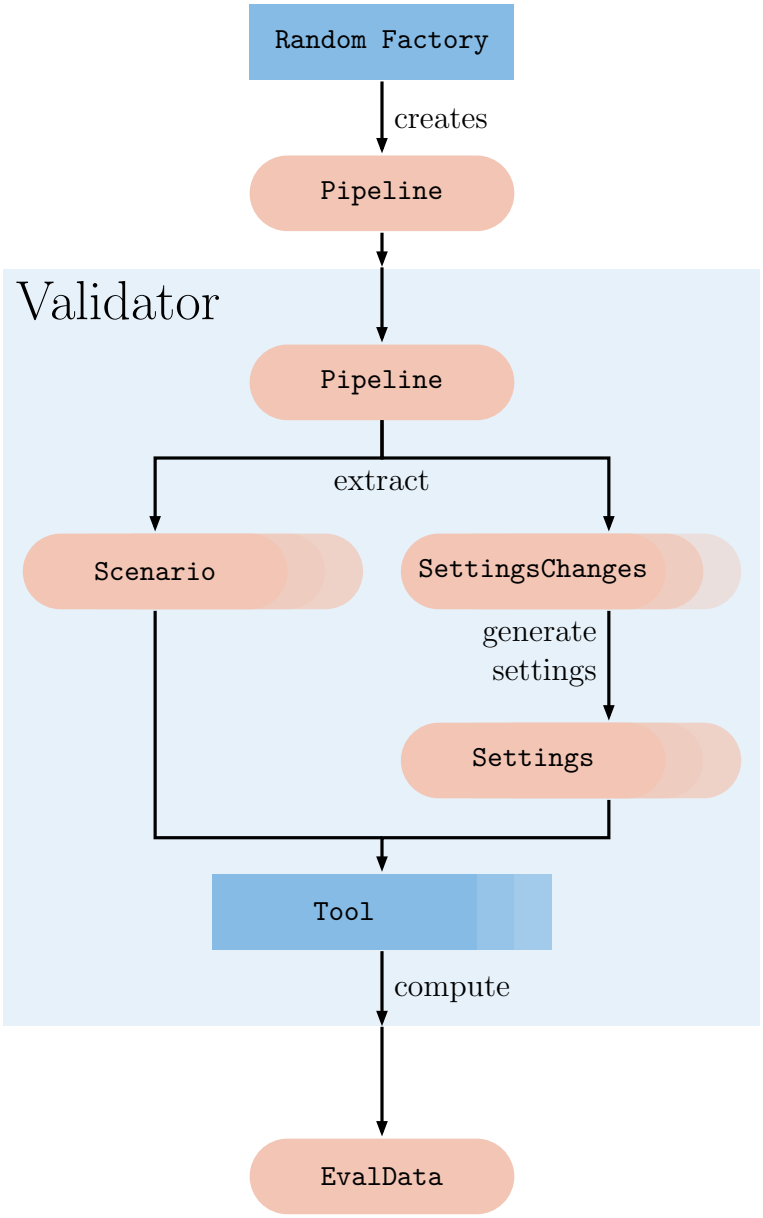


Figure 7: Overview and example workflow of WindProof. The blue boxes are classes, and the red nodes symbolize dataclasses. The workflow describes the `Random Factory` module producing a `Pipeline` instance, which is given to the `Validator` class. The `Validator` class then evokes the list of `SettingsChanges` objects to generate a list of `Settings` objects. These `Settings` objects are then together with a list of `Scenario` objects given to a list of `Tool` classes. These compute the simulation results, which are then returned from the `Validaotr` class as an `EvalData` instance.

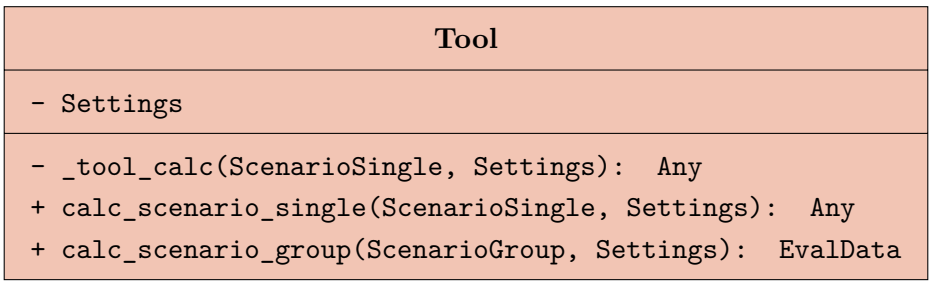


Figure 8: UML diagram of the `Tool` class. Classes colored red are abstract.

2.3 Settings

Moving on to the first input of the `Tool` class, which is the `Settings` class. The `Settings` class holds all the overarching information, which is not case-specific. The `Settings` class of WindProof, as seen in Figure 9, consists of a list of values and a list of `Range` objects. The values contain the actual settings, such as the wind bin size $bin_{size} = 1$. While the ranges define the allowed range of values for the settings, such as $0.1 \leq bin_{size} \leq 1$. If a change to a setting is out of the defined bounds, such as $bin_{size} \rightarrow 10$, the change is nullified.
These characteristics prevent unrealistic inputs, but they also have an additional purpose. They allow one to iterate over all possible values of a setting or combinations of settings, fine-tune hyperparameters, or quickly examine value changes between settings, which could contain clues for an error source.

▶ Implementation

Each setting in the `Settings` class has a name, or more specifically, a string key. Before a setting (a dictionary entry) is set to a new value, the `Settings` class checks whether it is in its `Range`. To do that quickly, WindProof requires all `Ranges` to implement the system function `__contains__(item)`, which is usually used in Collections such as Lists or Dictionaries.
WindProof provides four kinds of ranges, which are subclasses of the abstract `Range` class.

1. The `DiscRange`, or discrete range, consists of a list of all possible values.
2. The `ContRange` or contiouns range, has a start value, an end value and a stepsize, the stepsize is required when iterating over a `Range`.
3. The `AllRange` is ignored when iterating over it. It always returns `true` on a `__contains__(item)` call, and therefore, when used for a setting, allows that setting to take any value.
4. The `NoneRange` is also ignored when iterated over. It always returns `false` on a `__contains__(item)` call, and therefore, when used for a setting, allows a setting never to take any other value. Essentially, making a setting static.

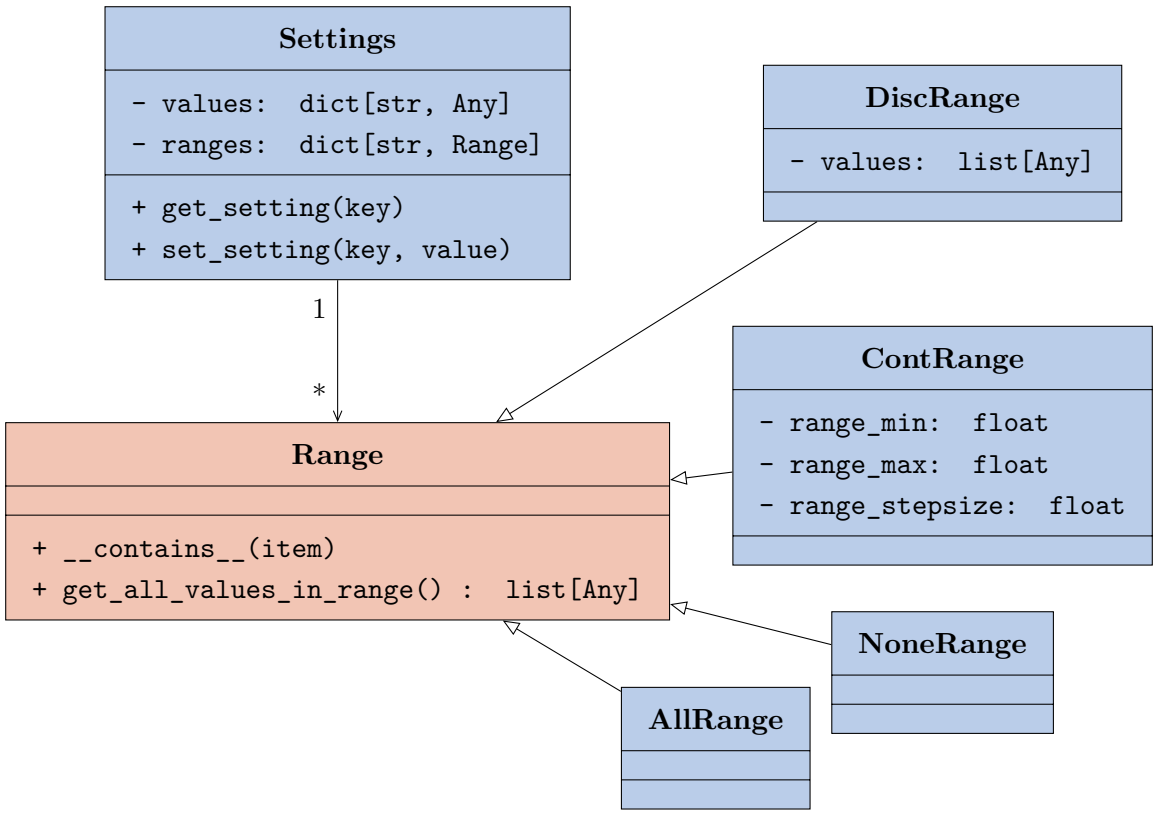


Figure 9: UML diagram of the `Settings` class and its associated `Range` objects. The subclasses of the `Range` class are also depicted. Classes colored red are abstract.

2.4 Scenarios

The `Scenario` class contains all the other information needed for a simulation. The class structure is depicted in Figure 10. The `Scenario` base class is abstract and has the basic information needed for a wind park simulation, meaning the information about the turbines, the turbines placements, and the wind data.
The turbine data, as described in section 1.2.5, is saved in a suited data class, the `Turbine` class. Each turbine's placement is saved as a point in 3D space.
How the wind data is saved depends on which subclass of the `Scenario` class is used. There are two subclasses, the `SecnarioSingle` and the `ScenarioGroup`:

▶ ScenarioSingle

The `ScenarioSingle` class is the most basic form of a scenario. Its wind data is a single windrose, as described in the section 1.2.4.

▶ ScenarioGroup

The `ScenarioGroup` class combines several scenarios using the same turbine types and placements, i.e., the same wind park setup but different wind. This is often used when one wants to simulate the same setup through several different wind speeds and directions to examine the different results. Therefore, a `ScenarioGroup` object has a list of windroses as its wind data.

▶ Implementation

The `Scenario` class itself is abstract to allow for easy extension to fit more complex calculations than the AEP calculation. In the `Scenario` class, the wind is not defined; that happens in the subclasses as described earlier. The `ScenarioGroup` class also has a class method to transform all scenarios into `ScenarioGroups` for uniform handling of the scenarios.

2.5 Pipeline & SettingsChanges

The `Pipeline` class is a way to hold the tool inputs for several simulations simultaneously and is also the preferred input for the `Validator` class, which the next section covers. As one may expect, and as depicted in Figure 7, the `Pipeline` class holds a list of scenarios. But it does not have a list of settings, but a list of `SettingsChanges`. A `SettingsChanges` is a class which, when called with a tool, returns a copy of the tool's default settings with some settings, defined in the `SettingsChanges` instance, changed to a new value. A `SettingsChanges` does not alter the tool's default settings but creates a single-use copy. This is designed this way for two reasons:

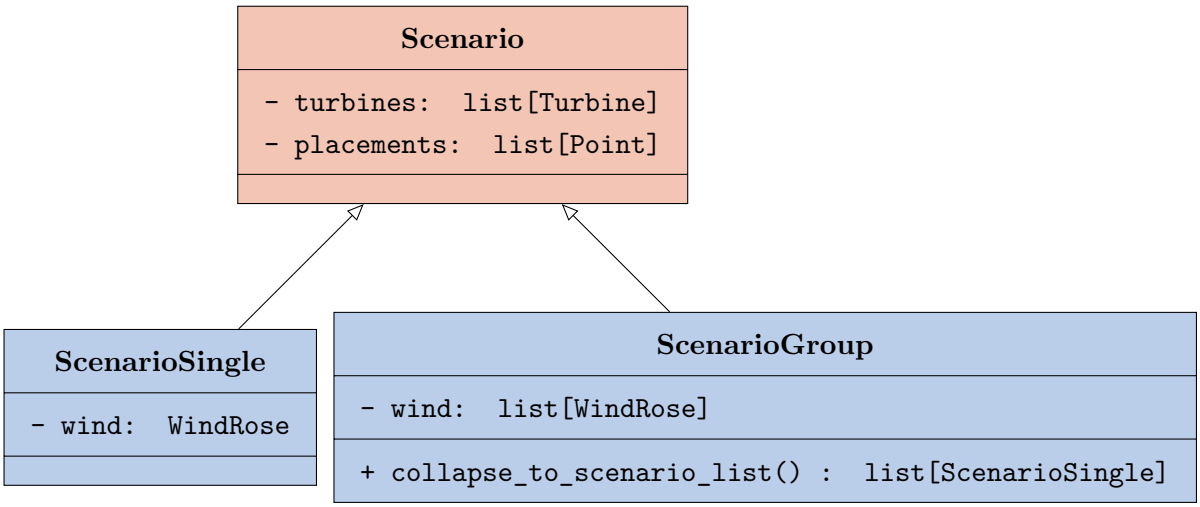


Figure 10: UML Diagram of `Scenario` class and its subclasses. Classes colored red are abstract.

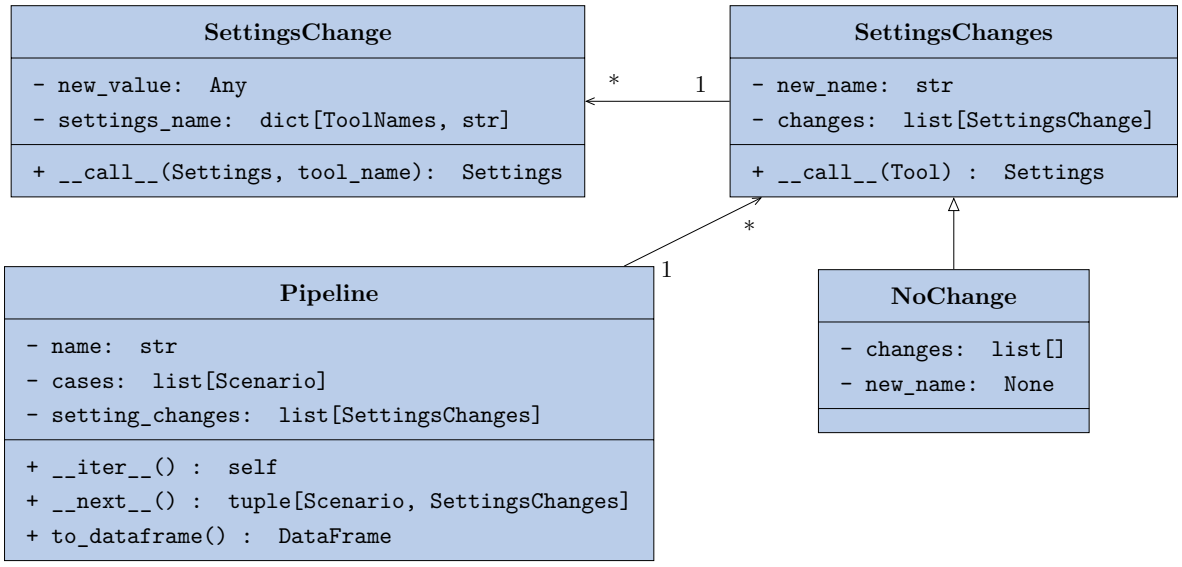


Figure 11: UML Diagram of `Pipeline`, `SettingsChanges`, `SettingsChange`, and `NoChange` classes.

1. Most of the time, only a single setting will deviate from the default. Using a `SettingsChanges` instance, the user does not have to re-enter all the default settings and must only change what must be altered.

2. The settings do not need to be reverted back to the default settings after the scenario is finished. This also improves the readability.

Pipelines can also dump themselves into a Pandas `DataFrame`. This is important for exporting random pipelines and analyzing the results of a random comparison.

▶ Implementation

To allow for easier syntax and use of the `SettingsChanges` class, the system function `__call__(Tool)` was implemented. When a `SettingsChanges` object, e.g., would be called "increase_roughness" and a `Tool` called "WindSim_AEP", the call to receive the to be used `Settings` object, would look like `increase_roughness(WindSim_AEP)`.
The `NoChange` class was implemented to beautify the process of defining no change from the default settings. The class is essentially an empty `SettingsChanges` object and has an easier constructor, being `__init__()`. To use the default settings in a `Pipeline`, just create `NoChange()`.
To make it easier and cleaner to integrate the cases in a pipeline, it has implemented the system functions `__iter__()` and `__next__()`. Therefore, to iterate over the `Scenarios` and `SettingsChanges` in a pipeline, a simple `for-in` loop is sufficient.

2.6 EvalData

The `EvalData` class is a subclass of the abstract `Data` class, as seen in Figure 12. The `Data` class itself is intended as a wrapper around a Pandas `DataFrame`.

The `EvalData` class is WindProof's way of saving and delivering data between modules and simulations. It specifies the format and column names of the DataFrame and provides additional utility functions to easily load and save data. The `EvalData` class is supposed to be used to save data across several different tools. It is also the output of the Validator class when inputting a `Pipeline` object.

▶ Implementation

The `Data` class and its subclasses are wrappers around a Pandas `DataFrame`. This was done to ease and simplify the process of data management, and the Pandas `DataFrame` is a great basis for several reasons:

- **Familiarity** The Pandas `DataFrame` is a widely used class when dealing with data in Python. It can also be used in various data science packages to analyze data further, especially from random test cases.

- **User Interface** The Pandas `DataFrame` has a powerful and user friendly access point.

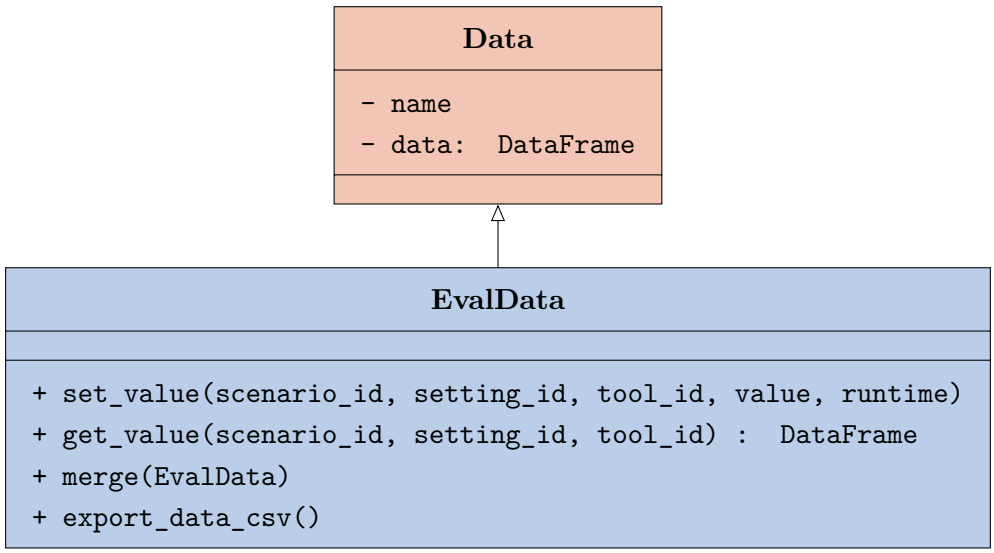


Figure 12: UML Diagram of `Data` and `EvalData` classes. Classes colored red are abstract.

- **Space efficiency** The Pandas `DataFrame` also has the advantage of being able to be exported to efficient data types, e.g., the `.feather` type.

The wrapper shortcuts the interaction process and defines static column names to make the data more uniform for workability. The names are `setting_name`, `scenario_name`, `runtime`, and the value's name, in this thesis, the AEP. The last column name of the `EvalData` class is `tool_name`; in the `ToolData` class, this is not needed since the tool's name is an attribute.

2.7 Validator

Figure 13 depicts the `Validator` class, which is the module of WindProof used to execute the simulations.
Therefore, the `Validator` class manages all the tools used in a comparison. It takes a `Pipeline` object or both a `Scenario` and a `SettingsChanges` instance and runs the tools on the data given. The results are then saved in an `EvalData` instance. In the case of calculating a whole pipeline, the `Validator` also evokes the `SettingsChanges` objects defined in the `Pipeline` instance.

▶ Implementation

The `Validator` class has a list of `Tool` objects and methods to compare the tools on `Senario` or `Pipeline` objects. In each comparison, the user can whitelist or blacklist certain tools. In the `validate` function, one `Tool` object is compared to the census, which is the average of all other tools available.

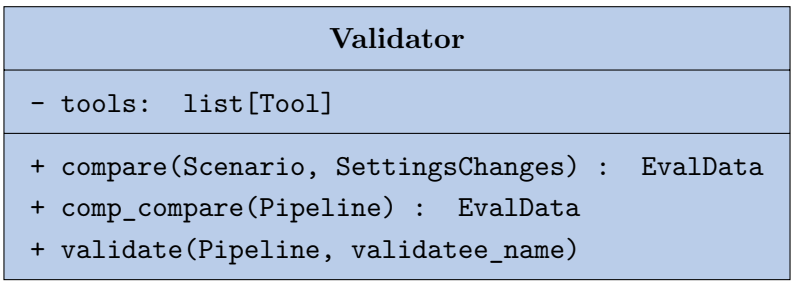


Figure 13: UML Diagram of the `Validator` class.

2.8 Random Factory

This WindProof module helps the user to examine the overall correctness of their tool. The module randomly generates test cases within realistic confines. These random pipelines can be used to examine the general error between two tools. In combination with the `Plotter` module or additional analytic resources, these random pipelines may also help to narrow down the source of an error, which can then be isolated with the help of WindProof's predefined test cases.

▶ Implementation

The random scenarios are created using the pythons `random` package. For the turbines, a random turbine type is chosen among the three "real" constant `Turbine` objects, being the Vestas V112-3.45, Enercon E115 3.000, and the Nordex N90/2500. The number of trubines is chosen by adding two random integers between 1 and 6. It is done in this way to decrease variance.
The random turbine placements are created, as illustrated in Figure 14, by generating random points and checking for sufficient space between turbines.
The random `WindRose` is defined as follows: The sector width is locked at 1°. The `measurement height` is created by adding two random integers between 5 and 50; this is done to lower variance. The `sector propabilities` are chosen in a monte-carlo-simulatio-ish manner. n points are randomly placed in a sector, then let n_i denote the number of points in the i-th sector, then the `sector probability` is $p_i = \frac{n_i}{n}$. Finally, each sector has a `wind distribution` with a random constant wind speed between 0 and 30 $\mathrm{m\,s^{-1}}$.

2.9 Plotter

This module takes `EvalData` instances to create different plots of the collected data using `seaborn`. It helps to draw conclusions out of simulated random pipelines.

▶ Implementation

The `seaborn` package was chosen due to its simple use and great synergy with the `Pandas` package

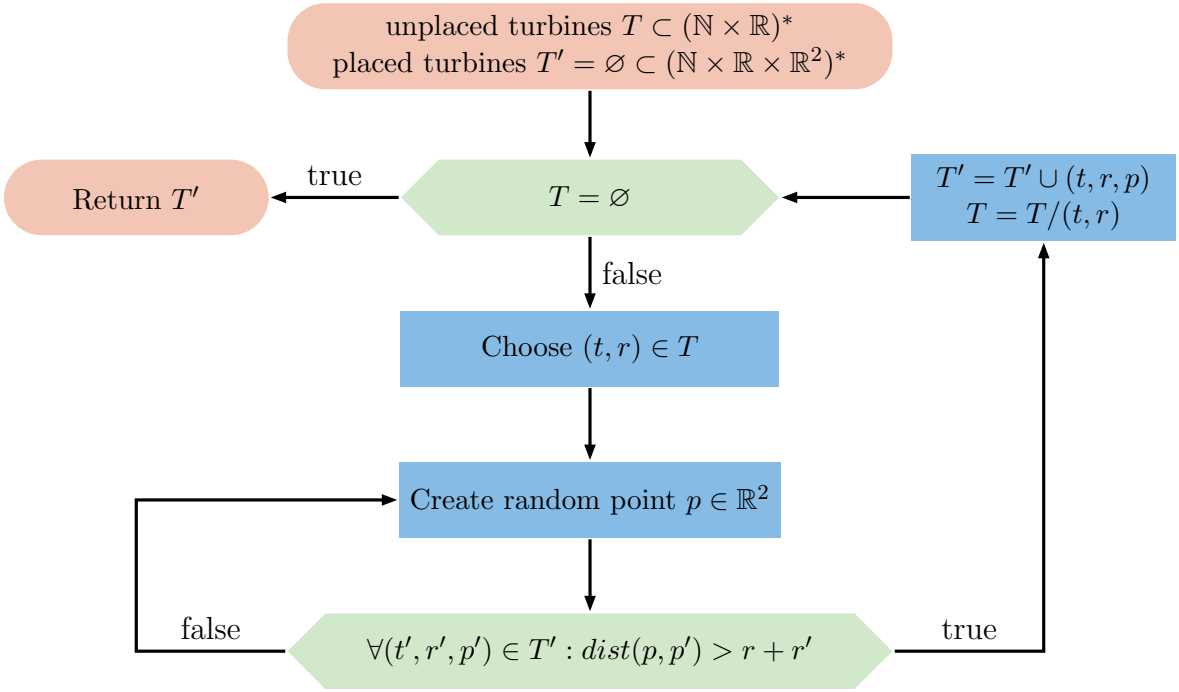


Figure 14: Flowchart of the random placement process. For each turbine, a random placement is generated; if a placement is too close to another turbine, a new placement is generated.

2.10 Constants

A tool with the WindProof framework is tested and compared with other tools in a list of test cases to validate it. WindProof provides a library of predefined test cases to ease this process. These are found in the constants module, which provides over 50 test cases, presented in detail in Section 3. WindProof also provides some predefined instances of each atomic component of a scenario to allow the user to easily build new test scenarios specific to their needs.

3 Test Scenarios

This section presents the different predefined scenarios in WindProof, how they are set up, and what errors they are trying to isolate. These scenarios are needed to validate tools with WindProof. Since WindProof works on an empirical basis, its validation processes are only as effective as its test cases. Errors that are not covered by WindProof can not be detected. Therefore, using a broad spectrum of test cases and considering as many edge cases as possible is essential. Additionally, random test cases should be used to catch any errors that the test cases may have missed.

3.1 Base Cases

These test cases use only one turbine. They test basic Annual Energy Production (AEP) calculations without any turbulence, interpolation methods, and different combinations of wind speeds. They will also serve as a control case or comparison basis for many more complex cases.

3.1.1 Vestas Single Direction Group

This test group features a single Vestas V112-3.45 turbine. The scenario group consists of a collection of windroses based on Distribution Vectors, where the wind always comes from the north, and the measurement height equals the turbine's hub height. The wind varies in speed within the group, as seen in Table 1.

Scenario Name	Wind Speed [$\mathrm{m\,s^{-1}}$]	Expected AEP [MW h]	Tests (for)
No Constant Wind	0	0	constant bias; structural errors
Too Low Constant Wind	2.9	0	cut-in definition
Low Constant Wind	4.5	1892.16	power curve interpolation
Too High Constant Wind	25.1	0	cut-out definition
Medium Constant Wind	10	23590.68	basic AEP calculation
Low & Medium Wind	4.5; 10	12741.42	multiple wind speed

Table 1: This table gives an overview of the Vestas Single Direction Group's scenarios.

▶ Too Low Constant Wind

This scenario's wind speed is just below the cut-in-speed c_{in} of the Vestas V112-3.45, with a $c_{in} = 3.0\mathrm{m\,s^{-1}}$. It checks for a proper *cut-in* of the power curve $V_p(x)$ since if the speed is interpolated from $V_p(3) = 7kW$ and $V_p(2) = 0kW$, it will be greater than 0.

3.1.2 Enercon Single Direction Group

This scenario group provides similar test cases to the Vestas Single Direction Group. In contrast to the latter, this scenario group uses a single Enercon E115 3.000. This turbine has a lower cut-in-speed $c_{in} = 2\ \mathrm{m\,s^{-1}}$, and the frequency of the data curves' anchor points is halved. An overview of the cases is displayed in Table 2.

Scenario Name	Wind Speed [$\mathrm{m\,s^{-1}}$]	Expected AEP [MW h]	Tests (for)
No Constant Wind	0	0	constant bias; structural errors
No Wind 1 Sector	0	0	correct use of sector probabilities
Constant 2 Wind	2	26.28	comparison; cut-in definition Enercon
Too Low Constant Wind	2.9	385.002	complex power curve interpolation
Low Constant Wind	4.5	2163.72	power curve interpolation
Medium Constant Wind	10	22600.8	basic AEP calculation
High Constant Wind	19.8	26280	interpretation of power curve near rated wind speed
Too High Constant Wind	25.1	0	cut-out definition

Table 2: This table gives an overview of the Enercon Single Direction Group's scenarios.

▶ No Wind Single Sector

This Scenario has only one sector with a width of 360°. It tests whether the tool can work with wind roses that do not have exactly 12 sectors.

3.1.3 Multiple Directions Group

This is the first scenario group that does not have the wind coming from only one direction. A Vestas V112-3.45 with hub height equal to the measurement height is used. This scenario group intends to isolate errors that happen during the summing of the sector-wise AEPs, interpretation of the sector probabilities, and using different wind distributions for each sector.

▶ 12 Sectors Medium Constant Winds

This windrose has 12 sectors. The sectors $1, 3, 5, 7$, and 9 all have a constant wind speed of $10\mathrm{m\,s^{-1}}$ and a sector probability of 0.2. The remaining sectors remain at $0\mathrm{m\,s^{-1}}$. This scenario splits the scenario Vestas Single Direction Group (Table 1): Medium

Wind into 5 from 12 sectors. It will detect any errors that occur while interpreting sector probabilities and summing up the resulting sector-wise AEPs.

▶ 8 Sectors Medium Winds

This scenario differentiates itself from the previous one in only one point. It has only 8 sectors in total. Four of them have a wind speed of $10\,\mathrm{m\,s^{-1}}$ and a probability of 0.25. This Scenario checks if the tool can handle different sector widths.

▶ 8 Sectors Medium Winds Inverted

Now, the sector probabilities of the previous scenario are inverted. That means that each non-zero speed has a zero sector probability. This windrose tries to catch all the tools that ignore or incorrectly use the sector probabilities but pass the previous two tests.

▶ 8 Sectors Multiple Constant Winds

This windrose also has 8 sectors; sectors 1 to 4 have a sector probability of 0.25. Sectors 1 and 5 have wind speeds of $0\,\mathrm{m\,s^{-1}}$, sectors 2 and 6 have $4.5\,\mathrm{m\,s^{-1}}$, 3 and 7 have $10\,\mathrm{m\,s^{-1}}$, and finally, sectors 4 and 8 have a speed of $19.8\,\mathrm{m\,s^{-1}}$. So, this scenario checks if a tool can handle different constant wind speeds.

3.2 Wake Model Test Cases

These cases cover all the basic wake interactions between groups of turbines. As described in the Preliminaries, WindProof evaluates its tool based on the Jensen Wake model.

3.2.1 Jensen Wake Intensity Group

This Scenario group tests the most simple Scenario where a turbine is impacted by wake. It uses 2 Vestas V112-3.45 turbines. Where the second turbine is located 300m westward of the first. This group tests the correct application of the Jensen model formulas and helps set up some more complex cases.

▶ No Interference

This is a control case for the next windrose. It uses a constant northbound wind with a wind speed of $10\,\mathrm{m\,s^{-1}}$. Since the turbines are aligned on the x-axis, the two turbines are parallel to the wind. Therefore, there should be no wake interaction between the turbines. Figure 15 also depicts this turbine setup.

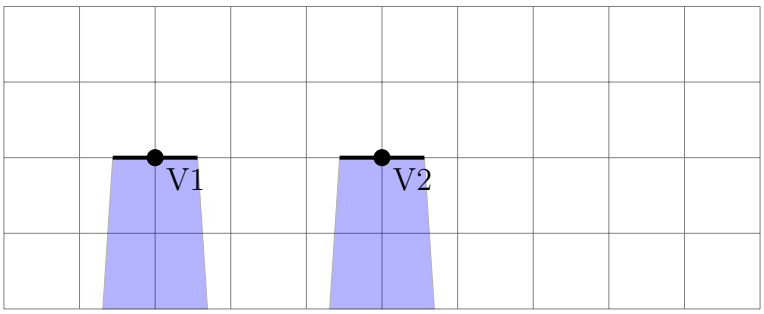


Figure 15: Jensen Wake Intensity Group turbine setup with a north wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

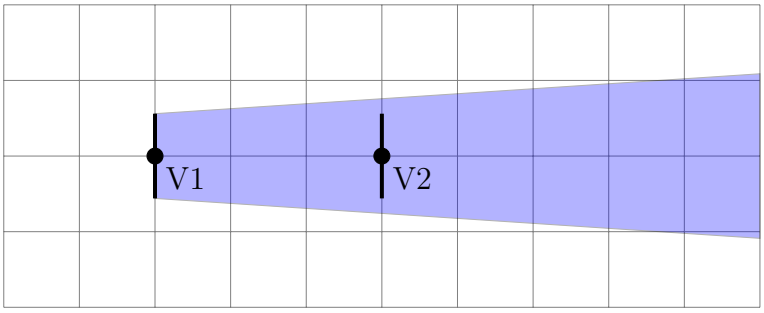


Figure 16: Jensen Wake Intensity Group turbine setup with west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

▶ Full Wake

This Scenario simulates the wake interaction between the two Vestas V112-3.45 Turbines, as seen in Figure 16. It uses a constant west wind with a speed of $10\,\mathrm{m\,s^{-1}}$ and a sector width of 1°. It follows that Turbine 2 is fully in the wake of Turbine 1. This scenario aims to isolate any mistakes when applying the basic wake deficit formulas of the Jensen wake model.

3.2.2 Multiple Turbine Types Group

This Multiple Turbine Types Group is an extension of the previous scenario group. It also explores the basic wake interactions between turbines but covers more edge cases, as displayed in Table 3. This begins with having two different turbines instead of one. The turbine setup uses one Vestas V112-3.45 and one Enercon E115 3.000. The Enercon E115 3.000 is set up 300m east from the Vestas V112-3.45.

▶ Basic Wake

This scenario aims to test the basic wake interaction for two different turbine types. The

Scenario Name	Wind Speed [$m\,s^{-1}$]	Wind Direction	Purpose
No Interference	10	North (0°)	control
Basic Wake	10	West (270°)	c. basic wake calculation
36 Sector Wake	10	West (270°)	c. Sector/Wind Direction Interpretation
Wake 19 Degree	10	complex s. below	c. Sector/Wind Direction Interpretation
No Wake	2	West (270°)	c. cut-in of ct-curve
No Wake Interpolation	2.9	West (270°)	c. interpolation of ct-curve

Table 3: This table gives an overview of the Multiple Turbine Types Group's scenarios.

windrose defines a constant west wind with a speed of $10 m\,s^{-1}$ and a sector width of 1°. The result is expected to differ from scenario Jensen Wake Intensity Group: Full Wake since the power and ct curves of the two turbine types differ.

▶ 36 Sector Wake

The only difference to scenario Multiple Turbine Types Group: Basic Wake is that the sector width is 10°, instead of 1°.

▶ Wake 19 Degree

The Wake 19 Degree windrose has a special sector probability distribution. The sector probability of sector 270 is 0.1. The neighboring sectors' probability decreases by 0.01 in each direction until it hits 0.

▶ No Wake

This Scenario is defined by a constant west wind with a constant wind speed of $2 m\,s^{-1}$. This is special because of the different cut-in-speeds of the two turbines. The Vestas V112-3.45 will stand still at this speed while the Enercon E115 3.000 spins.

3.2.3 Full Coverage

The Full Coverage scenario, also displayed in Figure 17, utilizes two Vestas V112-3.45 turbines, which are placed $500m$ apart on the x-axis, a west-bound constant wind with a speed of $10 m\,s^{-1}$ and 1° sector width. The AEP is expected to be slightly higher in comparison to the scenario Jensen Wake Intensity Group: Full Wake since the turbines are an extra $200m$ further apart.

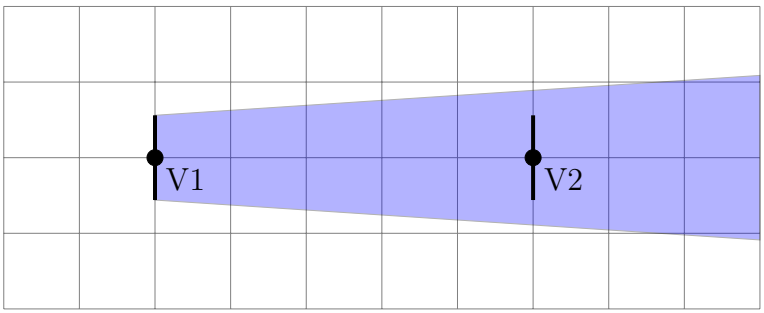


Figure 17: Full Coverage turbine setup with west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

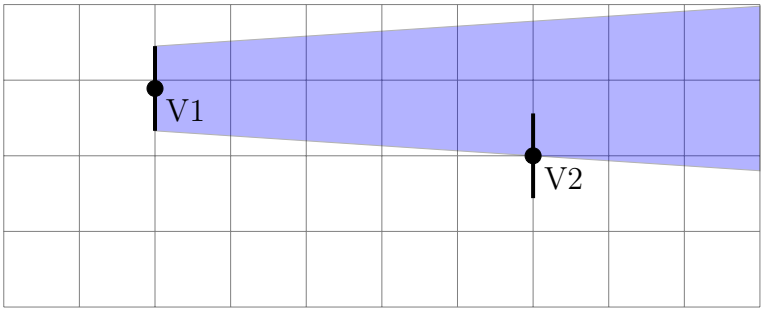


Figure 18: Partial Coverage turbine setup with west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

3.2.4 Partial Coverage

In combination with the previous scenario, this checks if the correct rotor blade coverage is calculated and applied to the wake deficit. The same windrose was used to test this effect. As Figure 18 depicts, Turbine 1 is placed at $500m$ west of the second turbine and has been shifted to $89m$ north. The intent of this placement is that with the wind used, Turbine 1's wake should hit the hub of Turbine 2. In this simulation, the effect of the wake is expected to be marginally stronger than half the wake strength of Full Coverage test case.

3.2.5 Basic Wake Intersection

The intent of the current scenario is to test the correct calculation of wake intersections. As seen in Figure 19, three Vestas V112-3.45 turbines arranged in a triangle are used to validate this interaction.
The windrose is defined by a constant westbound wind with a speed of $10\mathrm{m\,s^{-1}}$.
Note that the velocity deficits of Turibine 1 and 2 onto Turbine 3 are equal. For the resulting wake, it holds that $\delta_{total} = \sqrt{\delta_1^2 + \delta_2^2} = \sqrt{\delta_1^2 + \delta_1^2} = \sqrt{2\delta_1^2} = \sqrt{2}\sqrt{\delta_1^2} = \sqrt{2}\delta_1$. The wake influence is expected to be $\sqrt{2}$ times stronger than the wake influencing the

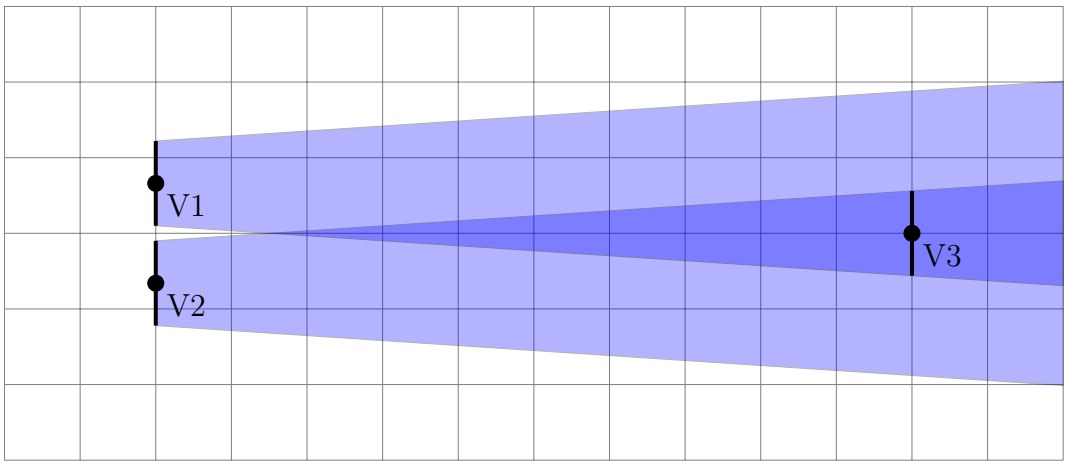


Figure 19: Basic Wake Intersection turbine setup with a west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

Turbine 2 in scenario Jensen Wake Intensity Group: Full Wake.

3.2.6 Complex Wake Intersection

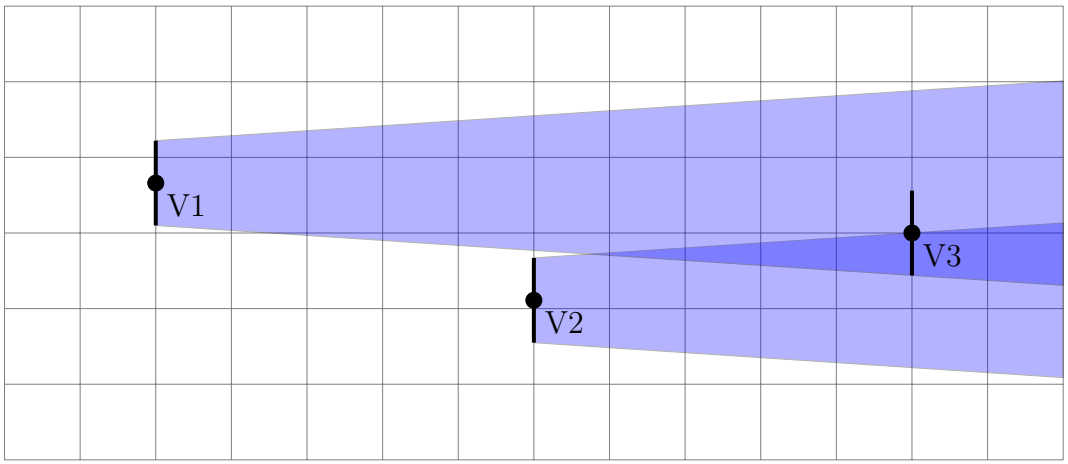


Figure 20: Complex Wake Intersection turbine setup with a west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

The Complex Wake Intersection scenario is similar to the scenario Basic Wake Intersection. It also uses 3 Vestas V112-3.45 turbines, arranged as seen in Figure 20. The wind is a constant westbound wind with a speed of 10 $\mathrm{m\,s^{-1}}$.
In conclusion, this Scenario combines three concepts of the AEP calculation using the Jensen wake model. It checks wake decay by having the wake-producing turbines at

different distances, wake coverage by having different lateral shifts in the turbine placement, and wake intersection by having two turbines, both influencing the third.

3.2.7 Three in a Row Group

This group tests for the resolution of wake dependencies. The three Vestas V112-3.45 turbines used in this scenario group are placed along the x-axis with a distance of $300m$ between them. Table 4 displays additional information.

scenario name	wind speed $[\mathrm{m\,s^{-1}}]$	wind direction	purpose
No Interference	4	North(0°)	control
Turbine 2 Inoperable	4	West(270°)	c. Wake Decay
Turbine 3 Inoperable	4.5	West(270°)	c. Wake Dependency below c_{in}
All Turbines Operable	10	West(270°)	c. Wake Dependency

Table 4: Overview of the Three in a Row Group's scenarios

▶ No Interference

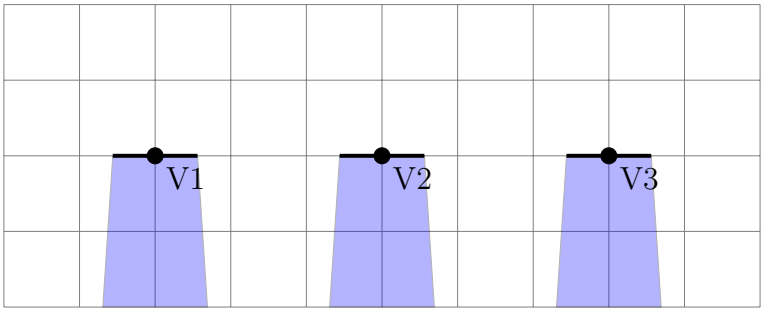


Figure 21: Three in a row turbine setup with north wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

The wind speed of this windrose is specially calibrated for this scenario, which is important for the next windrose. This rose serves as a comparison basis since, due to the wind coming from the north, the wind turbines will not interfere with each other.

▶ Turbine 2 Inoperable

At the used wind speed, the west-most turbine is creating enough turbulence to keep the wind speed below $3\mathrm{m\,s^{-1}}$ at a distance of $300m$. Since the cut-in-speed of the

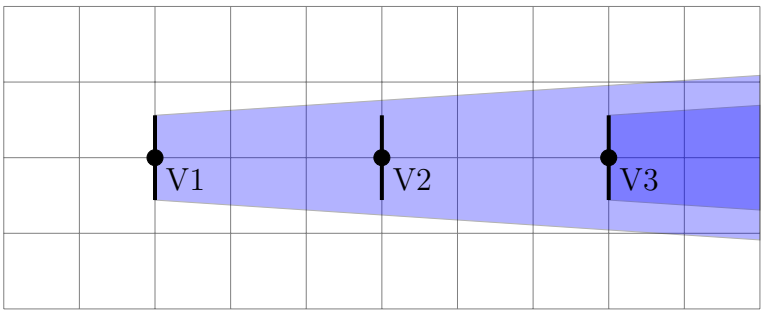


Figure 22: Three in a row turbine setup with slow west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

Vestas V112-3.45 is $c_{in} = 3\mathrm{m\,s^{-1}}$, it follows that the second turbine is not operating. But the wake from the first turbine is weak enough that at a distance of $600m$, the wind speed has again risen above the $3\mathrm{m\,s^{-1}}$ mark due to the wake decay. Note that this windrose has a sector width of $1°$.

▶ Turbine 3 Inoperable

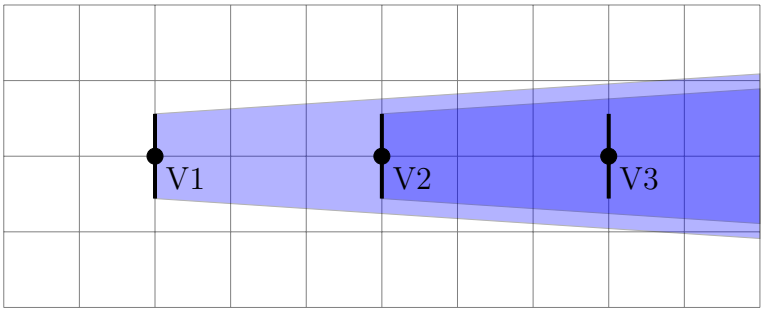


Figure 23: Three in a row turbine setup with medium west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

The increase in wind speed changes the interaction of the turbines. This time, the wake turbulence reduces the speed at the second turbine to barely above $3\ \mathrm{m\,s^{-1}}$. Consequentially, the second turbine is above its cut-in-speed and, therefore, is operative. But this ensures that with the additional turbulence the second turbine adds, that the third turbine is now inoperable.

▶ All Turbines Operable

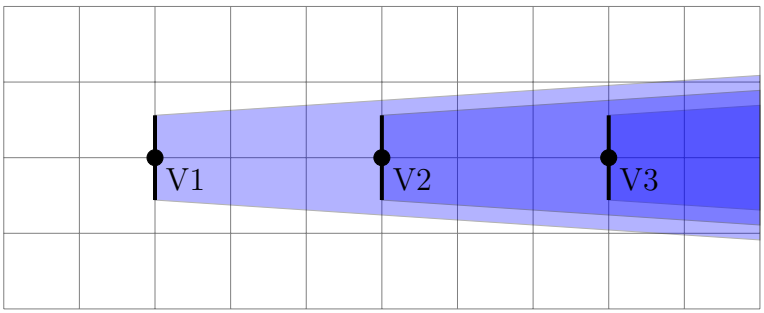


Figure 24: Three in a row turbine setup with a fast west wind. Blue trapezoids symbolize wake turbulence. The letter of a turbine denotes its type, while the number is the turbine number referenced in the text.

The second increase in wind speed also changes the scenario's behavior. The wake from the first turbine is not strong enough to bring the second turbine to a halt, and neither are the combined wakes strong enough to stop the third turbine from spinning.

3.3 Special Test Cases

The following test cases are stand-alone tests that cover specific errors that did not fit into the other sections.

3.3.1 Low Measurement Height Group

This scenario group tests the tool's ability to adjust to a lower measurement height. It uses a single Vestas V112-3.45 to achieve this.

▶ Control Wind

This scnario's windrose provides a comparison basis to check if the AEP is changing when altering the measurement height. The measurement height of this Scenario is $100m$, and the wind is a constant west wind with a speed of $10\mathrm{m\,s^{-1}}$.

▶ Low Height Wind

As the prior, this uses a constant $10\mathrm{m\,s^{-1}}$ wind speed. The measurement height is set to $10m$, which requires log shear to approximate the wind speed at the Vestas V112-3.45 's hub height, which is $100m$. With this windrose, the AEP is expected to be higher than the previous one since the starting wind speed is the same but is now measured closer to the ground. This means that the wind speed at hub height will be significantly higher.

3.3.2 Super Rough Group

This scenario group tests if all the calculations that are dependent on the surface roughness use the given value correctly. The surface roughness is relevant in two parts of the

calculations: First, the log shear, which means a low measurement height must be used again. Second is the wake decay factor. It follows that at least two turbines are required to test the correct use of the surface roughness. For that purpose, two Vestas V112-3.45 turbines are used, with one being $300m$ north of the other. Additionally, this scenario uses an unrealistically high surface roughness of $5m$.

▶ Wake Decay

This windrose uses a constant north wind with a speed of $10\mathrm{m\,s^{-1}}$. It checks if the changed surface roughness is correctly affecting the wake deficit decay. The AEP of this scenario is to be compared to the AEP of the scenario Jensen Wake Intensity Group: Full Wake since it has the same starting conditions. The AEP is expected to be slightly higher since, with a higher surface roughness, the wake decay factor is also greater, which in turn increases the wake radius at the affected turbine. This, in effect, raises the wake deficit decay, so it follows that the wind speed returns faster to the free flow speed.

▶ Log Shear

This time, a constant west wind with a speed of $10\mathrm{m\,s^{-1}}$ is used. It is chosen so as not to affect any turbine with wake. The measurement height is $10m$. The results of this simulation are to be compared to the results of scenario Low Measurement Height Group: Low Height Wind since it uses similar conditions, not including the higher surface roughness. Note that the low-measurement height group only uses one turbine, while this group uses two Vestas V112-3.45. The AEP of this simulation is expected to be much higher.

3.3.3 Raised Turbine

This scenario aims to test if a tool can handle turbines that are not on $z = 0$ but raised off the ground, e.g., through a hill. It uses a single Vestas V112-3.45 with $50m$ of elevation. The windrose consists of a west-bound constant $10\mathrm{m\,s^{-1}}$ wind. No difference is expected compared to scenario Vestas Single Direction Group (Table 1): Medium Constant Wind since both the wake decay factor as well as the log shear use the distance to the ground.

3.3.4 Terrain Wake Group

The Terrain Wake Group tests if a tool has implemented the 3D properties of a wake. It does this by using two Vestas V112-3.45 turbines. Turbine 2 is placed at $300m$ east of Turbine 1 and at $70m$ lower. That means that the wake of the Turbine 1 should only hit the top half of Turbine 2's rotor area.

▶ No Interference

This windrose is used to generate a control value. Since the wind is a constant north wind of $10\mathrm{m\,s^{-1}}$, the turbines should not interfere with each other. The AEP should be the same as in scenario Jensen Wake Intensity Group: No Interference

▶ Basic Wake

This windrose will test the wake interaction between the two turbines. The AEP is expected to be lower than in the control but not as low as in scenario Jensen Wake Intensity Group: Full Wake.

3.4 Other Test Cases

In this final section, I will review some scenarios that were created to test interactions, help create other tests, or further investigate errors.

3.4.1 Close Turbines

This was added to WindProof to further investigate the error presented and fixed in Section 4.2.4. One Nordex N90/2500 and Enercon E115 3.000 turbine are featured and placed $125m$ apart on the x-axis. The Nordex N90/2500 is placed in the West. This distance is just big enough for the turbines to not touch each other. The used wind blows with a constant speed of $4.5\mathrm{m\,s^{-1}}$ from SWW (240°), which means the Nordex N90/2500 turbine is the wake-producing turbine.

3.4.2 Nordex Control

This control case's purpose is to calculate the AEP of one Nordex N90/2500 turbine using a constant $4.5\mathrm{m\,s^{-1}}$ wind speed. The simulation result aids in a calculation in section 4.2.4.

3.5 Random Scenarios

There is one last way to test tools, which is provided by WindProof. These are the randomly generated scenarios and pipelines provided by the random factory module. It creates a scenario with between 2 and 12 turbines, with models picked randomly between the Vestas V112-3.45, Nordex N90/2500, or Enercon E115 3.000. These turbines are placed on a $2000m$ by $2000m$ square, keeping just enough distance to not destroy each other. Then, a wind rose is created, which has 360 sectors with normed sector probabilities, and for each sector, a random constant wind speed between 0 and $30\mathrm{m\,s^{-1}}$ is picked.
These random scenarios can be used to calculate the overall accuracy of a tool and try to cover all errors the test cases may have missed.
To achieve a more insightful conclusion, you can limit the randomness in several places, such as the number of turbines, or by only using single-direction winds.

4 Evaluation & Example Application

Tests must be conducted to evaluate WindProof's effectiveness in validating and comparing wind farm simulation tools. In this section, WindProof is applied to wind farm simulation tools, and then the results are evaluated.

4.1 Considered Tools

To serve that purpose in this thesis, two simulation tools are implemented into WindProof. The tools are the open-source python package `PyWake` [13] and `WindFarm3D`. WindProof implemented the `Tool`-subclasses for these simulation tools; they can be found in the WindProof modules `WindFarm3D` and `PyWake`.

4.1.1 PyWake

The first tool integrated into WindProof is `PyWake` [13], an open-source wind farm simulation tool from the DTU Wind, Technical University of Denmark. The tool is based on Python and is still actively improved. `PyWake` can simulate wind farms using several different models and calculate its AEP. The calculations are mostly implemented as matrix multiplications. This leads, with the use of multiple numerical Python libraries, to fast computation times.

4.1.2 WindFarm3D

`WindFarm3D` is the second tool implemented into WindProof in this thesis. The tool is developed by LuFG Theory of Hybrid Systems at RWTH University, with a partial student contribution. The tool's backend is programmed in Python. In addition to simulating wind farms and calculating the AEP, `WindFarm3D` also automatically fetches the site's terrain and building data. It also checks the wind turbines to ensure compliance with German government regulations.

4.1.3 WindPro

It was planned to use a third, commercially used simulation software, `WindPro`. Unfortunately, it was not possible to implement `WindPro`, in its current state, into WindProof. The problem is that `WindPro` is designed for commercial use and aims to make its processes and calculations as user-friendly as possible. To do that, `WindPro` sacrifices customizability in its calculation.
Only one type of calculation in `WindPro` meets the required level of custom inputs. Unfortunately, that calculation type has been deprecated. When doing a Jensen calculation with `WindPro` using its Python API, a calculation type must be set; since the required calculation type is deprecated, the calculation type is `ParkTypeUnknown`. However, `ParkTypeUnknown` harbors multiple deprecated algorithms; it follows that the required calculation can not be properly defined using the API. This renders WindPro unusable.

4.2 Tool Comparision & Troubleshooting

This section will evaluate and compare `PyWake` and `WindFarm3D` using WindProof. The goal is to find all the differences/errors between the two tools.

4.2.1 Base Cases

The evaluation starts with the base cases.

Scenario Name	Difference
Vestas Single Direction Group (Table 1)	
No Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Too Low Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Low Constant Wind	$3.703\,703\,703\,703\,695\,3 \times 10^{0}$
Medium Constant Wind	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Too High Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Enercon Single Direction Group (Table 2)	
No Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
No Wind 1 Sector	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Constant 2 Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Too Low Constant Wind	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Low Constant Wind	$2.842\,170\,943\,040\,401 \times 10^{-14}$
Medium Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
High Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Too High Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Multiple Directions Group	
12 Sectors Medium Constant Winds	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
8 Sectors Medium Winds	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
8 Sectors Medium Winds Inverted	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
8 Sectors Multiple Constant Winds	$0.125\,805\,944\,330\,849\,68 \times 10^{0}$

Table 5: This table shows the scenarios of the base test cases, which use `DistributionTable` objects as wind distribution and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. The marked cells show non-negligible errors.

To test the example Tools (`PyWake` and `WindFarm3D`), WindProof runs a Validator of these two tools on all of the base cases, with a constant wind speed. WindProof then calculates the relative difference as defined in Section 1.1.3 of the AEPs for each `Scenario`. Table 5 shows the results.
The value in the red cell is too high and will be investigated. The error source of

the yellow cell may be the same since the wind used in scenario Vestas Single Direction Group (Table 1): Constant Low Wind also occurs in scenario Multiple Directions Group: 8 Sectors Multiple Constant Winds, but with a lower probability. Therefore, the yellow cell is ignored until the error of the red cell is resolved.
The first observation made is that the scenario Enercon Single Direction Group (Table 2): Constant Low has a perfect score, disregarding the floating point precision. Since the only difference between the scenarios is that the Enercon Single Direction Group uses a Enercon E115 3.000 instead of a Vestas V112-3.45, it follows that the error depends on the turbine data. Comparing the two turbines yields the following differences:

- **Cut-in Speed:** Since the wind speed of $4.5\mathrm{m\,s^{-1}}$ is above the c_{in} of both turbines, which are $2\mathrm{m\,s^{-1}}$ and $3\mathrm{m\,s^{-1}}$, the c_{in} speed is irrelevant.
- **Rotor Diameter:** Because there is only 1 turbine, no wake calculation is needed. Therefore, the rotor diameter is also irrelevant.
- **Ct/Cp Curve:** For the same reason as the prior difference, the ct and cp curves have no influence over this error.
- **Power Curve:** By the process of elimination, the different power curves must be responsible. However, the turbine types are only compared to themselves, which means the actual values of the power curve have no influence over the difference. This leaves only the fact that the power curve of the Vestas V112-3.45 turbine has double the frequency of anchor points compared to the power curve of the Enercon E115 3.000, which only has anchor points at integer wind speeds.

Using the simulation results of the scenario Vestas Single Direction Group (Table 1): Low Constant Wind and calculating the result by hand, the tool causing the problem is determined. The correct AEP according to the Jensen model is defined by:

$$
\begin{aligned}
&(24[\mathrm{h/d}] \cdot 365[d]) \cdot \sum_{i=1}^{n} p_i \cdot \left(\sum_{v \in supp(X)} p(X = v) * T_p(v) \right) \\
=&8760[\mathrm{h}] \cdot \sum_{i=1}^{12} p_i \cdot \left(\sum_{v \in supp(X_i)} p(X_i = v) * V_p(v) \right) \\
\overset{p_9=1}{=}&8760[\mathrm{h}] \cdot \left(\sum_{v \in supp(X_9)} p(X_9 = v) * V_p(v) \right) \\
\overset{P(X_9=4.5)=1}{=}&8760[\mathrm{h}] \cdot V_p(4.5) \\
=&8760[\mathrm{h}] \cdot 208[\mathrm{kW}] \\
=&1822080[\mathrm{kWh}] \\
=&1822.08[\mathrm{MW\,h}]
\end{aligned}
$$

Scenario Name	Difference
Vestas Single Direction Group (Table 1)	
Low Constant Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Multiple Directions Group	
8 Sectors Multiple Constant Winds	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$

Table 7: This table shows the scenarios of the base test cases, which use `DistributionTable` objects as wind distribution and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. These results are calculated with `PyWake`'s restricted access to power curve data. Only values that differ from Table 5 are shown.

It follows that `WindFarm3D` is faulty in this case. Plugging the wrong AEP in the equation gives the wrong power curve value that `WindFarm3D` uses, denoted as $V_p^*(v)$:

$$1892.16[\text{MW h}] = 8760[\text{h}] \cdot V_p^*(4.5) \Rightarrow V_p^*(4.5) = 216[\text{kW}] \tag{4.1}$$

What becomes apparent when investigating $V_p^*(4.5)$ is that:

$$V_p^*(4.5) = \frac{V_p(4) + V_p(5)}{2} = \frac{123[\text{kW}] + 309[\text{kW}]}{2} = 216[\text{kW}] \tag{4.2}$$

It can be deduced that `WindFarm3D` interpolates at a point where it should not. Presumably, `WindFarm3D` only uses the integer anchor points of the Vestas V112-3.45 power curve. To prove this assumption, `PyWake`'s access to the turbine power curves is restricted; now, only the integer anchor point is given to `PyWake`. The simulation of the base cases now yields the results in Table 7.

Tool Name	AEP [MW h]
WindFarm3D	1892.16
PyWake	1822.08

Table 6: Simulation results of scenario Vestas Single Direction Group (Table 1): Low Constant Wind

The red cell now has a perfect score, which means the earlier assumption is correct. The yellow cell now also has a perfect score, meaning that its error originated from the same source.
No longer does any test case of the base cases show a significant error. This should lead to the assumption that both tools perform correctly on the 1-turbine cases. The tools will be simulated on 5000 random single turbine scenarios to validate that. The diff values are then calculated and plotted in a boxplot, which is Figure 25. Even the maximum outlier is less than $2.6 \cdot 10^{-13}$, meaning all errors are negligible. It follows, with very high certainty, that the tools operate correctly on any single turbine scenario. The restriction of `PyWake`'s power curve access is now permanent.

4.2.2 Wake Decay Factor

The 2-turbine cases' differences are listed in Table 8.

Scenario Name	Difference
Jensen Wake Intensity Group	
No Interference	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Full Wake	$7.259\,577\,323\,929\,676 \times 10^{0}$
Multiple Turbine Types Group (Table 3)	
No Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
No Interference	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Basic Wake	$8.023\,289\,129\,267\,496 \times 10^{0}$
36 Sector Wake	$8.150\,410\,878\,579\,066 \times 10^{0}$
Wake 19 Degree	$7.190\,617\,965\,615\,274 \times 10^{0}$
No Wake	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
No Wind Interpolation	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Full Coverage	$8.046\,184\,311\,817\,555 \times 10^{0}$
Partial Coverage	$0.428\,264\,702\,225\,789\,05 \times 10^{0}$

Table 8: This table shows the scenarios of the deficit test cases, which use 2 turbines and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. The green cells contain the scenarios with negligible or no errors.

Note that all acceptable values, those in the green cells, are from scenarios where the winds are too weak to spin both turbines or where the winds come from the north, and therefore, the turbines have no impact on each other. It is concluded that there is a grave systematic error in calculating the wake strength. Comparing the scenario Jensen Wake Intensity Group: Full Wake with scenario Multiple Turbine Types Group: Basic Wake, which only differ in the distances of the turbines in untested properties, yields the insight that the error intensifies as the distance lowers. This suggests that the error source must be the wake decay factor.

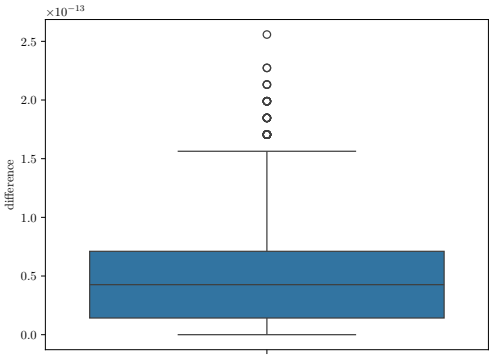


Figure 25: Boxplot of diff values from 1 turbine random pipeline. Simulated with `WindFarm3D` and `PyWake`

Investigating this manner reveals that `PyWake` does not approximate its wake decay factor but rather draws it from user input. With a default value of 0.1 this differs significantly from the value of `WindFarm3D`, which would be $k = \frac{0.5}{\ln \frac{100}{0.05}} \approx 0.066$, in the case of the Vestas V112-3.45. Approximating the k-parameter in WindProof before starting

Scenario Name	Difference	
Jensen Wake Intensity Group		
Full Wake	2.842 170 943 040 401	$\times 10^{-14}$
Multiple Turbine Types Group (Table 3)		
Basic Wake	2.842 170 943 040 401	$\times 10^{-14}$
36 Sector Wake	1.034 155 051 941 766 6	$\times 10^{0}$
Wake 19 Degree	2.256 683 728 774 078 2	$\times 10^{-11}$
Full Coverage	4.263 256 414 560 601	$\times 10^{-14}$
Partial Coverage	2.842 170 943 040 401	$\times 10^{-14}$

Table 9: This table shows the scenarios of the deficit test cases, which use 2 turbines and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. The wake decay factor of `PyWake` is now approximated by WindProof. The green cells contain scenarios with negligible or no errors. Only values that differ from Table 8 are shown.

the simulation in `PyWake` resolves this issue. But this fix creates a new issue because `PyWake`'s wake decay factor is used project-wide, while in `Windfarm3D`, the k-parameter is approximated individually for each turbine. To resolve this follow-up error, WindProof sets each turbine's hub height to $100m$. Another data set from WindProof proves this assumption, which can be investigated in Table 9.

As seen in the data, this change eliminated nearly every error in this group of cases. This leaves just the scenario Multiple Turbine Types Group (Table 3): 36 Sector Wake as the only case with a significant error.

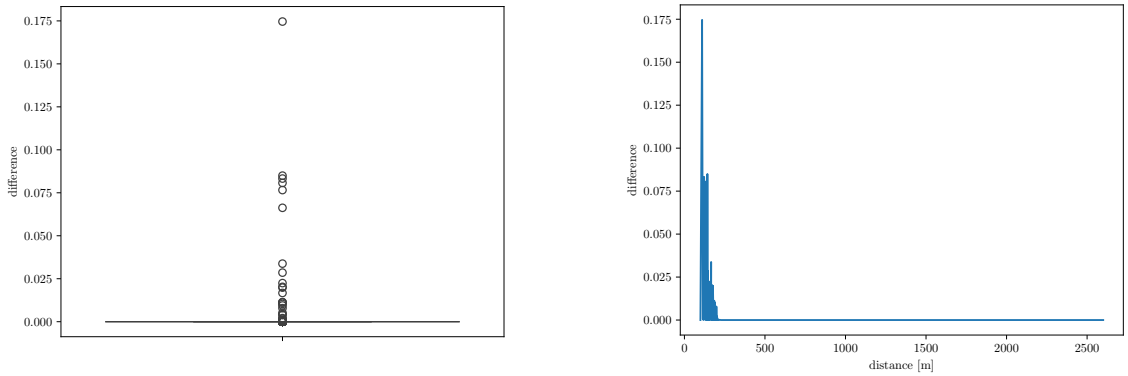
4.2.3 Sector Width

The AEP values of the cases scenario Multiple Turbine Types Group (Table 3): Basic Wake, scenario Multiple Turbine Types Group (Table 3): 36 Sector Wake, and scenario Multiple Turbine Types Group (Table 3): 19 Degree Wake are pulled and displayed in Table 10 to gain further insights into the problem.

Note that for Basic Wake and Wake 19 Degree scenarios, `PyWake` and `WindFarm3D` have identical results. The difference lies in the way 36 Sector Wake is interpreted. Since `WindFarm3D` has the same result in Basic Wake and 36 Sector Wake, it can be concluded that `WindFarm3D` always assumes that the wind is aligned with the sector middle line. WindProof interprets this the same way.
On the other hand, `PyWake` has the same results in 36 Sector Wake and Wake 19 Degree, which alludes that `PyWake` spreads the probability over the whole sector, with decreasing density towards the edges of the sector. While this is a more realistic behavior, it creates errors in these edge cases and, therefore, is suppressed from now on. To achieve this, each following scenario will have a sector width of $1°$.

Tool Name	Scenario Name	AEP [mWh]
	Multiple Turbine Types Group	
WindFarm3D	Basic Wake	$30\,920.600\,090\,478\,023 \times 10^0$
PyWake	Basic Wake	$30\,920.600\,090\,478\,034 \times 10^0$
WindFarm3D	36 Sector Wake	$30\,920.600\,090\,478\,023 \times 10^0$
PyWake	36 Sector Wake	$31\,243.708\,480\,139\,347 \times 10^0$
WindFarm3D	Wake 19 Degree	$31\,243.708\,480\,132\,293 \times 10^0$
PyWake	Wake 19 Degree	$31\,243.708\,480\,139\,343 \times 10^0$

Table 10: This table shows the AEPs of `PyWake` and `WindFarm3D` on three different scenarios.



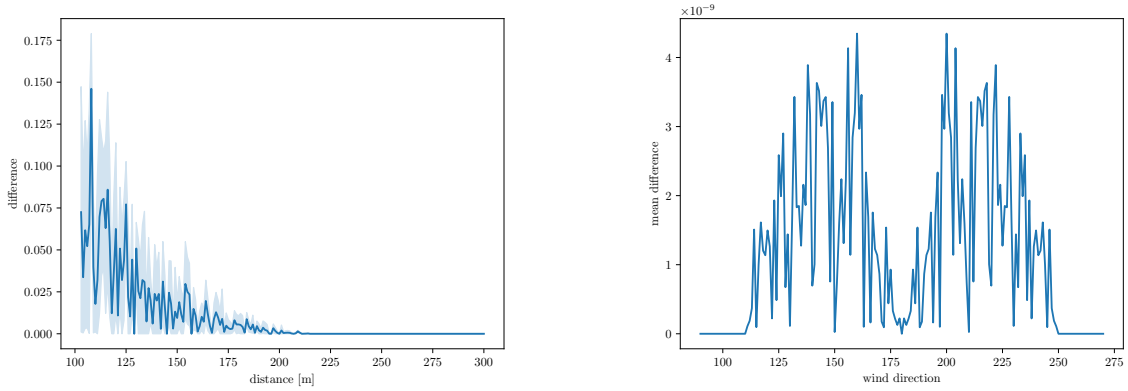
(a) Boxplot of difference values. No box nor whiskers can be seen since the median and both quantiles are zero.

(b) Graph of difference values dependent on the distance between turbines.

Figure 26: Data from the simulations results of a 2500 scenario 2 turbine random pipeline.

4.2.4 2 Turbine Cases

The test cases up to this point show no unexplained difference anymore. A pipeline of 2500 random 2-turbine scenarios is simulated to validate this result. Figure 26a displays a box plot of the differences. Note that there are significant errors. The differences are plotted depending on the distance between the two turbines in each scenario to investigate this error source. Figure 26b displays this plot. It is immediately clear that all scenarios with exceptionally high differences have a small distance between turbines. To investigate further, a function is added to WindProof's `Random Factory`. This function creates a pipeline, named "increasing distance pipeline", with random scenarios but with a fixed distance between the turbines; the distance iterates over a given interval, creating n random scenarios at each distance d. WindProof now simulates



(a) Linegraph of the difference dependent on the distance of the turbines from the scenarios of the increasing distance pipeline.

(b) Graph of mean difference values of all wind speeds dependent on wind direction angle.

Figure 27: Data from the simulations of custom pipelines using the `Random Factory` module

this custom pipeline and plots the differences dependent on d, as seen in Figure 27a.

It is apparent that the difference is anti-proportional to the distance between the turbines. Note that over a certain threshold, the difference is insignificant. The relationship between distance and difference is now known. To check the relationship between wind direction and speed, a function to create the "close turbines pipeline" is added to the `Random Factory`. In this pipeline, the distance between the two turbines is fixed at $125m$ while it iterates over each combination of angle (0 to 359) and speed (0 to 30 in 0.1 steps). But this makes the size of the pipeline problematic with $360 \cdot 301 = 108360$ different scenarios. Therefore, WindProof limits the angles to the range $[90, 270]°$ since the setup is symmetric on the x-axis and the windspeeds to the interval $[2.9, 5]\mathrm{m\,s^{-1}}$. This reduces calculations to the more manageable number of $181 \cdot 31 = 5611$ scenarios.

The simulation results, presented in Figure 27b, show that when the angle is near 90° or 270°, the error is near 0; going further away from the endpoints first increases the error, but then it decreases again when approaching 180°.

This suggests that the error source is in the turbine coverage calculation since there is no error when the coverage is either 1 or 0. This is unexpected since when taking a look at Table 9, the error in the partial coverage test case is negligible. The only difference is that the coverage test cases use the same turbine twice, while in the random pipeline, two different turbine types could be involved.

To investigate this assumption, another function is added to the `Random Factory` module. The function creates the "close turbine combinations" custom pipeline. In addition to the other custom pipeline, this pipeline also fixes the wind direction at 240° and iterates over all combinations of turbine types and all windspeeds $v \in [1.9, 5]\mathrm{m\,s^{-1}}$. The results can be seen in Figure 28.

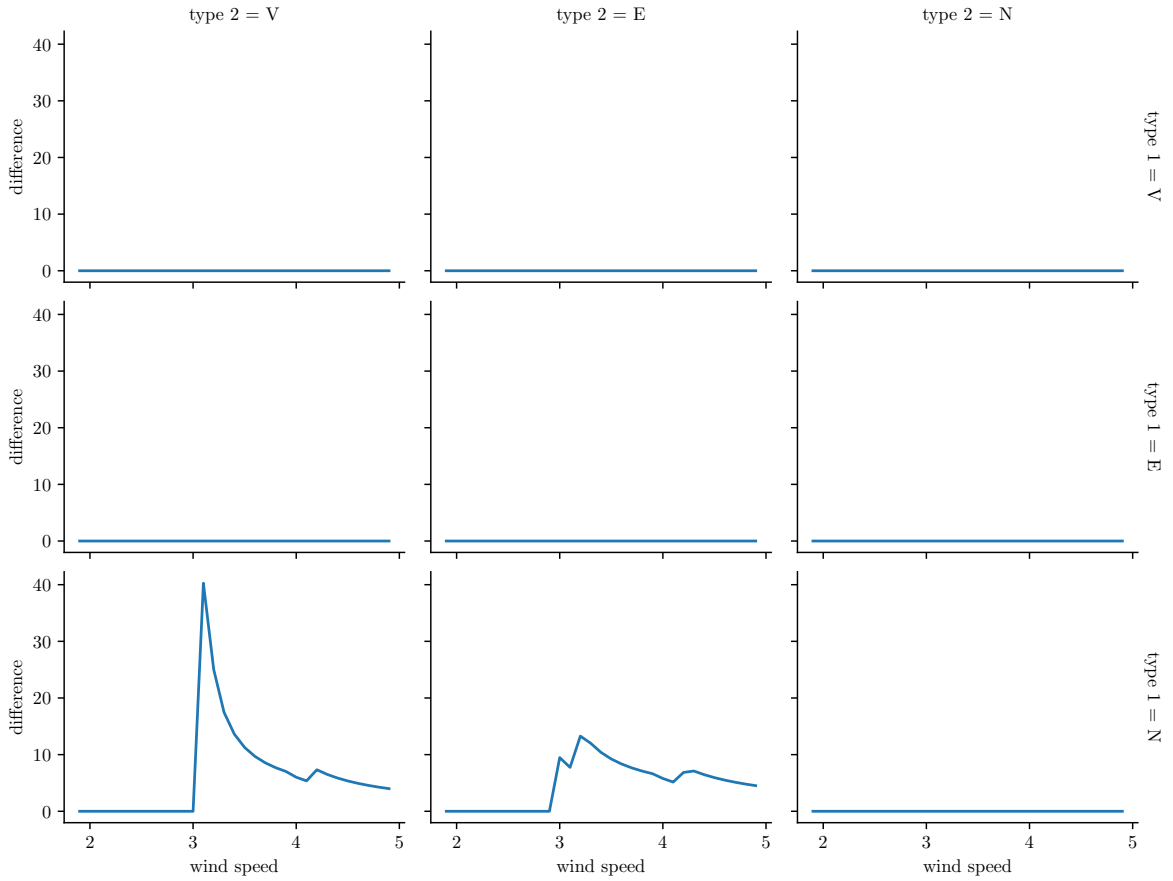


Figure 28: Line plot matrix of difference over wind speed. Type 1 is the turbine type of the wake-producing turbine, while type 2 is the turbine type of the wake-receiving turbine. The turbine types are abbreviated to V for Vestas V112-3.45, E for Enercon E115 3.000, and N for Nordex N90/2500.

These results prove the earlier assumption that the error only occurs in specific turbine combinations; more precisely, it occurs only when a Nordex N90/2500 influences another turbine type. To elaborate on this problem, a special test case is added to WindProof, scenario Close Turbines. The case will now be validated by hand.

Table 11 shows the test case results. To find the correct value, the wind speed is reverse-engineered. To do that, another special test case, scenario Nordex Control, is created; it has the same windrose but features only one Nordex N90/2500 turbine. The case has no error, but the result is used to calculate the Enercon E115 3.000's AEP in the prior case by subtracting the AEP of the Nordex N90/2500 from the result in Table 11. Then both results are divided 8760 and multiplied by 1000 to calculate the Enercon E115 3.000's wattage:

(WF) 282.71	$-81.468 = 201.242$	$201.242/8760 \cdot 1000 = 22.973$
(PW) 258.282	$-81.468 = 176.814$	$176.814/8760 \cdot 1000 = 20.185$

Since $N_{ct}(3.1) = 1$, it follows that the initial loss is 1.
This turbine setup's downwind and crosswind distances are calculated using trigonometry, resulting in a downwind distance of 108.253m and a crosswind distance of 62.5m. With these values and a wake decay factor approximation, the wake radius is determined:

Tool Name	AEP [mWh]
WindFarm3D	282.71
PyWake	258.282

Table 11: This table shows the simulation results of scenario Close Turbines using different tools

$$r_{wake}(108.253) = 45 + 108.253 \cdot \frac{0.5}{\ln \frac{100}{0.05}} = 52.121$$

Using this value, the wake decay factor and the turbine coverage are calculated:

$$w_{decay}(108.253) = \left(\frac{45}{52.121}\right)^2 = 0.74542$$
$$\beta = \frac{A_{int}}{A_{turb}} = \frac{3007.61}{10513.7} = 0.28607$$

This results in a velocity deficit of $1 \cdot 0.74542 \cdot 0.28607 = 0.2132422554$ at the Enercon E115 3.000.
When inversing the linear interpolation of the Enercon E115 3.000's power curve, the incoming wind speed at the turbine can be calculated. Which is:

$$\text{(WF) } 3 + 45.5 \cdot (v - 2) = 22.973 \Rightarrow v = 2.438967$$
$$\text{(PW) } 3 + 45.5 \cdot (v - 2) = 20.185 \Rightarrow v = 2.37767$$

When that speed is divided by the final velocity deficit, the free flow wind speed is retrieved, which is:

$$\text{(WF) } 2.438967/0.2132422554 = 3.1000023041579$$
$$\text{(PW) } 2.37767/0.2132422554 = 3.0221121422602324$$

This validation reveals that, in this case, `WindFarm3D` has the correct result. To investigate why the circle intersection function of `PyWake` calculates a wrong result, the methods that calculate the circle intersection area are extracted from both programs and are executed with several inputs.

The data in Table 12 suggests that `PyWake` calculates the wrong area if the radius of the receiving turbine is greater than the wake radius at a distance. To confirm this, the wake radii for all cells in Figure 28 are calculated as seen in Table 13.

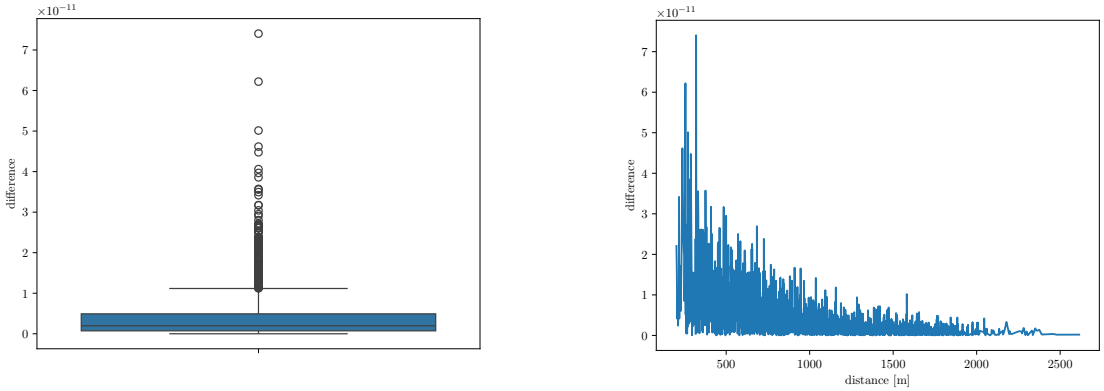
When comparing these radii with the graphs in 28, it becomes apparent that the assumption for the error's trigger was right. In this thesis, the minimal distance of turbines in the random scenarios is lifted to this no-error threshold, which is now calculated to stop the error from occurring. The threshold should be the distance d where the wake

Tool Name	Wake Radius	Receiving Turbine Radius	Area $[m^2]$
WindFarm3D	52.121	57.85	3007.6114
PyWake	52.121	57.85	3286.4581
WindFarm3D	52.121	45	1779.9863
PyWake	52.121	45	1779.9863
WindFarm3D	52.121	53	2519.5157
PyWake	52.121	53	2563.3179

Table 12: This table shows the circle intersection area calculated by `WindFarm3D` and `PyWake` with different parameters. Distance between circles is fixed at $62.5m$.

from \ to	Vestas	Enercon	Nordex
Vestas	(63.121; 56)	(63.121; 57.85)	(63.121; 45)
Enercon	(64.971; 56)	(64.971; 57.85)	(64.971; 45)
Nordex	(52.121; 56)	(52.121; 57.85)	(52.121; 45)

Table 13: This table shows the radii of each case in Figure 28. The radii are presented in the format (r_1, r_2), where r_1 is the wake radius, and r_2 is the rotor radius of the receiving turbine. In the red cells, $r_1 < r_2$ holds.



(a) Boxplot of difference values.

(b) Graph of difference values dependent on the distance between turbines.

Figure 29: Data from the simulations results of a 2500 scenario 2 turbine random pipeline, with updated parameters.

radius of a Nordex N90/2500 equals the rotor radius of an Enercon E115 3.000, and it is calculated by the following:

$$
\begin{aligned}
& N_r + \frac{0.5}{\ln \frac{100}{0.05}} \cdot d = E_r \\
\Rightarrow & \frac{0.5}{\ln \frac{100}{0.05}} \cdot d = E_r - N_r \\
\Rightarrow & d = \frac{E_r - N_r}{\frac{0.5}{\ln \frac{100}{0.05}}} \\
\Rightarrow & d \approx 195.343
\end{aligned}
$$

To account for calculation inaccuracies and the downwind distance being a bit higher than the turbine distance with some angles and regarding Figure 27a, the threshold is set to $200m$. Additionally, the rotor radius of the Nordex N90/2500is raised to $50m$. Another 2500 random scenario pipeline is run to confirm the effectiveness. Figure 29 shows the results. Considering these and that the maximum difference with $\approx 7.5 \cdot 10^{-11} < 10^{-10}$ is negligible, the 2 turbine cases are considered fixed.

4.2.5 Speed Adaption Factor

With all the errors of the 2-turbine cases identified, only the 3-turbine scenarios of the deficit cases remain. Table 14 depicts their difference values:

The only test case with a relevant error is scenario Three in a Row Group: All Turbines Operable. Since the rest of the scenarios of the Three in a Row Group have no relevant differences, it can be assumed that the error source is the resolution of the wake dependency found in the faulty scenario. Because scenario Jensen Wake Intensity

Scenario Name	Difference
Basic Wake Intersection	$2.842\,170\,943\,040\,401 \times 10^{-14}$
Complex Wake Intersection	$2.842\,170\,943\,040\,401 \times 10^{-14}$
Three in a Row Group (Table 4)	
No Interference	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Turbine 2 Inoperable	$4.263\,256\,414\,560\,601 \times 10^{-14}$
Turbine 3 Inoperable	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
All Turbines Operable	$7.180\,730\,119\,283\,453 \times 10^{0}$

Table 14: This table shows the scenarios of the deficit test cases, which use 3 turbines and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. The green cells contain scenarios with negligible or no errors.

Tool Name	AEP [mWh]
WindFarm3D	36 270.363 04
PyWake	39 076.328 74

Table 15: This table shows the simulation results of scenario Three in a Row Group: All Turbines Operable using different tools. Values rounded to 10^{-5}.

Group: Full Wake shows no relevant error, the problem must lie with the third turbine. Table 15 the concrete AEPs of scenario Three in a Row Group: All Turbines Operable so that the relevant values of the third turbine can be extracted by hand.

Subtracting the AEP of scenario Jensen Wake Intensity Group: Full Wake gives the AEP of just the third turbine. Dividing that by 8.760 results in the third turbine's power output in [kW]:

$$\begin{aligned} &\text{(WF) } 36270.36304 - 32748.42776 = 3521.93528 && 3521.93528/8.760 = 402.04741 \\ &\text{(PW) } 39076.32874 - 32748.42776 = 6327.90098 && 6327.90098/8.760 = 722.36313 \end{aligned}$$

By reversing the interpolation of the power curve, the speed at the turbine is calculated:

$$\begin{aligned} &\text{(WF) } 309 + (567 - 309) \cdot (v - 5) = 402.04741 \Rightarrow v = 5.36065 \\ &\text{(PW) } 567 + (927 - 567) \cdot (v - 6) = 722.36313 \Rightarrow v = 6.43156 \end{aligned}$$

The resulting velocity deficits for the third turbine are:

$$\begin{aligned}\text{(WF)}\ & 10 \cdot (1-\delta) = 5.36065 \Rightarrow \delta = 0.463935 \\ \text{(PW)}\ & 10 \cdot (1-\delta) = 6.43156 \Rightarrow \delta = 0.356844\end{aligned}$$

By using the formulas of section 1.3.2, the wake deficit from turbine 1 onto turbine 3 with $k \approx 0.0657$ is calculated:

$$\left(1 - \sqrt{1 - V_{ct}(10)}\right) \left(\frac{56}{56 + 600 \cdot k}\right)^2 = 0.173072$$

This velocity deficit is now used in combination with the squared sum to extract the wake influence from turbine 2 onto turbine 3:

$$\begin{aligned}\text{(WF)}\ & \sqrt{0.173072^2 + \delta^2} = 0.463935 \Rightarrow \delta = 0.430444 \\ \text{(PW)}\ & \sqrt{0.173072^2 + \delta^2} = 0.356844 \Rightarrow \delta = 0.312064\end{aligned}$$

To confirm which result is correct, the wake influence from turbine 2 onto turbine 3 is calculated using $k \approx 0.0657$ and the wind speed at turbine 2, $\approx 7.26284\text{m}\,\text{s}^{-1}$:

$$\left(1 - \sqrt{1 - V_{ct}(7.26284)}\right) \left(\frac{56}{56 + 300 \cdot k}\right)^2 \cdot \frac{10}{7.26284} \approx 0.430444$$

It follows that `WindFarm3D` delivers the correct result, investigating the factor multiplied with the velocity deficit by `WindFarm3D` yields:

$$\frac{0.430444}{0.312064} \approx 1.376872 \approx \frac{10}{7.26284}$$

This reveals that `PyWake` is missing the speed adaption factor. To confirm this assumption the factor is disabled from `WindFarm3D`'s calculation. To validate that this was the error source, WindProof recalculates the results. Table 16 displays the difference values.

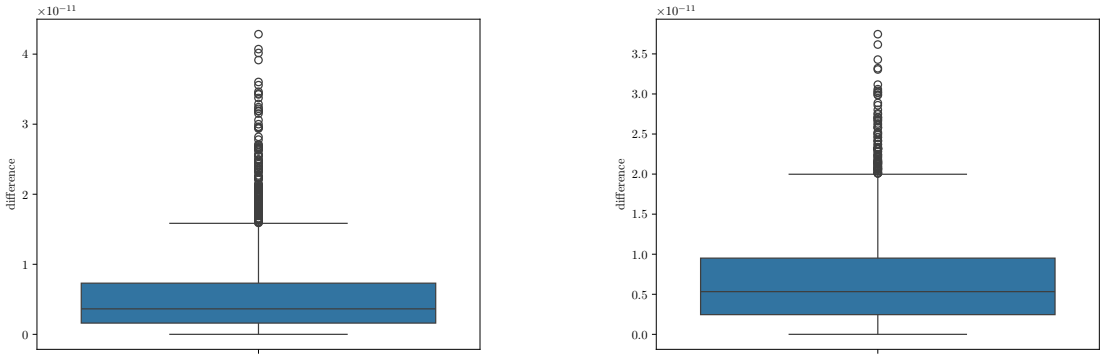
Now that all 3-turbine test cases are resolved, WindProof runs a pipeline containing 2500 random 3-turbine scenarios. As seen in Figure 30a, the results show no significant errors.

4.2.6 Special Test Cases

Table 17 displays the Special Test Cases simulation results. The only test case with a significant error is scenario Terrain Wake Group: Basic Wake. The concrete AEP values are displayed in Table 18 to investigate the error further.

Scenario Name	Difference
Three in a Row Group (Table 4)	
All Turbines Operable	$1.421\,085\,471\,520\,200\,4\times10^{-14}$

Table 16: This table shows the scenarios of the deficit test cases, which use 3 turbines and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. For this calculation, `WindFarm3D`'s velocity deficits are also calculated respective to the free-flow windspeed. The green cells contain scenarios with negligible or no errors. Only values that differ from Table 14 are shown.



(a) Boxplot of difference values.The number of turbines is fixed at three.

(b) Box plot of difference values. The number of Turbines varies between 2 and 10.

Figure 30: Data from 2500 random scenarios simulated by `WindFarm3D` and `PyWake`

Scenario Name	Difference
Low Measurement Height Group	
Control Wind	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Low Height Wind	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Super Rough Group	
Wake Decay	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Log Shear	$0.000\,000\,000\,000\,000\,00 \times 10^{0}$
Raised Turbine	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Terrain Wake Group	
No Interference	$1.421\,085\,471\,520\,200\,4 \times 10^{-14}$
Basic Wake	$16.964\,602\,467\,354\,44 \times 10^{0}$

Table 17: This table shows the scenarios of the special test cases and their respective difference measure, which is calculated with the AEPs of `PyWake` and `WindFarm3D`. The green cells contain scenarios with negligible or no errors.

Tool Name	AEP [mWh]
WindFarm3D	39 439.117 207
PyWake	32 748.427 756

Table 18: This table shows the simulation results of scenario Terrain Wake Group: Basic Wake using different tools

Note that `PyWake` calculates the same AEP for scenario Terrain Wake Group: Basic Wake and scenario Jensen Wake Intensity Group: Full Wake, which have the same setup but without the height difference. It follows that `PyWake` ignores the elevation levels of turbines. This result is also in accordance with the other special test cases regarding elevation. In the Raised Turbine test case, no difference to scenario Vestas Single Direction Group (Table 1): Medium Constant Wind is expected because the log shear is based on the turbine height relative to the ground. Analogously, no difference is expected between scenario Terrain Wake Group: No Interference and scenario Jensen Wake Intensity Group: No Interference because the wake decay factor is also based on the height relative to the ground. `PyWake` meets both of these expectations.
In conclusion, `PyWake` can not calculate 3D Wake properties. This means that level `PyWake` will calculate faulty results in most cases with different elevations. To be able to check for other errors in the test cases, `WindFarm3D` will also ignore all elevation changes from now on.

4.2.7 Flat Constant Speed Wind Parks

To close on the constant-speed wind parks, WindProof runs a pipeline with 2500 random scenarios. This time, the number of turbines varies between 2 and 10. As seen in Figure 30b, no error in this final validation reaches a significant level.

Conclusion and Outlook

This thesis presented WindProof, a validation framework for wind farm simulations with its general and extendable design. WindProof can compare different wind farm simulation tools and check them for errors. Section 3 unveiled the vast library of test cases WindProof provides to achieve this. These test cases cover the different aspects of wind farm simulation and AEP calculation, allowing for systematic troubleshooting on these simulation software. For any aspects the test cases might have missed, WindProof also provides a module to generate random test cases.

Finally, section 4 showed the application of WindProof onto the simulation tools `PyWake` and `WindFarm3D`. In this evaluation, WindProof discovered six differences between the tools, which are the following:

- **Power Curve Interpolation:** Windproof discovered a faulty interpretation of the power curve data in `WindFarm3D`, which led to an interpolation error.
- **Wake Decay Factor:** While `PyWake` took the site-wise Wake Decay Factor as a user input, `WindFarm3D` approximates it for each turbine.
- **Sector Width:** When the sector width is larger than 1°, `WindFarm3D` assumes the wind is coming from the sector middle line, while `PyWake` interpolates the probabilities for a smooth transition to the neighboring sectors.
- **Cirle Intersection:** A mistake was found in `PyWake`, where the circle intersection was not correctly determined if the wake radius was smaller than the radius of the wake-affected turbine.
- **Wind Speed Adaption Factor:** `PyWake` calculates the velocity deficit in relation to the wind speed at the wake-inducing turbine. In contrast, `WindFarm3D` determines the velocity deficit relative to the free flow wind speed.
- **Flat Terrain:** `PyWake` cannot process different elevations and assumes a flat terrain.

These differences caused the simulation results to differ significantly. Even one of these differences alone could produce relative errors of up to 16 %.

▶ Limitations

Unfortunately, WindProof has its limitations. The framework is constraint by the following:

- Since WindProof's analysis is empirical, the quality of this analysis depends on the quality of the test cases used. WindProof will not discover an error source if no test cases cover it. This problem was encountered several times while evaluating `PyWake` and `WindFarm3D`, where additional test cases were added while assessing.

- WindProof provides no standard values to which the simulation results can be compared. In this thesis, these values were calculated by hand when needed. It follows that to use WindProof effectively, the user is required to be proficient with the calculation model.
- The used tools must have an API for Python, or they can not be integrated into WindProof.
- WindProof has no automated analysis, which means that to draw the correct conclusions from the simulation results, the user must be familiar with the test cases and, therefore, with WindProof's data structures.

▶ Outlook & Future Work

To apply the results of this thesis, `PyWake` and `WindFarm3D` should fix these errors. WindProof could then be used to verify the effectiveness of the improvements.
Additionally, WindProof could be extended in several areas. Firstly, WindProof's test case collection can be extended to mitigate the effects of its limitations. Second, an automatic analysis module could be added to WindProof to help draw quick conclusions. It would also decrease the difficulty of using WindProof. Finally, a list of referential AEP values for each scenario could be added to WindProof. It would provide a comparison basis for the simulation results and decrease the difficulty of using WindProof.
To conclude this thesis, hopefully, WindProof could support the improvement and troubleshooting of wind farm simulation tools. It would also help to improve the planning and optimization of wind farms, making renewable energy more widely available and cheaper.

References

[1] (BWE), B. W. e.V. *Windenergie in Deutschland - Zahlen und Fakten*. Accessed: 2024-10-30. 2023. URL: https://www.wind-energie.de/themen/zahlen-und-fakten/deutschland/.

[2] (Destatis), S. B. *Stromerzeugung 2023: 56 % aus erneuerbaren Energieträgern*. Accessed: 2024-10-30. 2024. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2024/03/PD24_087_43312.html.

[3] Acker, T. and Chime, A. H. "Wind modeling using WindPro and WAsP software". In: *Norther Arizon University, USA* 1560000.8.8 (2011), p. 510.

[4] Archer, C. L. et al. "Review and evaluation of wake loss models for wind energy applications". In: *Applied Energy* 226 (2018), pp. 1187–1207. URL: https://www.sciencedirect.com/science/article/pii/S030626191830802X.

[5] Bartl, J., Pierella, F., and Sætrana, L. "Wake Measurements Behind an Array of Two Model Wind Turbines". In: *Energy Procedia* 24 (2012). Selected papers from Deep Sea Offshore Wind RD Conference, Trondheim, Norway, 19-20 January 2012, pp. 305–312. URL: https://www.sciencedirect.com/science/article/pii/S1876610212011538.

[6] Herbert-Acero, J. F. et al. "A Review of Methodological Approaches for the Design and Optimization of Wind Farms". In: *Energies* 7.11 (2014), pp. 6930–7016. URL: https://www.mdpi.com/1996-1073/7/11/6930.

[7] Jensen, N. O. *Note on wind generator interaction. [Wakes]*. 1983.

[8] Jeon, S., Kim, B., and Huh, J. "Comparison and verification of wake models in an onshore wind farm considering single wake condition of the 2 MW wind turbine". In: *Energy* 93 (2015), pp. 1769–1777. URL: https://www.sciencedirect.com/science/article/pii/S0360544215012967.

[9] Katic, I., Højstrup, J., and Jensen, N. O. "A simple model for cluster efficiency". In: *European wind energy association conference and exhibition*. A. Raguzzi. 1987, pp. 407–410.

[10] Maghnie, M. "Simulation and Layout Optimization of Offshore Wind Farms". MA thesis. LuFG Theory of Hybrid Systems at RWTH Aachen University, 2019.

[11] Miller, H. R. *Optimization: Foundations and Applications*. 2011.

[12] Parada, L. et al. "Wind farm layout optimization using a Gaussian-based wake model". In: *Renewable Energy* 107 (2017), pp. 531–541. URL: https://www.sciencedirect.com/science/article/pii/S0960148117300952.

[13] Pedersen, M. M. et al. *PyWake 2.5.0: An open-source wind farm simulation tool*. 2023. URL: https://gitlab.windenergy.dtu.dk/TOPFARM/PyWake.

[14] Rathmann, O. S., Frandsen, S., and Nielsen, M. "Wake Decay Constant for the Infinite Wind Turbine Array: Application of Asymptotic Speed Deficit Concept to Existing Engineering Wake Model". In: *Proceedings of the European Wind Energy Conference and Exhibition (EWEC), Technical Track "Wake"*. Paper ID 265. Risø-DTU National Laboratory for Sustainable Energy, Wind Energy Division. Roskilde, Denmark, 2010. URL: https://backend.orbit.dtu.dk/ws/portalfiles/portal/5118761/Rathmann_paper_ewec_2010.pdf.

[15] Seim, F., Gravdahl, A. R., and Adaramola, M. S. "Validation of kinematic wind turbine wake models in complex terrain using actual windfarm production data". In: *Energy* 123 (2017), pp. 742–753. URL: https://www.sciencedirect.com/science/article/pii/S0360544217301470.

[16] Statista. *Global Wind Energy Production 2000-2022*. Accessed: 2024-10-30. 2024. URL: https://www.statista.com/statistics/1031138/wind-energy-production-globally/.

[17] Tamura, T. "Towards practical use of LES in wind engineering". In: *Journal of Wind Engineering and Industrial Aerodynamics* 96.10 (2008). 4th International Symposium on Computational Wind Engineering (CWE2006), pp. 1451–1471. URL: https://www.sciencedirect.com/science/article/pii/S0167610508000263.

[18] Thomas, J. J. et al. "A comparison of eight optimization methods applied to a wind farm layout optimization problem". In: *Wind Energy Science* 8.5 (2023), pp. 865–891. URL: https://wes.copernicus.org/articles/8/865/2023/.

[19] University, R. A. *WindFarm3D Repository*. Accessed: 2024-10-28. 2022. URL: https://git.rwth-aachen.de/renewable-energy/windfarm-simulation.