Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de
Ingeniería Informática

Universidad
Complutense de Madrid

Facultad de
Informática

Optimal Conditions Study for a HIL Application for Control and Real-Time Simulation of a Wind Turbine theoretical Model via OPC-UA

Nolan Russ Asensio

Director/a: Matilde Santos Peñas
Co-director/a: Jesús Enrique Sierra García

Master's Thesis

Máster Universitario
en Ingeniería de Sistemas y Control

September 2024

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Nolan Russ Asensio

Firma del alumno

Agradecimientos

Me gustaría dedicar este espacio especialmente a mi padre y mi madre por apoyarme en todas mis aspiraciones académicas y de animarme a seguir superándome. Esta tesis es el resultado de los valores que me habéis inculcado y por ello y por mucho más siempre os estaré eternamente agradecido.

Resumen

Este trabajo fin de máster plantea como objetivo principal la implementación de una aplicación *Hardware In-the-Loop* para el control y la simulación en tiempo real de un modelo teórico de una turbina eólica de pequeña potencia a través de OPC-UA. Ello conllevó un estudio y entendimiento del modelo matemático del que parte el modelo de simulación y del que, tras una evaluación inicial heurística, se identificó la necesidad de resolver un problema relacionado con la configuración de las condiciones iniciales del sistema.

El problema requirió de una resolución simbólica y matemática de las ecuaciones del sistema, así como la definción y resolución del problema de optimización para el cual habían de cumplirse ciertas condiciones iniciales relacionadas con las derivadas de unas variables de la turbina.

Adicionalmente, la aplicación *Hardware In-the-Loop* hace uso de un controlador PID para controlar la salida de potencia eléctrica de la turbina eólica mediante la regulación del actuador de la pala. La aplicación incluye el desarrollo de una interfaz gráfica desde la cual se busca facilitar la interacción con el modelo para su uso por la comunidad académica en relación con energía eólica.

Palabras clave: HIL (Hardware-In-the-Loop), OPC-UA, modelo matemático, turbina eólica, condiciones iniciales, simulación, optimización

Abstract

This Master's thesis sets as an initial objective the implementation of a hardware in the loop application for control and simulation in real-time of a wind turbine theoretical model via OPC-UA. This required a thorough understanding and study of the mathematical model, which the simulation model is based and where an initial heuristic evaluation identified the need to solve a problem related to the setting of the initial conditions of certain variables of the turbine.

The problem required a symbolic and mathematical resolution as well as the definition and resolution of the optimization problem for which certain initial conditions or constraints, particularly related to the derivatives of certain variables, were met.

On a second note, the hardware in the loop application makes use of a PID controller to control the electrical power output of the wind turbine by regulating the angle of the blade pitch actuator. The application includes the development of a screen interface from which it is able to easily interact with the model and facilitate its use in the academic community related to wind energy.

Keywords: HIL (Hardware-In-the-Loop), OPC-UA, wind-turbine, mathematical model, initial conditions, simulation, optimization

Glossary

ANFIS Adaptive Neuro Fuzzy Inference System.

DC Direct Current.

GPU Graphics Processor Unit.

HIL Hardware-In-the-Loop.

HMI Human-Machine Interface.

IP Internet Protocol.

OPC-UA Open Platform Communication - Unified Architecture.

PID Proportional-Integral-Derivative.

PIL Processor-In-the-Loop.

PLC Programmable Logic Controller.

RAM Random Access Memory.

TIA Tottally Integrated Automation.

WECS Wind Energy Conversion System.

WT Wind-Turbine.

WTG Wind-Turbine Generator.

Contents

Glossary **xi**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objectives . . . 1
1.3 Document structure . . . 2
1.4 Workflow . . . 3

2 Fundamentals **5**

3 Materials and methods **9**
3.1 Simulink model . . . 9
3.2 Programmable Logic Controller (PLC) . . . 11
3.3 OPC-UA . . . 11
3.4 PID controller . . . 12

4 Definition of the initial conditions problem **15**
4.1 Model behaviour . . . 16
4.1.1 Real-Time Simulation . . . 17
4.1.2 Initial conditions mismatch . . . 18
4.1.3 Optimal conditions . . . 19
4.2 Mathematical description of the problem . . . 21

5 Optimal initial conditions for model simulation **23**
5.1 Initial parameters for simulation . . . 23
5.1.1 Solve for zero . . . 24
5.1.2 Optimization problem . . . 25
5.2 Sampling time . . . 27
5.2.1 Real-Time desktop solver . . . 28

6 Implementation of the HIL controller on OPC-UA **31**
6.1 Hardware-In-the-Loop (HIL) . . . 31

6.2 PID tuning 33

7 Results **39**

8 Conclusions and future works **45**

Bibliography and references **47**

A Read_OPC_Func **51**

B PLC program **53**

C Network Architecure **57**

D Simulink Model **59**

List of Figures

1.1 Workflow 4

2.1 Rotor aerodynamics model proposed by Gonzalez et al. [2011] 6
2.2 General scheme of the closed-loop control of a WT 6
2.3 Operation Regimes of a wind-turbine (Dirscherl et al. [2015]) 7
2.4 HIL Application overview 8

3.1 Step Response characteristics (Ledeneva et al. [2008]) 13

4.1 Initial Case 16
4.2 Wind model response for a nominal wind speed step 17
4.3 Model Simulation for $v_{ini} = 3m/s$ and ramp-up to $12m/s$. 18
4.4 Model Simulation at $v = 2m/s$ for $I_i = 0.001A$ and $\omega_i = 5.7rad/s$. 20

5.1 Responses for $\omega_i = 3.5rad/s$ and $I_{DC} = 2.6A$ 25
5.2 Responses for C_p, ω, $\frac{d\omega}{dt}$ and I_{DC} during the singularity event at wind speeds under cut-in-speed. 27
5.3 Simulations for different solver configurations 30

6.1 Block diagram of the HIL application 31
6.2 Real-Time open-loop simulation with θ sent through OPC-UA 33
6.3 State Machine Flow Diagram 34
6.4 System response for Case B.2 36
6.5 In red, PID disabled; in green, Case B.1; in blue, Case B.2 37
6.6 Consolidated PID tuning solution 37

7.1 HMI WTG States (I) 40
7.2 HMI WTG States (II) 40
7.3 HMI WTG states (III) 43
7.4 HMI WTG states (IV) 44
7.5 Simulation of the HIL WT control 44

C.1 Network Architecture Setup 58

D.1 Simulink Model 59

List of Tables

2.1 Theoretical model mechanical and electrical parameters 8

3.1 coefficients used by the model to obtain C_p 10
3.2 PID desired response 14

4.1 Classification of variables 21

5.1 Simulated Cases 27
5.2 Initial parameters configuration for the model simulation 27
5.3 Real-Time simulations for different sampling times 29

6.1 IP addresses of the systems that compose the solution 32
6.2 PID Tuning A 35
6.3 PID Tuning B 35

Chapter 1

Introduction

1.1 Motivation

Wind Energy has proven during the last 20 years to be an interesting and profitable source of renewable energy that demands constant innovation and a lot of maintenance efforts (Walford [2006]). The need of the industry for qualified and specialized technicians is rising and the interest from younger generations to be involved in this technology is also trending positively.

Currently, there are scarcely any tools and applications in the academic community that allow students and future technicians of the wind energy industry to familiarize themselves with control strategies and operational states of wind turbines. There are, however, many models and simulations for academic and professional use to study the behaviour of wind turbine systems. They require a good understanding of specific software and mathematical models to use them properly as well as a thorough understanding of model analysis and control systems in order to derive well thought out conclusions or results.

Due to this, the master's thesis at hand aims to tackle the complexity of wind turbine modelling and simulation in a more simplistic manner. It intends to simplify its ease of use by means of an application for future works in the academic community, such as open-loop simulations, and controller design and validation for closed-loop controls. Additionally, and due to its simplicity, it may also be used for classes, training and demonstrations in renewable energy. To this end, it defines the problems and difficulties related to real-time simulation as well as its requirements, and gives a method to overcome said difficulties.

1.2 Objectives

The main objective of this master's thesis is to provide a fully functional wind turbine monitorization and operation application that runs on one terminal, allowing for all components to be simulated to facilitate implementation. This application will include a Human-Machine

Interface focused on easy use and access. The interface will be designed in such a way for it to be intuitive as the target audience is considered to possess little to no understanding in wind turbine systems and their functionality. The main goal of the application is to enable students from renewable energy related studies to familiarize themselves with control operations and operational states of wind turbines.

For this, the master's thesis requires a study of typical modelling techniques and applications of wind turbines for simulation purposes. It also requires the know-how of implementing a hybrid solution where the simulation model and the Human-Machine Interface may interact with one another. Additionally, it will be necessary to implement a simple blade pitch controller solution that allows control and management of the electrical power output of the wind turbine.

In detail, the defined milestones for the master's thesis are as follows:

- Study of wind turbine mathematical modelling techniques
- Study of the wind turbine model behaviour and operational constraints
- Resolution of the wind turbine model for its use in Real-Time simulation
- Identification of the initial conditions for optimal simulation conditions to elude singularities and errors in the simulation of the model
- Development and tuning of a Real-Time controller for the wind turbine simulated model.
- Implementation of a communication protocol between controller and the wind turbine simulation model.
- Development and validation of a Human-Machine Interface to allow ease of access and use of the application.

1.3 Document structure

The following document is divided into chapters, where some of the developed solutions that are referenced throughout the document have been added in the Annex to facilitate reading.

Chapter 2 serves as a contextualization of the different technologies involved in the solution as a whole, giving a brief explanation of wind-turbine power generation and modelling. It goes into more detail as how wind-turbines may control power output and gives a brief description of the advantages of Hardware-In-the-Loop applications.

Chapter 3 dives into the different materials and methodologies studied and implemented to achieve the objectives previously defined. In particular, it covers the wind turbine Simulink

model and the Programmable Logic Controller programming and implementation via OPC-UA protocol with the Simulink model. It also justifies the type of controller chosen and provides a better understanding of the different parameters needed for tuning said controller.

Chapter 4 focuses on the definition of the problems faced with the Simulink model in order to apply it in a Real-Time Desktop Application. It attempts to develop an understanding of the Simulink model's behaviour and serves as the preamble for the following chapter.

Chapter 5 engages in identifying the need of defining initial parameters and start-up conditions for the real-time simulation. It goes into detail of the different methods that were explored for solving the mathematical problem defined in the previous chapter.

Chapter 6 proceeds to summarize of the Hardware-In-the-Loop solution development and implementation. This chapter includes the OPC-UA integration between the Programmable Logic Controller and the Simulink Real-Time Desktop Application as well as the definition of the state machine for the Human-Machine Interface application. It also includes the implementation and tuning of the PID controller.

Chapter 7 showcases different operational points of the consolidated solution by means of different sets of electrical power references input from the Human Machine Interface, evidencing the expected behaviour of the wind turbine within the same application.

Chapter 8 outlines the conclusions from the previous chapters and offers different points for discussion, identifying the unresolved topics and possible improvements for future efforts.

The document ends with the bibliography.

1.4 Workflow

Prior to application development, it was necessary to understand the behaviour of the model. For this, some pre-testing was conducted to understand the feasibility of the simulation model of study in a real-time desktop simulation. The workflow used is ilustrated as in Figure 1.1, where the first steps were to define the objectives to be accomplished and have a working understanding of the base model that would be the case of study. To help in the study of the model and to determine its limitations for this particular application, some pre-testing simulation was conducted, in real-time. This allowed to actually identify that the model was not able to run in real-time under certain initial conditions. As the simulation was not successful, it was then required to not only identify the problem but also describe the problem mathematically. These steps would then allow for defining and resolving an optimization problem whose method could be used to resolve similar cases identified in other models. The optimal solution found would then be retested in real-time to validate its use in the application.

Only then did the implementation of the communication protocol and the actual development of the HIL application begin. This phase included testing the OPC-UA communication between the controller and the Simulink model, the programming of the PLC with

the cotnroller strategy and state machine, the design and development of the HMI and the fine-tuning of a PID controller to ensure a stable response. Lastly, the application was fully validated by running the whole system in the same terminal for 15 minutes and implementing different operating scenarios to test the behaviour of the model under different conditions.

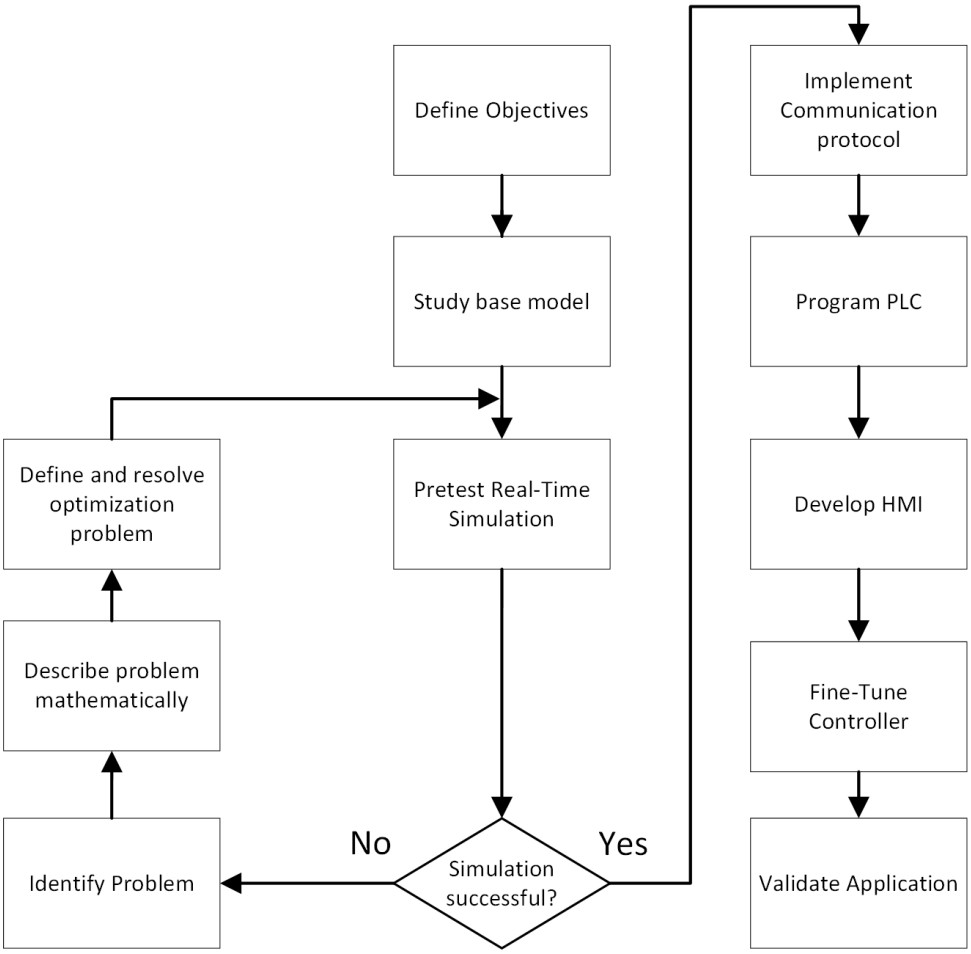


Figure 1.1: Workflow

Chapter 2

Fundamentals

Wind Energy Conversion Systems (WECS) mainly rely on the horizontal wind forces that come in contact with the blade surfaces and, as such, the associated wind power is heavily impacted by its variations. This is supported by Equation 2.1 that serves as the definition of wind power by area unit, where A is the theoretical surface of the blades in contact with the wind forces in m^2, ρ is the air density in kg/m^3 and v is the wind speed in m/s.

$$P = A\frac{1}{2}\rho v^3 \tag{2.1}$$

In variable-speed wind-turbine generator models the drive train may be modelled as a rotor and a generator inertia separated by a torsional spring (Burton et al. [2001]). That is to say, the basic structure of a wind-turbine may be comprised of a rotor for energy extraction, a gearbox, and the electrical generator (see Figure 2.1). The mechanical torque T_m generated by the wind forces is expressed in Nm and is defined by Equation 2.2 (Ackermann [2005]). R represents the radius of the theoretical circular area drawn by the rotation of the blades, λ constitutes the blade tip-speed ratio and C_p is the ratio of power extracted by the turbine when comparing to the wind power (Equation 2.1). This coefficient shall not exceed $C_p = 0.593$ as defined by Betz's law (Pizaña and H. [2015]).

$$T_m = \frac{1}{2}\rho\pi R^3 v^2 \frac{C_p}{\lambda}, \quad \text{where} \quad \lambda = \frac{R\omega}{v} \tag{2.2}$$

Electrical power output may then be controlled by means of the blade pitch angle represented by β or θ as it directly affects the blade's aerodynamics and, in essence, the blade's surface of contact with the horizontal wind forces. The model may then be coupled with a controller to ensure different controls objectives, such as, optimal power extraction, overall system stability to wind speed disturbances and even minimizing tower and nacelle oscillations due to wind forces in the x and y axis (Reisch et al. [2017]). This is known as the blade pitch controller.

The general controller scheme is shown in Figure 2.2, where the process is made out of the Turbine Aerodynamics and generator and the Blade Pitch control is comprised of the

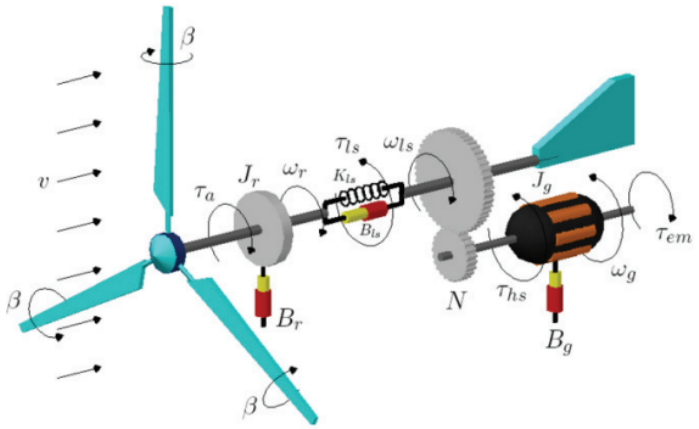


Figure 2.1: Rotor aerodynamics model proposed by Gonzalez et al. [2011]

controller and pitch angle actuator. A sensing element such as a power transducer would provide the power measurement to give the feedback element of the control scheme. The controller would allow for an input to set the controller objectives. As an example, if the controller objective is aimed to achieve a set electrical power output then the control logic will allow for a reference input of the desired value.

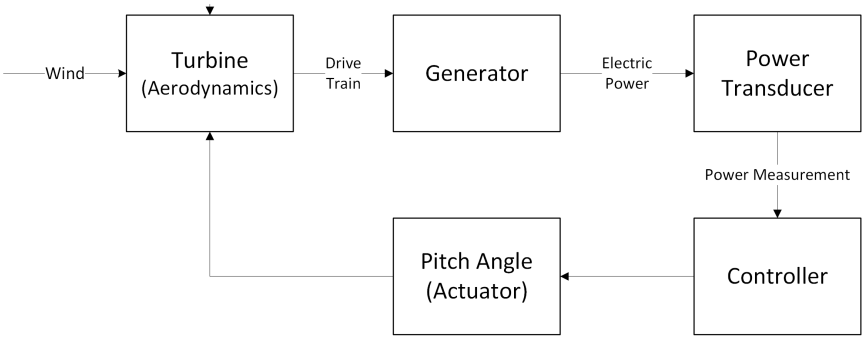


Figure 2.2: General scheme of the closed-loop control of a WT

Generally, blade pitch controllers are implemented by a Programmable Logic Controller (PLC) that is located in the control electrical cabinet of the wind-turbine. This controller will have electrical wiring connections to an electrical or hydraulics actuator, typically a servo-motor or hydraulic piston. Additionally, the PLC will have electrical wiring connections from several transducers that provide real-time measurements of the different systems that comprise the wind-turbine. Particularly, electrical measurements such as voltage and current and, thus, electrical power output from the generator. Due to this, it is common that in simulation and modelling studies a Hardware-In-the-Loop or HIL solution is developed. This allows for the use of mathematical models in consonance with real hardware to allow for a controlled and safe test environment.This allows to validate control algorithms or operations prior to their deployment on the real system (Nabi et al. [2004], Millitzer et al. [2019]). However, HIL environments are not plug and play solutions and are basically designed for

very specific scenarios, as they require a specific mathematical model that defines the process to control (Mihalič et al. [2022]).

Wind-turbines are also engineered with well defined operation regimes in consonance with the wind velocity, delimited by cut-in wind speed, nominal wind speed and cut-out wind speed. Nominal output electrical power plateaus after reaching nominal speed and is set to 0 when operating outside of the cut-in and cut-out wind speed boundaries, as shown in Figure 2.3. Operation Regime I corresponds to a stand-still or coasting operation depending on rotor aerodynamics and/or if a braking mechanism is engaged. In operation Regime II the wind-turbine is operated at variable speed between P_{min} and P_{nom} ensuring maximum power point with an optimum tip speed ratio (i.e. pitch angle at zero). Operation Regime III consists of the wind-turbine power operating at its rated value (i.e. nominal torque) and saturating at said P_{nom} value. Lastly, operation regime IV is a safety operation zone where a safety de-rating or shutdown sequence is enabled (i.e. by turning the blades out of the wind) when $v_{cut-out}$ is reached. Electrical power output of the wind-turbine at this wind speed is defined as P_{cut} and generally coincides with P_{nom}.

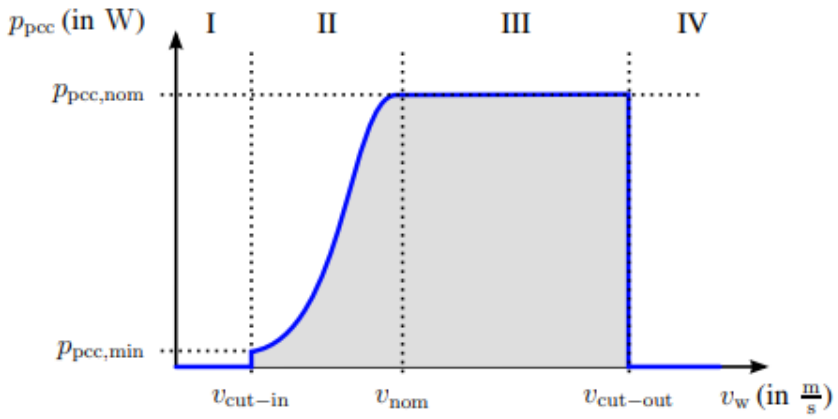


Figure 2.3: Operation Regimes of a wind-turbine (Dirscherl et al. [2015])

These fundamentals are focused on providing an understanding of the different aerodynamic, mechanical and electrical forces particularly relating to horizontal axis wind-turbines. The base mathematical model to be used for this master's thesis, which consists of a small-scale 7kW three-bladed horizontal wind-turbine where the mechanical and electrical parameters were summed up in Table 2.1. The mathematical model of this wind-turbine is used in a Hardware-In-the-Loop application aiming to simulate the physical plant or process needed by said application. Making use of the previous general scheme of a closed-loop control of a WT, the HIL application may be defined as in Figure 2.4. The Hardware-In-the-Loop would consist of the different aerodynamic, mechanical and electrical components of the wind-turbine, including the embedded pitch actuator and the power transducer that measures the power output (in orange). The emulated power measurement would then be fed to the controller via Open Platform Communications Unified Architecture (OPC-UA) and the

controller would produce a pitch angle reference for the actuator to follow.

Symbol	Description	Value	Units
ρ	Air denisty	1.223	kg/m^3
R	Radius	3.2	m
L_a	Motor armature inductance	13.5	mH
R_a	Motor armature resistance	0.275	Ω
k	Constant of the generator	23.31	-
Φ	Magnetic flux	0.264	$V/rad/s$
F	Friction coefficient	0.025	$Nm/rad/s$
J	Inertia momentum	6.53	Kgm^2
R_L	Load resistance	8	Ω

Table 2.1: Theoretical model mechanical and electrical parameters

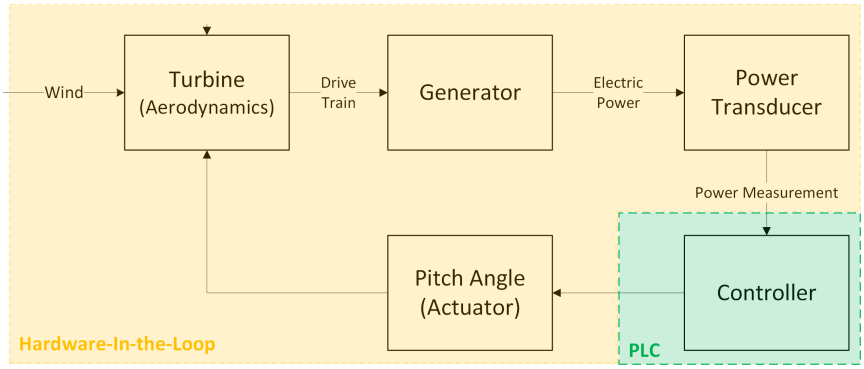


Figure 2.4: HIL Application overview

Lastly, while the application aims to emulate the full range of the wind-turbine's power curve and all four operation regimes, the following studies mainly focused on operation regimes II and III; that is to say, for wind speeds between cut-in and cut-out speeds.

Chapter 3

Materials and methods

This chapter relates to the different materials and tools that comprise the Hardware-in-the-loop simulation solution. The methodology followed for the development and testing of the solution was done in phases, ensuring each feature worked as intended prior to fully integrating each system. This is evidenced by the structure of Chapters 4, 5 and 6 where, after validating the model's behaviour under a particular set of points of operation, the controller development followed by ensuring communication via OPC-UA could be established.

3.1 Simulink model

The wind-turbine mathematical model is based on a small three-blade horizontal wind-turbine of 7kW, which has been documented in Mikati et al. [2012] implemented in "Electrical grid dependency analysis of hybrid distributed power systems" (Mikati et al. [2013]), and in the development of PIL environments to validate wind turbine controller models (Martinez et al. [2023]). The model does not include a gearbox and makes use of a permanent magnet DC motor to represent the rectified output of the electrical generator of the wind-turbine as used by Khan and Iqbal [2009]. The associated mathematical expressions have been summed up in Equation 3.1 (Mikati et al. [2012]), where ω represents rotor speed, T_m is the mechanical torque, I is the induced DC current, J is the inertia momentum, F is the friction coefficient, k is a constant for the generator, Φ is the magnetic flux, R_L is the resistive load and R_a, L_a and I_a are the resistance, inductance and current of the armature, and where each parameter is defined in Table 2.1.

$$\begin{aligned} J\frac{d\omega}{dt} &= T_m - k\phi I - F\omega \\ L_a\frac{dI}{dt} &= k\phi\omega - (R_L + R_a)I_a \end{aligned} \tag{3.1}$$

The rotor model makes use of the general Equation 3.2 to define the power coefficient C_p. $C_p(\lambda, \theta)$ is a non-linear function of the tip speed ratio λ and the pitch angle θ (Abdallah

et al. [2018]) where variations or approximations of the general equation are widely-used in different WECS applications (Thiringer and Linders [1993], Voltolini et al. [2012]). This particular model makes use of the coefficients that were approximated by Ackermann [2005] using multidimensional optimization for a variable-speed wind turbine and are detailed in Table 3.1. To emulate the dominant mechanism for over speed and power control of small wind turbines also known as furling (Audierne et al. [2010]), Mikati et al. [2012] simplified the effect with the following polinoms in equation 3.3, where C_p is calculated by function 3.4 depending on the wind speed.

$$\begin{aligned} C_{p,1}(\lambda,\theta) &= c_1\left(\frac{c_2}{\lambda_i} - c_3\theta - c_4\theta^{c_5} - c_6\right)\exp\frac{-c_7}{\lambda_i} \quad \text{where,} \\ \lambda_i &= \left[\left(\frac{1}{\lambda + c_8\theta}\right) - \left(\frac{c_9}{\theta^3+1}\right)\right]^{-1} \quad \text{and} \quad \lambda = \frac{R\omega}{v} \end{aligned} \tag{3.2}$$

$$\begin{aligned} C_{p,2} &= P_1 v^3 + P_2 v^2 + P_3 v + P_4 \\ C_{p,3} &= (v - (v_{change} + 2))P_5 + P_6 \end{aligned} \tag{3.3}$$

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
0.73	151	0.58	0.002	2.14	13.2	18.4	-0.02	-003

Table 3.1: coefficients used by the model to obtain C_p

$$C_p = \begin{cases} C_{p,1}(\lambda) & \text{for} \quad 0 \leq v < v_{change} \\ C_{p,2}(v) & \text{for} \quad v_{change} \leq v < v_{change} + 2 \\ C_{p,3}(v) & \text{for} \quad v_{change} + 2 \leq v < v_{final} \end{cases} \tag{3.4}$$

The model only takes into account linear horizontal wind forces as a measured disturbance input to the process. The pitch angle θ fits in as the manipulated variable and the output power P_{out} as the controllable variable of the process. P_{out} is passed through a saturation block in order to emulate the different operation regimes of a wind turbine that were defined in Chapter 2 and whose output is expressed by P_{cut} (see Equation 3.5). Other internal variables are exported out of the mathematical model for analysis purposes but do not play any roll in the overall scheme of the control system.

$$P_{cut}[kW] = \begin{cases} 0 \quad \text{for} \quad 0 \leq v < v_{cut-in} \\ P(v,\theta) \quad \text{for} \quad v_{cut-in} \leq v < v_{v_{nom}} \\ P_{nom} \quad \text{for} \quad v_{nom} \leq v < v_{cut-out} \\ 0 \quad \text{for} \quad v_{cut-out} \leq v \end{cases} \tag{3.5}$$

3.2 Programmable Logic Controller (PLC)

The PLC chosen for this implementation is from SIEMENS. In particular, a controller from the SIMATIC S7-1500 family in order to be able to run a simulated application in the same user-terminal via PLCSIM Advanced. This program is part of the STEP 7 environment and TIA Portal (Totally Integrated Automation) the center automation tool for SIMATIC. In fact, TIA Portal is the central tool for managing automation data and the associated editors in the form of a hierarchically structured project (Berger [2014]). This project includes the definition of the network diagram, the configuration of the SIMATIC devices that portray the network, the programming of the controller's logic in different low level and high level programming languages, and the design and programming of the visual interfaces. It also includes other tools to facilitate the integration of PID controllers or the generation of OPC-UA servers and clients not hosted by the SIMATIC controller.

The programming language used to develop the controller's logic is based on Sequential Control Logic (SCL), a high-level programming language well established in the automation community since the early 80's (Ashar et al. [1992]) and mainly used when implementing finite state machines.

3.3 OPC-UA

The communication protocol to facilitate the connection between the Siemens Programmable Logic Controller and the MATLAB Simulink model is based on OPC UA given its ease of implementation, flexibility and variability, and its rising use in the Industry 4.0 (Schleipen et al. [2016]). Open Platform Communications Unified Architecture (OPC-UA) is standardized by the IEC 62541 and the main advantage that it poses for this implementation is that is platform-independent and scalable (Lee and Sung [2022]).

OPC-UA is governed and maintained by the OPC Foundation and, as all OPC protocols, it is a client-server and/or server to server based communication protocol. In a Client-Server architecture, the server provides data and services and the client requests data and services. As a service-oriented architecture, various services are defined for tasks; such as discovery, session management, and data access. It is considered a secure communication protocol as it allows for the implementation of authentication and authorization mechanisms such as role-based access for different users and certificate generation, and digital signatures for data encryption. In particular, OPC-UA unifies all types of OPC, such as, DA (Data Access) and HDA (Historical Data Access) for real-time or historical data from an external source, and AE (Alarms and Events).

On a related note, it is common for one-to-one communications between two devices and, in particular, systems that implement quiescent algorithms (i.e. call to external functions) to include a heartbeat as to verify and check whether the connected device is still calculating its

logic and sending packets (i.e. still alive). This loss of packets and data may not necessarily be related to a link-down issue and that is where this concept is of most interest. A very common heartbeat method consists of implementing an increasing counter for each connected device, described by Kawazoe Aguilera et al. [1997]. In essence, if the neighboring connected device or process were to crash or disconnect, the counter would stop increasing and, thus, missing a "heartbeat". A heartbeat, however, is a means to an end and not a failure detection system in itself. It is up to the system designer to implement the logic of what is considered a communication failure. For instance, implementing a condition where a certain amount of time where the heartbeat has been lost would be considered a communication failure.

3.4 PID controller

Blade pitch control is one of the vastly studied and simulated case-studies in the systems and control engineering academic community for which there has been many different implementations of controller strategies, some more complex or convoluted than others. Laks et al. [2011] and Dunne [2016] analyze the need of optimization by means of feed-forward controllers due to the varying nature of the wind resource whereas Elsisi et al. [2021] investigates the design of an ANFIS-Based blade pitch controller. Blade pitch control however is not solely focused on wind speed variations but it is also used for vibration control for which Fitzgerald et al. [2019] proposes a wavelet-based approach in the control strategy. In general, most academic studies implement their base case-comparison controller based on a standard proportional integral (PI) controller due to its simple structure and robustness. These articles also notice that a simple PI controller requires to be based on an ideal model and well-defined mathematical models. However, its simplicity allows for PI controllers to be used as a starting point when studying the response of a process and to be used as the base comparison for more complex controllers.

PID controllers correspond to a control structure with feedback control, that is to say, the output of the process or plant $y(t)$ is measured and passed back to be compared with the input reference or set-point $r(t)$ for the control system. This is defined as the error $e(t)$ and could be expressed as $e(t) = r(t) - y(t)$. The error $e(t)$ is then fed to the controller so that it can calculate the action $u(t)$ expected for the process to follow and respond accordingly. The description of the PID controller in its continuous form is defined in Equation 3.6 where K_p is the proportional gain, K_i is the integral gain and K_d is the derivative gain. The main objective of the PID controller is to improve the dynamic response and to reduce the steady-state error (Hasanien [2013]). It is modeled by its transfer function in Equation 3.7.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{3.6}$$

$$G(s) = k_p + \frac{k_i}{s} + k_d s \tag{3.7}$$

The inherent definition of the controller tackles the shift or change in the reference value and aims to control the output or response of the process. The controller's influence of the process' response can be adjusted via the proportional, integral and derivative gains by understanding the definition of each term. K_p will give a proportional input with the error, K_i enables the control system to add up the previous errors to the input, and K_d gives an addition from the rate of change in the error to the system control input. Any control system then requires an initial PID tuning phase and may be followed by a PID optimization process. Different approaches have been developed since the original publication of Ziegler and Nichols [1942], especially in the field of artificial intelligence as there are many implications and limitations on the Ziegler-Nichols method (Hasanien [2013]). However, more complex optimization methods require a proper definition of the optimization objectives in order to provide a functional solution. In this case, the optimization objective is just limited to ensuring system stability in normal ranges of operation, so the Ziegler-Nichols method is a valid starting point and, in particular, PID tuning by means of studying its step response.

In this process, the closed-loop measurement or feedback consists of the generated electrical power output of the wind turbine (controlled variable). The set-point or reference to follow is the desired electrical power output and the manipulated variable consists of the blade pitch. Wind speed plays the role of the perturbation into the system. That is to say, the electrical power output is influenced by one uncontrolled variable (wind speed) and one manipulated variable (blade pitch by means of a power reference). The PID tuning shall study two step responses, one for a change in reference and another for a change in wind speed. As PID parameters cannot be modified during normal operation of the controller, the final solution should ensure best-response scenario for both step type changes.

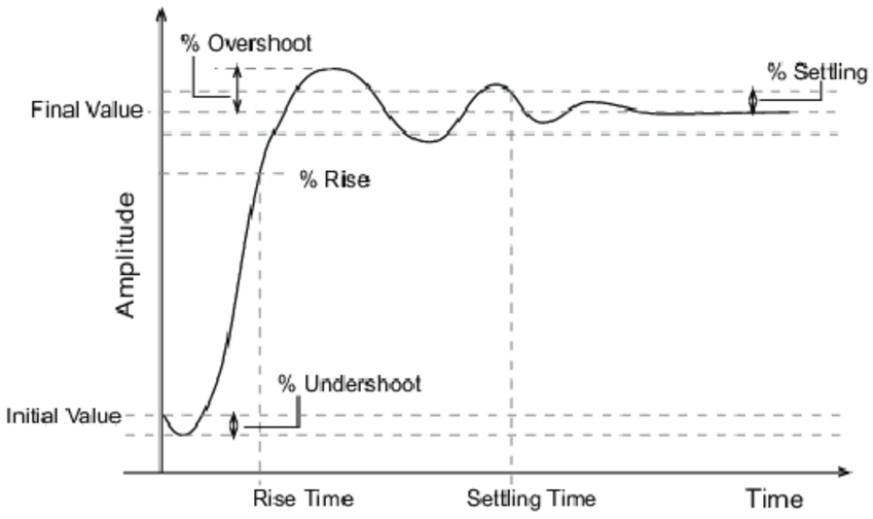


Figure 3.1: Step Response characteristics (Ledeneva et al. [2008])

The step-response may be evaluated by the different characteristics of the plotted response in the continuous domain as shown in Figure 3.1. Overshoot, settling time and rising time acceptable values can be defined in order to tune the PID Controller. The optimization objectives for the blade pitch controller could be defined as in Table 3.2, following typical expected responses by electrical grid utilities. Values under the set limits are acceptable.

Characteristics	**Accepted Limits**	**Units**	**Comment**
Undershoot	0	%	
Overshoot	5	%	Maximum overshoot allowed
Rise Time	500	ms	Allowed time for the controller to respond to the step change
Settling Band	15	%	Allowed variations after reaching steady-state
Settling Time	30	s	Allowed time for the output to achieve 90% of the Final Value

Table 3.2: PID desired response

Chapter 4

Definition of the initial conditions problem

While studying the original wind turbine model, the need of establishing certain initial conditions for the simulation to reach its operational steady state was identified. These are founded on the model's mathematical Equations 3.1 and 3.2 and can be summed up to be wind speed, blade pitch, rotor speed and DC Current. Also, Mikati et al. [2012] states that this particular wind-turbine model was found to behave best with small wind speed oscillations. That is to say, the wind model should not include abrupt stops or big wind gusts. The study of the model started by understanding this wind speed limitation and the aerodynamic model.

The main problem faced during real-time simulation involved the need of applying a fixed sampling time. It was detected that the simulation would come to an unexpected end when set with a high sampling time (e.g. $\Delta t = 0.01s$) whereas a smaller sampling time would allow to have a successful simulation; however the end-result would accumulate an offset with actual clock time (loss of real-time simulation). Additionally, different initial condition configurations had a different outcome for different sampling times, where it was also detected that the higher the derivatives of current and rotor speed, the earlier the error in the simulation appeared or, in contrast, the offset was more prominent. This gave the impression that different initial conditions would allow to adjust this increment of time to be the optimal for the application by minimizing the derivatives for the initial instances of the simulation. Nonetheless, a proper understanding of the problem was needed to allow for a way to resolve the optimal initial conditions for which the simulation may run in real-time.

The model behaviour study then focused on understanding its feasibility in a real-time simulation and the influence of adjusting the initial conditions of rotor speed and generator current. This would serve as the preamble to later formalize the definition of the problem mathematically. The resolution of the problem is tackled in Chapter 5.

4.1 Model behaviour

The starting point for a more detailed understanding of the behaviour of the model began by establishing a constant wind speed close to the nominal power output defined in the wind-turbine's power curve. The simulation was conducted with no time step constraints for the solver and evidences the model's start-up transition until reaching the steady state at nominal power, shown in Figure 4.1. The plotted variables consist of the power output calculated by the model, dc current and voltage, rotor speed, torque, and the simulated wind speed by the wind model. An overshoot is evident in mostly all plotted variables and was replicated for any wind speed input. During this overshoot, V_{dc} saturates at 230V. The rotor aerodynamics block is constructed by a spatial filter and a saturator, as wind speed must not be negative nor zero.

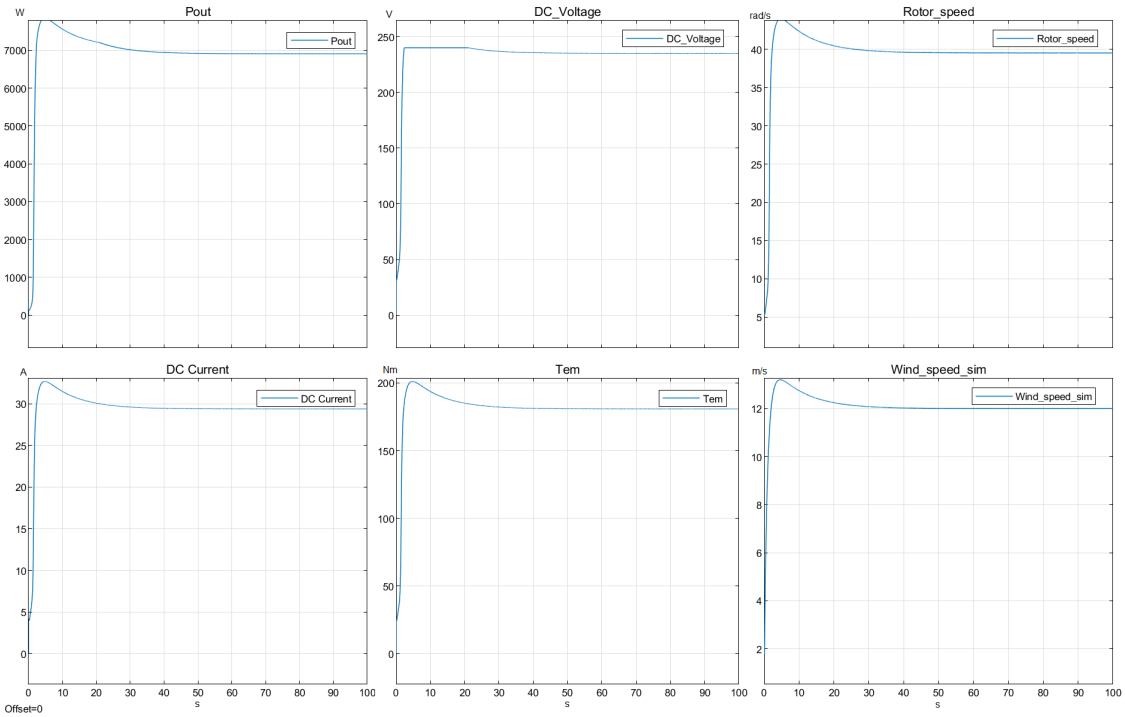


Figure 4.1: Model simulation at nominal wind speed for $I_i = 0.001A$ and $\omega_i = 5.7rad/s$.

To further understand how the initial wind speed input (in green) is processed by the model and the influence of each block filter over the aerodynamic model, scopes were temporarily added for the outputs of the lag filter (in blue) and spatial filter (in orange) and represented in Figure 4.2. It could be concluded that the lead-lag compensator block was at fault for inducing said overshoot in the overall response of the model. This behaviour enables a more accurate theoretical response to the influence of wind speed changes over the rotor speed and, thus, over the power output.

Further open-loop model testing where the wind speed was simulated to ramp up from 0 to 15m/s showed that the model is unable to start and reach steady state, showcasing

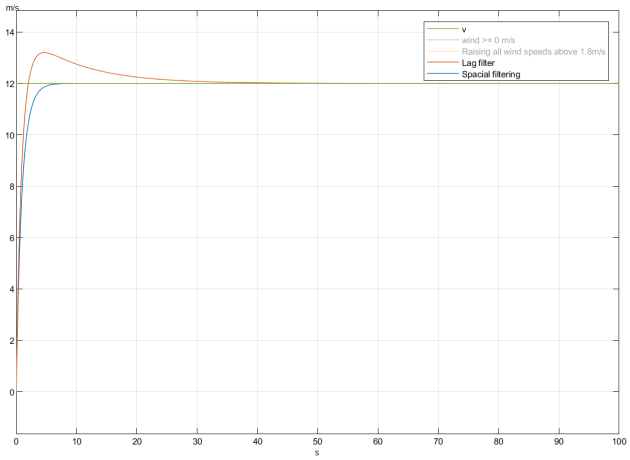


Figure 4.2: Wind model response for a nominal wind speed step

a transient state in the model where certain initial parameters need to match up in order to ensure the wind-turbine is able to follow the theoretical power curve. This constraint is mentioned in Mikati et al. [2012] and establishes a start-up wind speed of 3m/s. The following simulation would then induce a wind speed ramp starting at 3m/s where it could be seen the major influence in the first instances of the spatial filter of the wind to mechanical power model mentioned previously. The initial instances ($t < 3s$) of this simulation shown in Figure 4.3 represent a non-linear or unnatural behaviour where the angular rotor speed ω seems to rise up and down until it stabilizes, allowing for the electrical power output to follow the expected theoretical power curve.

4.1.1 Real-Time Simulation

One of the most critical requirements for the HIL application is to ensure the simulated model is able to run in real-time, that is to say, the simulation solver must calculate the model's variables synchronized in real-time. First instances of the Real-Time simulations showcased a need to use a variable-step solver, as the model heavily relies on the use of integrators to obtain the rotor speed and dc current variables, which in some instances may need a faster solving time-step.

However, it is a requirement for the HIL application to have a fixed time-step as to ensure controller and simulation are never out of sync. If this were the case, the only way to re-sync both entities would be to restart the simulation, which also involves resetting the communications. This is not very robust for the objective at hand, so a fixed time-step should be used. When implementing a fixed time-step at $t = 0.01s$ it was found that the simulation cannot solve the model and runs into an exception where the solution for a calculated variable was not finite during the first instances. Faster sampling times did not fall into this issue but the simulation would start adding up some offset from actual real-time as the computational cost is higher. For this particular requirement, the sampling time must

be equal or less than the controller's time-step ($T_s \leq T_{controller}$). Other considerations to keep in mind in order to define the sampling time is that the quicker the sampling time, the higher the computational cost, not only for the simulation, but it may also have an effect on the communications, inducing lag and loss of packages. Additionally, the application itself does not require very fast responses from the controller as it involves a mechanical actuator (blade pitch control).

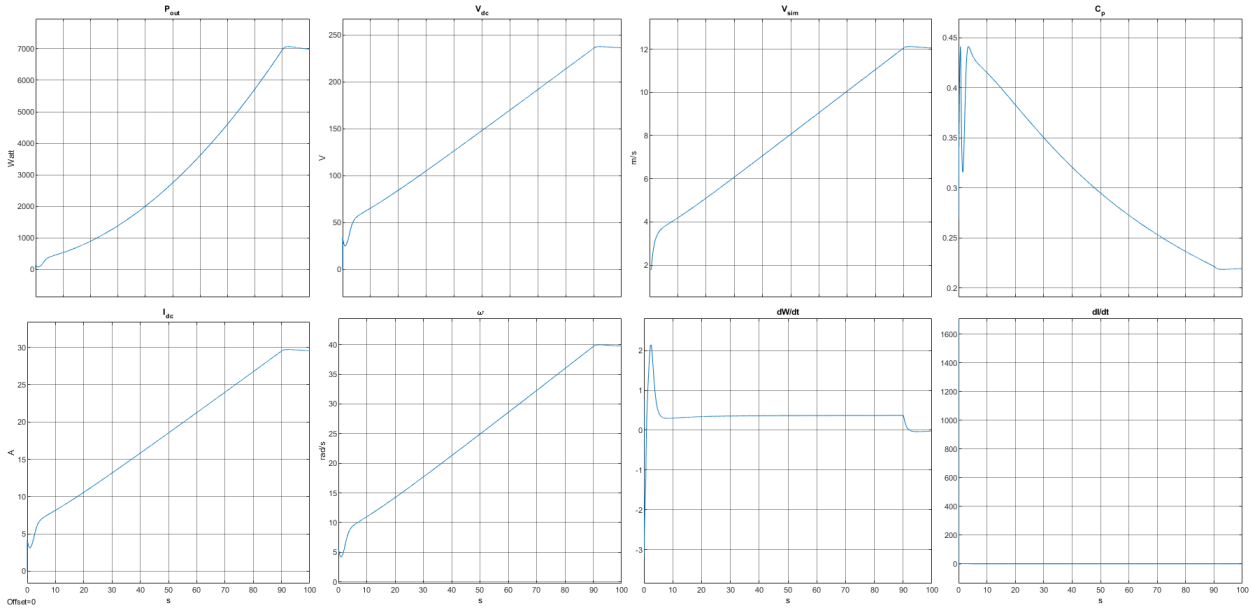


Figure 4.3: Model Simulation for $v_{ini} = 3m/s$ and ramp-up to $12m/s$.

4.1.2 Initial conditions mismatch

The origin of this issue is believed to be tied to a mismatch on the initial parameters of ω and I as evidenced by their corresponding time derivatives, $\frac{d\omega}{dt}$, and $\frac{dI}{dt}$ where their values seem to be about -3 and 2000, respectively. In effect, this was demonstrated while running the simulation without adjusting the initial dc current and rotor speed parameters, for which the initial calculated mismatch at $t = 0$ was especially high for DC Current (see Figure 4.3, where the value of $\frac{dI}{dt}$ was around 2000). This initial mismatch in the initial parameters induces a non-linear behaviour of C_p where it is seen to oscillate during the first instances of the simulation.

In order to have a closer look into these initial instances and try to identify why the model requires a faster sampling time during these initial instances, different initial parameters for current and rotor speed were set, revealing that the smaller the derivatives during the initial time-step the smoother the curves. Also, when running with a fixed time-step of $t = 0.01s$ the unexpected end of the simulation would take effect at a later step. This breakthrough defined the necessity to understand in detail how the variables of rotor speed and generator

current were calculated by the model and their relation with the time steps. Special attention had to be given to $\frac{d\omega}{dt}$ and $\frac{dI}{dt}$.

Additionally, the simulation was not run in real-time but at a fixed wind speed of $v = 2m/s$ not only to demonstrate the initial mismatch at values closer to the lowest input value allowed by the model ($v_{min} = 1.8m/s$) but also to understand these initial instances and to reveal the model's behaviour at wind speeds under cut-in speed. Under these circumstances, the model is unable to reach the theoretical steady-state and seems to get stuck in an initial transient state where rotor speed and current trend to zero (see Figure 4.4). The first six seconds of this simulation allow for a closer study of what each variable has been calculated to be for $t = 0s$ and the initial error given by $\frac{dW}{dt}$ and $\frac{dI}{dt}$. It is then evident the saturator blocks for v and ω configured in the model play a major role in this initial instance of the simulation, as any value under this pre-configured parameter will create a mismatch in the derivative calculations.

Figure 4.4 also shows the difference between the output variables P_{out} where even if the simulation works outside of the normal operation ranges of cut-in and cut-out wind speeds, electrical power output is still calculated by the model but the discrete logical output of $P = 0W$ is set through P_{cut}. It is also evident that the calculated outputs derive from non-linear equations and are highly influenced by each other. In particular, it is clear the relation between I_{dc}, ω and C_p with v, where the error in $\frac{dW}{dt}$ starts to accumulate until ω saturates to its lower limit and then the error stays somewhat constant close to zero. The error can be defined as the difference between the expected value of ω and I for a certain wind speed input. That is to say, for no change of wind speed, $\frac{d\omega}{dt}$ and $\frac{dI}{dt}$ would be expected to be zero or close to zero.

This phenomenon defines the limit for which the model is able to produce power. Any wind speed under a certain minimum value v_{min} will trigger that the derivatives for ω and I get de-synchronized with the theoretical value calculated by the model's equations, as C_p falls too low and no electrical power output can be extracted from the aerodynamic wind forces. This point of operation, however, fits into the normal operation of a wind-turbine for which a cut_in_speed is defined, needing to implement into the purely mathematical model this scenario by limiting some key parameters to not fall under a certain threshold where the wind-to-mechanical power is not capable of creating enough rotor speed to induce DC current generation and, thus, electrical power output. In essence, it is necessary to establish a P_{min} as real wind-turbines do in normal operation when in a ready-to-produce state.

4.1.3 Optimal conditions

So, prior to continuing with heuristic attempts to find a viable solution to these simulation problems, a more symbolic and mathematics approach was introduced and the need to define the mathematical model arose, especially to define and identify the correlation between

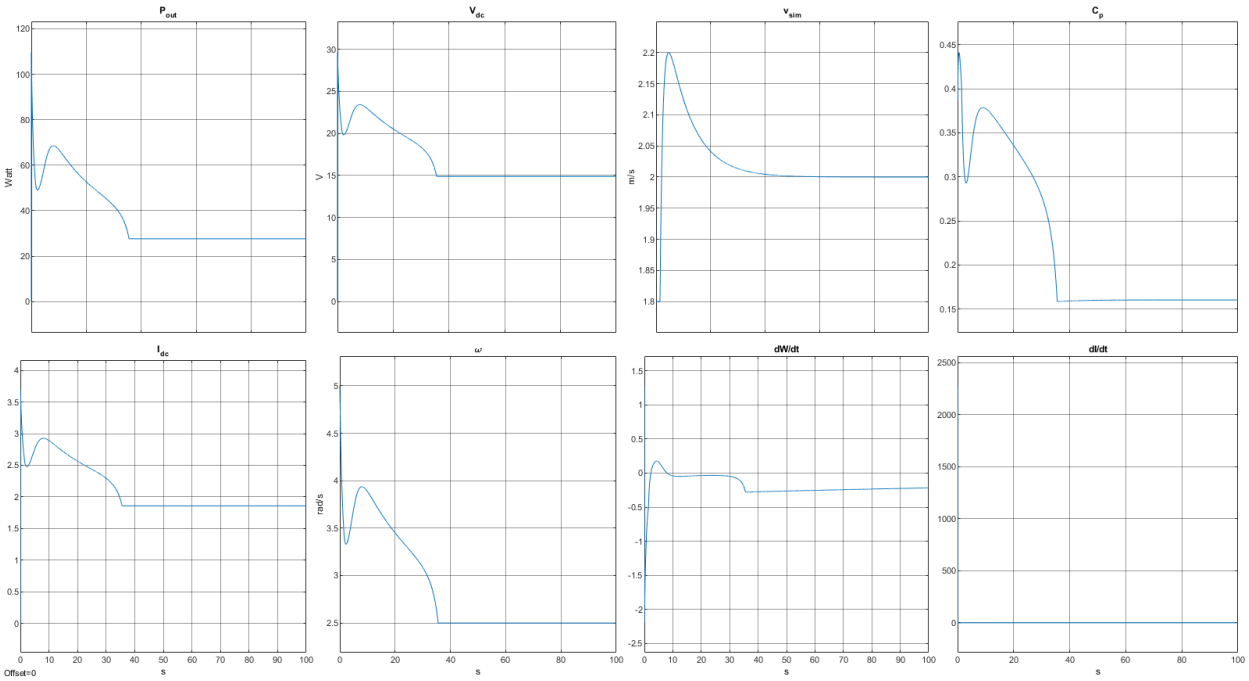


Figure 4.4: Model Simulation at $v = 2m/s$ for $I_i = 0.001A$ and $\omega_i = 5.7rad/s$.

variable v, ω and I. This would allow to implement different methods to resolve for different optimal set of solutions of initial conditions, under certain constraints or limits. In particular, the solution would aim to identify the minimum initial wind speed value at which the simulation is able to start. At this point of operation, the goal would be to find the values of rotor speed and generator current for which their derivative are zero or close to zero, i.e., minimize the initial mismatch of the theoretical value of these variables and the calculated by the model for $t = 0s$.

Only after this initial mismatch being minimized, could one then look into adjusting the optimal sampling time for this set of optimal initial conditions as it was witnessed that real-time simulation behaviour was influenced by changing these initial conditions for the same sampling time step. At a fixed sampling time, the solution may not be finite if the error accumulated between time steps is too large. This situation is more prominent when the derivatives $\frac{d\omega}{dt}$ and $\frac{dI}{dt}$ are high. This approach in real-time the step size, as hinted by the simulation solver when running into the error.

In summary, the optimal conditions solution intends to reduce the step size for the simulation. As very small time-steps were not a viable solution as an offset appeared to be desynchronizing the simulation's real-time, the other way to reduce the step size was to ensure small increments of error between each fixed-time steps. The highest error increment of the simulation was observed during the initialization process, where the initial conditions set for rotor speed and generator current should be properly adjusted. These initial conditions are needed for the first calculation of the derivatives $\frac{d\omega}{dt}$ and $\frac{dI}{dt}$, at $t = 0s$, as ω and I in the model are calculated by means of integrating their corresponding derivative.

4.2 Mathematical description of the problem

The previous observations elude to the need for understanding the mathematical relationship between the variables that comprise the model. This is known as computational causality, where the different systems and parts that the model has been divided into (aerodynamic model, mechanical and torque model, electrical model) and described in Section 2 not only depend on each other, but are also reliant on the contour constraints of the system and the specified linkage between each part. The first step in this endeavor focuses on classifying the variables into known or unknown in accordance to their nature. Known variables are:

- constants and parameters that do not change during the course of the simulation, such as, generator load impedance, coefficients and blade dimensions.

- global inputs to the model, such as, pitch and wind speed.

- state variables which are calculated by means of integrating the derivative counterpart, such as rotor speed and dc current.

Any other variables present in the model and that do not fall into the previous definitions would be considered unknown variables. Also, the derivative result of the state variables which become auxiliary variables are considered unknown variables as well. The problem may then be represented by $F(x, y) = 0$ where x is a vector of all the known variables of the model and y the vector for the unknown variables.

The problem definition relies on equations 2.2, 3.1 and 3.2 from which the variables have been classified as described in Table 4.1. The model has 5 unknown variables and 5 equations, where each unknown variable may be matched with one equation without having to match the same equation for two unknown variables. The model would then be defined mathematically as in Equations 4.1, 4.2, 4.3, 4.4 and 4.5.

Known Variables	t, ρ, R, v, J, k, F, I, ω, L_a, R_L, R_a, $c_{1:9}$, θ
Unknown Variables	T_m, C_p, λ, $\frac{d\omega}{dt}$, $\frac{dI}{dt}$

Table 4.1: Classification of variables

$$T_m = \frac{1}{2}\rho\pi R^3 v^2 \frac{C_p}{\lambda} \tag{4.1}$$

$$C_p = c_1\left(\frac{c_2}{\left[\left(\frac{1}{\lambda+c_8\theta}\right)-\left(\frac{c_9}{\theta^3+1}\right)\right]^{-1}} - c_3\theta - c_4\theta^{c_5} - c_6\right)\exp\frac{-c_7}{\left[\left(\frac{1}{\lambda+c_8\theta}\right)-\left(\frac{c_9}{\theta^3+1}\right)\right]^{-1}} \tag{4.2}$$

$$\lambda = \frac{R\omega}{v} \tag{4.3}$$

$$\frac{d\omega}{dt} = \frac{1}{J}(T_m - k\phi I - F\omega) \tag{4.4}$$

$$\frac{dI}{dt} = \frac{1}{L_a}[k\phi\omega - (R_L + R_a)I_a] \tag{4.5}$$

These equations and variables essentially constitute the base for the model's simulation where the first step is to assign values to the constant variables, whereas the state variables ω and I would be calculated by means of integrating their derivatives $\frac{d\omega}{dt}$ and $\frac{dI}{dt}$, respectively. The way the simulation may calculate these variables rely on its integration algorithm that may very well be based on Euler's explicit method. In this method, the ordinary differential equation (ODE) $\frac{dx}{dt} = f(x,t)$ is solved by means of an initial given value of x_i where the next value of the variable in the following instance x_{i+1} corresponds to the sum of the initial or previous instance value and the value of the derivative of x in accordance to the time interval Δt that has taken place. This is defined in Equation 4.2. Algebraic variables such as C_p or T_{em} are calculated in each time instance from the value of the state variables calculated in that instance.

$$x_{i+1} = x_i + f(x_i, t_i)\cdot\Delta t \tag{4.6}$$

This definition gives an understanding as to what may cause the simulation to fail for a certain time step, where if the initial mismatch in ω and I that is inherited from an incorrect x_i configuration and given the evident mathematical dependency between the model's variables, the calculation of these variables in the next instance may not be finite. Due to this, it is considered that a proper definition of Δt and the initial values for ω and I that minimize the initial mismatch in their corresponding derivatives for $t = 0s$ would aid in solving the identified problem. In other words, high changes or steps in the calculated variables of ω and I are related to their corresponding derivatives. If their derivatives are high but small increments of these variables are required, then Δt must be small as to reduce the error in the simulation ($\Delta x = \frac{dx}{dt}\Delta t$). Due to the Real-Time simulation having a fixed step time requirement for the OPC-UA application, it is needed to find the initial conditions for which the value of the derivatives is limited during the start-up of the simulation.

Chapter 5

Optimal initial conditions for model simulation

5.1 Initial parameters for simulation

Due to the problem defined in Chapter 4, the aim of the model study then shifted to resolve both symbolically and numerically the mathematical equations. This would help to understand the dependencies of the initial parameters between each other and to calculate an initial state where the increment of rotor speed (see Equation 5.1) and DC current were zero or close to zero. The main dependency found mostly related to the power coefficient calculation, as all four variables are present. Also, it is clear that DC current, rotor speed and C_p are dependent on one another. The problem was then solved by identifying operation ranges of the model (see Equation 5.6) based on the case study of the mathematical equations and validating through the open-loop simulation of the model for different initial parameters. See in Table 5.1 the different cases simulated of major relevance.

$$\frac{d\omega}{dt} = k_1 \frac{v^2 C_p}{\omega} - k_2 I - k_3 \omega \tag{5.1}$$

$$\frac{dI}{dt} = \frac{K\omega - (R_L + R_a) I_a}{L_a} \tag{5.2}$$

$C_p(\lambda, \theta)$ was defined in Equation 3.2 and is a function of λ and blade pitch angle, where λ is the relation between rotor speed and blade tip speed ($\lambda = \frac{R\omega}{v}$). In order to ensure an unconstrained start-up situation, the blade pitch angle was set to 0. This falls in line with normal operation of the wind-turbine, where closed-loop control of the pitch angle would start to take effect once operational wind speed ranges are achieved. The symbolic resolution of C_p is then simplified in Equation 5.4 showing the dependencies with the rest of the variables that have been identified in the initial step problem.

$$C_{p,1}(\lambda, 0) = c_1 \left(\frac{c_2}{\lambda_i} - c_6 \right) \exp\left(\frac{-c_7}{\lambda_i} \right)$$
$$\lambda_i = \left(\frac{1}{\lambda} - c_9 \right)^{-1} \tag{5.3}$$

$$C_{p,1} = \left(k_4 \frac{v}{\omega} - k_5 \right) \exp\left(k_6 - k_7 \frac{v}{\omega} \right) \tag{5.4}$$

From Equations 5.1 and 5.4 we can observe that it is necessary to set a minimum value for ω throughout the whole simulation to ensure no divisions by zero take place. This limit, ω_{min}, also needs to ensure a certain minimum C_p value calculated for $v_{min} = 3m/s$ to not fall into a singularity. This is due to the intrinsic correlation of C_p and ω. The singularity in question can be explained by alluding to the definition of C_p and its physical meaning. This value gives a relative value between the wind forces in contact with the blade surface necessary to transmit movement into the rotor (angular velocity). This generated mechanical power is then translated to electrical power via the DC generated current. That is to say, if C_p falls under a certain threshold, wind power transmitted to the blades will not be sufficient to generate angular velocity in the rotor in order to generate DC current. This phenomenon predicates the need of ensuring a starting point of sufficient initial rotor speed to kick-start the model, not only at the initialization of the simulation but also after a stop in power generation after singularity is achieved. This situation is also reachable if pitch angle reaches a certain upper threshold, as C_p is a function of θ. Additionally, $C_{p,1}$ calculation was corrected in the intended equation as an erratum was detected in the original Simulink model.

5.1.1 Solve for zero

A first attempt to find a solution to this problem focused on finding the initial values of ω and I_{DC} for a certain wind speed value for which $\frac{d\omega}{dt} = 0$ and $\frac{dI_{dc}}{dt} = 0$, that is to say, $F(x) = 0$. The solver in question made use of the Levenberg-Marquardt algorithm which works in empirical iterations to find the minimal value for variable "x" where $F(x) = 0$ up to a certain tolerance. The tolerance for the solver was set to $1e_{-2}$. As one could expect, the result put out by the solver was for $\omega = 0$ and I_{dc}=0, i.e. the wind turbine is not generating any power as C_p is zero.

Given the incoherent result of this attempt, a second method was tried that relied on a solver that makes use of the Nelder-Mead simplex algorithm based on making an initial guess for solving the problem centered on a predefined x_0. The initial guess was calculated by solving $\frac{dI}{dt} = 0$ for an initial wind speed of 1.8m/s, which corresponds to the initial wind speed of the simulation at $t = 0s$ (as the Simulink model includes a saturator block whose minimum value is set to 1.8 m/s). The solver defined the result for the initial guess to be $\omega = 3.5rad/s$ and $I_{dc} = 2.6A$, and the simulation has been plotted in Figure 5.1, showing

$\frac{dw}{dt}$ and $\frac{dI}{dt}$ closer to zero at $t = 0s$.

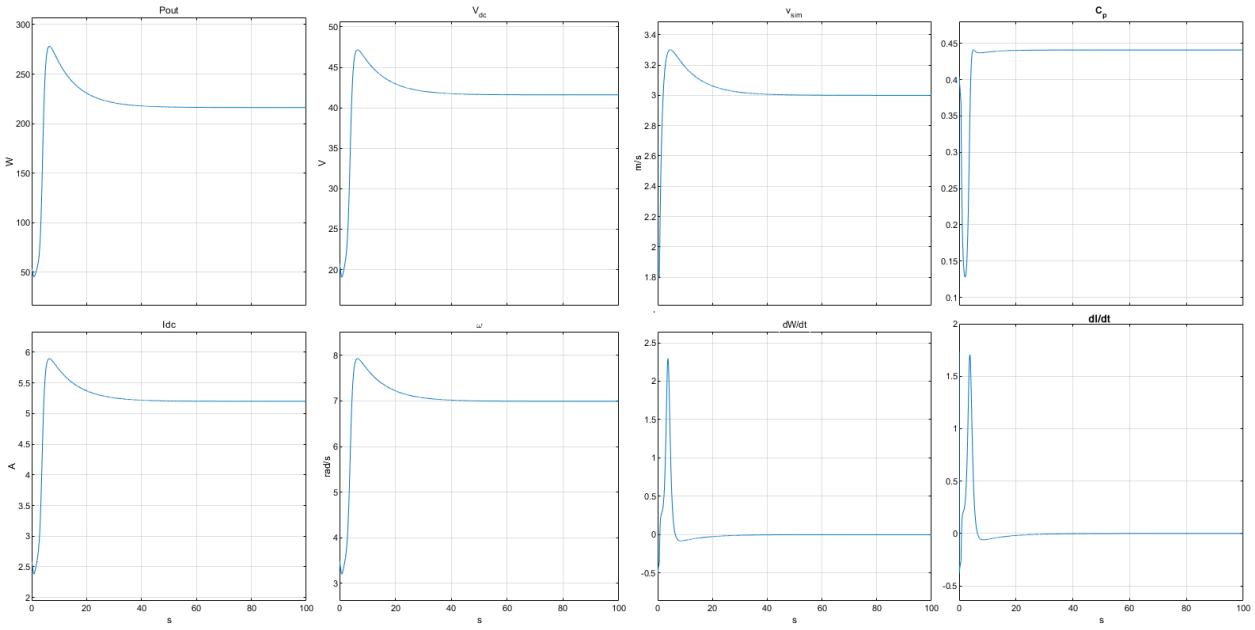


Figure 5.1: Responses for $\omega_i = 3.5rad/s$ and $I_{DC} = 2.6A$

Essentially, the results given by the solver even with the initial guess being quite close to the desired output were solutions that still fell under a certain value of ω (for example, 1.8 rad/s) that provides a very low C_p and, thus, not valid start-up parameters. This is due to not considering wind speed as an unknown variable as well and relying on the predefined $v_{min} = 3m/s$ assumption when solving for zero. However, for $t = 0s$ wind speed is lower than $v = 3m/s$ so v_{min} also has to be re-defined.

5.1.2 Optimization problem

This problem then becomes an optimization problem, where it is needed to define the function $f(x)$ to minimize and identify the parameters x to modify . In this case, it consists of a multi-objective optimization problem formulated as:

$$\min_{x \in X}(f_1(x), f_2(x), \ldots, f_k(x))$$

where the integer k is the number of objectives, three for this application (v_0, ω_0, and I_0). The functions to be minimized consist of the absolute values for the time derivatives of ω and I for a specific value of C_p, defined in Equation 5.5.

$$F(\omega, I_{DC}, v) = \begin{cases} \left|\frac{d\omega}{dt}\right| \\ \left|\frac{dI}{dt}\right| \\ C_{p_{min}} - C_{p,1} \end{cases} \tag{5.5}$$

The constraints of the optimization problem are subject to the accepted initial values of ω, I and v at the initial instance of the simulation, which have been defined in Equation 5.6. The lower and upper limits for the parameters were set in Equation 5.7.

$$\begin{aligned} v &\geq v_{min} \\ \omega &\geq \omega_{min} \\ C_p &\geq C_{p_{min}} \\ \theta &= 0 \\ \left|\frac{d\omega}{dt}\right| &\leq Threshold\ A \\ \left|\frac{dI}{dt}\right| &\leq Threshold\ B \end{aligned} \tag{5.6}$$

Thresholds A and B for the integrator errors were determined empirically in accordance to the observed $C_{p_{min}}$ needed to ensure the model does not run into the singularity. This was done by running different simulations close to and under cut-in wind speed, and inducing step changes in the input wind speed. As stated before, the model calculates ω and I_{dc} by means of integrating the error of those variables, being the only independent variable v in Equations 5.1 and 5.2. This threshold however may be surpassed in normal operation when simulating at higher wind speeds. Threshold A was established at $0.01\frac{rad}{s}$ and threshold B at $1A$. $C_{p_{min}}$ was identified to be around 0.1, and v_{min} and ω_{min} were pre-set in the Simulink model at $1.8m/s$ and $2.5rad/s$, respectively.

$$\begin{aligned} v_{min} \leq v &< v_{cut\_out\_speed} \\ \omega_{min} \leq \omega &< 50 \\ 0 < I_{DC} &\leq 1500 \end{aligned} \tag{5.7}$$

Lastly, the optimization problem was solved by specifying the values of the input parameters for which to establish the solver's initial guess. These were set to $\omega = 3.5rad/s$, $I_{DC} = 2.6A$ and $v = 3m/s$.

Table 5.1 shows different simulated cases that were of interest in order to summarize the process of finding the best initial parameters for the simulation. In particular, the solved values from the optimization problem solver are used in Case 2, where error in initial rotor speed and DC current are close to zero and the constraints of the nonlinear multi-variable functions are met. Case 0 corresponds to the inherited initial parameters of the Simulink model, case 1 corresponds to the calculated I_{dc} at the inherited initial parameters for $\frac{dI}{dt} \approx 0$, and case 3 corresponds to the solved initial parameters at cut-in speed where $\frac{d\omega}{dt} \approx 0$ and $\frac{dI}{dt} \approx 0$ are met; however this solution did not meet $C_p \geq C_{p_{min}}$.

The singularity where ω approaches zero is still achieved at a later time step if wind speed remains constant or if wind speed would drop slightly as shown in Figure 5.2. The

Case	v(m/s)	ω(rad/s)	I_{dc}(A)	C_p	$\frac{d\omega}{dt}$	$\frac{dI_{dc}}{dt}$
0	**3**	**5.7015**	0.001	0.39678	17.812	2598.4
1	**3**	5.7015	**4.24**	0.39678	13.814	0.016535
2	**2.0354**	3.318	2.467	**0.2948**	$5.605e{-3}$	-0.2975
3	3	2.267	1.6859	0.014142	**4.54e-5**	**-4.979e-3**

Table 5.1: C_p and derivatives values for different solver solutions ($\theta = 0°$).

simulation initial parameters were then adjusted as in Table 5.2 to allow for some additional error in the first instances of the simulation so not to allow the calculations of C_p and ω to drop under an unrecoverable value.

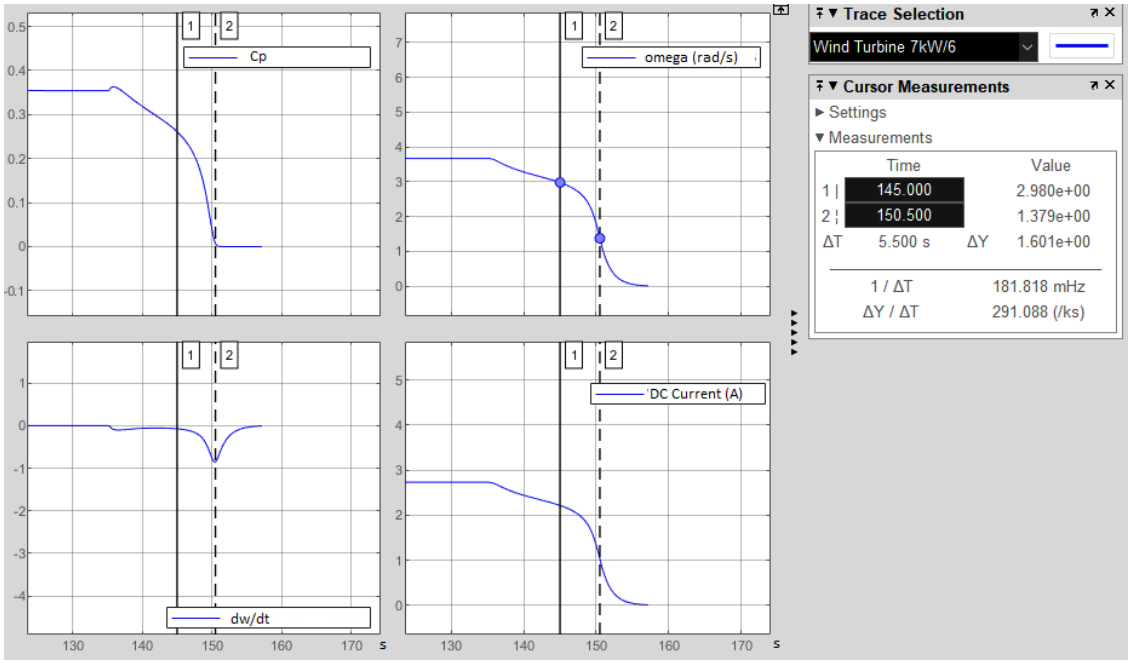


Figure 5.2: Responses for C_p, ω, $\frac{d\omega}{dt}$ and I_{DC} during the singularity event at wind speeds under cut-in-speed.

From these results v_{min} was modified in the original model to saturate at a lower limit of $v_i = 2.036\ m/s$ to ensure singularity does not occur during real-time simulation.

v	θ	ω	I
2.036	0	3.248	2.416

Table 5.2: Initial parameters configuration for the model simulation

5.2 Sampling time

Simulink is a high-resource demanding application and implements different solver types for prioritising resource efficiency and a fast simulation. This means the sampling time

for the solver may not stay consistent, depending on the calculations that need to take place at each time step. HIL environments require that the sampling time of the controller and the simulated model generally match. Realistically, the time step constraint actually relies on the simulation model being the same as or faster than the controller. Common practice, however, is for both sampling times be the same or close to the same as very high sampling rates for the simulation require higher performance machines or take over a lot of the machine's resources (Liu [2000]). This application runs two emulated environments on one terminal, so this should be avoided. For this reason, the study then shifted to validating the least restrictive sampling time where the simulation results were still reliable for the unconstrained model's response. This is needed in order to ensure synchronization between the mathematical model running in Simulink and the real-time PLC.

This is achieved by introducing the Real-Time Synchronization block into the Simulink model and synchronizing both, the sampling time of the PID controller and the runtime. The PID sampling time was set to $T = 0.1s$ as the expected response times of the pitch controller are not meant to be very fast. Also, the values being exchanged between the PLC and the simulated model through OPC-UA were read/sent every $T = 0.01s$ on the PLC side. The Simulink model was validated to run at this latter fixed sampling time by including some temporal manual knobs to modify wind speed and pitch variables in real time. The behaviour of the model was similar when comparing to the behaviour of previous non real-time simulations.

5.2.1 Real-Time desktop solver

In order to understand the model's sampling rate limits and behaviour, different solver configurations were tested prior to introducing the Real-Time Synchronization block. As a baseline for the study, the simulation was run in real-time with variable-step and auto-selected solver with a tolerance of $1e-3$. A step signal was induced for the pitch angle reference to also measure simulation responsiveness.

The solver was configured with a fixed sampling time of $T_{s_1} = 0.01s$, $T_{s_2} = 1ms$, $T_{s_3} = 0.1ms$ and $T_{s_4} = 1\mu s$. For T_{s_4}, the simulation requires a lot of computational resources and accumulates a considerable delay to actual real time. For T_{s_3} the simulation also accumulates some latency but is not as prominent as for T_{s_4}. In this particular sampling time, two solver types were tested, ode3 and ode5; however, no improvement in accuracy was detectable. For reference, an ode3 solver uses the Bogacki-Shampine Formula integration technique to compute the state derivatives whereas the ode5 solver uses the fifth-order Dormand-Prince, translating to a higher computational cost. T_{s_2} on the other hand, showed no latency whatsoever. Lastly, T_{s_1} proved to be too restrictive as the simulation would end unexpectedly at about $t = 2.51s$ due to the fact the error accumulated between time steps is too big and the simulation result of the derivative trends to infinity. Figure 5.3a illustrates

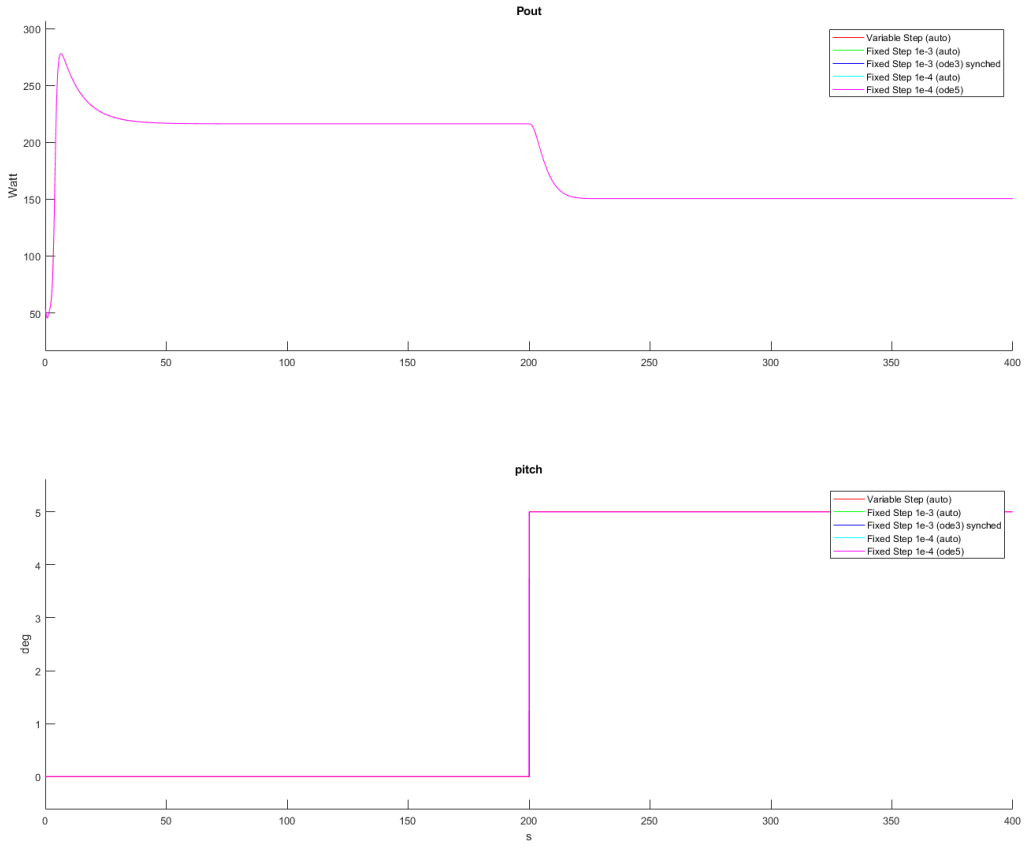
the results previously described and Figure 5.3b particularly shows no significant difference between a variable step and fixed step solver. Table 5.3 provides an overview of the different configurations tested. The offset was quantified by running the simulation for one minute and subtracting how much of the actual simulation was simulated.

Configuration	T_s(s)	Solver	Offset (s/min)	Observations
1	0.01	ode3	-	Simulation runs into an exception at $t = 2.51s$
2	1e-3	ode3	-	
3.1	1e-4	ode3	.8	Some latency, real-time simulation is lost
3.2	1e-4	ode5	1.2	Some latency, real-time simulation is lost
4	1e-6	ode3	56	Latency and may exceed max. missed ticks count, real-time simulation is lost

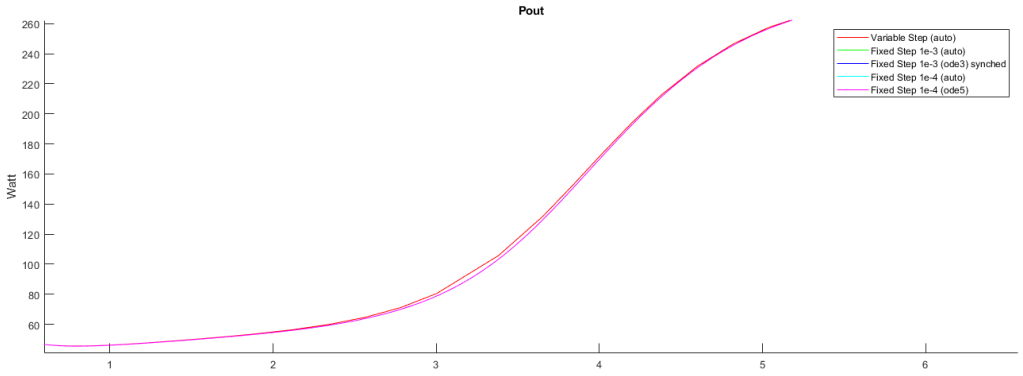
Table 5.3: Real-Time simulations for different sampling times

Simulation time results and latency may vary on the terminal's specifications and processing power. For reference, these simulations were run on a computer desktop with an AMD Ryzen 7 5800X 8-Core Processor 3.80 GHz and 16GB of installed RAM. The GPU consisted of an NVIDIA GeForce RTX 2070.

Figure 5.3 also represents (in blue) the case study of a fixed step solver with the Real-Time Synchronization block set to $T = 0.01s$. This response overlaps with the rest of the other configurations, evidencing no visible loss of accuracy and validating these settings.



(a) P_{out} simulation for a step in pitch reference



(b) Zoomed in from 5.3a during the initial seconds of the simulation

Figure 5.3: Simulations for different solver configurations

Chapter 6

Implementation of the HIL controller on OPC-UA

The network diagram shown in Figure 6.1 defines the S7-1500 PLC as the OPC-UA Server, whereas the simulated model in Simulink serves as the OPC-UA Client. Communication between the Human-Machine Interface running on WinCC will communicate with the PLC via TCP/IP. The solution is defined so that all systems may be simulated or run on the same machine. However, the solution is defined in a way that it may be implemented with real hardware systems, as each simulated device can be assigned an IP address. These IP addresses will all fall under the same range by ensuring they are part of the same LAN (Local Area Network).

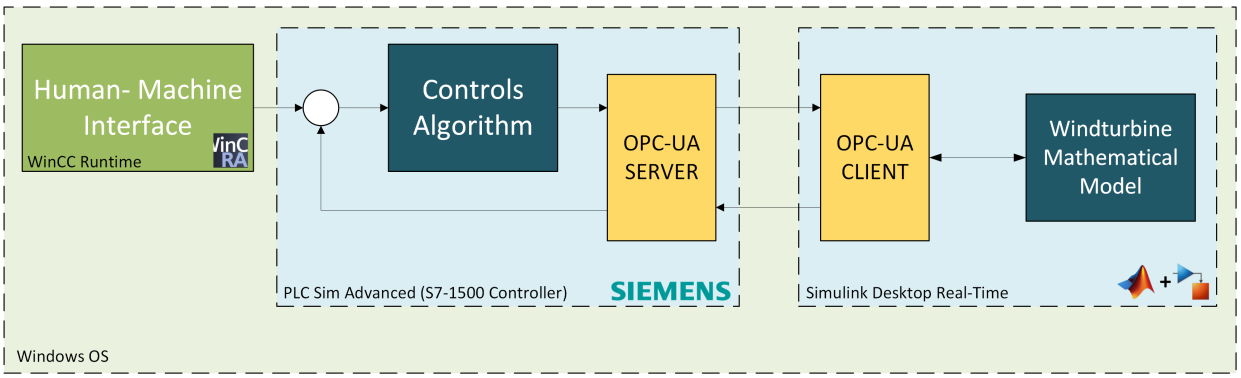


Figure 6.1: Block diagram of the HIL application

6.1 Hardware-In-the-Loop (HIL)

The Hardware-in-the-loop solution was composed of the Simulink model studied in the previous sections running as a Real-Time Desktop application and an emulated Programmable Logic Controller. The PLC simulation application creates a virtual Ethernet adapter emulating a physical Ethernet connection between the user-terminal and the PLC. The PLC

instance and Virtual Network Adapter configurations can be referenced in Annex C, which follow the IP address definition in Table 6.1. Note that HMI and Real-Time Desktop application share IP address, since both environments run in the same terminal.

Device	IP Address
PLC	192.168.20.1
HMI	192.168.20.250
Real-Time Desktop Application	192.168.20.250

Table 6.1: IP addresses of the systems that compose the solution

The OPC-UA data exchange functionality was established by introducing an Interpreted Matlab Function block in Simulink (see Annex D) that would call to an external function stored in the workspace directory. This function is detailed in Annex A. It creates a client session targeting the PLC IP address on a user-defined port where the OPC-UA server is published. The OPC-UA server initially had only two variables for testing purposes. The variables were θ and power output. The latter serves as the controlled variable for the PID controller and θ is the manipulated variable aimed for the servo-motor actuator of the blade pitch. The PID block calculated by the PLC was set to manual where it allowed to manually manipulate the percentage of the output of the controller, allowing for an open-loop control and test communication, and the response of the model to the inputs from the PLC through OPC-UA. This input test is shown in Figure 6.2 and shows how the model behaves with different pitch references and wind speeds in real-time simulation. Power output reaches values over the rated nominal power for wind speed higher than nominal speed as the furling function in the model was disabled and the PID controller had not been configured yet during this test iteration. This test focused solely on establishing proper propagation of the variables via OPC-UA and apporpiate response of the model to pitch angle references. At higher pitch angle and constant wind speed, power output is derated.

After successfully establishing the OPC-UA communication between the Simulink model and the Siemens controller, the Human-Machine Interface (HMI) was designed, configured and developed. The main function of the screen is to allow inputs by the operator to control the pitch angle reference manually or to allow for the PID controller to calculate the actuator reference according to the error between the active power set-point and the calculated power output sent via OPC-UA by the Simulink model. The HMI also includes a wind turbine icon that changes states depending on the operation mode of the controller or the different operation regimes, as defined in Figure 2.3. A dedicated Function Chart (FC) programmed in SCL was created to handle the read and write signals between the HMI and the PLC (see Annex B).

The same coded function handles the state machine associated to Figure 6.3, for a total of six states. The states are explicitly updated in the screen as well by use of the "WtgStatus" integer. The default value of this integer is set to 0, to coincide with the "NO COMMS"

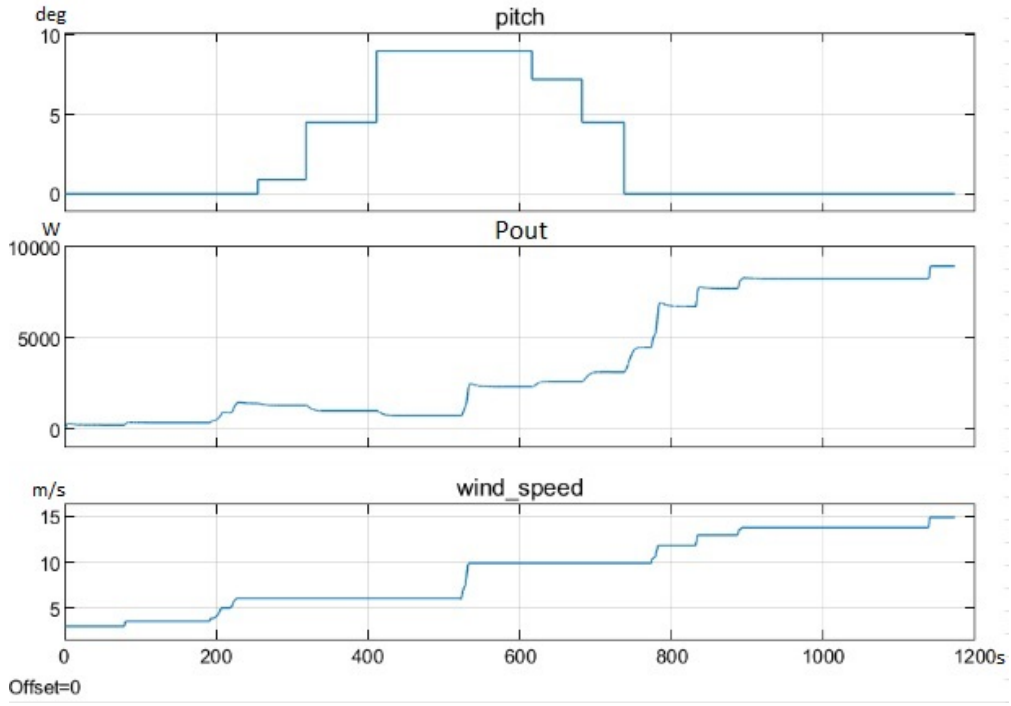


Figure 6.2: Real-Time open-loop simulation with θ sent through OPC-UA

status. This status allows the operator to easily know whether there is an established OPC-UA connection with the Simulink Model or not. This was solved by implementing a heartbeat handler between the Interpreted Matlab function and the controller, where the latter expects the OPC Client to count up in the register "heartbeat" for every function call. If by "t=1s" the heartbeat register is not updated, the controller raises a communications error flag. This is the condition that is used by the state machine to account for this event. State "1" allows the implementation of an E-STOP condition for the controller that would disable the Start Command and disengage the blades. The operator also has control over the Start Command so the state in-between E-STOP and RUN is defined with integer "2". This state is known as Ready to Produce and has been given the name of "PAUSE", where the blades are disengaged from the wind horizontal forces. States "3" and "4" both relate to normal operation of the wind turbine, when the blade pitch angle is controlled by the PID controller. In particular, state "4" allows to distinguish whenever nominal or rated power output is reached (i.e. 7kW) and the wind-turbine's output power is derated. Lastly, when the operator sets the manual blade pitch angle by bypassing the PID controller, this is defined with state "5".

6.2 PID tuning

Once the simulation environment was operational, the PID controller associated to operational states "3" and "4" was manually fine-tuned by adjusting the proportional, integral

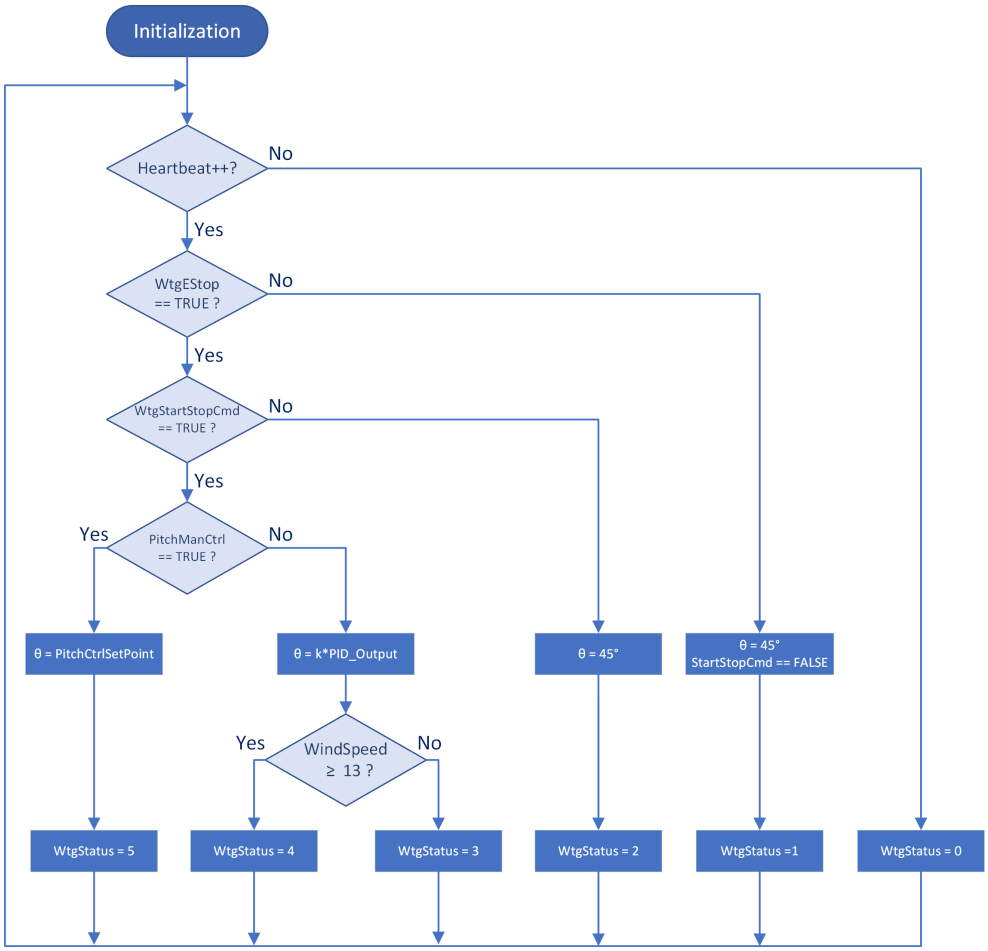


Figure 6.3: State Machine Flow Diagram

and derivative constants. The main goal of this tuning, however, focused on ensuring a stable and controlled response to keep the solution simple, following the expected behaviour defined in Chapter 3. Initial parameters of $k = 0.01$, $\tau_i = 1.333$ and $\tau_d = 0.5$ produced a very fast response, where the actuator would saturate and oscillate between its maximum and minimum operation range, basically emulating the effect of an ON-OFF controller. The parameters were then set to ensure a slower and attenuated response but noticeable enough where reference steps of 500W ensured the set-point was reached in a timely manner. Different setups under different conditions were tested and their response quantified by analyzing overshoot, settling time and steady-state oscillations to the same amplitude step. These indicators were measured in Real-Time by means of the embedded graphs object implemented in the HMI and Simulink scope output.

The first case focused on ensuring a stable response when switching between an uncontrolled and controlled state. This was recreated by setting the blade pitch control to manual with $\theta = 0°$ and a fixed wind speed of $10m/s$. Under these conditions, the output power sets at around $4.5kW$. The different PID configurations tested are shown in Table 6.2.

The next case test iteration focused on negative step response during curtailed operation

Case A	k	τ_i	τ_d	Overshoot (%)	Settling Time (s)
1	0.06	50	0.2	0	168
2	0.08	20	0.8	0	39.7
3	0.08	10	1.2	0	25.3

Table 6.2: Case A: PID tuning setups for a linear step response from an uncurtailed operation.

of the wind turbine (i.e. limited power output by means of the controller). Setting as a starting point a power reference of $4000W$ and blade pitch control enabled, the power set-point was then changed to $3000W$.

Case B	k	τ_i	τ_d	Overshoot (%)	Settling Time (s)
1	0.08	10	1.2	0	7.473
2	0.08	20	0.8	1.62	11.170

Table 6.3: Case B: PID tuning setups for a linear step response during curtailed operation.

For all iterations of Case A and B, undershoot was found to be non-existent, the response time was quantified to be at least the sampling time of the process $T = 0.1s$, and the response fits in with the settling band criteria as no wind speed disturbance is present (ideal model). Simulation results for Case B.2 are shown in Figure 6.4 as reference, where pitch angle reference, output power, wind speed and actual pitch actuator angle were represented. In particular, the figure shows the value of output power at the initial instance of the step change in power reference and the time instance for which the response reaches the desired value. This allowed to easily identify the overshoot of the response.

In order to expand on the fine-tuning of the PID parameters, a random disturbance was introduced into the wind speed input of the model in order to emulate the wind's unpredictability. This was achieved adding to the wind speed variable a uniform random source between $\pm 0.5m/s$ with a sampling time of $T = 0.65s$. This would validate the adjusted PID parameters to the closest operation conditions of the process. Figure 6.5 compares the system response under same operating conditions as the previous PID parameters of Case B. The observations from the ideal step response tests are also present in this scenario, where Case B.2 shows a bit of an overshoot. As for steady-state performance, Case B.1 reflects a more attenuated response but by overcompensating the pitch response to wind fluctuations and slightly limiting the power output. It is also observed the high influence of the wind speed over the power output as the response is fairly similar for any test case.

The final adjustment ($k_p = 0.06$, $k_i = 30$, $k_d = 0.4$) for which the system complies with the settling band requirement and an overall balanced response from the previous observations is shown in Figure 6.6. This consolidated tune was validated by implementing different step changes in power set-points from the HMI whilst applying the wind speed perturbations. It can be observed how the pitch actuator angle goes to zero whenever the

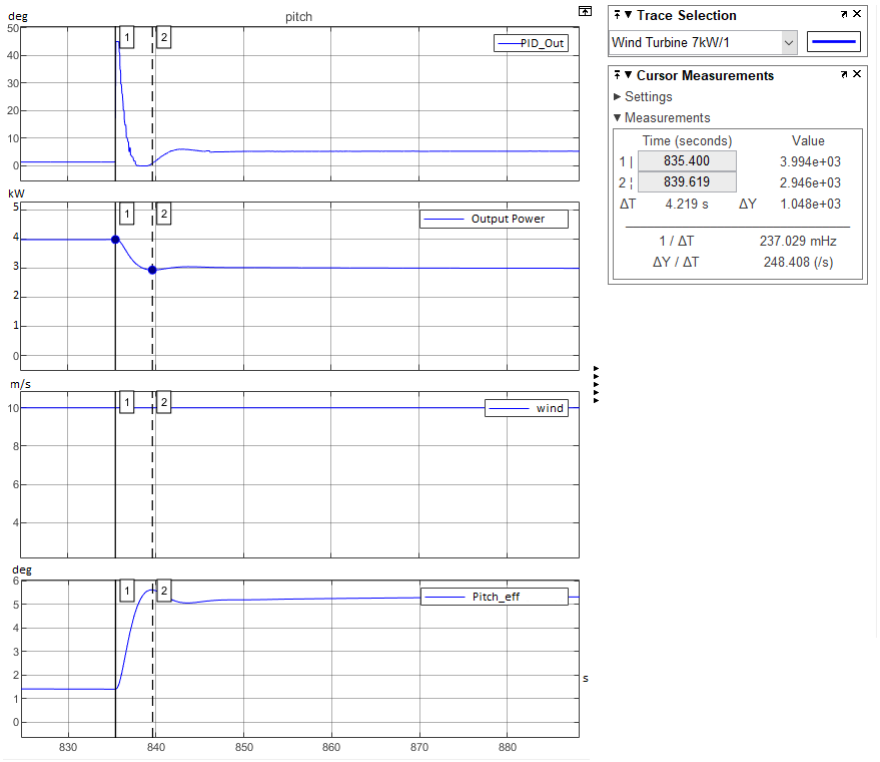


Figure 6.4: System response for Case B.2

theoretical power output for the corresponding wind speed falls under P_{ref}. It can also be seen how the blade pitch controller engages when $v = 13m/s$ to follow $P_{ref} = 6kW$ between timestamps 1000s and 1200s. Pitch actuator angle then changes back to be closer to 0° when P_{ref} is set to 6.5kW in timestamp 1250s.

In order to comply with the initial conditions identified for the model in Chapter 5, all simulations were conducted by implementing a ramp-up for the wind speed variable starting at $v = 3m/s$ and a power output above P_{min} as to ensure pitch reference is $\theta = 0°$ at start-up. This was implemented at the Simulink model by means of a switch block (see Annex D). The ramp would be disabled when wind speed reaches 10m/s. This value is inconsequential and could be set to any other value inside the operation range (between 3m/s and 13m/s).

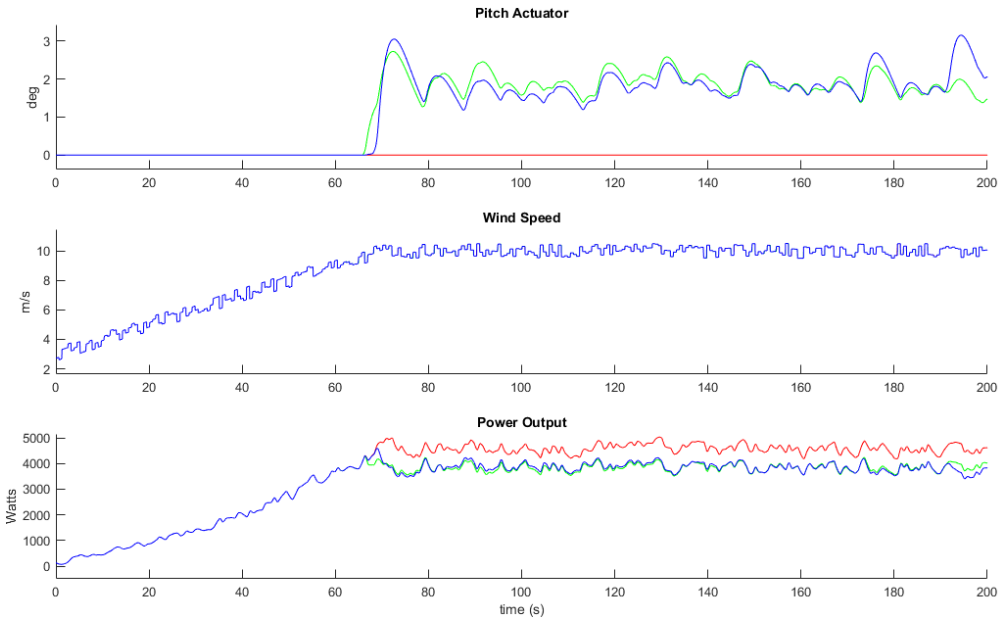


Figure 6.5: In red, PID disabled; in green, Case B.1; in blue, Case B.2

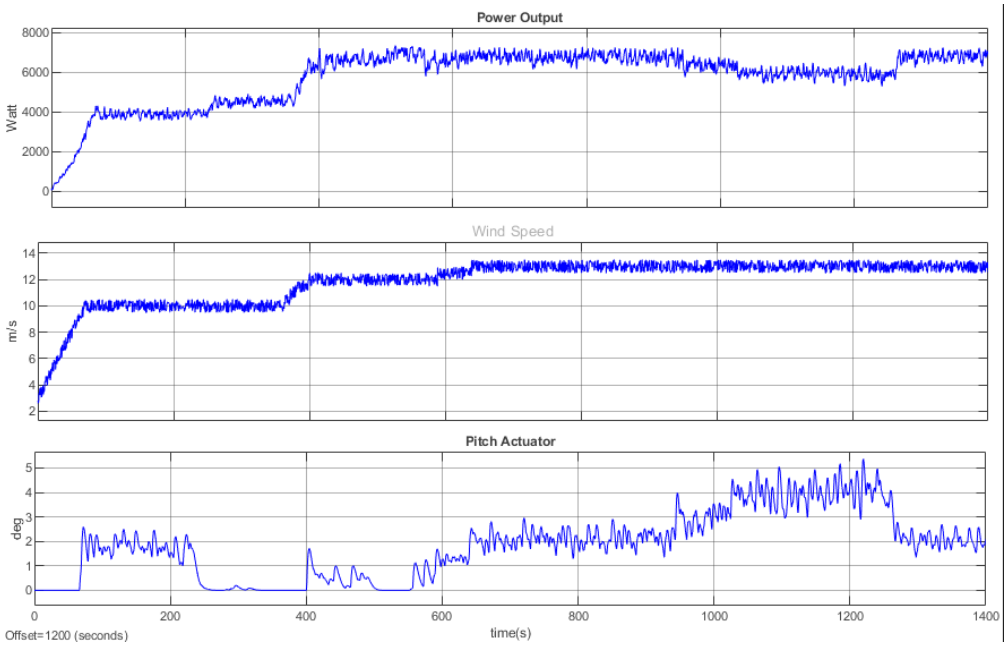


Figure 6.6: Consolidated PID tuning solution

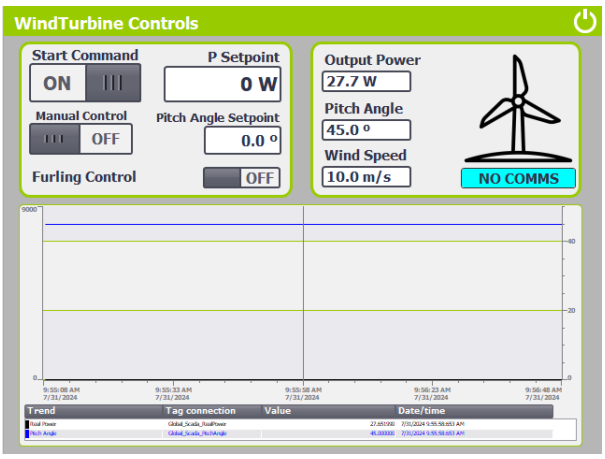
Chapter 7

Results

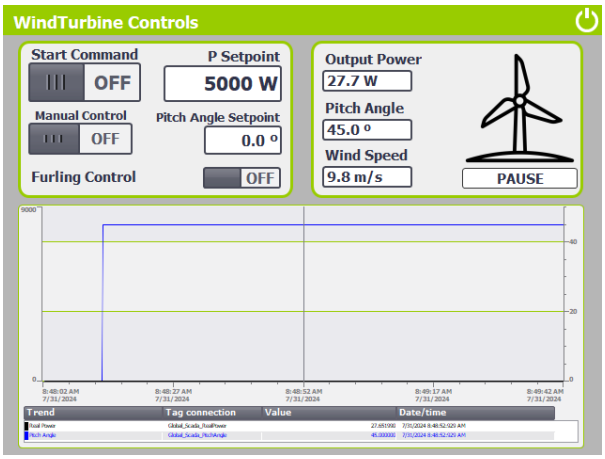
The following chapter presents the full-scale application simulation, resulting from the previous chapters. The simulation environment was set up by opening a Matlab Simulink session on the same terminal as the PLC Simulator instance (PLC SIM Advanced) and HMI runtime (WINCC). The simulated PLC was uploaded with the developed project in TIA PORTAL and was monitored in real-time via the same application. The HMI runtime was also simulated from the TIA PORTAL application. Once the PLC was in a running state, the Matlab workspace holding all the parameters of the Simulink model was set up. The simulink model was then run in Real-Time as described in the previous chapters.

The HMI was designed to include a real-time graphics object that represents the Electrical Power Output read via OPC-UA from the Simulink model and the PID output reference calculated in the cyclic interrupt function. The Simulink model was designed with a scope to observe the pitch reference received by the model from the PID Output, the power output from the model sent to the PLC, the simulated wind speed in correspondence to the wind speed input to model, and the pitch actuator output simulated by the Simulink model (see Figure 7.5). The simulation aimed to validate all the different operational statuses defined for the wind turbine and the response of the model for different wind speeds and power set-points, discussed in detail in Figures 7.1, 7.2, 7.3 and 7.4.

Figure 7.1a represents the scenario where communications with the simulink model is lost. The HMI will display a "NO COMMS" tag and will hold the last received values from the OPC-UA client. No time-out error Sequence Of Events procedure has been defined in the PLC as the mechanical actuator of the blades is not directly controlled by this system, i.e., the behaviour of the actuator in this scenario needs to be defined by the model. As this is a simulated model, there is no risk of harm to any mechanical parts when communications are lost. Figure 7.1b, on the other hand, shows the behaviour of the Simulink model when the start command is disabled via the HMI (switch on the upper left corner of the screen). For this operation state, PID output is manually set to 45° and it is shown how the power output stays at P_{min} even if the wind speed is at $10m/s$. The HMI displays a tag under the wind-turbine icon that reads "PAUSE" where the wind turbine is ready to produce but is

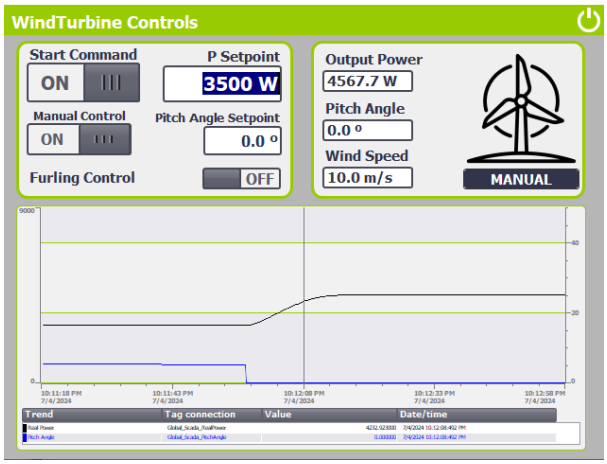


(a) Loss of communications event

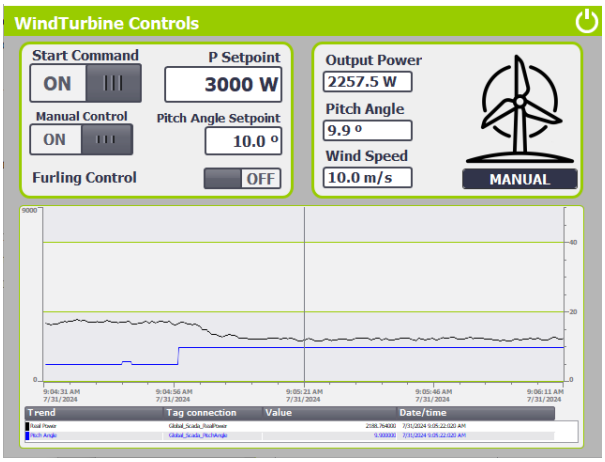


(b) Start command disabled

Figure 7.1: HMI WTG States (I)



(a) Manual pitch angle reference of 0°



(b) Manual pitch angle reference of 10°

Figure 7.2: HMI WTG States (II)

subjected to an aerodynamic stop (i.e. the attack angle of the blades is close to being in parallel to the wind forces).

Figure 7.2 represents the HMI configuration when the operator is required to control the pitch actuator manually. By enabling the manual control, one is then able to set the pitch angle reference required. Figure 7.2a illustrates a step-down response to a pitch reference of 0° and shows the unconstrained power output calculated by the model for a variable wind speed of $10m/s$. Figure 7.2b represents a step-up response to a pitch reference of 10° for the same wind speed value. It can be seen how the model responds by dropping about half of its previous value. In both cases, it reflects that the model does not follow the power set-point of 3500W and 3000W, respectively, as set in the HMI.

Figure 7.3 encapsulates the initialization of the simulation and wind turbine operation in regime II. It displays a smooth power output at the initial instances during the ramp-up from a wind speed of 3m/s up to 10 m/s, in contrast to the first instances of the simulation

analyzed in Chapter 4. Two initialization situations were set to understand the behaviour of the simulated model when faced by a power set-point during wind speed ramp-up. Initially, an unconstrained power set-point was set in figure 7.3a, where power output plateaus when wind speed reaches the end of the pre-configured ramp (10m/s). For the second simulation case, however, power reference for the controller was set to 3000W. It was observed the same ramp-up response. Then, once the power reference is achieved by the output power, the PID controller engages and starts regulating the pitch actuator whilst wind speed keeps ramping up (see Figure 7.3b).

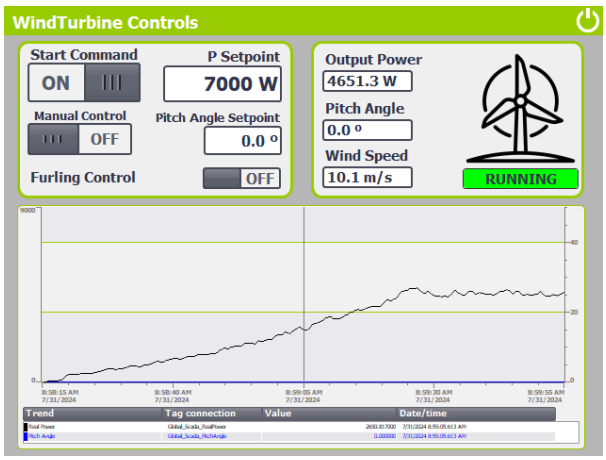
Once steady state was reached for a wind speed of 10m/s, the wind speed input was modified to 12m/s where now it is seen how the PID output dampens the influence of this disturbance over the desired power output (see Figure 7.3c). The PID corrects the response in a timely manner but shows a bit of oscillation. This is expected as the PID was not tuned for compensating the influence of high wind speed changes over the model. On the other hand, Figure 7.3d does demonstrate a controlled response of a power reference step-up, for which the PID was tuned for. In this response, the overshoot is minimal, practically non-existent and the response is subject to the model's lead-lag block that is embedded in the blade's algorithm. Figure 7.3e shows the simulation's response to a power set-point of zero. The simulation behaves as expected, as the PID saturates its calculation to 45 degrees and the model runs down to P_{min}.

Lastly, Figure 7.4 shows a nominal operation state corresponding to operation regime III, where wind speed falls over the nominal wind speed defined for the WT. In particular, Figure 7.4b also depicts the saturation of the power set-point as the PID does not take into account any value above the nominal power defined for this wind turbine model. In both cases, the PID is controlling at a set power reference of 7000W and the model is responding appropriately as the calculated electrical power output stays within the reference.

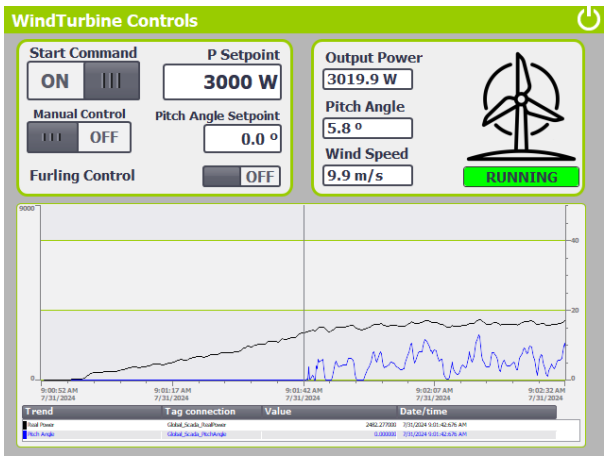
As stated at the beginning of the chapter, Figure 7.5 shows the previous operational states and responses. The simulation begins at $v_{min} = 2.036m/s$ and $\theta = 0°$ for which ω_{ini} and I_{ini} had been set in accordance to Table 5.2. Wind speed is set in the next step to 3m/s and ramps up to 10m/s. Some noise was induced into the wind speed variable input as discussed in previous chapters (see Annex D for the final Simulink model). In parallel, as the power set-point was set to 3000W in the HMI, it is observed how the PID engages once the power output is near the reference. The PID output ramps up in unison to the wind speed until wind speed reaches 10m/s, at which time the actuator also sets at around 5 degrees. At around $t = 200s$, it is known that the manual control of the PID has been enabled as the PID output sits flat for some seconds. Electrical power output is then manually controlled by means of adjusting the pitch actuator angle. PID control is re-engaged at around $t = 300s$ as to prepare the next state of operation. Shortly after, wind speed input is changed to $v = 12m/s$ (rated nominal wind speed) and the PID tries to control the sudden shift in wind speed. As this was not taken into account in the tuning objectives, a slight peak in electrical

power output is reflected but is quickly corrected to stay at the power set-point of 3000W.

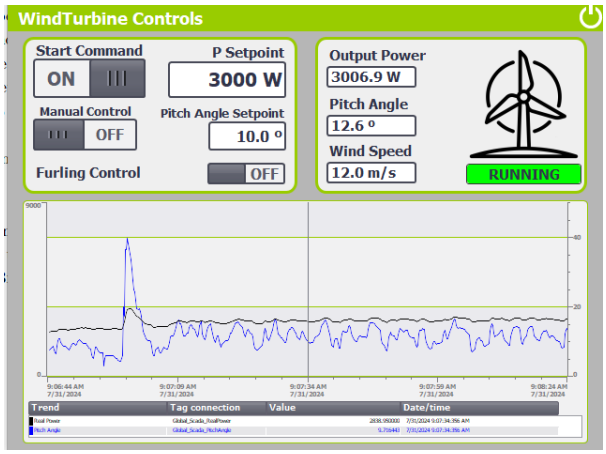
The next transient instance is induced close to $t = 500s$ where the power set-point is modified to 5000W. In this case, however, the response is much more controlled as this scenario was considered during the fine tuning of the PID. This scenario is kept for a couple minutes until another wind speed change is set in the model to induce a curtailment operation. The spike in power is replicated as expected and, once the power output settled back to 5000W, the power set-point was set to 7000W (rated nominal power). The response is also controlled and fits into the controller requirements defined in 3.2. While in this operation regime, a set-point of 9000W was set to emulate a very typical utility compliance test (active power curtailment) where the system cannot go higher than its structural limit (i.e. rated nominal power or contractual power). The system behaved as expected by staying at 7000W (refer to Figure 7.4b. The simulation was ended after sending a zero set-point power reference to the controller, for which the PID saturated at its maximum degree of actuation, resulting in the electrical power output calculated by the model to reach P_{min}.



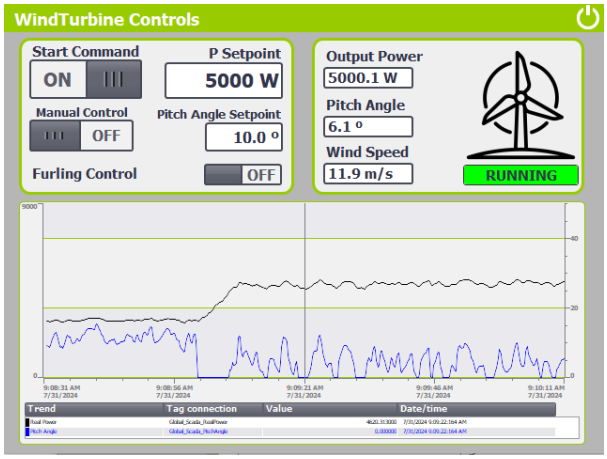
(a) Ramp-Up uncurtailed



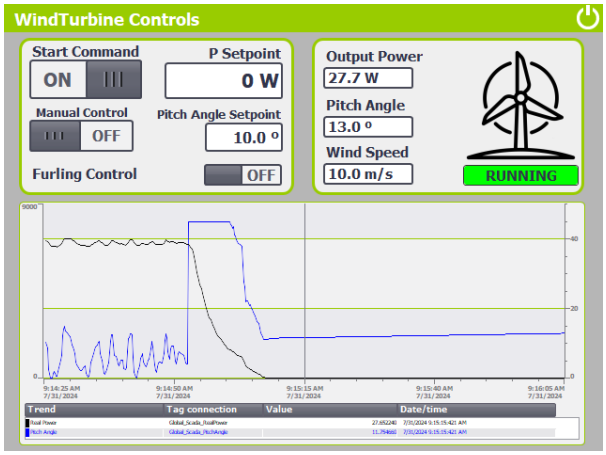
(b) Ramp-Up with a power set-point of 3kW



(c) Wind Speed Step Change

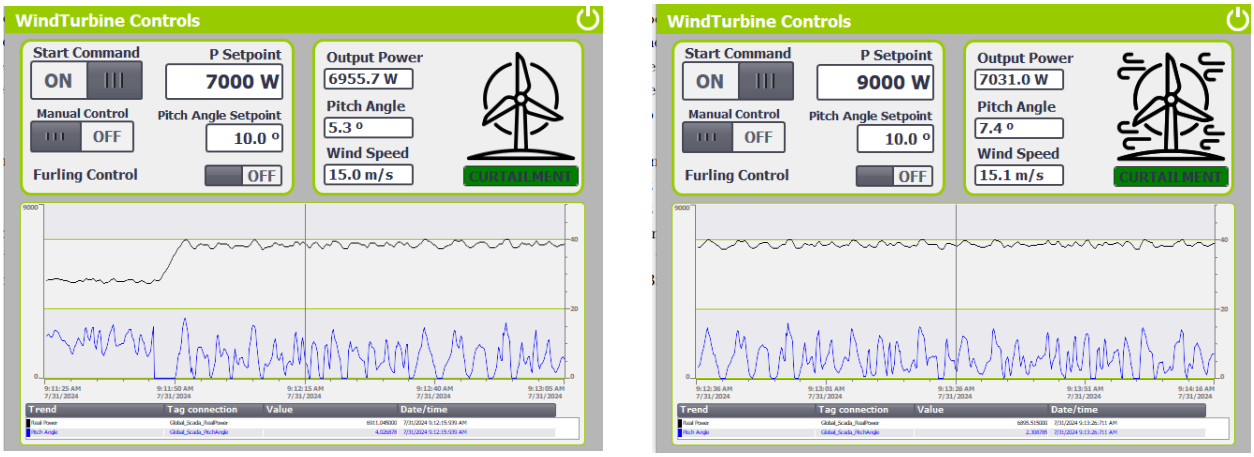


(d) Power set-point step change



(e) P-zero set-point

Figure 7.3: HMI WTG states (III)



(a) Wind speed power curtailment

(b) Power set-point curtailment

Figure 7.4: HMI WTG states (IV)

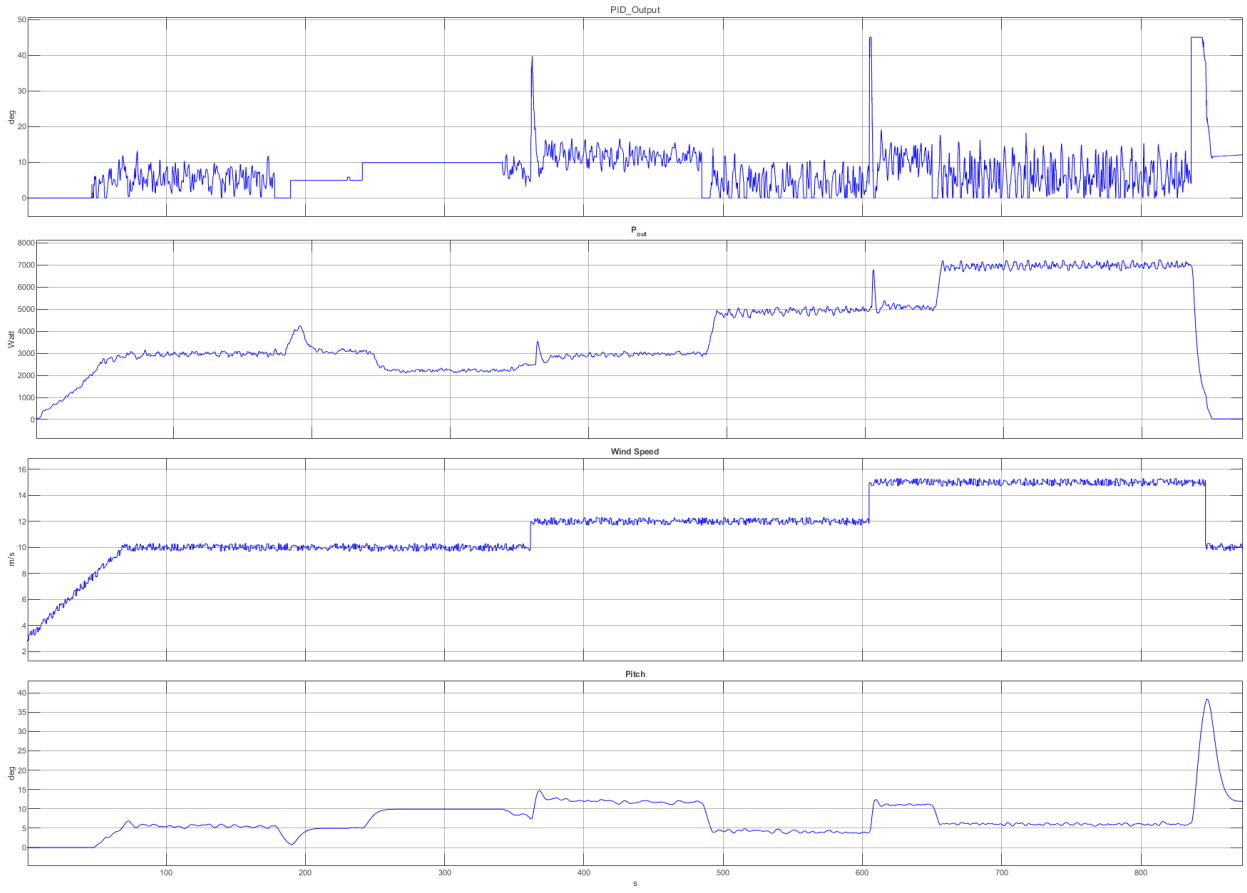


Figure 7.5: Simulation of the HIL WT control

Chapter 8

Conclusions and future works

During the definition of the model's constraints it was evident of the heavy correlation between the variables of wind speed, rotor speed, blade pitch and generator DC current by means of the power coefficient in a wind-turbine. From the equations defined it was validated that the only independent variables of the model were v and θ, so it was obvious that the higher the step change in v or θ the higher the error in ω, propagating through to the rest of the calculated variables. In normal operation, the power coefficient $C_{p,1}$ would decrement whilst the wind speed increases. Also, a turning point was detected for the calculation of C_p at wind speeds lower than the cut in speed of $3m/s$, where this variable saturates at $C_{p_{max}}$ and start trending towards 0 for a decrease in wind speed. The singularity identified in Chapter 5 was caused after entering this point of operation that allows the calculation of C_p to reach 0. The singularity was also reached when a lower saturation limit of ω was exceeded, creating an increment in the error which would propagate to I_{dc} and, thus, end in having an unrecoverable de-synchornized calculation of ω, and enabling C_p to fall under $C_{p_{min}}$. This state of operation could alternatively be fixed by implementing a hybrid control model. A switch case scenario could be used to reset and synchronize the integrators at this turning point when changing from a discrete to continuous state.

The implementation of OPC UA for establishing the communication between the simulated model and the controller is of particular interest for the technological industry due to its versatility and flexibility. One of the advantages of using an OPC-UA server is that it facilitates the definition of the points list of variables to be exchanged between server and client. In this case, only two variables were exchanged, but the solution may be easily modified to include a real analogue wind sensor such as an anemometer connected to one of the analog inputs of the PLC. The analog input given by the anemometer can then be processed by transforming the integer value translated by the Analog Input Module from the Electrical 4-20mA or 0-10V measurement into a Real value measurement. This value can then be published through the OPC-UA server and introduced into the simulated model as a real-time variable for the simulated model, expanding the Hardware-in-the-loop solution. This expansion can easily be accomplished with the newer version of MATLAB R2024a, which

includes OPCUA specific Simulink blocks that are easily configurable and would substitute the Interpreted MATLAB function implemented in the current solution.

The modular design of the solution enables for the implementation and testing of different controllers running on the PLC controller. Also, the system controller was not fully optimized to ensure full-power output or adjusted for more robust responses to wind variations. So, future works may expand on the implementation of more complex control algorithms and use the tuned PID controller as a base for comparing their performance. As the system control is encapsulated in the PLC controller, a feature could be added to the HMI to allow the operator to shift between different types of control algorithms to evidence their different influence over the model. In addition, one could also add different types of operational controls to the simulation, such as a ramp-up/ramp-down active power reference for smooth start-up and shutdown procedures.

In summary, the full development of the intended objective was achieved where all the simulation environments may coexist in one terminal, facilitating the implementation and expansion of the solution. The design of the HMI is believed to be simple as very little knowledge of the development environments is needed for its operation, and all options and controls are encapsulated in one window. However, a small guide would be useful in order to successfully set up the simulation environment for academic and educational purposes and demonstrations.

Bibliography

M. E. Gonzalez, F. J. Vázquez, F. Morilla, and D. M. Díaz. Modelo matemático y modos de operación de un aerogenerador de velocidad variable. *IX Congreso Internacional sobre Innovación y Desarrollo Tecnológico*, 9 2011. URL http://www.dia.uned.es/~fmorilla/Publicaciones/2011_%20Aerogenerador_CIINDET2011.pdf.

C. Dirscherl, C. Hackl, and K. Schechner. *Modellierung und Regelung von modernen Windkraftanlagen: Eine Einführung*, pages 1540–1614. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-642-30096-7. doi: 10.1007/978-3-642-30096-7_24. URL https://doi.org/10.1007/978-3-642-30096-7_24.

Y. Ledeneva, R. A. García Hernández, and A. Gelbukh. *Automatic Estimation of Parameters of Complex Fuzzy Control Systems*. 10 2008. ISBN 978-953-7619-20-6. doi: 10.5772/6268.

C. A. Walford. Wind turbine reliability :understanding and minimizing wind turbine operation and maintenance costs. 3 2006. doi: 10.2172/882048. URL https://www.osti.gov/biblio/882048.

T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi. *Wind Energy Handbook*. John Wiley & Sons, 2001. ISBN 9780471489979. URL https://books.google.es/books?id=4UYm893y-34C.

T. Ackermann. *Wind Power in Power Systems*. Wiley, 2005. ISBN 9780470012673. URL https://books.google.es/books?id=f7MDGCiOgiMC.

F. Pizaña and Jorge H. Sobre la ley de betz. 10 2015. doi: 10.13140/RG.2.2.20972.46727.

S. Reisch, G. Jacobs, D. Bosse, and D. Matzke. Challenges and opportunities of full size nacelle testing of wind turbine generators. In *The Proceedings of the JSME international conference on motion and power transmissions 2017*, pages 10–01. The Japan Society of Mechanical Engineers, 2017.

S. Nabi, M. Balike, J. Allen, and K. Rzemien. An overview of hardware-in-the-loop testing systems at visteon. 2004.

J. Millitzer, D. Mayer, C. Henke, T. Jersch, C. Tamm, J. Michael, and C. Ranisch. Recent developments in hardware-in-the-loop testing. In *Model Validation and Uncertainty Quantification, Volume 3: Proceedings of the 36th IMAC, A Conference and Exposition on Structural Dynamics 2018*, pages 65–73. Springer, 2019.

F. Mihalič, M. Truntič, and A. Hren. Hardware-in-the-loop simulations: A historical overview of engineering challenges. *Electronics*, 11(15):2462, 2022.

M. Mikati, M. Santos, and C. Armenta. Modelado y simulación de un sistema conjunto de energía solar y eólica para analizar su dependencia de la red eléctrica. *Revista Iberoamericana de Automática e Informática industrial*, 9(3):267–281, jul. 2012. doi: 10.1016/j.riai.2012.05.010. URL https://polipapers.upv.es/index.php/RIAI/article/view/9597.

M. Mikati, M. Santos, and C. Armenta. Electric grid dependence on the configuration of a small-scale wind and solar power hybrid system. *Renewable Energy*, 57:587–593, 2013. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2013.02.018. URL https://www.sciencedirect.com/science/article/pii/S0960148113001274.

A. Martinez, J. E. Sierra, and L. Santos M., Leija. Entorno PIL para la validación de controladores de turbinas eólicas basados en IEC-61131. *XLIV Jornadas de Automática*, pages 807–812, 2023. doi: 10.17979/spudc.9788497498609.807.

M.J. Khan and M.T. Iqbal. Analysis of a small wind-hydrogen stand-alone hybrid energy system. *Applied Energy*, 86(11):2429–2442, 2009. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2008.10.024. URL https://www.sciencedirect.com/science/article/pii/S0306261908002699.

M. E. Abdallah, O. M. Arafa, A. Shaltot, and G. A. A. Aziz. Wind turbine emulation using permanent magnet synchronous motor. *Journal of Electrical Systems and Information Technology*, 5(2):121–134, 2018. ISSN 2314-7172. doi: https://doi.org/10.1016/j.jesit.2018.03.005. URL https://www.sciencedirect.com/science/article/pii/S231471721830031X.

T Thiringer and J Linders. Control by variable rotor speed of a fixed-pitch wind turbine operating in a wide speed range. *IEEE Transactions on Energy Conversion (Institute of Electrical and Electronics Engineers); (United States)*, 8:3, 9 1993. ISSN 0885-8969. doi: 10.1109/60.257068. URL https://www.osti.gov/biblio/5682015.

H. Voltolini, M.H. Granza, J. Ivanqui, and R. Carlson. Modeling and simulation of the wind turbine emulator using induction motor driven by torque control inverter. pages 1–6, 11 2012. ISBN 978-1-4673-2412-0. doi: 10.1109/INDUSCON.2012.6453399.

E. Audierne, J. Elizondo, L. Bergami, H. Ibarra, and O. Probst. Analysis of the furling behavior of small wind turbines. *Applied Energy*, 87(7):2278–2292, 2010. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2009.11.019. URL `https://www.sciencedirect.com/science/article/pii/S0306261909005066`.

H. Berger. *Automating with SIMATIC S7-1500: configuring, programming and testing with STEP 7 Professional*. John Wiley & Sons, 2014.

P. Ashar, S. Devadas, and A. R. Newton. *Sequential logic synthesis*. Springer Science & Business Media, 1992.

M. Schleipen, S. Gilani, T. Bischoff, and J. Pfrommer. Opc ua & industrie 4.0 - enabling technology with high diversity and variability. *Procedia CIRP*, 57:315–320, 2016. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2016.11.055. URL `https://www.sciencedirect.com/science/article/pii/S2212827116312094`. Factories of the Future in the digital environment - Proceedings of the 49th CIRP Conference on Manufacturing Systems.

S. Lee and M. Sung. Opc-ua agent for legacy programmable logic controllers. *Applied Sciences*, 12(17), 2022. ISSN 2076-3417. doi: 10.3390/app12178859. URL `https://www.mdpi.com/2076-3417/12/17/8859`.

M. Kawazoe Aguilera, W. Chen, and S. Toueg. Heartbeat: A timeout-free failure detector for quiescent reliable communication. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms*, pages 126–140, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69600-1.

J. Laks, L. Pao, A. Wright, N. Kelley, and B. Jonkman. The use of preview wind measurements for blade pitch control. *Mechatronics*, 21(4):668–681, 2011.

F. Dunne. *Optimizing blade pitch control of wind turbines with preview measurements of the wind*. PhD thesis, University of Colorado at Boulder, 2016.

M. Elsisi, M. Tran, K. Mahmoud, M. Lehtonen, and M. M. F. Darwish. Robust design of anfis-based blade pitch controller for wind energy conversion systems against wind speed fluctuations. *IEEE Access*, 9:37894–37904, 2021. doi: 10.1109/ACCESS.2021.3063053.

B. Fitzgerald, A. Staino, and B. Basu. Wavelet-based individual blade pitch control for vibration control of wind turbine blades. *Structural Control and Health Monitoring*, 26 (1):e2284, 2019.

H. M. Hasanien. Design optimization of pid controller in automatic voltage regulator system using taguchi combined genetic algorithm method. *IEEE Systems Journal*, 7(4):825–831, 2013. doi: 10.1109/JSYST.2012.2219912.

J. B. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *ASME Transactions*, 64:759–768, 1942.

J. W.S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

Appendix A

Read_OPC_Func

```
function [x] = Read_OPC_Func_HB(y)
% variables
persistent init_Server;
persistent init_Nodes;
persistent uaClient;
persistent Var_Node_In;
persistent Var_Node_Out;
persistent Var_Node_Hb;
persistent testVal;
persistent heartbeatClient;
%initialize variables
if (isempty(init_Server))
 testVal = 0;
 init_Server = 0;
 init_Nodes = 0;
 heartbeatClient = 0;
end
%OPC UA server (PLC) address and connecting client (Simulink)
   to the server
if init_Server == 0
 init_Server = 1;
 uaClient = opcua('192.168.20.1',4840);
 connect(uaClient);
end
%define variable nodes in the server
if uaClient.isConnected == 1 && init_Nodes == 0
 init_Nodes = 1;
```

```
27  %find DB "OpcInterface" of the server
28  DB_Node = findNodeByName(uaClient.Namespace, 'OpcInterface', '
      -once');
29  %find variables "fanForce" and "currentAngle" in the DB "
      OpcInterface"
30  Var_Node_In = findNodeByName(DB_Node, 'pitchAngle', '-once');
31  Var_Node_Out = findNodeByName(DB_Node, 'currentPower', '-once'
      );
32  Var_Node_Hb = findNodeByName(DB_Node, 'heartbeat', '-once');
33 end
34 %read and write variables of the server
35 if uaClient.isConnected == 1 && init_Nodes == 1
36  %read "fanForce" value from server and store in "val"
37  [val, ~, ~] = readValue(uaClient, Var_Node_In);
38  %assign input y of the function to "currentAngle" variable
39  writeValue(uaClient, Var_Node_Out, y);
40  %assign "val" to variable "testVal"
41  testVal = val;
42  %send heartbeat
43  heartbeatClient=heartbeatClient+1;
44  writeValue(uaClient, Var_Node_Hb, heartbeatClient)
45 end
46 %assign "fanForce" ("testVal") value to the output x of the
      function
47 x = double(testVal);
48 end
```

Appendix B

PLC program

SCADA [FC1]

SCADA Properties							
General							
Name	SCADA	**Number**	1	**Type**	FC	**Language**	SCL
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
SCADA	Void		

```
0001 //Write to SCADA
0002 "Global".Scada.PitchAngle := "OpcInterface".pitchAngle;
0003 "Global".Scada.RealPower := LREAL_TO_REAL("OpcInterface".currentPower);
0004 "Global".Scada.WindSpeed := "OpcInterface".windSpeed;
0005
0006 //Read from SCADA
0007 "Global".ControllerOpc.manualOutputEnable := "Global".Scada.PitchManCtrl;
0008 "Global".ControllerOpc.manualOutputPercentage := REAL_TO_INT("Global".Scada.PitchCtrlSetPoint * 100 / "Global".WtgPara-
     meters.MAX_ANGLE);
0009 "Global".ControllerOpc.powerSetpoint := REAL_TO_INT("Global".Scada.ActivePowerSetpoint);
0010 "OpcInterface".furlingEnable := "Global".Scada.FurlingCtrlEn;
0011
0012 //WTG State Machine
0013 IF "Global".CommsFlagError = FALSE THEN
0014   IF "Global".Scada.WtgEStop = FALSE THEN
0015     IF "Global".Scada.WtgStartStopCmd = TRUE THEN
0016       IF "Global".Scada.PitchManCtrl = TRUE THEN
0017         "Global".Scada.WtgStatus := 5;
0018       ELSE
0019         IF "Global".Scada.WindSpeed >= "Global".WtgParameters.NOM_SPEED THEN
0020           "Global".Scada.WtgStatus := 4;
0021         ELSE
0022           "Global".Scada.WtgStatus := 3;
0023         END_IF;
0024       END_IF;
0025
0026     ELSE
0027       "Global".Scada.WtgStatus := 2;
0028     END_IF;
0029   ELSE
0030     "Global".Scada.WtgStatus := 1;
0031   END_IF;
0032 ELSE
0033   "Global".Scada.WtgStatus := 0;
0034 END_IF;
```

Symbol	Address	Type	Comment
"Global".CommsFlagError		Bool	
"Global".ControllerOpc.manualOut-putEnable		Bool	
"Global".ControllerOpc.manualOut-putPercentage		Int	
"Global".ControllerOpc.powerSet-point		Int	
"Global".Scada.ActivePowerSetpoint		Real	Setpoint input for Power Output Reference
"Global".Scada.FurlingCtrlEn		Bool	0: Disable, 1: Enable
"Global".Scada.PitchAngle		Real	Real-time pitch angle feedback
"Global".Scada.PitchCtrlSetPoint		Real	Setpoint input for Manual Pitch Control
"Global".Scada.PitchManCtrl		Bool	0: Auto, 1: Manual
"Global".Scada.RealPower		Real	Real-time electrical power output from the Windturbine
"Global".Scada.WindSpeed		Real	Real-time wind speed measurement
"Global".Scada.WtgEStop		Bool	0: Disable, 1: Enable
"Global".Scada.WtgStartStopCmd		Bool	0: Stop, 1: Start
"Global".Scada.WtgStatus		Int	0: NO COMM, 1: EMERG, 2: PAUSE, 3: RUN, 4: CURTAILED, 5: MANUAL
"Global".WtgParameters.MAX_ANGLE		Real	
"Global".WtgParameters.NOM_SPEED		Real	
"OpcInterface".currentPower		LReal	
"OpcInterface".furlingEnable		Bool	
"OpcInterface".pitchAngle		Real	
"OpcInterface".windSpeed		Real	

Cyclic interrupt [OB30]

Cyclic interrupt Properties							
General							
Name	Cyclic interrupt	**Number**	30	**Type**	OB	**Language**	SCL
Numbering	Automatic						
Information							
Title		**Author**		**Comment**		**Family**	
Version	0.1	**User-defined ID**					

Name	Data type	Default value	Comment
▼ Input			
Initial_Call	Bool		Initial call of this OB
Event_Count	Int		Events discarded
▼ Temp			
tempOutputPercentage	Real		
Constant			

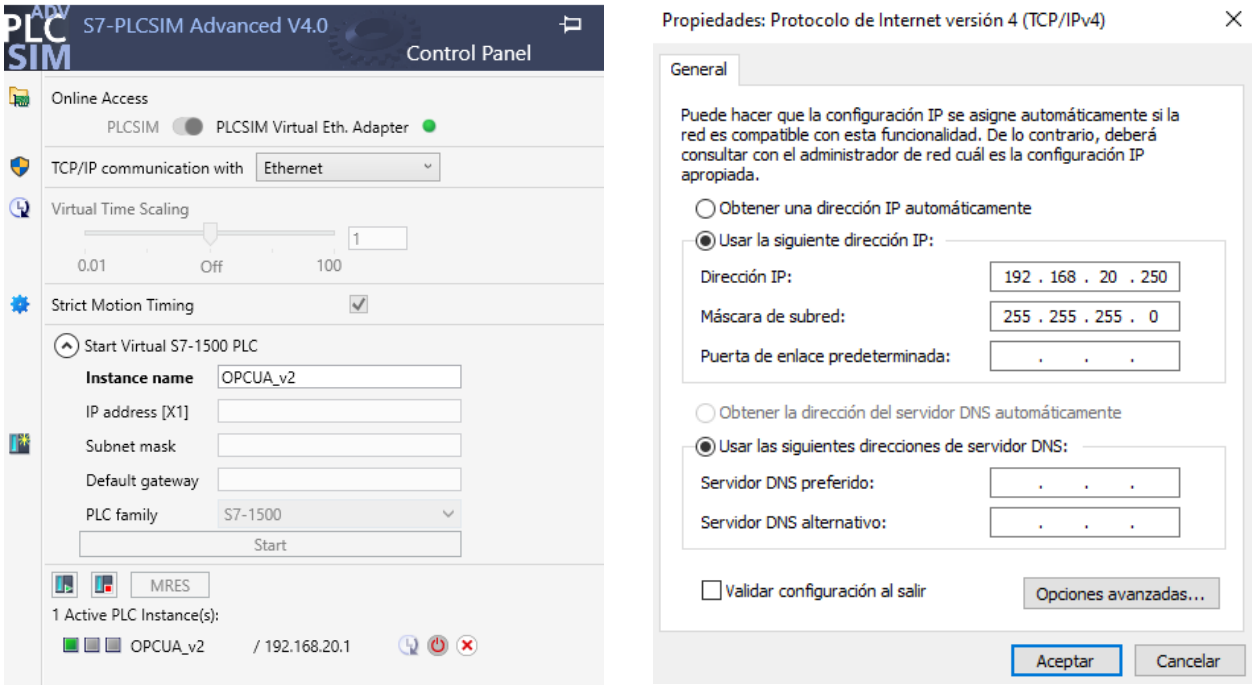
```
0001 "InstPIDCompactOPC"(Setpoint:="Global".ControllerOpc.powerSetpoint,
0002           Input:=LREAL_TO_REAL("OpcInterface".currentPower),
0003           ManualEnable:="Global".ControllerOpc.manualOutputEnable,
0004           ManualValue:="Global".ControllerOpc.manualOutputPercentage,
0005           ModeActivate:=TRUE,
0006           Output=>#tempOutputPercentage,
0007           Mode:="Global".ControllerOpc.operatingMode);
0008
0009 //Scaling the output of PID Controller (0..100%) to maximum angle (0..90°)
0010 "OpcInterface".pitchAngle := #tempOutputPercentage * "Global".WtgParameters.MAX_ANGLE / 100;
0011
0012 IF "OpcInterface".heartbeat = 0 THEN
0013   "Global".heartbeatServer := 0;
0014 ELSIF
0015   "Global".heartbeatServer < "OpcInterface".heartbeat THEN
0016   "Global".heartbeatServer := "OpcInterface".heartbeat;
0017   "Global".errCounter := 0;
0018   "Global".CommsFlagError := FALSE;
0019 ELSE
0020   "Global".errCounter += 1;
0021   IF "Global".errCounter > 9 THEN
0022     "Global".CommsFlagError := TRUE;
0023   END_IF;
0024
0025 END_IF;
0026
0027
```

Symbol	Address	Type	Comment
"Global".CommsFlagError		Bool	
"Global".ControllerOpc.manualOutputEnable		Bool	
"Global".ControllerOpc.manualOutputPercentage		Int	
"Global".ControllerOpc.operatingMode		Int	
"Global".ControllerOpc.powerSetpoint		Int	
"Global".errCounter		Int	
"Global".heartbeatServer		LInt	
"Global".WtgParameters.MAX_ANGLE		Real	
"OpcInterface".currentPower		LReal	
"OpcInterface".heartbeat		LInt	
"OpcInterface".pitchAngle		Real	
#tempOutputPercentage		Real	

Appendix C

Network Architecure

The following Annex gives an overview of the configured Network Architecture for the application to run in the same terminal.



(a) PLCSIM Advanced Instance deployment (b) Virtual Ethernet Adapter Properties

Figure C.1: Network Architecture Setup

Appendix D

Simulink Model

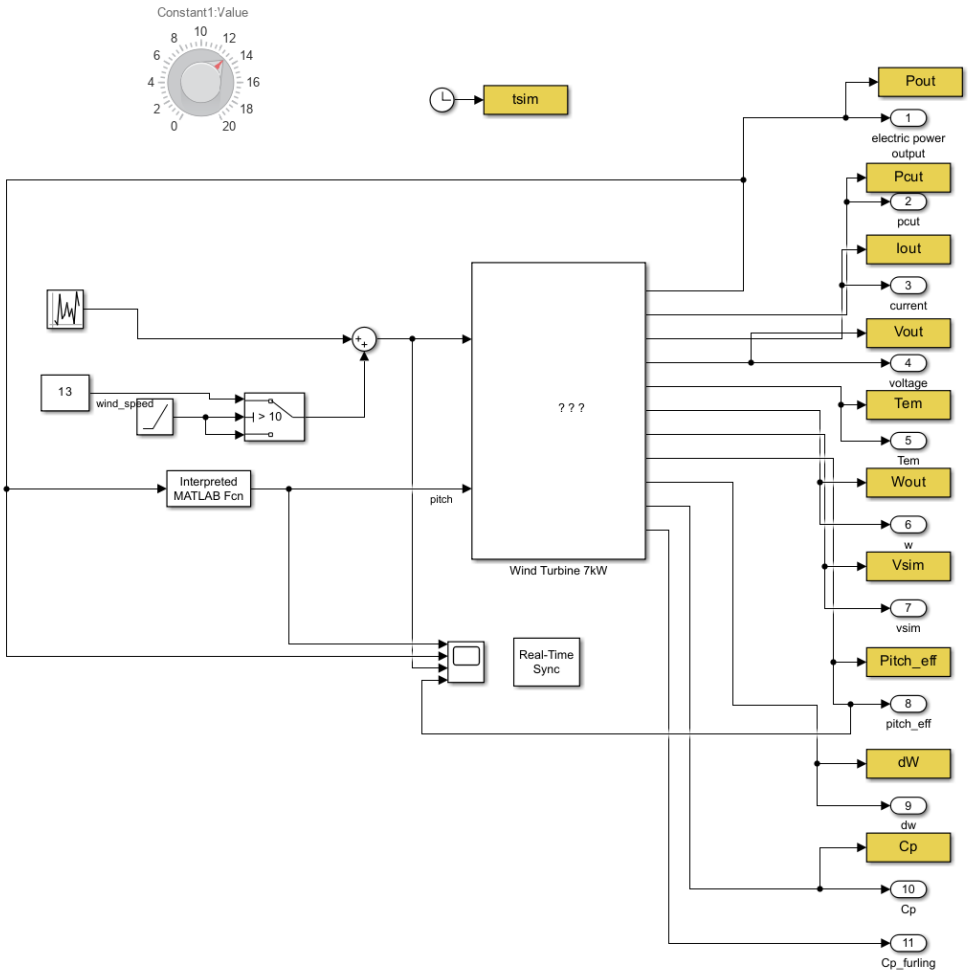


Figure D.1: Simulink Model