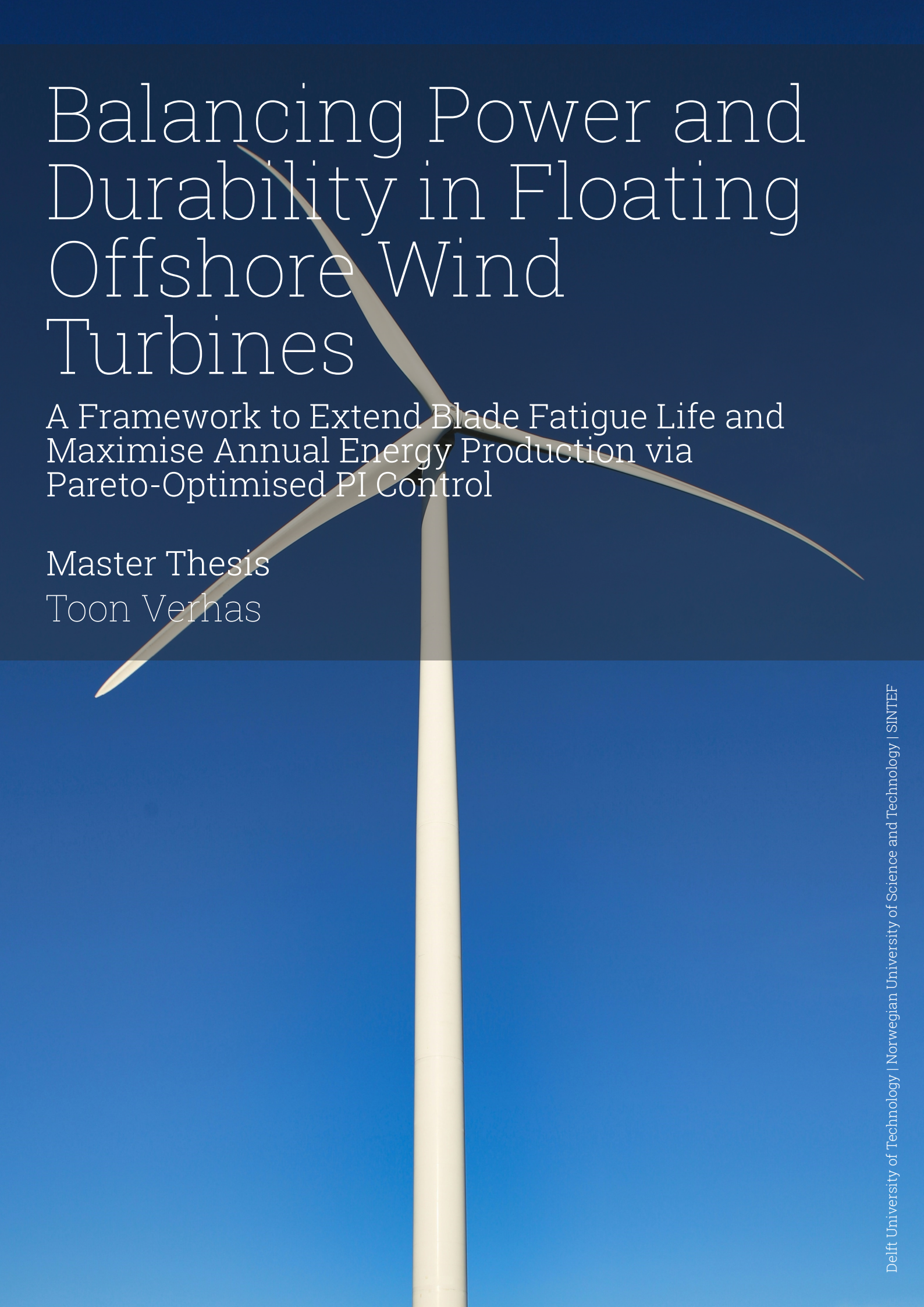
Balancing Power and Durability in Floating Offshore Wind Turbines

A Framework to Extend Blade Fatigue Life and Maximise Annual Energy Production via Pareto-Optimised PI Control

Master Thesis

Toon Verhas



Delft University of Technology | Norwegian University of Science and Technology | SINTEF

This page was intentionally left blank.

Balancing Power and Durability in Floating Offshore Wind Turbines

A Framework to Extend Blade Fatigue Life and Maximise Annual Energy Production via Pareto-Optimised PI Control

by

Toon Verhas

to obtain the degrees of

Master of Science in Offshore & Dredging Engineering

at the Delft University of Technology

&

Master of Science in Wind Energy

at the Norwegian University of Science and Technology,

to be defended publicly on Friday June 20, 2025.

Thesis committee:	Prof. dr. ir. JW. van Wingerden,	TU Delft, supervisor
	Prof. dr. ir. D. T. Nguyen,	NTNU, supervisor
	Ir. S. B. Lysthaug,	SINTEF Ocean, supervisor
	Dr. ir. A. Jarquin Laguna,	TU Delft
	Dr. ir. S. Schreier,	TU Delft
Thesis duration:	January 1, 2025 – June 20, 2025	
Student number:	TU Delft - 5006546	NTNU - 123738

Cover: An offshore wind turbine is shown against a blue sky [52].

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft NTNU SINTEF

Preface

I did it!

This thesis marks the final chapter of my time as a student, and the six-year journey leading up to it has been an amazing adventure, to say the least. Living in the Netherlands, Denmark, and Norway has been transformative, and along the way, I've met some incredible people who have been instrumental to my story.

First and foremost, I would like to thank the people who have guided me over the past few months and helped bring this thesis to a successful end. Thank you to Simen Lysthaug, who gave me the opportunity to experience what a company like SINTEF Ocean is like, and for the hospitality that made my time in Trondheim so enjoyable. I also want to thank my supervisors, Dong Trong Nguyen and Jan-Willem van Wingerden. You both helped me out when the going got tough, and I sincerely appreciate the support I received.

There are some other people I want to shine a spotlight on. Starting with my parents, who have been there every step of the way. From dropping me off at school when I was three to helping me check my car and packing some extra food for a trip to Norway. I am eternally grateful for the way they have always stood behind me. The same goes for my brother. He is the one I always try to make proud, and the one I admire the most. Thank you to the three of you. I hope I'll continue to make you proud in the next steps.

Besides family, I would be remiss not to mention my amazing friends.

Anton, Camille, Louis, Hanlin, Korneel, Bo, Boris, Sebastien, Killian, and many more. You push me to stay ambitious, and I can't express how proud I am of all of you. My time as a student was the best because of you. May the worldwide meetups continue forever. Love you all.

Lastly, love you Fé. Thank you for the mental and emotional support this past year. It propelled me to excel academically and personally, and I hope I can do the same for you.

Toon Verhas
Delft, June 2025

Abstract

Floating offshore wind energy is a technology that has gained significant interest over the past few years due to its potential to unlock vast, high-quality wind resources far from shore and in deeper waters, areas that fixed-bottom turbines cannot reach. However, harnessing this resource presents unique challenges, among them are maximising annual energy production (AEP) while minimising blade fatigue damage. These objectives inherently conflict. Aggressive control settings that boost energy capture tend to increase loads, accelerating material fatigue. Crucially, both AEP and fatigue life depend on the turbine controller. By systematically varying a set of key features of a proportional–integral (PI) controller, this thesis sets up a workflow to investigate how controller parameters shape the trade-off between AEP and blade fatigue. To navigate this multi-objective landscape, Pareto optimisation is employed, generating a front of controller configurations that balance energy yield against blade durability. Economic viability is assessed through levelised cost of energy (LCOE) calculations, linking extended fatigue life to deferred maintenance costs and potential improved lifetime. Within the existing literature, where past studies have treated AEP, fatigue, and control in isolation, this work offers a unified framework, demonstrating how slight adjustments in controller settings can unlock significant performance gains. The core contribution of this thesis lies in the framework itself. An end-to-end roadmap is presented, combining metocean data analysis from Utsira Nord, aero-servo-hydrodynamic simulations using SIMA, AEP analysis and fatigue life estimation, Pareto front construction, and LCOE impact assessment. Key results show that blade life can be extended by over 15% with less than a 1% drop in AEP, translating to a meaningful reduction in LCOE of at least 3% under typical economic assumptions. These findings highlight the opportunity to explore the controller parameter space further with this system, paving the way for more finely tuned strategies that optimise the balance between the two objectives.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AEP	Annual Energy Production
ATD	Active Tower Damper
CAPEX	Capital Expenditure
DEL	Design Equivalent Load
FEM	Finite Element Model
FOWT	Floating Offshore Wind Turbine
IEA	International Energy Agency
JONSWAP	Joint North Sea Wave Project
LCOE	Levelised Cost of Energy
MPC	Model Predictive Control
MORL	Multi-Objective Reinforcement Learning
NREL	National Renewable Energy Laboratory
O&M	Operations & Maintenance
OPEX	Operational Expenditure
PDF	Probability Density Function
PI	Proportional-Integral
RPO	Response Power Operator
ROSCO	Reference OpenSource Controller
RNA	Rotor Nacelle Assembly
SIMA	Simulation Workbench for Marine Applications
SWL	Still Water Level
TSR	Tip Speed Ratio
UMaine	The University of Maine

Symbols

Symbol	Definition	Unit
A	System Matrix	[-]
a	Strength Coefficient	[-]
A_r	Rotor Swept Area	[m^2]
b	Fatigue Exponent	[-]
B	Input Matrix	[-]
C_p	Power Coefficient	[-]
C_t	Thrust Coefficient	[-]
D	Accumulated Damage	[-]

Symbol	Definition	Unit
F	Fatigue Life	[years]
f	Frequency	[Hz]
$f(U)$	Wind Speed Distribution	[-]
H	Wave Height	[m]
H_s	Significant Wave Height	[m]
I_u	Turbulence Intensity	[%]
I_x	Moment of Inertia about the x-axis	[m^4]
I_y	Moment of Inertia about the y-axis	[m^4]
J	Rotational Inertia	[$kg \cdot m^2$]
J_p	Polar Moment of Inertia	[m^2]
k	Shape Factor	[-]
K_I	Integral Gain	[-]
K_P	Proportional Gain	[-]
$K_{\beta_{float}}$	Floating Feedback Gain	[-]
m_0	Zeroth Spectral Moment	[m^2]
m_{blade}	Blade Mass	[kg]
M_x	Moment around x-axis	[Nm]
M_y	Moment around y-axis	[Nm]
M_z	Moment around z-axis	[Nm]
N	Number of Cycles	[-]
N_g	Gearbox Ratio	[-]
P	Power	[W]
Q	Mechanical Torque	[Nm]
r	Discount Rate	[%]
R	Radius	[m]
R_i	Inner Radius	[m]
R_o	Outer Radius	[m]
S_a	Stress Amplitude	[Pa]
S_{eq}	Equivalent Stress Amplitude	[Pa]
$S_u(z,f)$	Power Spectral Density	[$\frac{m^2}{rad/s}$ or $\frac{(m/s)^2}{rad/s}$]
T	Wave Period	[s]
T_p	Peak Wave Period	[s]
T_r	Rotor Thrust	[N]
u	Controller Input	[-]
u^*	Friction Velocity	[m/s]
U	Wind Velocity	[m/s]
y	Controller Output	[-]
z	Height	[m]
z_0	Surface Roughness Length	[m]
c	Scale Factor	[m/s]
c_w	Wave Speed	[m/s]
n	Project Life Time	[years]
α	Power Law Exponent	[-]
β	Blade Pitch Angle	[$^\circ$]
η	Free Surface Elevation	[m]
η_{gb}	Gearbox Efficiency	[-]

Symbol	Definition	Unit
κ	von Kármán constant	[-]
λ	Tip Speed Ratio	[-]
λ_w	Wave Length	[m]
ρ	Density	[kg/m^3]
σ_m	Mean Stress	[Pa]
σ_U	Standard Deviation of Wind Speed	[m/s]
σ_y	Bending Stress in y	[Pa]
σ_z	Bending Stress in z	[Pa]
σ_{vm}	von Mises Stress	[Pa]
τ	Torsional Stress	[Pa]
τ_a	Aerodynamic Torque	[Nm]
τ_g	Generator Torque	[Nm]
ω	Natural Frequency	[Hz]
ω	Rotor Speed	[rpm]
ω_g	Generator Speed	[rpm]
ω_w	Wave Radian Frequency	[rad/s]
Ω	Rotor Angular Velocity	[rad/s]
ζ	Damping Ratio	[-]

1

Introduction

1.1. Background

The transition to renewable energy is essential to mitigating climate change and reducing dependence on fossil fuels. Offshore wind energy has emerged as a key contributor to this transition, offering vast energy potential and more consistent wind resources compared to onshore installations. This underscores the importance of maximising energy production, which remains a key objective in the effort to fully harness renewable resources. However, the expansion of offshore wind is moving in the direction of floating offshore wind turbines (FOWTs) and this remains an expensive undertaking. The levelised cost of energy (LCOE) for floating wind is still significantly higher than that of fixed-bottom offshore wind farms, primarily due to the high upfront capital expenses, the complexity of floating structures, installation challenges, and larger distances off the coast. As of 2035, the estimated LCOE for floating wind projects is approximately 165 €/MWh, compared to 76 €/MWh for bottom-fixed offshore wind [37].

One of the primary cost drivers in FOWTs is component reliability and maintenance, with wind turbine blades being among the most critical structural elements. Additionally, wind turbine blades are difficult components to recycle due to their large size, complex structure, and the materials used in their manufacturing [59]. Furthermore, blade failures account for a significant proportion of major component failures, leading to extensive downtime and expensive replacements [17]. Even though complete blade failures are rare, a recent example of this occurred at the Vineyard Wind offshore wind project in July of 2024, where a blade sustained severe damage, as shown in Figure 1.1. This incident sparked significant media attention and raised concerns about blade durability. Therefore, thinking about blade failure is highly relevant, especially with increasing turbine sizes and loads.

Figure 1.1: Damaged blade at the Vineyard Wind offshore wind project [32].

A crucial factor influencing both energy production and blade fatigue loads in wind turbines is the control strategy. The industry standard, proportional-integral (PI) control, has been widely adopted because of its robustness and effectiveness in regulating turbine performance. However, the two main objectives that define turbine performance, maximising annual energy production (AEP) and extending fatigue, are fundamentally in conflict.

Combining these two goals into a single control approach requires a method that can evaluate both aspects simultaneously. Pareto optimisation provides a structured way to explore the trade-off between competing objectives, identifying solutions that are balanced. Although some research has used Pareto methods in wind turbine control, previous work has focused on different aspects of the system. For example, Brandetti et al. (2023) applied Pareto optimisation to investigate the trade-off between maximising power and minimising torque fluctuations [9]. To date, no study has directly examined how different gain-scheduled PI controllers influence AEP, blade fatigue life, and economic performance in a combined analysis. While each of these aspects has been explored separately in the literature, their integration into a unified framework remains largely unaddressed.

1.2. Research Objective

The above-mentioned gap is addressed by creating such a framework. Starting of by executing a systematic parametric tuning of a PI controller that generates proportional and integral gains, resulting in a spectrum of gain-scheduled PI controllers for floating offshore turbines. This research uses the Reference OpenSource Controller (ROSCO), developed by the U.S. National Renewable Energy Laboratory (NREL). ROSCO allows flexible tuning of the controller, making it well-suited for systematic performance comparisons across different gain configurations. These controllers are implemented in simulations using the SIMA software, driven by site-specific metocean data. The simulation results are used to compute annual energy production and blade fatigue life. A Pareto front analysis is then carried out to map the trade-off between these two objectives and identify optimal controller configurations. Finally, a separate economic assessment evaluates how changes in fatigue life and maintenance needs affect the levelised cost of energy, providing insight into the long-term financial implications of different control strategies.

In essence, the research objective can be defined as:

Develop and apply a modular framework to assess how variations in underlying ROSCO controller parameters, used to derive gain schedules, affect both annual energy production and blade fatigue life in floating offshore wind turbines, and to evaluate the economic implications through a levelised cost of energy analysis.

1.3. Research Questions

To achieve this objective, the following key research questions are explored:

- What PI-based control strategies are in use for floating offshore wind turbines, and how do their core parameter settings influence performance?
- What are important controller parameters affecting the trade-off between fatigue and annual energy production in FOWT control strategies?
- How significant do the different parameter settings within the same PI controller law affect blade fatigue reduction and power performance in FOWTs?
- What role do metocean conditions play in optimising a PI controller?
- How can Pareto front analysis be applied to optimise a PI controller for FOWTs that balance blade fatigue life and energy yield?
- How does extending the fatigue life of turbine blades impact the long-term economic viability and energy production of FOWTs?
- How can a modular control framework be applied to optimise floating offshore wind turbine performance across a range of operational scenarios?

By addressing these research questions, this study aims to fill the gap in the existing literature and contribute valuable insights, laying the groundwork for future research.

1.4. Structure of the Thesis

The thesis is organised into five chapters that guide the reader from background to conclusions. Chapter 2 begins with a critical review of proportional integral control strategies for floating offshore wind turbines, the foundations of annual energy production calculations, fatigue life estimation for composite blades, Pareto optimisation methods and common financial assessment models. In Chapter 3, the offshore environment is described through wind and wave spectral representations before presenting the theory behind the ROSCO generator torque and blade pitch control with adaptations for improved efficiency and incorporation of floating platforms. The chapter also covers AEP calculation, fatigue analysis via rainflow counting and S-N curves, Pareto front concepts and the levelised cost of energy formula. Chapter 4 details the modular framework, including SIMA and ROSCO model setup and validation, the selection and variation of controller parameters, metocean scenario generation and the procedures for evaluating AEP, fatigue and LCOE. In Chapter 5, model validation results precede the presentation of metocean statistics, simulated AEP and fatigue outcomes, the constructed Pareto front and LCOE comparisons for chosen controller cases. Finally, Chapter 6 interprets key findings and discusses limitations, and Chapter 7 offers the final remarks and recommendations.

2

Literature Review

Floating offshore wind turbines have garnered significant interest in recent years, resulting in a growing body of literature. This chapter will synthesise the current state of the art on key developments in PI control for FOWTs, the importance of maximising annual energy production, opportunities for extending blade fatigue life, the balancing of conflicting objectives, and the economic implications of these factors.

2.1. PI Control Strategies for FOWTs

Proportional-Integral control has long been the industry standard in the wind energy industry. Due to its simplicity and robust nature. For floating wind turbines, strategies for PI control have modifications in order to secure adequate performance.

The ROSCO controller literature, presented by Abbas et al. (2022), serves as an excellent baseline and will be used throughout the remainder of this thesis. ROSCO employs gain-scheduled PI controllers for both generator torque and collective blade pitch control, with its tuning process driven by the turbine's power coefficient surface. Notably, it incorporates a floating-specific feedback module to account for platform dynamics, enhancing stability in FOWT applications. Its modular architecture and automated tuning via the ROSCO toolbox make it versatile and easily adaptable across different turbine designs, providing a robust reference framework for further control strategy development [1].

Another interesting feedback approach was presented by Lenfest et al. (2020). In this study the challenges of applying proportional-integral (PI) control to floating offshore wind turbines were addressed, particularly the destabilising effects from the interaction between the turbine controller and platform dynamics. They proposed a straightforward tuning strategy that incorporates nacelle velocity feedback into an existing gain-scheduled PI controller. Utilising a two-degree-of-freedom model focusing on tower-top fore-aft and rotor angular displacements, they developed a method to efficiently schedule feedback gains across various turbine and hull configurations. Their approach was evaluated against baseline land-based and detuned controllers using an example system under multiple load cases. The results demonstrated that the modified controller effectively mitigated adverse platform-controller interactions, enhancing overall system

stability without the need for complex tuning procedures [31]. This work underscores the importance of tailored PI control strategies in FOWTs to ensure reliable performance amidst the unique dynamic challenges posed by floating platforms.

A similar modification can be found in the paper by Olondriz et al. (2019) where an enhanced detuned and gain-scheduled PI controller was presented. This was done by incorporating a blade load feedback loop that uses blade-root flapwise bending moment measurements. This modification helps distinguish between wind- and wave-induced motions, leading to improved platform stabilisation and reduced blade loads under rough sea conditions, with a slight increase in tower loads observed in simulations [46].

Similarly, Meng et al. (2023) demonstrated that incorporating a tower top velocity feedback loop into a PI-based blade pitch controller can effectively mitigate floater pitch instabilities in floating wind turbines. A block diagram, which is presented in the paper but is applicable to several of the mentioned studies, can be seen in Figure 2.1. Their experimental study on a scaled turbine model revealed that an appropriately tuned tower feedback gain helps suppress the oscillations induced by negative aerodynamic damping [38].

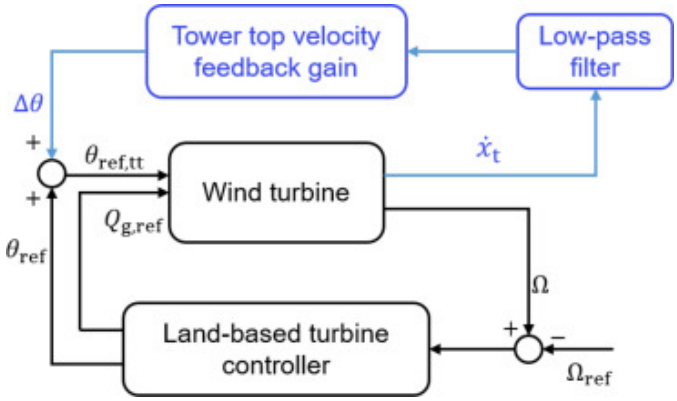


Figure 2.1: Block diagram of a controller with tower top velocity feedback loop [38].

López-Romero et al. (2024) present a PI-based Active Tower Damper (ATD) architecture that uses blade-pitch modulation. This is done via two simple PI loops (one on tower-top acceleration, one on velocity) to mitigate tower vibrations on a floating turbine model in OpenFAST. They compare both configurations in the time and frequency domains and demonstrate clear reductions in vibration amplitudes and frequency response peaks across the tower spectrum, confirming that a standard PI framework can effectively damp tower motion without added hardware complexity [34].

Another way of improving the effectiveness of the controller is by feedforward control. More specifically, Scholbrock et al. (2016) looked into the possibility of using LiDAR to detect incoming wind. In their synthesis of the existing state of the art, they concluded that integrating this can positively impact power regulation [56].

Additionally, gain scheduling has been the focus of several recent studies, highlighting its relevance in the development of control strategies for wind turbines. Kumar and Detroja (2024) propose a novel method for designing gain-scheduled PI controllers for wind turbines using Multi-Objective Reinforcement Learning (MORL). Their research addresses

the challenge of achieving optimal control performance across varying wind conditions while managing competing objectives such as power regulation and system stability. By leveraging MORL, the authors generate gain schedules that adapt to different operating regions, replacing conventional manual tuning with a data-driven approach. Their framework is tested on a floating wind turbine model, where the MORL-tuned PI controller demonstrates improved energy capture compared to traditional gain-scheduled methods [47]. This study highlights the potential of intelligent control tuning methods to enhance the adaptability and effectiveness of PI-based strategies for FOWTs.

Lemmer et al. (2019) introduced a robust gain-scheduled PI controller tailored for floating offshore wind turbines, addressing the pitch instability challenges inherent in such systems. Unlike advanced multi-input-multi-output controllers that often require additional platform state measurements, their approach utilises a standard PI controller structure, feeding back only the rotor speed error. This simplicity enhances robustness by reducing sensitivity to unmodeled dynamics and facilitates full automation of the model-based control design process. A tailored linearised coupled dynamic model, incorporating detailed hydrodynamic viscous drag, was employed to design the controller, with stability margin serving as the primary design criterion across varying wind speeds. The resulting gain scheduling function differs fundamentally from that of fixed-bottom turbines, reflecting the unique dynamics of floating platforms. Performance evaluations against advanced controllers and fixed-bottom turbine versions demonstrated that, despite its simplicity, the proposed controller meets common design requirements, offering a viable and robust solution for FOWT applications [30].

These studies collectively underscore the adaptability and enduring relevance of PI control strategies in the floating offshore wind sector. By incorporating platform-specific feedback mechanisms, gain scheduling, and even data-driven tuning approaches like reinforcement learning, researchers continue to push the boundaries of what simple PI frameworks can achieve. As floating wind turbines face increasingly complex dynamic environments, these tailored enhancements ensure that PI control remains a robust, scalable, and effective solution for next-generation offshore wind applications.

2.2. Importance of Annual Energy Production

Annual Energy Production quantifies the total electrical energy a turbine or wind farm delivers over a year, making it the key performance indicator for both technical assessment and financial viability. For industry players, it is often an indication to analyse potential sites and compare turbine capabilities so that they can then make informed decisions.

Lee and Fields (2021) conducted a comprehensive review to understand and quantify the biases, losses, and uncertainties associated with wind energy production predictions. Recognising that overestimations in annual energy production can have significant financial implications for wind projects, they analysed over 150 sources, including industry reports and academic studies, to assess the accuracy of preconstruction energy yield assessments. Their findings indicated a historical trend of overprediction in AEP estimates, with recent improvements attributed to advancements in modelling and measurement techniques. However, they also identified persistent uncertainties, particularly related to wake effects, environmental losses, and wind speed variability. The study emphasised the need for standardised assessment frameworks, such as the IEC 61400-15, and highlighted the importance of continued efforts to validate and reduce

prediction uncertainties to enhance the reliability and financial viability of wind energy projects [29].

Amaral et al. (2022) addressed the computational challenges of estimating the Annual Energy Production (AEP) of floating offshore wind turbines during early design stages. To reduce reliance on thousands of fully coupled time-domain simulations, they proposed a frequency–time domain approach. This method uses Response Power Operators (RPOs), which relate average turbine power output to single-degree-of-freedom platform motions of varying frequency and amplitude. These RPOs, generated using OpenFAST for the IEA 15 MW WindCrete spar-buoy model, are combined with site-specific spectral wave data to estimate AEP. Their approach was validated against conventional simulation-based results and showed good agreement, offering a faster and reasonably accurate alternative for preliminary AEP assessment [5].

Sedaghat et al. (2017) investigated how to determine the optimal rated wind speed for variable speed wind turbines in order to maximise AEP. They introduced a new capacity value metric, which integrates the turbine's power curve with site-specific wind speed distributions modelled using the Weibull function. Their findings showed that the optimal rated wind speed depends strongly on the shape factor of the wind distribution, typically falling between 2 and 5 times the annual mean wind speed. For example, at a mean wind speed of 4 m/s, the optimal rated speed ranged between 9 and 20 m/s depending on the shape factor. The study highlighted that a mismatch between turbine rating and wind site characteristics could reduce AEP by as much as 43 percent, emphasising the importance of tailored turbine design for specific wind sites [58].

Fontanella et al. (2024) examined how waves and platform dynamics affect the AEP of floating offshore wind turbines. They analysed four FOWT designs ranging from 5 to 15 MW, supported by spar and semi-submersible platforms, under realistic Mediterranean Sea conditions. The study found that, contrary to the theoretical possibility of waves enhancing power output by inducing rotor motion against the wind, actual platform designs aim to minimise such movements to reduce structural loads. Consequently, the anticipated power gains from wave-induced motions are not realised in practice. Additionally, the research highlighted that wind turbulence, rather than platform motion, is the primary driver of power fluctuations. Overall, the AEP of the analysed FOWTS was found to be 1.5% to 2.5% lower than that of fixed-bottom turbines, primarily due to platform tilt reducing the rotor's effective wind-facing area. These findings underscore the importance of considering platform dynamics in the accurate estimation of AEP for floating wind projects [20].

The impact of the controller on the AEP can not be overstated. And in simulations, it is important to come as close to reality as possible. Therefore, Song and Paek (2020) developed a dynamic wind turbine model incorporating torque and pitch control algorithms which were tuned to match the power curve of the target wind turbine, including a peak shaving feature, to predict AEP. The model was validated against measured data, achieving a prediction error as low as 0.16% [63].

These studies demonstrate that AEP is influenced not only by the wind resource and turbine design but also by the accuracy of modelling techniques, the quality of control strategies, and the dynamic behaviour of floating platforms. As the floating wind sector continues to expand, refining these aspects is crucial to ensure reliable performance predictions and secure investor confidence through accurate energy yield assessments.

2.3. Fatigue Life Estimation and Extension

Fatigue life estimation is a critical aspect of wind turbine reliability, as turbine components, particularly blades, experience variable and cyclic loading that can lead to material degradation over time. Accurate fatigue estimation is essential for effective maintenance planning and the design of more resilient turbine structures.

Fatigue analysis for composites is hard, but nevertheless insightful. It is clear form the existing literature and was also found by Jahani et al. (2022) that predicting the fatigue life of composite blades is challenging because of environmental variability (wind, waves, temperature, humidity), a lack of comprehensive fatigue data accompanied by large scatter, non-linear damage accumulation in the material, and uncertainties introduced during manufacturing [24].

Finite element models (FEM), commonly used for fatigue analysis, is often seen as the most accurate way of fatigue analysis. This method was done by multiple researchers such as Grujicic et al. (2010) [23] and Kulkarni et al. (2018) [27], in both cases, the external loads on a 5 MW wind turbine were estimated using time domain models and were subsequently distributed on the nodes of the FEM. The latter of the two employed deep neural networks to predict long-term fatigue behaviour of wind turbine blades and found that blade life was estimated to be 23.6 years. While this can be seen as an accurate way of predicting fatigue life, computational effort of FEM is expensive and often too cumbersome.

Another study that used FEM was executed by Liu et al. (2023), who evaluated the fatigue life of offshore composite wind turbine blades at the Zhoushan Islands by first compiling one year of site-specific wind measurements at Jintang Island and using a modified blade element momentum theory to derive time-domain aerodynamic loads, which were then mapped onto a full-scale finite element model of the blade for cycle-by-cycle damage tracking. They incorporated composite failure criteria and progressive damage laws within the FEM to capture stiffness degradation under varying load conditions and then applied rainflow counting alongside S–N fatigue curves and the Palmgren–Miner rule to compute cumulative damage. Their analysis clearly demonstrated that the point of interest is most definitely the blade root, which can be seen in Figure 2.2. Across three different composite skin materials, the calculations yielded fatigue life estimates of 19 to 22 years, in close agreement with the design lifespans specified for Chinese offshore turbines [33].

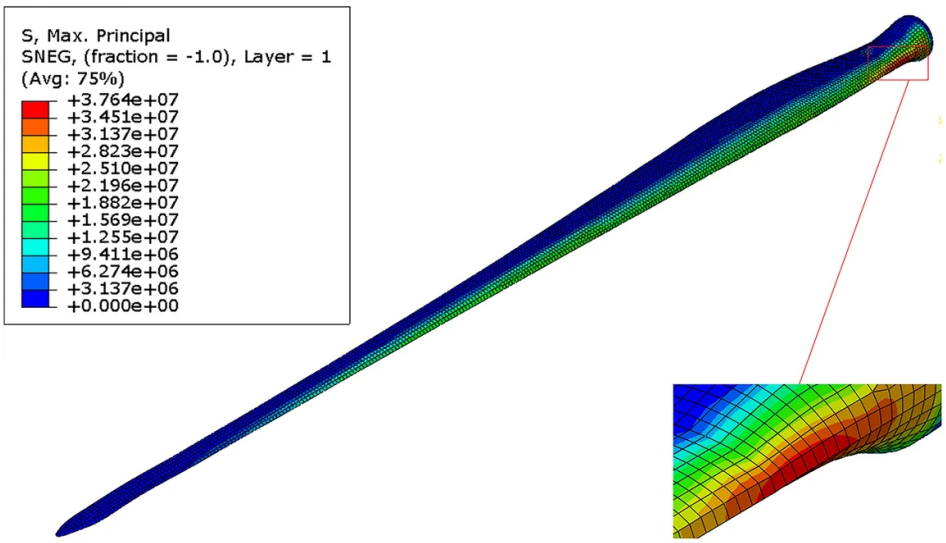


Figure 2.2: The maximum principal stress distribution along the blade [33].

Other options were explored as well. Sanchez et al. (2016) compared two distinct modelling approaches for fatigue estimation and remaining useful life predictions in wind turbine blades. The study contrasts the traditional rainflow counting algorithm, which decomposes complex load histories into equivalent cycles based on the Palmgren-Miner rule, with a fatigue stiffness degradation model that predicts damage progression by quantifying the reduction in material stiffness due to cyclic loading. Using blade root moment signals from high-fidelity FAST simulations under various constant wind speed scenarios, the authors demonstrated that while the rainflow method offers a straightforward empirical means of assessing fatigue damage, it can be computationally demanding and sensitive to parameter selection. Conversely, the stiffness degradation model provides a more physically meaningful prediction of damage evolution and remaining useful life, albeit at the cost of increased modelling complexity. Both approaches consistently indicate that higher mean stresses lead to accelerated damage accumulation, underscoring the importance of tailored fatigue estimation techniques for wind turbine blade prognostics [54].

This rainflow counting strategy was also used by Sirigu et al. (2024), where they conducted a multiaxial fatigue failure comparison between three aeroelastic models in OpenFAST. The results showed that there was a significant impact on expected lifetime of the blades based on the selection of the model, with outcomes ranging from 23.2 years to 1.2 years depending on the model. On the other hand, the models consistently predict similar patterns of where fatigue damage accumulates and the wind speeds at which it becomes critical [61].

Gao et al. (2022) also performed a multiaxial fatigue assessment of 15 MW floating turbine blades on three compliant platform types by applying nonlinear beam theory to derive three-dimensional blade displacements under combined wind-wave loading and converting those into both maximum principal and normal strain histories for rainflow-based damage counting. They found that blades mounted on floaters with very low rotational stiffness experience the highest fatigue damage rates, highlighting the critical influence of platform compliance on blade durability [22].

Additionally, Requate and Meyer (2020) investigated model predictive control strategies aimed at mitigating overall fatigue loads in wind turbines by leveraging active load control techniques. Their study focused on pitch and torque control adaptations that dynamically respond to fluctuating wind conditions, effectively redistributing aerodynamic forces to reduce structural fatigue. By integrating real-time load estimation with adaptive control adjustments, the proposed methodology demonstrated significant improvements in turbine lifespan while maintaining stable power output. The authors validated their approach through high-fidelity simulations, showing that optimised control strategies could achieve meaningful reductions in damage-equivalent loads across various wind scenarios. Their findings reinforce the importance of advanced control systems in fatigue load mitigation and highlight potential synergies with supervisory switching control strategies for floating offshore wind turbines [51].

Moreover, Smilden et al. (2018) analysed the effects of site-specific control strategies for fatigue load reduction of the monopile in fixed-bottom offshore wind turbines. Their study highlighted that fatigue life is highly sensitive to variations in environmental conditions such as water depth and soil properties. Using time-domain aero-hydro-servo-elastic simulations of a 10 MW wind turbine at Dogger Bank, the authors evaluated control strategies that extend fatigue life by dynamically adjusting pitch and torque control mechanisms based on site-specific conditions. The study demonstrated that a combination of active aerodynamic damping, active generator torque control, and soft cut-out strategies could lead to a 50% reduction in fatigue damage over a 20-year lifespan. However, they also emphasised the trade-offs, such as increased pitch actuator wear and fluctuations in power output. Their findings underscore the importance of integrating control strategies into the overall wind turbine design process to account for variations in fatigue loads across different site conditions [62].

Together, these studies show that fatigue analysis is highly variable, influenced by modelling choices, site conditions, and material behaviour. Despite these uncertainties, it remains a critical factor in wind turbine design and operation. Accurate lifetime estimation and the potential to extend it through targeted control strategies form a clear opportunity for improving reliability. This makes fatigue analysis and mitigation a highly relevant and worthwhile focus for further research in the field.

2.4. Navigating Conflicting Objectives

Floating offshore wind turbine control involves managing multiple, often conflicting objectives. Addressing these trade-offs requires methods that can evaluate and balance competing performance metrics. This section reviews relevant strategies from the literature that deal with such multi-objective challenges, including but not limited to Pareto-based approaches, and highlights how these methods inform controller design and evaluation in wind energy applications.

One critical component that is often researched is the tower. Lara et al. (2023) present a multi-objective control strategy aimed at simultaneously stabilizing power output and mitigating tower vibrations in wind turbines. Their approach integrates collective pitch control with two active damping mechanisms. One addresses frontal oscillations through additional pitch modulation, and another targets lateral displacements via generator torque adjustments. The control parameters are optimised using multi-objective techniques and multi-criteria decision-making methods, implemented through simulations in FAST

and MATLAB/Simulink. Under extreme wind direction changes, the proposed control scheme effectively reduces tower vibrations without significantly impacting power generation, demonstrating a successful balance between these conflicting objectives [28].

Oh et al. (2015) present a novel control algorithm that addresses the conflicting objectives inherent in floating offshore wind turbine operation. In their work, the authors address the trade-off between reducing tower loads and minimising power fluctuations, a challenge arising from the negative damping characteristics of conventional pitch control strategies at above-rated wind speeds. Their approach employs a region-switch mechanism within the pitch control loop, whereby the controller operates at a lower frequency in moderate wind conditions to enhance damping and suppress tower vibrations, and switches to a higher frequency in stronger winds to ensure prompt pitch responses for optimal power regulation [45]. The simulation results demonstrate that this adaptive strategy successfully balances load reduction against power capture, offering valuable insights for this research.

Fitzgerald and Sarkar (2024) present an observer-based individual pitch control strategy that navigates the conflicting objectives of power regulation and overall structural load mitigation in floating offshore wind turbines. Their approach employs a Kalman filter to estimate the turbine's full state using readily available measurements, such as blade strain and accelerometer data, in situations where direct state measurement is impractical. This estimated state is then used within a state-feedback framework for individual blade pitch control, enabling the system to simultaneously maintain optimal rotor speed and reduce structural loads that contribute to fatigue. Comparative analyses demonstrate that the observer-based controller achieves performance on par with full state-feedback designs, while surpassing traditional PI controllers in managing the trade-off between power output and load reduction [18].

When it comes to Pareto optimisation, Odgaard et al. (2016) explore the use of Pareto optimality in tuning a linear model predictive controller (MPC) for wind turbines. Their method involves generating Pareto curves to analyse the trade-off between maximising power output and minimising tower fore-aft fatigue loads. The MPC is evaluated on a high-fidelity Vestas wind turbine simulator, showing that it can achieve power outputs comparable to an industrial baseline controller while significantly reducing structural loads [44].

In another study, Zalkind et al. (2022) looked at optimising floating wind turbine control with the AEP and tower base damage equivalent loads (DELs) as the main objectives to optimise. This was done by implementing different thrust limits and then trying to find a balance, of which an optimisation plot can be seen in Figure 2.3. Doing this resulted in the finding that an increase in 1% AEP resulted in 5% increase in tower base DELs and vice versa [70]. This highlights the interconnectedness of the controller, the production of energy and the structural integrity of the system.

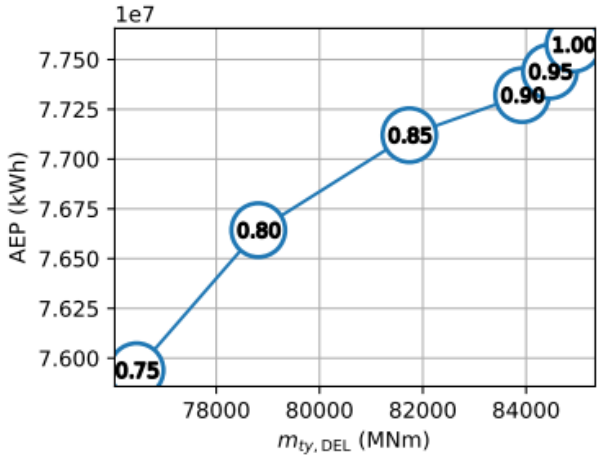


Figure 2.3: Results of setting different thrust limits (from 0.75 to 1.00) on the AEP and tower base damage equivalent loads [70].

Brandetti et al. (2023) applied Pareto optimisation to analyse multi-objective torque control strategies. Their work formulated a trade-off between power maximisation and torque fluctuation minimisation, revealing that while the baseline controller achieves near-optimal energy capture, a slight reduction in controller bandwidth (resulting in a power decrease of about 2%) can lead to an 80% reduction in torque actuation effort. These findings underscore the effectiveness of Pareto optimisation in balancing control performance and structural load reduction, thereby enhancing overall turbine reliability [9].

Together, these studies make clear that navigating the trade-off between power production and structural load mitigation is central to effective wind turbine control, especially for floating systems, where platform dynamics further complicate this balance. Pareto optimisation emerges as a powerful tool to explore these trade-offs and guide controller design. It not only supports better lifetime and performance outcomes but also offers a structured way to evaluate the impact of control decisions. What is clear, however, is that blade fatigue life and AEP have not yet been thoroughly examined in such a way.

2.5. Economic Impact

The economic feasibility of the proposed strategy will be influenced by two main points, added revenue by extending the life of the system and the reduction in maintenance costs. Previous studies have mapped out how both these elements play a role in the financial aspect of the wind turbine industry.

Some studies qualitatively address the economic impact. Rubert et al. (2019), for example, propose a comprehensive decision-making framework that leverages structural health monitoring data to evaluate the economic viability of wind turbine lifetime extension. By integrating long-term structural health measurements with fatigue analysis simulations, the study reduces uncertainties in estimating remaining useful life and quantifies the impact on the levelised cost of energy. Their analysis reveals that a strategic lifetime extension can be economically attractive when no major component replacements are required [53].

Additionally, Pustina et al. (2023) aimed to address the challenge of reducing sea-induced vibratory loads on floating offshore wind turbines, which can significantly impact structural integrity and maintenance costs. They developed a multi-layer control strategy combining resonant controllers targeting wave-induced loads and a PI controller for mitigating rotor-induced vibrations. The control system was validated through simulations on both 5 MW and 15 MW wind turbines mounted on spar and semi-submersible platforms. Results demonstrated substantial reductions in blade root flapping moments and rotor nacelle assembly loads, indicating potential for decreased maintenance requirements and extended component lifespans. While the study did not provide exact LCOE figures, the authors highlighted that such load reductions could contribute to lowering the LCOE [50].

Furthermore, Florian and Sørensen (2015) developed a fracture mechanics model to assess the remaining life of wind turbine blades, focusing on crack propagation in adhesive joints. The model integrates with a risk-based maintenance decision framework to optimise operation and maintenance (O&M) planning. Their analysis revealed that preventive maintenance, guided by regular inspections and timely repairs, significantly reduces the likelihood of catastrophic blade failures. Specifically, they found that corrective repairs, which involve blade replacement, are substantially more expensive and result in longer turbine downtimes compared to preventive interventions. By optimising inspection intervals and repair thresholds, the study demonstrated potential reductions in total O&M costs and associated revenue losses. These findings underscore the economic benefits of incorporating advanced control strategies and condition monitoring systems in floating offshore wind turbines [19].

A more quantitative method was done by Shafiee (2024), who introduces PESTLE, an analysis of political, economic, social, technological, legal and environmental factors, to build a six-stage decision framework for extending offshore wind-turbine life. He starts by identifying key degradation modes and estimating remaining useful life, then assesses structural integrity before carrying out economic and environmental evaluations. A comparison of three extension technologies (remanufacturing, retrofitting and reconditioning) is done. The results show that a five- to ten-year service-life extension can defer a $3–5 million-per-MW repowering cost, improve return on investment by up to 20 percent and reduce levelized cost of energy by roughly 5–15 percent [59].

More recently, Amlashi and Lotfizadeh (2025) present a probabilistic framework for assessing the levelized cost of energy in floating offshore wind farms, emphasising the significant impact of operational expenditures (OPEX) on LCOE variability. Their Monte Carlo simulations reveal that fluctuations in OPEX contribute more to LCOE uncertainty than variations in capital expenditures (CAPEX). Notably, increasing OPEX from 0.1 to 0.4 million EUR/MW/year substantially raises the probability of exceeding a characteristic LCOE threshold, whereas similar adjustments in CAPEX have a lesser effect [6].

These studies collectively highlight the significant economic benefits of extending the life of wind turbines through advanced control strategies, maintenance optimisation, and structural health monitoring. By reducing the need for costly replacements and deferring major repowering expenses, these strategies can improve return on investment, reduce maintenance costs, and lower the LCOE. Meaning that the integration of these approaches would surely be a significant improvement.

2.6. Research Gap and Justification

The primary objective of this thesis is to integrate key findings from various research areas, specifically PI control, annual energy production, fatigue life assessment, Pareto optimisation, and the economic aspects of floating wind turbines, into a cohesive framework. Although individual studies have explored these topics separately, there is a significant gap in the literature regarding their combined application to optimise control strategies for floating wind turbines. Furthermore, research into Pareto optimisation for simultaneously balancing blade fatigue life and AEP remains limited.

This work aims to fill this gap by investigating the ROSCO PI control strategy, focusing on tuning the controller to test these two conflicting objectives. By applying Pareto front analysis, this research will compare various gain-scheduled controller settings and benchmark them against the baseline ROSCO performance, all done with site-specific data, to create realistic simulation scenarios. This way, the research builds on concepts that exist in literature, but stands out as a novel approach.

The hypothesis is that, while prioritising blade fatigue life may lead to a reduction in power output, the resultant extension of blade life will ultimately contribute to greater cumulative energy production and an improved overall business case for floating wind turbines. By integrating control gain tuning with comprehensive metocean data, fatigue analysis, AEP analysis, and LCOE calculation, this research provides an innovative approach that has not been explored in the current literature.

3

Theoretical Background

This chapter provides the theoretical background for the study, covering the offshore environment, PI control fundamentals, and methods for controller tuning. The key performance metrics, such as annual energy production and blade fatigue life, are introduced, along with the concept of Pareto optimisation. The chapter concludes with an overview of the levelised cost of energy as an economic assessment tool.

3.1. The Offshore Environment

Floating offshore wind turbines are subjected to complex and often severe environmental conditions, where both wind and wave dynamics play a critical role in system performance and structural loading. To establish a solid foundation for subsequent analysis, it is essential to first explore the theoretical principles governing these environmental forces. This section will examine the key characteristics of wind and wave conditions and their spectral representations.

3.1.1. Wind

Wind originates from pressure gradients in the Earth's atmosphere, which are primarily driven by the uneven heating of the planet's surface by solar radiation. Since warm air is less dense than cooler air, it rises, prompting air from high-pressure regions to move toward low-pressure areas in an attempt to restore balance. This continuous redistribution of air is what we perceive as wind.

In the context of wind energy, offshore locations offer particularly favourable conditions. Over open water, the absence of physical obstacles such as trees and buildings allows the wind to flow more freely. As a result, offshore winds are typically stronger, more uniform, and less turbulent than their onshore counterparts, making them ideal for power generation [16, 49].

While understanding global wind behaviour helps grasp the overall resource potential, site-specific wind characteristics are critical for designing and operating a wind turbine effectively. Especially in load modelling and control optimisation, detailed knowledge of local wind conditions is essential. One such local phenomenon is the vertical wind profile, shown in Figure 3.1.

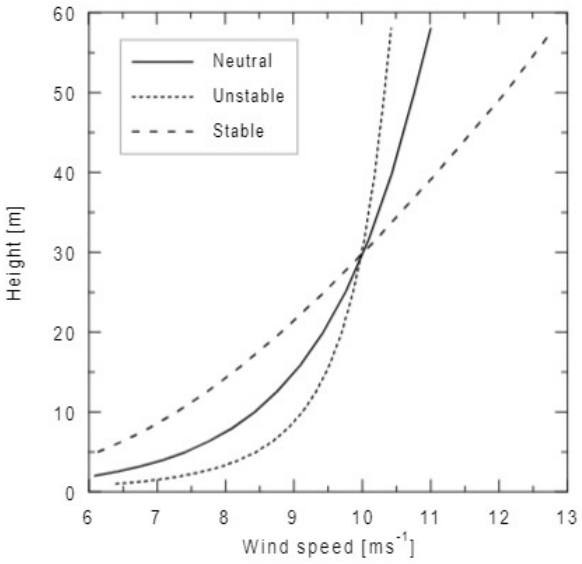


Figure 3.1: Vertical wind profile for neutral, unstable, and stable conditions [49].

This variation in wind speed with altitude is commonly known as wind shear. Accurately capturing this behaviour is essential not only for assessing turbine placement but also for estimating aerodynamic loads during the design phase. When wind measurements are only available at a single height, theoretical profiles can be employed to estimate wind speeds at other elevations. For neutrally stable atmospheric conditions, the logarithmic wind profile shown in Equation 3.1 is typically used, where u^* represents the friction velocity, κ is the von Kármán constant, and z_0 is the surface roughness length.

$$U = \frac{u_*}{\kappa}\left[\ln\frac{z}{z_0}\right] \tag{3.1}$$

By using a known reference wind speed, U_{ref}, at a certain height, z_{ref}, this relationship can be reformulated to estimate wind speeds at any other height, as shown in Equation 3.2.

$$\frac{U}{U_{ref}} = \frac{\ln\frac{z}{z_0}}{\ln\frac{z_{ref}}{z_0}} \tag{3.2}$$

Alternatively, the power law is another empirical method often used to approximate vertical wind profiles, especially when limited data is available. As shown in Equation 3.3, this model introduces an exponent, α, which characterises the rate at which wind speed increases with height and typically ranges between 0.10 and 0.20 for offshore conditions [16, 35].

$$\frac{U}{U_{ref}} = \left(\frac{z}{z_{ref}}\right)^{\alpha} \tag{3.3}$$

In addition to wind shear, turbulence plays a critical role in turbine load assessments. It refers to irregular fluctuations in wind speed caused by thermal instabilities and terrain

variability. These include short-lived gusts that can induce significant, unpredictable loads on turbine components. Unlike the gradual variations captured in vertical profiles, turbulence manifests over shorter timescales, from seconds to minutes, and is typically characterised by the turbulence intensity, I_u, as defined in Equation 3.4. Here, σ_u is the standard deviation of the wind speed over a given time window, typically between 10 to 60 minutes with adequate sampling resolution.

$$I_u = \frac{\sigma_u}{U} \tag{3.4}$$

In offshore environments, turbulence intensity under neutral stability conditions is generally lower than onshore, often around 8% [49]. Nevertheless, it remains a key factor in turbine design and control, as it directly impacts fatigue loading and operational reliability.

With a solid understanding of wind behaviour, the next step is to model its variability over time. Because wind is inherently unsteady, effective analysis requires methods that capture its dynamic nature. One such method is the wind energy spectrum, which describes how wind energy is distributed across different frequencies. Among the various spectral models available, the Kaimal spectrum is widely adopted for simulating turbulent wind fields in engineering applications [8]. In this thesis, the Kaimal model was chosen to represent the spectral distribution of longitudinal wind velocity fluctuations. It can be expressed as:

$$S_u(z,f) = \frac{105{u^*}^2 z/U(z)}{\left[1+33\left(\frac{fz}{U(z)}\right)\right]^{5/3}} \tag{3.5}$$

where:

- $S_u(z,f)$ is the power spectral density function for the longitudinal turbulent component.
- f is the frequency in Hz.
- u^* is the friction velocity, which is related to the standard deviation by: $\sigma^2 = 5.7{u^*}^2$.

These relationships are implemented in the wind modelling setup to accurately represent the incoming wind conditions, thereby improving the stability and reliability of the simulation results.

3.1.2. Waves

Understanding wave behaviour is fundamental for modelling the hydrodynamic forces acting on floating offshore wind turbines. Waves play a critical role in the dynamic response of floating structures, influencing motions, mooring loads, and fatigue life. Therefore, a theoretical understanding of wave mechanics is essential before performing any detailed analysis.

To begin, defining the key terminology that describes ocean waves is useful, as these foundational concepts are used throughout wave modelling and analysis [57]. A visual overview of typical wave characteristics is provided in Figure 3.2.

- **Wave Height (H):** The vertical distance between the wave crest and trough. This is a key indicator of wave energy and potential structural impact.
- **Wavelength (λ):** The horizontal spacing between successive crests or troughs, giving insight into wave propagation.
- **Wave Period (T):** The time interval between two passing wave crests at a fixed location. It determines the wave frequency and is central to spectral analysis.
- **Wave Frequency (f):** Defined as $f = \frac{1}{T}$, it quantifies how often waves pass a point per unit time.
- **Radian Frequency (ω_w):** Given by $\omega = 2\pi f = \frac{2\pi}{T}$, this angular form is preferred in spectral models.
- **Wave Speed (c_w):** Also known as phase speed, calculated as $c_w = \frac{\lambda}{T}$, indicating how quickly a wave crest travels.
- **Amplitude (a_w):** Half the wave height, corresponding to the wave's maximum displacement from the still water level.
- **Still Water Level (SWL):** The calm sea surface without wave action, used as a reference elevation.
- **Free Surface Elevation (η):** The instantaneous deviation of the sea surface from the SWL, varying with time due to wave motion.

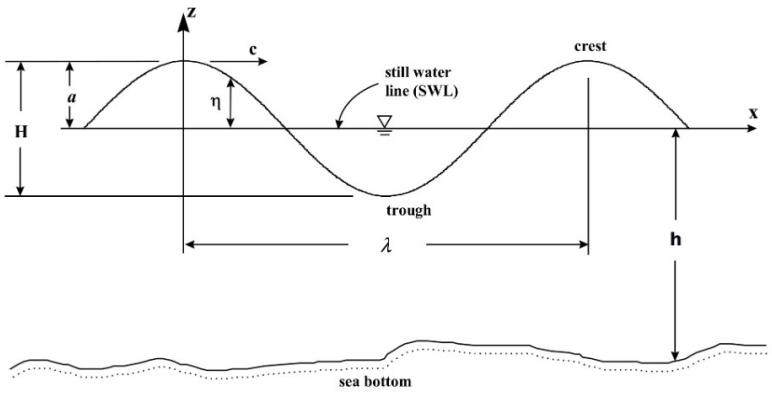


Figure 3.2: Regular traveling wave with wave characteristics [3].

In reality, ocean waves do not follow regular patterns but are instead irregular and random in nature. To capture this complexity, wave spectra are employed. A wave spectrum provides a statistical description of how wave energy is distributed over frequencies, allowing for the modelling of realistic sea states with variable wave heights, directions, and periods.

Among the spectral models used in ocean engineering, the JONSWAP spectrum is particularly prominent and used in the subsequent simulations [64]. The spectrum is defined as:

$$S(\omega) = \alpha \frac{g^2}{\omega_w^5} \exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega_w}\right)^4\right] \gamma^{\exp\left[-\frac{1}{2}\left(\frac{\omega_w - \omega_p}{\sigma\omega_p}\right)^2\right]} \tag{3.6}$$

In this formulation, α is the Phillips constant, ω_p the peak frequency, γ the peak parameter, and σ a spectral width parameter. This model assumes that sea surface elevation can be treated as a zero-mean Gaussian process composed of sinusoidal components with random phases and amplitudes. Such modelling enables simulations that more accurately represent the complex interaction between waves and offshore structures.

Key derived parameters from the spectrum, which will be used to set up the simulation scenarios, include:

- **Significant Wave Height (H_s):** A statistical measure approximating the average height of the highest third of waves in a given sea state. It is related to the zeroth spectral moment m_0 by:

$$H_s = 4\sqrt{m_0}, \quad \text{where} \quad m_0 = \int_0^\infty S(\omega_w)\, d\omega_w \tag{3.7}$$

- **Peak Period (T_p):** The period corresponding to the spectral peak frequency ω_p. This represents the dominant wave period and is calculated as:

$$T_p = \frac{2\pi}{\omega_p} \tag{3.8}$$

By incorporating these spectral models, of which a power spectral density plot can be seen in Figure 3.3, it is possible to simulate a wide range of wind and wave conditions and evaluate the performance and durability of floating wind turbines under realistic environmental loading.

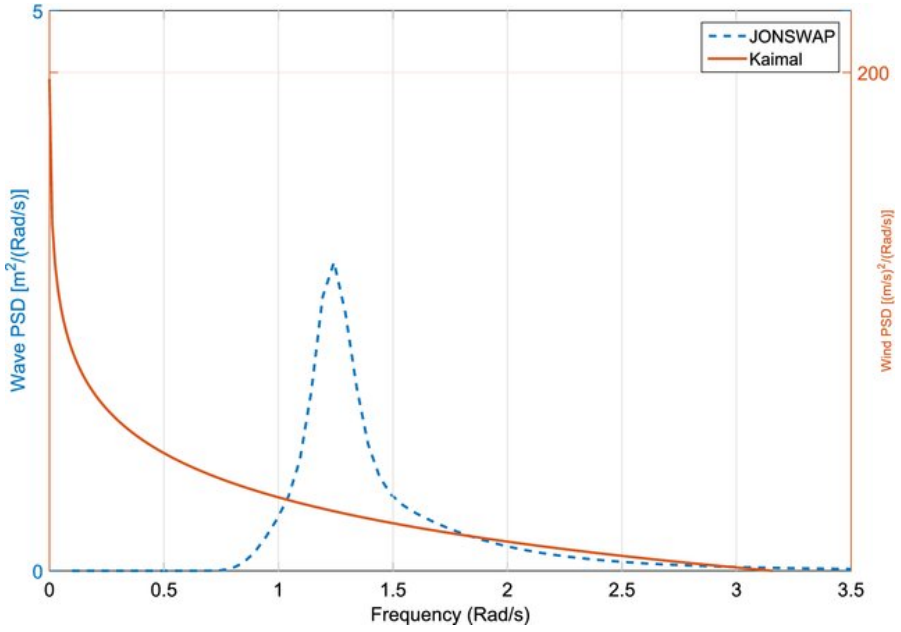


Figure 3.3: Comparison of wind (Kaimal) and wave (JONSWAP) power spectral densities [26].

3.2. Control Technology in Wind Energy

The control of wind turbines plays a crucial role in ensuring efficient power extraction, structural integrity, and operational reliability across a wide range of wind and wave conditions. This section presents the theoretical foundation underlying wind turbine control systems, beginning with the primary control objectives, maximising power capture below rated wind speeds and limiting loads and power above rated speeds. Generator torque control and blade pitch control are examined in detail, as well as approaches for tuning proportional-integral controllers to meet these varying objectives. In addition to foundational methods, further modifications are discussed to improve overall functionality and facilitate more specific tuning. The theory on this subject is specific to the ROSCO controller and therefore largely based on its documentation [1].

3.2.1. Control Objectives and Strategies

Current large-scale wind turbines mostly operate using a ”variable speed, variable pitch” control strategy, meaning that rotor speed and blade pitch angle are actively adjusted throughout the operational range to optimise performance and efficiency. As shown in Figure 3.4, different control strategies apply across distinct wind speed regions. Region 1, which lies below the cut-in wind speed, is characterised by no power production. In Region 2, between the cut-in and rated wind speeds, the objective is to maximise energy capture. Here, rotor speed increases approximately linearly with wind speed, generator torque increases quadratically, and power output grows cubically, while the pitch angle remains nearly constant. In Region 3, from the rated wind speed to the cut-out speed, both torque and pitch control are active. The aim here shifts from maximising energy to limiting loads and maintaining operation at rated power.

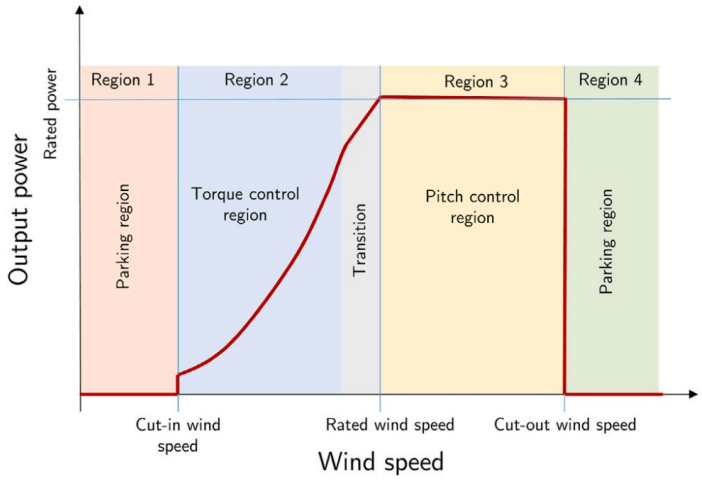


Figure 3.4: Operational regions of the wind turbine based on wind speed [68].

To be specific about ROSCO, the strategy is a PI control strategy. Proportional-Integral control is one of the most widely used control strategies in industrial applications, including wind turbine control. It is designed to regulate system outputs by adjusting control inputs based on both the instantaneous error and the accumulated past error. The PI controller consists of two main components:

- **Proportional term**: Directly responds to the instantaneous error, providing an immediate corrective action proportional to the magnitude of the deviation.
- **Integral term**: Accounts for the accumulated error over time, ensuring the elimination of steady-state errors and improving long-term accuracy.

In general, the control law can be mathematically expressed as:

$$y = K_P u + K_I \int_0^T u dt, \tag{3.9}$$

where u is an input to the controller, y is the output from the controller passed to the wind turbine, and K_P and K_I are the proportional and integral gains, respectively.

3.2.2. Generator Torque Control

As previously discussed, generator torque control is most prominent in Region 2, where the turbine operates below rated wind speeds. The primary objective in this region is to maximise power production. To this end, it is useful to define the power coefficient, expressed as:

$$C_p = \frac{P}{\frac{1}{2}\rho\pi R^2 U^3} \tag{3.10}$$

where P is the extracted power, ρ is the air density, R is the rotor radius, and U is the incoming wind speed. The denominator represents the available power in the wind across the swept rotor area.

Maximising C_p involves maintaining an optimal tip speed ratio (TSR), as the power coefficient reaches its maximum at a specific value of TSR for each blade pitch angle. This relationship is illustrated in Figure 3.5. The tip speed ratio, λ, is defined as:

$$\lambda = \frac{\omega R}{U} \tag{3.11}$$

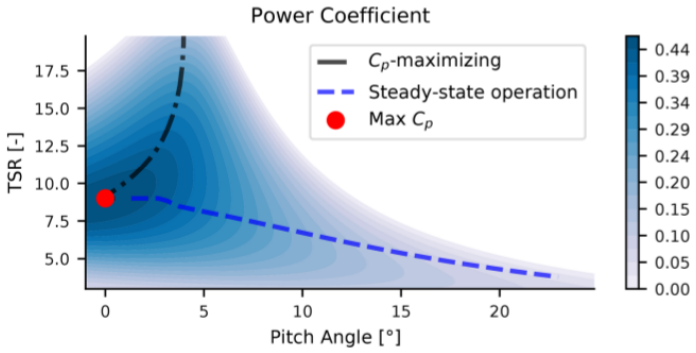


Figure 3.5: C_p surface for the IEA 15 MW wind turbine [1].

Therefore, the objective in this region is to maintain the optimal TSR to ensure maximum aerodynamic efficiency. Determining the corresponding torque is done by first recalling the definition of mechanical torque:

$$Q = \frac{P}{\omega} \tag{3.12}$$

Substituting the expression for power from the definition of C_p gives:

$$Q = \frac{C_p \frac{1}{2} \rho \pi R^2 U^3}{\omega} \tag{3.13}$$

Next, using the expression for wind speed in terms of TSR:

$$U = \frac{\omega R}{\lambda} \tag{3.14}$$

Substituting this into the torque equation results in a quadratic relationship between torque and rotor speed:

$$Q = \frac{C_p \frac{1}{2} \rho \pi R^5 \omega^2}{\lambda^3} \tag{3.15}$$

This shows that, under the assumption of a constant C_p and λ, the generator torque is approximately proportional to the square of the rotor speed in Region 2. This quadratic relationship forms the basis for the torque control strategy aimed at maximising energy capture in below-rated wind conditions.

To implement this control strategy in practice, the torque control law must be expressed in terms of generator-side quantities. The rotor and generator speeds are related by the gearbox ratio N_g as:

$$\omega = \frac{\omega_g}{N_g}, \tag{3.16}$$

and the generator torque τ_g is related to the rotor-side torque Q through the gearbox efficiency η_{gb}:

$$\tau_g = \frac{Q}{N_g \eta_{gb}}. \tag{3.17}$$

Substituting both expressions into the rotor torque equation yields:

$$\tau_g = \frac{1}{N_g \eta_{gb}} \cdot \frac{C_p \frac{1}{2} \rho \pi R^5 \left(\frac{\omega_g}{N_g}\right)^2}{\lambda^3} = \frac{C_p \frac{1}{2} \rho \pi R^5}{\lambda^3 N_g^3 \eta_{gb}} \omega_g^2. \tag{3.18}$$

This results in the following control law for generator torque:

$$\tau_g = K\omega_g^2, \tag{3.19}$$

where the constant K is defined as:

$$K = \frac{\pi\rho R^5 C_{p,max}}{2\lambda_{opt}^3 N_g^3 \eta_{\text{gb}}}. \tag{3.20}$$

This quadratic control law is commonly implemented in wind turbine controllers, including ROSCO, for optimal torque tracking in below-rated conditions.

In addition to the commonly used control approach known as the $K\omega^2$ law, ROSCO also implements a TSR tracking control strategy. This strategy relies on a PI controller, where the general PI expression (see Equation 3.9) is adapted to regulate the generator torque based on the deviation from a reference generator speed:

$$\Delta\tau_g = K_P\left[\omega_{g,\text{ref}} - \omega_g(t)\right] + K_I \int_0^T \left[\omega_{g,\text{ref}} - \omega_g(t)\right] dt, \tag{3.21}$$

where $\omega_g(t)$ is the instantaneous generator speed, and $\omega_{g,\text{ref}}$ is the reference generator speed, defined as:

$$\omega_{g,\text{ref}} = N_g \frac{\lambda_{\text{opt}}\hat{v}}{R}, \tag{3.22}$$

with $\hat{v}$ representing the estimated rotor-effective wind speed provided by the wind speed estimator. The aim of this control strategy is to maintain the optimal tip speed ratio λ_{opt}, thereby maximising the aerodynamic efficiency of the turbine in below-rated conditions.

In above-rated wind conditions, the objective of the control strategy shifts from maximising energy capture to limiting mechanical loads and maintaining rated power output. This is achieved by regulating generator torque based on either a constant torque or constant power strategy. The generator torque setpoint $\tau_{g,\text{ar}}(t)$ is defined as:

$$\tau_{g,\text{ar}}(t) = \begin{cases} \dfrac{P_{\text{rated}}}{\omega_{g,\text{rated}}}, & \text{(constant torque)} \\ \dfrac{P_{\text{rated}}}{\omega_g(t)}, & \text{(constant power).} \end{cases} \tag{3.23}$$

In both strategies, the generator torque is no longer governed by the $K\omega_g^2$ law used in Region 2, but rather directly constrained to maintain rated electrical output. Transition to above-rated operation typically occurs when the blade pitch exceeds a predefined threshold, indicating that aerodynamic control has become active to regulate rotor speed.

If TSR tracking is used below rated, the transition to above-rated operation involves capping the generator torque such that $\tau_g(t) \leq \tau_{g,\text{ar}}(t)$, ensuring the turbine does not exceed its rated capacity. A setpoint smoother is often used to gradually shift the reference generator speed, enabling the torque to saturate at its limit while reducing transient loads.

3.2.3. Blade Pitch Control

As previously stated, when transitioning into Region 3, the blade pitch controller becomes active. This PI controller determines the collective blade pitch angle to maintain the rotor speed at a reference value, straightforwardly defined as the rated generator speed. The control law can be expressed as:

$$\Delta\beta = K_P \left[\omega_{\mathrm{g,rated}} - \omega_g(t)\right] + K_I \int_0^T \left[\omega_{\mathrm{g,rated}} - \omega_g(t)\right] \, dt \tag{3.24}$$

The controller must account for physical constraints such as maximum pitch angle, rate limits of the pitch actuators, and mechanical tolerances, which can influence the stability and responsiveness of the control system. Additionally, the effectiveness of the blade pitch PI controller is highly dependent on the appropriate selection of its gains.

3.2.4. Controller Tuning

To tune the generator torque and blade pitch controllers, a mathematical model of the wind turbine system, known as the plant model, is required. This model allows for the derivation of controller gains based on system dynamics. While a full wind turbine model is nonlinear and high-dimensional, a simplified first-order representation can be used to capture the dominant dynamics for controller design purposes.

A commonly used approximation for the rotational dynamics of the drivetrain is:

$$\dot{\omega}_g = \frac{N_g}{J} \left(\tau_a - N_g \tau_g \eta_{gb}\right), \tag{3.25}$$

In this model formulation, J is the total rotational inertia of the turbine, and the term τ_a represents the aerodynamic torque and is given by the formula below. Here, $A_r = \pi R^2$ is the rotor swept area.

$$\tau_a = \frac{1}{2} \rho A_r \frac{C_p(\lambda, \beta)}{\omega} U^3. \tag{3.26}$$

The first-order linearization of this equation at a steady-state operational point is then obtained.

$$\Delta\tau_a = \Gamma_{\omega_g}\Big|_{op} \Delta\omega_g + \Gamma_\beta\Big|_{op} \Delta\beta + \Gamma_U\Big|_{op} \Delta U, \tag{3.27}$$

here "*op*" refers the operational value for ω_g, β, and U at which linearization occurs. Additionally,

$$\Gamma_{\omega_g} = \frac{\partial\tau_a}{\partial\omega_g}, \; \Gamma_\beta = \frac{\partial\tau_a}{\partial\beta}, \text{ and, } \Gamma_U = \frac{\partial\tau_a}{\partial U}. \tag{3.28}$$

Now, Equation 3.25 can be rewritten with this linearised form in mind. This results in the plant model used to tune the controllers and is denoted by,

$$\Delta\dot{\omega}_g = A(U_{op})\Delta\omega_g + B_{\tau_g}\Delta\tau_g + B_\beta(U_{op})\Delta\beta + B_U(U_{op})\Delta U, \tag{3.29}$$

where,

$$A(U_{op}) = \frac{N_g}{J}\frac{\partial\tau_a}{\partial\lambda}\frac{\partial\lambda}{\partial\omega_g} \tag{3.30}$$

$$\frac{\partial\tau_a}{\partial\lambda} = \frac{1}{2}\rho A_r R U^2 \frac{1}{\lambda_{op}^2}\left(\frac{\partial C_p}{\partial\lambda}\lambda_{op} - C_{p,op}\right) \tag{3.31}$$

$$\frac{\partial\lambda}{\partial\omega_g} = \frac{1}{N_g}\frac{R}{U_{op}} \tag{3.32}$$

This linearised model of the plant serves as the basis for independently tuning the controllers in both above-rated and below-rated operating regions. It is important to note that during the tuning process, the disturbance input matrix B_U, associated with wind speed variations, is assumed to be zero.

In order to use this to tune the controller however, it is important to recall the expressions for the general PI control law in Equation 3.9, the specific control laws for generator torque and blade pitch in Equation 3.21 and Equation 3.24, respectively, and the linearized model in Equation 3.29. Combining these equations makes it possible to obtain a differential equation that relates $\Delta\omega_{g,ref}$ and $\Delta\omega_g$. By subsequently taking the Laplace transform and setting $\Delta\omega_{g,ref} = 0$, the transfer function of the closed-loop system results in,

$$H(s) = \frac{\Delta\omega_g(s)}{\Delta\omega_{g,ref}(s)} = \frac{B(K_P(U_{op})s + K_I(U_{op}))}{s^2 + (BK_P(U_{op}) - A(U_{op}))s + BK_I(U_{op})}, \tag{3.33}$$

This second-order system allows for a definition of the proportional and integral gains as,

$$K_P(U_{op}) = \frac{1}{B}(2\zeta_{des}\omega_{des} + A(U_{op})) \tag{3.34}$$

$$K_I(U_{op}) = \frac{\omega_{des}^2}{B}, \tag{3.35}$$

here the gains are defined as a function of the desired damping ratio ζ_{des} and the desired natural frequency ω_{des}. By modifying these parameters, the response and therefore the performance, given a certain objective, can be analysed.

In the abovementioned equations, B can be B_{τ_g} or B_β depending on the controller that is being tuned. So, in the event that the generator torque controller is tuned for the TSR tracking torque controller in the below rated wind speed region and with the assumption that $\Delta\beta = 0$,

$$B = B_{\tau_g} = \frac{-N_g^2}{J} \tag{3.36}$$

Similarly, the gains can be found for the collective blade pitch controller. But in this case, the tuning process has added complexity due to the fact that optimal gains vary across the whole wind speed range. This calls for a set of gains that are clearly defined for a variety of operational conditions. This is called gainscheduling, and it has been shown that this improves blade pitch controller performance. In the ROSCO workflow, the C_p surface seen in Figure 3.5 is used for this.

For the scenario where constant torque is selected in above rated wind conditions, there are no changes to the formulation of $A(U_{op})$ from Equation 3.30. If constant power is the chosen strategy, $A(U_{op})$ is rewritten as,

$$A(U_{op}) = \frac{1}{J}\frac{\partial \tau_a}{\partial \lambda}\frac{\partial \lambda}{\partial \omega_g} - B_{\tau_g}\frac{P_{rated}}{\omega_{g,rated}^2}. \tag{3.37}$$

In both instances, B is redefined as,

$$B = B_\beta(U_{op}) = \frac{N_g}{J}\frac{\partial \tau_a}{\partial \beta} = \frac{N_g}{2J}\rho A_r R U_{op}^2 \frac{1}{\lambda_{op}^2}\left(\frac{\partial C_p}{\partial \beta}\bigg|_{\lambda_{op},\beta_{op}} \lambda_{op}\right). \tag{3.38}$$

The relationship between the rated power coefficient and the operational TSR is given by

$$C_{p,op} = C_{p,rated}\left(\frac{\lambda_{op}}{\lambda_{rated}}\right)^3. \tag{3.39}$$

Using this expression along with the C_p surface, the corresponding steady-state blade pitch angles $\beta_{op}(\lambda_{op})$ can be determined for any given tip-speed ratio. During steady-state operation above rated wind speed, the generator speed is typically constant, making the TSR a function of wind speed alone. This allows us to define $\beta_{op}(U)$, and consequently express $A(U_{op})$ and $B_\beta(U_{op})$ as $A(\beta_{op})$ and $B_\beta(\beta_{op})$ for controller tuning purposes. Accordingly, the blade pitch controller gains become $K_P(\beta_{op})$ and $K_I(\beta_{op})$, allowing the gain schedule to be implemented as a function of blade pitch angle rather than estimated wind speed.

3.2.5. Additional Control Modifications

With the core control theory now addressed, it is valuable to explore several additional capabilities of the ROSCO controller. These modifications serve two main purposes. First, while the primary control strategy consists of the generator torque controller in Region 2 and the blade pitch controller in Region 3, the transition between these two regions presents opportunities for refinement. Second, since this controller will be applied to a floating offshore wind turbine FOWT, it is necessary to account for the dynamics introduced by the platform's motion.

Peak Shaving

To improve controller performance near the rated wind speed, a peak shaving, or thrust limiting, strategy can be implemented. This approach targets the region where thrust loads are typically highest, aiming to reduce structural loading by slightly curtailing power production. While this alleviates the loads caused by the thrust, it comes at the cost of reduced power, which is a trade-off that will be important.

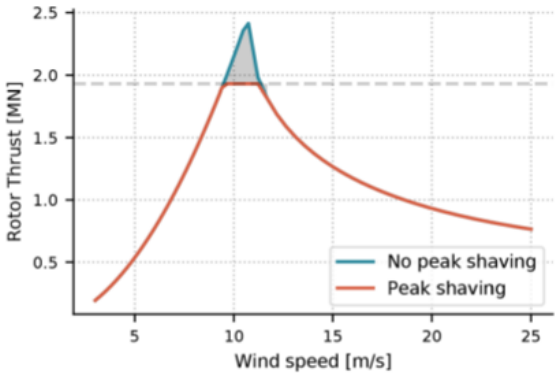


Figure 3.6: Rotor thrust with and without peak shaving for the IEA 15 MW wind turbine [1].

This is done by making use of the equation for rotor thrust, which is:

$$T_r(U) = \frac{1}{2}\rho A_r U^2 C_t(\lambda, \beta). \tag{3.40}$$

So when a limit on the thrust is imposed, commonly as a percentage reduction of the peak thrust, a blade pitch angle can be found for each TSR through a C_t surface comparable to the C_p surface.

Setpoint Smoothing

Similar to peak shaving, setpoint smoothing aims to improve the transition between the generator torque and blade pitch controllers around rated wind speed. By gradually adjusting control setpoints, this method reduces abrupt changes and conflicting control actions, leading to smoother operation and lower transient loads.

This is especially important in the transition zone, where both controllers may become active. Without smoothing, the torque controller and the blade pitch controller might simultaneously attempt to control the generator speed, resulting in instability or increased mechanical stress.

To mitigate this, setpoint smoothing works by dynamically shifting the generator speed reference of the saturated controller, typically the torque controller, away from the actual operating point. This limits it from reacting while the blade pitch controller takes over control. In practice, this means that once the pitch controller begins to act, the generator speed reference for the torque controller is increased slightly, pushing it further into saturation. This ensures that only one controller influences the generator speed at a time.

Mathematically, this is done by defining an offset of the rotor speed set point as:

$$\Delta\omega = \left[\underbrace{\left(\frac{\beta - \beta_{\min}}{\beta_{\max}} \right)}_{\Delta\beta} k_{vs} - \underbrace{\left(\frac{\tau_{g,\max} - \tau_g}{\tau_{g,\max}} \right)}_{\Delta\tau} k_{pc} \right] \omega_{g,rated}. \tag{3.41}$$

In this formulation, k_{vs} and k_{pc} are dimensionless tuning parameters greater than zero, while β_{max} represents the blade pitch angle at the turbine's cut-out wind speed. The structure of the equation ensures that $\Delta\beta = 0$ during below-rated operation, and $\Delta\tau = 0$ during above-rated operation. To facilitate a smooth transition between the torque and pitch controllers, the reference generator speeds are adjusted using piecewise logic, presented below, based on the sign of $\Delta\omega$. This shift prevents the simultaneous activation of both controllers. An example of its effect, clearly demonstrating a clearer transition from the above to below rated region, is shown in Figure 3.7.

$$\omega_{ref,\tau} = \begin{cases} \omega_{ref,\tau} - \Delta\omega & \Delta\omega \geq 0 \\ \omega_{ref,\tau} & \Delta\omega < 0 \end{cases} \tag{3.42}$$

$$\omega_{ref,\beta} = \begin{cases} \omega_{ref,\beta} & \Delta\omega \geq 0 \\ \omega_{ref,\beta} - \Delta\omega & \Delta\omega < 0 \end{cases} \tag{3.43}$$

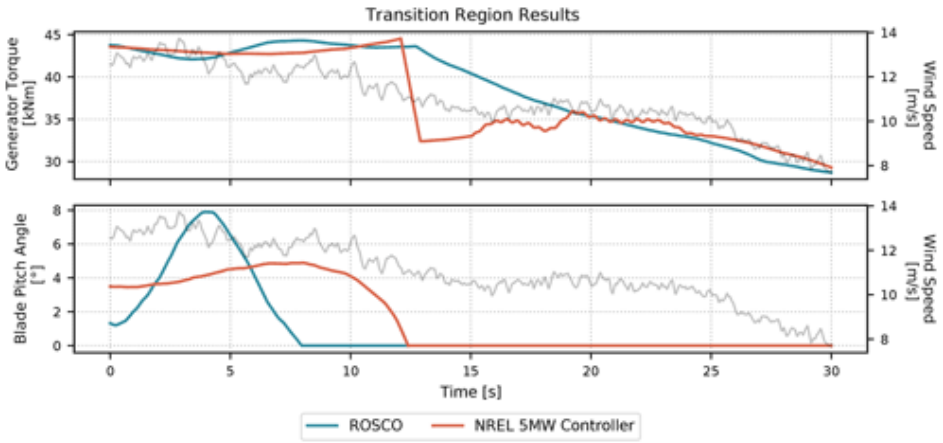


Figure 3.7: Example of the setpoint smoothing strategy in the near rated wind speed region [1].

Floating Specific Control Strategy

In floating offshore wind turbines, the motion of the platform introduces additional dynamics that can interfere with standard control strategies, especially in above-rated wind conditions. To address this, the ROSCO controller incorporates a compensation mechanism that enhances pitch control by reacting to the movement of the platform itself.

Specifically, the controller adds a feedback term based on the nacelle fore-aft velocity, denoted $\Delta\dot{x}_n$, into the blade pitch control loop. This allows the system to counteract disturbances caused by platform motion more effectively. The traditional PI control law for blade pitch is thus extended as follows:

$$\Delta\beta = K_P \left[\omega_{\mathrm{g,rated}} - \omega_g(t)\right] + K_I \int_0^T \left[\omega_{\mathrm{g,rated}} - \omega_g(t)\right] \, dt + K_{\beta_{float}} \Delta\dot{x}_n. \tag{3.44}$$

where $K_{\beta_{float}}$ is the gain applied to the platform motion feedback.

To ensure that the floating feedback does not interfere with structural resonance frequencies, ROSCO applies a series of filters. A high-pass filter with a cutoff frequency of 0.01 rad/s removes low-frequency drift, while a low-pass filter targets frequencies above the platform's first fore-aft natural frequency. Additionally, a notch filter is implemented to suppress the tower's fore-aft eigenfrequency component from the feedback signal. These filters collectively ensure that only relevant dynamic content from the platform motion is fed back into the pitch controller.

A Bode plot of the complete filter configuration, seen in Figure 3.8, illustrates how these elements shape the feedback response and isolate the desired frequency range for improved stability in floating applications.

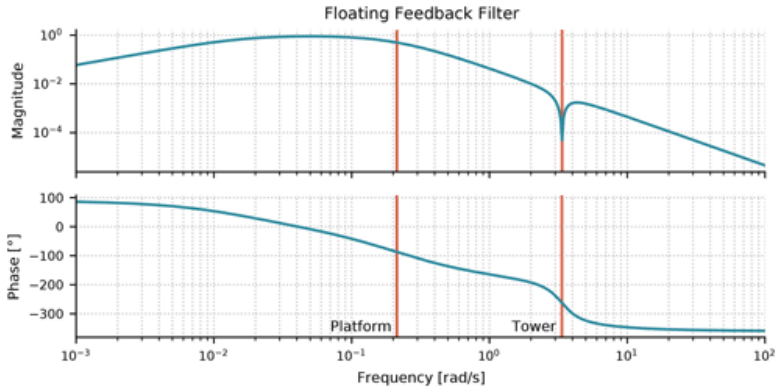


Figure 3.8: Bode plot showing the filters used for the IEA 15 MW turbine mounted on the UMaine floater [1].

3.3. Annual Energy Production

The annual energy production is a key metric in wind turbine performance evaluation, representing the expected yearly energy output based on site-specific wind conditions and turbine characteristics. In essence, the goal is to maximise this value, as this will result in greater economic viability.

To calculate AEP, the wind conditions have to be considered. Since wind speeds vary throughout the year, their probability of occurrence is typically modelled using the Weibull distribution:

$$f(U) = \frac{k}{c} \left(\frac{U}{c}\right)^{k-1} e^{-(U/c)^k} \tag{3.45}$$

where k is the shape parameter and c is the scale parameter. This distribution is flexible and well-suited to representing wind patterns at both onshore and offshore sites. Generally, a higher k indicates less variability in wind speeds, while c reflects the characteristic wind speed of the site.

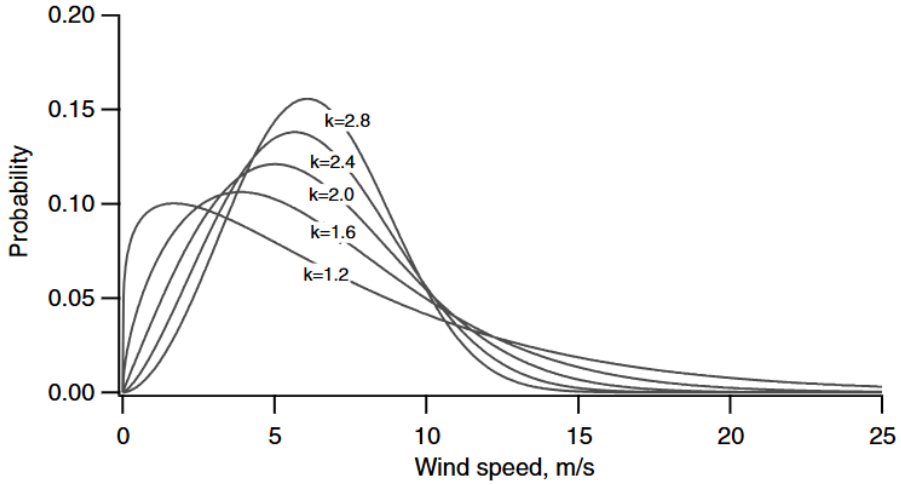


Figure 3.9: Example of a Weibull probability density function with different shape factors [35].

The average power output is computed by integrating the product of the power curve and the wind speed probability density function:

$$\bar{P} = \int_0^{U_{\text{max}}} P(U) f(U) dU \tag{3.46}$$

Here, $P(U)$ is the turbine's power output at wind speed U represented by a power curve as in Figure 3.4, and $f(U)$ is the probability of occurrence of that wind speed. In some analytical approaches, mechanical or electrical efficiencies may be included in $P(U)$. However, simulation tools such as SIMA already incorporate these effects in the model configuration.

Finally, the AEP is determined by multiplying the average power by the number of hours in a year:

$$AEP = \bar{P} \times 8760 \tag{3.47}$$

This methodology provides a realistic and probabilistic estimate of annual energy output and is widely used in industry and academic assessments of wind turbine performance [35].

3.4. Blade Fatigue Life Estimation

Another element of this research is evaluating the controller's influence on the blade fatigue life, therefore, it is necessary to review the theory used on the subject. Blade fatigue life estimation involves converting time-varying loads into damage predictions using a sequence of theoretical steps, seen in Figure 3.10. The sequence involves stress calculation, followed by cycle counting, S–N curve application, and finally, damage accumulation.

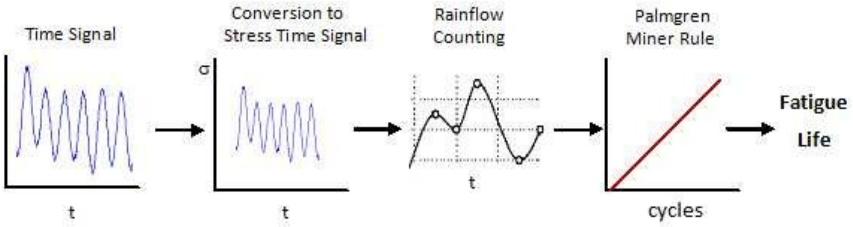


Figure 3.10: Rainflow counting method for fatigue life estimation [54].

3.4.1. Stress Calculation

At the blade root, stresses arise due to bending and torsion from the applied moments in all three directions. These must be combined into a single equivalent stress to evaluate fatigue and structural loading.

In chapter 4, it will be shown that the blade root has a thin-walled circular cross-section. For such a section, the computations to obtain the stresses are as follows [7, 10].

First, the total flap (z-axis) moment is composed of two primary contributions:

$$M_{z,\mathrm{tot}}(t) = M_{z,\mathrm{aero}}(t) + M_{z,\mathrm{grav}}(t), \tag{3.48}$$

where $M_{z,\mathrm{aero}}(t)$ is the aerodynamic moment about the flapwise axis, and $M_{z,\mathrm{grav}}(t)$ is the gravity-induced moment caused by the blade's weight during rotation.

The gravity moment is known to fluctuate periodically with the rotational speed of the rotor (often referred to as the 1P frequency, i.e., once-per-revolution), and is expressed as:

$$M_{z,\mathrm{grav}}(t) = m_{\mathrm{blade}}\, g\, r_{\mathrm{cg}}\, \sin(\Omega t), \tag{3.49}$$

where m_{blade} is the blade mass, g is gravitational acceleration, $r_{\mathrm{cg}} = \frac{2}{3}R_{\mathrm{blade}}$ is approximately the distance from the blade root to its center of gravity, and Ω is the rotor's angular velocity (in rad/s), derived from generator speed and gearbox ratio.

This additional moment only appears in the flapwise direction because the weight of the blade, when rotating in the horizontal plane, causes a sinusoidal variation in the flapwise moment as the blade rotates through different angular positions. This effect is not present in the edgewise direction (y-axis) and is negligible in torsion for current purposes.

The stresses at the outer fibre of the root cross-section are then calculated using:

$$\sigma_z(t) = \frac{M_y(t)\, R}{I}, \quad \sigma_y(t) = \frac{M_{z,\mathrm{tot}}(t)\, R}{I}, \tag{3.50}$$

where $M_y(t)$ and $M_z(t)$ are the time-varying edgewise and flapwise bending moments, respectively, R is the outer radius of the cross-section, and I is the second moment of area, also referred to as moment of inertia, for bending.

For a thin-walled circular ring, the second moment of area is:

$$I = \frac{\pi}{4}\left(R^4 - (R-t)^4\right) \approx \pi R^3 t, \tag{3.51}$$

where $t \ll R$ is the wall thickness. The polar moment of inertia is approximately:

$$J_p = I_x + I_y = 2I. \tag{3.52}$$

The torsional stress from the twisting moment $M_x(t)$ is:

$$\tau(t) = \frac{M_x(t)\, R}{J_p}. \tag{3.53}$$

Since stresses occur in multiple directions simultaneously, a scalar equivalent stress is needed to assess material yield or fatigue. The von Mises stress is a widely used criterion for this purpose, combining normal and shear stresses into a single expression:

$$\sigma_{vm}(t) = \sqrt{\sigma_y^2(t) + \sigma_z^2(t) + 3\,\tau^2(t)}. \tag{3.54}$$

This scalar value can then be directly compared to material properties such as uniaxial fatigue limits or yield strengths, which are typically defined under simple loading conditions.

3.4.2. Rainflow Cycle Counting

The rainflow counting method is a widely used technique for identifying and quantifying fatigue cycles in variable-amplitude loading conditions. It is particularly useful for analysing stress-time histories in structural and mechanical components subjected to fluctuating loads. The method works by decomposing a complex load history into individual stress reversals that contribute to fatigue damage.

In practice, the Python code, which is called the `rainflow` module, that would be used to execute the rainflow counting scans through the list of turning points, matches each peak with the next valley (and vice versa) that “encloses” it, and removes that pair from further consideration once it forms a full cycle, this procedure can be seen in Figure 3.11. Any remaining unpaired peaks or valleys at the ends of the record count as half-cycles.

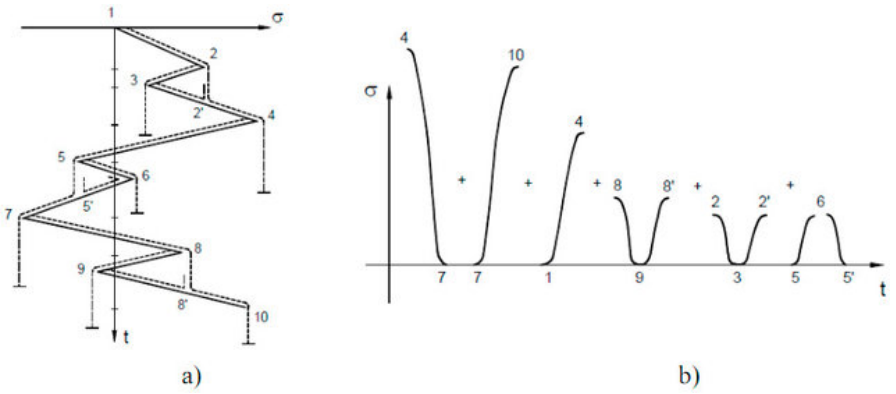


Figure 3.11: Rainflow counting method showing the stress history and the equivalent cycles [65].

By focusing on closed loops, which correspond to true fatigue cycles defined by their stress range and mean level, the method filters out minor, non-damaging fluctuations.

By systematically extracting these cycles, the rainflow algorithm allows for accurate fatigue life prediction using damage accumulation models such as the Palmgren-Miner rule. The rainflow method is preferred over simpler cycle-counting techniques, as it provides more realistic fatigue load characterisation by preserving load sequence effects, thereby improving structural integrity assessments [15].

3.4.3. S−N Curve Utilisation

In the context of floating offshore wind turbines, components such as the blade root experience repeated cyclic loading over millions of cycles throughout their design life. These load fluctuations can lead to fatigue damage even if the peak stresses remain well below the ultimate tensile strength of the material. Since such structures are designed for high reliability and long service life, accurately assessing fatigue life is essential. The S–N curve provides a fundamental relationship between stress amplitude and fatigue life, illustrated in Figure 3.12. Using this allows for a quantification of how long a component can sustain a given load history before failure. This is critical for assessing control strategies aimed at load reduction, such as those developed in this study.

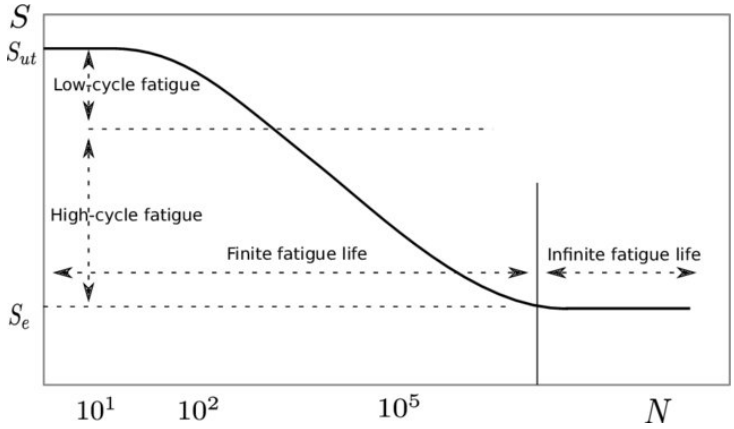


Figure 3.12: Example of a typical S-N curve [2].

High-cycle fatigue behaviour is often represented by the Basquin relation [55]. This can be expressed as:

$$S_a = a\, N^{-1/b}, \tag{3.55}$$

where S_a is the stress amplitude, N is the number of cycles to failure, a the strength coefficient, and b the fatigue exponent.

To incorporate the effect of non-zero mean stress in the load cycles, the Goodman correction defines an equivalent stress amplitude. The equation is commonly represented as a linear relationship between mean stress and alternating stress, defining the maximum allowable alternating stress a material can endure at a given mean stress before fatigue failure occurs [69].

$$S_{\text{eq}} = \frac{S_a}{1 - (\sigma_m / \text{UTS})}, \tag{3.56}$$

where UTS is the material's ultimate tensile strength. This equivalent amplitude can then be directly compared to the S–N curve to estimate fatigue damage accumulation.

3.4.4. Cumulative Damage

Subsequently, the use of the Palmgren-Miner Rule, commonly known as Miner's Rule, is necessary. This is a method used in fatigue analysis to predict the lifespan of materials subjected to varying stress levels. It operates on the principle of cumulative damage, allowing for an estimation of when a material is likely to fail under cyclic loading conditions [43]. The rule is mathematically represented as:

$$D = \sum_i \frac{n_i}{N_i} \tag{3.57}$$

In this equation, D is the total accumulated damage, n_i represents the number of cycles experienced at a specific stress level S_i, and N_i denotes the number of cycles to failure at that same stress level S_i, typically derived from S-N curves. According to Miner's Rule, failure is anticipated when the total damage D reaches or exceeds 1:

$$D \geq 1 \, . \tag{3.58}$$

In practice, Miner's Rule offers a straightforward approach to assess cumulative damage under variable amplitude loading. However, it is important to note that this rule assumes linear damage accumulation and does not account for load sequence effects, which can influence the actual fatigue life of materials.

3.5. Pareto Optimisation for Conflicting Objectives

In the design and control of wind turbines, it is a challenge to optimise multiple conflicting objectives. Two primary objectives are maximising power output and minimising fatigue loads on structural components such as blades. Improving one often leads to the detriment of the other.

Pareto optimisation offers a framework to address such conflicts by identifying a set of optimal trade-off solutions, known as the Pareto front, of which an example, fitting to this case, can be seen in Figure 3.13. A solution is considered Pareto optimal if no objective can be improved without compromising at least one other objective [39]. In this context, the Pareto front represents the set of control strategies where any attempt to reduce fatigue loads and thereby extend fatigue life will likely come at the cost of decreased power output, and vice versa.

Mathematically, this multi-objective optimisation problem can be formulated as:

$$\max_{\mathbf{k}} \quad \begin{cases} f_1(\mathbf{k}) = P(\mathbf{k}) \\ f_2(\mathbf{k}) = F(\mathbf{k}) \end{cases} \tag{3.59}$$

where:

- $\mathbf{k}$ represents the control inputs and gains based on the tuning.
- $P(\mathbf{k})$ is the power output resulting in the AEP.
- $F(\mathbf{k})$ is the fatigue life of the blades.

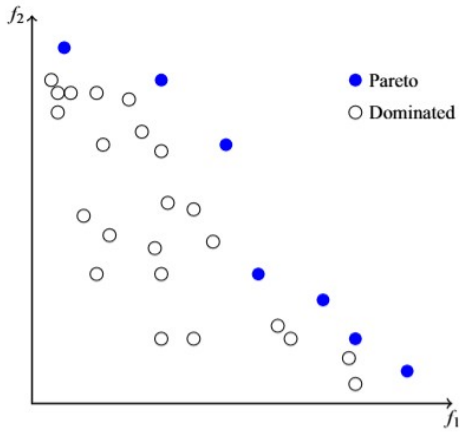


Figure 3.13: Pareto front illustrating the Max–Max compromise between two conflicting objectives [11].

By evaluating the performance of various tuning configurations and plotting their outcomes in objective space, a Pareto front is constructed. This allows for informed selection of controller settings that offer desirable trade-offs, such as a marginal reduction in AEP in exchange for substantial gains in fatigue life. However, the selection of the final optimal solution from the Pareto front ultimately depends on the decision maker's priorities and preferences, making it inherently subjective and difficult to define in absolute terms.

3.6. Assessing Economic Impact Through LCOE

The levelised cost of energy is a widely used metric to assess the cost of energy-producing technologies by combining capital expenditures, operational costs, and energy production into a single comparable figure. It is particularly valuable for evaluating trade-offs in controller design where changes in turbine behavior can affect both energy yield and fatigue life. The LCOE is given by:

$$LCOE = \frac{\sum_{i=1}^{n}(CAPEX + OPEX)(1+r)^{-i}}{\sum_{i=1}^{n} AEP(1+r)^{-i}}, \tag{3.60}$$

where n is the number of years that the project is active, CAPEX represents the capital expenditure (e.g., construction, equipment, installation), OPEX is the annual operational and maintenance cost, r is the discount rate, and AEP is the annual energy production in year i [36].

The LCOE reflects the average cost per unit of electricity generated over the turbine's lifetime, allowing for direct comparison between different control strategies. Although all controller tunings evaluated in this work use the same baseline control logic, the individual parameter settings, such as proportional-integral gains, setpoint smoothing, or peak shaving thresholds, can significantly affect both AEP and the expected turbine lifetime, which can alter n, the economic lifetime. Furthermore, fatigue mitigation can reduce $OPEX$ over time by lowering failure rates or extending the service intervals, making LCOE a relevant and sensitive metric for controller optimisation.

4

Modeling and Simulation Methods

This chapter outlines the methodology used to develop and evaluate the controllers for floating offshore wind turbines. It details the simulation framework, the model and performance evaluation metrics, ensuring transparency and reproducibility.

4.1. Simulation Framework

As this research is driven by simulation experiments, Figure 4.1 outlines the overall workflow. First, five key controller parameters were systematically varied and passed to the ROSCO toolbox, whose tuning routines generated a unique `DISCON.IN` file for each configuration. These controller files and the accompanying ROSCO dynamic library were then loaded into SIMA to perform time-domain simulations across a range of wind speeds and hydrodynamic conditions, each running for 660 s with a 0.01s timestep. SIMA produced time series of power output and blade root bending moments, which were exported to Excel and post-processed in Python to calculate annual energy production and expected fatigue life. These two metrics form the points of a Pareto front, illustrating the trade-off between energy yield and structural longevity. The generation of wind and wave scenarios is described in detail in section 4.5. Subsequent sections explain the SIMA environment and the ROSCO toolbox implementation.

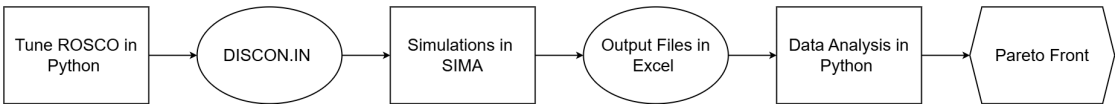


Figure 4.1: Workflow of the analysis.

4.1.1. SIMA

SIMA is a versatile modelling environment from SINTEF, designed to simulate and analyse (floating) marine structures, with specific extensions for offshore wind turbines [60]. It combines the SIMO module, which handles rigid-body motions under wind, wave, and current loading, with RIFLEX, which captures the dynamic response of flexible components such as mooring lines, risers, and blades. Users can seamlessly integrate control algorithms and realistic wind fields into the same framework, enabling full studies of

turbine performance and structural dynamics. The overall data flow and component interactions within SIMA are summarised in Figure 4.2.

As can be seen in the figure, SIMA supports importing three-dimensional wind fields generated by external tools such as TurbSim, which will be further discussed in section 4.5, as well as simpler 2D or uniform profiles directly within the workbench. Control algorithms can be linked seamlessly by implementing user-defined input files, which were the `DISCON.IN` files and dynamic libraries from ROSCO in this case. Once a simulation is complete, SIMA's integrated post-processing environment enables automated extraction of key metrics and data export, including Excel tables for further analysis.

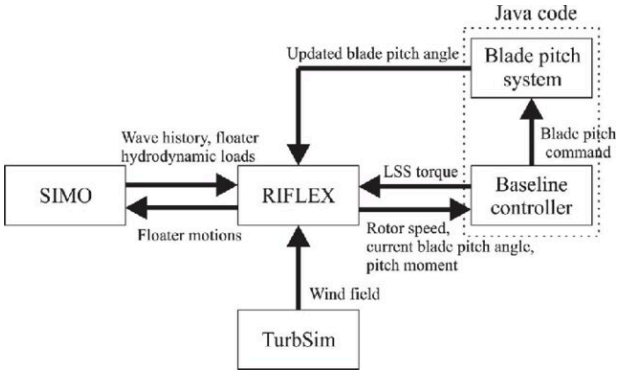


Figure 4.2: Data transmission scheme between SIMO, RIFLEX, and controller [14].

4.1.2. ROSCO Toolbox

The ROSCO toolbox provides a systematic framework for tuning floating wind turbine controllers through integration with OpenFAST models. This Python-based tool automates the derivation of optimal control parameters while ensuring stability across operational regimes [1, 42].

It uses the OpenFAST input file (`.fst`) for the IEA 15 MW turbine on the UMaine semisubmersible, extracting turbine geometry, aerodynamic coefficients, and structural properties. Rotor performance data are provided by the `Cp_Ct_Cq.IEA15MW.txt` lookup table, while the `IEA15MW.yaml` configuration file is used to adjust the five tuning parameters (ζ_{pc}, ω_{pc}, `ps_percent`, `ss_cornerfreq`, `ss_pcgain`) for each experimental case. Execution of the ROSCO tuning routines produces a `DISCON.IN` file containing over 128 parameters that define the functionality of the controller. Together with the corresponding dynamic library, these files are then used as inputs to the SIMA time-domain simulation environment.

4.2. Floating Offshore Wind Turbine Model

By now, it is clear that the model used for all simulations is the IEA 15 MW reference turbine mounted on the UMaine semi-submersible platform with its three-line catenary mooring system. Figure 4.3 illustrates this setup in SIMA, where the turbine is subjected to combined wind and wave loads. The model is capable of fully coupled aero-hydro-servo-elastic simulations. This allows realistic data of turbine performance, structural loading, and platform motion under site conditions. All components were configured in SIMA by SINTEF Ocean.

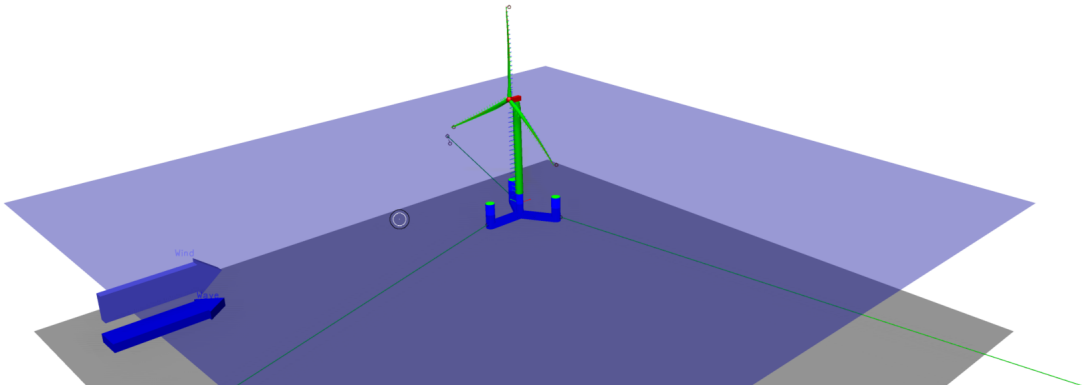


Figure 4.3: Setup of IEA 15 MW wind turbine mounted on the UMaine semi-submersible floater in SIMA.

The IEA 15 MW reference wind turbine, with its main characteristics in Table 4.1, was selected for its status as a widely adopted benchmark in floating wind research and the availability of detailed design documentation. In SIMA, the tower and blades are represented as a series of cross-sections of cylinders and airfoils, respectively. This provides a high-fidelity aerodynamic and structural model that closely reproduces real behaviour while remaining computationally efficient.

Table 4.1: Main characteristics of the IEA Wind 15-MW Turbine [21].

Parameter	Value
Rated power	15 MW
Rotor orientation and configuration	Upwind, three blades
Rotor, hub diameter	240 m, 7.94 m
Hub height	150.0 m
Cut-in, rated, cut-out wind speed	3.0 m/s, 10.59 m/s, 25.0 m/s
Minimum, maximum rotor speed	5.0 rpm, 7.56 rpm
Overhang, shaft tilt, pre-cone	11.35 m, 6.0°, -4.0°
RNA, tower mass	1017 t, 860 t

The UMaine VolturnUS-S reference platform was chosen for this simulation, the design properties of this platform are listed in Table 4.2. Developed as a semisubmersible design, it is tailored to support the IEA 15 MW reference wind turbine. This platform is known for its modularity and adaptability in deep-water environments, with a four-column configuration offering enhanced stability.

Table 4.2: UMaine VolturnUS-S reference platform characteristics [4].

Parameter	Value
Hull Displacement	20,206 m^3
Hull Steel Mass	3,914 t
Tower Interface Mass	100 t
Ballast Mass (Fixed/Fluid)	2,540 t/ 11,300 t
Draft	20 m
Freeboard	15 m
Vertical Centre of Gravity from SWL	-14.94 m
Vertical Centre of Buoyancy from SWL	-13.63 m
Roll Inertia about Center of Gravity	1.251E+10 kg-m^2
Pitch Inertia about Center of Gravity	1.251E+10 kg-m^2
Yaw Inertia about Center of Gravity	2.367E+10 kg-m^2

The mooring system properties in Table 4.3 were chosen to ensure the stability of the floating platform in deep-water environments. The configuration includes three evenly spaced chain mooring lines. This is modeled in SIMA as straight, taut elements whose stiffness, mass distribution, and drag characteristics match those listed below.

Table 4.3: Mooring system properties [4].

Parameter	Value
Mooring System Type	Chain Catenary
Line Type	R3 Studless Mooring Chain
Line Breaking Strength	22,286 kN
Number of Lines	3
Anchor, Fairlead Depth	200 m, 14 m
Anchor, Fairlead Radial Spacing	837.6 m, 58 m
Nominal Chain Diameter	185 mm
Extensional Stiffness	3270 MN
Line Unstretched Length	850 m
Fairlead Pretension	2,437 kN
Fairlead Angle from SWL	56.4°

4.3. Control Strategy

In this study, the ROSCO controller is implemented for the IEA15 MW turbine mounted on the UMaine floater. The controller is initially configured using the baseline power-optimal gains, which have been tuned by NREL for maximum power tracking. This baseline configuration is used to run simulations under predefined metocean conditions, establishing a reference for performance in terms of annual energy production and the fatigue life of the blades.

Subsequently, a series of distinct tuning experiments is conducted by making use of the tuning toolbox [42]. In each experiment, five key controller parameters are systematically varied, these are listed below. This sets up the parameter space that can be

investigated. For each set of these parameters, the ROSCO tuning procedure is executed to generate new proportional and integral gains across the operational range of pitch angles, thereby producing a new controller input file. These multiple controller configurations are then used to run simulations under the same predefined metocean conditions. The chosen values for the controller settings can be seen in Table 4.4.

- **Pitch controller damping ratio** (ζ_{pc}): controls the trade-off between responsiveness and overshoot in blade pitch actuation. Lower values yield faster tracking of wind speed changes (potentially increasing AEP), whereas higher values suppress oscillations.
- **Pitch controller natural frequency** (ω_{pc}): determines how quickly the pitch loop responds to disturbances. A higher ω_{pc} accelerates corrective action for improved power tracking but can excite structural modes, while a lower ω_{pc} smooths actuator movements.
- **Peak shaving percentage** (`ps_percent`): limits the maximum thrust in the transition to the above-rated wind conditions, capping extreme loads. Increasing peak shaving sacrifices a portion of peak power but potentially extends component life.
- **Setpoint smoothing corner frequency** (`ss_cornerfreq`): applies a low-pass filter to the reference pitch setpoint, reducing controller sensitivity to high-frequency wind and wave fluctuations. A lower corner frequency enhances smoothing.
- **Setpoint smoother gain bias percentage** (`ss_pcgain`): adjusts the weight of the smoothed setpoint in the pitch loop. A larger bias amplifies the smoothing effect, whereas a smaller bias allows the controller to more closely follow the raw, unsmoothed setpoint for energy optimisation.

The baseline ROSCO controller is configured with a pitch damping ratio of $\zeta_{pc} = 1.0$, a pitch natural frequency of $\omega_{pc} = 0.2$ rad/s, a peak shaving fraction of `ps_percent`=0.8, a setpoint smoother corner frequency of `ss_cornerfreq`$= 0.17952$rad/s, and a smoother gain bias of `ss_pcgain`$= 0.1$. These settings strike a balance between rapid power capture and moderate load mitigation.

These five parameters were chosen because they govern both the pitch loop, which has the greatest potential to reduce fatigue under above rated winds and high loading, and the transition into generator torque control around rated speed. The pitch damping ratio and natural frequency determine how aggressively the blades respond to gusts, while the peak shaving percentage, setpoint smoother corner frequency, and smoother gain bias control how gradual that response is. The below rated torque control region remains at its default settings to ensure maximum power capture, but smoothing the switch between torque and pitch control can also improve performance. By systematically varying a targeted set of controller parameters across ten configurations, a representative spectrum of turbine behavior is captured. These configurations span from more aggressive to more conservative control settings, enabling a structured exploration of the trade-off between energy production and fatigue. Rather than isolating the impact of each parameter, the focus lies on how different combinations influence overall performance, forming the basis for a Pareto-based roadmap for controller tuning.

Table 4.4: Tuning parameter variations for different simulation cases

Case	ζ_{pc}	ω_{pc}	ps_percent	ss_cornerfreq	ss_pcgain
C1	0.7	0.35	0.6	0.2	0.1
C1.1	0.7	0.3	0.6	0.2	0.1
C1.2	0.7	0.4	0.6	0.2	0.1
C1.3	0.8	0.35	0.6	0.2	0.1
C1.4	0.6	0.35	0.6	0.2	0.1
C2	1.0	0.30	0.7	0.15	0.2
C3	1.2	0.25	0.8	0.1	0.3
C4	1.5	0.20	0.9	0.08	0.4
C5	2.0	0.18	1.0	0.05	0.5
C6	2.5	0.15	0.85	0.07	0.35
C7	3.0	0.12	0.75	0.09	0.25
C8	3.5	0.10	0.65	0.12	0.15
C9	4.0	0.08	0.55	0.14	0.1
C10	4.5	0.06	0.45	0.16	0.05

4.4. Model Validation

To verify the correct implementation of the simulation environment, the behaviour of the ROSCO controller within SIMA is validated against reference results from the literature. The IEA 15 MW reference turbine has been extensively studied, with publicly available data on its control performance under various wind conditions. Since ROSCO is a well-established controller developed by NREL, its expected behaviour is known, allowing for a direct comparison to ensure that the SIMA setup is correctly configured [1].

Simulations are conducted across the full operational wind speed range, generating key performance metrics such as generator torque, tip-speed ratio, and rotor speed. These outputs are then compared to published results from the ROSCO literature. Similar output between the two confirms that the controller functions as intended within the SIMA environment. Any discrepancies are examined to identify potential issues in the setup. This validation step ensures that subsequent controller tuning is based on a reliable and accurately implemented simulation framework.

4.5. Metocean Data Analysis for Simulation Scenarios

To ensure that the simulations are based on realistic environmental conditions, a detailed analysis of the Utsira Nord metocean dataset was conducted. The process of this analysis is visually laid out in Figure 4.4. The dataset provides comprehensive metocean conditions for the Utsira Nord area, a site located in the North Sea off the coast of Norway, seen in Figure 4.5. The dataset spans from 1982 to 2022, offering hourly time series data for key environmental variables, including wind speed, wave height, peak wave period, and more. These data are collected at various heights above the surface, providing a detailed representation of the atmospheric and oceanic conditions at different levels [13]. The analysis focused on extracting representative metocean conditions that define meaningful simulation scenarios for evaluating turbine performance.

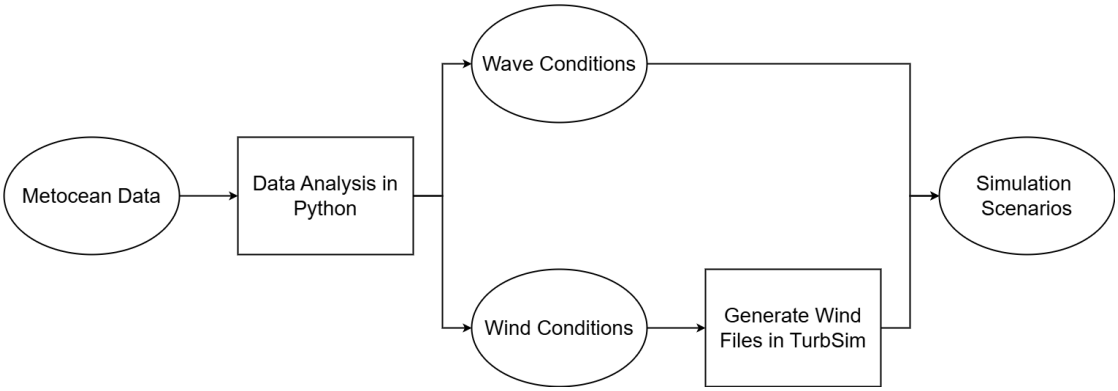


Figure 4.4: Workflow to identify the right simulation scenarios based on the Utsira Nord data.

Utsira Nord is particularly relevant for floating offshore wind turbine projects due to its deep-water location. The region's significant water depths, ranging from 200 to over 300 meters, make it an ideal candidate for floating wind turbine technology, as this technology is designed specifically to operate in locations where traditional fixed-bottom turbines would not be feasible. The site's challenging metocean conditions, including strong wind patterns and variable wave heights, also make it suitable for testing and simulating the performance of floating wind turbines [12]. The importance of utilising established metocean data in simulations cannot be overstated. It allows for a more precise assessment of turbine performance, fatigue life, and energy production potential, while also accounting for the inherent uncertainties and variability of offshore conditions.

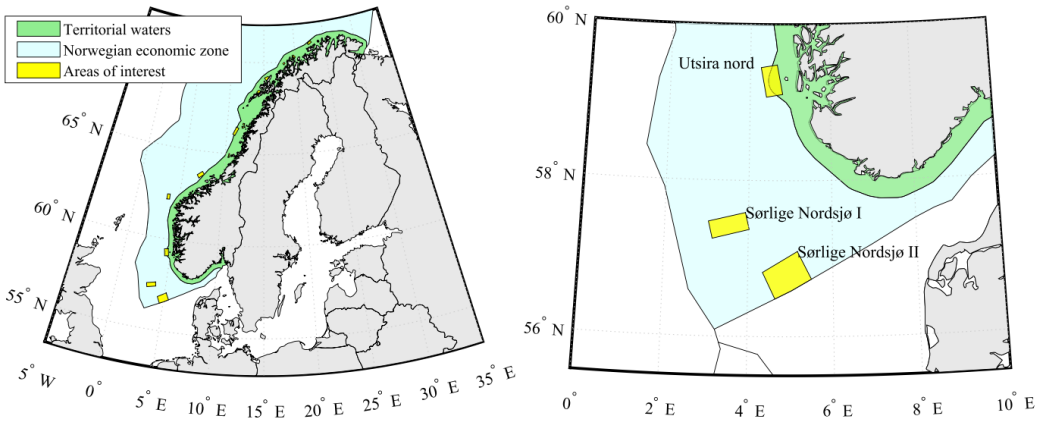


Figure 4.5: Potential wind farm areas in Norway with a close up of the Utsira Nord location [12].

The dataset used was limited to five files covering the years 2018–2022, which were merged along the time dimension to create a continuous dataset. A Weibull distribution was fitted to the full wind speed data to extract the distribution parameters later used to weight the simulation cases. All conditions were evaluated at a height of 150 meters, corresponding to the hub height of the simulated floating wind turbine. Only wind speeds within the turbine's operational range (3–25 m/s) were retained to exclude non-operational extremes.

A histogram was plotted to characterise the distribution of wind speeds within the operational range. Wind speeds were grouped into 1 m/s bins. For each bin, the corresponding friction velocity (u^*) was calculated, and the turbulence intensity was derived using the standard definition (Equation 3.4). The average wind speed, turbulence intensity, and friction velocity in each bin served as the basis for defining wind conditions in the simulations.

For the wave conditions, the significant wave height (H_s) and peak wave period (T_p) were averaged within each wind bin to reflect the coupling between wind and sea state. To evaluate their variability across wind speeds, box plots were generated for both H_s and T_p. This visualisation provided insight into the spread of hydrodynamic conditions within each bin and helped select representative values for simulation.

The statistical analysis produced a summary table that defines the environmental input conditions for each simulation case, including values for wind speed, friction velocity, turbulence intensity, significant wave height, and peak wave period. These values were used to model the environmental loading conditions in SIMA.

Turbulent wind fields were generated using TurbSim [25], with the Kaimal turbulence model selected due to its common use in offshore wind energy applications. Input parameters for TurbSim were taken directly from the bin-wise analysis of wind conditions. In contrast, wave characteristics such as H_s and T_p were manually specified in SIMA based on the JONSWAP spectrum. This combined approach ensures that simulations reflect realistic environmental forcing for accurate assessment of turbine response, fatigue loads, and control system performance.

4.6. Performance Evaluation

The goal of this research was to investigate the effect of the controller on the annual energy production and the blade fatigue life. Therefore, this section provides the methodology of how these metrics were calculated.

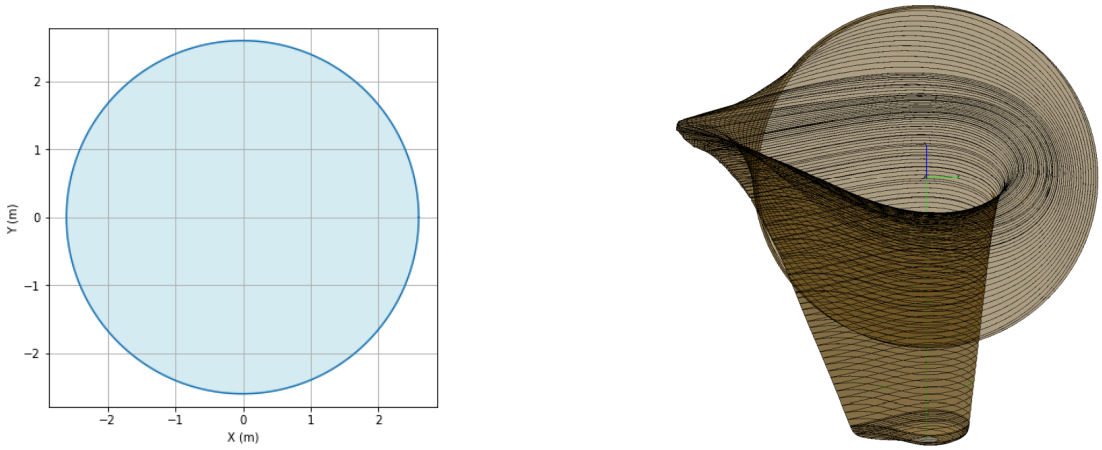
4.6.1. Annual Energy Production

The annual energy production (AEP) is determined by integrating the power output across different wind speeds while considering the probability of occurrence of each wind speed. Using SIMA, power output is computed for a range of wind speeds based on simulations of the wind turbine's performance under different conditions. This involves running simulations at discrete wind speeds, such as $U = 3, 4, \ldots, 25$ m/s, and extracting the corresponding power output values. By overlaying this with the weibul distribution, the mean power was straightforwardly computed, following the theory in section 5.3.

4.6.2. Fatigue Life

One of the key outputs from the simulations is the aerodynamic moment at the blade root about each axis, providing data for M_x, M_y, and M_z. These correspond to the torsional and bending moments. However, In order to translate this to stress magnitudes, a couple of structural computations need to be performed, previously laid out in subsection 3.4.1.

For this, it is essential to look at the blade's geometry at the root, which can be seen in Figure 4.6. More specifically, the inertia of the cross-section. The blade root is a circular shape with a diameter of 5.20 m and a 100 mm wall thickness [21, 71]. Since a thin-walled assumption can be made and by making use of Equation 3.51, it was found that the moment of inertia is, $I = 5.21120$ m^4



(a) Blade root geometry of the airfoil.

(b) Entire blade from the root view [21].

Figure 4.6: Comparison of the blade root geometry and the overall blade view from the root.

Further using the equations from subsection 3.4.1, the stress time series at each wind speed was computed. Each simulation yielded the von Mises stress and its individual components over the simulation period corresponding to a specific wind speed. However, because wind speed in operation is inherently variable, these stress time series must be combined to represent the full operational range.

A Weibull distribution was fitted to the measured wind speed data, providing a statistical representation of the wind speed probability. This Weibull fit was then used to assign a weight to the stress time series from each wind speed. Although the resulting effective stress time series is an average, it accurately reflects both the magnitude of the stresses and their frequency of occurrence. High-amplitude stress events at higher wind speeds, while less frequent, are incorporated with lower weights, and the more common, lower stress levels at lower wind speeds receive higher weights. This weighted combination, therefore, captures the cumulative damage potential over the turbine's lifetime and serves as a robust basis for subsequent fatigue life estimation.

The blade roots of modern offshore wind turbines are constructed from glass-fibre/epoxy composites whose mechanical properties depend strongly on fibre orientation, ply count, and laminate stacking sequence. Because Basquin and Goodman parameters reported in the literature vary widely, a representative set of values was selected: $UTS = 400$ MPa, fatigue exponent $b = 7.0$, and fatigue strength coefficient $a = 80$ MPa [40, 48, 71]. Although using blade-specific material data would yield more precise fatigue predictions for a single control strategy at a given site, this study compares ten controller configurations. Applying identical material parameters across all cases ensures that differences in computed fatigue life arise solely from controller effects and supports an apples-to-apples comparison while retaining realistic composite behaviour.

4.6.3. Pareto Front Optimisation

A Pareto front analysis is used to visualise the trade-off between blade fatigue life and annual energy production. For each controller tuning case, both AEP and blade fatigue life are computed under identical metocean conditions, and the performance metrics are plotted to form a Pareto front. This front displays the set of non-dominated solutions, where any improvement in one metric necessarily results in a decrease in the other.

Each unique controller configuration, defined by its tuning parameters, is evaluated to identify settings that enhance fatigue life while delivering acceptable AEP. The Pareto front visualisation facilitates the selection of the optimal controller design by clearly indicating those configurations that achieve significant improvements in fatigue resistance with minimal energy production penalties. This systematic approach ensures that the final controller design meets the operational performance and reliability requirements for floating offshore wind turbines.

Selection of the “optimal” controller is driven primarily by fatigue life extension, reflecting the priority of this objective function. However, to quantify this problem, the energy gain over the expected extension of the life is calculated. This requires an important assumption, which is that if the blade fatigue life is extended, that means that the wind turbine actually operates that much longer. A sensitivity study then evaluates how the ranking of controllers changes if the lifetime extension assumption is not as absolute, ensuring that the chosen tuning remains robust to uncertainties in blade replacement criteria.

4.6.4. Economic Impact Assessment

The levelised cost of energy is calculated using Equation 3.60, where typical values for capital expenditure (CAPEX) of 6500 EUR/MW and annual operations and maintenance costs (OPEX) of 100 EUR/MW/yr are taken from industry reports [37, 41, 67]. Initially, a discount rate of $r = 7\%$ is assumed, to reflect the cost of capital. The project lifetime n is initially set equal to the baseline blade fatigue life, and the annual energy production AEP_i uses the values computed for each of the controllers. For the optimally tuned controller, the blade life extension determined from the fatigue analysis increases n accordingly, while its slightly lower AEP replaces the baseline value. By inserting these inputs into the LCOE formula, direct comparison of the baseline and optimal controller scenarios reveals whether the improved durability, as well as any associated OPEX savings, offsets the reduction in annual energy yield. A sensitivity study is also performed by varying r and the assumed OPEX reduction to assess the robustness of the economic advantage.

5

Results

This chapter presents the results of the study, organised into three main sections: model validation, metocean data analysis, and performance evaluation of the proposed control strategies. First, the ROSCO controller implementation in SIMA is validated against NREL benchmark results to ensure model reliability. Next, environmental data from the Utsira Nord site is statistically analysed to characterise local wind and wave conditions used in the simulations. Finally, performance results are reported for a range of controller configurations in terms of annual energy production and estimated fatigue life, followed by a Pareto analysis that illustrates the trade-offs between power output and structural longevity.

5.1. Model Validation

The ROSCO implementation in SIMA was benchmarked against NREL's reference results for the IEA 15 MW turbine operating in uniform wind. Figure 5.1 presents the baseline SIMA outputs (blade pitch, generator torque, TSR and rotor speed) across wind speeds of 3–25 m/s using the default ROSCO settings. Figure 5.2 reproduces the corresponding NREL reference curves (right panel). The agreement between the two confirms that the implementation of the turbine, platform, and ROSCO controller in SIMA is correct.

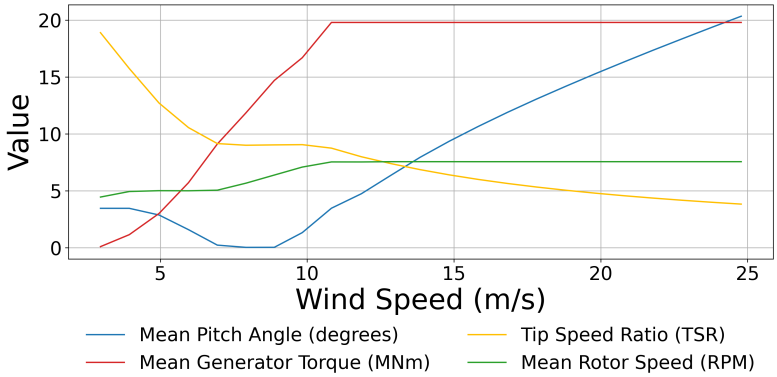


Figure 5.1: Pitch angle, generator torque, TSR and rotor speed plot for validation of the model and the controller.

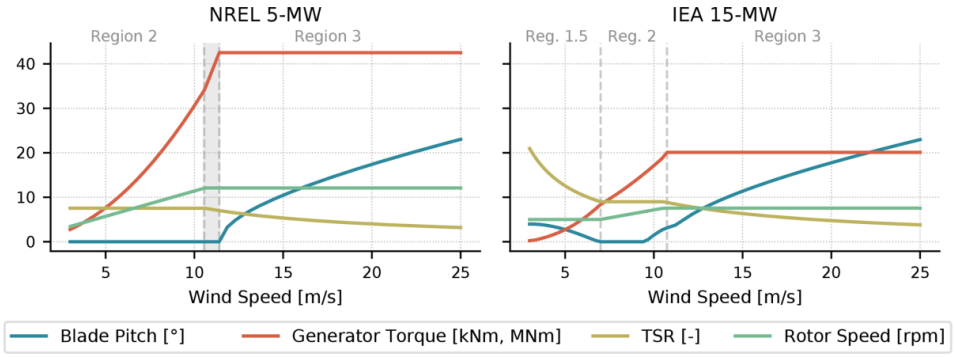


Figure 5.2: Pitch angle, generator torque, TSR and rotor speed plot obtained from NREL [1].

5.2. Metocean Data Analysis

This section presents the environmental statistics derived from the Utsira Nord dataset for 2018–2022, as used to define the simulation scenarios.

5.2.1. Wind Speed Distribution

The histogram in Figure 5.3 shows the frequency of hourly wind speeds between 3 m/s and 25 m/s in 1 m/s bins. Figure 5.4 presents the standalone Weibull probability density function fit (shape $k = 1.93$, scale $c = 11.22$ m/s) to the full wind speed record, which has a median wind speed of 9.87 m/s. The wind speed distribution is positively skewed, with the highest occurrence in the 8–9 m/s bin and a decreasing frequency toward higher speeds. Additionally, it was found that turbulence intensities ranged from 8.58% to 9.90% for the entire operational wind speed range.

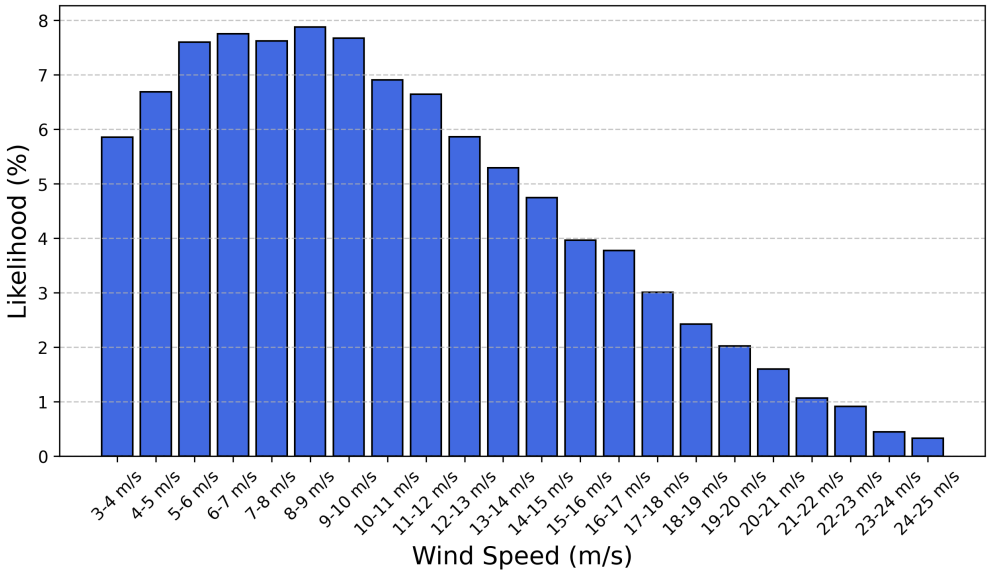


Figure 5.3: Histogram of wind speeds, within operational conditions, at Utsira Nord between 2018 and 2022.

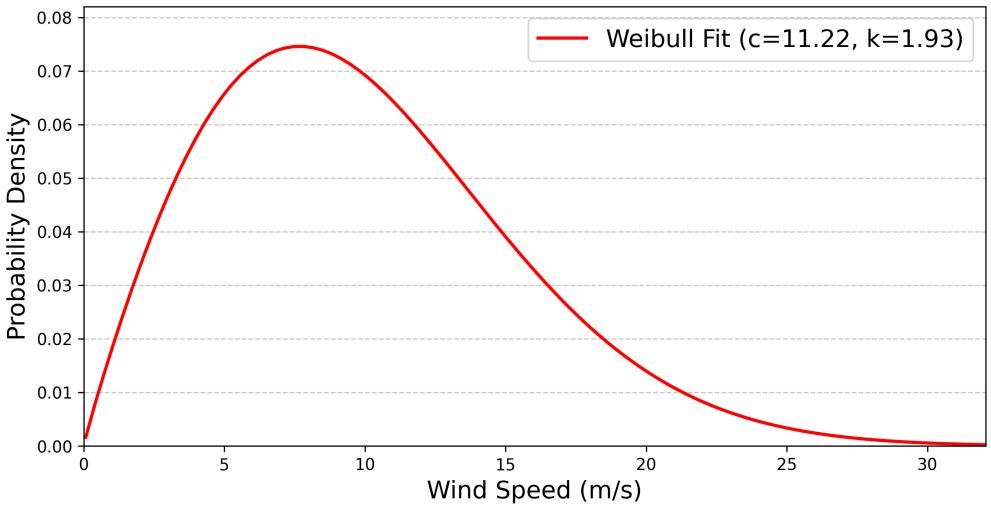


Figure 5.4: Probability of wind speed fitted by a Weibull distribution.

5.2.2. Wave Climate Statistics

Box plots in Figure 5.5 and Figure 5.6 illustrate how significant wave height H_s and peak period T_p vary with wind speed. In Figure 5.5, the interquartile range and whisker lengths of H_s are narrow at low wind speeds and broaden as wind speed increases, with fewer outliers at higher speeds. Conversely, Figure 5.6 shows T_p boxes that narrow with increasing wind speed, also accompanied by a reduction in outlier counts.

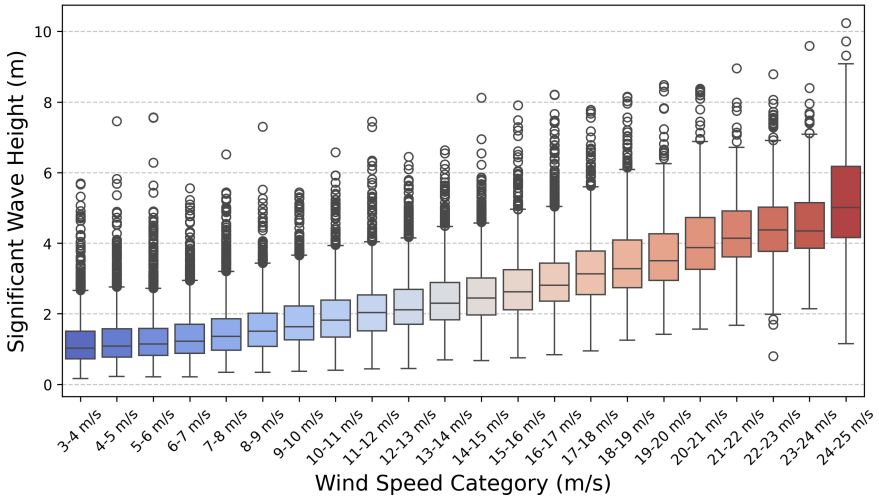


Figure 5.5: Box plot of wind speed vs. significant wave height.

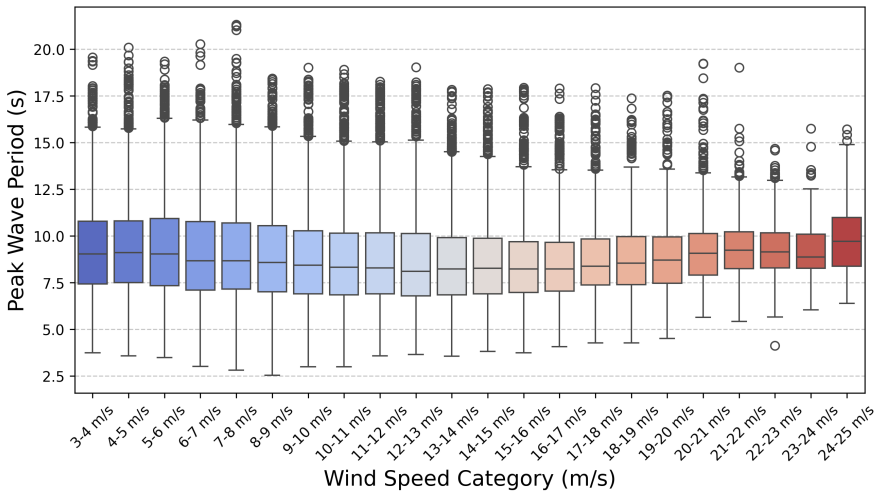


Figure 5.6: Box plot of wind speed vs. Peak Period.

5.3. Performance Evaluation

This section presents the results from the simulations in terms of AEP and Fatigue. Subsequently, the Pareto front is presented, and the analysis of those outputs is established.

5.3.1. Annual Energy Production

Table 5.1 presents the annual energy production (AEP) for all evaluated controller configurations, expressed in MWh and as a percentage change relative to the baseline case (Case B). As expected, the baseline controller achieves the highest AEP at 71643 MWh. Most alternative configurations result in a slight reduction in AEP, with Case C2 showing the largest decrease at -3.49%. Conversely, Case C5 performs almost identically to the baseline, with only a -0.09% drop in AEP. Overall, the variations in AEP across all cases are relatively modest.

Table 5.1: Annual energy production for each controller configuration

Case	AEP [MWh]	△AEP [%]
B	71 643.01	-
C1	70 373.04	-1.77
C1.1	71 115.98	-0.74
C1.2	70 054.67	-2.22
C1.3	70 367.79	-1.78
C1.4	70 536.66	-1.54
C2	69 139.39	-3.49
C3	70 448.97	-1.67
C4	71 450.30	-0.27
C5	71 578.61	-0.09
C6	71 402.97	-0.34
C7	70 719.97	-1.29
C8	70 651.92	-1.38
C9	71 185.95	-0.64
C10	71 431.01	-0.30

5.3.2. Fatigue Life

Table 5.2 shows the estimated fatigue life for each controller configuration and the percentage change compared to the baseline. The baseline controller yields a fatigue life of 25.7 years. Several configurations result in substantial improvements. Cases C1.2, C1.3, C1.4, C3, and C7 reach fatigue life values above 33 years, with Case C1.2 achieving the highest at 34.7 years (+35.02%). In contrast, Case C2 results in the lowest fatigue life at 25.6 years, a slight decrease of -0.39% compared to the baseline.

Table 5.2: Estimated fatigue life for each controller configuration

Case	Fatigue life [yrs]	△Fatigue life [%]
B	25.7	-
C1	31.6	+22.96
C1.1	29.1	+13.23
C1.2	34.7	+35.02
C1.3	33.7	+31.13
C1.4	34.5	+34.24
C2	25.6	-0.39
C3	34.5	+34.24
C4	29.3	+14.01
C5	29.2	+13.62
C6	30.4	+18.29
C7	34.4	+33.85
C8	26.7	+3.89
C9	29.7	+15.56
C10	26.7	+3.89

5.3.3. Pareto Front of AEP vs. Fatigue Life

The scatter plot in Figure 5.7 visualises the trade-off between AEP and fatigue life across all controller configurations. From the distribution of data points, it is possible to have an idea of an approximate Pareto front, which appears to pass through configurations C1.4, C3, C4, C5, C6, and C7. These controllers offer strong trade-offs, combining relatively high AEP with substantially increased fatigue life. In contrast, the remaining configurations exhibit characteristics of dominated solutions. Notably, Case C2 stands out as an outlier as it underperforms in both metrics.

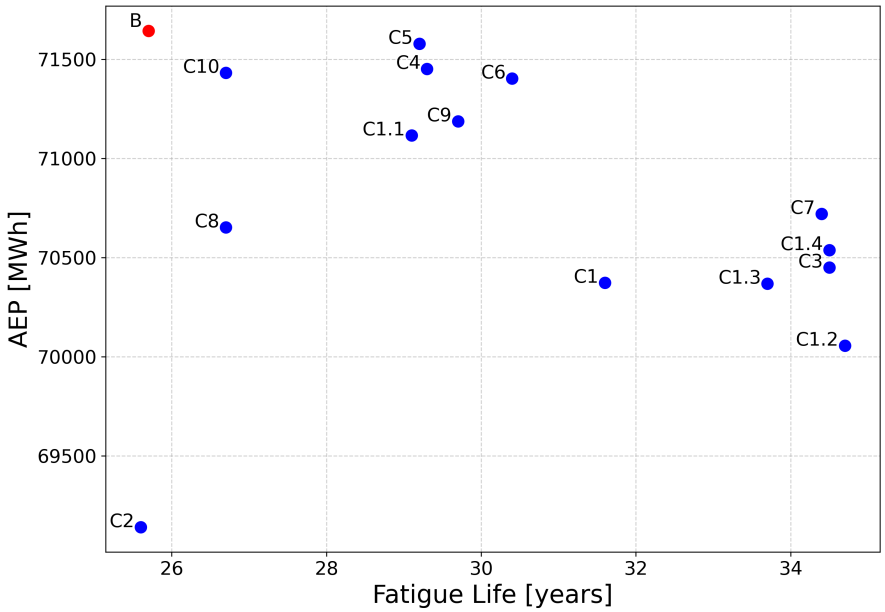


Figure 5.7: Pareto front comparing AEP and fatigue life for all strategies.

5.3.4. Cumulative Energy Gain

The bar plot in Figure 5.8 shows the relative change in total energy production for each controller, assuming longer fatigue life allows extended turbine operation. Most configurations lead to an increase in total energy production, with Cases C1.2, C1.4, C3, and C7 showing the highest improvements of 32.03%, 32.17%, 32.00%, and 32.13%, respectively. Other configurations, such as C1, C1.3, C4, C5, C6, and C9, also exhibit notable increases, ranging from 13.52% to 28.79%. Conversely, Case C2 results in a decrease of -3.87%. The remaining cases show more modest gains in total energy production.

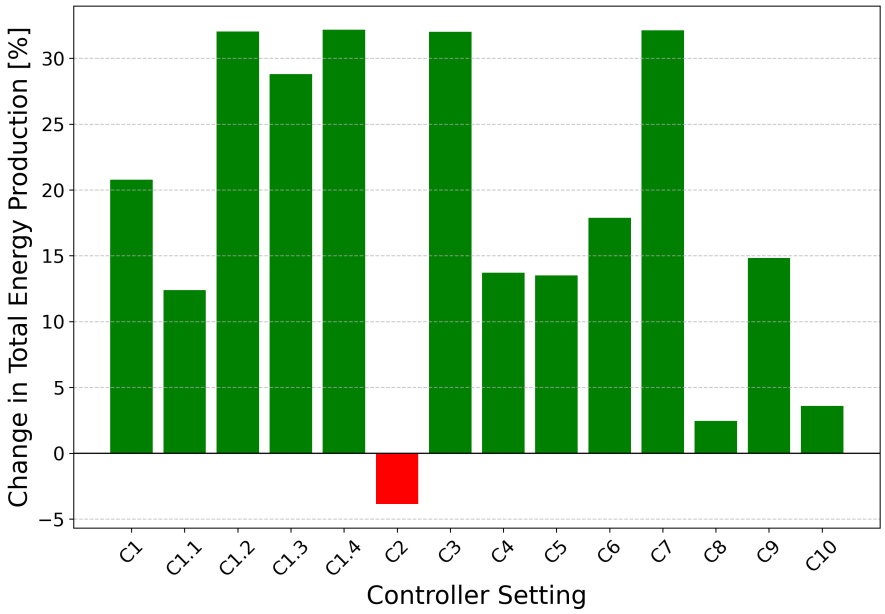


Figure 5.8: Additional energy generated as a function of extended fatigue life.

To test the robustness of the results, a sensitivity analysis is performed by modifying the fatigue life constraint. Instead of using each controller's actual fatigue life, a fixed threshold of 28.27 years, 10% above the baseline, is applied. As shown in Figure 5.9, most controllers still yield a positive energy gain. Controller C5 achieves the highest gain at 9.90%, followed by C4 (9.70%), C6 (9.63%), C9 (9.30%), and C1.1 (9.19%).

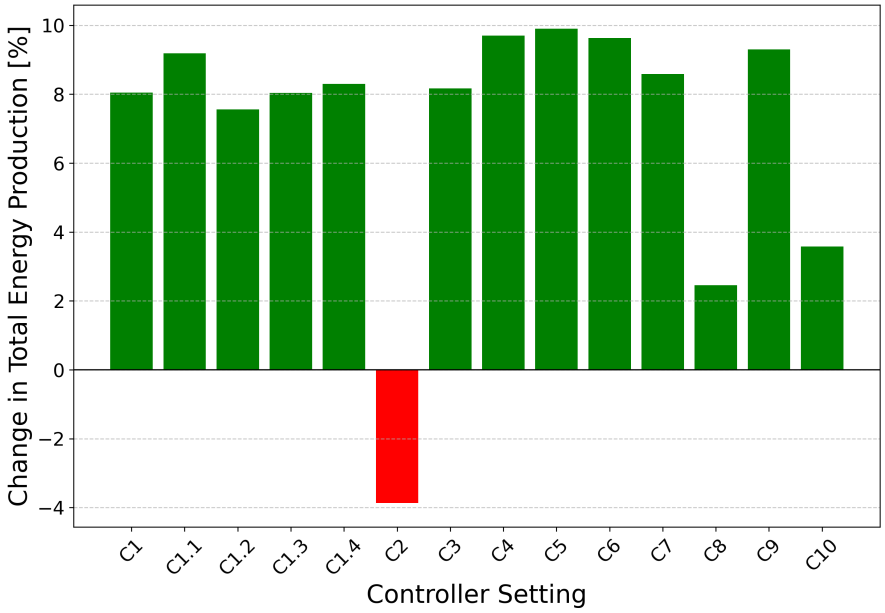


Figure 5.9: Additional energy generated as a function of extended fatigue life at the limit of 28.27 years.

5.4. Cost Evaluation: LCOE Comparison

At this point, C1.4 and C5 were identified as the optimal choices in the above-mentioned cases, with the condition for choosing being the fact that they have the most added energy. One case being if the assumption is that the extension in fatigue life directly dictates the operational lifetime, and one being if the goal is to increase fatigue life and operational life by 10%, respectively. For these cases, an LCOE comparison was made the results of which can be found in Figure 5.10. These results are plotted for varying discount rates, for 7%, 8.5% and 10%. For a 7% discount rate, the baseline turbine yields an LCOE of 136.02 EUR/MWh. C1.4 reduces that to 128.02 EUR/MWh (a 5.8% drop), and C5 reaches 131.90 EUR/MWh (a 3.0% drop). Higher rates raise all LCOEs but preserve the same ranking and savings.

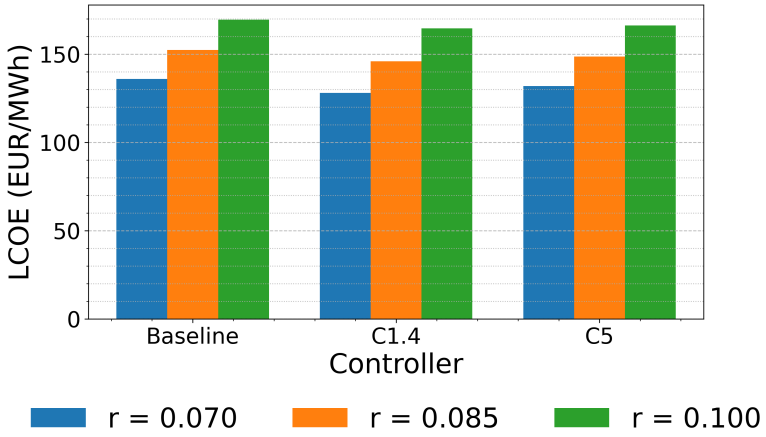


Figure 5.10: Comparison of LCOE for the baseline and the optimally tuned controllers for varying discount rates.

Subsequently, a scenario was evaluated in which the operational lifetime remains fixed at the baseline value (25.7 years), but annual OPEX for C1.4 and C5 are reduced by 10% to reflect less frequent blade repairs. Under these assumptions, the resulting LCOE values for a 7% discount rate, shown in Figure 5.11, still favour the tuned controllers. The baseline LCOE remains at 136.02 EUR/MWh. While C1.4 falls slightly below that value with an LCOE of 135.60 EUR/MWh, C5 drops to 134.04 EUR/MWh.

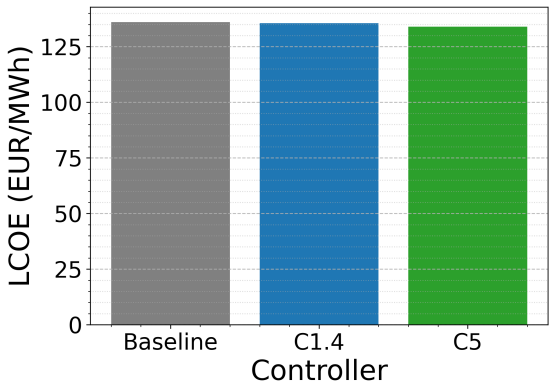


Figure 5.11: Comparison of LCOE for the baseline and the optimally tuned controllers for reduced OPEX.

6

Discussion

In this chapter, the findings presented in chapter 5 are examined. First, the metocean data at Utsira Nord are validated to confirm that the five-year record captures the predominant wind regimes and wave climates. Subsequent sections then address annual energy production, fatigue life, Pareto trade-offs between power and durability, cumulative energy gains with sensitivity considerations, and finally, the levelised cost of energy, each time linking back to the site conditions and control strategies identified earlier.

6.1. Metocean Data at Utsira Nord

To validate the site-specific analysis, the results are compared with the metocean analysis by Cheynet et al. (2024) [12]. Over the 2018–2022 record at 150 m hub height, the highest frequency of winds was found between 5 and 10 m/s, and a median speed of 9.87 m/s was determined, closely matching the 9.7 m/s found by Cheynet. The fitted Weibull parameters (scale $c = 11.22$, shape $k = 1.93$) align with Cheynet's values ($c = 11.6$, $k = 1.97$), and the histogram together with the right-skewed fit confirms a regime dominated by moderate winds punctuated by rare strong events [12].

The wave statistics similarly reflect Cheynet's joint wind–wave climatology. Mean significant wave height H_s is seen to increase from approximately 1.16 m in the lowest wind bins to around 5.28 m at the highest, illustrating the greater energy transfer from wind to waves as wind speed rises. In the box plots, widening interquartile ranges at higher speeds indicate enhanced variability under gustier conditions, while the reduced number of outliers suggests that once a threshold wind energy is reached, wave heights cluster more tightly around an equilibrium state.

Peak period T_p remains near 9 s across most wind bins. With increasing wind speed, boxes and whiskers narrow and outliers diminish because wave spectra converge toward a dominant frequency once the water body and wind duration permit full wave development. These trends reproduce Cheynet's characterization of a moderately energetic offshore environment, thereby ensuring that subsequent energy production and load simulations are based on realistic metocean inputs [12].

6.2. Annual Energy Production Results

The annual energy production results from the simulations fall within the expected output range for a 15 MW wind turbine, with values around 70,000 MWh. This is consistent with typical expectations for a turbine of this capacity and further validates the correct implementation of both the controller and the simulation setup [66]. The AEP values for the various controller configurations are generally in line with the baseline (B) case, confirming that the modifications to the controller do not produce unrealistic or excessively large deviations in performance.

Overall, the changes in AEP are moderate across all the controller tuning cases. This is not surprising, considering that the bulk of the tuning was focused on the pitch controller and the transition point. Since the most prominent wind speeds in the operational regime of the turbine are in the below-rated wind speed region, where the generator torque controller is typically the dominant controller, it is expected that the impact of changes to the pitch controller would be relatively minor. The result is that the changes in AEP remain moderate, with only slight deviations from the baseline case.

Two cases stand out due to the most noticeable changes in AEP, C5 and C2. These configurations show the most extreme differences from the baseline, making them worth examining in more detail.

For C5, the AEP is 71,578.61 MWh, which results in a minimal decrease of just 0.09% compared to the baseline. The power curve for C5, illustrated in Figure 6.1, closely mirrors that of the baseline, indicating that the turbine operates almost identically to the baseline case across different wind speeds. This suggests that the changes in the controller settings for C5 have little impact on overall energy production, with the turbine maintaining similar performance levels.

In contrast, C2 shows a more significant reduction in AEP, with a 3.49% decrease. The power curve for C2, illustrated in Figure 6.1, reveals a noticeable delay in reaching rated power once the wind speed exceeds the rated value, compared to the baseline. This delayed response indicates that the controller's settings in C2 affect the turbine's ability to quickly reach and maintain rated power, ultimately leading to lower energy production.

The AEP results for Cases C1 to C1.4 provide insight into the effect of tuning the natural frequency and damping ratio of the pitch controller. Case C1, the initial reference tuning, shows a 1.77% reduction in AEP compared to the baseline. Decreasing the natural frequency in Case C1.1 improves AEP to a reduction of only 0.74%, whereas increasing it in Case C1.2 results in the largest AEP loss of the subset, at 2.22%. This suggests that a lower natural frequency can improve energy capture. Similarly, increasing the damping ratio in C1.3 yields a small improvement relative to C1, with a 1.78% drop in AEP, while decreasing the damping ratio in C1.4 leads to a slightly better result of -1.54%. These trends imply that the impact of the damping ratio is less pronounced than that of the natural frequency.

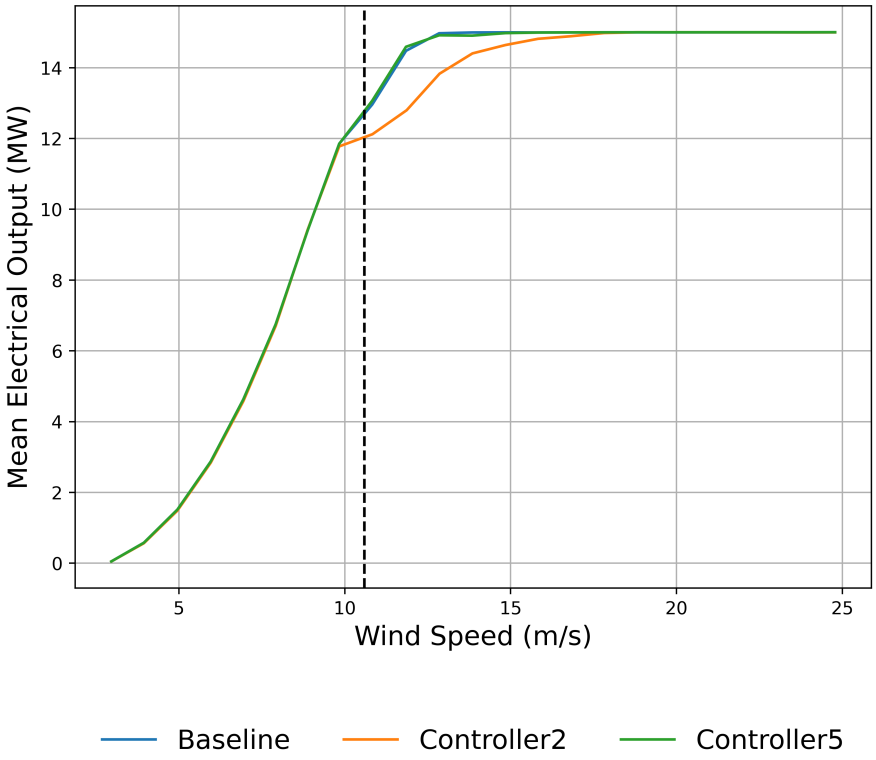


Figure 6.1: Power curves for C5 and C2 compared to the baseline.

6.3. Fatigue Life Results

The fatigue analysis reveals significant differences in performance between controller configurations. All tested controllers, except for one, resulted in an improvement in fatigue life relative to the baseline. This highlights the effectiveness of optimising pitch control specifically for high-load operating regions. However, it also confirms the sensitivity of fatigue calculations: minor adjustments in gain parameters can lead to substantial changes in estimated fatigue life. This is a direct consequence of how fatigue damage accumulates, not linearly with cycle count, but exponentially with stress amplitude.

The two most noteworthy cases in terms of fatigue behaviour are C2 and C1.2. To gain more insight into the mechanisms behind their performance, the rainflow-counted stress amplitudes and corresponding cycle counts were compared to the baseline, as shown in Figures 6.2 and 6.3.

For C2, the overall stress-cycle distribution is similar to the baseline, but a notable increase in the number of low-stress cycles is observed. This increase in low-amplitude cycling is not inherently damaging to. More importantly, C2 exhibits a slightly higher count of high-stress cycles, defined here as cycles exceeding 5 MPa. Specifically, the high-stress cycle counts are 72.0 for the baseline and 71.0 for C2, a negligible change in absolute terms, but C2 also has significantly more low-stress cycles (462.0 vs. 323.5), so the slight decrease in fatigue life is expected; it is only a 0.39% drop after all.

In contrast, C1.2 achieves a more meaningful reduction in high-stress cycles. The total number of such cycles drops to 66.0, corresponding to only 12.44% of its total rainflow

cycles compared to 18.2% for the baseline. This decrease may seem small in percentage terms, but because fatigue damage is proportional to the stress amplitude raised to the 7th power ($m = 7$ in this analysis), it results in a disproportionately large benefit in terms of fatigue life. A small reduction in high-amplitude loading can thus translate into a significant improvement in blade lifespan.

These results reinforce the idea that, for composite materials used in rotor blades, fatigue is largely driven by the highest stress ranges. The increase in low-stress cycling in C2 does not alter the blade fatigue life that much, while C1.2's ability to reduce high-amplitude events directly translates to a reduction in damage.

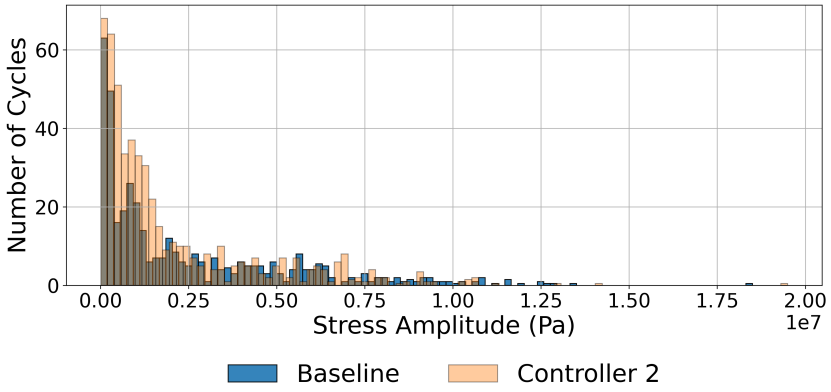


Figure 6.2: Stress amplitudes and cycle counts for Controller 2 compared to the baseline.

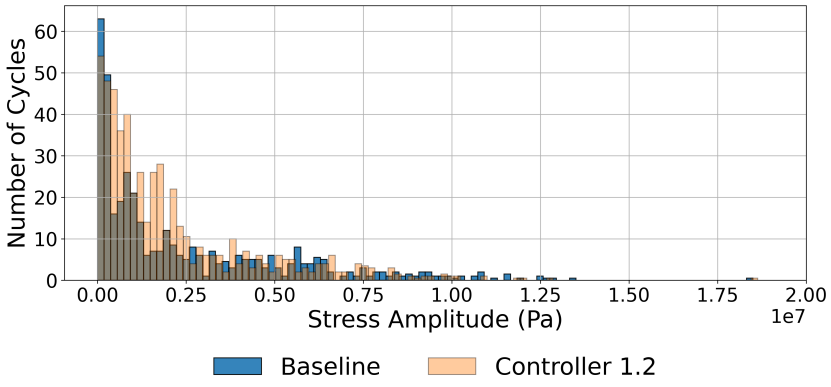


Figure 6.3: Stress amplitudes and cycle counts for Controller 1.2 compared to the baseline.

The fatigue life results for Cases C1 to C1.4 highlight the influence of pitch controller tuning on structural loading. Case C1, which applies a general tuning change from the baseline, shows a 22.96% improvement in fatigue life. Reducing the natural frequency in Case C1.1 lowers the controller's responsiveness, leading to smoother pitch actions and a moderate fatigue life gain of 13.23%. Conversely, increasing the natural frequency in Case C1.2 results in the highest observed fatigue life across all configurations, at +35.02%. This suggests that a faster-reacting pitch controller can effectively mitigate load excursions under dynamic wind conditions, thereby reducing fatigue. The impact of damping ratio changes is subtler: increasing the damping in Case C1.3 leads to a

31.13% gain, while decreasing it in Case C1.4 results in a slightly higher improvement of 34.24%. These outcomes indicate that a higher natural frequency combined with relatively low damping offers the best load reduction in this system. In general, while both parameters play a role, the natural frequency appears to have a more pronounced influence on fatigue life than the damping ratio.

6.4. Pareto Front

Given the wide range of the parameter space and the distinct differences between each controller configuration, no direct correlation between individual parameters and performance metrics can be reliably established. This was intentional, as the goal was not to isolate single parameter effects but to explore the broader system response to varied tuning.

The initial cases C1 to C1.4 provide useful insights into how changes in natural frequency and damping ratio influence fatigue life and energy production within a controlled subset of parameters.

The larger parameter space covers a more diverse and intentionally varied set of configurations, which prevents drawing direct conclusions about individual parameters. Despite this designed variation, the results reveal a clear and well-defined Pareto front. This shows that even small adjustments in controller settings can balance fatigue reduction and energy production effectively.

The appearance of this Pareto front confirms that the evaluation framework can successfully identify trade-offs and guide further controller development toward improved and optimised solutions.

6.5. Cumulative Energy Gain

The results on cumulative energy gain illustrate that several tuned controllers outperform the baseline in terms of total energy production when extended fatigue life is taken into account. Controllers C1.2, C1.4, C3, and C7 show the highest gains, all exceeding 32%, indicating that their settings lead to an effective balance between load mitigation and power output stability. Other cases, such as C1, C1.3, C4, C5, C6, and C9, also display consistent improvements ranging between roughly 13% and 29%.

While the diversity in parameter combinations makes it difficult to attribute performance gains to individual tuning changes, the overall spread of results supports the idea that significant lifecycle benefits can be achieved through targeted controller adjustments. The negative outcome of Case C2 underscores the importance of careful tuning, as not all modifications lead to positive results.

To assess the robustness of these findings, a sensitivity analysis was conducted by fixing the fatigue life to a uniform threshold (10% above baseline). Even under this constraint, most controllers continued to deliver energy gains, with C5, C4, C6, C9, and C1.1 ranking highest. In this case these mentioned cases all achieve cumulative energy gains of over 9%. This consistency across both individual and fixed-lifetime scenarios suggests that the tuning approach, while broad in parameter variation, can lead to reliable improvements in long-term performance.

6.6. Levelised Cost of Energy Analysis

The LCOE comparison provides a practical view of how controller tuning affects overall economic performance. The reductions achieved by C1.4 and C5 may seem modest in percentage terms, but they represent meaningful cost savings over the turbine's lifetime. C1.4 shows a 5.8% reduction in LCOE, and C5 follows with a 3.0% decrease compared to the baseline. These results suggest that tuning the controller with fatigue performance in mind can contribute to lower energy costs. When discount rates rise, cost reductions persist at higher rates. These reductions may appear modest, but over the lifetime, they translate into significant savings in the levelised cost of energy.

A scenario was tested with fixed lifetime (25.7 years) but 10 % lower O&M costs for C1.4 and C5. In this case, C5's LCOE falls to 134.04 EUR/MWh, and C1.4 has an LCOE of 135.60 EUR/MWh. This demonstrates that even without extending turbine life, modest O&M savings from improved control could yield a tangible reduction in LCOE.

This supports the idea that controller design choices, when aligned with long-term objectives like fatigue life and power output, can influence not only technical performance but also financial viability.

The findings also support the structure of the full workflow. They show how early-stage controller adjustments can propagate through to affect long-term outcomes, including cost metrics like LCOE. This strengthens the case for integrating fatigue-aware control strategies in the design of future floating wind systems.

7

Conclusion and Recommendations

This thesis has developed and implemented an end-to-end framework for tuning PI controllers in floating offshore wind turbines. The framework combines metocean data analysis, controller tuning, high-fidelity aero-hydro-servo simulations, AEP estimation, fatigue assessment using rainflow counting and S-N curves, Pareto front generation to balance AEP and fatigue objectives, and levelised cost of energy modelling. It provides a transparent and reproducible workflow from environmental input to economic output.

The results indicate that even small adjustments to controller parameters can lead to significant improvements in fatigue performance with minimal impact on energy production. In the most favourable cases, annual energy production decreased by less than 1%, while blade fatigue life improved by at least 13%. In other scenarios, fatigue life was extended by up to 33%, with a slightly higher AEP reduction of 1.3%. These enhancements also translate into a lower levelised cost of energy, with reductions of up to 6% under realistic economic assumptions. This highlights the value of evaluating and tuning control strategies early in the design process, as these decisions can result in substantial long-term benefits for structural reliability and overall project economics. The developed framework supports this process by enabling systematic exploration of control parameters and their trade-offs, helping researchers and engineers to align control design with site-specific conditions and project objectives.

While this thesis focused on developing a functional framework to assess controller performance, the emphasis on establishing the overall structure meant that less attention was given to the specific influence of individual control parameters on AEP and blade fatigue life. As a result, it remains difficult to draw clear conclusions about which parameter settings are responsible for certain outcomes. A logical next step would be to explore the parameter space more systematically, isolating the effects of each variable to better understand their individual contributions and interactions. Additionally, a Bayesian optimisation could be explored to enhance this research. This would strengthen the framework's ability to guide controller design decisions with greater precision. Further improvements could also include incorporating artificial intelligence for adaptive tuning based on real-time turbine performance, as well as expanding the analysis to account for multidirectional wind and wave conditions, seasonal variations, and evolving structural health throughout the turbine's operational life.

References

[1] Nikhar J. Abbas et al. “A reference open-source controller for fixed and floating offshore wind turbines”. In: *Wind energy science* 7.1 (Jan. 2022), pp. 53–73. DOI: 10.5194/wes-7-53-2022. URL: https://doi.org/10.5194/wes-7-53-2022.

[2] Morteza Ahmadivala. “Towards optimal maintenance planning of existing structures based on time-dependent reliability analysis”. PhD thesis. Université Clermont Auvergne, Dec. 2020.

[3] Alberto Montanari. *Mechanics of sea waves*. URL: https://www.albertomontanari.it/node/123.

[4] Christopher Allen et al. *Definition of the UMAINE VolturnUS-S reference Platform developed for the IEA Wind 15-Megawatt Offshore reference wind turbine*. Tech. rep. NREL, July 2020. DOI: 10.2172/1660012. URL: https://doi.org/10.2172/1660012.

[5] R Amaral et al. “A frequency-time domain method for annual energy production estimation in floating wind turbines”. In: *Journal of Physics Conference Series* 2265.4 (May 2022), p. 042025. DOI: 10.1088/1742-6596/2265/4/042025. URL: https://doi.org/10.1088/1742-6596/2265/4/042025.

[6] Hadi Amlashi and Omid Lotfizadeh. *Life cycle management and probabilistic levelized cost of energy analysis of floating offshore wind farms*. Springer, Jan. 2025, pp. 1749–1759. DOI: 10.1007/978-3-031-69626-8\{_}145. URL: https://doi.org/10.1007/978-3-031-69626-8_145.

[7] Ferdinand Beer et al. *Mechanics of materials*. McGraw-Hill, Jan. 2011.

[8] Edén Bojórquez et al. “Comparison of spectral density models to simulate wind records”. In: *KSCE Journal of Civil Engineering* 21.4 (July 2016), pp. 1299–1306. DOI: 10.1007/s12205-016-1460-y. URL: https://doi.org/10.1007/s12205-016-1460-y.

[9] Livia Brandetti et al. “Analysis and multi-objective optimisation of wind turbine torque control strategies”. In: *Wind energy science* 8.10 (Oct. 2023), pp. 1553–1573. DOI: 10.5194/wes-8-1553-2023. URL: https://doi.org/10.5194/wes-8-1553-2023.

[10] Tony Burton et al. *Wind Energy Handbook*. John Wiley and Sons, Ltd, May 2011. DOI: 10.1002/9781119992714. URL: https://doi.org/10.1002/9781119992714.

[11] Yongtao Cao, Byran J. Smucker, and Timothy J. Robinson. “A hybrid elitist pareto-based coordinate exchange algorithm for constructing multi-criteria optimal experimental designs”. In: *Statistics and Computing* 27.2 (Feb. 2016), pp. 423–437. DOI: 10.1007/s11222-016-9630-9. URL: https://doi.org/10.1007/s11222-016-9630-9.

[12] Etienne Cheynet, Lin Li, and Zhiyu Jiang. “Metocean conditions at two Norwegian sites for development of offshore wind farms”. In: *Renewable Energy* 224 (Feb. 2024), p. 120184. DOI: 10.1016/j.renene.2024.120184. URL: https://doi.org/10.1016/j.renene.2024.120184.

[13] Etienne Cheynet, Lin Li, and Zhiyu Jiang. *Metocean Conditions at Utsira Nord with NORA3 (1982-2022)*. Oct. 2023. DOI: 10.5281/zenodo.10048048. URL: https://zenodo.org/doi/10.5281/zenodo.10048048.

[14] Seongpil Cho, Jongseo Park, and Minjoo Choi. “Fault classification of a blade pitch system in a floating wind turbine based on a recurrent neural network”. In: *Journal of Ocean Engineering and Technology* 35.4 (July 2021), pp. 287–295. DOI: 10.26748/ksoe.2021.018. URL: https://doi.org/10.26748/ksoe.2021.018.

[15] Shima Najem Clarke et al. “Effect of Cycle-Counting Methods on effective stress range and number of stress cycles for Fatigue-Prone details”. In: *Transportation Research Record Journal of the Transportation Research Board* 1740.1 (Jan. 2000), pp. 49–60. DOI: 10.3141/1740-07. URL: https://doi.org/10.3141/1740-07.

[16] Joao Cruz and Mairead Atcheson. *Floating Offshore Wind Energy*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29398-1.

[17] Samuel Evans et al. “A simple method for modelling fatigue spectra of small wind turbine blades”. In: *Wind Energy* 24.6 (Nov. 2020), pp. 549–557. DOI: 10.1002/we.2588. URL: https://doi.org/10.1002/we.2588.

[18] Breiffni Fitzgerald and Saptarshi Sarkar. “Observer based pitch control for load mitigation and power regulation of floating offshore wind turbines”. In: *Journal of Physics Conference Series* 2647.3 (June 2024), p. 032003. DOI: 10.1088/1742-6596/2647/3/032003. URL: https://doi.org/10.1088/1742-6596/2647/3/032003.

[19] Mihai Florian and John Dalsgaard Sørensen. “Wind Turbine Blade Life-Time Assessment Model for Preventive planning of operation and maintenance”. In: *Journal of Marine Science and Engineering* 3.3 (Sept. 2015), pp. 1027–1040. DOI: 10.3390/jmse3031027. URL: https://doi.org/10.3390/jmse3031027.

[20] Alessandro Fontanella et al. “Assessing the impact of waves and platform dynamics on floating wind-turbine energy production”. In: *Wind energy science* 9.6 (June 2024), pp. 1393–1417. DOI: 10.5194/wes-9-1393-2024. URL: https://doi.org/10.5194/wes-9-1393-2024.

[21] Evan Gaertner et al. *IEA WIND TCP Task 37: Definition of the IEA 15-Megawatt Offshore Reference Wind Turbine*. Tech. rep. NREL, Mar. 2020. DOI: 10.2172/1603478. URL: https://doi.org/10.2172/1603478.

[22] Ju Gao, Bert Sweetman, and Shanran Tang. “Multiaxial fatigue assessment of floating offshore wind turbine blades operating on compliant floating platforms”. In: *Ocean Engineering* 261 (Aug. 2022), p. 111921. DOI: 10.1016/j.oceaneng.2022.111921. URL: https://doi.org/10.1016/j.oceaneng.2022.111921.

[23] M. Grujicic et al. “Multidisciplinary design optimization for Glass-Fiber Epoxy-Matrix Composite 5 MW Horizontal-Axis Wind-Turbine blades”. In: *Journal of Materials Engineering and Performance* 19.8 (Feb. 2010), pp. 1116–1127. DOI: 10.1007/s11665-010-9596-2. URL: https://doi.org/10.1007/s11665-010-9596-2.

[24] Kamal Jahani, Robert G. Langlois, and Fred F. Afagh. “Structural dynamics of offshore Wind Turbines: A review”. In: *Ocean Engineering* 251 (Mar. 2022), p. 111136. DOI: 10.1016/j.oceaneng.2022.111136. URL: https://doi.org/10.1016/j.oceaneng.2022.111136.

[25] B. J. Jonkman. *TurbSim User’s Guide: Version 1.50*. Tech. rep. NREL, Sept. 2009. DOI: 10.2172/965520. URL: https://doi.org/10.2172/965520.

[26] Meysam Karimi, Brad Buckham, and Curran Crawford. “A fully coupled frequency domain model for floating offshore wind turbines”. In: *Journal of Ocean Engineering and Marine Energy* 5.2 (May 2019), pp. 135–158. DOI: 10.1007/s40722-019-00134-x. URL: https://doi.org/10.1007/s40722-019-00134-x.

[27] Pravin A Kulkarni, Ashwinkumar S Dhoble, and Pramod M Padole. “Deep neural network-based wind speed forecasting and fatigue analysis of a large composite wind turbine blade”. In: *Proceedings of the Institution of Mechanical Engineers Part C Journal of Mechanical Engineering Science* 233.8 (Sept. 2018), pp. 2794–2812. DOI: 10.1177/0954406218797972. URL: https://doi.org/10.1177/0954406218797972.

[28] Manuel Lara et al. “Multi-objective optimization for simultaneously designing active control of tower vibrations and power control in wind turbines”. In: *Energy Reports* 9 (Jan. 2023), pp. 1637–1650. DOI: 10.1016/j.egyr.2022.12.141. URL: https://doi.org/10.1016/j.egyr.2022.12.141.

[29] Joseph C. Y. Lee and M. Jason Fields. “An overview of wind-energy-production prediction bias, losses, and uncertainties”. In: *Wind energy science* 6.2 (Mar. 2021), pp. 311–365. DOI: 10.5194/wes-6-311-2021. URL: https://doi.org/10.5194/wes-6-311-2021.

[30] Frank Lemmer et al. “Robust gain scheduling baseline controller for floating offshore wind turbines”. In: *Wind Energy* 23.1 (Sept. 2019), pp. 17–30. DOI: 10.1002/we.2408. URL: https://doi.org/10.1002/we.2408.

[31] Eben Lenfest et al. “Tuning of Nacelle Feedback Gains for Floating Wind Turbine Controllers Using a Two-DOF Model”. In: *NREL* (Aug. 2020). DOI: 10.1115/omae2020-18770. URL: https://doi.org/10.1115/omae2020-18770.

[32] Anastasia E. Lennon. *Vineyard Wind turbine blade sustains damage offshore*. July 2024. URL: https://newbedfordlight.org/vineyard-wind-turbine-blade-sustains-damage-offshore/.

[33] P. F. Liu et al. “Fatigue life Evaluation of offshore composite wind turbine blades at Zhoushan Islands of China using wind site data”. In: *Applied Composite Materials* 30.4 (Jan. 2023), pp. 1097–1122. DOI: 10.1007/s10443-022-10098-1. URL: https://doi.org/10.1007/s10443-022-10098-1.

[34] M.A. López-Romero et al. “PI-based active tower damper for offshore wind turbines”. In: *IFAC-PapersOnLine* 58.7 (Jan. 2024), pp. 515–520. DOI: 10.1016/j.ifacol.2024.08.114. URL: https://doi.org/10.1016/j.ifacol.2024.08.114.

[35] J. F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind Energy explained*. John Wiley and Sons, Ltd, Dec. 2009. DOI: 10.1002/9781119994367. URL: https://doi.org/10.1002/9781119994367.

[36] A. Martinez and G. Iglesias. “Multi-parameter analysis and mapping of the levelised cost of energy from floating offshore wind in the Mediterranean Sea”. In: *Energy Conversion and Management* 243 (June 2021), p. 114416. DOI: 10.1016/j.enconman.2021.114416. URL: https://doi.org/10.1016/j.enconman.2021.114416.

[37] Angel McCoy et al. *Offshore Wind Market Report: 2024 Edition*. Tech. rep. NREL, Aug. 2024. DOI: 10.2172/2434297. URL: https://doi.org/10.2172/2434297.

[38] Fanzhong Meng et al. “Experimental study of floating wind turbine control on a TetraSub floater with tower velocity feedback gain”. In: *Renewable Energy* 205 (Jan. 2023), pp. 509–524. DOI: 10.1016/j.renene.2023.01.073. URL: https://doi.org/10.1016/j.renene.2023.01.073.

[39] Kaisa Miettinen. *Nonlinear multiobjective optimization*. Springer Science & Business Media, Jan. 1998. DOI: 10.1007/978-1-4615-5563-6. URL: https://doi.org/10.1007/978-1-4615-5563-6.

[40] Ciprian Ionu Morăra et al. “Analysis of the effect of fiber orientation on mechanical and elastic characteristics at axial stresses of GFRP used in wind turbine blades”. In: *Polymers* 15.4 (Feb. 2023), p. 861. DOI: 10.3390/polym15040861. URL: https://doi.org/10.3390/polym15040861.

[41] Anders Myhr et al. “Levelised cost of energy for offshore floating wind turbines in a life cycle perspective”. In: *Renewable Energy* 66 (Feb. 2014), pp. 714–728. DOI: 10.1016/j.renene.2014.01.017. URL: https://doi.org/10.1016/j.renene.2014.01.017.

[42] NREL. *ROSCO. Version 2.4.1*. 2021. URL: https://github.com/NREL/ROSCO.

[43] R.P.L. Nussen and D.R.V. Van Delft. *Alternative fatigue formulations for variable amplitude loading of fibre composites for wind turbine rotor blades*. Elsevier Ltd., Jan. 2003, pp. 563–574. DOI: 10.1016/s1566-1369(03)80125-5. URL: https://doi.org/10.1016/s1566-1369(03)80125-5.

[44] Peter Fogh Odgaard et al. “On using Pareto optimality to tune a linear model predictive controller for wind turbines”. In: *Renewable Energy* 87 (Oct. 2016), pp. 884–891. DOI: 10.1016/j.renene.2015.09.067. URL: https://doi.org/10.1016/j.renene.2015.09.067.

[45] Yongoon Oh et al. “Control algorithm of a floating wind turbine for reduction of tower loads and power fluctuation”. In: *International Journal of Precision Engineering and Manufacturing* 16.9 (Aug. 2015), pp. 2041–2048. DOI: 10.1007/s12541-015-0265-0. URL: https://doi.org/10.1007/s12541-015-0265-0.

[46] Joannes Olondriz et al. “A blade load feedback control for floating offshore wind turbines”. In: *Journal of Physics Conference Series* 1222.1 (May 2019), p. 012014. DOI: 10.1088/1742-6596/1222/1/012014. URL: https://doi.org/10.1088/1742-6596/1222/1/012014.

[47] Kranthi Kumar P and Ketan P Detroja. “Gain Scheduled PI controller design using Multi-Objective Reinforcement Learning”. In: *IFAC-PapersOnLine* 58.7 (Jan. 2024), pp. 132–137. DOI: 10.1016/j.ifacol.2024.08.023. URL: https://doi.org/10.1016/j.ifacol.2024.08.023.

[48] João Pacheco et al. “Experimental evaluation of fatigue in wind turbine blades with wake effects”. In: *Engineering Structures* 300 (Dec. 2023), p. 117140. DOI: 10.1016/j.engstruct.2023.117140. URL: https://doi.org/10.1016/j.engstruct.2023.117140.

[49] Erik L. Petersen et al. “Wind power meteorology. Part I: climate and turbulence”. In: *Wind Energy* 1.1 (Sept. 1998), pp. 2–22. DOI: 10.1002/(sici)1099-1824(199809)1:1. URL: https://doi.org/10.1002/(sici)1099-1824(199809)1:1%3C2::aid-we15%3E3.0.co;2-y.

[50] L. Pustina et al. “A novel resonant controller for sea-induced rotor blade vibratory loads reduction on floating offshore wind turbines”. In: *Renewable and Sustainable Energy Reviews* 173 (Dec. 2022), p. 113073. DOI: 10.1016/j.rser.2022.113073. URL: https://doi.org/10.1016/j.rser.2022.113073.

[51] Niklas Requate and Tobias Meyer. “Active control of the reliability of wind turbines”. In: *IFAC-PapersOnLine* 53.2 (Jan. 2020), pp. 12789–12796. DOI: 10.1016/j.ifacol.2020.12.1941. URL: https://doi.org/10.1016/j.ifacol.2020.12.1941.

[52] Yu Rie. *An offshore wind turbine is shown against a blue sky*. Mar. 2024. URL: https://unsplash.com/photos/a-wind-turbine-is-shown-against-a-blue-sky-MKjHHMbIO-I.

[53] T. Rubert et al. “Wind turbine lifetime extension decision-making based on structural health monitoring”. In: *Renewable Energy* 143 (May 2019), pp. 611–621. DOI: 10.1016/j.renene.2019.05.034. URL: https://doi.org/10.1016/j.renene.2019.05.034.

[54] Hector Sanchez et al. “Analysis of two modeling approaches for fatigue estimation and remaining useful life predictions of wind turbine blades”. In: *PHM Society European Conference* 3.1 (July 2016). DOI: 10.36001/phme.2016.v3i1.1640. URL: https://doi.org/10.36001/phme.2016.v3i1.1640.

[55] J. Schijve. *Fatigue of structures and materials*. Springer Science & Business Media, Dec. 2008.

[56] Andrew Scholbrock et al. “Lidar-enhanced wind turbine control: Past, present, and future”. In: *2022 American Control Conference (ACC)* (July 2016), pp. 1399–1406. DOI: 10.1109/acc.2016.7525113. URL: https://doi.org/10.1109/acc.2016.7525113.

[57] Harry Schulz, André Simoes, and Raquel Lobosco. *Hydrodynamics*. BoD – Books on Demand, Oct. 2011.

[58] Ahmad Sedaghat et al. “Determination of rated wind speed for maximum annual energy production of variable speed wind turbines”. In: *Applied Energy* 205 (Aug. 2017), pp. 781–789. DOI: 10.1016/j.apenergy.2017.08.079. URL: https://doi.org/10.1016/j.apenergy.2017.08.079.

[59] Mahmood Shafiee. “Extending the lifetime of offshore wind turbines: challenges and opportunities”. In: *Energies* 17.16 (Aug. 2024), p. 4191. DOI: 10.3390/en17164191. URL: https://doi.org/10.3390/en17164191.

[60] SINTEF. *Offshore wind - SIMA*. URL: https://www.sintef.no/en/software/sima/offshore-wind/.

[61] Massimo Sirigu et al. “The importance of aeroelasticity in estimating multiaxial fatigue behaviour of large floating offshore wind turbine blades”. In: *Heliyon* 10.4 (Feb. 2024), e26017. DOI: 10.1016/j.heliyon.2024.e26017. URL: https://doi.org/10.1016/j.heliyon.2024.e26017.

[62] Emil Smilden et al. “Site-specific controller design for monopile offshore wind turbines”. In: *Marine Structures* 61 (July 2018), pp. 503–523. DOI: 10.1016/j.marstruc.2018.03.002. URL: https://doi.org/10.1016/j.marstruc.2018.03.002.

[63] Yuan Song and Insu Paek. “Prediction and validation of the annual energy production of a wind turbine using WindSIM and a dynamic wind turbine model”. In: *Energies* 13.24 (Dec. 2020), p. 6604. DOI: 10.3390/en13246604. URL: https://doi.org/10.3390/en13246604.

[64] Asgeir J. Sørensen. *Marine Cybernetics: Towards Autonomous Marine Operations and Systems*. Lecture Notes, Report UK-18-76. Trondheim, Norway: Department of Marine Technology, Norwegian University of Science and Technology (NTNU), 2018.

[65] Carlos D.S. Souto et al. “Fatigue Damage Tool (FDT) - A tool for fatigue damage assessment according to design codes”. In: *Procedia Structural Integrity* 22 (Jan. 2019), pp. 376–385. DOI: 10.1016/j.prostr.2020.01.047. URL: https://doi.org/10.1016/j.prostr.2020.01.047.

[66] Clara M. St Martin et al. “Wind turbine power production and annual energy production depend on atmospheric stability and turbulence”. In: *Wind energy science* 1.2 (Nov. 2016), pp. 221–236. DOI: 10.5194/wes-1-221-2016. URL: https://doi.org/10.5194/wes-1-221-2016.

[67] Tyler Stehly, Patrick Duffy, and Daniel Mulas Hernando. *Cost of Wind Energy Review: 2024 Edition*. Tech. rep. NREL, Nov. 2024. DOI: 10.2172/2479271. URL: https://doi.org/10.2172/2479271.

[68] A. Subbulakshmi et al. “Recent advances in experimental and numerical methods for dynamic analysis of floating offshore wind turbines — An integrated review”. In: *Renewable and Sustainable Energy Reviews* 164 (May 2022), p. 112525. DOI: 10.1016/j.rser.2022.112525. URL: https://doi.org/10.1016/j.rser.2022.112525.

[69] Herbert Sutherland and John Mandell. “Optimized Goodman diagram for the analysis of fiberglass composites used in wind turbine blades”. In: *43rd AIAA Aerospace Sciences Meeting and Exhibit* (Jan. 2005). DOI: 10.2514/6.2005-196. URL: https://doi.org/10.2514/6.2005-196.

[70] Daniel Zalkind et al. “Floating wind turbine control optimization”. In: *Journal of Physics Conference Series* 2265.4 (May 2022), p. 042021. DOI: 10.1088/1742-6596/2265/4/042021. URL: https://doi.org/10.1088/1742-6596/2265/4/042021.

[71] Yuchen Zhang et al. “Aerodynamic and structural analysis for blades of a 15MW floating offshore wind turbine”. In: *Ocean Engineering* 287 (Sept. 2023), p. 115785. DOI: 10.1016/j.oceaneng.2023.115785. URL: https://doi.org/10.1016/j.oceaneng.2023.115785.

A

Code

A.1. Model Validation

The following script uses the output of SIMA for a standard constant wind condition simulation to validate the model and the functionality of the ROSCO controller.

Listing A.1: Model Validation Script

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the Excel file
file_path = "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
    Attachments/Desktop/THESIS/Specialization␣Project/Simulations/
    SIMA_Simulations/Constant_Wind_Sim/Constant_Wind_Sim_15MW.xlsx"
excel_data = pd.ExcelFile(file_path)

rated_wind_speed = 10.59
# Rotor diameter and radius
rotor_diameter = 240  # in meters
rotor_radius = rotor_diameter / 2  # in meters

# Initialize lists to store results
wind_speeds = []
mean_rotor_speeds = []
mean_aero_forces_x = []
mean_pitch_angles = []
mean_generator_torques = []
mean_generator_outputs = []
tip_speed_ratios = []

# Iterate through each sheet (each wind speed)
for sheet_name in excel_data.sheet_names:
    # Extract wind speed from the sheet name
    wind_speed = float(sheet_name.split("_")[1])
    wind_speeds.append(wind_speed)

    # Load the data from the sheet
    data = excel_data.parse(sheet_name, header=1)
    data = data[1:].reset_index(drop=True)  # Drop the units row
```

```
32
33
34
35      # Compute mean values for each variable
36      mean_rotor_speed = data['Rotor speed (rpm)'].mean()
37      mean_rotor_speeds.append(mean_rotor_speed)
38      mean_aero_forces_x.append(data['Aero force X-dir in shaft system'].mean()
            )
39      mean_pitch_angles.append(data['Pitch angle blade 1, Line: bl1foil'].mean
            ())
40      mean_generator_torques.append(data['Mechanical generator torque on LSS'].
            mean())
41      mean_generator_outputs.append(data['Electrical generator output'].mean())
42
43
44      # Compute rotor speed in radians per second
45      angular_velocity = mean_rotor_speed * (2 * np.pi / 60)  # Convert RPM to
            rad/s
46
47      # Compute TSR
48      tsr = (angular_velocity * rotor_radius) / wind_speed
49      tip_speed_ratios.append(tsr)
50
51  # Sort the results by wind speed
52  sorted_data = sorted(zip(wind_speeds, mean_rotor_speeds, mean_aero_forces_x,
        mean_pitch_angles, mean_generator_torques, mean_generator_outputs))
53  wind_speeds, mean_rotor_speeds, mean_aero_forces_x, mean_pitch_angles,
        mean_generator_torques, mean_generator_outputs = zip(*sorted_data)
54
55
56
57
58  # Plot Mean Rotor Speed vs Wind Speed
59  plt.figure(figsize=(8, 6))
60  plt.plot(wind_speeds, mean_rotor_speeds, '-', label="Mean Rotor Speed", color
        ='b')
61  plt.axvline(rated_wind_speed, color='k', linestyle='--', label="Rated Wind
        Speed (10.59 m/s)")
62  plt.xlabel("Wind Speed (m/s)")
63  plt.ylabel("Mean Rotor Speed (RPM)")
64  plt.grid(True)
65  plt.legend()
66  plt.show()
67
68  # Plot Mean Aerodynamic Force in X-Direction vs Wind Speed
69  plt.figure(figsize=(8, 6))
70  plt.plot(wind_speeds, mean_aero_forces_x, '-', label="Mean Thrust Force",
        color='r')
71  plt.axvline(rated_wind_speed, color='k', linestyle='--', label="Rated Wind
        Speed (10.59 m/s)")
72  plt.xlabel("Wind Speed (m/s)")
73  plt.ylabel("Mean Thrust Force (N)")
74  plt.grid(True)
75  plt.legend()
76  plt.show()
77
78  # Plot Mean Pitch Angle vs Wind Speed
79  plt.figure(figsize=(8, 6))
```

```
plt.plot(wind_speeds, mean_pitch_angles, '-', label="Mean Pitch Angle", color='g')
plt.axvline(rated_wind_speed, color='k', linestyle='--', label="Rated Wind Speed (10.59 m/s)")
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Mean Pitch Angle (degrees)")
plt.grid(True)
plt.legend()
plt.show()

# Plot Mean Generator Torque vs Wind Speed
plt.figure(figsize=(8, 6))
plt.plot(wind_speeds, mean_generator_torques, '-', label="Mean Generator Torque", color='m')
plt.axvline(rated_wind_speed, color='k', linestyle='--', label="Rated Wind Speed (10.59 m/s)")
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Mean Generator Torque (Nm)")
plt.grid(True)
plt.legend()
plt.show()

# Plot Mean Electrical Generator Output vs Wind Speed
plt.figure(figsize=(8, 6))
plt.plot(wind_speeds, mean_generator_outputs, '-', label="Mean Generator Output", color='c')
plt.axvline(rated_wind_speed, color='k', linestyle='--', label="Rated Wind Speed (10.59 m/s)")
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Mean Electrical Generator Output (W)")
plt.grid(True)
plt.legend()
plt.show()

# Plot Tip Speed Ratio vs Wind Speed
plt.figure(figsize=(8, 6))
plt.plot(wind_speeds, tip_speed_ratios, '-', label="Tip Speed Ratio (TSR)", color='b')
plt.axvline(10.59, color='k', linestyle='--', label="Rated Wind Speed (10.59 m/s)")
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Tip Speed Ratio (TSR)")
plt.grid(True)
plt.legend()
plt.show()
```

A.2. Utsira Nord Data Analysis

The following Python script processes metocean data for wind and wave characteristic analysis.

Listing A.2: Metocean Data Analysis Script

```
import xarray as xr
import pandas as pd
import numpy as np
# import glob
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from scipy.stats import weibull_min

# Define the folder path containing all NetCDF files
folder_path = "C:/Users/toon4/OneDrive - Delft University of Technology/
    Attachments/Desktop/THESIS/Utsira Nord Data/"

# List of all NetCDF files for 2018-2022
file_names = [
    "metOcean_UN_2018.nc",
    "metOcean_UN_2019.nc",
    "metOcean_UN_2020.nc",
    "metOcean_UN_2021.nc",
    "metOcean_UN_2022.nc"
]

# Load and merge datasets
datasets = [xr.open_dataset(folder_path + file) for file in file_names]
ds = xr.concat(datasets, dim="time")  # Merge along the time dimension

# Set your location and height index
location_index = 50  # Change this index as needed
height_index = 4  # For 150 meters height

# Extract the wind speed (U), significant wave height (hs), and peak wave
    period (tp)
wind_speed = ds["U"][height_index, location_index, :].values.flatten()
significant_wave_height = ds["hs"][location_index, :].values.flatten()
peak_wave_period = ds["tp"][location_index, :].values.flatten()
u_star = ds["u_star"][location_index, :].values.flatten()

# Filter out wind speeds outside the operating range (3 to 25 m/s)
valid_indices = (wind_speed >= 3) & (wind_speed <= 25)
wind_speed = wind_speed[valid_indices]
significant_wave_height = significant_wave_height[valid_indices]
peak_wave_period = peak_wave_period[valid_indices]
u_star = u_star[valid_indices]

# Define wind speed bins (1 m/s increments from 3 to 25)
wind_speed_bins = np.arange(3, 26, 1)  # Bins: 3-4, 4-5, ..., 24-25
wind_speed_labels = [f'{i}-{i+1} m/s' for i in range(3, 25)]

# Assign each wind speed to a category (bin)
wind_speed_categories = pd.cut(wind_speed, bins=wind_speed_bins, labels=
    wind_speed_labels, include_lowest=True, right=False)

# Convert to categorical with ordering
wind_speed_categories = pd.Categorical(wind_speed_categories, categories=
    wind_speed_labels, ordered=True)

# Create a DataFrame with the filtered data
data = pd.DataFrame({
    'Wind Speed (m/s)': wind_speed,
    'Wind Speed Category': wind_speed_categories,
    'Significant Wave Height (m)': significant_wave_height,
    'Peak Wave Period (s)': peak_wave_period,
    'Friction Velocity (u_star) (m/s)': u_star
})

# Calculate the likelihood (probability) of each wind speed category
```

```
category_counts = data['Wind Speed Category'].value_counts(normalize=True).sort_index() * 100  # percentage

# Calculate the mean significant wave height and peak wave period for each wind speed category
category_means = data.groupby('Wind Speed Category').agg({
    'Significant Wave Height (m)': 'mean',
    'Peak Wave Period (s)': 'mean'
})

# Combine the likelihood and mean values into a final table
distribution_table = pd.concat([category_counts, category_means], axis=1)
distribution_table.columns = ['Likelihood (%)', 'Mean Significant Wave Height (m)', 'Mean Peak Wave Period (s)']

# Ensure full table display
pd.set_option('display.expand_frame_repr', False)  # Prevents column truncation
pd.set_option('display.max_columns', None)  # Ensures all columns are shown
pd.set_option('display.float_format', '{:.2f}'.format)  # Consistent decimal format
# Calculate the standard deviation of the wind speed (u-component) within each wind speed category
category_std = data.groupby('Wind Speed Category')['Wind Speed (m/s)'].std()

# Calculate the mean wind speed within each category
category_mean = data.groupby('Wind Speed Category')['Wind Speed (m/s)'].mean()

# Calculate turbulence intensity (TI) for each category
turbulence_intensity = category_std / category_mean

# Add the turbulence intensity to the distribution table
distribution_table['Turbulence Intensity (%)'] = turbulence_intensity*100

# Display the updated distribution table with turbulence intensity
#print(distribution_table)


# Plots

plt.figure(figsize=(10, 5))
plt.bar(distribution_table.index, distribution_table['Likelihood (%)'], color='royalblue', edgecolor='black')
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Likelihood (%)")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(data['Wind Speed (m/s)'], data['Significant Wave Height (m)'], alpha=0.3, color='darkblue')
plt.xlabel("Wind Speed (m/s)")
plt.ylabel("Significant Wave Height (m)")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

plt.figure(figsize=(12, 6))
```

```
sns.boxplot(x=data['Wind Speed Category'], y=data['Significant Wave Height (m)'], palette="coolwarm")
plt.xlabel("Wind Speed Category (m/s)")
plt.ylabel("Significant Wave Height (m)")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(x=data['Wind Speed Category'], y=data['Peak Wave Period (s)'], palette="coolwarm")
plt.xlabel("Wind Speed Category (m/s)")
plt.ylabel("Peak Wave Period (s)")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()


# Dataset with individual wind speed values
def get_wave_params_for_speed(df, target_speed, delta=0.5):
    subset = df[(df["Wind Speed (m/s)"] >= target_speed - delta) &
                (df["Wind Speed (m/s)"] <= target_speed + delta)]
    return subset["Significant Wave Height (m)"].mean(), subset["Peak Wave Period (s)"].mean(), subset["Friction Velocity (u_star) (m/s)"].mean()

# Define exact wind speeds for SIMA
exact_wind_speeds = np.arange(3, 26, 1)

# Compute values per wind speed
sima_conditions = pd.DataFrame({
    'Wind Speed (m/s)': exact_wind_speeds,
    'Significant Wave Height (m)': [get_wave_params_for_speed(data, ws)[0] for ws in exact_wind_speeds],
    'Peak Wave Period (s)': [get_wave_params_for_speed(data, ws)[1] for ws in exact_wind_speeds],
    'Friction Velocity (u_star) (m/s)': [get_wave_params_for_speed(data, ws)[2] for ws in exact_wind_speeds]
})

# print(sima_conditions)

# Fit Weibull distribution to all wind speed data (before filtering)
shape, loc, scale = weibull_min.fit(ds["U"][height_index, location_index, :].values.flatten(), floc=0)

# Use the actual full range of wind speed data
x = np.linspace(ds["U"][height_index, location_index, :].min(), ds["U"][height_index, location_index, :].max(), 100)
weibull_pdf = weibull_min.pdf(x, shape, loc, scale)

# Plot Weibull distribution
plt.figure(figsize=(10, 5))
plt.plot(x, weibull_pdf, 'r-', linewidth=2, label=f'Weibull Fit (={scale:.2f},k={shape:.2f})')
plt.xlim(0, x.max())
plt.ylim(0, max(weibull_pdf) * 1.1)
# Labels and title
plt.xlabel("Wind Speed (m/s)")
```

```
plt.ylabel("Probability Density")
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

A.3. Annual Energy Production

The following Python script uses the SIMA electrical power output and calculates the AEP given the Weibull fit of the wind speed.

Listing A.3: Annual Energy Production Script

```
import pandas as pd
import numpy as np

# Weibull distribution parameters
lambda_weibull = 11.22
k_weibull = 1.93
hours_per_year = 8760

# File paths for different controller cases
controller_files = {
    "Baseline": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_Baseline_Output.xlsx",
    "Controller1": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C1_Output.xlsx",
    "Controller2": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C2_Output.xlsx",
    "Controller3": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C3_Output.xlsx",
    "Controller4": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C4_Output.xlsx",
    "Controller5": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C5_Output_new.xlsx",
    "Controller6": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C6_Output.xlsx",
    "Controller7": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C7_Output.xlsx",
    "Controller8": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C8_Output.xlsx",
    "Controller9": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C9_Output.xlsx",
    "Controller10": "C:/Users/toon4/OneDrive - Delft University of Technology
        /Attachments/Desktop/THESIS/Results/Cases/SIMA_C10_Output.xlsx",
    "Controller11": "C:/Users/toon4/OneDrive - Delft University of Technology
        /Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C11_Output.xlsx"
        ,
    "Controller12": "C:/Users/toon4/OneDrive - Delft University of Technology
        /Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C12_Output.xlsx"
        ,
    "Controller13": "C:/Users/toon4/OneDrive - Delft University of Technology
        /Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C13_Output.xlsx"
        ,
    "Controller14": "C:/Users/toon4/OneDrive - Delft University of Technology
        /Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C14_Output.xlsx"


}

```

```
# Weibull PDF
def weibull_pdf(U, lambda_weibull, k_weibull):
    return (k_weibull / lambda_weibull) * (U / lambda_weibull)**(k_weibull -
        1) * np.exp(-(U / lambda_weibull)**k_weibull)

# Store results
results = []

for controller_name, file_path in controller_files.items():
    excel_data = pd.ExcelFile(file_path)

    wind_speeds = []
    mean_generator_outputs = []

    for sheet_name in excel_data.sheet_names:
        data = excel_data.parse(sheet_name, header=1)
        data = data[1:].reset_index(drop=True)
        data = data.iloc[600:]

        wind_speed = data['Incoming␣wind␣speed␣X-dir␣in␣shaft␣system'].mean()
        wind_speeds.append(wind_speed)

        mean_power = data['Electrical␣generator␣output'].clip(upper=15e6).
            mean()
        mean_generator_outputs.append(mean_power)

    # Sort data by wind speed
    wind_speeds, mean_generator_outputs = zip(*sorted(zip(wind_speeds,
        mean_generator_outputs)))

    # AEP calculation
    AEP = 0
    for i in range(len(wind_speeds) - 1):
        U_lower = wind_speeds[i]
        U_upper = wind_speeds[i + 1]
        P_U = mean_generator_outputs[i]

        pdf_lower = weibull_pdf(U_lower, lambda_weibull, k_weibull)
        pdf_upper = weibull_pdf(U_upper, lambda_weibull, k_weibull)

        integration = (P_U * pdf_lower + P_U * pdf_upper) * (U_upper -
            U_lower) / 2
        AEP += integration

    AEP_kWh = AEP * hours_per_year / 1000

    results.append({
        "Controller": controller_name,
        "AEP_kWh": AEP_kWh
    })

# Display results
results_df = pd.DataFrame(results)
print(results_df)
```

The following script plots the characteristic power curve among other curves, which are used to investigate behaiour in energy production, used in the discussion.

Listing A.4: Power Curve Script for Discussion

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Increase global font sizes
plt.rcParams['axes.labelsize'] = 15    # axis titles
plt.rcParams['xtick.labelsize'] = 10   # x-axis numbers
plt.rcParams['ytick.labelsize'] = 10   # y-axis numbers
plt.rcParams['legend.fontsize'] = 15   # legend text

# Choose controllers to compare and wind speed
controllers_to_plot = ["Baseline", "Controller2", "Controller5"]

# File paths for different controller cases
controller_files = {
    "Baseline": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_Baseline_Output.xlsx",
    "Controller1": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C1_Output.xlsx",
    "Controller2": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C2_Output.xlsx",
    "Controller3": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C3_Output.xlsx",
    "Controller4": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C4_Output.xlsx",
    "Controller5": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C5_Output_new.xlsx",
    "Controller6": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C6_Output.xlsx",
    "Controller7": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C7_Output.xlsx",
    "Controller8": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C8_Output.xlsx",
    "Controller9": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C9_Output.xlsx",
    "Controller10": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C10_Output.xlsx",
    "Controller11": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C11_Output.xlsx",
    "Controller12": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C12_Output.xlsx",
    "Controller13": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C13_Output.xlsx",
    "Controller14": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C14_Output.xlsx"

}

# Initialize a dictionary to store results for each controller
controller_data = {}

# Function to extract data for a given controller
def extract_data(file_path):
    excel_data = pd.ExcelFile(file_path)
    rated_wind_speed = 10.59
    rotor_diameter = 240  # in meters
```

```
    rotor_radius = rotor_diameter / 2  # in meters

    # Initialize lists to store results for the current controller
    wind_speeds = []
    mean_rotor_speeds = []
    mean_pitch_angles = []
    mean_generator_torques = []
    mean_output = []
    tip_speed_ratios = []

    # Iterate through each sheet (each wind speed)
    for sheet_name in excel_data.sheet_names:
        data = excel_data.parse(sheet_name, header=1)
        data = data[1:].reset_index(drop=True)
        data = data.iloc[600:]

        wind_speed = data['Incoming wind speed X-dir in shaft system'].mean()
        wind_speeds.append(wind_speed)

        # Compute mean values for each variable
        mean_rotor_speed = data['Rotor speed (rpm)'].mean()
        mean_rotor_speeds.append(mean_rotor_speed)
        mean_pitch_angles.append(data['Pitch angle blade 1, Line: bl1foil'].
            mean())
        mean_generator_torques.append(data['Mechanical generator torque on
            LSS'].mean() / 1e6)  # Convert Nm to MNm
        mean_output.append(data["Electrical generator output"].mean() / 1e6)

        # Compute rotor speed in radians per second
        angular_velocity = mean_rotor_speed * (2 * np.pi / 60)  # Convert RPM
             to rad/s

        # Compute TSR
        tsr = (angular_velocity * rotor_radius) / wind_speed
        tip_speed_ratios.append(tsr)

    # Sort the results by wind speed and convert to lists
    sorted_data = sorted(zip(wind_speeds, mean_rotor_speeds,
        mean_pitch_angles,
                             mean_generator_torques, tip_speed_ratios))
    wind_speeds, mean_rotor_speeds, mean_pitch_angles, mean_generator_torques
        , tip_speed_ratios = map(list, zip(*sorted_data))

    return wind_speeds, mean_rotor_speeds, mean_pitch_angles,
        mean_generator_torques, mean_output, tip_speed_ratios

# Load data for the selected controllers
for controller in controllers_to_plot:
    file_path = controller_files[controller]
    wind_speeds, mean_rotor_speeds, mean_pitch_angles, mean_generator_torques
        , mean_output, tip_speed_ratios = extract_data(file_path)
    controller_data[controller] = {
        "wind_speeds": wind_speeds,
        "mean_rotor_speeds": mean_rotor_speeds,
        "mean_pitch_angles": mean_pitch_angles,
        "mean_generator_torques": mean_generator_torques,
        "mean_output": mean_output,
        "tip_speed_ratios": tip_speed_ratios
    }
```

```
94
95 # Plot Mean Rotor Speed vs Wind Speed for selected controllers
96 plt.figure(figsize=(8, 6))
97 for controller in controllers_to_plot:
98     plt.plot(controller_data[controller]["wind_speeds"], controller_data[
       controller]["mean_rotor_speeds"], label=f"{controller}")
99 plt.axvline(10.59, color='k', linestyle='--', label="Rated␣Wind␣Speed␣(10.59␣
   m/s)")
100 plt.xlabel("Wind␣Speed␣(m/s)")
101 plt.ylabel("Mean␣Rotor␣Speed␣(RPM)")
102 plt.legend()
103 plt.grid(True)
104 plt.show()
105
106 # Plot Mean Pitch Angle vs Wind Speed for selected controllers
107 plt.figure(figsize=(8, 6))
108 for controller in controllers_to_plot:
109     plt.plot(controller_data[controller]["wind_speeds"], controller_data[
       controller]["mean_pitch_angles"], label=f"{controller}")
110 plt.axvline(10.59, color='k', linestyle='--')
111 plt.xlabel("Wind␣Speed␣(m/s)")
112 plt.ylabel("Mean␣Pitch␣Angle␣(degrees)")
113 plt.legend()
114 plt.grid(True)
115 plt.show()
116
117 # Plot Mean Generator Torque vs Wind Speed for selected controllers
118 plt.figure(figsize=(8, 6))
119 for controller in controllers_to_plot:
120     plt.plot(controller_data[controller]["wind_speeds"], controller_data[
       controller]["mean_generator_torques"], label=f"{controller}")
121 plt.axvline(10.59, color='k', linestyle='--')
122 plt.xlabel("Wind␣Speed␣(m/s)")
123 plt.ylabel("Mean␣Generator␣Torque␣(MNm)")
124 plt.legend()
125 plt.grid(True)
126 plt.show()
127
128 # Plot Mean Output vs Wind Speed for selected controllers
129 plt.figure(figsize=(8, 6), dpi = 500)
130 for controller in controllers_to_plot:
131     plt.plot(controller_data[controller]["wind_speeds"], controller_data[
       controller]["mean_output"], label=f"{controller}")
132 plt.axvline(10.59, color='k', linestyle='--')
133 plt.xlabel("Wind␣Speed␣(m/s)")
134 plt.ylabel("Mean␣Electrical␣Output␣(MW)")
135 plt.legend(loc='upper␣center', bbox_to_anchor=(0.5, -0.2), ncol=3, frameon=
   False)
136 plt.grid(True)
137 plt.show()
138
139 # Plot Tip Speed Ratio vs Wind Speed for selected controllers
140 plt.figure(figsize=(8, 6))
141 for controller in controllers_to_plot:
142     plt.plot(controller_data[controller]["wind_speeds"], controller_data[
       controller]["tip_speed_ratios"], label=f"{controller}")
143 plt.axvline(10.59, color='k', linestyle='--')
144 plt.xlabel("Wind␣Speed␣(m/s)")
145 plt.ylabel("Tip␣Speed␣Ratio")
```

```
plt.legend()
plt.grid(True)
plt.show()
```

A.4. Blade Fatifue Life

The following Python script uses the SIMA aerodynamic moment output and calculates the estimated blade fatigue life given the Weibull fit of the wind speed.

Listing A.5: Blade Fatigue Life Estimation Script

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import rainflow

# Blade geometry and material properties
D = 5.2
R = D / 2
t = 0.1
R_o = R
R_i = R_o - t
I_x = I_y = (np.pi / 4) * (R_o**4 - R_i**4)
J = I_x + I_y

# Blade and environment constants
m_blade = 65000.0
blade_R = 120.0
r_cg = 2 / 3 * blade_R
g = 9.81

# Weibull parameters
lambda_weibull = 11.22
k_weibull = 1.93

# Goodman + Basquin parameters
UTS = 400.0  # MPa
m = 7.0
a = 80.0

# Controller file paths
controller_files = {
    "Baseline": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_Baseline_Output.xlsx",
    "Controller1": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C1_Output.xlsx",
    "Controller2": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C2_Output.xlsx",
    "Controller3": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C3_Output.xlsx",
    "Controller4": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C4_Output.xlsx",
    "Controller5": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C5_Output_new.xlsx",
    "Controller6": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C6_Output.xlsx",
    "Controller7": "C:/Users/toon4/OneDrive - Delft University of Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C7_Output.xlsx",
```

```
    "Controller8": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C8_Output.xlsx",
    "Controller9": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C9_Output.xlsx",
    "Controller10": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C10_Output.xlsx",
    "Controller11": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C11_Output.xlsx",
    "Controller12": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C12_Output.xlsx",
    "Controller13": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C13_Output.xlsx",
    "Controller14": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C14_Output.xlsx"
}

def weibull_pdf(U, lam, k):
    return (k / lam) * (U / lam)**(k - 1) * np.exp(-(U / lam)**k)

def calculate_stresses(M_x, M_y, M_z):
    sigma_y = M_y * R / I_y
    sigma_z = M_z * R / I_x
    tau = M_x * R / J
    sigma_vm = np.sqrt(sigma_y**2 + sigma_z**2 + 3 * tau**2)
    return sigma_y, sigma_z, tau, sigma_vm

def process_controller(name, filepath):
    excel_data = pd.ExcelFile(filepath)
    stress_results = {}

    for sheet_name in excel_data.sheet_names:
        data = excel_data.parse(sheet_name, header=1)
        data = data[1:].reset_index(drop=True)
        data = data.iloc[600:]

        M_x = data["Aero moment around X-axis in shaft system"]
        M_y = data["Aero moment around Y-axis in shaft system"]
        M_z_aero = data["Aero moment around Z-axis in shaft system"]
        rpm = data["Rotor speed (rpm)"].astype(float).values

        n = len(M_z_aero)
        dt = 600 / n
        t = np.arange(n) * dt
        Omega = (rpm / 60) * 2 * np.pi
        M_z_grav = m_blade * g * r_cg * np.sin(Omega * t)
        M_z_total = M_z_aero.values + M_z_grav

        sigma_vm = np.zeros(n)
        for i in range(n):
            _, _, _, sigma_vm[i] = calculate_stresses(M_x.iloc[i], M_y.iloc[i], M_z_total[i])

        stress_results[sheet_name] = sigma_vm

    # Weight by Weibull PDF
    weights = {
```

```
        k: weibull_pdf(float(k), lambda_weibull, k_weibull)
        for k in stress_results.keys()
    }
    total_w = sum(weights.values())
    weights = {k: v / total_w for k, v in weights.items()}

    sigma_eff = np.zeros(len(next(iter(stress_results.values()))))
    for k, sigma in stress_results.items():
        sigma_eff += weights[k] * np.array(sigma)

    # Rainflow cycles
    cycles = list(rainflow.extract_cycles(sigma_eff))
    df_cycles = pd.DataFrame([c[:3] for c in cycles], columns=["range", "mean
        ", "count"])

    # Bin
    n_bins = 100
    bins = np.linspace(df_cycles["range"].min(), df_cycles["range"].max(),
        n_bins+1)
    df_cycles["bin"] = pd.cut(df_cycles["range"], bins=bins)

    grouped = (
        df_cycles.groupby("bin", observed=True)
        .apply(lambda g: pd.Series({
            "count": g["count"].sum(),
            "mean_st": np.average(g["mean"], weights=g["count"])
        }))
        .reset_index()
    )

    # Damage and life
    S_range = np.array([iv.mid for iv in grouped["bin"]]) / 1e6
    S_mean = grouped["mean_st"].to_numpy() / 1e6
    counts = grouped["count"].to_numpy()
    S_amp = S_range / 2
    denom = np.clip(1 - S_mean / UTS, 0.1, None)
    S_eq = S_amp / denom
    N_i = 0.5 * (a / S_eq)**m
    d_i = counts / N_i
    D_blk = d_i.sum()
    blocks_per_year = 365 * 24 * 6
    life_years = 1.0 / (D_blk * blocks_per_year)

    print(f"{name}: Estimated fatigue life = {life_years:.1f} years")

    return name, life_years

# Run the analysis for all controller files
results = {}
for name, path in controller_files.items():
    controller_name, life = process_controller(name, path)
    results[controller_name] = life

# Optional: Plot all fatigue lives
plt.figure(figsize=(12, 6))
plt.bar(results.keys(), results.values())
plt.ylabel("Estimated Fatigue Life (Years)")
plt.xticks(rotation=45)
plt.title("Fatigue Life Comparison Across Controllers")
```

```
plt.grid(True)
plt.tight_layout()
plt.show()
```

The following script is used to plot the stress amplitude and cycle counts for controller configurations. This is used for the additional plots in the discussion.

Listing A.6: Blade Stress Amplitude and Cycles Script for Discussion

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import rainflow

# Increase global font sizes
plt.rcParams['axes.labelsize'] = 25    # axis titles
plt.rcParams['xtick.labelsize'] = 20   # x-axis numbers
plt.rcParams['ytick.labelsize'] = 20   # y-axis numbers
plt.rcParams['legend.fontsize'] = 25   # legend text

# Blade geometry and material properties
D = 5.2
R = D / 2
t = 0.1
R_o = R
R_i = R_o - t
I_x = I_y = (np.pi / 4) * (R_o**4 - R_i**4)
J = I_x + I_y

# Blade and environment constants
m_blade = 65000.0
blade_R = 120.0
r_cg = 2 / 3 * blade_R
g = 9.81

# Weibull parameters
lambda_weibull = 11.22
k_weibull = 1.93

# Goodman + Basquin parameters
UTS = 400.0  # MPa
m = 7.0
a = 80.0

# Controller file paths
controller_files = {
    "Baseline": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_Baseline_Output.xlsx",
    "Controller1": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C1_Output.xlsx",
    "Controller2": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C2_Output.xlsx",
    "Controller3": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C3_Output.xlsx",
    "Controller4": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C4_Output.xlsx",
    "Controller5": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C5_Output_new.xlsx",
    "Controller6": "C:/Users/toon4/OneDrive␣-␣Delft␣University␣of␣Technology/
        Attachments/Desktop/THESIS/Results/Cases/SIMA_C6_Output.xlsx",
```

```
    "Controller7": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C7_Output.xlsx",
    "Controller8": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C8_Output.xlsx",
    "Controller9": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C9_Output.xlsx",
    "Controller10": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/SIMA_C10_Output.xlsx",
    "Controller11": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C11_Output.xlsx",
    "Controller12": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C12_Output.xlsx",
    "Controller13": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C13_Output.xlsx",
    "Controller14": "C:/Users/toon4/OneDrive - Delft University of Technology/Attachments/Desktop/THESIS/Results/Cases/extra/SIMA_C14_Output.xlsx"

}

def weibull_pdf(U, lam, k):
    return (k / lam) * (U / lam)**(k - 1) * np.exp(-(U / lam)**k)

def calculate_stresses(M_x, M_y, M_z):
    sigma_y = M_y * R / I_y
    sigma_z = M_z * R / I_x
    tau = M_x * R / J
    sigma_vm = np.sqrt(sigma_y**2 + sigma_z**2 + 3 * tau**2)
    return sigma_y, sigma_z, tau, sigma_vm

def process_controller(name, filepath):
    excel_data = pd.ExcelFile(filepath)
    stress_results = {}

    for sheet_name in excel_data.sheet_names:
        data = excel_data.parse(sheet_name, header=1)
        data = data[1:].reset_index(drop=True)
        data = data.iloc[600:]

        M_x = data["Aero moment around X-axis in shaft system"]
        M_y = data["Aero moment around Y-axis in shaft system"]
        M_z_aero = data["Aero moment around Z-axis in shaft system"]
        rpm = data["Rotor speed (rpm)"].astype(float).values

        n = len(M_z_aero)
        dt = 600 / n
        t = np.arange(n) * dt
        Omega = (rpm / 60) * 2 * np.pi
        M_z_grav = m_blade * g * r_cg * np.sin(Omega * t)
        M_z_total = M_z_aero.values + M_z_grav

        sigma_vm = np.zeros(n)
        for i in range(n):
            _, _, _, sigma_vm[i] = calculate_stresses(M_x.iloc[i], M_y.iloc[i], M_z_total[i])

        stress_results[sheet_name] = sigma_vm
```

```
92
93      # Weight by Weibull PDF
94      weights = {
95          k: weibull_pdf(float(k), lambda_weibull, k_weibull)
96          for k in stress_results.keys()
97      }
98      total_w = sum(weights.values())
99      weights = {k: v / total_w for k, v in weights.items()}
100
101     sigma_eff = np.zeros(len(next(iter(stress_results.values()))))
102     for k, sigma in stress_results.items():
103         sigma_eff += weights[k] * np.array(sigma)
104
105     # Rainflow cycles
106     cycles = list(rainflow.extract_cycles(sigma_eff))
107     df_cycles = pd.DataFrame([c[:3] for c in cycles], columns=["range", "mean
            ", "count"])
108
109     # Create evenly spaced bins for stress amplitudes
110     n_bins = 100
111     bins = np.linspace(df_cycles["range"].min(), df_cycles["range"].max(),
            n_bins+1)
112     df_cycles["bin"] = pd.cut(df_cycles["range"], bins=bins)
113
114     # Group data by bin and sum the cycle counts
115     grouped = (
116         df_cycles.groupby("bin", observed=True)
117         .apply(lambda g: pd.Series({
118             "count": g["count"].sum(),
119             "mean_st": np.average(g["mean"], weights=g["count"])
120         }))
121         .reset_index()
122     )
123
124     return grouped
125
126
127
128 def plot_comparison_bar(name1, grouped1, name2, grouped2):
129     # Plotting the stress amplitudes vs cycle count for two controllers
130     plt.figure(figsize=(12, 6), dpi=300)
131
132     # Get the left edge of each bin for both controllers
133     bin_edges_1 = grouped1["bin"].apply(lambda x: x.left).astype(float).
            values
134     bin_edges_2 = grouped2["bin"].apply(lambda x: x.left).astype(float).
            values
135
136     # Get bin widths for both controllers
137     bin_widths_1 = np.diff(grouped1["bin"].apply(lambda x: x.right).astype(
            float).values)
138     bin_widths_2 = np.diff(grouped2["bin"].apply(lambda x: x.right).astype(
            float).values)
139
140     # Plot bars for each bin
141     plt.bar(bin_edges_1, grouped1["count"], width=bin_widths_1[0], align='
            edge', edgecolor="black", alpha=0.9, label=name1)
142     plt.bar(bin_edges_2, grouped2["count"], width=bin_widths_2[0], align='
            edge', edgecolor="black", alpha=0.4, label=name2)
```

```
143
144     # Labels and title
145     plt.xlabel("Stress␣Amplitude␣(Pa)")
146     plt.ylabel("Number␣of␣Cycles")
147
148     # Show legend
149     plt.legend(loc='upper␣center', bbox_to_anchor=(0.5, -0.2), ncol=3,
            frameon=False)
150
151
152     # Grid and tight layout for clarity
153     plt.grid(True)
154     plt.tight_layout()
155     plt.xlim(right=2e7)
156     plt.show()
157
158 # Load data and process for Baseline, Controller 2, and Controller 3
159 controller_2_path = controller_files["Controller2"]
160 controller_12_path = controller_files["Controller12"]
161 baseline_path = controller_files["Baseline"]
162
163 # Process Baseline, Controller 2, and Controller 3
164 grouped_baseline = process_controller("Baseline", baseline_path)
165 grouped_controller_2 = process_controller("Controller2", controller_2_path)
166 grouped_controller_12 = process_controller("Controller12", controller_12_path
        )
167
168 # Plot comparison for Baseline vs Controller 2
169 plot_comparison_bar("Baseline", grouped_baseline, "Controller␣2",
        grouped_controller_2)
170
171 # Plot comparison for Baseline vs Controller 3
172 plot_comparison_bar("Baseline", grouped_baseline, "Controller␣1.2",
        grouped_controller_12)
173
174
175
176 def quantify_stress_ranges(grouped, threshold=0.5e7):
177     bin_centers = grouped["bin"].apply(lambda x: (x.left + x.right) / 2).
            astype(float)
178     low_stress = grouped["count"][bin_centers < threshold].sum()
179     high_stress = grouped["count"][bin_centers >= threshold].sum()
180     total = low_stress + high_stress
181     return {
182         "low_stress": low_stress,
183         "high_stress": high_stress,
184         "low_stress_pct": low_stress / total * 100,
185         "high_stress_pct": high_stress / total * 100
186     }
187
188 baseline_stats = quantify_stress_ranges(grouped_baseline)
189 controller2_stats = quantify_stress_ranges(grouped_controller_2)
190 controller12_stats = quantify_stress_ranges(grouped_controller_12)
191
192 print("Baseline:", baseline_stats)
193 print("Controller␣2:", controller2_stats)
194 print("Controller␣1.2:", controller12_stats)
```

A.5. Pareto Front

The following Python script uses the calculated AEP and blade fatigue life to plot the Pareto front. Additionally, the gain in total energy is plotted.

Listing A.7: Pareto Front Script

```
import matplotlib.pyplot as plt

controllers = [
    'B', 'C1', 'C1.1', 'C1.2', 'C1.3', 'C1.4', 'C2', 'C3', 'C4', 'C5', 'C6',
        'C7', 'C8', 'C9', 'C10'

]

aep = [val / 1000 for val in [
    71643010.00, 70373040.00, 71115980.00, 70054670.00, 70367790.00,
        70536660.00, 69139390.00, 70448970.00,
    71450300.00, 71578610.00, 71402970.00, 70719970.00,
    70651920.00, 71185950.00, 71431010.00

]]

fatigue_life = [
    25.7, 31.6, 29.1, 34.7, 33.7, 34.5 , 25.6, 34.5, 29.3, 29.2, 30.4, 34.4,
        26.7, 29.7, 26.7

]


# Plot
plt.figure(figsize=(10, 7), dpi=300)

# Plot all except B
for i in range(1, len(controllers)):
    plt.scatter(fatigue_life[i], aep[i], color='blue', s=80)
    plt.annotate(controllers[i], (fatigue_life[i], aep[i]), textcoords="
        offset␣points", xytext=(-5, 0), ha='right', fontsize=15)

# Highlight baseline (B)
plt.scatter(fatigue_life[0], aep[0], color='red', s=80)
plt.annotate('B', (fatigue_life[0], aep[0]), textcoords="offset␣points",
    xytext=(-5, 3), ha='right', fontsize=15)

# Labels and grid
plt.xlabel("Fatigue␣Life␣[years]", fontsize=20)
plt.ylabel("AEP␣[MWh]", fontsize=20)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.tight_layout()
plt.show()

# Baseline total energy production
baseline_energy = aep[0] * fatigue_life[0]
relative_energy = [(a * f - baseline_energy) / baseline_energy * 100 for a, f
     in zip(aep, fatigue_life)]

# Colors based on increase/decrease
colors = ['green' if val >= 0 else 'red' for val in relative_energy[1:]]
```

```
48
49 # Print the relative energy change for each controller
50 for controller, change in zip(controllers[1:], relative_energy[1:]):
51     print(f"Controller␣{controller}:␣{change:.2f}%")
52
53
54 # Bar plot
55 plt.figure(figsize=(10, 7), dpi = 300)
56 bars = plt.bar(controllers[1:], relative_energy[1:], color=colors)
57 plt.axhline(0, color='black', linewidth=1)
58 plt.ylabel('Change␣in␣Total␣Energy␣Production␣[%]', fontsize=20)
59 plt.xlabel('Controller␣Setting', fontsize=20)
60 plt.xticks(rotation=45)
61 plt.xticks(fontsize=15)
62 plt.yticks(fontsize=15)
63 plt.grid(axis='y', linestyle='--', alpha=0.7)
64 plt.tight_layout()
65 plt.show()
66
67
68
69 # Apply the sensitivity constraint (maximum fatigue life of 30 years)
70 adjusted_fatigue_life = [f if f <= 28.27 else 28.27 for f in fatigue_life]
71
72
73 # Recalculate the total energy production with the new adjusted fatigue life
       values
74 relative_energy_adjusted = [(a * f - baseline_energy) / baseline_energy * 100
        for a, f in zip(aep, adjusted_fatigue_life)]
75
76 # Colors based on increase/decrease
77 colors_adjusted = ['green' if val >= 0 else 'red' for val in
       relative_energy_adjusted[1:]]
78
79 # Bar plot
80 plt.figure(figsize=(10, 7), dpi=300)
81 bars_adjusted = plt.bar(controllers[1:], relative_energy_adjusted[1:], color=
       colors_adjusted)
82 plt.axhline(0, color='black', linewidth=1)
83 plt.ylabel('Change␣in␣Total␣Energy␣Production␣[%]', fontsize=20)
84 plt.xlabel('Controller␣Setting', fontsize=20)
85 plt.xticks(rotation=45)
86 plt.xticks(fontsize=15)
87 plt.yticks(fontsize=15)
88 plt.grid(axis='y', linestyle='--', alpha=0.7)
89 plt.tight_layout()
90 plt.show()
91
92 # Print the relative energy change for each controller under the sensitivity
       assumption
93 for controller, change in zip(controllers[1:], relative_energy_adjusted[1:]):
94     print(f"Controller␣{controller}:␣{change:.2f}%")
95
96
97
98 # Baseline values
99 baseline_aep = aep[0]
100 baseline_fatigue = fatigue_life[0]
101
```

```
# Percentage change in AEP and Fatigue Life
percent_change_aep = [(val - baseline_aep) / baseline_aep * 100 for val in
    aep]
percent_change_fatigue = [(val - baseline_fatigue) / baseline_fatigue * 100
    for val in fatigue_life]

# Print results
for i, controller in enumerate(controllers):
    print(f"{controller}: ΔAEP = {percent_change_aep[i]:+.2f}%, ΔFatigue Life
        = {percent_change_fatigue[i]:+.2f}%")
```

A.6. LCOE

The following Python script calculates the LCOE for the optimal control cases, done for different discount rates.

Listing A.8: LCOE Calculation Script

```
import numpy as np
import matplotlib.pyplot as plt
# Increase global font sizes
plt.rcParams['axes.labelsize'] = 25    # axis titles
plt.rcParams['xtick.labelsize'] = 20   # x-axis numbers
plt.rcParams['ytick.labelsize'] = 20   # y-axis numbers
plt.rcParams['legend.fontsize'] = 25   # legend text

def calculate_lcoe_corrected(capex, opex, r, n, aep):
    """
    Corrected LCOE calculation:
    - capex: Capital expenditure paid at year 0 (EUR)
    - opex: Annual O&M cost (EUR/year)
    - r: Discount rate (decimal)
    - n: Project lifetime (years)
    - aep: Annual energy production (MWh/year)
    """
    years = np.arange(1, n + 1)
    discounted_opex   = opex * (1 + r) ** (-years)
    discounted_energy = aep  * (1 + r) ** (-years)
    return (capex + discounted_opex.sum()) / discounted_energy.sum()

# Cost assumptions
capex_per_mw = 6500 * 1000    # EUR per MW
opex_per_mw  = 100  * 1000    # EUR per MW-year
rating_mw    = 15             # MW
capex = capex_per_mw * rating_mw
opex  = opex_per_mw  * rating_mw

# Controller cases and data
controllers   = ['Baseline', 'C1.4', 'C5']
aep_values    = {
    'Baseline': 71643.01,
    'C1.4':     70536.66, # MWh/year
    'C5':       71578.61

}
fatigue_life = {
    'Baseline': 25.7,
    'C1.4':     34.5 ,     # years
    'C5':       28.27
```

```
42
43 }
44
45 # Discount rates to compare
46 discount_rates = [0.07, 0.085, 0.1]
47
48 # Compute LCOE for each controller at each discount rate
49 lcoe_results = {ctrl: [] for ctrl in controllers}
50 for r in discount_rates:
51     for ctrl in controllers:
52         n   = fatigue_life[ctrl]
53         aep = aep_values[ctrl]
54         lcoe = calculate_lcoe_corrected(capex, opex, r, n, aep)
55         lcoe_results[ctrl].append(lcoe)
56
57 # Plotting
58 x = np.arange(len(controllers))
59 width = 0.25
60
61 plt.figure(figsize=(10, 6), dpi=300)
62 for i, r in enumerate(discount_rates):
63     values = [lcoe_results[ctrl][i] for ctrl in controllers]
64     plt.bar(x + i*width, values, width, label=f"r = {r:.3f}")
65
66 plt.xticks(x + width, controllers)
67 plt.xlabel("Controller")
68 plt.ylabel("LCOE (EUR/MWh)")
69 plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), ncol=3, frameon=
    False)
70 plt.grid(axis='y', linestyle='--', alpha=0.9)
71 plt.minorticks_on()
72 plt.grid(which='minor', axis='y', linestyle=':', alpha=0.9)
73 plt.tight_layout()
74 plt.show()
75
76
77 # Print all LCOE values
78 for ctrl in controllers:
79     values = lcoe_results[ctrl]
80     print(f"{ctrl}:")
81     for r, val in zip(discount_rates, values):
82         print(f"  r = {r:.3f} → LCOE = {val:.2f} EUR/MWh")
```

The following Python script calculates the LCOE for the optimal control cases, done for reduced OPEX values.

Listing A.9: LCOE Calculation Script for OPEX reduction

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 # Increase global font sizes
4 plt.rcParams['axes.labelsize'] = 25    # axis titles
5 plt.rcParams['xtick.labelsize'] = 20   # x-axis numbers
6 plt.rcParams['ytick.labelsize'] = 20   # y-axis numbers
7 plt.rcParams['legend.fontsize'] = 25   # legend text
8
9
10 def calculate_lcoe_corrected(capex, opex, r, n, aep):
11     """
12     Corrected LCOE calculation:
```

```
    - capex: Capital expenditure paid at year 0 (EUR)
    - opex: Annual O&M cost (EUR/year)
    - r: Discount rate (decimal)
    - n: Project lifetime (years)
    - aep: Annual energy production (MWh/year)
    """
    years = np.arange(1, n + 1)
    discounted_opex   = opex * (1 + r) ** (-years)
    discounted_energy = aep  * (1 + r) ** (-years)
    return (capex + discounted_opex.sum()) / discounted_energy.sum()

# Cost assumptions
capex_per_mw = 6500 * 1000    # EUR per MW
opex_per_mw  = 100  * 1000    # EUR per MW-year
rating_mw    = 15             # MW
capex = capex_per_mw * rating_mw
opex  = opex_per_mw  * rating_mw

# Controller cases and data
controllers   = ['Baseline', 'C1.4', 'C5']
aep_values    = {
    'Baseline': 71643.01,
    'C1.4':     70536.66, # MWh/year
    'C5':       71578.61

}
fatigue_life = {
    'Baseline': 25.7,       # years
    'C5':       25.7,       # now same as baseline
    'C1.4':       25.7
}

# Modify C5 OPEX: 10% drop
opex_values = {
    'Baseline': opex,
    'C5':       opex * 0.9,
    'C1.4':       opex * 0.9
}

# Single discount rate
discount_rate = 0.07  # 7%

# Compute LCOE for each controller
lcoe_results = {}
for ctrl in controllers:
    n        = fatigue_life[ctrl]
    aep      = aep_values[ctrl]
    opex_ctrl = opex_values[ctrl]
    lcoe_results[ctrl] = calculate_lcoe_corrected(capex, opex_ctrl,
        discount_rate, n, aep)

# Print results
for ctrl, lcoe in lcoe_results.items():
    print(f"{ctrl}: LCOE = {lcoe:.2f} EUR/MWh")

# Plotting
plt.figure(figsize=(7, 5), dpi=300)
plt.bar(lcoe_results.keys(), lcoe_results.values(), color=['gray','tab:blue',
    'tab:green'])
```

```
70 plt.xlabel("Controller")
71 plt.ylabel("LCOE␣(EUR/MWh)")
72 # Major grid lines
73 plt.grid(axis='y', linestyle='--', alpha=0.9)
74
75 # Turn on minor ticks and draw minor grid lines
76 plt.minorticks_on()
77 plt.grid(which='minor', axis='y', linestyle=':', alpha=0.5)
78 plt.tight_layout()
79 plt.show()
```