POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

IMPLEMENTATION AND BENCHMARKING OF MODEL BASED AND MODEL FREE CONTROLLERS FOR WIND TURBINES

Supervisors

Prof. Gioacchino CAFIERO

Prof. Gaetano IUSO

Prof. Miguel Alfonso MENDEZ

PhD Lorenzo SCHENA

PhD Emmanuel GILLYNS

Candidate

Sebastiano RANDINO

Academic Year 2022/2023

Abstract

Wind turbines are sophisticated machines designed to harness the raw power of the wind and convert it into electrical energy. Developing a controller for wind turbines is a challenging task due to their complex nature. Their operation is indeed influenced by turbulent external winds and characterized by nonlinear dynamics dictated by aerodynamics. The state of the art in wind turbine control employs classical controllers, but recently the community started to explore AI solutions to improve their performance. This thesis focuses on the development and benchmarking of such control techniques. It articulates on both a numerical and experimental phases.

On the numerical side, a custom-built Python environment is developed to simulate the behavior of different controllers. The model utilizes a simplified, nonlinear first-order representation of wind turbines. The project begins by establishing baseline control techniques, including the widely-used Proportional Integral (PI) controller and the K omega squared technique. These techniques are validated against the existing controllers, specifically the ROSCO package developed by NREL, demonstrating a high level of agreement. Model-based methodologies such as Model Predictive Control (MPC) and data-driven approaches like Genetic Programming (GP), Bayesian Optimization (BO), and Reinforcement Learning (RL) are explored. Comparisons are conducted to evaluate their performance.

The experimental campaign is conducted in the Low Speed Wind Tunnel L-2B at the von Karman Institute. The wind tunnel features a test section area of 0.35 cm x 0.35 cm, with a maximum incoming flow velocity of 35 m/s. Efforts are dedicated to implementing the reading of rotational speed from an encoder with an Arduino master-slave architecture, which interacts with a Python code responsible for wind turbine control. Experimental testing involves the PI controller and the optimization of the PI controller's weights using Bayesian Optimization (BO). Lastly, a specific experimental setup is designed to simulate real-world conditions in wind turbine farms, positioning one wind turbine downstream of another. This configuration enables the validation of the controller's performance in mitigating wake interference and optimizing overall wind farm performance.

In conclusion, based on the promising outcomes achieved with data-driven controllers in the numerical simulations, further experimental testing of these controllers would be of great interest for future research.

Acknowledgements

I would like to thank everyone who supported me throughout my thesis project at the von Karman Institute, starting with my supervisor, Prof. Miguel Alfonso Mendez, and my two advisors, PhD Lorenzo Schena and PhD Emmanuel Gillyns, for their invaluable assistance. A heartfelt thank you to Lorenzo for his insights in the numerical part and to Emmanuel for his guidance during the experimental phase. Furthermore, I would like to extend my thanks to PhD Nicolas Coudou for his support and for granting me permission to use the wind turbine model he developed.

Lastly, I would like to express my gratitude to my supervisors at Politecnico di Torino, Prof. Gioacchino Cafiero and Prof. Gaetano Iuso, who not only instilled in me a deep passion for experimental aerodynamics over the years but also made the VKI experience possible.

Table of Contents

List of Tables

List of Figures

Acronyms

WECS
Wind Energy Conversion System

BEM
Blade Element Momentum (theory)

PID
Proportional Integrative Derivative (controller)

MPC
Model Predictive Control

BO
Bayesian Optimization

GP
Genetic Programming

RL
Reinforcement Learning

ANN
Artificial Neural Network

DRL
Deep Reinforcement Learning

DDPG
Deep Deterministic Policy Gradient

Chapter 1

Introduction

1.1 General Introduction to Wind Turbines

A wind turbine is a Wind Energy Conversion System (WECS). It refers to the infrastructure and technology used to harness wind energy and convert it into electrical energy.

Over time, many designs for a wind turbine have been proposed. The two main ones are classified depending on the rotor position:

- **vertical-axis** wind turbines;
- **horizontal-axis** wind turbines;

There are pro and cons for each type of wind turbine. The most important feature of a vertical-axis turbine is that the transmission devices and generator are located at ground level. Accessibility is simplified, eliminating the need for climbing the turbine and reducing associated costs. Additionally mounting the generator separately reduces structural loads on the turbine and mitigates noise and vibration issues. Another advantage is that a vertical turbine can capture the wind energy from any direction. However, these advantages are cancelled by the fact that the rotor capture a low percentage of the wind speed. Furthermore, their higher mechanical complexity and maintenance requirements can increase operational costs. As a result, the utilization of vertical-axis wind turbines has significantly decreased in recent decades [1].

Nowadays, horizontal-axis two-bladed or three-bladed rotors are predominantly employed in commercial wind turbines connected to the grid. The main components of an horizontal-axis wind turbine include the rotor, nacelle and tower. The rotor is located at the top of the tower and its purpose is to convert as much as possible of the kinetic energy of the wind into mechanical energy, which is then used to spin the low-speed shaft inside the nacelle. A gearbox is used to increase the rotational

speed of the low-speed shaft, which in turn drives a second shaft to power the electrical generator. There is also a yaw mechanism that turns the rotor and nacelle to face the wind [2].

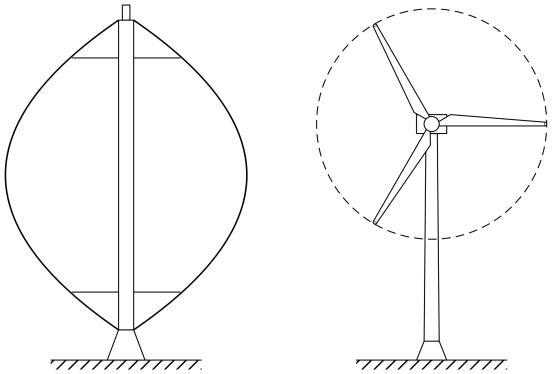

Figure 1.1: Examples of vertical and horizontal axis wind turbine, reference [1]

Discussing about horizontal-axis wind turbines, it is important to note that various configurations can be employed:

- **constant rotor speed**: the rotor maintains a constant speed. This configuration is capable of extracting the maximum power from the wind, but only at a specific wind speed. One advantage of this setup is its simplicity in extracting power from the wind. However, these turbines have become less prevalent due to their limitations in adapting to varying wind conditions;
- **variable rotor speed**: this is the configuration that is widely used in modern wind turbines. It offers higher aerodynamic efficiency for a range of wind speeds, but it requires electrical power processing to ensure the generated electricity is fed into the grid at the correct frequency;
- **fixed pitch angle**: the orientation of the blades along the longitudinal axes remains constant. This design simplifies the mechanical structure and control system of the turbine;
- **variable pitch angle**: it is possible to rotate all the blades or a portion of them. This is achieved through the use of hydraulic or electric actuators. The advantage of load reduction offered by variable pitch turbines has led to their dominance in modern utility-scale turbine markets.

In this thesis horizontal wind turbines with variable rotor speed and variable pitch angle are treated for the advantages depicted above.

1.2 Control and Safety Systems

The requirement for control has been present since the inception of wind turbines. Initially, the primary objectives of control were to restrict power and speed to predefined limits, ensuring safe operation of the turbine during high wind conditions. Early wind turbines relied on rudimentary mechanical devices to achieve these goals. However, as wind turbines grew in size and power output, the demands for control became more stringent, necessitating the development of advanced regulatory mechanisms.
There are different levels of controls in a wind turbine [3]:

- **supervisory control**: involves transitioning between operational states, including stand-by, start-up, power production, shutdown, and stopped with fault. It ensures smooth transitions and monitors various conditions to optimize turbine performance. For example, in low wind speeds, the turbine is often stopped to avoid inefficiencies and insufficient energy production. Similarly, during high wind speeds, the turbine can reach its load limit, requiring a halt to prevent damage. The supervisory control acts as a controller, stopping the turbine when load limits are exceeded or system faults occur. Its primary objective is to maintain safe and efficient turbine operation.
- **closed loop control**: is an essential software-based system that dynamically adjusts the operational state of the wind turbine to maintain it within a predefined operating curve or characteristic. It performs various functions such as controlling the blade pitch or adjusting generator torque to maximize power extraction or implementing power limitation strategies. These objectives will be further discussed in the following section.
- **safety system**: is separate from the main control system and is responsible for bringing the turbine to a safe condition in case of serious problems. It acts as a backup to the main control system and takes over if the main system fails to ensure safety. It includes various fail-safe actions such as disconnecting electrical systems and applying the shaft brake. The safety system can be triggered by events like rotor overspeed, operator-pressed emergency stop button, or other faults indicating potential control issues.

1.2.1 Closed Loop Control

In this thesis, the focus will be solely on closed loop control, as the supervisory control is not a subject of interest for control design. Operational control can be divided into different regions, each with its own control objective. Conversely, the question is to find a well balanced compromise among them. These partial goals can be summed up in the following [1]:

- **energy capture**: maximize the capture of energy while considering important operational constraints such as rated power, rated speed, and cut-out wind speed;
- **mechanical loads**: this objective encompasses various aspects, including alleviating transient loads, mitigating high-frequency loads, and avoiding resonance;
- **power quality**: condition the generated power in order to comply with interconnection standards.

In this thesis, only the energy capture aspect will be considered, while the other two aspects will be neglected.

There are different regions in operational control that correspond to specific control strategies for energy capture. Some of the common regions include [4]:

- **Region 0**: it is the region at low wind speed where the power available in the wind is low compared to losses in the turbine system. This velocity is called cut-in wind speed v_{min}. The turbine in this region is turned off.
- **Region 1**: is the region for mid-range wind speeds when the wind speed is above the cut-in wind speed $v > v_{min}$. In this region, the wind turbine operates at its maximum aerodynamic efficiency and extracts the maximum available power from the wind, regulating the rotational speed with the generator torque. The pitch angle in this region is kept constant at its optimal value.
- **Region 1.5**: is when the wind speed is larger than the wind speed that corresponds with rated rotor speed, but less than the wind speed that corresponds with rated power. Both torque and pitch control are applied in this transitional region between Region 1 and 2.
- **Region 2**: when the wind speed is above the rated wind speed $v > v_{rated}$, the power output of the wind turbine is limited to its rated value to prevent exceeding electrical and mechanical load limits. In this region, the generator torque is maintained at a constant value, and the pitch angle of the blades is actively controlled. The purpose of controlling the pitch angle is to adjust the rotational speed of the turbine and keep it at the rated value, ensuring that the turbine consistently produces the rated power output.
- **Region 3**: this region occurs when the wind speed exceeds the cut-out wind speed of the turbine, denoted as v_{max}. In Region 3, the blades of the turbine are pitched such that the rotor thrust is reduced to zero. This means that the wind turbine is effectively shut down in this regime because it is not designed to operate under such extreme wind conditions.

It is usual to represent the power extraction as a curve in relation to the wind speed. The figure above shows the power envelope of a specific wind turbine, the NREL 5MW (1.2.1), which will be utilized for designing and testing numerically the controllers in this thesis.

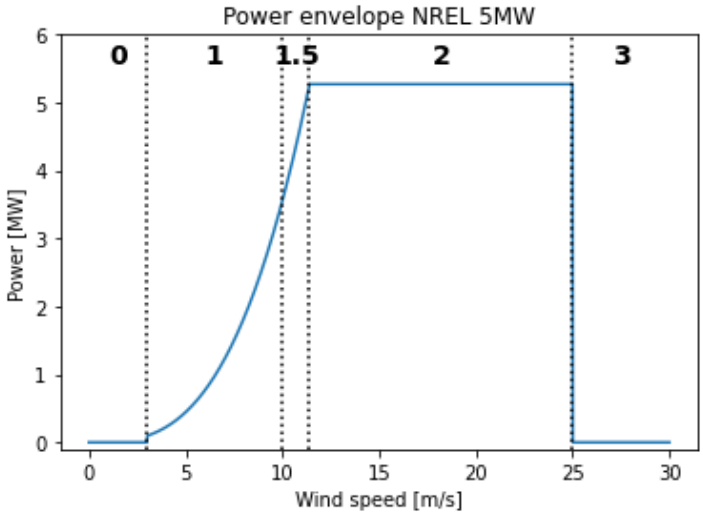

Figure 1.2: NREL 5MW Power Envelope

In Figure 1.2, the different operational regions of a wind turbine can be observed. It is important to note that this plot represents the steady-state ideal conditions in which the wind turbine is expected to operate. In reality, due to the unsteady nature of the wind load (turbulent wind), the turbine will experience operation in non-steady conditions throughout its lifespan. This implies that the turbine may not always follow the power curve depicted in the plot. The goal of the closed-loop controller is to try to operate the wind turbine as closely as possible to this curve.

NREL 5MW Wind Turbine

The NREL 5MW wind turbine is a widely used reference model for designing wind turbine control systems. It is a modern utility-scale turbine with a variable speed and variable pitch control system, and it has been extensively tested and validated in various research projects. The turbine's advanced control algorithms enable it to operate at high efficiency across a wide range of wind speeds, while also maintaining safe and reliable operation. As a result, the NREL 5MW turbine serves as a benchmark for the development of new wind turbine control strategies and technologies.

In the table below, it can be seen all the data related to this turbine [5]:

Parameter	Value
Rated power	5 MW
Cut-in wind speed	3 m/s
Rated wind speed	11.4 m/s
Cut-out wind speed	25 m/s
Rotor diameter	126 m
Hub height	90 m
Blade chord length	4.19 m
Blade twist angle	-13.308° to 4.275°
Rated rotor speed	12.1 RPM
Gearbox ratio	97:1
Generator efficiency	96%

Table 1.1: Data of NREL 5MW wind turbine

1.2.2 Sensors and Actuators

The wind turbine control system comprises various sensors and actuators, along with a hardware and software system. This system receives input signals from the sensors, processes them, and produces output signals to control the actuators.
The actuators in a wind turbine are:

- **yaw motor**: aligns the nacelle with the main direction of the wind;
- **generator**: it can be commanded to follow a desired torque or load. It determines how much torque is extracted from the turbine. The net torque on the rotor depends on the input torque from the wind and the torque from the generator. The generator torque it can be used to change the acceleration and deceleration of the rotor;
- **blade-pitch motor**: there is a motor for each blade, so the blades can be controlled collectively or independently. Changing the pitch angle, the aerodynamic torque due to the wind vary and the rotor speed can be controlled;

The most commonly used sensors include:

- **rotor speed sensor**: measures the rotational speed of the wind turbine rotor;
- **anemometer**: measures wind speed. It helps determine the intensity of the wind, which is essential for optimizing the turbine's operation;

- **pitch position sensor**: is responsible for monitoring the angle of the turbine blades. It enables precise control of blade pitch adjustment;
- **electrical power sensor**: measures the electrical power output of the wind turbine. It provides real-time data on the energy generated, allowing for efficient power management and performance monitoring.

1.3 The Wind

Wind turbines are typically positioned in the lower region of the atmosphere known as the shear layer. This layer, referred to as the planetary shear layer, directly engages with the Earth's surface. Due to the non-slip condition, the velocity at ground level is zero, leading to a significant velocity gradient. Given the large scale of wind turbine systems, the flow is characterized by a high Reynolds number and exhibits turbulent behavior. Generally, the height of the shear layer is approximately 2 km. One fascinating aspect of surface winds is how their kinetic energy is distributed across different frequencies, known as the van der Hoven spectrum. If the van der Hoven spectrum is analyzed in this region, it can provide valuable insights into the characteristics of wind turbulence.

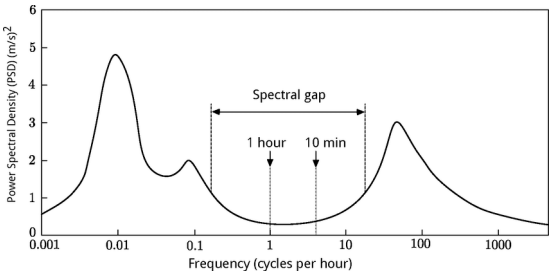

Figure 1.3: Typical van der Hoven spectrum, reference [6]

For instance, in Figure 1.3, the spectrum exhibits two energy peaks at two different frequencies, approximately 0.01 cycles/hour and 50 cycles/hour. This allows us to decompose the wind into two components:

$$v = v_m + v_t \tag{1.1}$$

one corresponding to the first frequency, which represents the mean part of the wind v_m, and the other representing the turbulent part v_t.
Considering both components of the wind, the power carried by the wind with density ρ passing through a section A is given by:

$$P_v = \frac{1}{2}\rho A v^3 \tag{1.2}$$

which represents the source from which the wind turbine extracts power.

1.4 Wind Turbine Aerodynamics

A parameter to evaluate the performance of a wind turbine in its ability to extract power from the wind is the coefficient of power:

$$C_P = \frac{P}{P_v} = \frac{P}{\frac{1}{2}\rho A v^3} \tag{1.3}$$

which is defined as the ratio of the power produced by the wind turbine P to the power of the wind P_v, which is the maximum available power. The power coefficient is determined by the aerodynamics of the wind turbine and can be calculated using various methods.

1.4.1 Actuator Disc Model

One of the simplest approaches is to use an actuator disk model. In this model, the rotor is represented as a permeable disc that slows down the incoming wind flow, leading to a localized pressure drop at the disk.
If one denote the velocity at the rotor as v_D, typically the velocity at the actuator disc is expressed as:

$$v_D = (1 - a)v \tag{1.4}$$

where a is defined as the induction factor.
From the conservation of flow rate in the flow tube passing through the turbine rotor section A, and by applying Bernoulli's equation to derive the pressure drop on the rotor, one can obtain:

$$F_D = 2\rho A v^2 a(1 - a) \tag{1.5}$$

that is the force that is produced by the rotor F_D. Then, the power harvested from the airflow by the actuator disc can be determined using the following equation:

$$P = F_D v_D = 2\rho A v^3 a(1 - a) \tag{1.6}$$

Therefore, in this case, it can be found that:

$$C_P = 4a(1 - a)^2 \tag{1.7}$$

From here, it can be observed that the maximum possible power coefficient, also known as the Betz limit, is reached at a value of $a = 1/3$, and it is equal to $C_{P_{max}} = 0.593$. This represents the upper limit for any turbine, indicating that it is impossible to extract all the available wind energy.
The actuator disc model has several limitations. It simplifies the representation of flow, neglects the influence of the shape of the blade aerodynamic profiles, assumes

a uniform velocity distribution, and lacks accuracy at high angles of attack while considering the flow as frictionless. These limitations can lead to inaccuracies when predicting power output, loads, and stall behavior of wind turbines. However, this model is important to realize the upper limit to the maximum energy that can be extracted from the wind.

1.4.2 Blade Element Momentum theory

A second possible approach to calculate the power coefficient is the Blade Element Momentum (BEM) method. The BEM method considers the variation of aerodynamic forces across the rotor's span, resulting in improved accuracy for predicting turbine performance. It also takes into account the influence of blade geometry and airfoil characteristics, enabling a more comprehensive analysis of the flow field. The flow tube encompassing the swept area of the turbine is divided into infinitesimally small concentric annular stream tubes. Each stream tube is treated independently.

By considering the relative velocity impacting each profile of the blade, it is possible to calculate the lift and drag forces per unit length for each blade element, knowing the aerodynamic coefficients of the profile:

$$f_L = \frac{\rho c}{2} v_{rel} C_L(\alpha) \tag{1.8}$$

$$f_D = \frac{\rho c}{2} v_{rel} C_D(\alpha) \tag{1.9}$$

where c is the chord length, v_{rel} is the composition between the upstream wind speed and the tangential blade element speed and α is the incidence angle. This angle is the combination of the angle between the local flow direction ϕ and the pitch angle β:

$$\alpha(r) = \phi(r) - \beta(r) \tag{1.10}$$

It is dependent on the radial position r of the considered blade element. From the the lift and drag forces the rotational torque per unit length can be computed:

$$\tau_r = \frac{\rho c}{2} v_{rel}^2 r[C_L(\phi(r) - \beta(r))sin(\phi(r)) - C_D(\phi(r) - \beta(r))cos(\phi(r))] \tag{1.11}$$

By integrating (1.11) along the blades length, torque produced by the entire rotor can be obtained. By multiplying it by the rotational velocity, the power generated P can be obtained. Typically, the power generated is expressed in adimensional form using the coefficient of pressure:

$$C_P(\lambda, \beta) = \frac{P}{P_v} \tag{1.12}$$

From the previous equation, it can be observed that the power coefficient is dependent from the tip speed ratio λ and the pitch angle β of the blades:

$$\lambda = \frac{\omega_r R}{v} \tag{1.13}$$

where ω_r is the rotational speed of the rotor and R is the turbine's rotor radius.

1.5 Framework and Goals of this Thesis

This project originated under the guidance of the research group led by my supervisor, Prof. Mendez. His research group is applying data-driven controllers in various fluid dynamics contexts [7].
In particular, one of my two advisors, PhD Lorenzo Schena, is working on data-driven approaches for wind turbine controllers [8]. In collaboration with my other PhD advisor, Emmanuel Gillyns, they have implemented an experimental setup to experimentally test these solutions [9].

Therefore, my thesis is based on this framework, aiming to develop controllers for wind turbines in Region 2. This is done in a numerical environment and subsequently apply the knowledge obtained from the numerical simulations in an experimental setup. The numerical objectives involve creating a simulation environment that adequately represents the dynamics of a wind turbine. Once this environment is established, the focus shifts to the development of baseline controllers that serve as references for benchmarking more advanced model-based and model-free controllers. The aim is to compare the performance of each different controller and evaluate the advantages and disadvantages of using model-free methods for wind turbine control. Finally, based on the obtained numerical results, a proof of concept is sought by conducting experimental tests on a small-scale wind turbine model to validate the findings from the numerical simulations.

1.6 Thesis Outline

The thesis is organized as follows: Section 2 provides the methodology used for the numerical implementation of the controllers. It includes a review of the theoretical foundations underlying the development of controllers.

Section 3 offers a comprehensive overview of the experimental setup and delineates the methodology employed in the experimental phase. Additionally, this section presents the objectives of the various test cases.

In Section 4, results are presented. The focus in the numerical part is on comparing the performance of the different controllers, including highlighting

the learning curves of the data-driven controllers. The obtained experimental results are also presented, although direct comparisons between the numerical and experimental parts are not possible due to the architecture of the experimental controller.

Section 5 contains the conclusions drawn from the results presented in the thesis.

Lastly, Section 6 outlines potential future work and research directions.

Chapter 2

Numerical Methods and Objective

The process of developing a wind turbine controller, as outlined in [10], can be summarized in the following steps: determining control objectives, developing a simplified dynamic model, applying control theory, and performing dynamic simulations to evaluate closed-loop system performance.

In this chapter, a comprehensive explanation of each step in the wind turbine controller development process is provided. Firstly, the control objectives and employed model are presented. Finally, the implementation of baseline controllers, model-based controllers, and model-free controllers is discussed. The results of the simulation for each controller can be found in Chapter 4.

2.1 Control Objectives

The main objective is to develop controllers focusing on Region 2 of operation. As discussed in (1.2.1), the control objective in Region 2 is to maintain the power produced by the wind turbine at its rated value:

$$P = P_{rated}$$

This is achieved by controlling the rotational speed to remain at its nominal value through the pitch angle of the blades, while maintaining the generator torque at its nominal level so that the produced power, which is the product of these two quantities, remains constant. In this sense, it can be said that the set-point to be maintained and therefore the primary objective of the controller is to keep the rotor rotational speed at the nominal level.

Due to the turbulent nature of the wind, it is possible that for certain periods of time, even though the controller mostly operates in Region 2, it may enter Region

1 or 1.5. For this reason, in this thesis is developed a baseline controller for Region 1, in case the wind conditions lead to operation in that region. The objectives of a controller in Region 1 can be summarized as follows. Contrary to Region 2, the goal in Region 1 is to maximize power output. In this case, the set-point is to track the optimal rotational speed using torque based on the wind speed.

2.2 Employed Models

The second step of the controller's development, it is to have a model that represents the dynamics of the system. To capture the dynamic behavior of a wind turbine, it is essential to model the entire environment in which it operates. In this section, the dynamic model that will be utilized to simulate the behavior of the wind turbine is presented. Subsequently, the corresponding wind model employed in the simulations is discussed.

2.2.1 First Order Model

The level of complexity for the model used in control design is determined by the desired objectives. Various approaches utilize different techniques to model turbine dynamics. One such approach is the multi-body approach, which divides the system into multiple rigid bodies that interact with each other through constraints.

For this preliminary investigation, a simplified first order model has been utilized, encompassing the essential aspects of wind turbine dynamics. This model is widely employed by control engineers in wind applications [2] [4], rendering it suitable for an initial comparative analysis between model-free and conventional controllers. The assumptions of the model are:

- the system is treated as a rigid body. Deformations of the blades, tower and other parts of the system are not allowed. This means that the aeroelastic coupling between the structure and the external aerodynamics is not considered;
- The power coefficient follows the map proposed by the Blade Element Momentum (BEM) theory;
- the inertia momentum considered in the model is only that of the rotor, while the others are considered negligible;
- the dynamics of the system are solely dependent on the wind speed at the center of the rotor, and therefore, a uniform distribution of wind on the blades is assumed;
- the response of torque to a control input is considered instantaneous.

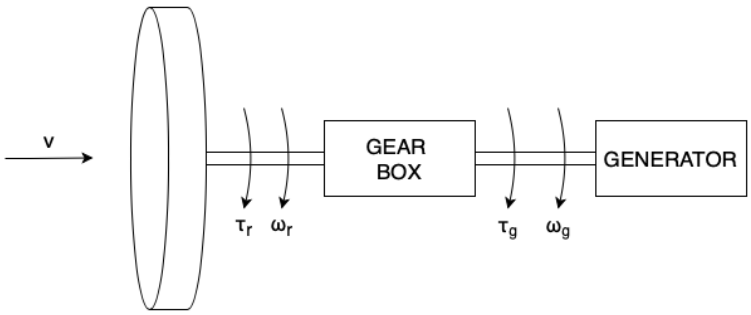

Figure 2.1: First order model of the wind turbine

The model used in this study is illustrated in Figure 2.1. By applying the torque equilibrium equation in the shaft of the turbine rotor, the following expression is obtained:

$$J\dot{\omega_r} = \tau_{aero} - \tau_{load} \tag{2.1}$$

where J is the inertia momentum of the rotor, ω_r is the rotational speed of the rotor, τ_{aero} is the aerodynamic torque of the rotor and τ_{load} is the reaction torque in the rotor shaft that is applied by the gearbox. To calculate the reaction torque on the rotor shaft, one must take into account the efficiency of the gear box η. This efficiency can be used to relate the torque of the generator to the load torque on the rotor shaft, as shown in the following equation:

$$\eta = \frac{\omega_g \tau_g}{\omega_r \tau_{load}} = \frac{\omega_r N_g \tau_g}{\omega_r \tau_{load}} = \frac{N_g \tau_g}{\tau_{load}} \tag{2.2}$$

$$\tau_{load} = \frac{N_g}{\eta} \tau_g \tag{2.3}$$

where N_g is the gear box ratio, τ_g is the net torque of the generator and ω_g is the angular velocity of the generator. This relationship is important for understanding the power transfer between the rotor and generator.
The aerodynamic torque of the rotor is given by:

$$P_r = \frac{1}{2}\rho A v^3 C_P(\lambda, \beta) \tag{2.4}$$

$$P_r = \omega_r \cdot \tau_{aero} \tag{2.5}$$

$$\tau_{aero} = \frac{P_r}{\omega_r} = \frac{1}{2}\rho A v^3 \frac{C_P(\lambda, \beta)}{\omega_r} \tag{2.6}$$

At the end we came up with the following equation:

$$\dot{\omega_r} = \frac{1}{J}\left(\frac{1}{2}\rho A \frac{C_P(\lambda,\beta)}{\omega_r} v^3 - \frac{N_g}{\eta}\tau_g\right) \tag{2.7}$$

The given equation is a first-order nonlinear ordinary differential equation with the wind speed v that represents the forcing term. It can be seen from this equation that indeed the system can be controlled through the torque of the generator τ_g and the pitch angle β, which are the two terms that can affect the dynamic of the turbine.

The choice of this first-order model is very limiting in real-world applications, especially for large-scale wind turbines. However, it is useful for gaining a simplified understanding of how to develop controllers. Even with a more advanced model, the steps to follow essentially remain the same, albeit with increased difficulty in their development.

In order to solve (2.7), the power coefficient $C_P(\lambda,\beta)$ for the NREL 5 MW wind turbine was computed using the Blade Element Momentum formulation, as illustrated in (1.4.2), using the WISDEM CCBlade Python package [11]. The power coefficient map obtained for the NREL 5MW turbine is a 2D contour plot that depends on pitch angle and tip speed ratio, as can be seen in Figure 2.2.

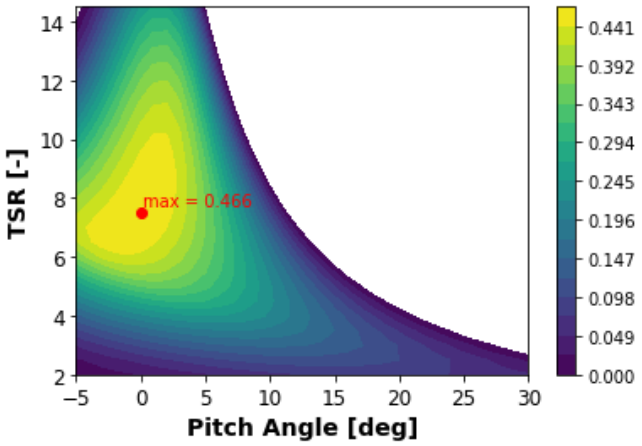

Figure 2.2: NREL 5MW C_P 2D map

The relationship between the power coefficient C_P, the tip speed ratio λ and the pitch angle β is a non-linear function that is specific to each turbine. Also it can be observed that the maximum power coefficient is indeed smaller than the Betz limit:

$$C_{P_{max}} = 0.446$$

and it is obtained at the optimal pitch angle:

$$\beta_{opt} = 0°$$

and at the optimal tip speed ratio:

$$\lambda_{opt} = 7.5$$

Pitch Subsystem

As said in (1.2.2) actuators are necessary to change the pitch angle of the blades. In modern applications, hydraulic or electromechanical actuators are utilized to efficiently control the speed and power limitations. The pitch actuator, which is a nonlinear servo, typically rotates all the blades, or a portion of them, simultaneously.

To enhance the realism in the simulator, in this project the dynamic behavior of the pitch actuator is incorporated. The controller generates a desired pitch angle β_d, but due to the actuator's inherent dynamics, it cannot instantaneously reach this angle. As a result, there exists a delay between the desired angle and the actual angle achieved, denoted as β. Despite its non-linearity, the pitch actuator can be modeled as a first-order dynamic system with amplitude and derivative saturation.

The dynamic behaviour of the pitch actuator operating in its linear region is described by the differential equation:

$$\dot{\beta} = -\frac{1}{\tau}\beta + \frac{1}{\tau}\beta_d \tag{2.8}$$

where β_d is the desired angle, β is the actual angle and τ is the typical time constant and it depends on the specific actuator. The scheme of the dynamic of the pitch actuator could be sum-up in Figure 2.3.

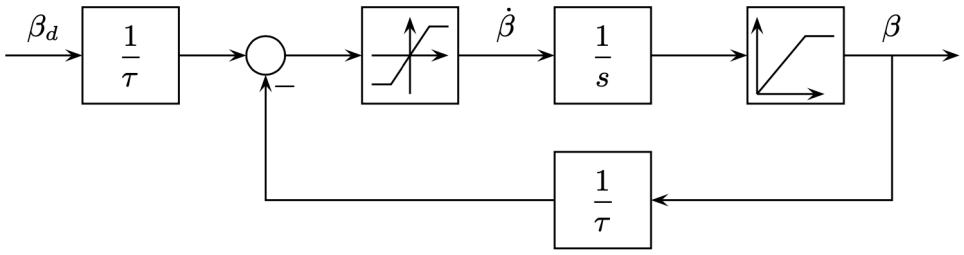

Figure 2.3: Pitch actuator model, reference [1]

The equations (2.7) and (2.8) represent the equations that need to be simultaneously solved to obtain the numerical solution of the turbine dynamics. To solve each of these two ODEs, a fourth-order Runge-Kutta method is implemented in a Python script. Further details about the numerical scheme used in this simulator can be found in Appendix A.

2.2.2 Synthetic generated Wind Speed

To solve Equation (2.7), the input of wind speed is required. The wind was simulated using the TurbSim Python package [12], which provides a comprehensive framework for generating realistic wind profiles. By using TurbSim, the wind characteristics, such as wind speed, turbulence intensity, and wind direction, can be accurately modeled to reflect real-world conditions.

The wind is generated through an input file, from which various parameters can be specified. The first section of the input file contains specifications for the turbine/model, including time step and simulation duration. The second section of the input file contains specifications for the Meteorological Boundary Conditions. These include the spectral model used, turbulence intensity settings, the mean wind profile, the reference wind speed height and mean stream-wise wind speed.

The two most important parameters that influence the temporal behavior of the wind speed are the spectral models and the wind profile. The spectral models define the spectral model used to generate wind velocity fluctuations. Examples of spectral models include the Kaimal model, the von Karman model, and the smooth terrain model. These models describe the distribution of wind velocity fluctuations as a function of frequency. The wind profile specifies the mean wind profile of the shear layer. There are different types of wind profiles, such as the logarithmic profile, the power law profile, and the low-level jet profile. These profiles depict the variation of wind velocity with height above the ground and thus represent the behavior of the planetary boundary layer.

In Table 2.1, some of the parameters used to generate an example of synthetic wind are summarized.

Parameter	**Value**
Time Step	0.05 s
Analysis Duration	300 s
Hub Height	90 m
Spectral Model Used	von Karman spectrum
Turbulence Intensity Settings	10%
Mean Wind Profile	Power law
Mean Streamwise Wind Speed	16 m/s

Table 2.1: Parameters of TurbSim input file

The wind profile taken as an example follows the power law, which is expressed in the following form:

$$v(z) = v(z_{ref}) \left(\frac{z}{z_{ref}} \right)^{\alpha} \tag{2.9}$$

where z is the height above ground, $v(z)$ is the wind speed at height z, z_{ref} is a reference height above ground where the wind speed $V(z_{ref})$ is known and α is the power law exponent.

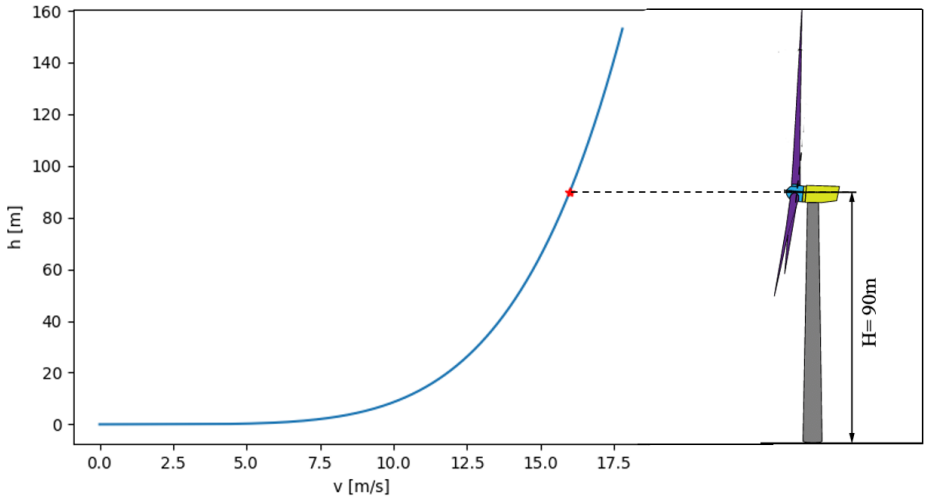

Figure 2.4: Example of a TurbSim wind profile generated with the power law

In Figure 2.4, the velocity profile obtained, which represents the mean of the turbulent velocity field, is depicted. The time series used for simulation is extracted from the point on the profile that aligns with the hub height of the wind turbine, highlighted in red in Figure 2.4. Figure 2.5 illustrates the time series extracted from this specific point. As depicted in this figure, the generated wind exhibits turbulence, with notable velocity variations. This presents an ideal testing environment to evaluate the performance of different controllers in a realistic scenario.

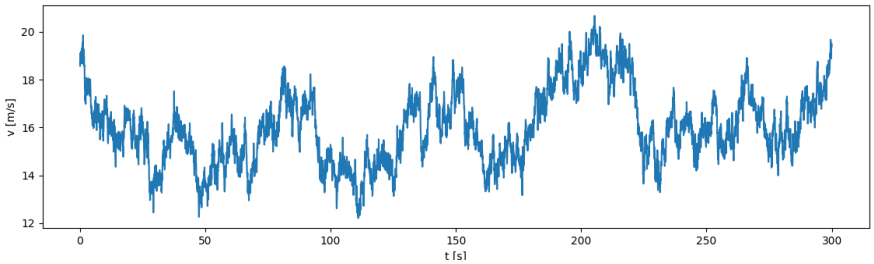

Figure 2.5: Wind time series extracted from the wind profile

Wind Speed Filter

According to the literature, a low-pass filter was employed to filter the wind speed. This filtering approach helps in improving the performance and stability of the control system by reducing the impact of rapid wind speed variations on the turbine's response.

In this thesis, a first-order digital low-pass Butterworth filter was used. The cut-off frequency of the filter was set to one-fourth of the blade resonance frequency, as specified in the NREL 5 MW technical document [5]:

$$f_{c.o.} = \frac{f_r}{4} = 0.25Hz \tag{2.10}$$

The cutoff frequency in the filter corresponds to the frequency at which the signal is attenuated by -3dB, as observed from the Bode plot of the filter in Figure 2.6.

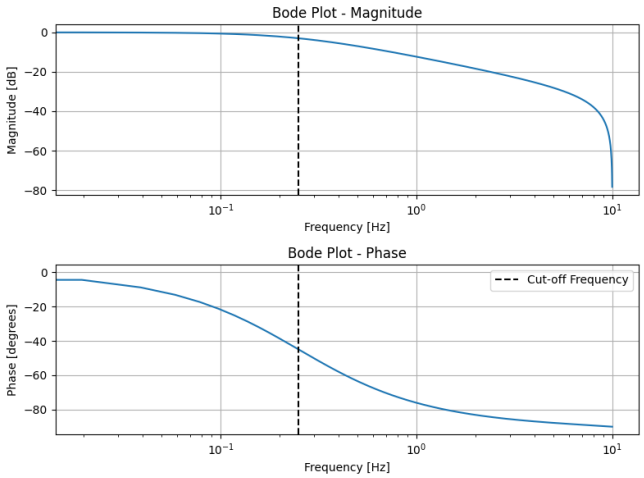

Figure 2.6: Bode diagram of the employed first-order Butterworth low-pass filter

In Figure 2.7, the filtered wind speed of the generated wind signal in Section (2.2.2) is illustrated.

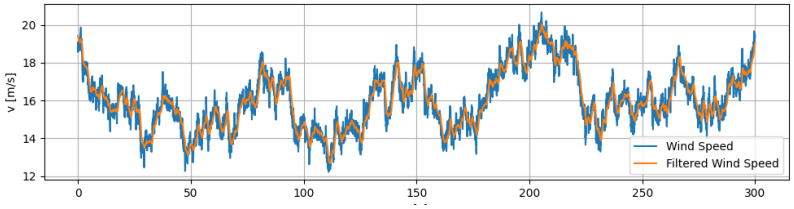

Figure 2.7: Example of wind speed after being filtered

The choice of a first-order filter was made because, although its attenuation is not the strongest possible, it introduces less phase deviation in the filtered signal compared to higher-order filters.

2.2.3 Controller Scheme

Each controller developed in this thesis interacts with the plant/environment as depicted in Figure 2.8. The controller operates in a closed-loop configuration. It can take both the rotational speed ω and filtered wind speed v_{filt} states as inputs. Depending on the control region, the controller's output will either be the pitch angle β or the torque τ.

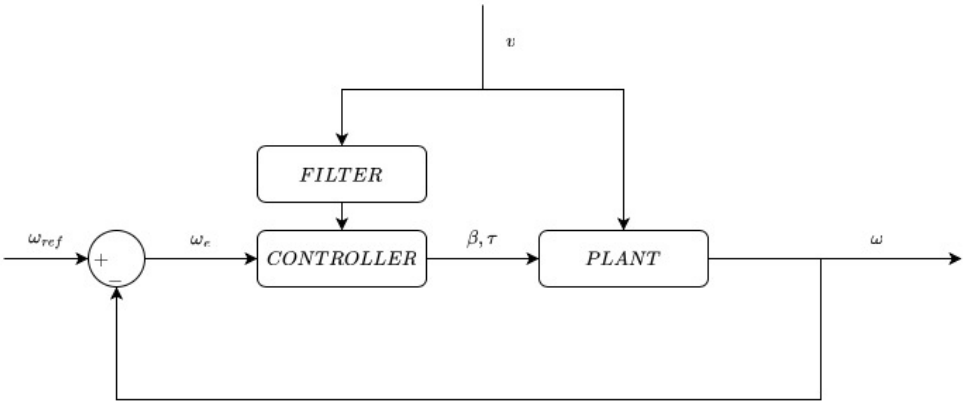

Figure 2.8: General scheme of the closed-loop controller

2.3 Baseline Controllers

The initial phase of this thesis was dedicated to developing controllers based on the mathematical model of the wind turbine. These controllers represent the current state of the art in wind turbine control and are widely utilized as a reference for benchmarking purposes.
Before describing the implementation of the baseline controllers, it is necessary to discuss the linearization of the model, which is essential in linear control theory.

2.3.1 Model Linearization

The development of a baseline controller relies on linear theories, necessitating the linearization of the nonlinear turbine model. Hence, it is crucial to choose an appropriate operating point for linearization. The model described by Equation (2.7) can be linearized around a specific operating point 0 using the Taylor series:

$$\begin{aligned} \dot{w} &= f(w, v, \beta, \tau_g) = \\ &= f(w_0, v_0, \beta_0, \tau_{g0}) + \left(\frac{\partial f}{\partial w}\right)_0 (w - w_0) + \left(\frac{\partial f}{\partial v}\right)_0 (v - v_0) + \\ &+ \left(\frac{\partial f}{\partial \beta}\right)_0 (\beta - \beta_0) + \left(\frac{\partial f}{\partial \tau_g}\right)_0 (\tau_g - \tau_{g0}) + ... \end{aligned} \tag{2.11}$$

If the series is truncated at the first order, the nonlinear equation can be approximated as follows:

$$\dot{w} = A_0 + A(w - w_0) + B_v(v - v_0) + B_\beta(\beta - \beta_0) + B_\tau(\tau_g - \tau_{g0}) \tag{2.12}$$

where the coefficients A_0,A,B_v,B_β and B_τ are described in the following.
The first coefficient is just the value of the left side of (2.7) evaluated on the operating point 0:

$$A_0 = f(w_0, v_0, \beta_0, \tau_{g0}) \tag{2.13}$$

If the point is a stady-state point this term is zero.
The second term is the partial derivative with respect to the rotational speed of the rotor:

$$A = \left(\frac{\partial f}{\partial w}\right)_0 = \frac{1}{J}\left(\frac{\partial \tau_a}{\partial \lambda}\right)_0 \left(\frac{\partial \lambda}{\partial w}\right)_0 \tag{2.14}$$

$$\left(\frac{\partial \lambda}{\partial w}\right)_0 = \frac{R}{v_0}$$

$$\left(\frac{\partial \tau_a}{\partial \lambda}\right)_0 = \frac{1}{2}\rho A R \frac{v_0^2}{\lambda_0^2}\left[\left(\frac{\partial C_P}{\partial \lambda}\right)_0 \lambda_0 - C_{P_0}\right]$$

Due to the lack of an analytical solution for the partial derivative of C_P with respect to λ, this derivative has been approximated using a central finite difference scheme centered around the point 0, utilizing the data from the discrete C_P map:

$$\left(\frac{\partial C_P}{\partial \lambda}\right)_0 = \frac{C_P(\lambda_0 + h) - C_P(\lambda_0 - h)}{2h}$$

The third term represents the partial derivative with respect to the wind speed:

$$B_v = \left(\frac{\partial f}{\partial v}\right)_0 = \frac{1}{J}\left(\frac{\partial \tau_a}{\partial v}\right)_0 \tag{2.15}$$

$$\left(\frac{\partial \tau_a}{\partial v}\right)_0 = \frac{1}{2}\rho A \frac{1}{\omega_0}\left[\left(\frac{\partial C_P}{\partial v}\right)_0 v_0^3 + 3C_{P_0} v_0^2\right]$$

$$\left(\frac{\partial C_P}{\partial v}\right)_0 = \left(\frac{\partial C_P}{\partial \lambda}\right)_0 \left(\frac{\partial \lambda}{\partial v}\right)_0$$

$$\left(\frac{\partial \lambda}{\partial v}\right)_0 = -\frac{\omega_0 R}{v_0^2}$$

The fourth term is the term sensitive to the pitch angle:

$$B_\beta = \left(\frac{\partial f}{\partial \beta}\right)_0 = \frac{1}{J}\frac{1}{2}\rho A \frac{v_0^3}{\omega_0}\left(\frac{\partial C_P}{\partial \beta}\right)_0 \tag{2.16}$$

Similarly to the previous case, the partial derivative of C_P with respect to β is calculated using a central finite difference scheme:

$$\left(\frac{\partial C_P}{\partial \beta}\right)_0 = \frac{C_P(\beta_0 + h) - C_P(\beta_0 - h)}{2h}$$

Last term is the derivative of the torque, that is already a linear term:

$$B_\tau = -\frac{N_g}{J\eta_g} \tag{2.17}$$

Thanks to linearization, the system's behavior around an operating point can be described by a first-order linear differential equation. However, this model's accuracy diminishes as one moves away from the linearization point.

2.3.2 PI Pitch Controller

The state-of-the-art approach for pitch control in Region 2 involves the utilization of a Proportional-Integrative-Derivative (PID) control law:

$$\beta(t) = k_P \omega_e(t) + k_I \int_0^t \omega_e(\tau) d\tau + k_D \frac{d\omega_e(t)}{dt} \tag{2.18}$$

where $\omega_e = \omega - \omega_{rated}$ is the error between the actual rotor speed and the desired rotor speed (in this case the rated rotor speed) and the parameters k_P, k_I, and k_D represent the proportional, integral, and derivative gains of the PID controller, respectively.
In wind turbine control, the derivative term in a PID controller is often not used

because it can amplify measurement noise and introduce instability [13]. At the end the controller law is just a PI:

$$\beta(t) = k_P\omega_e(t) + k_I \int_0^t \omega_e(\tau)d\tau \tag{2.19}$$

According to linear control theory, the gains k_P and k_I are obtained from the linearization of the system around a specific operating point (2.3.1). Typically, the linearization point is selected as the steady-state value within the operating region of the turbine.
In this case, the linearization point chosen is:

- $\omega_0 = \omega_{rated}$;
- $v_0 = 17m/s$;
- $\tau_0 = \tau_{rated}$;
- $\beta_0 = \beta_{ss}$;

The steady-state point is chosen to satisfy the desired objectives in Region 2. Multiple steady-state points can be arbitrarily selected within Region 2 based on the expected wind speed value. In this case, linearization was performed around the mean operational wind speed within Region 2 of 17 m/s. To obtain the pitch angle value that ensures this steady-state operation β_{ss}, the non-linear equation was solved:

$$\frac{1}{2}\rho A\frac{C_P(\beta_{ss})}{\omega_{rated}}v_0^3 - \tau_{rated} = 0 \tag{2.20}$$

The linearized equation expressed in terms of perturbed variables with respect to the steady-state value is as follows:

$$\Delta\dot{\omega} = A\Delta\omega + B_\beta\Delta\beta + B_v\Delta v \tag{2.21}$$

where the torque term is not present as the torque is kept constant in Region 2 and where:

- $\Delta\omega = w - \omega_{rated}$
- $\Delta\beta = \beta - \beta_{ss}$
- $\Delta v = v - v_{lin}$

These perturbations are assumed to represent small deviations of these variables away from their equilibrium values at steady state.
By taking the Laplace transform of the controller equation using perturbed values:

$$\Delta\beta = k_P\Delta\omega + k_I \int_0^T \Delta\omega dt \tag{2.22}$$

$$\Delta\beta(s) = k_P\Delta\omega(s) + k_I\frac{\Delta\omega(s)}{s} \tag{2.23}$$

the controller transfer function can be derived:

$$C(s) = \frac{\Delta\beta(s)}{\Delta\omega(s)} = k_P + \frac{k_I}{s} \tag{2.24}$$

At the same way, if the Laplace transform is applied to (2.21):

$$\Delta\dot{\omega} = A\Delta\omega + B_\beta\Delta\beta + B_v\Delta v \tag{2.25}$$

$$s\Delta\omega(s) = A\Delta\omega(s) + B_\beta\Delta\beta(s) + B_v\Delta v(s) \tag{2.26}$$

and putting (2.24) into (2.26):

$$(s - A)\Delta\omega = B_\beta\left(k_P + \frac{k_I}{s}\right)\Delta\omega + B_v\Delta v \tag{2.27}$$

one can derive the transfer function that describes the closed-loop system:

$$H(s) = \frac{\Delta\omega(s)}{\Delta v(s)} = \frac{B_v s}{s^2 - (B_\beta k_P + A)s - B_\beta k_I} \tag{2.28}$$

It can be noticed that this system has the same poles of a simple second-order system:

$$s^2 - (B_\beta k_P + A)s - B_\beta k_I = 0 \tag{2.29}$$

$$s^2 + 2\omega_n\zeta s + \omega_n^2 = 0 \tag{2.30}$$

By defining the desired natural frequency ω_n and damping ratio ζ of the rotor speed response, one can obtain the weights of the controller:

$$k_P = -\frac{1}{B_\beta}(2\zeta\omega_n + A) \tag{2.31}$$

$$k_I = -\frac{\omega_n^2}{B_\beta} \tag{2.32}$$

In linear control theory, it is known that changing these two parameters can have various effects on the system's behavior:

- ζ is the damping ratio and it is a dimensionless quantity that determines the decay of oscillations in the system's natural response. If ζ is less than 1, the system exhibits oscillatory behavior. Conversely, if ζ is greater than 1, the system displays non-oscillatory behavior.

- ω_n is the natural frequency of the system and it regulates the frequency response of the system.

In order to investigate the effects of different damping ratios ζ on the system, some simulations are conducted by applying a step function to the wind input. This enables the observation of the controller's response for various ζ values.

The simulation is done under the following conditions:

- the simulation takes $t = 200s$;
- a step wind from $v = 17m/s$ to $18m/s$ is applied to the system after $t = 100s$;
- $\zeta = [0.5,1,2]$, while ω_n is keep constant;

The response of the rotor speed can be observed under these conditions, providing insights into its behavior and dynamics.

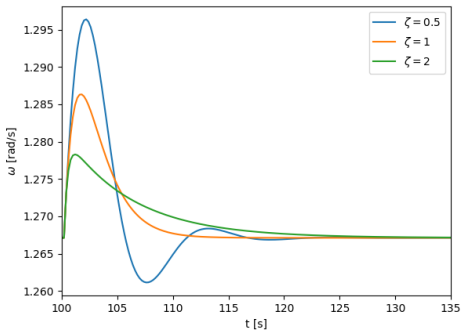

Figure 2.9: Rotor speed response under a step change in wind speed from $v = 17$m/s to 18m/s with different values of ζ (fixed ω_n)

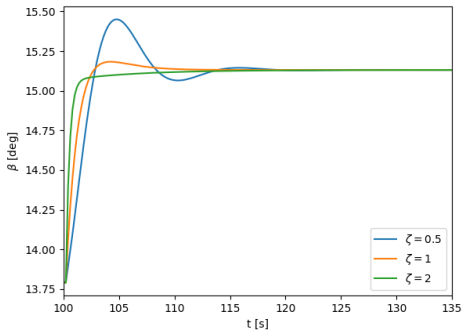

Figure 2.10: Pitch angle response under a step change in wind speed from $v = 17$m/s to 18m/s with different values of ζ (fixed ω_n)

In Figures 2.9 and 2.10, the results of these simulations are presented. Under the condition of $\zeta = 0.5$, as expected, the rotor speed exhibits a damped oscillation. When $\zeta = 1$, the response becomes critically damped, characterized by the fastest possible decay time. Increasing ζ to 2 leads to a longer decay time as the solution. It is generally undesirable for the pitch angle to exhibit oscillatory behavior. The optimal value for ζ in this application is indeed 1 because it provides the faster time response, which is well-suited for applications such as wind turbines where external wind conditions change rapidly.

The effect of the parameter ω_n was studied by conducting an identical simulation while varying the value of the coefficient ω_n and keeping the value of ζ fixed at 1.

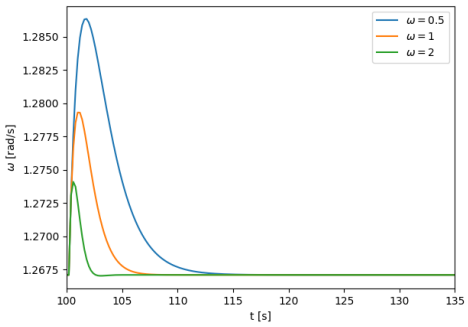

Figure 2.11: Rotor speed response under a step change in wind speed from $v = 17$m/s to 18m/s with different values of ω_n (fixed ζ)

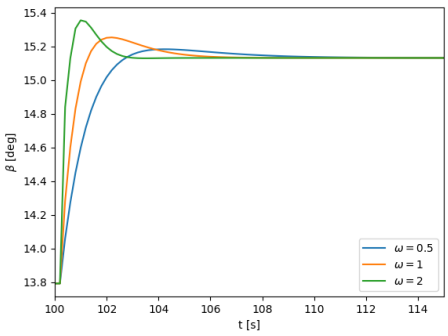

Figure 2.12: Pitch angle response under a step change in wind speed from $v = 17$m/s to 18m/s with different values of ω_n (fixed ζ)

From Figure 2.11 and 2.12, it is observed that the pitch angle exhibits an overshoot in all cases. Among the different values of ω_n, the smoothest overshoot is observed when $\omega_n = 0.5$. On the other hand, higher values of ω_n result in a shorter decay time but with a higher overshoot. The desired compromise is to have a smooth overshoot in order to reduce fatigue on the turbine blades. A commonly suggested value in the literature [4] is $\omega_n = 0.6$.

Based on the conducted test cases, the two optimal parameters suitable for controlling the wind turbine are as follows:

$$\omega_n = 0.6$$

$$\zeta = 1$$

Finally, from the coefficients obtained through the linearization process:

- $A = -0.238 \ \ 1/\text{s}$
- $B_v = 0.029 \ \ \text{rad}/(\text{ms})$
- $B_\beta = -1.229 \ \ 1/\text{s}^2$

the PI gains can be determined:

$$k_p = 0.782s$$

$$k_i = 0.292$$

Closed loop Bode Diagram

To analyze the controller's performance under the oscillatory behavior of the turbulent wind, the Bode diagram of (2.28) is plotted.

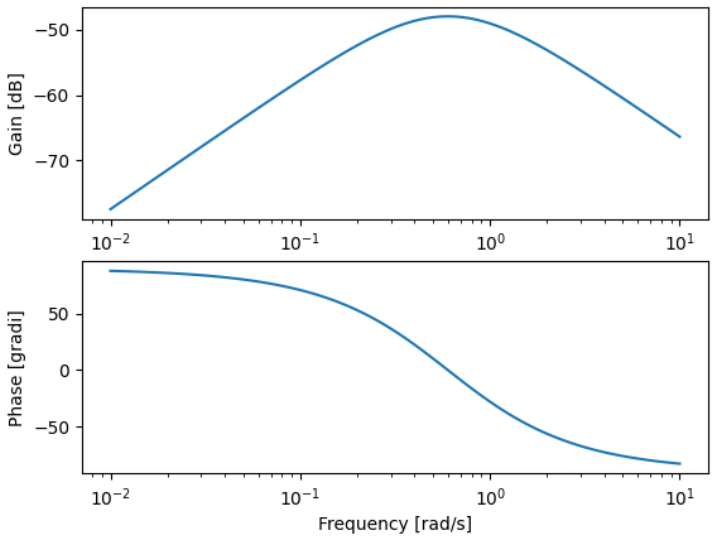

Figure 2.13: Bode diagram of the closed-loop transfer function of the system

From Figure 2.13, one can observe that there exists a specific frequency at which the controller exhibits the lowest attenuation and no phase shift in the output signal.

The expected operating region is the one between 0.1-0.5 rad/s, which represents the typical turbulence frequencies. This is precisely the region where the controller experiences minimal phase shift but also less attenuation.

2.3.3 PI Scheduling

The problem with using linear control theory for a nonlinear system is that the design performance is only guaranteed near the design operating point. In applications such as wind turbines, where the external wind force undergoes rapid variations, the system experiences frequent changes in its operating point, deviating from the design point. As a consequence, the system's performance may deviate from the expected behavior.

Figure 2.14 depicts the response of the turbine's rotational speed when subjected to two different step of the wind: a step from 17 m/s to 18 m/s (in blue) and a step from 12 m/s to 13 m/s (in orange). When the operating point is altered, it becomes

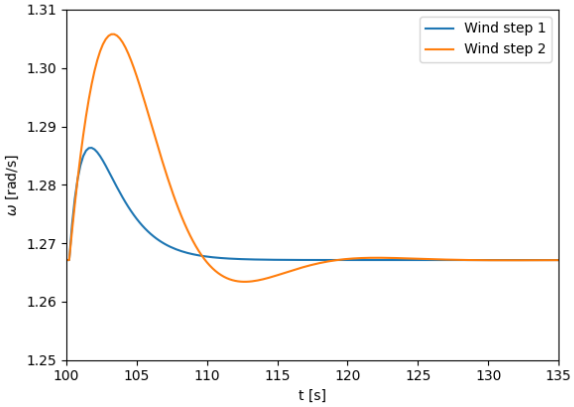

Figure 2.14: Rotor speed response without scheduling under two different steps in wind speed: from $v = 17$m/s to 18m/s in blue and from $v = 12$m/s to 13m/s in orange

apparent that the system's response, using the previously tuned gains, deviates from the desired response demonstrated earlier. Specifically, when subjected to a wind step from $12m/s$ to $13m/s$, the system exhibits oscillatory behavior, as indicated by the orange curve. The approach to address this issue involves selecting different weights by different choosing linearization points around various operating points.

The procedure remains the same as in the previous section, but this time there is no specific wind speed chosen for linearization. Instead, the coefficients of Equation (2.21) are dependent on the wind speed:

$$\Delta\dot{\omega} = A(v)\Delta\omega + B_\beta(v)\Delta\beta + B_v(v)\Delta v \tag{2.33}$$

From Equation (2.31) and Equation (2.32), it is evident that both gains, k_P and k_I, are dependent on the wind speed (Figure 2.15). In this case, the controller requires the external wind speed as an additional input to adjust the gain values.

One of the drawbacks of gain scheduling is that it limits the frequency response analysis to small intervals around specific wind speeds. Since the gains change with different wind speeds, the linear analysis techniques typically employed for linear systems become limited in their applicability. This leads to the concept of a nonlinear proportional-integral (PI) controller, where the controller's behavior becomes dependent on the wind speed and non-linearity is introduced into the controller.

From Figure 2.16, it can be observed that by implementing scheduling, the controller's performance can be aligned with the design specifications, resulting in beneficial effects.

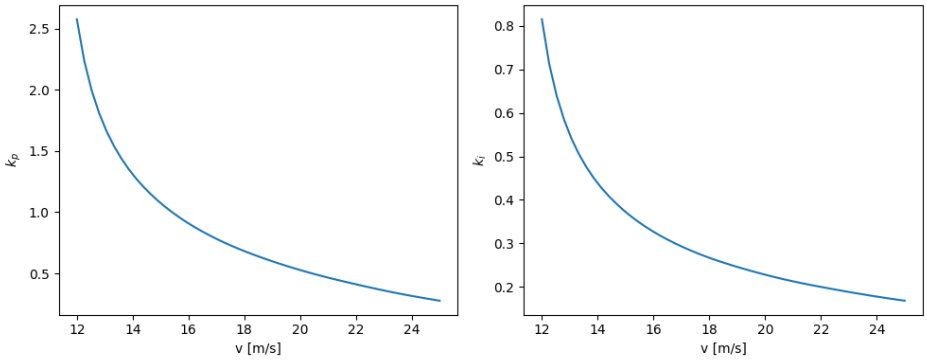

Figure 2.15: Proportional and integral gains as a function of wind speed

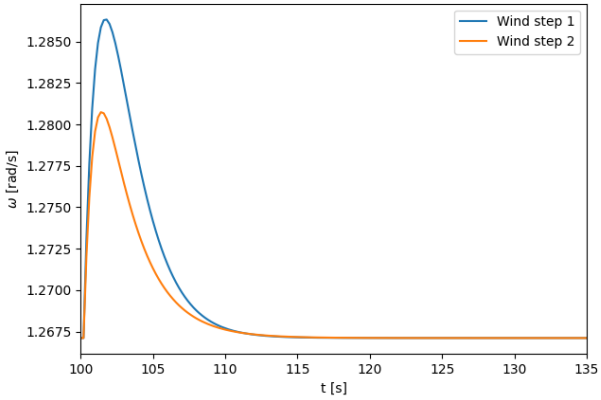

Figure 2.16: Rotor speed response with scheduling under two different steps in wind speed: from $v = 17$m/s to 18m/s in blue and from $v = 12$m/s to 13m/s in orange

2.3.4 $K\omega^2$ Torque Controller

The need to develop a baseline controller, as mentioned in Section (2.1), arises from the requirement to handle situations where, for brief periods or certain wind speeds, the controller may enter Region 1-1.5. In this thesis, the $k\omega^2$ torque controller, which is widely used in the state-of-the-art, was chosen as the baseline controller. However, the Proportional-Integral (PI) controller, employed for pitch control, can also be adapted for the torque control. Since the maximum power coefficient $C_{P_{max}}$ is achieved at the optimal Tip Speed Ratio λ^*, the reference velocity w_{ref} should be adjusted as follows:

$$w_{ref} = \lambda^* \frac{v}{R} \tag{2.34}$$

The $K\omega^2$ controller for wind turbines implements a torque control strategy to regulate the generator torque, aiming to achieve maximum power output while maintaining safe operating conditions. The aim of the controller is to control the generator torque, in order to reach the $C_{P_{max}}$ at steady state conditions.

One can set the torque at the steady state value as:

$$\dot{\omega} = \frac{1}{2}\rho A R^3 \frac{C_{P_{max}}}{\lambda_*^3}\omega^2 - \frac{N_g}{\eta}\tau_g = 0 \tag{2.35}$$

and the torque generator can be determined by:

$$\tau_g = \frac{1}{2}\rho A R^3 \frac{C_{P_{max}}}{\lambda_*^3}\frac{\eta}{N_g}\omega^2 = K\omega^2 \tag{2.36}$$

where:

$$K = \frac{1}{2}\rho A R^3 \frac{C_{P_{max}}}{\lambda_*^3}\frac{\eta}{N_g} \tag{2.37}$$

It can be observed that the coefficient K depends on both the turbine characteristics and the environmental conditions, as well as the level of turbine degradation. Moreover, this controller does not require wind velocity as a control feedback, but the only required feedback is the rotation speed of the turbine.

In this case, the controller was also tested using a wind step of 1 m/s intensity from $5m/s$ to $6m/s$. As observed in Figure 2.17, the rotor rotation speed adjusts itself to reach the optimal TSR. Additionally, it can be seen that the coefficient of power tends to approach its maximum value under steady-state conditions.

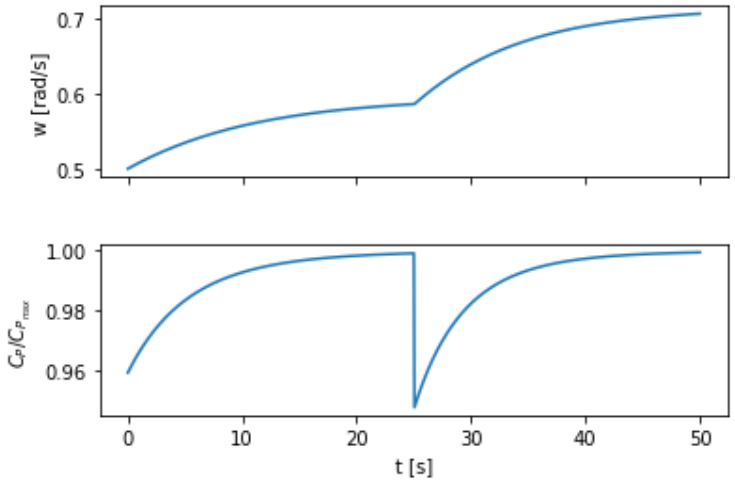

Figure 2.17: Response of rotational speed and dimensionless power coefficient with his maximum value to a step change in wind speed from 5 m/s to 6 m/s

2.4 Advanced Model Based Controller

In the application of classical controllers to wind turbines, there are several limitations associated with these approaches. As mentioned earlier, these methods rely on a linear model that is accurate only near the linearization point. Hence, there is a need to find a controller that can optimize performance across the entire operating range. One such controller that can meet these requirements is Model Predictive Control (MPC).
MPC is a control approach that is still based on the wind turbine model, but it offers several advantages [14]:

- it allows for optimal decision-making for dynamic systems, considering both the current state and predicted future behavior;
- it offers the capacity to handle constraints, such as pitch rate and rotor speed, by incorporating them into the optimization problem formulation;
- it allows for handling Multi-Input-Multi-Output (MIMO) control problems.

While this thesis focuses on the development of Single-Input-Single-Output (SISO) controllers, the first two advantages mentioned above are promising for the development of such controllers.

2.4.1 Model Predictive Control (MPC)

MPC enables optimal decision-making for dynamic systems subject to operational constraints. It relies on predicting future system behaviors and iteratively optimizing control signals to achieve desired objectives while ensuring compliance with specific constraints.
In this thesis, a linear and discrete Model Predictive Control (MPC) approach is developed as a proof of concept. Although a non-linear continuous MPC would have been a more suitable choice in theory, its implementation complexity led to the preference for the linear MPC, which shares fundamental similarities in key aspects of the control strategy.
MPC is associated with the following definitions:

- **sampling time** Δt: is the time interval between consecutive measurements of system states (for continuous-time systems $\Delta t = 0$);
- **time horizon** N_T: is the total number of time steps of the entire simulation;
- **prediction horizon** N: is the number of time steps of the prediction window over which future states are estimated and optimized;

- **cost function** J: is a mathematical formulation that quantifies the performance objectives of the control problem. By minimizing this function, the optimal control inputs can be found.

Referring to (2.21) the transformation into the discrete-time linear time-invariant (LTI) system is:

$$\Delta\omega^{k+1} = A^d\Delta\omega^k + B_\beta^d\Delta\beta^k + B_v^d\Delta v^k \tag{2.38}$$

It can be noticed that the matrices A, B_τ, and B_v of the continuous system are different from the matrices of the discrete system, denoted with the superscript ^{d}. When applying the explicit Euler method to discretize the linear ordinary differential equation, it is observed that:

$$A^d = 1 + \Delta tA$$

$$B_\beta^d = \Delta tB_\beta$$

$$B_v^d = \Delta tB_v$$

The constrained sets for the state and control input can be defined as follows:

$$\mathbb{X} = \{\Delta\omega \in \mathbb{R} : \mathbf{F_x}\Delta\omega \leq \mathbf{g_x}\} \tag{2.39}$$

$$\mathbb{U} = \{\Delta\beta \in \mathbb{R} : \mathbf{F_u}\Delta\beta \leq \mathbf{g_u}\} \tag{2.40}$$

In this case, the essential constraints regarding the rotational speed can be defined as follows:

$$0 \leq \omega \leq 2 \cdot \omega_{rated} \tag{2.41}$$

This set could be written in terms of the perturbed values:

$$-\omega_{rated} \leq \Delta\omega \leq \omega_{rated} \tag{2.42}$$

One can write these constraints in a compact way:

$$\mathbf{F_x}\Delta\omega \leq \mathbf{g_x} \tag{2.43}$$

where the matrices are:

$$\mathbf{F_x} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \mathbf{g_x} = \begin{bmatrix} \omega_{rated} \\ \omega_{rated} \end{bmatrix} \tag{2.44}$$

Constraints for the pitch controller can be defined as follows:

- pitch angle limits:

$$\beta_{min} \leq \beta \leq \beta_{max} \tag{2.45}$$

- pitch rate limits:

$$\dot{\beta}_{min} \leq \dot{\beta} \leq \dot{\beta}_{max} \tag{2.46}$$

If the pitch angle ratio constraints are discretized using the explicit Euler method:

$$\dot{\beta} = \frac{\beta - \beta^{k-1}}{\Delta t} \tag{2.47}$$

they can be rewritten as follows:

$$\beta^{k-1} + \dot{\beta}_{min}\Delta t \leq \beta \leq \beta^{k-1} + \dot{\beta}_{max}\Delta t \tag{2.48}$$

Considering both the pitch angle and the pitch angle ratio saturation constraints that need to be respected, the overall constraints can be summarized as follows:

$$max(\beta_{min}, \beta^{k-1} + \dot{\beta}_{min}\Delta t) \leq \beta \leq min(\beta_{max}, \beta^{k-1} + \dot{\beta}_{max}\Delta t) \tag{2.49}$$

Pitch angle constraints can be expressed concisely as follows:

$$\mathbf{F_u}\Delta\beta \leq \mathbf{g_u} \tag{2.50}$$

where the matrices are:

$$\mathbf{F_u} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \mathbf{g_u} = \begin{bmatrix} min(\beta_{max}, \beta^{k-1} + \dot{\beta}_{max}\Delta t) - \beta_{ss} \\ -max(\beta_{min}, \beta^{k-1} + \dot{\beta}_{min}\Delta t) + \beta_{ss} \end{bmatrix} \tag{2.51}$$

The cost function with a prediction horizon N at time instant k is defined as follow:

$$J_k = \Delta\omega_{k+N|k}^T Q_N \Delta\omega_{k+N|k} + \sum_{i=k}^{k+N-1} \Delta\omega_{i|k}^T Q \Delta\omega_{i|k} + \Delta\beta_{i|k}^T R \Delta\beta_{i|k} \tag{2.52}$$

where the predicted state $\Delta\omega_{i|k}$ and control input $\Delta\beta_{i|k}$ at time instant i are computed during the prediction step at time instant k. The coefficients R and Q are weight coefficients that can be arbitrarily chosen and represent the importance in optimizing the states or the control law. For instance, if the matrix R is larger than Q, it implies that the control law is more important than the desired state. Optimizing the control law might be necessary to minimize the energy expended in controlling the system. On the contrary, as chosen in this thesis, having a greater Q matrix than R means giving more importance to the state being controlled. The coefficient Q_N is called the terminal cost coefficient and is necessary to achieve stability. The stability of the optimal control problem can be demonstrated if the terminal cost Q_N is appropriately chosen [15].
By assuming an LQR feedback law $\beta_k = K\omega_k$, the terminal cost coefficient can be determined by imposing that the optimal cost should decrease at the next step:

$$J_k^* \geq J_{k+1}^*$$

where the superscript * represents the optimal solution. As shown in [15], the terminal set can be found by solving the discrete-time Riccati equation:

$$Q + K^\top RK - Q_N + (A^d + B^d_\beta K)^\top Q_N (A^d + B^d_\beta K) = 0 \tag{2.53}$$

The MPC approach, given the current state $\Delta\omega_{k|k}$, aims to optimize the cost function J_k over a certain time window composed of $N < N_T$ timesteps:

$$J_k = \Delta\omega^T_{k+N|k} Q_N \Delta\omega_{k+N|k} + \sum_{i=k}^{k+N-1} \Delta\omega^T_{i|k} Q \Delta\omega_{i|k} + \Delta\beta^T_{i|k} R \Delta\beta_{i|k}$$

that is subjected to the following constraints:

$$\Delta\omega^{k+1} = A^d \Delta\omega^k + B^d_\beta \Delta\beta^k + B^d_v \Delta v^k$$

$$\mathbf{F_x}\Delta\omega \leq \mathbf{g_x}$$

$$\mathbf{F_u}\Delta\beta \leq \mathbf{g_u}$$

The result of the MPC is a control sequence over the time horizon, from which only the first value of the sequence $\beta_{k|k}$ will be taken and applied to the system. The same optimization problem is repeated for each time step in the MPC approach. In this thesis, (2.38) is simplified as follows:

$$\Delta\omega^{k+1} = A^d \Delta\omega^k + B^d_\beta \Delta\beta^k \tag{2.54}$$

The cancellation of the external wind disturbance forcing term was carried out due to the absence of the capability to predict future wind conditions in the considered architecture. This action resulted in assuming a constant and fixed wind speed at time instant k across the entire prediction horizon. It is important to note that new wind turbine architectures equipped with LIDAR technology enable wind speed prediction within a time window. However, for the purpose of conducting a comparison with other controllers' architectures, the choice was made to refrain from using this predictive capability and, instead, assume a constant wind speed for the entire prediction horizon.
As shown in [16], the previously described optimization problem can be reformulated to make it more manageable and easier to implement. From (2.54) it can be obtained:

$$\begin{bmatrix} \omega_{k|k} \\ \omega_{k+1|k} \\ \cdot \\ \cdot \\ \cdot \\ \omega_{k+N|k} \end{bmatrix} = \begin{bmatrix} I \\ A^d \\ \cdot \\ \cdot \\ \cdot \\ A^{dN} \end{bmatrix} \omega_k + \begin{bmatrix} 0 & 0 & \cdot & \cdot & \cdot & 0 \\ B_\beta & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ A^{dN-1}B_\beta & A^{dN-2}B_\beta & \cdot & \cdot & \cdot & B_\beta \end{bmatrix} \begin{bmatrix} \beta_{k|k} \\ \beta_{k+1|k} \\ \cdot \\ \cdot \\ \cdot \\ \beta_{k+N-1|k} \end{bmatrix} \tag{2.55}$$

In matrix form, it becomes:

$$\mathbf{\Omega_k} = \mathbf{A_X}\omega_k + \mathbf{B_X}\mathbf{U_k} \tag{2.56}$$

Similarly, by defining:

$$\mathbf{Q_X} = \begin{bmatrix} Q & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & Q & 0 \\ 0 & \cdot & 0 & Q_N \end{bmatrix}; \mathbf{R_U} = \begin{bmatrix} R & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & R & 0 \\ 0 & \cdot & 0 & R \end{bmatrix}; \tag{2.57}$$

the cost function (2.52) can be represented:

$$J_k = \mathbf{\Omega_k}^T \mathbf{Q_X} \mathbf{\Omega_k} + \mathbf{U_k}^T \mathbf{R_U} \mathbf{U_k} \tag{2.58}$$

By expressing:

$$\mathbf{F_X} = \begin{bmatrix} \mathbf{F_x} & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \mathbf{F_x} & 0 \\ 0 & \cdot & 0 & \mathbf{F_x} \end{bmatrix}; \mathbf{g_X} = \begin{bmatrix} \mathbf{g_x} \\ \cdot \\ \mathbf{g_x} \\ \mathbf{g_x} \end{bmatrix}; \mathbf{F_U} = \begin{bmatrix} \mathbf{F_u} & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \mathbf{F_u} & 0 \\ 0 & \cdot & 0 & \mathbf{F_u} \end{bmatrix}; \mathbf{g_U} = \begin{bmatrix} \mathbf{g_u} \\ \cdot \\ \mathbf{g_u} \\ \mathbf{g_u} \end{bmatrix}; \tag{2.59}$$

the constraints can be represented in:

$$\mathbf{F_X}\mathbf{\Omega_k} \leq \mathbf{g_X} \tag{2.60}$$

$$\mathbf{F_U}\mathbf{U_k} \leq \mathbf{g_U} \tag{2.61}$$

Finally, by setting the following matrices:

$$\mathbf{z} = \begin{bmatrix} \mathbf{\Omega_k} \\ \mathbf{U_k} \end{bmatrix}; \mathbf{H} = \begin{bmatrix} \mathbf{Q_X} & \mathbf{0} \\ \mathbf{0} & \mathbf{R_U} \end{bmatrix}; \tag{2.62}$$

the cost function can be represented with a single vector:

$$J_k = \mathbf{z}^T \mathbf{H} \mathbf{z} \tag{2.63}$$

The constraints can also be combined into a single vector:

$$\mathbf{F}\mathbf{z} \leq \mathbf{g} \tag{2.64}$$

by defining the matrices:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F_X} & \mathbf{0} \\ \mathbf{0} & \mathbf{F_U} \end{bmatrix}; \mathbf{g} = \begin{bmatrix} \mathbf{g_X} \\ \mathbf{g_U} \end{bmatrix}; \tag{2.65}$$

The equation (2.54) can be compacted in:

$$\mathbf{F_{eq}z} = \mathbf{g_{eq}} \tag{2.66}$$

by combining the matrices:

$$\mathbf{F_{eq}} = \begin{bmatrix} \mathbf{I} & -\mathbf{B_U} \end{bmatrix}; \mathbf{g_{eq}} = \mathbf{A_X}\omega_k; \tag{2.67}$$

The original optimization problem can be represented as a quadratic programming (QP) problem as below:

$$\begin{aligned} \text{minimize} \quad & \mathbf{z}^T\mathbf{Hz} \\ \text{subject to} \quad & \mathbf{Fz} \leq \mathbf{g} \\ & \mathbf{F_{eq}z} = \mathbf{g_{eq}} \end{aligned}$$

In this form, this QP problem can be solved using the Python package cvxopt [17]. In MPC this optimization problem is solved during each time instant k and the first element of U_k^* is applied to the system:

$$\beta_k = \beta_{k|k}$$

in whitch the superscript * represents the optimal solution.

2.5 Model Free Controllers

In the context of model-free controllers, a control problem can be seen as an optimization problem where the objective is to find the optimal control function while satisfying the system dynamics. The goal is to minimize or maximize a function that quantifies the controller's performance. Control methods can be categorized as white, grey, or black, depending on the level of system knowledge utilized in the optimization process. A white approach (as the controllers described before) relies heavily on the analytical description of the system, while a black-box or model-free approach relies solely on input-output data and interacts directly with the system. These models have the flexibility to describe nonlinear functions without the need for prior knowledge or assumptions, making them suitable for problems that are challenging to address analytically or replicate accurately in numerical simulations. Machine learning methods, such as Genetic Programming and Reinforcement Learning, and global optimization techniques, such as Bayesian Optimization, have gained prominence in the field of control.

By harnessing the potential of machine learning, the complex control challenges encountered in wind turbine systems can be address. As discussed in the previous

sections, these challenges often involve nonlinear dynamics and may lack a precise mathematical representation. Machine learning approaches, with their model-free nature, offer significant advantages such as adaptability and robustness, enabling the controllers to adapt to changing environmental conditions and uncertainties. These characteristics make machine learning particularly suitable for wind turbines, which are subjected to varying wind speeds and turbulence, leading to dynamic complexities.

In the subsequent sections, some of the most important model-free control approaches and their applications are presented.

2.5.1 Bayesian Optimization (BO)

The Bayesian Optimization (BO) is arguably the most popular "surrogate-based", derivative-free, global optimization tool. In its simplest mathematical form, the problem can be formulated as follows:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \ f(\mathbf{x}) \tag{2.68}$$

The key components of BO include:

- **cost function**: in this case denoted as $f(\mathbf{x})$, represents the objective that needs to be minimized. It maps input parameters $\mathbf{x}$ to a scalar value that measures the performance or quality of a given configuration.
- **acquisition function**: determines the utility or desirability of evaluating a specific point in the parameter space. It combines information from the current state of knowledge about the cost function and its uncertainty. The acquisition function guides the search by balancing exploration (sampling points with high uncertainty) and exploitation (sampling points with high potential for improvement).

BO leverages these components to iteratively search for the optimal solution, aiming to minimize the number of evaluations required while efficiently exploring the parameter space. In its most classic form the BO uses a Gaussian process for regression of the cost function under evaluation and the acquisition function to decide where to sample next. BO is particularly useful when the objective function is expensive to evaluate, as it intelligently selects new points to evaluate based on the surrogate model's predictions [18].

For these reasons, BO is a powerful approach for optimizing black-box functions. When applied to black-box optimization problems, BO aims to find the global optimum of an objective function without making any assumptions about its analytical form or structure.

The BO process can be summarised as:

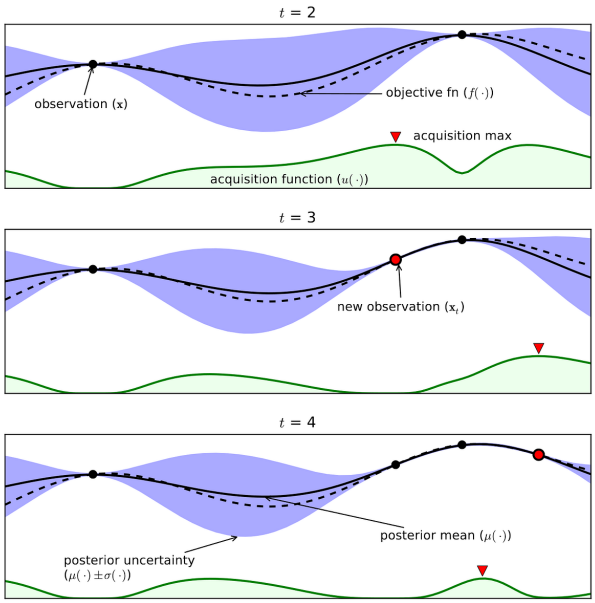

Figure 2.18: BO iterative process, reference [18]

- Initially, a surrogate model, such as a Gaussian process, is fitted to the observed data points. This surrogate model represents the unknown response surface of the objective function.
- The acquisition function guides the search by evaluating the utility of sampling a new point in the search space.
- The optimization process iteratively selects the next point to evaluate based on the acquisition function. The objective function is then evaluated at this point, and the surrogate model is updated with the new observation.
- This iterative process continues until a stopping criterion, such as a maximum number of iterations or a convergence threshold, is met.

Implementation

In this scenario, the training of a model-free controller is converted into a black-box optimization problem by prescribing the parametric form of the control law (Figure 2.19). The problem at hand involves determining the optimal gains for a Proportional-Integral (PI) controller, without any prior knowledge of the underlying system model. The aim of Bayesian Optimization (BO) is to identify the parameter values of k_P and k_I that minimize the cost function value J over an episode:

$$J(k_P, k_I) = \sum_t \frac{(\omega - \omega_{rated})^2}{N} \tag{2.69}$$

It should be noted that the cost function is a function of the gains of the PI controller, as these gains determine the dynamic response of the system.

The problem (2.68) can be reformulated as follows:

$$\mathbf{k}^* = \arg\min_{\mathbf{k}} \; J(\mathbf{k}) \tag{2.70}$$

where:

$$\mathbf{k} = \begin{bmatrix} k_P \\ k_I \end{bmatrix} \tag{2.71}$$

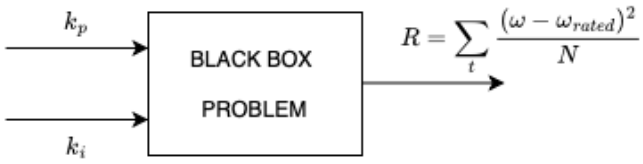

Figure 2.19: BO interpreted as a black box optimization problem of a parameterized control law

In this thesis, the Python package "gp minimize" from Skopt was utilized to execute the optimization code. Skopt is a library in Python that provides tools for Bayesian optimization [19]. Gp minimize is a specific function within Skopt that implements the Gaussian Process-based Bayesian optimization algorithm for finding the minimum of a given objective function.

Gp minimize offers various parameters for Bayesian optimization. In this work, the following parameters were used:

- The dimension of the search space, representing the bounds for the two gains of the PI controller, was selected based with their theoretical values:

  $$k_P = [0{,}100] \tag{2.72}$$

  $$k_I = [0{,}100] \tag{2.73}$$

  This ensures that the optimization process explores a relevant range of values within the parameter space to find an optimal solution.

- "gp hedge" acquisition function was used. It probabilistically choose one of the following acquisition functions at every iteration: "LCB" (Lower Confidence Bound),"EI" (Expected Improvement),"PI" (Probability of Improvement).

- The remaining parameters were left at their default values.

2.5.2 Genetic Programming (GP)

Genetic programming (GP) algorithms offer the advantage of exploring a large solution space and effectively handling non-linearities in optimization problems. In contrast to the parametric optimization approach (e.g., Bayesian Optimization), the method used in this application involves optimizing both the parameters and the structure of the program. Specifically, Genetic Programming (GP) falls within the category of Evolutionary Algorithms (EA), which draw inspiration from the natural process of evolution. EA commences with a population of candidate solutions and applies genetic operators like mutation and crossover iteratively to generate new candidates. The fitness of each solution is evaluated based on an objective function, and the fittest individuals are selected to reproduce and pass on their genetic information to the next generation. Through this process, the algorithm converges towards an optimal solution to the problem at hand. In the context of this thesis, GP will be applied to find the wind turbine's control law, utilizing its ability to optimize both parameters and program structures to achieve effective control strategies.

There are several key components that need to be defined to program with GP. One of them is the **primitive set** (Figure 2.20) which in turn is composed of:

- **function set**: is a set of arithmetic functions that allow the algorithm to test and explore during his search $(+, -, \cdot, ...)$;
- **terminal set** that is composed by:
  - **arguments**: such as some inputs of the system $(x, ...)$;
  - **constants**: some particulars constant value that one want to pass to the GP algorithm $(1.0, ...)$;

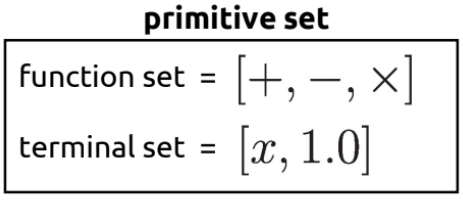

Figure 2.20: Primitive set of GP, reference [20]

In the context of the tree formalism, the representation of the primitive set can be achieved. For instance, considering a function from the function set, like multiplication with two inputs, the resulting tree structure exhibits two branches (Figure 2.21).

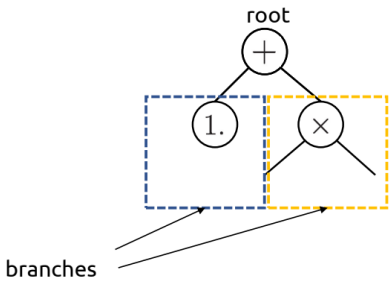

Figure 2.21: Root of the tree formalism, reference [20]

Each branch can be associated with an argument, a constant value from the terminal set, or another function with additional branches. The tree ends when all its branches are occupied by values from the terminal set (Figure 2.22).

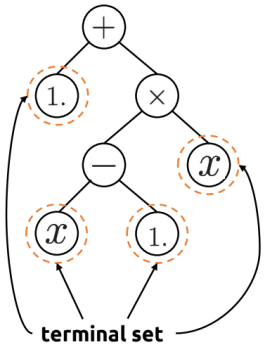

Figure 2.22: Example of a tree, reference [20]

The tree can thus be interpreted as a function:

$$f(x) = x(x-1) + 1$$

Every tree represents a function and is referred to as an **individual**. The optimization process involves evolving these individuals. The way in which individuals evolve is possible thanks to the genetic operations of:

- **replication**: involves the creation of offspring by replicating genetic material from selected parent individuals;
- **mutation**: involve creating sub-trees. By replacing a terminal set element with a function from the function set, the tree's complexity can be increased;

- **crossover**: is the genetic operation where two individuals exchange genetic material, resulting in a new individual that combines characteristics from both parents.

In Figure 2.23, the genetic operations are summarized graphically.

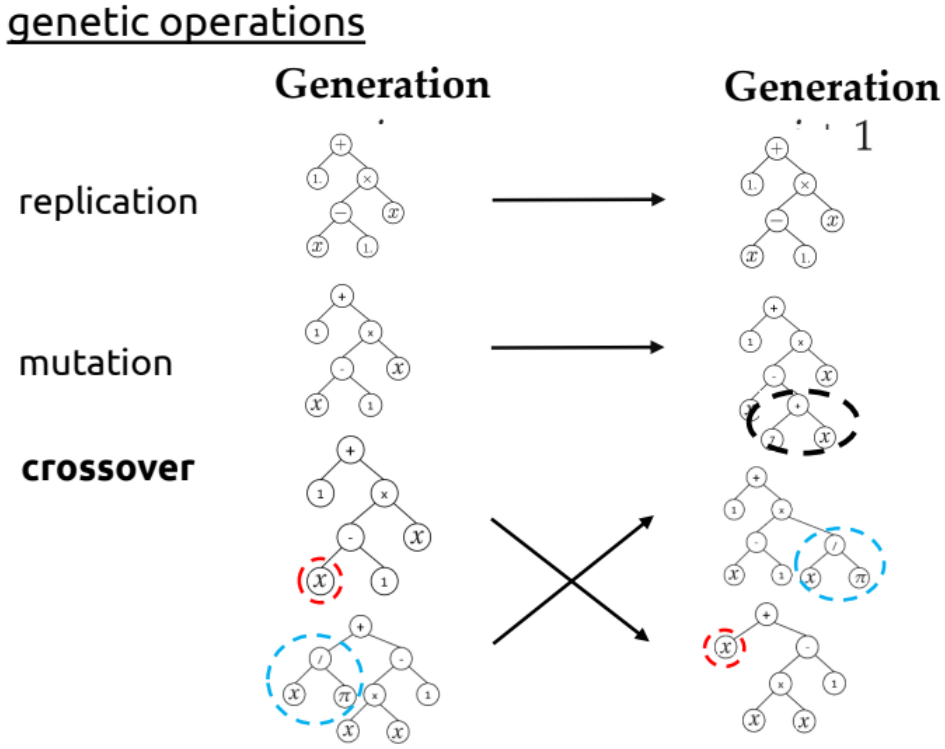

Figure 2.23: Replication, mutation and crossover in GP, reference [20]

Another essential element in GP is the **cost function**. Once the search space is defined using the primitive set, the quality of an individual can be assessed through the cost function. As mentioned in the previous sections, the cost function is a mathematical function that assigns a scalar value to an individual, representing its fitness. A lower value of the cost function indicates a better individual in terms of the optimization objective.

The workflow followed by GP is illustrated in Figure 2.24:

- the initial step involves having a given population consisting of a pool of trees;
- each tree represents a solution candidate that takes specific arguments as inputs and produces corresponding outputs to be evaluated using the cost function;

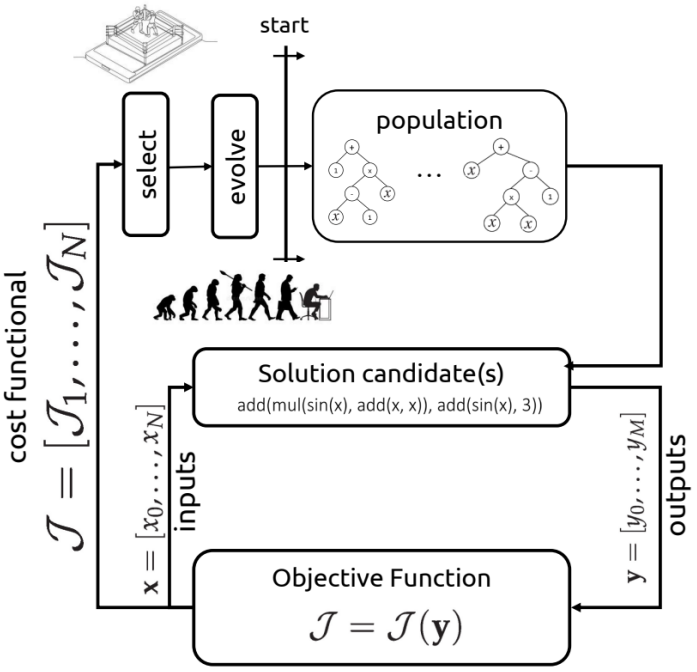

Figure 2.24: GP workflow, reference [20]

- after obtaining the fitness values for each individual in the population, a selection process is performed to retain the better individuals, ensuring their progression to the next generation;
- finally, the best individuals are evolved by applying genetic operations such as mutation and crossover, leading to the creation of new individuals for the subsequent iterations.

The initialization of the population holds great significance as it defines the initial search space in which an optimal solution is sought. The optimization process is profoundly influenced by the initial conditions, necessitating a well-defined initial population that allows for thorough exploration of all potential opportunities within the search space.

Firstly, the tree needs to be defined for:

- **depth**: refers to the distance from the root node to the farthest leaf node, representing the number of levels or generations within the tree structure;
- **length**: refers to the total number of nodes or elements present in the genetic programming tree, including both internal nodes and terminal nodes.

There are mainly 3 methods to carry out the formation of a tree:

- **full method**: given a maximum depth nodes are chosen randomly in the function set. Then the leafs are filled randomly with terminals (Figure 2.25).

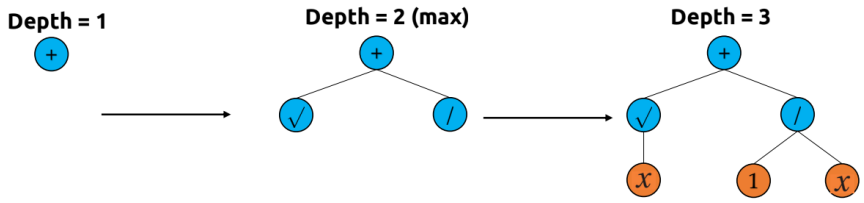

Figure 2.25: Formation of a tree with the full method, reference [20]

- **grow method**: the nodes are chosen not only by the function set but also from the terminal set randomly until the maximum depth is satisfied (Figure 2.26).

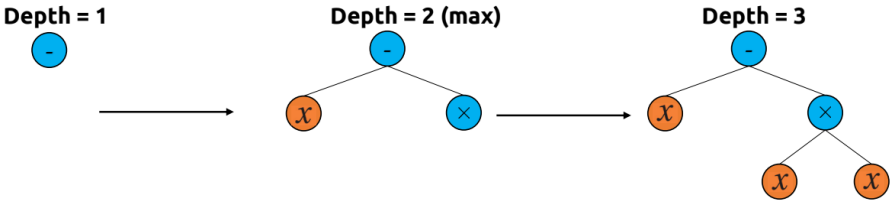

Figure 2.26: Formation of a tree with the grow method, reference [20]

- **half-half method**: is a combination of the two previous methods, where half of the population is generated using one method and the other half using the other method.

After defining the primitive set, the evaluation of each individual in the population is performed using the cost function to assess their fitness. This step allows us to measure the quality of the solutions and identify the better-performing individuals.

When presented with a list of cost functions for each individual in the population, two primary methods can be employed to select the best individuals to retain in the optimization algorithm:

- **tournament selection**: this algorithm takes randomly N individual from the population and take the best one (Figure 2.27).

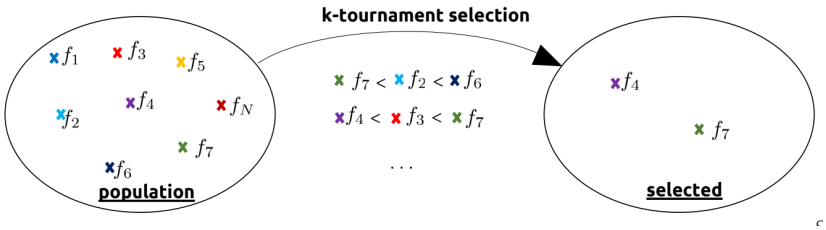

Figure 2.27: Tournament method for the selection of the individuals, reference [20]

- **fitness proportional**: this method discard worst and evolve best according to their obtained fitness (Figure 2.28).

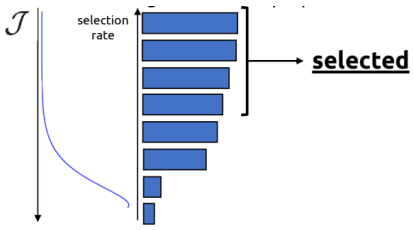

Figure 2.28: Fitness proportional method for the selection of the individuals, reference [20]

Once the individuals have been selected, the evolution process involves applying genetic operations and potentially performing eliminations. Various approaches can be utilized in this context:

- **eaSimple**: is the simplest evolutionary algorithm without the elimination part. The parent population $\mathbf{P}\mu$ is duplicated, and all the genetic operations are performed on this subset. Then, the parents are replaced by the offspring $\mathbf{P}\lambda$, and the next step of the optimization problem is carried out.

- $\mu + \lambda$ **method**: is an evolution of the previous approach. The populations of parents and offspring are kept separate, and the best individuals from both populations are selected. This method retains the best solutions from the previous optimization, aiding the convergence of the algorithm in the subsequent steps.

- (μ, λ) **method**: only individuals from the offspring population are selected for the next generation. In the literature, it is often suggested to have a larger

parent population compared to the offspring population, and a typical optimal ratio is:

$$\frac{\mathbf{P}\mu}{\mathbf{P}\lambda} \approx 5$$

Implementation

In this application of GP, the objective is to identify the optimal policy for controlling the pitch angle during above-rated operation. The following presents all the parameters and methods utilized to configure the GP application for pitch angle control.

In that thesis, the DEAP framework is utilized, which is a Python library offering tools for creating and executing evolutionary algorithms and genetic programming [21]. With DEAP, the primitive set can be defined, including the specification of the individual's size. In the genetic programming algorithm developed for this project, the operation set includes:

- `add` (addition, arity 2)
- `sub` (subtraction, arity 2)
- `mul` (multiplication, arity 2)
- `protectedSquare` (protected square, arity 1)
- `abs` (absolute value, arity 1)
- `protectedSqrt` (protected square root, arity 1)
- `protectedDiv` (protected division, arity 2)
- `protectedLog` (protected logarithm, arity 1)
- `protectedExp` (protected exponential, arity 1)
- `sin` (sine, arity 1)
- `tanh` (hyperbolic tangent, arity 1)

The `protected` functions are similar to their counterparts, but include safeguards to prevent certain mathematical errors (such as division by zero or taking the square root of a negative number). These operations were chosen based on their relevance to the problem being addressed and their ability to express a wide range of mathematical relationships.

As terminal set arguments were chosen:

- the `instantaneous rotor speed`

- the `instantaneous wind speed`
- the `time-integral of the error`:

$$\int_0^T e(t)\,dt = \int_0^T \left[\omega(t) - \omega_{rated}(t)\right]\,dt$$

On the other hand, as terminal set variables, random constants ranging from -1 to 1 and the rated wind speed value w_{rated} were taken. These choices were made based on their relevance to the problem at hand and the nature of the optimization algorithm. The inclusion of the time-integral of the error allows the algorithm to account for past performance and adjust accordingly, while the random constants offer an element of exploration to the search process. Finally, the use of the rated rotor speed value ω_{rated} ensures that the resulting solution will perform optimally under normal operating conditions.

After the initialization of the population is done, the cost function, which needs to be minimized in this case, is defined as in (2.69):

$$J = \sum_t \frac{(\omega - \omega_{rated})^2}{N} \tag{2.74}$$

A half-and-half method is utilized, as previously described, with a minimum and maximum tree depth of 1 and 3 respectively. By utilizing this method, the genetic programming algorithm can explore and evolve individuals with varying depths to find an optimal policy for controlling the wind turbine's pitch angle.
In this simulation, tournament selection is used to select individuals for the next generation in which the size of our tournament algorithm is 7.

The size of the population is a crucial parameter that needs to be carefully chosen. A larger population allows for a bigger search space, potentially leading to better solutions, but it also increases the computational time. The $(\mu + \lambda)$ method was chosen, which combines the parent and offspring populations and selects the best individuals from both. The following lines summarize the other parameters have been used :

- 50 is the number of individuals for each population;
- 65 is the maximum number of offspring;
- 0.65 is the crossover probability;
- 0.25 is the mutation probability;
- the algorithm runs for 43 generations, after which it identifies the best individual that represents the solution to the optimization problem.

2.5.3 Reinforcement Learning (RL)

Machine Learning (ML) is a field of artificial intelligence where algorithms enable computers to learn from data and improve their performance on a specific task without being explicitly programmed.

It is possible to identify three categories of ML:

- **supervised learning**: is the branch of machine learning that maps from input to output. It involves training the model on a known data set, allowing it to generalize and predict the correct output for unseen inputs.
- **unsupervised learning**: learns to map input data to itself. Its primary objective is to discover inherent structures, patterns, or representations within the data, such as clustering similar data points together.
- **reinforcement learning**: is a sub-field of machine learning that deals with the problem of an agent learning how to make sequential decisions in an environment to maximize a cumulative reward. The agent interacts with the environment and receives feedback in the form of rewards or penalties based on its actions. The goal of RL is to find an optimal policy or strategy that guides the agent to take actions that lead to the highest possible long-term reward.

Below, the most important concepts underlying RL are introduced:

- **environment**: refers to the external system or the context in which an agent operates and interacts. It encompasses all dynamics that the agent interacts with, including the state of the system, possible actions the agent can take, and the feedback received in response to its actions.
- **agent**: is an entity or an algorithm that interacts with the environment to learn and make decisions. The agent's objective is to maximize the reward it receives from the environment over time.
- **state** $s(t)$: represents the current situation or configuration of the environment at a specific point in time.
- **action** $a(t)$: represents the specific move or decision that the agent can take in response to a given state.
- **reward** $r(t)$: is a scalar value or feedback signal that the environment provides to the agent after it takes a particular action in a given state.
- **policy** π: is a strategy to map the agent's actions based on its current state or the observed environment. It specifies the agent's behavior and guides its

decision-making process to maximize the expected cumulative reward over time. The policy in RL can be either stochastic or deterministic, guiding the agent's decision-making process with randomness or determinism, respectively.

- **value function** $V(s)$ or $Q(s,a)$: is a function that estimates the expected cumulative reward an agent can achieve from a specific state or state-action pair, based on a given policy. It serves as a critical tool for guiding the agent's decision-making process to maximize long-term rewards during the learning process.

In general, a Reinforcement Learning (RL) algorithm follows the scheme of the Figure 2.29.

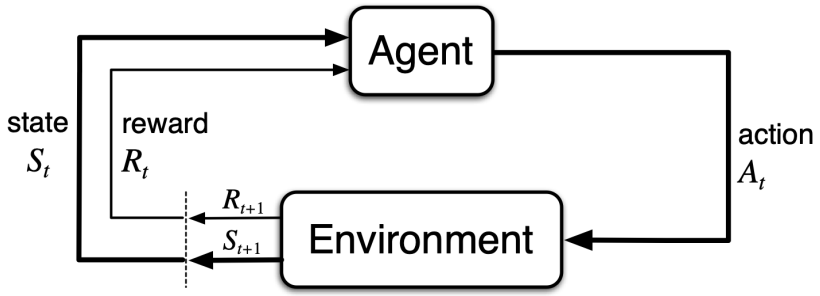

Figure 2.29: RL algorithm, reference [22]

It works by allowing an agent to interact with an environment over multiple time steps. The agent takes actions based on a policy to explore and learn from the consequences of its decisions. Through trial and error, the agent receives rewards or penalties, updating its knowledge and improving its policy to maximize cumulative rewards over time, leading to a more effective decision-making strategy.

It is important to emphasize that there are two approaches to RL: model-based and model-free. The main difference lies in the fact that the model-based approach uses a learned model of the environment to plan actions, while the model-free approach directly learns the policy or value function without explicitly modeling the environment. In this thesis, the model-free approach has been chosen.

The total discounted reward represents the sum of the rewards an agent expects to receive from the current time step to the end of the episode, where future rewards are discounted by a factor to account for the uncertainty and time preference:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.75}$$

where:

- r_{t+k+1} is the reward received at time step $t+k+1$;

- γ is the discount factor and $0 \leq \gamma \leq 1$;

The total discounted reward serves as a measure of the cumulative rewards an agent can expect to receive over time, factoring in the time preference and uncertainty through the discount factor. The discount factor determines the importance of immediate rewards relative to future rewards. A smaller γ emphasizes immediate rewards, while a larger γ values future rewards more.

As mentioned earlier, the value function is an estimate of the potential reward achievable starting from state s_k and following policy π for the action selection process until the end of the episode:

$$V^{\pi}(s_k) = \mathbb{E}_{\pi}\left[G_t \mid s_k\right] \tag{2.76}$$

where $\mathbb{E}_{\pi}$ represents the expected return conditioned on the state s_k, considering the actions chosen according to the policy π. In essence, the equation expresses the expected cumulative reward in the future when starting from state s_k and following the policy π. At the same way, it can be defined the action-value function as the expectation of the sum of rewards an agent can obtain starting from state s_k, taking action a_k, and then following policy π for the action selection process until the end of the episode:

$$Q^{\pi}(s_k, a_k) = \mathbb{E}_{\pi}\left[G_t \mid s_k, a_k\right] \tag{2.77}$$

The only difference between the value function and the action-value function is that in the latter, there is no averaging over the policy, as the action being evaluated is assumed to be already known. Both (2.76) and (2.77) are known also as Bellman equations.

In summary, the objective of RL is to maximize the value function or the action-value function of the initial state by following a policy π that has been parameterized with some weights $\mathbf{w}$.

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) refers to the combination of deep learning techniques with reinforcement learning algorithms. In DRL, Artificial Neural Networks (ANN) are utilized to approximate the value function, policy function, or a combination of both in the RL process. These neural networks enable the RL agent to effectively handle high-dimensional input spaces and learn intricate representations of the environment.

A neural network is a computational model inspired by the structure and function of the human brain. It consists of interconnected layers of artificial neurons that process and transform data, allowing it to learn patterns, make predictions, and solve complex tasks through a process known as deep learning. A neuron is a

computational units that hold a number, typically between 0 and 1. As said previously, neurons are organized into layers: an input layer, one or more hidden layers, and an output layer. Neurons within each layer are fully connected to neurons in the subsequent layer. The input layer receives data, and the hidden layers process and transform it through weighted connections and activation functions, while the output layer generates the final predictions or results of the network. The functioning of a neural network can be better explained by considering, for example, the structure proposed in the Figure 2.30.

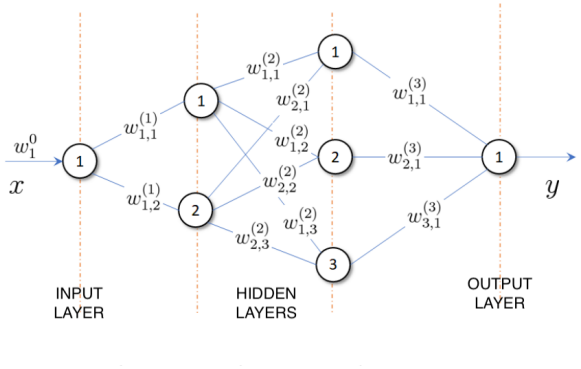

Figure 2.30: ANN scheme, reference [20]

Taking the example of the first neuron in the second hidden layer, this is obtained by:

$$y_1^{(3)} = \sigma(w_{1,1}^{(2)} y_1^{(2)} + w_{2,1}^{(2)} y_2^{(2)} + b_1^{(3)}) \tag{2.78}$$

where:

- $y_i^{(j)}$ is the value associated at the neuron i in the layer j;
- $w_{i,k}^{(j)}$ is the weight of the neuron i in the layer j that relate the neuron k;
- $b_i^{(j)}$ is the bias, an additional parameter added to each neuron that allows the network to model non-linear relationships, of the neuron i in the layer j;
- σ is the activation function. Since each neuron should be a value between 0 and 1, this function scales the output to ensure it falls within the range of 0 and 1. One of the most commonly used functions is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.79}$$

The output of the third layer it can be written in matrix form:

$$\mathbf{y^{(3)}} = \sigma(\mathbf{W_{23}} \mathbf{y^{(2)}} + \mathbf{b^{(3)}}) \tag{2.80}$$

where:

$$\mathbf{W_{23}} = \begin{bmatrix} w_{1,1}^{(2)} & w_{2,1}^{(2)} \\ w_{1,2}^{(2)} & w_{2,2}^{(2)} \\ w_{1,3}^{(2)} & w_{2,3}^{(2)} \end{bmatrix} \mathbf{y^{(2)}} = \begin{bmatrix} y_1^{(2)} \\ y_1^{(2)} \end{bmatrix} \mathbf{b^{(3)}} = \begin{bmatrix} b_1^{(3)} \\ b_2^{(3)} \\ b_3^{(3)} \end{bmatrix} \tag{2.81}$$

Therefore, using the matrix formula, starting from the input layer, one can proceed through the network layers to reach the output layer and obtain the output values. Of course, both the input layer and the output layer can consist of multiple neurons, and there can be multiple hidden layers in between. The function obtained at the end is highly nonlinear and can approximate any function.

The architecture described above is a feed-forward neural network, where data moves in a single direction, from the input layer to the output layer. There are no cycles or feedback in the architecture, and data flows only forward. Another example of neural network architectures is the recurrent neural network. It is an architecture that allows cyclic connections between neurons.

To achieve the desired performance of the ANN, it requires training, which involves adjusting the weights. The main steps to train a neural network are initializing the parameters, performing a forward pass for predictions, calculating the loss, and iteratively updating the parameters using optimization algorithms like gradient descent to minimize the loss and improve performance.

Deep Deterministic Policy Gradient

In this thesis, the Deep Deterministic Policy Gradient (DDPG) is employed. DDPG is a model-free off-policy RL algorithm that combines deep learning and policy gradient methods to handle continuous action spaces. It is particularly well-suited for tasks with continuous control, such as wind turbine control. DDPG maintains two neural networks, an actor network and a critic network (Figure 2.31).

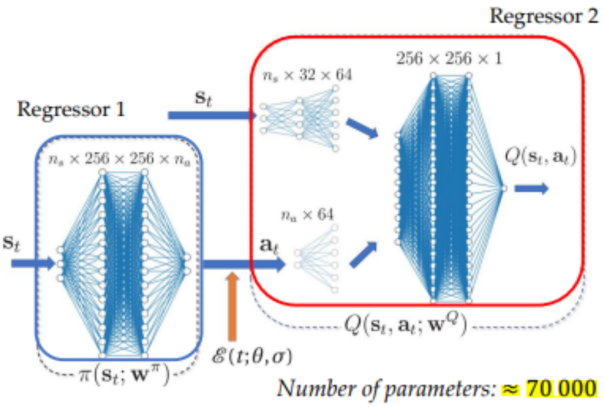

Figure 2.31: ANN for DDPG, actor-acritc model, reference [20]

The actor network takes the current state of the environment as input and outputs a continuous action. The critic network, on the other hand, takes both the current state and the action as inputs and estimates the expected cumulative reward. The core idea behind DDPG is to use the actor network to approximate the optimal policy and the critic network to approximate the optimal action-value function. The actor network is updated by following the gradient of the expected cumulative reward with respect to the policy parameters. The critic network is updated using the temporal difference error, which measures the discrepancy between the estimated value of a state-action pair and the actual observed reward. One key aspect of DDPG is the use of replay buffers, which store the agent's experiences (state, action, reward, next state) in a memory buffer. During training, samples are randomly drawn from the replay buffer to break the sequential correlation and improve the learning efficiency. Overall, DDPG leverages deep neural networks to learn both a policy and an action-value function to guide the agent's decision-making process in continuous action spaces. It combines the benefits of policy gradient methods and the stability of Q-learning to achieve effective and stable learning in complex environments.

Implementation

In this thesis, the Python package Stable Baselines [23] with TensorFlow [24] was used to implement DDPG. Stable Baselines is a powerful library that provides a collection of state-of-the-art reinforcement learning algorithms, including DDPG.

The most important aspect of the implementation is creating the environment with which DDPG interacts. Stable Baselines provides support for creating custom environments, and the structure of the environment implemented in this thesis is as depicted in the Figure 2.32.

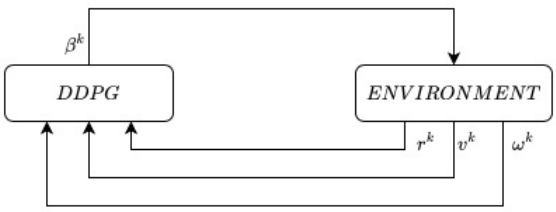

Figure 2.32: DDPG structure implemented in this thesis

The environment interacts with the agent at each time step k. At each time step, the environment provides the agent with the 2-dimensional states (rotational

speed ω^k and filtered wind speed v^k) as well as the instantaneous reward r^k:

$$r^k = -(\omega^k - \omega_{rated})^2 \tag{2.82}$$

The agent then generates a single continuous output action β^k based on this information.

The default configuration in Stable Baselines for the actor and critic networks consists of two fully connected hidden layers with 64 neurons each (MLP policy).

The number of trainable parameters in the DDPG policy using Stable Baselines with two state values and a single continuous action can be estimated to be approximately 257 parameters for the actor network and 8257 parameters for the critic network. The total number of parameters to be trained is 8514.

2.5.4 Algorithms

Each model free controller was trained using the same environment to enable meaningful comparisons.

The performance evaluation is conducted within episodes of duration T = 300 s. Each episode defines the length of one simulation, and since the simulation's time step is $\Delta t = 0.025s$, the total length of one episode is $T/\Delta t = 12000$ time steps. The chosen time T is adequate as it strikes a balance between allowing sufficient time for the controller's performance to settle and avoiding prolonged initial transients that may render them less meaningful. Additionally, from a physical standpoint, it is significant as it corresponds to approximately 60 rotations under rated conditions.

Five different wind profiles were generated using TurbSim, each characterized by varying levels of turbulence intensity and spanning the entire Region 2 (Table 2.2).

Wind	Mean Wind Speed [m/s]	Turbulence Intensity (TI) [%]
Wind1	13	5
Wind2	15	8
Wind3	17	9
Wind4	20	11
Wind5	23	12

Table 2.2: Set of the wind profiles and their characteristics used for training the model-free controllers

These profiles were randomly generated to capture the stochastic nature of real-world wind conditions. The wind signal was passed through a filtered control system, as explained earlier.

An instantaneous reward signal $r(t)$ is produced at each time step (for RL):

$$r(t) = -(\omega(t) - \omega_{rated})^2$$

and a cumulative reward signal R is produced at the end of the episode (for BO and GP):

$$R = \sum_t \frac{(\omega - \omega_{rated})^2}{N}$$

Both measure the controller performances and are maximum (for RL) and minimum (for GP and BO) when the control law is optimal.

BO and GP updates the control law at the end of the episodes (line 11-12) while the reinforcement learning algorithm updates at each episode.

The following algorithm was followed:

Algorithm 1 Approach for the Optimal Control Law Derivation

1: Initialize a possible control law $\beta = \pi(v; w)$;
2: **for** n in (1, $N_{episodes}$) **do**
3: Initialize the inflow velocity v_∞ by randomly choosing one of the 5 wind profiles in Region 2;
4: **for** t in (1, T) **do**
5: Feed the current state $s = (v_t, \omega_t)$ to the controller;
6: Get action β_{t+1} following current control law π;
7: Evolve the wind-turbine state $s_t \rightarrow s_{t+1}$;
8: Compute the reward r_t associated with the change of state $s_t \rightarrow s_{t+1}$ and current action β_t;
9: Update control law weights (for DDPG);
10: **end for**
11: Update control law weights (for BO);
12: Update control law π (for GP);
13: **end for**

Chapter 3

Experimental Methods and Objective

3.1 Control Objective

The objective of the experimental campaign is to test some of the various controllers developed in the numerical part in wind tunnel with a scale model of a wind turbine. The intention is to provide a proof of concept for the actual functioning in a real application of the previously developed controllers.

The control objectives of this experimental campaign were to develop a control system that reproduces the same conditions as Region 2. As explained earlier, the goal is to maintain a constant rotational speed by simulating a condition where the wind speed is above the rated speed. However, the wind turbine model used does not allow control through pitch. For this reason, rotational speed control is achieved through torque. This leads to slight differences in the control law compared to those developed in the numerical part.

The paragraph structure closely resembles the methodology employed in the numerical section. It begins with an introductory section explaining the experimental setup, followed by a description of how the power coefficient curve of the turbine is derived. The subsequent section presents the implementation of the developed controllers (PI, PI with scheduling, and PI with BO) in the experimental phase. Lastly, the setup is modified in the final phase by introducing a second wind turbine upstream to generate a turbulent wake that affects the second controlled turbine. This particular test case facilitates the statistical evaluation of the controller's performance under these specific conditions.

3.2 Experimental Setup

The setup for this study had already been designed by my two advisors at the von Karman Institute, Emmanuel Gillyns and Lorenzo Schena [8]. My contribution to the research involved refining the existing setup by enhancing the architecture, for instance, by improving the master-slave architecture of Arduino. The experimental setup is illustrated Figure 3.1.

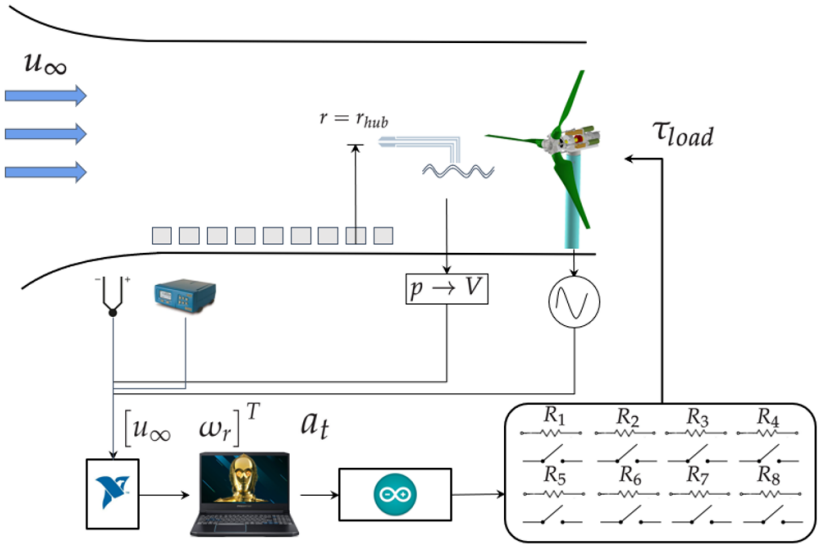

Figure 3.1: Experimental Setup, courtesy of Lorenzo Schena and Emmanuel Gillyns

The experimental campaign was carried out in the subsonic wind tunnel L2B at the von Karman Institute. This wind tunnel is an open-circuit facility designed for low-speed testing. A diagram of the wind tunnel is provided in Figure 3.2 and 3.3 for a visualization of the wind tunnel's layout.
The facility incorporates an axial fan, driven by a variable-speed DC motor, positioned at the end of the wind tunnel, enabling a continuous velocity range in the test section from 0 to 35 m/s. Air is drawn in through an air intake and then directed through meshes to dissipate turbulent structures. Afterwards, it undergoes acceleration within a test chamber featuring minimal residual turbulence before being expelled and decelerated by a divergent. The test section of the facility can be modified, with the L2A test section measuring $S_{t_1} = 0.35m \times 0.35m$ and the L2-B test section $S_{t_2} = 0.5m \times 0.6m$ that is employed for this experimental campaign. The L2B test section is of the open jet type, which means that the model needs to be positioned within the potential core zone with dimensions of

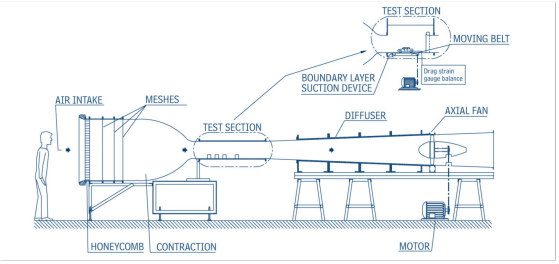

Figure 3.2: Scheme of the L2-B wind tunnel at the von Karman Institute, reference [25]

Figure 3.3: L2-B at the von Karman Institute

$0.35m \times 0.35m$. Essentially, it is as if a $0.35m \times 0.35m$ section is being utilized, but with the advantage of having more space available for model installation.

The experiments involve the use of a three-bladed wind-turbine scale model, which was designed by PhD Nicolas Coudou at the von Karman Institute. This model has a rotor diameter of 0.15 meters and a hub height of 0.2 meters. It serves as a representative scaled version of a 2 MW offshore wind turbine, specifically scaled at 1/440 of the Vestas V66-2MW model. The rotors of the wind turbine are constructed using the HAM-STD HS1-606 airfoil and are optimized through the Blade Element Momentum (BEM) theory to perform efficiently under an incoming velocity of 8 m/s at hub height [26]. Each rotor is attached to a direct current (DC) motor (Faulhaber 1331T006SR) that functions as a generator, allowing power extraction from the incoming wind and enabling turbine control. The motor is directly connected to the rotor, and an encoder is integrated within the motor to read the rotational speed of the rotor. In Figure 3.4 is presented an image of the turbine model.

The blockage caused by the turbine, considering the test section of $S_f = 0.35m \times 0.35m$, can be calculated as follows:

$$B_F = \frac{S_m}{S_t} = \frac{\pi(7.5cm)^2}{(0.35m)^2} = 14.425\% \tag{3.1}$$

The blockage is significantly high and approaching the limit that guarantees accurate measurements. In this case, the increased blockage could affect the velocity measurements. Unfortunately, it was not possible to decrease this value due to limitations in the available time.

A Pitot tube was placed in front of the wind turbine at the hub height to receive feedback on the wind speed state of the system (Figure 3.5). Although the Pitot tube position would have been more appropriate at the beginning of the test section to obtain an accurate indication of the free stream velocity, this configuration was

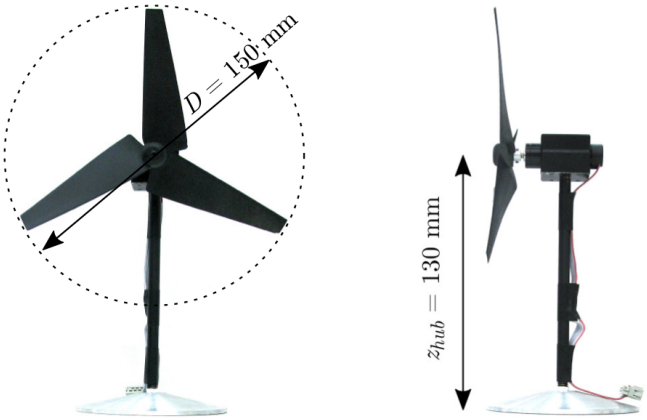

Figure 3.4: Wind turbine model utilized in the experimental campaign, designed by Nicolas Coudou [26]

chosen because the relevant wind information for control purposes is the one at the center of the rotor. This configuration also reflects real-world applications as the only feasible measurement of the wind is behind the rotor.

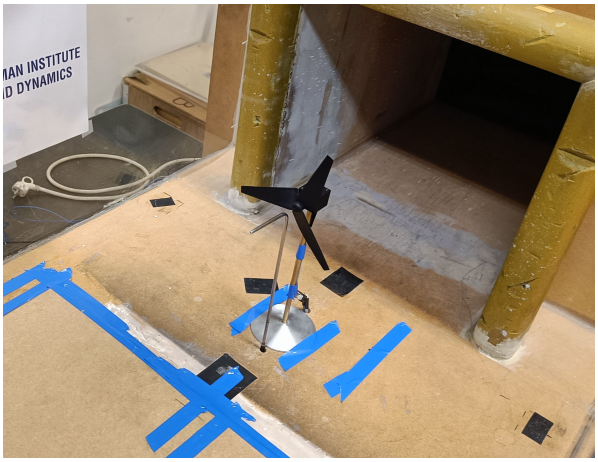

Figure 3.5: Wind turbine position in the test section and the relative Pitot tube position used for wind speed feedback

The wind speed measurement was obtained using a differential pressure transducer connected to the total pressure tap of the Pitot tube and a static pressure tap on the side wall of the test section. Prior to data acquisition, the transducer was calibrated, and the calibration procedure can be found in Appendix B.

A thermocouple was used to obtain temperature measurements in the test section

(which are useful for calculating density). The thermocouple is pre-calibrated with a calibration constant of:

$$K = 100°\text{C}/V$$

The VKI's local servers provide access to atmospheric pressure data, which is measured using a Druck DPI 150 Precision Pressure Indicator installed in the control room of S1 facility at the institute. The data acquisition process can be easily managed using Python, allowing users to retrieve the information directly from the VKI servers.

The rotation speed was acquired using the encoder mounted on the wind turbine generator. The encoder was then connected to an Arduino, which is capable of calculating the rotation speed and sending it to the PC.

The measurements of extracted electrical power were obtained by multiplying the voltage difference across the generator with the current flowing through the circuit. The load to be applied to the wind turbine was achieved by connecting electrical resistors to the generator. By using relays controlled by Arduino, it is possible to change the value of the resistors.

All the data was acquired using a National Instruments DAQ system, with a 4-channel card connected. The data acquisition was performed using the Python package called nidaqmx [27]. This solution allows for a more straightforward management of the data acquired through the NI board, as it is directly extracted within the Python code.

3.2.1 Arduino Architecture

The configuration used involves connecting two Arduino boards together using the I2C communication protocol, which utilizes two signal lines: SDA (Serial Data Line) and SCL (Serial Clock Line) (Figure 3.6). In this setup, one Arduino is designated as the master while the other Arduino acts as the slave:

- The Arduino slave is responsible for continuously measuring the rotation speed of the turbine. The Arduino interrupt function was used to detect each rising edge of the electrical signal. Knowing that the encoder resolution is 50 pulses per revolution, the interrupt function made it possible to measure the time difference Δt between two rising edges of the signal. From the delta time, the rotation speed of the turbine could be calculated:

$$\omega = \frac{2\pi}{\Delta t \cdot 50}[rad/s] \tag{3.2}$$

  A two-Arduino configuration was used because the use of the interrupt function in Arduino blocks the execution of any other command in the main loop. Since the master Arduino is responsible for sending commands to the relays, it

is not possible to measure the speed with the master Arduino as it would interrupt these functions. Therefore, the sole task of the Arduino slave is to continuously read the rotational speed.

- The Arduino master is responsible for requesting the rotation speed from the slave as needed and sending commands to the relays. The master Arduino can communicate with the PC via the serial port using the Python package Pyserial [28]. Some of the functionalities that can be used through Python on the Arduino master are:

  – Changing the acquisition frequency of the rotation speed: is accomplished by using the request event function in Arduino, which utilizes the I2C communication protocol between the two Arduinos. The request event is invoked at different time intervals based on the acquisition frequency sent to the master Arduino via serial communication.

  – Controlling when to start and stop the data transmission from the Arduino slave: is also managed through a flag that activates/deactivates the request event function of Arduino. This flag is responsible for initiating or halting the data transmission as needed.

  – Sending commands to the relays to change the resistances: is achieved by using a serial command that activates or deactivates the pins of the relays.

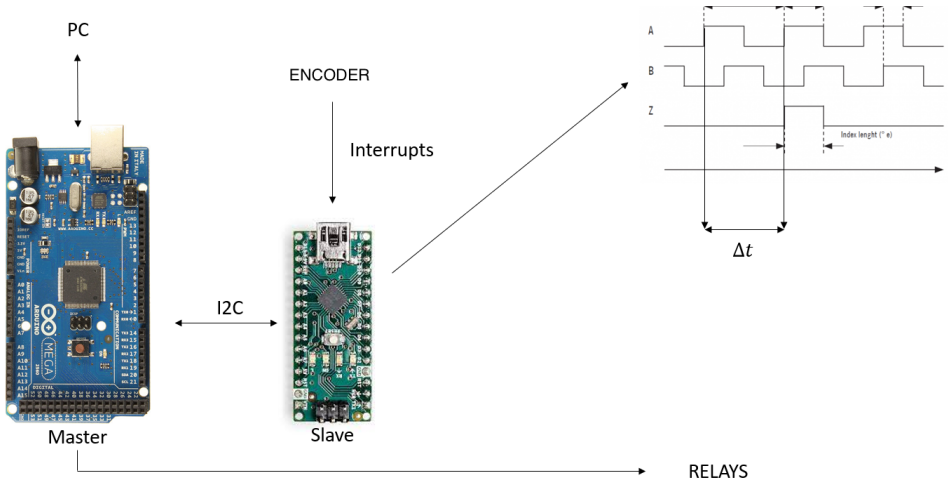

Figure 3.6: Master-Slave Arduino architecture

3.2.2 Power Measurement and Resistors

Figure 3.7 shows the electrical schematic.

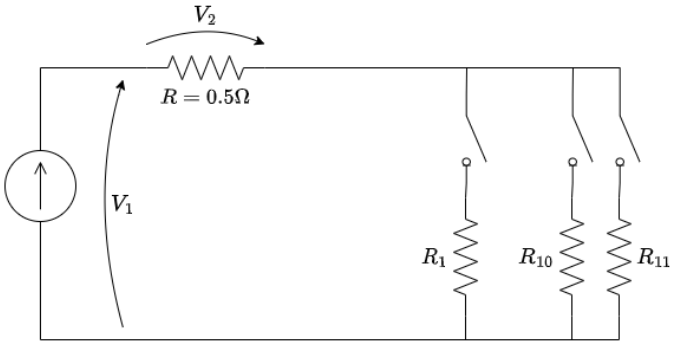

Figure 3.7: Electrical circuit employed for power measurement and load adjustment on the turbine

The power measurement was conducted by measuring the voltage difference V_1 and V_2. A series resistor of 0.5 Ohm was connected to measure the current flowing through the circuit:

$$I = \frac{V_2}{0.5\Omega}$$

Therefore, the electrical power produced by the generator is calculated as follows:

$$P = V_1 \cdot I$$

The resistors connected in parallel are 11, and they have the following values: $R_1 = 1.5\Omega$, $R_2 = 3\Omega$, $R_3 = 3\Omega$, $R_4 = 12\Omega$, $R_5 = 24\Omega$, $R_6 = 48\Omega$, $R_7 = 96\Omega$, $R_8 = 192\Omega$, $R_9 = 384\Omega$, $R_{10} = 768\Omega$ and $R_{11} = 1536\Omega$.

The combinations that can be made using the relays with the 11 resistors are:

$$\text{actions} = 2^{11} = 2048$$

Therefore, the system can be manipulated with 2048 different resistance values. The minimum and maximum values of these resistances that can be applied, considering the resistors are connected in parallel, are respectively:

$$R_{min} = \frac{1}{\frac{1}{R_1} + ... + \frac{1}{R_{11}}} = 0.66696\Omega$$

$$R_{max} = 1536\Omega$$

In the following, each resistance value will be indicated as a numerical value that corresponds to an action on the system. The value 1 corresponds to the maximum resistance, while the value 2048 corresponds to the minimum resistance value.

3.3 Power coefficient curve

The first experimental test aims to obtain the power coefficient (C_P) curve. Since the turbine has a fixed pitch angle, the 2D curve becomes a 1D curve relative to a specific value of the pitch angle.

3.3.1 Measurement setup

The measurement chain is summarized in Figure 3.8.

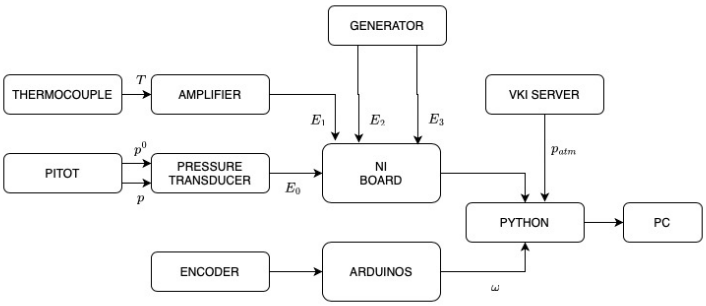

Figure 3.8: Power coefficient measurement chain

The experimental procedure began by setting a certain rotational speed of the wind tunnel fan while keeping the wind speed constant. Subsequently, data acquisition took place by changing the applied action value through the relays, thus setting different resistance values, and gradually acquiring the various data points.

From the NI board, the potential difference coming from the pressure transducer E_0, the amplifier of the thermocouple E_1 and the generator E_2 and E_3 were acquired from the smallest to the largest action in a Python script using:

$$t_s = 15s$$

$$f_{s,NI} = 1000\text{Hz}$$

The advantage of using serial communication with Arduino is that while the data is being acquired with the NI board, Arduino continuously fills the serial port with data, which can be read after the NI acquisition is completed. This allows for simultaneous measurements of rotation speed along with the other data:

$$f_{s,Arduino} = 100\text{Hz}$$

A certain amount of time was waited between each measurement from one action to another to eliminate disturbances caused by transients:

$$t_{transient} = 10s$$

As mentioned before, the atmospheric pressure data was obtained through a Python script that interacts with VKI servers.

3.3.2 Post processing

The post-processing of the data was performed by taking the average values over t_s. The test chamber temperature is obtained using the calibration relationship provided by the manufacturer, as indicated in (3.2):

$$T = K \cdot E_1 \tag{3.3}$$

From the temperature, the density in the test chamber can be obtained:

$$\rho = \frac{p_{atm}}{RT} \tag{3.4}$$

From the calibration of the pressure transducer (B.1), it is possible to derive the value of dynamic pressure $p_q = p^0 - p$ measured by the Pitot tube, from which the velocity can be determined:

$$v = \sqrt{\frac{2p_q}{\rho}} \tag{3.5}$$

from which it is possible to calculate the available power in the wind:

$$P_{wind} = \frac{1}{2}\rho\pi R^2 v^3 \tag{3.6}$$

and the tip speed ratio:

$$\lambda = \frac{\omega R}{v} \tag{3.7}$$

The electrical power produced by the generator was obtained as indicated in (3.2.2):

$$I = \frac{E_3}{0.5\Omega} \tag{3.8}$$

$$P = E_2 \cdot I \tag{3.9}$$

from which it is possible to determine the electrical power coefficient:

$$C_P = \frac{P}{P_{wind}} \tag{3.10}$$

It should be noted that the obtained power coefficient is related to electrical power. If one wants to obtain the coefficient of mechanical power developed, one should introduce a efficiency factor η that quantifies the relative losses in the conversion system. The same procedure was repeated for different wind speeds. Results are presented in (4.2.1).

3.4 Baseline Controllers

The workflow of the experimental closed-loop wind turbine controller is summarized in Figure 3.9.

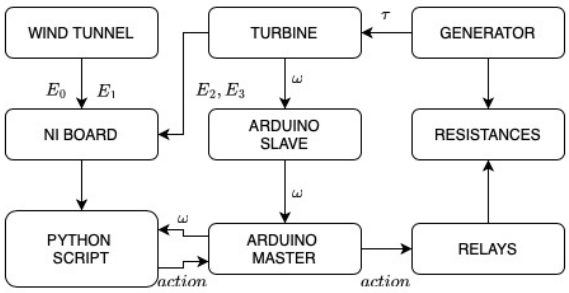

Figure 3.9: Scheme of the closed-loop experimental controller

The most crucial part of the controller is the Python script that wraps the entire facility. At each time step, dynamic pressure and temperature data are acquired from the NI board. Meanwhile, the serial communication receives data from the Arduino master. At the end of each acquisition, the data is processed in real-time as done in (3.3.2). Specifically, the data was acquired (thus defining the control time step) every time step of $t_s = 0.2s$, with a sampling frequency of NI $f_{s,NI} = 1000Hz$ and Arduino $f_{s,Ard} = 100Hz$. These processed data serve as inputs to the controller, also implemented in the Python code, which produces the action to be applied. This action is sent to the Arduino master, which controls the relays to connect/disconnect various resistances attached to the generator. As a result, the load on the generator changes, and consequently, the torque applied to the wind turbine model also changes. It should be noted that the final outcome is a closed-loop controller. The controller utilizes wind speed and turbine rotation speed as feedback or system states at each time step. For each time step, there are certain characteristic times associated with this procedure, which are summarized in Figure 3.10.

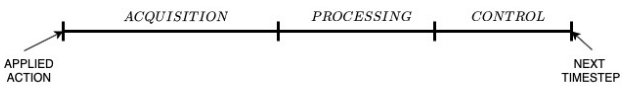

Figure 3.10: Characteristic times inside each time step

For each time step:

- a certain time interval is spent acquiring data from the NI board and through serial communication with Arduino. The data acquisition time can be specified via Python, but there is a stochastic component of acquisition time that cannot be predicted, which is around 0.15 seconds. This delay exceeds the data acquisition time, during which data from the NI board is not received, but the data from the turbine rotation speed is still received. This is because the rotation speed data comes from the buffer of the serial communication with Arduino, which continues to fill up during that time.;

- the acquired data is processed in real time within another time interval during the time step;

- with the processed data, another portion of the time step is required for the calculations to be performed by the controller in order to generate the control output.

3.4.1 Experimental Linearization

As mentioned in (4.2.1), the objective of the controller is to accurately simulate control in Region 2 by utilizing torque as the control variable. This aligns with the desired control objective in the numerical section of the study. In (2.3.2), to find the gains of the PI controller, the system was linearized around the chosen operating point. However, in this particular case, where there is no experimental model of the turbine, the associated linear system can be determined experimentally. As done in (2.3.2), the initial design of the PI controller was performed choosing a steady-state operating point of linearization. In this experimental case, the chosen operating point was:

$$v_0 = 3.285 m/s$$

$$\omega_0 = 400 rad/s$$

$$action_0 = 240$$

To obtain the linear system associated with this operating point, the time constant and gain were experimentally derived by applying a known step action to the system $\Delta action = 40$. By analyzing the system's behavior following this perturbation, the time constant and gain can be estimated.

The transient on the rotational speed caused by this step action was recorded/acquired with $f_{s,Arduino} = 100Hz$ (Figure 3.11).

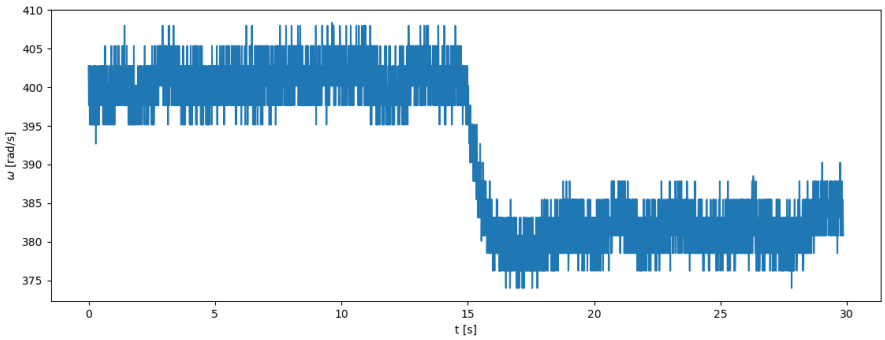

Figure 3.11: Rotational speed of the model wind turbine during the transient caused by a known step in torque

Assuming that the system is linear near this operational point, the transient response was fitted using the solution of a classical linear ODE:

$$\omega(t) = \omega_0 e^{-\frac{t}{\tau}} + \omega_1 \left(1 - e^{-\frac{t}{\tau}}\right) \tag{3.11}$$

Fitting (3.11) the gain and time constant associated with the linear system were determined (Figure 3.12):

$$\tau_t = 0.949s$$

$$K = \frac{\Delta\omega}{\Delta action} = -0.52559 rad/s$$

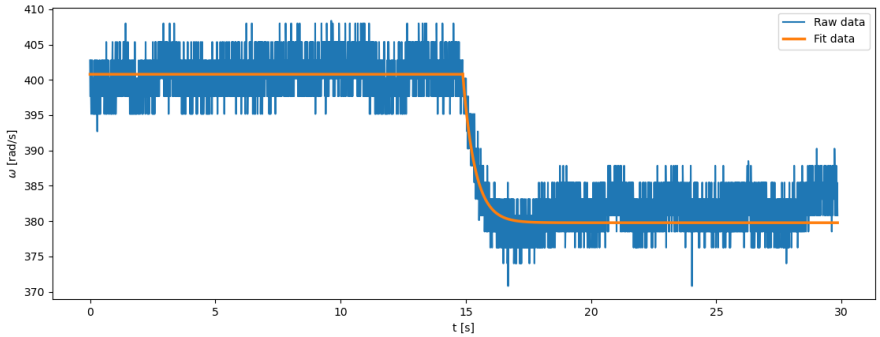

Figure 3.12: Raw data of the rotational speed during the transient in blue, with the fitted curve represented in orange

Therefore, the linear ODE that can approximate the dynamics of the turbine around the operational point is:

$$\tau_t \dot{\Delta\omega} = -\Delta\omega + K\Delta action \tag{3.12}$$

which can be rewritten in the form:

$$\Delta\dot{\omega} = A\Delta\omega + B_a\Delta action \tag{3.13}$$

It is important to emphasize that equation (3.13) does not have a physically meaningful interpretation. In this case, since there is no available correlation for the torque applied to the rotor, instead of using the torque, the applied action was used as a variable, which does not have a direct physical meaning as it is a discrete value defined arbitrarily. Nevertheless, the linear system obtained in this way is useful for tuning the controller, which has the applied action as its output.

Knowing the values of coefficients A and B_a, the tuning can be performed as done in (2.3.2) by assigning the values of ω_n and ζ derived from the second-order transfer function in order to achieve the desired response:

$$k_p = -\frac{1}{B_a}(2\zeta\omega_n + A) \tag{3.14}$$

$$k_i = -\frac{\omega_n^2}{B_a} \tag{3.15}$$

3.4.2 PI Controller

The resulting control law of the baseline PI controller is as follows:

$$action(\omega) = action_0 + k_p(\omega - 400rad/s) + k_i\int_0^t(\omega - 400rad/s)d\tau \tag{3.16}$$

As can be seen from the previous equation the controller takes as input only the rotation speed ω at each time-step.

Several tests with different values of ζ (damping ratio) and ω_n (natural frequency) were conducted to observe the effects on the response of the system when these parameters are changed. Results are presented in (4.2.2).

3.4.3 PI Schedule

As seen in the numerical part, the scheduling of the PI controller brings significant benefits to the control law.

Since the model of the experimental turbine is unavailable, it is being attempted to derive it through experimentation. In this case, in order to develop a scheduling controller, the action values required to maintain a constant rotational speed (400 rad/s) under steady-state conditions at different wind velocities were determined experimentally.

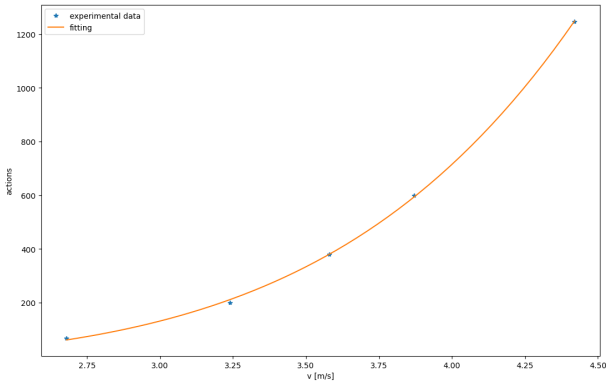

Figure 3.13: Fitted curve of the action values required to maintain a constant rotational speed under steady-state conditions

The stationary points were fitted with the following curve:

$$action(v) = a + b \cdot v^c \tag{3.17}$$

The best fit for those points is (Figure 3.13):

$$action(v) = -21.5879 + 0.37838 \cdot v^{5.4630328}$$

with an R^2 of:

$$R^2 = 0.999879$$

The weights $k_P(v)$ and $k_i(v)$ were determined as explained in (3.4.1), with $\omega_n = 0.6$ and $\zeta = 1$, varying the wind speeds as done in (2.3.3).

The resulting PI controller, therefore, is as follows:

$$action(\omega, v) = action(v) + k_p(v)(\omega - 400rad/s) + k_i(v) \int_0^t (\omega - 400rad/s)d\tau \tag{3.18}$$

In this case, the controller takes as input both rotational speed ω and wind speed v from the Pitot tube placed at hub height at each time-step.

A preliminary test of the PI controller with scheduling was tested by varying the wind speed in the wind tunnel and results are presented in (4.2.3).

3.5 Model Free Controller

Just like in the numerical part, experimental efforts were made to apply the principles of model-free controllers in a practical application. In this case, for the sake of simplicity and immediacy, Bayesian Optimization (BO) was applied to determine the weights of the PI controller, as described in Section (2.5.1).

3.5.1 PI with BO

The application of BO for experimental rotor speed control is exactly the same as described in (2.5.1). What changes in this case is the parameterized control law that BO needs to optimize. In this case the following policy was passed to BO with the following parameterization:

$$action(\omega, v) = action(v) + k_p(\omega - 400rad/s) + k_i \int_0^t (\omega - 400rad/s)d\tau \quad (3.19)$$

with:

$$action(v) = a \cdot v^b \quad (3.20)$$

The controller does not directly rely on the "model" of the turbine. In fact, in previous sections, assumptions were made regarding the dynamics (such as the assumption that it follows a linear model near a linearization point), which were then experimentally derived. However, in this case, we directly feed the parameterized control law to the optimization of Bayesian Optimization (BO) in order to obtain the coefficients from the optimization. For this reason, it can still be considered model-free approach. The result of BO will be the four coefficients a, b, k_p and k_i.

The cost function for each episode is defined as stated in (2.5.1):

$$J(a, b, k_p, k_i) = \sum_t (\omega - 400rad/s)^2 \quad (3.21)$$

The dimension of the search space, representing the bounds for the four parameters, were:

$$a = [0,5] \quad (3.22)$$

$$b = [0,5] \quad (3.23)$$

$$k_p = [0,20] \quad (3.24)$$

$$k_i = [0,20] \quad (3.25)$$

The stopping criterion for the optimization was set to reach a maximum of 30 episodes. Each episode lasted for $T = 60$ s with a time step of $\Delta t = 0.3$ s, resulting in a length of approximately $T/\Delta t \approx 200$ time steps. In each episode, to enhance training, the rotation speed of the wind tunnel fans was randomly varied to change the wind velocity.

In this case, the 'gp hedge' acquisition function (implemented in 'gp minimize') was also used, and the remaining parameters were left at their default values.

Results of this implementation is in section (4.2.4).

3.6 Two Turbine Test Case

Lastly, a specific experimental setup was designed to simulate real-world conditions in wind turbine farms, where one wind turbine is positioned downstream of another (Figure 3.14). This configuration served as a test case to validate the performance of the controller in mitigating wake interference and optimizing overall performance.

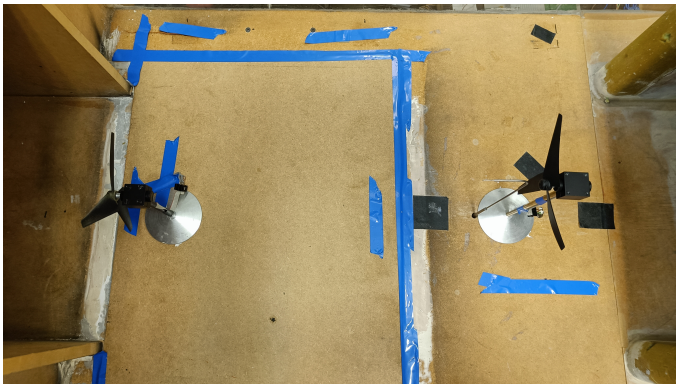

Figure 3.14: Experimental setup modified by incorporating an additional upstream wind turbine

To assess the impact of the upstream wind turbine, the first step involved measuring the wake generated by the first turbine in front of the second, which is the turbine to be controlled.

The wake velocity profile was measured using Pitot tube measurements. It is known that the Pitot tube is not the ideal method for characterizing turbulent airflow. A hot-wire probe would have been more appropriate for this purpose. However, by taking certain precautions such as minimizing the length of the connecting tubes between the Pitot tube and the pressure transducer, a reasonably accurate indication of the average wake velocity profile can be obtained.

The Pitot tube can be vertically maneuvered using a tower installed beneath the test section (Figure 3.15).

The wake profile measurement was carried out by moving the Pitot tube vertically and acquiring the data for:

$$f_s = 5000Hz$$

$$t_s = 15s$$

The acquisition time is sufficiently long to capture the average flow behavior along the wake. The obtained average velocity profile of the wake is shown in the section (4.2.5).

The final part of this thesis focused on testing the PI scheduling and optimized

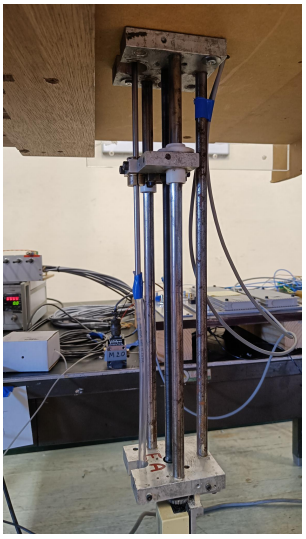

Figure 3.15: Tower installed beneath the test section to enable vertical movement of the Pitot tube

PI controllers with BO on the disturbance caused by the wake of the first turbine. The results of this latter part are presented in (4.2.5).

Chapter 4

Results

4.1 Numerical Results

In this section, the numerical results obtained in this thesis are presented. Firstly, a comparison against ROSCO is conducted to assess the robustness of the in-house baseline controllers. Next, the learning performances of the different model-free controllers are individually examined and compared. Finally, the results of each controller in terms of performance is presented.

4.1.1 Validation of Baseline Controllers against ROSCO

The validation of the developed baseline controllers, along with the ODE solver, has been performed by comparing the obtained results with those obtained using the ROSCO (Rotor Optimization for COntrol) software [4]. ROSCO is a Python package specifically designed for wind turbine control and optimization. It provides a set of tools and functionalities to simulate and analyze wind turbine behavior, as well as to develop and test control algorithms. Overall, ROSCO provides a comprehensive framework for wind turbine control analysis, optimization, and algorithm development, making it a valuable tool for researchers and engineers in the field of wind energy.

The validation of the various baseline controllers was performed by applying the same wind profile with the parameters of the NREL 5 MW wind turbine (1.2.1). The obtained results were compared in terms of rotor rotational speed. Comparisons were made for both torque control and pitch control in Region 1 and 2 respectively.

A preliminary test was conducted with a highly turbulent wind profile, having an average velocity of 5 m/s for a duration of 200 seconds, in Region 1 to assess the performance of the $K\omega^2$ controller.

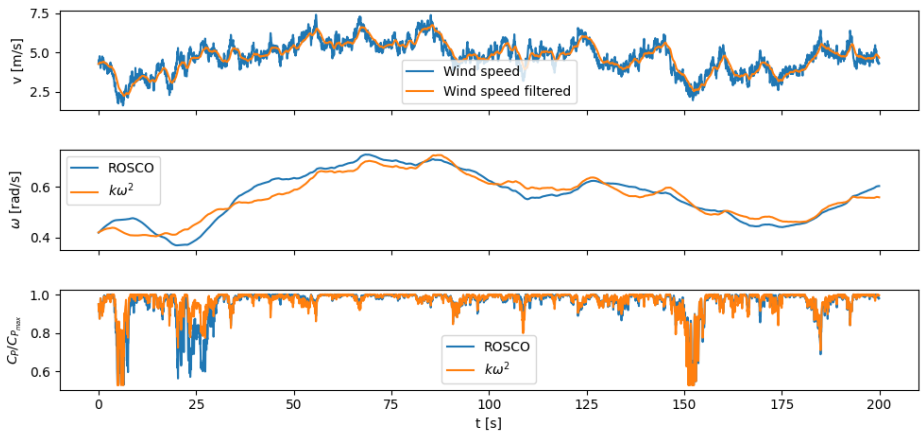

Figure 4.1: Validation of the baseline $k\omega^2$ controller against ROSCO in Region 1

As can be observed from Figure 4.1, the developed controller aligns well with the ROSCO controller.

Results were also compared with PI controllers, both with and without scheduling, on a turbulent wind profile with an average velocity of 16 m/s in Region 2 for a duration of 200 seconds.

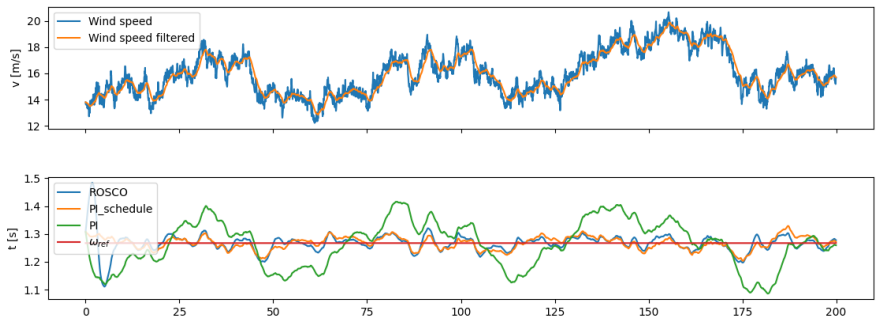

Figure 4.2: Validation of the baseline PI controllers against ROSCO in Region 2

In this case as well, there is a strong agreement between the results obtained from ROSCO and the ones developed in this study (Figure 4.2). Furthermore, it is highlighted that implementing scheduling on the PI controller brings benefits in terms of control performance.

The validation of these baseline controllers with state-of-the-art software provides a solid foundation for comparing them with other controllers developed subsequently. This allows for meaningful comparisons and assessments of the

performance and effectiveness of different control strategies. By establishing a strong baseline, it becomes possible to evaluate and benchmark the performance of new controllers developed in the following parts against these validated baseline controllers.

4.1.2 Data Driven Controllers Leaning Curves

In this section, the learning curves of the different model-free controllers are presented. Studying the learning curve offers insights into the performance and convergence behavior of the controllers over time. By analyzing the learning curve, one can observe how the controller's performance improves or stabilizes with increasing experience or training iterations. This information is crucial for understanding the effectiveness and efficiency of the control algorithms.

BO

Figure 4.3 displays the plots illustrating the cost function and the values of the two parameters as functions of the optimization episodes, as defined in Section 2.5.4.

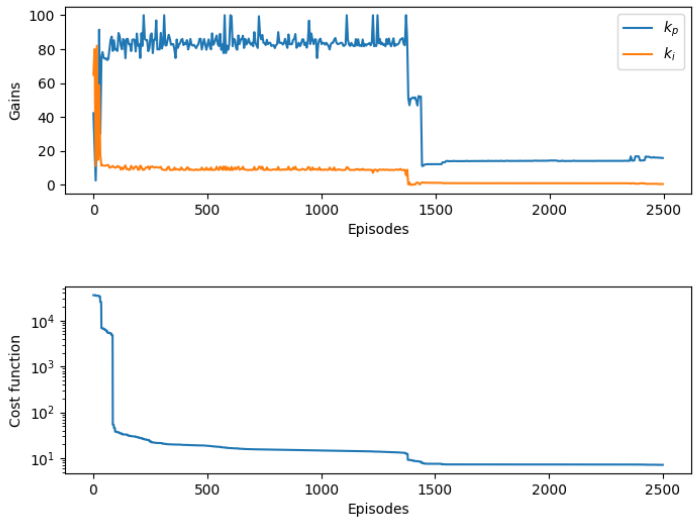

Figure 4.3: Learning curve of the numerical optimization using BO

The optimization of weights using Bayesian optimization led to the following result:

$$k_p = 15.852$$

$$k_i = 0.507$$

Comparing the weights obtained with the scheduling in section (2.3.3), it is evident that the proportional gain is significantly higher than the scheduling values. On the other hand, the integral gain appears to be of the same order of magnitude, and it is an average of the integral gains derived from the scheduling.

GP

Through the process of genetic programming the control law was obtained, showcasing the discovered relationships between input parameters and the desired output:

$$\begin{gathered}\beta(t,\omega,v) = \omega + \int_0^t (\omega - \omega_{rated})dt + \\ -log\left(tanh\left((log\left(tanh\left(H\right)\right) - \omega_{rated}\right) - log\left(\omega - \omega_{rated}\right)\right)\right)\end{gathered} \tag{4.1}$$

where H is:

$$\begin{gathered}H = log\left(\frac{\sqrt{\int_0^t(\omega - \omega_{rated})dt}}{tanh\left(\int_0^t(\omega - \omega_{rated})dt\right) + log\left(\int_0^t(\omega - \omega_{rated})dt\right)}\right) + \\ +\sqrt{v \cdot \int_0^t (\omega - \omega_{rated})dt} - sin\left(log(\omega)\right) - 1\end{gathered} \tag{4.2}$$

The control law obtained through GP is complex, exhibiting high nonlinearity and utilizing the integral of the error. It incorporates a wide range of functions from the function set, reflecting the nonlinear nature of the control problem. The resulting function tree captures the dependencies and past information through the integration of the error. This complexity is expected, given the highly nonlinear nature of the wind turbine control problem. Figure 4.4 illustrates the cost function for each episode.

RL

Figure 4.5 presents the learning curve of reinforcement learning over 200 episodes, depicting the cumulative reward R. The cumulative reward is obtained by summing the individual instantaneous rewards r for each time step within an episode.

It can be observed that the method converges, and increasing the number of iterations would likely lead to further convergence.

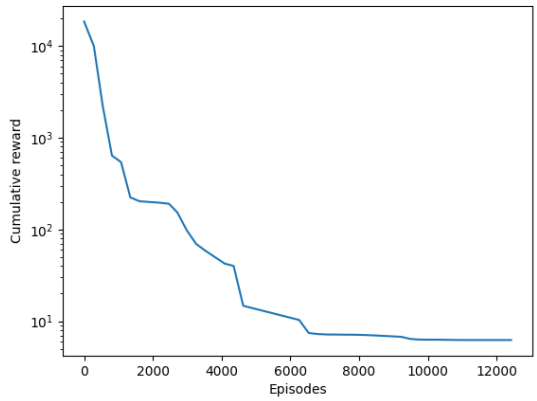

Figure 4.4: GP learning curve

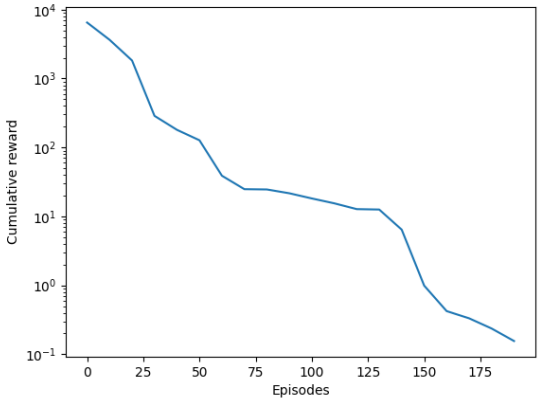

Figure 4.5: DDPG learning curve

Learning Curves Comparison

In the graph below, the cost functions of Bayesian Optimization (BO) and Genetic Programming (GP) are compared with the cumulative reward of Reinforcement Learning (RL). The RL curve is modified by changing its sign to be compared with the others approaches. Additionally, the reward R obtained from the PI scheduling controller in a single episode has also been added, as a reference.

The different rewards as a function of episodes are shown in Figure 4.6 following logarithmic graph on both axes.

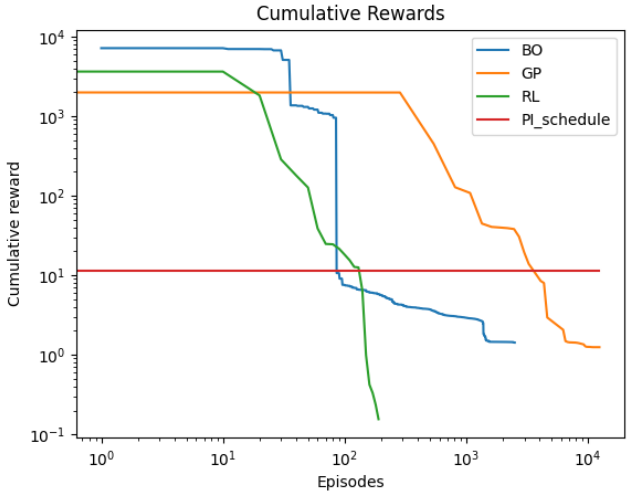

Figure 4.6: Comparison of the learning curves of the model-free controllers, with the cost function of the Baseline PI controller used as a reference (shown in red)

The minimum reward and number of training episodes are summarized in Table 4.1.

Controller	Number of training episodes	Reward Min $[(rad/s)^2]$
PI scheduling	—	11.26785
BO	2500	1.42754
GP	12420	1.24940
RL	200	0.15554

Table 4.1: Number of training episodes and the corresponding minimum reward obtained during the training of the model-free controllers

From Table 4.1, the effectiveness of RL can be observed both in terms of minimum reward and learning. Furthermore, BO achieves convergence with fewer iterations rather than GP, but GP achieves a slightly lower reward.

4.1.3 Controllers Benchmarking

To evaluate the performance of the different controllers, they were tested on five different wind profiles that covered the entire range of average values in control Region 2. Furthermore, the controllers were also tested on wind profiles that were different from those used during the training phase.

The characteristics of each wind profile are summarized in Table 4.2.

Wind	Mean Wind Speed [m/s]	Turbulence Intensity (TI) [%]
Wind1	12.5	4
Wind2	14	8
Wind3	16	9
Wind4	19	11
Wind5	22	12

Table 4.2: Set of the wind profiles and their characteristics used for testing the model-free controllers

The reward used to evaluate the performance is chosen as:

$$J = \sum_{t} (\omega - \omega_{rated})^2 \tag{4.3}$$

The obtained results are presented in Figure 4.7 and Table 4.3, represented in terms of $R = \frac{1}{J}$, so that in this way higher values indicate better performance.

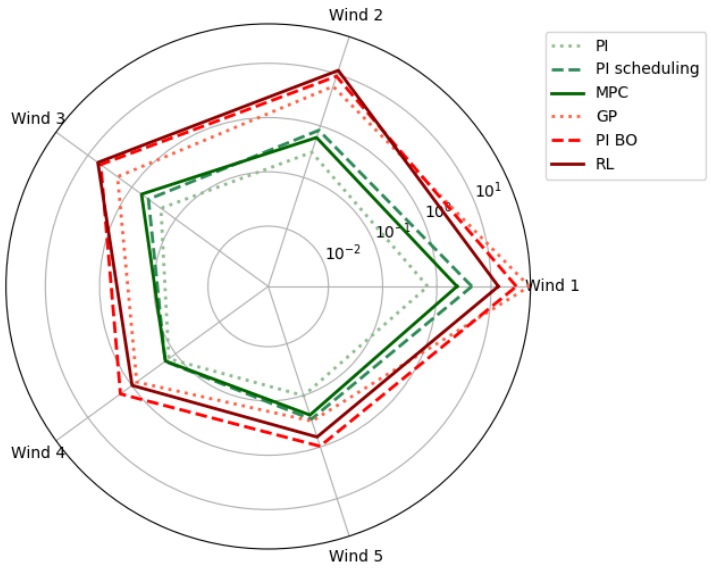

Figure 4.7: Performance of each controller for each testing wind speed presented in a radar diagram. Model-free controllers are depicted in red, while model-based controllers are shown in green.

In Figure 4.7, the model-based controllers are represented in green, while the

model-free controllers are represented in red. It is evident that there is a gap between the two families of controllers.

From these results, it is evident that data-driven controllers consistently outperform model-based controllers that rely on system modeling and linearization. In particular, the best performances are highlighted in bold in Table 4.3.

Controllers	Wind1	Wind2	Wind3	Wind4	Wind5
PI	0.67776	0.30713	0.21694	0.14326	0.10138
PI scheduling	4.42931	0.81805	0.41696	0.17343	0.28945
MPC	2.35280	0.59302	0.60023	0.17050	0.24005
PI BO	29.28858	9.10788	5.09868	**1.81539**	**0.96480**
GP	**50.73824**	5.81132	2.11630	0.77073	0.31587
RL	13.65467	**11.83278**	**5.93398**	0.98210	0.63939
Best Controller	GP	RL	RL	BO	BO

Table 4.3: Numerical results of the performance of each controller in terms of episode rewards for each testing wind speed

It can be inferred that one of the best controllers overall is the one based on RL. As observed in the comparison of the learning curves, RL was able to learn better than the others and in significantly fewer iterations. However, it should be noted that the PI optimized with BO controller also yields good results, even considering the fact that it does not take the wind speed as an input. One of the advantages of this PI controller is that it does not require scheduling and external wind speed feedback, making it well-suited for real-world applications where wind speed measurement is prone to significant errors and uncertainties.

4.2 Experimental Results

In this section, the numerical results obtained in this thesis are presented. The first part focuses on the power curve of the scaled model wind turbine. Following that, the experimental results of the developed controllers are presented. Finally, the results and comparison applying the controllers in the two-wind-turbine configuration are discussed.

4.2.1 Power Curve

The power curve of the model wind turbine is depicted in Figure 4.8.

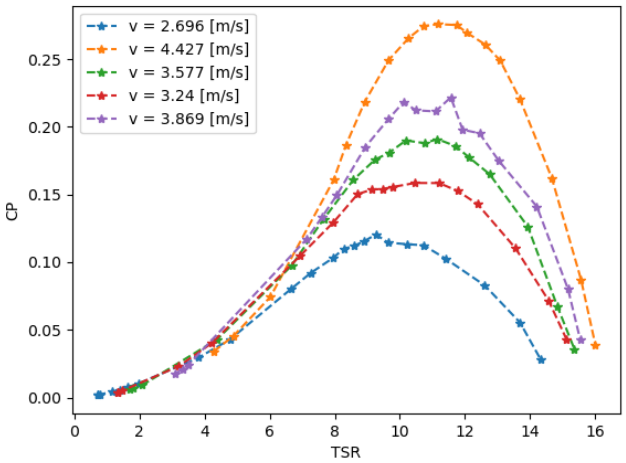

Figure 4.8: Experimental power curves of the wind turbine

It is immediately apparent that the power curves do not follow the theoretical expectations. In theory, these curves should collapse into a single curve for all wind speeds. This discrepancy could be due to different facts:

- the model blocking in the test chamber could be excessive. To overcome this issue, the free stream velocity could be corrected, as indicated in [29], using correction factors that take into account the configuration of the test section (open jet in this case) and the blockage factor B_F.
- the position of the Pitot tube, at the hub of the turbine, is not suitable. In fact, the Pitot tube is placed in the divergent zone of the flow tube that passes through the turbine rotor. As a result, the measured wind velocity will be lower than the actual free stream velocity. Referring to the actuator disc theory (1.4), it is as if one measure v_D instead of v. Due to the available

instrumentation, it was not possible to place a Pitot tube immediately after the convergent section to effectively measure the free stream velocity.

- another aspect to consider is that the wind turbine model is not rigid, and it could deform with increasing wind velocity, resulting in a change in the shape of the blade.

The small model of the turbine does not allow for pitch control, so it would be reasonable to focus on torque control (Region 1). The goal of this controller, as seen in previous chapters, is to always operate at the maximum power coefficient.

However, when plotting the power coefficient as a function of the applied action (Figure 4.9), it can be observed that the maximum power coefficient is consistently obtained at approximately the same action value. Therefore, using a different control law with different objectives would be better to observe the effect of control on the wind turbine.

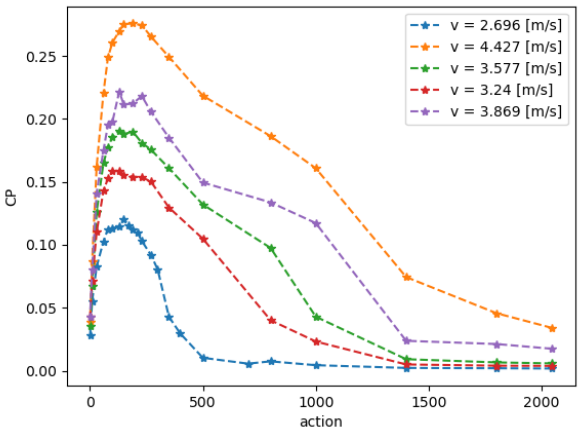

Figure 4.9: Power coefficient in function of the actions applied

A control law that would be more effective is simulating control in Region 2 using torque, which involves maintaining a constant rotational speed. The objective becomes identical to that of the numerical part, with the only difference being that control is achieved through generator torque. It should be noted that this inevitably does not coincide with the strict power control objective of Region 2, which is to maintain a constant power output at the rated value, due to the change in torque with wind speed. Nonetheless, it is useful for simulating control in Region 2 when only torque control is available.

Figure 4.10 illustrates the relationship between rotational speed and actions under steady-state conditions for two different wind speeds (approximately 2.5 m/s and 4.5 m/s), providing insights into defining control objectives.

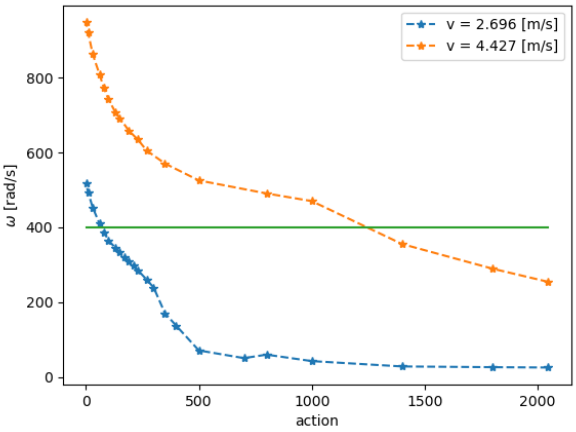

Figure 4.10: Rotational speed vs actions

An appropriate control objective would be to maintain the rotational speed at 400 rad/s, effectively simulating the maximum rated speed at this velocity. From the graph, it is evident that changing the wind speed has a significant impact on the required action value to sustain this rotational speed. Therefore, this enables a clearer observation of the controller's effect, highlighting its influence under different wind speed conditions.
It is important to note that the controller is designed to operate within a specific speed range, namely between 2.5 m/s and 4.5 m/s. Beyond this interval, there is no action value that can guarantee a steady-state speed of 400 rad/s.

4.2.2 PI Baseline

To test the baseline PI controller developed in (3.4.2), several experiments were conducted at a constant wind speed of approximately $v = 3.2m/s$, varying the values of ζ and ω_n.
In the first set of tests, the value of ζ was changed while keeping ω_n constant to observe the system's response. Subsequently, ω_n was varied while ζ remained fixed.

The controller was activated with an initial condition of rotational velocity around 600 rad/s, simulating a step change in the operating condition. The results of these tests are presented in Figure 4.11 and 4.12.

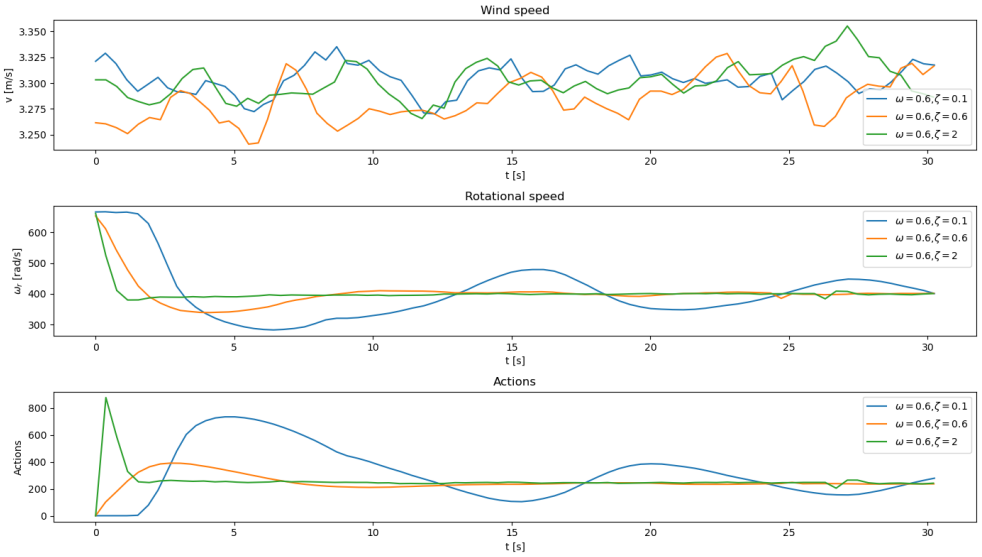

Figure 4.11: Response of the model wind turbine with the PI baseline controller at different values of ζ (with fixed ω_n)

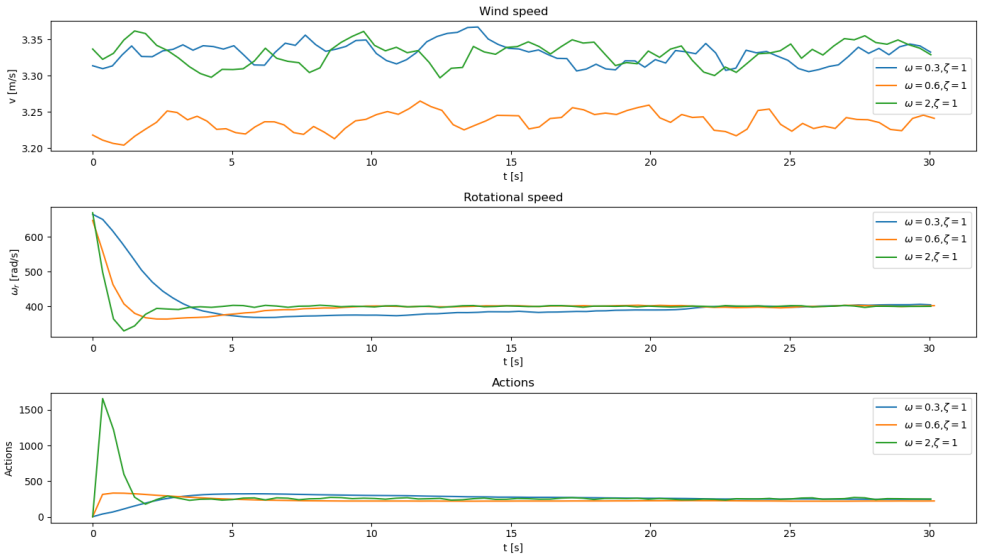

Figure 4.12: Response of the model wind turbine with the PI baseline controller at different values of ω_n (with fixed ζ)

As can be seen from Figure 4.11 and 4.12, the behavior aligns with the expectations from theory: for $\zeta < 1$, the system is underdamped, and the frequency response increases as the ω_n parameter grows. As done in (2.3.2), to tune the gains

of the baseline PI controller, the values of the two parameters chosen are:

$$\omega_n = 0.6$$

$$\zeta = 1$$

4.2.3 PI Schedule

A preliminary test case was conducted to evaluate the performance of the PI controller with scheduling. The wind speed in the wind tunnel was varied, and the controller was activated with an initial condition of rotational velocity at approximately 600 rad/s. To obtain a parameter measuring the effectiveness of the controller, a previous test was conducted on the turbine without the controller's action. This was done by maintaining a constant input of action=10 to the system. Results are shown in Figure 4.13.

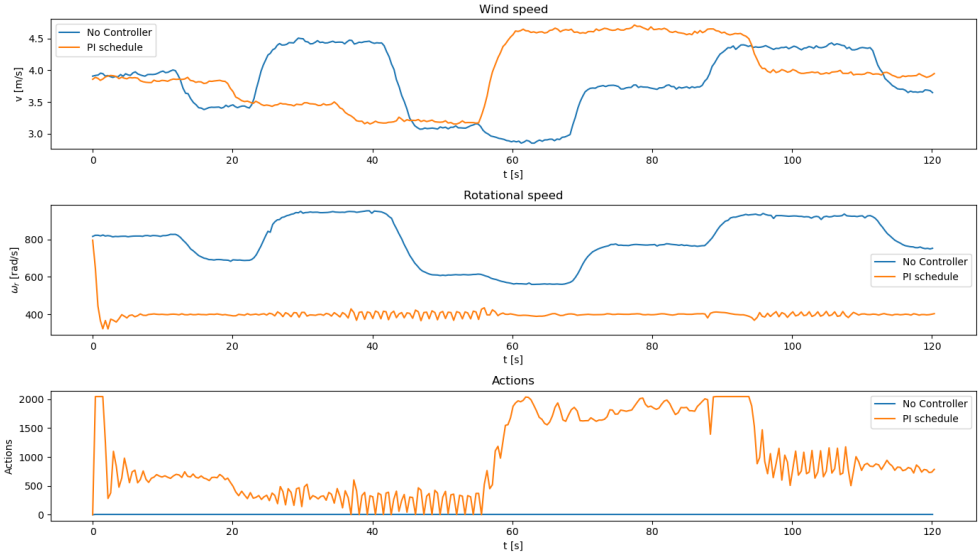

Figure 4.13: Test case of the experimental PI scheduling controller by varying the speed in the test section

Clearly, it is impossible to make direct comparisons between the case with the controller and the case without it due to the wind speed, which, as seen from the plot, cannot be varied exactly in the same way between the two tests. However, by comparing the overall trends of the two cases, differences can be observed. As can be seen from the graph, the controller has a significant effect on controlling the rotational velocity. The blue line represents the case without control, while the orange line represents the case with the controller. The controller is able to effectively respond to variations in the wind.

4.2.4 PI with BO

The results obtained with BO in terms of learning curve are presented in Figure 4.14.

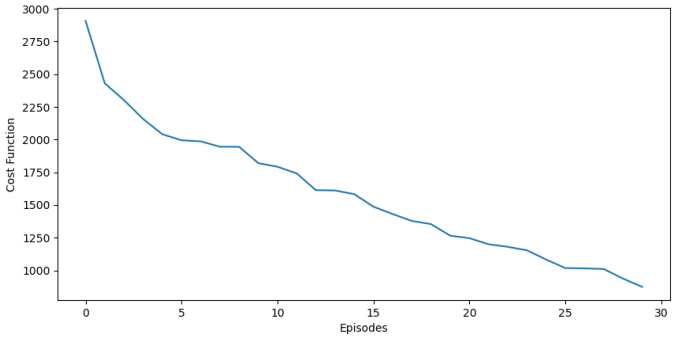

Figure 4.14: Experimental Learning Curve BO

Even though the training was stopped before achieving absolute convergence as it can be seen by the slope of the learning curve, it is evident that the method is improving. The value of the four optimal parameters are: $a = 1.39454$, $b = 4.62470$, $k_p = 13.62306$ and $k_i = 4.24846$. A similar test as the previous controller was conducted to evaluate the effectiveness of this controller (Figure 4.15).

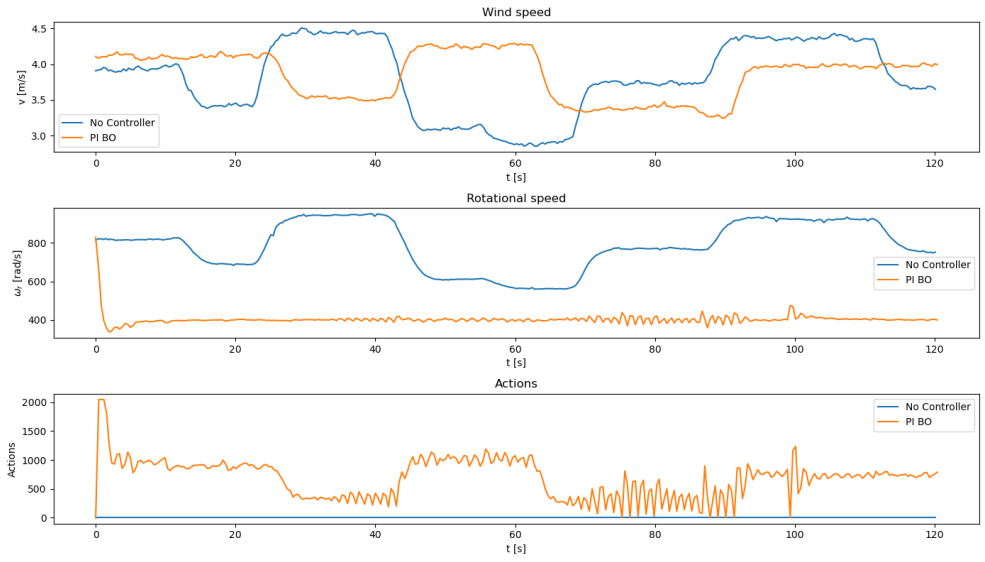

Figure 4.15: Test case of the experimental PI optimized with BO by varying the speed in the test section

4.2.5 Two Turbine Test Case

The introduction of an additional wind turbine model upstream of the controlled wind turbine generates a wake profile that impacts the second turbine under control. The results of this wake profile, measured using a Pitot tube, are presented in Figure 4.16.

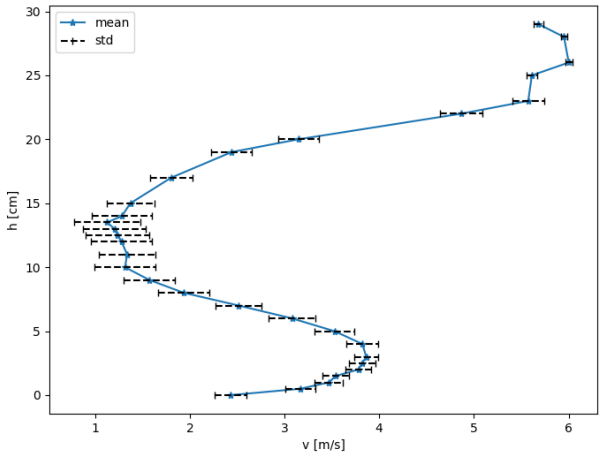

Figure 4.16: Wake wind speed profile that affects the controlled wind turbine

Figure 4.16 depicts the average values of the wind speed profile at the location of the controlled turbine, with each data point accompanied by its corresponding standard deviation, providing an indication of the turbulence level. A small portion of the boundary layer can be observed, while at the top, the strong wake caused by the first turbine is visible.

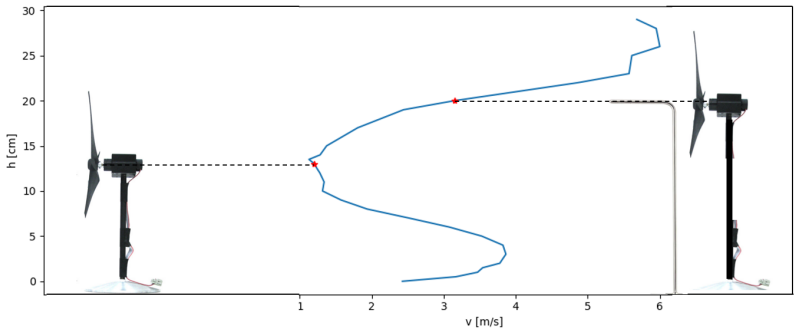

Figure 4.17: The two turbine models and the wake caused by the first turbine, which impacts the second

In Figure 4.17, one can observe how the inflow changes for the second turbine. The strong influence of the upstream turbine on the second turbine located 40 cm downstream can be observed. The first turbine has a height of 13 cm, while the second turbine has a height of 20 cm. The turbine to be controlled will be partially exposed to the wake and partially operating in the free stream. The hub of the turbine will experience an average velocity of 3 m/s. Furthermore, by examining the values of the standard deviation for each point in Figure 4.16, it can be observed that the highest turbulence occurs at the center of the wake.

In Figure 4.18, the difference in the wind signal sensed by the Pitot tube at the hub height of the second turbine (indicated in red on the graph) with and without the first turbine is observed, confirming that the presence of the upstream turbine causes a strong unsteady field.

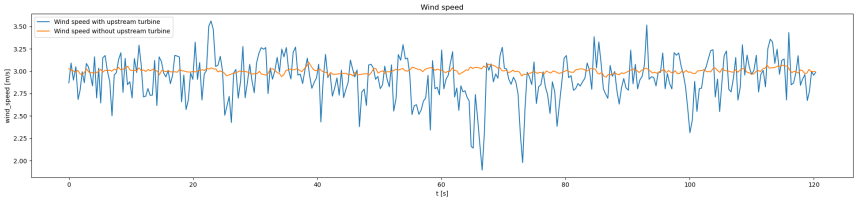

Figure 4.18: Wind speed sensed by the Pitot tube with and without the upstream turbine

In Figure 4.18, it can be observed that the field exhibits significant unsteadiness, with fluctuations exceeding 1 m/s. This indicates that this configuration is suitable for testing and comparing different controllers under these wind conditions. Although the wind profile may vary in each test, the turbulent nature enables statistical comparisons. To obtain meaningful statistical results, the different controllers were tested over a sufficiently long period of $T = 120$ s. Additionally, to ensure statistical reliability, four separate tests were conducted for each type of controller.

As observed from the results of sections (4.2.3) and (4.2.4), the controllers that takes the wind speed as input exhibits oscillatory behavior. This, in contrast to the results obtained in (4.2.2), could be attributed to the controller's strong sensitivity to small variations in wind. For this reason, before passing the wind signal to the controller, it was filtered as indicated in (2.2.2). The cutoff frequency of the filter was chosen based on the time constant obtained in (3.4.1):

$$f_{c.o.} = \frac{1}{\tau_t} = 1.054Hz$$

In Figure 4.19, the filtered wind speed can be observed.

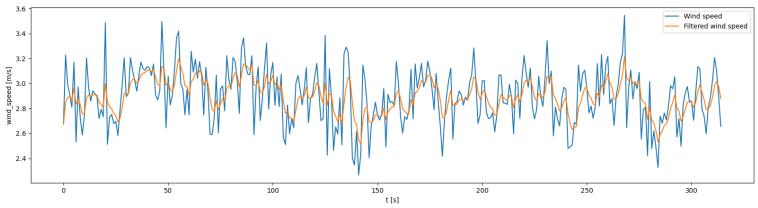

Figure 4.19: Filtered wind speed sensed by the Pitot tube

Finally, the performance of the controllers was evaluated in the presence of the upstream turbine. Figure 4.20 illustrates the results for three scenarios: the case without a controller in blue, the case with a PI controller with scheduling in orange, and the case with a PI controller optimized using BO in green. For the sake of clarity, filtered wind speed is presented in this figure.

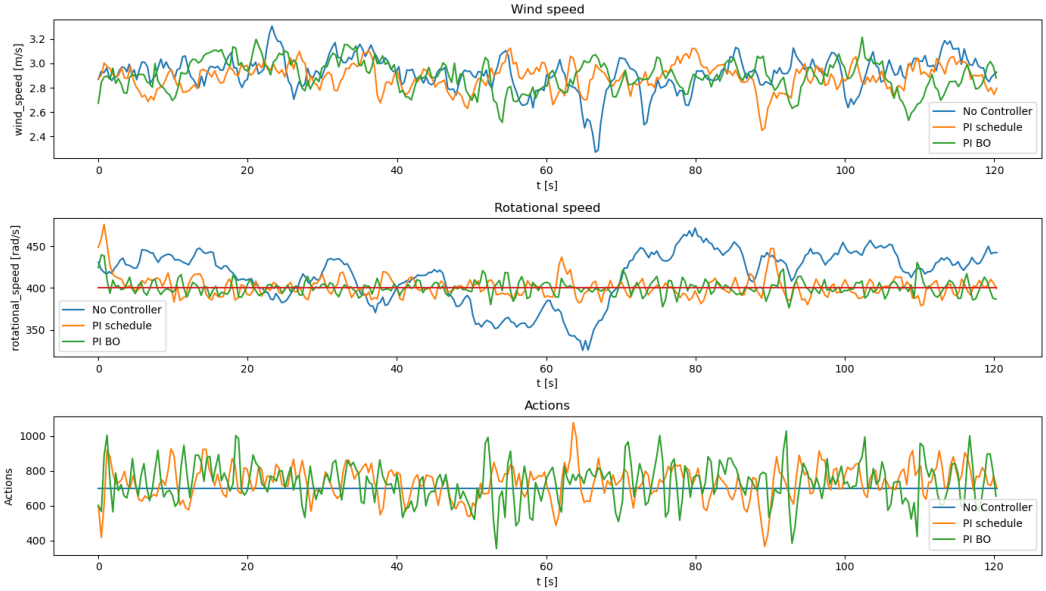

Figure 4.20: Test case with two wind turbines: the case without a controller depicted in blue, the case with a PI controller with scheduling shown in orange, and the case with a PI controller optimized using BO represented in green

It can be observed that both the PI controllers effectively mitigates the fluctuations caused by the upstream turbine, leading to a smoother and more stable response.

Although not highly accurate, the performance of the two controllers can be compared by evaluating the following cost function over episodes of T=120s with

the following cost function:

$$J = \sum_{t} (\omega - 400 rad/s)^2 \tag{4.4}$$

The results are summarised in Table 4.4, with the values dimensionless compared to the cost function obtained without a controller.

Controllers	Test 1	Test 2	Test 3	Test 4
No control	1	–	–	–
PI scheduling	0.11302	0.10482	0.09588	0.14049
PI BO	**0.07349**	**0.07132**	**0.05889**	**0.05982**

Table 4.4: Experimental results of the performance of each controller in terms of dimensionless episode rewards for each testing wind speed

This latest test case and the results from various tests confirm the trend observed in the numerical section. In this case, the controller optimized with BO outperforms the model-based controller obtained through experimental modeling of the wind turbine.

Chapter 5

Conclusions

The main objective of this thesis was to implement and compare different controllers for wind turbines, with a primary focus on operating within Region 2. In this operational Region, the goal was to maintain the power output at the rated value by controlling the rotational speed through the pitch angle of the blades while ensuring a constant generator torque. The model-based controllers developed in this thesis include classical controllers commonly employed in the state-of-the-art, such as Proportional-Integral (PI) and Model Predictive Control (MPC). On the other hand, the model-free controllers developed are based on machine learning techniques, including Bayesian Optimization (BO), Gaussian Processes (GP), and Reinforcement Learning (RL). The comparison primarily aimed to evaluate the performance of model-based and model-free controller families, using both numerical simulations and experimental validations.

The numerical study results showed that, in general, model-free controllers outperformed model-based controllers. Within the model-free controller family, certain approaches demonstrated superior training efficiency and performance. Specifically, RL showed the best overall performance and required fewer training episodes, although it necessitated training more parameters compared to BO and GP. However, BO showed excellent results, closely matching the performance of RL. Notably, BO has an advantage over RL as it doesn't rely on wind velocity feedback, which is prone to noise in current measurement techniques. This suggests that a simple PI controller may be sufficient for certain applications that prioritize good performance rather than achieving the absolute best outcome. On the other hand, GP exhibited lower sample efficiency, requiring a larger number of system interactions due to its quasi-random search characteristics.

In contrast to model-based controllers, data-driven controllers effectively capture the dynamic behavior of wind turbines by utilizing input-output data, rather than relying on simplified models. The model-based approach may not always account for minor details or deviations from the assumed model, while data-driven methods

offer flexibility to adapt to such variations. This thesis addressed this aspect by intentionally neglecting pitch actuator dynamics in the model-based controller, recognizing the challenges of accurately modeling every aspect due to uncertainties and real-world complexities. By incorporating these deviations into the simulator, the thesis aimed to simulate the challenges of accurately modeling the system, mirroring real-world scenarios where achieving a perfect model may be impossible.

In conclusion, the experimental part of the study confirmed the findings presented in the numerical section. The implementation techniques proposed in the numerical analysis were validated through experimental results. The performance evaluation of the baseline PI controller and the PI controller with BO optimization in a realistic environment, simulating wake interactions caused by an upstream turbine in a wind farm, indicates the relevance of these implementations in practical scenarios. Furthermore, the statistical analysis demonstrates that the performance of the BO-based controller is superior to that of the model-based controller. Despite the model-based controller relying on a model derived from experimental data, the model-free controller outperformed it by effectively managing the interactions between the controller and the wind turbine.

Chapter 6

Future Works

This thesis provides valuable insights and lays the groundwork for promising future research directions. Several potential improvements and areas for future studies can be identified:

- Firstly, in the numerical part, the model used is a highly simplified representation of wind turbine dynamics. Future studies can focus on enhancing the simulation model by incorporating approaches such as multibody dynamics, which consider the deformation of various turbine components, particularly the blades and the tower. These deformations give rise to an aeroelastic coupling between the structure and the external wind. Subsequently, it is necessary to validate the different controllers tested in this study with these more advanced models.

- The formulation of the MPC could be extended to a continuous and nonlinear model. This difference in implementation, as mentioned, could lead to an improvement in the performance of this controller.

- The model-free controllers were implemented using Python libraries, with basic parameters left unchanged. A more in-depth investigation could involve adapting the different parameters based on the problem and assessing whether there are any counter-trends in the learning performance compared to the baseline case.

- From an experimental perspective, numerous improvements can be made. Firstly, it is possible to attempt a calibration model between the load applied to the wind turbine using resistors and the actual mechanical torque applied by the generator to the turbine rotor. This enables a comparison of the accuracy of a first-order model (as utilized in the numerical section) in modeling the behavior of a small wind turbine. From here, the development of various

controllers can be expanded, allowing for a direct comparison. Furthermore, by implementing a system to change the pitch angle, it would be possible to test the numerical controls in a manner that ensures correspondence.

- Experimental testing of all the model-free controllers could be conducted to determine if they indeed reflect the trends observed in the numerical section, further validating their performance in real-world scenarios.

Appendix A

Numerical Methods Employed

The numerical scheme used to solve the control problem can be summarized by the following figure:

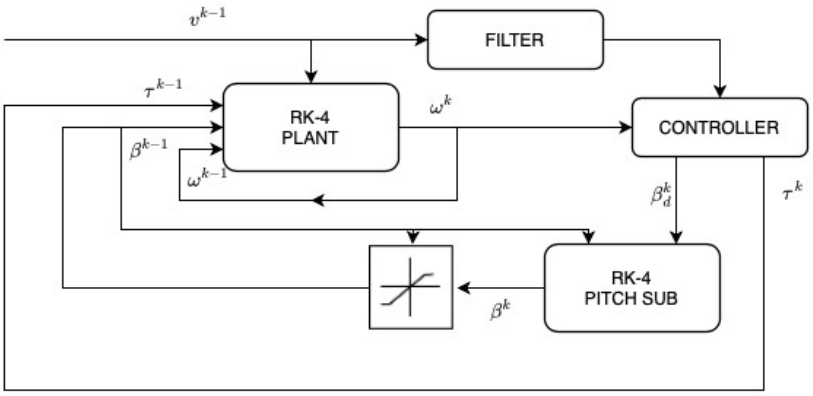

Figure A.1: Numerical scheme

The scheme highlights the numerical evolution from one iteration $k-1$ to the subsequent iteration k. The essential components of the controller are the turbine model and the pitch subsystem model, which interact with the controller at each iteration. To solve the ODEs of the turbine model and the pitch subsystem model, a fourth-order Runge-Kutta method has been employed:

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= hf(t_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

where h is the step size, t_n represents the current time, y_n is the current approximation of the solution, and $f(t_n, y_n)$ is the derivative function evaluated at (t_n, y_n).

All the developed and utilized codes in this thesis can be found at the following link:
https://github.com/SebastianoRandino/SebastianoRandinoMasterThesisCodes.git

Appendix B

Pressure Transducer Calibration

The measurements of total and static pressure difference, obtained from the Pitot tube to calculate the wind velocity, were derived from a Validyne membrane differential pressure transducer. The pressure transducer is associated with a Validyne demodulator, which allows for adjustments of the zero and slope of the calibration curve:

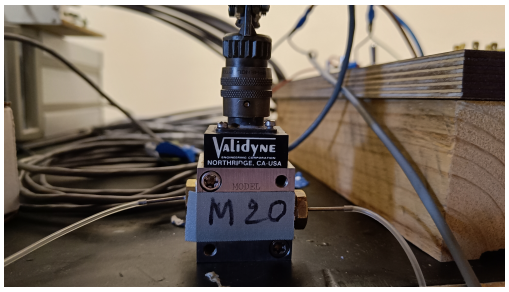

Figure B.1: M20 Validyne

Figure B.2: Validyne Demodulator

The membrane used (M20) can withstand up to double the pressure compared to 860 Pa. The calibration process involved adjusting the values of the demodulator to align the zero point when the pressure difference is zero, and maximizing the potential difference for the maximum expected pressure reading in the test chamber ($v \approx 10[m/s]$):

$$\Delta p = p^0 - p = \frac{1}{2}\rho v_{inf}^2 \approx 61Pa$$

This was done to maximize the sensitivity of the pressure readings as much as possible. The calibration was performed using a Betz water manometer, knowing

the values of the applied pressure difference to the pressure sensor:

Figure B.3: Betz manometer

Indeed, one output of the transducer was connected in common with the pressure gauge, and the other two outputs, both from the Betz gauge and the transducer, were left open to ambient pressure.
The data was acquired for:

$$t_s = 1s$$

$$f_s = 1000Hz$$

The obtained calibration curve is as follows:

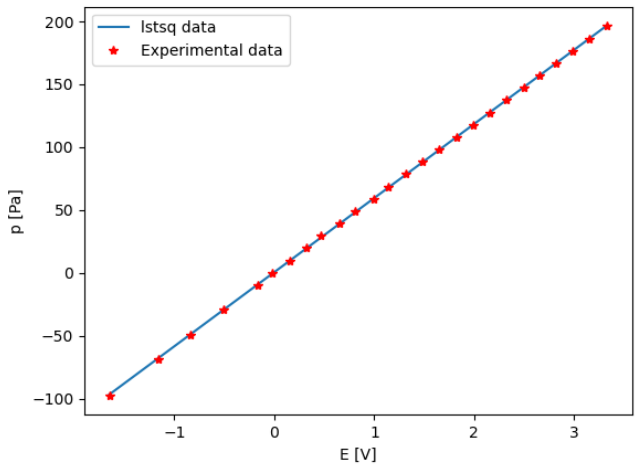

Figure B.4: Pressure transducer calibration curve

$$p = k_p E + b = 58.966E + 0.561 \tag{B.1}$$

with an R2 parameter of:

$$R^2 = 0.9999433$$

indicating that the calibration is considered a very good accuracy.

Bibliography

[1] F.D. Bianchi, H. de Battista, and R.J. Mantz. *Wind Turbine Control Systems: Principles, Modelling and Gain Scheduling Design*. Advances in Industrial Control. Springer London, 2010. ISBN: 9781849966115. URL: https://books.google.it/books?id=46SicQAACAAJ (cit. on pp. 1–3, 17).

[2] Lucy Y Pao and Kathryn E Johnson. «Control of wind turbines». In: *IEEE Control systems magazine* 31.2 (2011), pp. 44–62 (cit. on pp. 2, 14).

[3] Tony Burton, Nick Jenkins, David Sharpe, and Ervin Bossanyi. *Wind energy handbook*. John Wiley & Sons, 2011 (cit. on p. 3).

[4] Nikhar J Abbas, Daniel S Zalkind, Lucy Pao, and Alan Wright. «A reference open-source controller for fixed and floating offshore wind turbines». In: *Wind Energy Science* 7.1 (2022), pp. 53–73 (cit. on pp. 4, 14, 27, 75).

[5] Jason Jonkman, Sandy Butterfield, Walter Musial, and George Scott. *Definition of a 5-MW reference wind turbine for offshore system development*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009 (cit. on pp. 5, 20).

[6] Tony El Tawil. *Van Der Hoven spectrum*. URL: %7Bhttps://www.researchgate.net/figure/Van-Der-Hoven-spectrum-13_fig15_323543174/actions#reference%7D (cit. on p. 7).

[7] Fabio Pino, Lorenzo Schena, Jean Rabault, and Miguel A Mendez. «Comparative analysis of machine learning methods for active flow control». In: *Journal of Fluid Mechanics* 958 (2023), A39 (cit. on p. 10).

[8] Lorenzo Schena, Emmanuel Gillyns, Wim Munters, Sophia Buckingham, and Miguel Alfonso Mendez. «Wind Turbine Control using Machine Learning techniques». In: *ECCOMAS Congress 2022-8th European Congress on Computational Methods in Applied Sciences and Engineering*. 2022 (cit. on pp. 10, 58).

[9] Emmanuel Gillyns, Sophia Buckingham, and Grégoire Winckelmans. «Implementation and Validation of an Algebraic Wall Model for LES in Nek5000». In: *Flow, Turbulence and Combustion* 109.4 (2022), pp. 1111–1131 (cit. on p. 10).

[10] Allan D Wright and LJ Fingersh. *Advanced control design for wind turbines; Part I: control design, implementation, and initial tests*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2008 (cit. on p. 13).

[11] Katherine Dykes, S Andrew Ning, George Scott, and Peter Graf. *WISDEM®(Wind Plant Integrated System Design and Engineering Model)*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2021 (cit. on p. 16).

[12] Bonnie J Jonkman. *TurbSim user's guide*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2006 (cit. on p. 18).

[13] Morten H Hansen, Anca Hansen, Torben J Larsen, Stig Øye, Poul Sørensen, and Peter Fuglsang. «Control design for a pitch-regulated, variable speed wind turbine». In: (2005) (cit. on p. 24).

[14] Wai Hou Lio, JA Rossiter, and Bryn L Jones. «A review on applications of model predictive control to wind turbines». In: *2014 Ukacc international conference on control (control)*. IEEE. 2014, pp. 673–678 (cit. on p. 32).

[15] Michael Fink. «Implementation of Linear Model Predictive Control–Tutorial». In: *arXiv preprint arXiv:2109.11986* (2021) (cit. on pp. 34, 35).

[16] Midhun T Augustine. «MODEL PREDICTIVE CONTROL USING MATLAB». In: (2021) (cit. on p. 35).

[17] Joachin Dahl and Lieven Vandenberghe. «Cvxopt: A python package for convex optimization». In: *Proc. eur. conf. op. res.* Vol. 2. 2006, p. 3 (cit. on p. 37).

[18] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. «Taking the human out of the loop: A review of Bayesian optimization». In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175 (cit. on pp. 38, 39).

[19] Stanislav Markov. «Skopt documentation». In: (2017) (cit. on p. 40).

[20] Samuel Ahizi, Pedro Marques, Jan Van den Berghe, Lorenzo Schena, Fabio Pino, Matilde Fiore, Romain Poletti, and Miguel Mendez. «Hands-on Machine Learning for Fluid Dynamics, Second Edition». In: Jan. 2023 (cit. on pp. 41–46, 52, 53).

[21] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. «DEAP: Evolutionary Algorithms Made Easy». In: *Journal of Machine Learning Research* 13 (July 2012), pp. 2171–2175 (cit. on p. 47).

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 50).

[23] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. «Continuous control with deep reinforcement learning». In: *arXiv preprint arXiv:1509.02971* (2015) (cit. on p. 54).

[24] TensorFlow Developers. «TensorFlow». In: *Zenodo* (2022) (cit. on p. 54).

[25] *Van Der Hoven spectrum*. URL: %7Bhttps://www.vki.ac.be/index.php/facilities-other-menu-148/low-speed-wind-tunnels/48-research-and-consulting/facilities/low-speed-wind-tunnels/60-low-speed-wind-tunnel-l-2b%7D (cit. on p. 59).

[26] Nicolas Coudou, Maud Moens, Yves Marichal, Jeroen Van Beeck, Laurent Bricteux, and Philippe Chatelain. «Development of wake meandering detection algorithms and their application to large eddy simulations of an isolated wind turbine and a wind farm». In: *Journal of physics: Conference series*. Vol. 1037. 7. IOP Publishing. 2018, p. 072024 (cit. on pp. 59, 60).

[27] *NI-DAQmx Python Documentation*. https://nidaqmx-python.readthedocs.io (cit. on p. 61).

[28] Chris Liechti. «PySerial documentation». In: *Versión: 2.6, Diciembre 2011* (2016) (cit. on p. 62).

[29] Abdelgalil Eltayesh, Magdy Bassily Hanna, Francesco Castellani, AS Huzayyin, Hesham M El-Batsh, Massimiliano Burlando, and Matteo Becchetti. «Effect of wind tunnel blockage on the performance of a horizontal axis wind turbine with different blade number». In: *Energies* 12.10 (2019), p. 1988 (cit. on p. 83).