A FAST.Farm and MATLAB/Simulink Interface for Wind Farm Control Design

Coen-Jan Smits

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

A FAST.Farm and MATLAB/Simulink Interface for Wind Farm Control Design

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Coen-Jan Smits

19 June 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Copyright © Delft Center for Systems and Control (DCSC)
All rights reserved.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

A FAST.Farm and MATLAB/Simulink Interface for Wind Farm Control Design

by

Coen-Jan Smits

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: 19 June 2023

Supervisor(s):

dr.ir. Riccardo Ferrari

ir. Jean Gonzalez Silva

Reader(s):

dr.ir. Nitin Myers

Abstract

Increasing the efficiency of wind farms is important for speeding up the transition from fossil fuels to renewable energy sources. Current wind farm control relies on maximization of power generation of individual turbines. However, research has demonstrated that plant-wide wind farm control could optimize the performance of a wind farm. Wind farm simulation tools are crucial in designing, testing, and validating wind farm controllers. Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) is a recently developed multi-physics engineering tool for modeling power performance and structural loads by solving the aero-hydro-servoelastic dynamics of each individual turbine within a farm. FAST.Farm aims to balance the need for accurate modeling of the relevant physics while maintaining low computational costs. However, designing controllers in FAST.Farm lacks flexibility and interactivity compared with MATLAB/Simulink. The capabilities of FAST.Farm for control design purposes can be extended through a co-simulation with MATLAB/Simulink. Therefore this thesis presents a FAST.Farm and MATLAB/Simulink interface. Consequently, this interface was used to implement and simulate wind farm controllers in FAST.Farm. FAST.Farm and MATLAB/Simulink are coupled by linking the individual Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST) instances in FAST.Farm to MATLAB with the use of an Message Passing Interface (MPI) and MATLAB Executable (MEX) functions. An Active Power Controller (APC) was implemented and simulated in FAST.Farm with the use of this interface. The APC responds to grid requirements through the control of wind farm power output. Comparing FAST.Farm and Simulator fOr Wind Farm Applications (SOWFA) simulation results of the APC shows that FAST.Farm reduces the computation time for a 10-minute simulation from 24 hours to 15 minutes, with little detriment to the accuracy of the simulation results. Although SOWFA remains the preferred validation tool, the FAST.Farm and MATLAB/Simulink interface supports developing and accelerates testing advanced closed-loop control at the wind turbine and wind farm levels.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This is it, the end of my time being a student. I am very grateful for all the people I have met, the opportunities I have got, and the lessons I have learned. I would not be the person I am today without the people around me.

I am very proud to present to you my thesis. I am honored to have had the opportunity to present my thesis at the DeepWind conference in Trondheim. It was a great opportunity to engage with other researchers and discuss the current developments in the offshore wind industry. Throughout this research, I have learnt a lot, and I am deeply thankful to all who have supported me during this process. I would like to thank my supervisors Riccardo Ferrari and Jean Gonzalez Silva for their assistance during the time span of this thesis. I would especially like to thank Jean for all his time and enthusiasm during our weekly meetings. It was common for us to spend two hours discussing my thesis work, this really helped me make progress in my thesis. In addition, I would like to thank Valentin Chabaud, without your help I would not be able to design the interface.

Finally, I would like to thank my parents, family, friends, and girlfriend for their support over the last years. They were always there for me.

Delft, University of Technology
19 June 2023

Coen-Jan Smits

“The more I learn, the less I know”

Chapter 1

Introduction

1-1 Introduction to Research

There is a consensus in the scientific community that we need to transition from fossil fuels to renewable energy sources quickly in order to attenuate the rise of the average temperature on earth [8, 9]. Large onshore and offshore wind farms are being built to increase the share of renewable energy in the current power systems. The share of wind energy in the total energy production in Europe was around 16% in 2020, with the expectation that this will rise to 25% in 2025 [10, 11]. To reach the ambitious goal of decarbonization, for example, the Dutch government recently assigned additional zones in the North Sea for offshore wind farm development [12].

The development of wind farm controllers progresses as well, as the number and size of wind farms increase, but is still at the research stage. Current practice defers control to the turbine level, where individual wind turbine controllers rely on the concept of greedy control. In greedy control, each wind turbine in a wind farm optimizes its own power production, without sharing information about the aerodynamic coupling with other turbines in the farm. However, research has demonstrated that applying these greedy controllers across the farm is sub-optimal in terms of overall wind farm performance [13, 14]. Current research on wind farm control aims to develop controllers which take into account the wake interactions within a wind farm and the performance of individual turbines [15, 16]. These controllers aim to minimize the costs of wind energy by considering, among others, structural degradation and grid integration objectives. For designing these controllers, it is important that the relevant physics of a wind farm are accurately modeled so that the estimated turbine performance corresponds to reality.

There already exist several engineering tools for modeling the physics of a wind farm. These tools are used for wind farm control synthesis and validation. A distinction can be made between low-, medium- and high-fidelity tools. Low-fidelity models, e.g. FLow Redirection and Induction in Steady State (FLORIS) [17], are based on parametric flow models. These models estimate only quasi-steady wake characteristics for wind farm layout optimization

and cannot predict dynamic flow development. On the other hand, high-fidelity tools like Simulator fOr Wind Farm Applications (SOWFA) [18] estimate the flow field in detail by resolving the governing equations (Navier-Stokes) in three-dimensional (3D) space. However, high-fidelity tools are impractical in use because of the high computation time. Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) [19] is a recently developed medium-fidelity multiphysics engineering tool by National Renewable Energy Laboratory (NREL) which aims to balance the need for accurate modeling of the relevant physics while maintaining low computational cost. This tool uses a simplified (two-dimensional) version of the governing equations in which the Navier-Stokes (NS) equations are approximated with a thin shear layer approximation that is less computationally expensive than high-fidelity tools [6, 20]. Therefore, FAST.Farm shows to be a promising tool for designing wind farm controllers and simulating wind farm behavior [21].

The current procedure of wind farm control synthesis in FAST.Farm is by creating wind farm controllers in Fortran language. These wind farm controllers are called Super Controllers (SC), which are often compiled in a Dynamic Link Library (DLL). The SC-DLL is essentially identical to the SC available in SOWFA [22]. An SC-DLL is used in conjunction with individual wind turbine controllers designed in the style of the DISCON DLL [23]. A drawback of creating a wind farm controller in a Fortran DLL is that Fortran is a compiled language made for performance and lacks flexibility and interactivity compared with MATLAB/Simulink. MATLAB/Simulink is a widespread tool in the control community in the academic world which has well-established toolboxes for designing high-level controllers. A MATLAB/Simulink interface with FAST.Farm would allow the wind community to easily implement wind farm control algorithms from the MATLAB platform, as well as extend those controllers with pre-built features.

The first contribution of this thesis is to present the first, to the best of the author's knowledge, interface between FAST.Farm and MATLAB/Simulink, including its creation and how to operate it. Such an interface extends the capability of FAST.Farm for control design purposes. This interface is realized by linking the individual turbine controllers in the FAST.Farm tool to MATLAB through an Message Passing Interface (MPI) and MATLAB Executable (MEX) functions. This interface supports a co-simulation between FAST.Farm and MATLAB/Simulink, making it able to run the controllers, at both turbine and wind farm levels, and simulation simultaneously. To validate the interface, the FAST.Farm simulation results of a yaw controller implemented in the MATLAB interface were compared with the simulation results of the same yaw controller implemented in a Fortran DLL. The second contribution of this thesis is the implementation and simulation of an Active Power Controller (APC) in FAST.Farm with the MATLAB interface. This APC responds to grid requirements through the control of farm power output. Consequently, the simulation results of this APC are compared with the simulation results of the same controller in SOWFA to illustrate the differences between the newly developed tool FAST.Farm and SOWFA. FAST.Farm is a promising tool in terms of computation time, simulating a 10-minute event in FAST.Farm took only 15 minutes instead of 24 hours in SOWFA. However, SOWFA is still preferred as validation tool, as flow field simulations of FAST.Farm could deviate over time with respect to SOWFA.

1-2 Research Questions and Contributions

The main research question at the start of this thesis was: Till what extend is it possible to develop an interface between FAST.Farm & MATLAB/Simulink. The contributions of this thesis are the creation of a FAST.Farm and MATLAB/Simulink interface, the implementation of wind farm controllers in this interface and a comparison of the simulation results of an APC simulated in FAST.Farm and SOWFA. The interface designed in this thesis can be used by other researchers, see GitHub for the source-code [24]. This interface enables researchers designing and testing wind farm controllers in FAST.Farm instead of SOWFA, thereby reducing the computation time for a 10-minute simulation from 24 hours to 15 minutes. The following three research questions will be answered during this thesis:

- **To what extent is it possible to develop an interface between FAST.Farm and MATLAB/Simulink?**

  A FAST.Farm and MATLAB/Simulink interface is realized by linking the DISCON DLLs in the Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST) modules of FAST.Farm to MATLAB with the use of an MPI and MEX functions. The interface is extended to Simulink with the use of System-Functions (S-Functions). Chapter 3 discusses the operation of FAST.Farm, and chapter 4 outlines the creation of the interface. The interface is validated by implementing and simulating a yaw controller in both a Fortran DLL and in the MATLAB interface.

- **To what extent can this interface be used for designing controllers in MATLAB/Simulink and simulating controllers in FAST.Farm?**

  The designed interface offers the possibility to include a farm-level and a turbine-level controller. The turbine-level controller can be either the ROSCO/DTUWEC standard bladed style DLL, or manually designed in MATLAB/Simulink. Chapter 5 explains the implementation of controllers inside the interface. FAST.Farm and MATLAB/Simulink can exchange controller commands by exchanging the avrSWAP matrix.

- **How do the simulation results of an active power controller differ between FAST.Farm and SOWFA?**

  An APC has been designed (based on the papers of Silva et al. [16, 25]) and implemented in the FAST.Farm and MATLAB interface. A comparison between the simulation results of this APC simulated in FAST.Farm and SOWFA is outlined in chapter 6. FAST.Farm offers the possibility to include, e.g., tower vibrations. But, as FAST.Farm is a medium-fidelity simulation tool, wake dynamic estimations could deviate from SOWFA estimations. SOWFA is still preferred as validation tool.

The FAST.Farm and MATLAB/Simulink interface has already been presented at the DeepWind conference in Trondheim. In addition, a conference paper [26] has been submitted for publication.

1-3 Outline

The structure of this thesis is as follows. First, the operation of wind turbines, wind farm control methods, and wind farm simulation tools are outlined in chapter 2. Chapter 3 summarizes the working of FAST.Farm and explains the current implementation of wind farm controllers in FAST.Farm. Next, chapter 4 discusses the creation of the FAST.Farm and MATLAB/Simulink interface developed during this thesis. Subsequently, chapter 5 outlines the implementation and functionalities of the wind farm controllers simulated with this interface in FAST.Farm. The results of these simulations are showed and commented in chapter 6. Chapter 7 contains the discussion and the further research opportunities. Finally, chapter 8 concludes this thesis with a conclusion.

Chapter 2

Wind Farm Simulation and Control

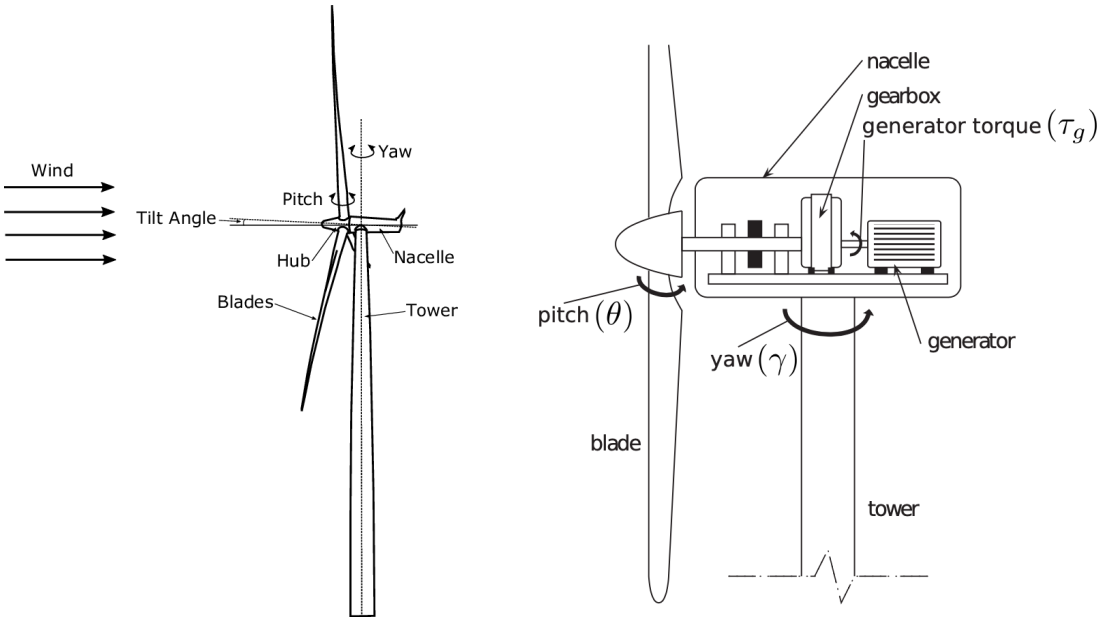
2-1 Introduction

This chapter provides background information on the operation of a wind turbine, the operation of wind farm controllers, and the various wind farm simulation tools. Designing wind farm controllers necessitates an understanding of the working of wind turbines. Consequently, simulation tools are essential for evaluating controller performance.

2-2 Operation of Wind Turbine

Wind turbines come in all shapes and sizes, depending on their use and location [27]. The upwind horizontal-axis wind turbine is the most used wind turbine. One of the main advantages of upwind horizontal-axis wind turbines is that the blades are always facing fully into the wind. The horizontal-axis wind turbine consists of a tower, a nacelle and a rotor (hub with blades), see Figure 2-1. In the nacelle, the rotational movements of the rotor are converted to electric power through a generator. A gearbox functions as a transmission between the rotor and the generator. Most operating horizontal-axis wind turbines have three degrees of freedom (control variables), the yaw angle, the blade pitch angle (actually three degrees of freedom, one for every blade) and the generator torque. The power generation of the turbine and the load distribution on the turbine can be controlled by changing these control variables. Some papers propose the use of tilt angle control and cone angle control [30]. Tilt and cone angle control could be used for wake steering. Active Flow Control (AFC) is an alternative to pitch control. With AFC the airflow over a particular section of the blade of a turbine is modified. This could reduce the dynamic loads on a turbine with less control effort [31]. Examples of AFC are trailing edge flaps and shape-changing blades. Tilt angle control, cone angle control and AFC are quite rare at the moment and more scientifically interesting control options.

- Blade pitch angle (θ): Ability to control the rotor blades with respect to the axis of rotation aligned with the blades. All blades can be individually controlled.



(a) General overview [28] **(b)** Main components and control variables [29]

Figure 2-1: Horizontal-axis upwind wind turbine.

- Yaw (γ): The rotation of the nacelle, with the axis of rotation aligned with the tower. The yaw angle is the angle between the incoming wind direction and the axial rotor axis.
- Generator torque (τ_g): The generator converts mechanical power into electrical power.
- Tilt angle: The nacelle can be tilted with respect to the horizontal axis of the nacelle.
- Cone angle: The cone angle is defined as the angle between the rotor plane and the blade axis.

During my research, I focused on upwind horizontal-axis wind turbines capable of controlling the blade pitch angle, the yaw angle and the generator torque. Horizontal-axis wind turbines are widely used in practice. Tilt angle control, cone angle and AFC are more scientifically interesting control options and have not been applied yet on large wind farms.

2-2-1 Power Production of Wind Turbine

The power production of large wind turbines is usually expressed in Megawatts (MW). The largest installed wind turbine in 2021 has a power capacity of over 14MW, according to [32]. Approximately 1,000 households can be supplied with electricity per MW of power. Keep in mind that the power capacity does not equal the power production (megawatt hour), the power production depends on the weather and control settings. So, a 14MW turbine with an average power production of 7 MW per hour could supply electricity for 7,000 households. Figure 2-2 shows a typical wind turbine power curve. Three regions can be distinguished in this curve. In region one, it is not cost-effective to 'turn on' (increase generator torque of)

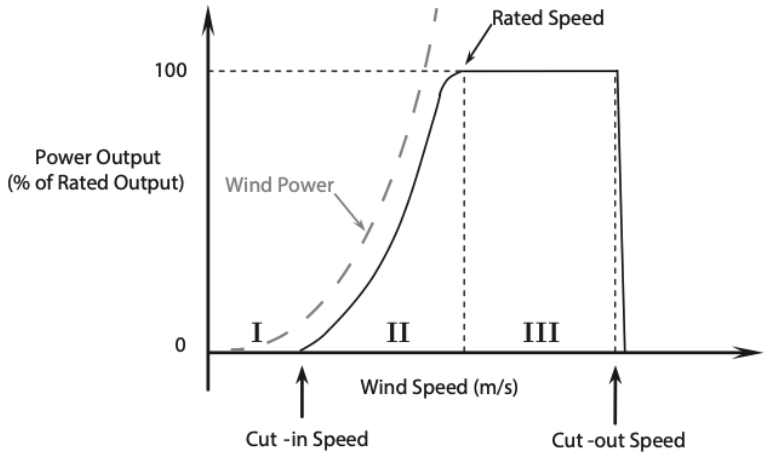


Figure 2-2: Typical wind turbine power curve [1].

the turbine, the wind speed is lower than the cut-in wind speed. In this region, the torque is set to zero so that the wind can be used to accelerate the rotor for start-up. In region two the turbine is 'turned on' and tries to fulfill the objective of the wind turbine controller (e.g. maximize the power production). In region three the turbine has reached its power production limit and is rated to a maximal power output (turbine's rated power). The power production of a wind turbine is limited to protect the electrical and mechanical components. At too high wind speeds wind turbines are turned off (fixed).

A turbine generates electricity by converting the rotational movements (kinetic energy) of the rotor into electrical power (electric energy) through a generator. The maximum kinetic power (in [W]) that is available in a flow with the size of the rotor disk is calculated as follows:

$$P_{\mathrm{w}} = \frac{1}{2}\rho A U^3, \tag{2-1}$$

where ρ is the air density in [kg/m^3], A is the rotor surface in [m^2] and U is the flow velocity in [m/s]. The power expression depends quadratic on the length of the rotor blades r, as the rotor surface is calculated as πr^2. So, extending the blades by a factor of two will raise the total available power by a factor of four. However, a wind turbine is not able to extract all the available power (kinetic energy) from the flow. The flow behind a wind turbine is required to have velocity. The Betz limit is the theoretical limit for energy extraction by a rotor [33]. The maximum amount of wind power that can be extracted by a wind turbine is given by:

$$P = C_{\mathrm{P}}(\theta, \lambda, \gamma)\frac{1}{2}\rho A U_\infty^3, \tag{2-2}$$

where $C_{\mathrm{P}}(\theta, \lambda, \gamma)$ is the dimensionless power coefficient, this is the ratio of generated power by the wind turbine to the available power in the wind (the Betz limit). U_∞ [m/s] is the free-stream wind velocity at the hub height. Based on literature [33], There are several approaches available for calculating C_P. One commonly used method to calculate C_P is by using a lookup-table containing the (turbine specific) relation between the Tip Speed Ratio (TSR) and blade pitch angle. In practice, the maximum power coefficient lies around 0.45 for horizontal-axis wind turbines [34]. The power production of a wind turbine can be maximized by maximizing

C_P, this corresponds to setting the rotor plane perpendicular to the incoming wind direction. In single turbine control this is called greedy control, maximizing power generation of a single wind turbine.

2-2-2 Turbine-Induced Forces

A turbine is subjected to alternating forces and to mean values of forces (ultimate load). Alternating forces occur for instance due to the rotation of the rotor. The mean value of forces is influenced by the aerodynamic forces working on the turbine. The frequency and mean value of the forces working on a turbine influence the lifespan of a wind turbine. A wind turbine could be subjected to a different set of forces, depending on the location and control settings. For example, offshore turbines are also subjected to hydrodynamic loads. The ultimate load working on a turbine can be reduced by reducing the thrust force. In general, by reducing the power capture of a turbine the thrust force is lowered. A balance should be struck between allowable loads and desired power generation. Active power control shows to be a solution to distribute (and so lower) the loadings along the wind turbines in a wind farm.

Gravitational loading, inertial loading and aerodynamical loading are the most dominant sources for the loading of upwind horizontal-axis wind turbines [35]. Gravitational loading is caused by the gravitational field of the Earth. A blade rotating downward and a blade rotating upward experience different forces, this causes a sinusoidal loading on the blades. Inertial loading occurs when the rotation speed of the wind turbine changes. The flow that passes the wind turbine causes aerodynamical loading. The aerodynamical loading on different wind turbines in a wind farm can vary greatly because of wake effects in the flow in the wind farm. The lifespan of a turbine depends on the magnitude and duration of these (alternating) loadings. Damage Equivalent Load (DEL) is a measure that can be used to quantify loading in order to compare different loading types [36]. Wind turbine and wind farm control could have as objective also to mitigate load effects. On a wind farm level, controls are mostly used to mitigate the effect of wake (aerodynamical loading) on downstream wind turbines. A possible problem in controlling a wind turbine can appear by operating in a non-designed operation point where a flow separation might occur characterizing a stall behavior. This can increase loads and might lead to the shutdown of the machine.

Stall Stall in wind turbines is referred to as the loss of lift force of a blade. A blade can lose its lift force due to changes in its aerodynamic behavior (flow separation). Stall results in lower power generation and could eventually stop the rotor from rotating. Stall has to be taken into account during the design process of a controller. A pitch offset helps to prevent that the wind turbine starts operating in the stall region. Stall will most likely occur during the derating of a turbine.

Thrust Force The thrust force represents the force a wind turbine exerts on the wind flowing through the rotor. Figure 2-3 presents the wind speed distribution and the trust force working on a wind turbine. The direction, magnitude and change over time of the thrust force give a good impression of the loadings on a turbine. The thrust force (in [N]) corresponds to the

amount of energy extracted from the flow:

$$F = C_{\mathrm{T}}(\theta, \lambda, \gamma)\frac{1}{2}\rho A U_{\infty}^{2}, \tag{2-3}$$

with U_{∞} [m/s] as the free-stream wind velocity at the hub height and $C_{\mathrm{T}}(\theta, \lambda, \gamma)$ as the dimensionless thrust force coefficient. C_T is a function of the tip-speed ratio, λ, blade pitch angle, θ, and yaw angle, γ. ρ is the air density in [kg/m^3] and A is the surface area of the rotor in [m^2]. The thrust force coefficient describes the force exerted by the turbine in the axial direction to the incoming momentum of the flow. A high thrust force coefficient corresponds to turbulent wake very close to the rotor [37]. Again, there are several approaches available for calculating the thrust force coefficient. One commonly used method to calculate C_{T} is by using a (turbine specific) lookup-table containing the relation between the TSR and blade pitch angle.

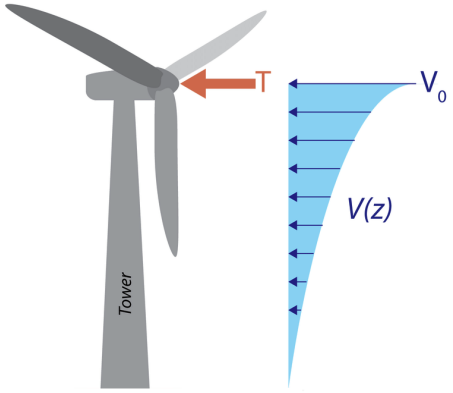


Figure 2-3: Simple wind speed distribution and trust force working on a wind turbine [2].

2-2-3 Effects of Wake

The extraction of energy from the flow results in changes in the wind flow (lower wind velocity) downstream of a wind turbine. The altered flow downstream a wind turbine is called the wake. Figure 2-4 illustrates how the (normally invisible) wake effects in a wind farm could look like. Upwind wind turbines can influence the performance of downwind wind turbines through their wake effects. Taking into account the wake effects in a wind farm in order to maintain a specific level of performance is the key objective of wind farm modelling and control. But which wake dynamics are important for a control-oriented wind farm model is still an open question [38]. The characteristics of a wake are space-, time-, and parameter-dependent. Space-dependent, the wake effects closer downstream are different from far downstream; Time-dependent, the turbine and surrounding flow change over time; And parameter-dependent, external variables influence the behavior of the wake. An external variable is for example the temperature of the flow. Multiple tools are available for modelling the wake effects in a flow, the next section outlines some available tools.

A wake has multiple typical characteristics and can occur in different ways. Below is a list of several characteristics and causes [38].

Figure 2-4: Picture of wakes in an offshore wind farm [3].

- Wind velocity deficit: A wind turbine extracts energy from the flow, the velocity of the wind in the wake decreases.
- Increased turbulence intensity: The rotation of the blades could for example increase the turbulence intensity.
- Wake recovery: The wind velocity downwind of a wind turbine could recover to the free-stream velocity due to mixing.
- Wake meandering: A wake shows horizontal and vertical oscillations over time. The wake is not fixed to a specific position or shape, see Figure 2-4.
- Wake expansion: A wake expands over time, see Figure 2-4. Wakes expanse in horizontal and vertical direction.
- Wake deflection: A wake could diverge in the latitudinal direction from the rotor centre, because of, e.g., a yawed turbine or blade rotations.
- Wake skewing: The Coriolis forces cause a wind-direction change with height in the atmospheric boundary layer, this is called veer. A wake could skew (have a slope) because of veer.
- Vertical wind sheer: The wind velocity differs at different heights. Ground friction has as effect that the wind speed increases with height. This has as effect that the wake properties change with height.

2-3 Wind Farm Control

Current wind farm control relies on the concept of greedy control. However, research has demonstrated that the overall performance of a wind farm can be increased by taking into account the wake interactions between wind turbines in a wind farm [14,38–40]. This section

addresses the use of wind farm controls for optimizing wind farm power and structural load performance.

2-3-1 Greedy Control

Current control of wind turbines in a wind farm relies on the concept of greedy control. Greedy control focuses on individually optimizing the power production of a wind turbine. Greedy control does not take into account the aerodynamic coupling (wake effects) between wind turbines in a wind farm. In general, a wind turbine can extract the most power from the flow by placing the rotor perpendicular to the flow field. Maximizing C_P (see section 2-2-1) is the primary objective of a greedy controller [13]. A greedy controller aims to steer C_{P} to its highest aerodynamic efficiency, C_{Pmax}. The blade pitch angle is straightforward to control, this angle can be maintained at the optimal efficiency point quite easy. The TSR on the other hand depends on the incoming wind speed and the angular velocity of the rotor. The incoming wind speed is constantly changing. Generator torque control can be used to change the angular velocity of the turbine rotor, and so change the TSR. Notice that maximizing C_{P} holds as long as the wind speed is higher than the cut-in wind speed and the turbine's rated generator speed is not reached (region two of power curve). The torque is set to zero when the wind speed is lower than the cut-in wind speed. When the turbine's rated power is reached, pitch control or yaw misalignment can be applied to limit the generator speed of the turbine.

A wind farm operating with greedy controllers in a decentralized manner is less complex to implement than a wind farm controller considering wind turbine interactions. But, the wind farm power production could be higher with a wind farm controller than operating with individual greedy controllers. For the next sections/chapters, I define 'greedy control' as the concept in which all the wind turbines in a wind farm operate with greedy controllers in a completely decentralized manner.

2-3-2 Motivation for Wind Farm Control

Applying wind farm control could serve different objectives. In general, the goal of wind farm control is to minimize the cost of wind energy. This main objective can be translated into three technical objectives, namely minimizing structural degradation, maximizing power production, and active power control. Depending on the objective, different wind farm controllers can be designed. In general, the aim of applying wind farm control is not to pursue one objective but increase the power performance while at the same time minimising structural degradation.

Power Production Maximization Power production maximization is about maximizing the power produced by the wind farm. With the aid of a wind farm controller, the wind farm power production can be increased in comparison to greedy control. A downside of power production maximization is that the turbines in a farm are subjected to relatively high structural degradation because of the interaction among themselves. Therefore, maximizing power does not mean minimizing costs because of increasing maintenance costs.

Load Minimization Load minimization is about minimizing the loads that act on a wind turbine. Turbines experience different types of loads, such as aerodynamical loading, gravitational loading, and inertial loading. Wind farm controllers can be used to distribute the load, e.g. thrust forces, equally between all turbines. Load minimization is important for reducing maintenance costs, and so minimizing the cost of wind energy.

Active Power Control Active power control provides grid services in order to improve the quality of wind energy. Examples of active power control are frequency control and power reference tracking. Active power control algorithms could have secondary control objectives such as load minimization. Active power control can be used to minimize the maintenance cost of turbines and on the other side provide a stable energy supply.

2-3-3 Wind Farm Control Methods

Literature proposes three notable wind farm control methods for wind turbines with three degrees of freedom, namely Axial Induction Control (AIC), Wake Redirection Control (WRC) and Wake Mixing (WM) [41]. These control methods aim to mitigate the effect of aerodynamic coupling within wind farms. Several papers propose to use floating wind farms [42,43], floating wind farms would allow repositioning of the turbines in a wind farm. Turbine repositioning is a notable fourth wind farm control method, but not jet applied on large offshore wind farms. Therefore, turbine repositioning is not discussed in this thesis report.

Axial Induction Control AIC is a wind farm control method which was in first place devised to maximize power production. AIC works by changing the axial induction of upwind turbines away from optimal settings in order to mitigate the downstream wake effects and so increase the power generation of downwind wind turbines. In general, changing the axial induction setting away from optimal settings means that an upwind wind turbine is subjected to less aerodynamic loading and so produces less power. For instance, see the work of Annoni et al. [44], for an analysis of AIC in two different simulation environments. Figure 2-5 visualise the effect of AIC on wake mitigation. The axial induction factor can be adjusted by changing the blade pitch angles or generator torque but is also influenced by the yaw angle. The axial induction factor is the ratio of the difference between U_∞ and the wind velocity at the rotor U_r to U_∞:

$$\alpha = \frac{U_\infty - U_\mathrm{r}}{U_\infty} \tag{2-4}$$

However, AIC would not significantly increase the power generation of a wind farm according to research from among others Kheirabadi et al. [14] and Gebraad et al. [45]. But, AIC shows to be a viable concept for load mitigation and active power control.

Wake Redirection Control WRC focuses on purposely misaligning the rotor of a turbine with respect to the incoming flow. By misaligning the rotor, downstream wake deflects from downstream turbines. So, wake effects of upstream turbines would not at all or partially overlap a downwind turbine, see Figure 2-6 for a visualisation. The upstream misaligned turbine generates less power, but this loss in power generation is compensated by the increase in the power generation of the downwind turbine. WRC can be achieved with yaw actuation,

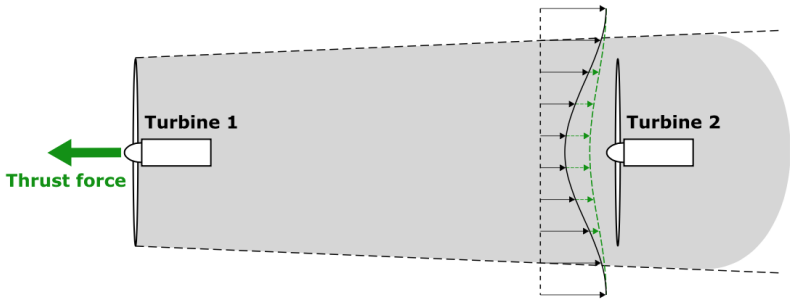


Figure 2-5: The thrust force on turbine two and the downstream wake effects of turbine one decrease when the axial induction factor of turbine one is reduced. Reduced wake effects have as effect that turbine two is exposed to greater wind speeds and so generates more power.

individual pitch control and tilt actuation. WRC shows to be potential for raising wind farm efficiency according to, among others, Kheirabadi et al. [14] and Gebraad et al. [46]. A downside of WRC is that current turbines are not designed to be yawed into the wind. Yawed turbines experience an increase in dynamic load on some parts of the turbine, this could result in higher maintenance costs.

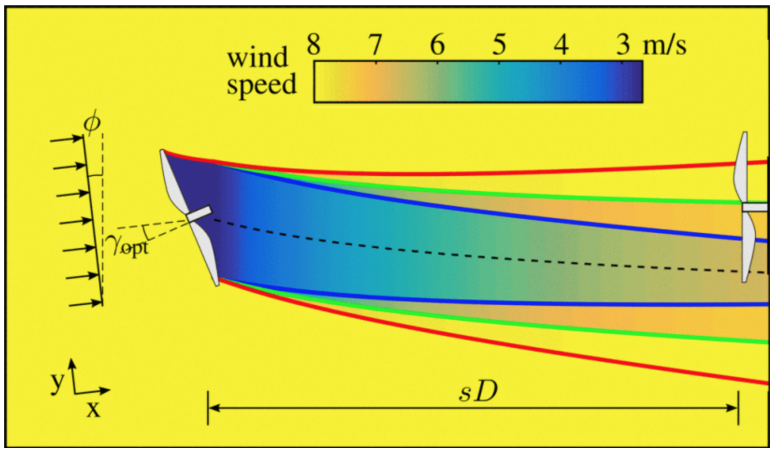


Figure 2-6: For each wind inflow angle ϕ, the control scheme sets the upwind turbine (left) at a constant yaw misalignment γ_{opt} to optimize the wind farm power output. The turbines are spaced s rotor diameters (D) apart. [4]

Wake Mixing WM is a novel concept in which upstream turbines are dynamically uprated and downrated on short timescales but at low frequency to induce additional wake recovery. This method could minimize wake losses further downstream, see Figure 2-7 for a visualisation. WM can be achieved by dynamically changing the generator torque or the individual blade pitch angles. Constantly yawing the turbine could also result in faster wake recovery [47]. Frederik et al. [5] recently came up with a new method in which wake mixing is achieved by using dynamic individual pitch control. A downside of WM control is the fluctuation of the thrust force. However, the frequency of the oscillation is really low such that it does

not significantly affect the structure of the turbines. Another downside is that the turbine is subjected to relatively high structural loads during the up- and downrating.

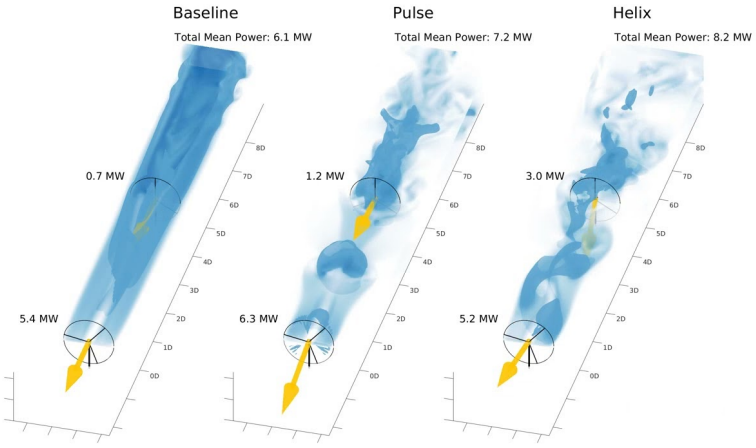


Figure 2-7: The Baseline case shows the power generation of two aligned wind turbines, the darker blue, the lower the wind speed. The Pulse and Helix cases are two wake mixing control approaches. This picture is a screenshot from a simulation created by Frederik et al. [5].

2-4 Wind Farm Simulation Tools

It is too expensive and time-consuming to design and test controls solely on field tests. Therefore wind farm simulation tools were developed to, among others, design and test controls. The fidelity and mathematical structure of simulation tools can vary greatly, but simulation tools always consist on one hand of a turbine model and on the other hand of a flow model. The turbine model predicts the interaction between the turbine structure and flow. The flow model predicts the wake effects in the flow. The turbine model gets as input a flow field from a flow model, and the flow model gets as input the turbine loadings from the turbine model. For control design, it is of importance that the relevant physics of a wind turbine are accurately modelled so that the estimated turbine and controller performance correspond to reality. Inaccurately tuned simulation tools would lead to incorrectly tuned controls which could lower the power production and increase the loading. Wind turbine and wind farm simulation tools can be distinguished based on the level of fidelity. A distinction can be made between low-, medium- and high-fidelity tools. Table 2-1 gives an overview of different (flow) simulation tools used in literature [38, 39]. A higher model fidelity means a higher computational time and so higher computational costs.

2-4-1 Low-Fidelity Simulation Tools

Simulation tools that use parametric flow models for estimating the flow field in a wind farm belong to the group of low-fidelity simulation tools. Parametric flow models are used to estimate only the most dominant wake characteristics, such as wake deflection and velocity deficit. Low-fidelity models show large errors in the prediction of near-wake zones [57]. But, in

Model	Fidelity	Fundamentals	Comp. effort
Jensen Park Model [48]	Low	Static 2D parametric	Order of seconds
Frandsen Model [49]	Low	Static 2D parametric	Order of seconds
FLORIS [17]	Low	Static 2D parametric	Order of seconds
FLORIDyn [50]	Low	Dynamic 2D parametric	Order of seconds
DWM Model [51]	Medium	Dynamic 2D Navier-Stokes	Order of minutes
FAST.Farm [6]	Medium	Dynamic 2D Navier-Stokes	Order of minutes
Ainslie [52]	Medium	Dynamic 2D Navier-Stokes	Order of minutes
WFSim [53]	Medium	Dynamic 2D Navier-Stokes	Order of minutes
SP-Wind [54]	High	Dynamic 3D Navier-Stokes	Order of days
WakeFarm [55]	High	Dynamic 3D Navier-Stokes	Order of days
UTDWF [56]	High	Dynamic 3D Navier-Stokes	Order of days
SOWFA [18]	High	Dynamic 3D Navier-Stokes	Order of days

Table 2-1: Overview of different flow field simulation tools.

some cases they show accurate predictions for far-wake zones when uncertainty is included in the models [58]. This is still an active field of research. Examples of parametric wake models are the Jensen model [48] and FLow Redirection and Induction in Steady State (FLORIS) [17]. In general, these models estimate steady-state situations in a wind farm, such as a given inflow direction for all wind turbines. These low-fidelity flow models are interesting for online control because of the low computational costs.

2-4-2 Medium-Fidelity Simulation Tools

The 3-Dimension (3D) Large-Eddy Simulation (LES) method can be expressed in a 2-Dimension (2D) model, as Boersma et al. [59] show in their paper. Simulation tools based on the 2D Navier-Stokes (NS) equations for estimating the flow field are referred to as medium-fidelity simulation tools. Tools using NS equations for estimating the flow field are based on Computational Fluid Dynamics (CFD). Medium-fidelity tools have a simplified turbulence model to induce wake recovery. Models based on the 2D NS equations are way less computationally expensive than 3D models. Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) [6] and the Dynamic Wake Meandering (DWM) model [51] are examples of models which uses 2D NS equations to solve the flow field dynamics. Note that in 2D NS equation flow models the inflow from above and below is either estimated or neglected. This makes these models less computational expensive but also less truthful.

2-4-3 High-Fidelity Simulation Tools

High-fidelity simulation tools estimate the flow field by resolving the NS equations in 3D. The Direct Numerical Simulation (DNS) method simulates turbulent flows by directly solving the obtained set of equations on a very dense grid, capturing all eddy scales. This is the most accurate way for simulating turbulent flow fields. DNS is computationally expensive, the obtained set of equations is huge because every cell in the wind farm has its own 3D NS equations. The LES method resolves the governing equations on a coarser mesh, but

can approximate the smaller-scale eddies with sub-grid models. Small-scale turbulence is then calculated within each coarse cell using this sub-grid model. Simulator fOr Wind Farm Applications (SOWFA) [60] is in literature a commonly used tool based on LES for predicting the flow fields. High-fidelity simulation tools are typically used to validate or test controllers.

2-5 Summary

The purpose of this chapter was to outline the working of a wind turbine, introduce different wind farm control methods, and to overview different wind farm simulation tools. Horizontal-axis wind turbines can be controlled by changing the generator torque, blade pitch angles, and yaw angle. Wind farm control methods like WRC and active power control can be used to increase the performance of a wind farm compared to greedy controlled wind farms. Next, wind farm simulation tools can be used to evaluate the performance of wind farm controllers. Wind farm simulation tools can be subdivided in low- medium- and high-fidelity simulation tools. This overview is needed in order to understand the design of wind farm controllers and the purpose of wind farm simulation tools. The next chapter outlines the principles of FAST.Farm.

Chapter 3

Principles of FAST.Farm

3-1 Introduction

Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) is a recently developed medium-fidelity multiphysics engineering tool by National Renewable Energy Laboratory (NREL) which aims to balance the need for accurate modeling of the relevant physics while maintaining low computational cost. This chapter explains the working or FAST.Farm.

3-2 Working of FAST.Farm

FAST.Farm is a multiphysics engineering tool for predicting the power performance and structural loads of wind turbines within a wind farm [6, 20]. FAST.Farm solves the aero-hydro-servo-elastic dynamics of each individual turbine with the use of Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST). In addition FAST.Farm considers additional physics for wind farm-wide ambient wind in the atmospheric boundary layer; a wind farm super controller; and wake deficits, advection, deflection, meandering, and merging. FAST.Farm can be used to develop and simulate wind farm controls on a wind farm level. FAST.Farm considers the 2-Dimension (2D) Navier-Stokes (NS) equations to solve the flow field dynamics. In 2D NS equation flow models the inflow from above and below is either estimated or neglected. This makes the tool less computational expensive than high-fidelity simulation tools. This simulation environment is mostly written in Fortran. All module states in FAST.Farm are expressed in discrete time. FAST.Farm can be run in either serial mode or parallel mode. Each (OpenFAST) instance can be run on a separate thread in a computer in parallel mode. In this thesis the parallel mode of FAST.Farm is used.

FAST.Farm is composed of multiple submodels, each submodel represents a different physical domain of the wind farm [19]. The different submodels are interconnected through the FAST.Farm Driver, see Figure 3-1. FAST.Farm follows the programming requirements of the FAST modularization framework. The number of submodels connected to the driver depends

on the number of wind turbines within a wind farm. For every wind turbine an OpenFAST submodel and a wake dynamics module, have to be connected to the driver. Only one Super Controller and one Ambient Wind & Array Effects (AWAE) module are connected to the driver. Every submodel contains so-called 'input files'. The settings of the submodels have to be defined in the input files of FAST.Farm. These input files contain information about, for example, the inflow wind field, the turbine dynamics, and the grid size. There are more than hundred parameters that can be manually adjusted in all the input files. Every different real live situation requires a different set of parameters. The next subsections outline the different submodels of FAST.Farm.

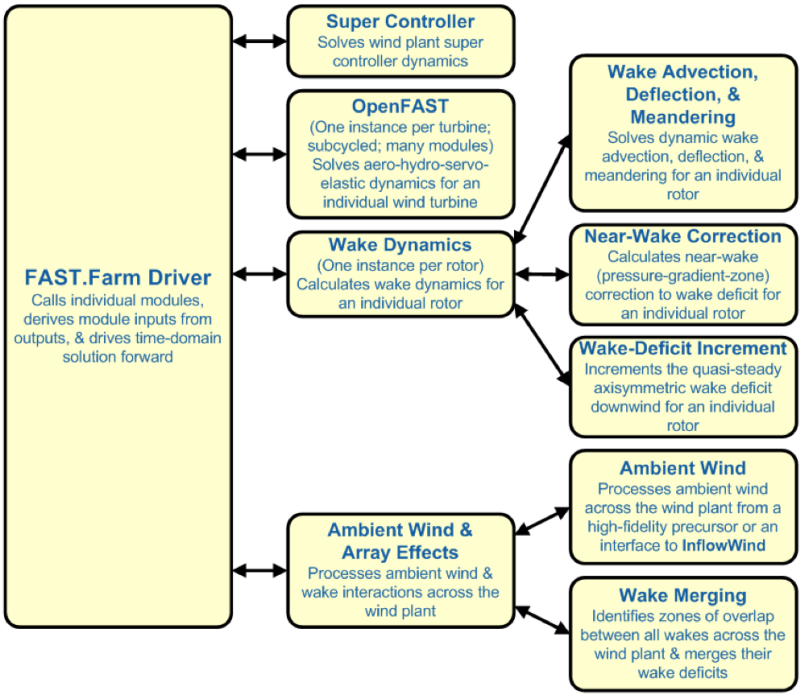


Figure 3-1: FAST.Farm submodel hierarchy [6].

3-2-1 FAST.Farm Driver

The FAST.Farm driver couples all the individual modules together and drives the overall time-domain solution forward. In addition, the driver reads an input file of simulation parameters, checks the validity of these parameters, initializes the modules, writes results to a file, and releases memory at the end of the simulation. The driver could be seen as the glue that connects everything together. The OpenFAST module is (in general) the computationally slowest module of FAST.Farm. Because, the OpenFAST module has the highest sampling time compared to the other modules.

3-2-2 Wind Farm Super Controller

The Super Controller (SC) is a centralized wind-farm-wide controller that controls all the wind turbines in the farm. The SC receives information, by means of exchanging the avr-SWAP matrix [61], from all the individual OpenFAST wind turbine models about, e.g., the blade pitch angle and generator torque. Based on this information the SC determines updated control variables and sends commands to all the individual wind turbines. The output of the SC could be global controller commands and/or individual turbine controller commands. The behavior of the SC depends on the control objective, e.g. active power control, load minimization or power maximization. The Simulator fOr Wind Farm Applications (SOWFA) SC proposed by Flemming et al [22, 62] is essentially identical to the SC module in FAST.Farm. The SC has to be written in a Fortran DLL and could communicate with the individual turbines by exchanging the avrSWAP matrix. See section 4-3-4 for an explanation of the avr-SWAP matrix. The SC has (in general) the lowest sampling rate of all FAST.Farm modules. The SC sampling rate is equal to the low-dimension time-step of FAST.Farm.

3-2-3 OpenFAST

The OpenFAST module [7] models the dynamics (loads and motions) of distinct turbines in a wind farm. For every turbine in a wind farm a separate OpenFAST module is connected to the driver. OpenFAST is able to capture the environmental excitations and coupled system response of a turbine. Environmental excitations for offshore turbines are wind inflow, waves, current and ice. With coupled system response of a turbine is referred to the response between the rotor, drivetrain, nacelle, tower, controller and substructure. OpenFAST itself, like FAST.Farm, is also an interconnection of various modules, each corresponding to different physical domains of the coupled aero-hydro-servo-elastic solution, see Figure 3-2. The origin of a turbine is defined as the intersection of the undeflected tower centerline and the mean sea level. OpenFAST uses the disturbed wind across a high-resolution wind domain around the turbine as input. Some settings that can be modified in the input files of OpenFAST are, for example, the tower dynamics, and the blade stiffness. For accuracy and numerical stability

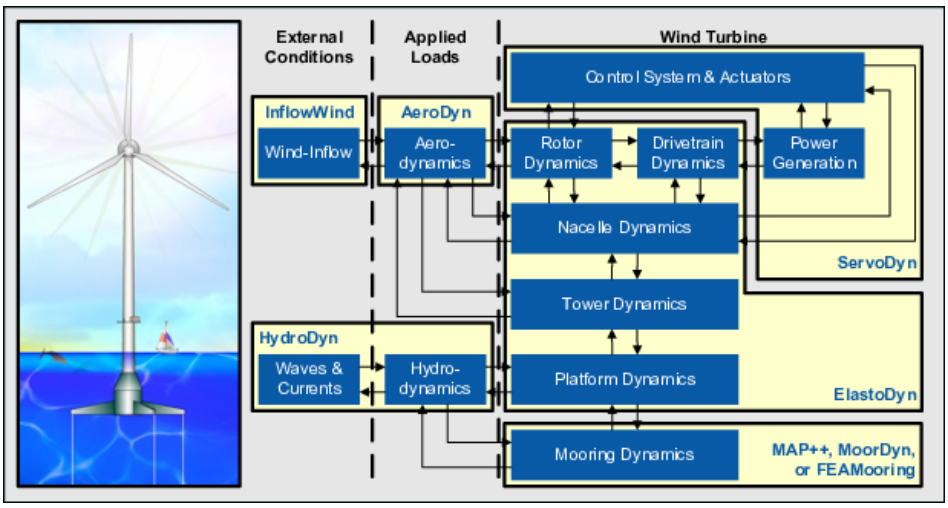


Figure 3-2: The OpenFAST modules [7].

reasons, the OpenFAST time-step is usually smaller than the FAST.Farm timestep. This results in OpenFAST being sub-cycled within a FAST.Farm time-step. In addition, for the purpose of coupling OpenFAST and FAST.Farm, the OpenFAST module functions in discrete time. See figure 3-3 for the relationship between the time-step of the OpenFAST module, and the low- and high-resolution domain time-step of FAST.Farm. The OpenFAST instance could contain an internal turbine controller, in this case the SC sends reference signals (e.g. power reference) instead of turbine commands.

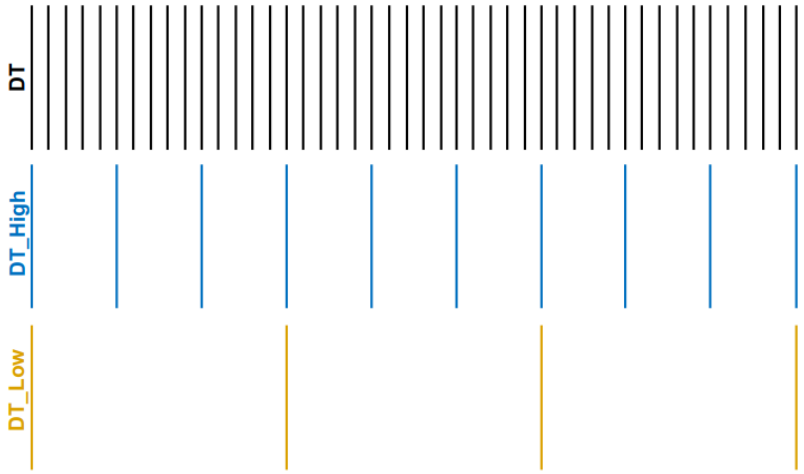


Figure 3-3: Illustration of the timescale ranges for OpenFAST (DT) and the FAST.Farm low- (DT_Low) and high- (DT_High) resolution domain [6].

3-2-4 Wake Dynamics

The wake dynamics module calculates the wake dynamics for an individual rotor. Each wake plane in FAST.Farm is treated as a radial finite-difference grid. The wake dynamics module has three submodels: wake advection, deflection, and meandering; near-wake correction; and wake-deficit increment. For every turbine in a wind farm a separate wake dynamics module is connected to the driver. The calculation of the wake dynamics involves many user-specified parameters [63]. The user can, e.g., specify the radial increment, number of radii and the number of wake planes. The number of wake planes should be big enough in order to pass forward the wake effects. Most wake dynamic parameters of FAST.Farm were calibrated on SOWFA simulations [64, 65]. In these SOWFA simulations the wind turbine was modeled according to the Actuator Line Model (ALM). The sampling rate of the wake dynamics is equal to the high-resolution domain time-step in FAST.Farm.

- **Wake Advection, Deflection & Meandering** The wake advection, deflection & meandering module solves the dynamic wake advection, deflection and meandering for a turbine. Wake advection results, e.g., from a step change in yaw angle. Therefore a wake deflects in latitudinal direction and shows horizontal and vertical oscillations (meandering) over time.

- **Near-Wake Correction** The near-wake correction treats the near-wake correction of the wake deficit. This submodel computes the wake velocity deficits at the rotor disk. The wake velocity deficits are an inlet boundary condition for the wake-deficit evolution.

- **Wake-Deficit Increment** The wake-deficit increment shifts the quasi-steady-state axisymmetric wake deficit nominally downwind.

3-2-5 Ambient Wind & Array Effects

The AWAE module processes ambient wind and wake interactions across the wind farm. This module has two submodels, ambient wind and wake merging. Ambient wind has to be available in two different resolutions in both space and time. FAST.Farm needs a low-resolution wind domain throughout the wind farm, because wind will be spatially averaged across wake planes within the AWAE module. FAST.Farm needs high-resolution wind domains around each wind turbine for accurate load calculation by OpenFAST. The dimensions of the low- and high-resolution domain have to be specified in the FAST.Farm input file.

- **Ambient Wind** The ambient wind submodel processes ambient wind across the wind farm. Ambient wind may come from either a precursor simulation (in SOWFA) or an interface to the *InflowWind* module in OpenFAST (see Figure 3-2).

- **Wake Merging** The wake merging submodel identifies zones of overlap between all wakes across the wind farm by finding wake volumes that overlap in space. Subsequently, the wake volumes/deficits are merged. Figure 3-4 visualizes the wake merging for closely spaced rotors.

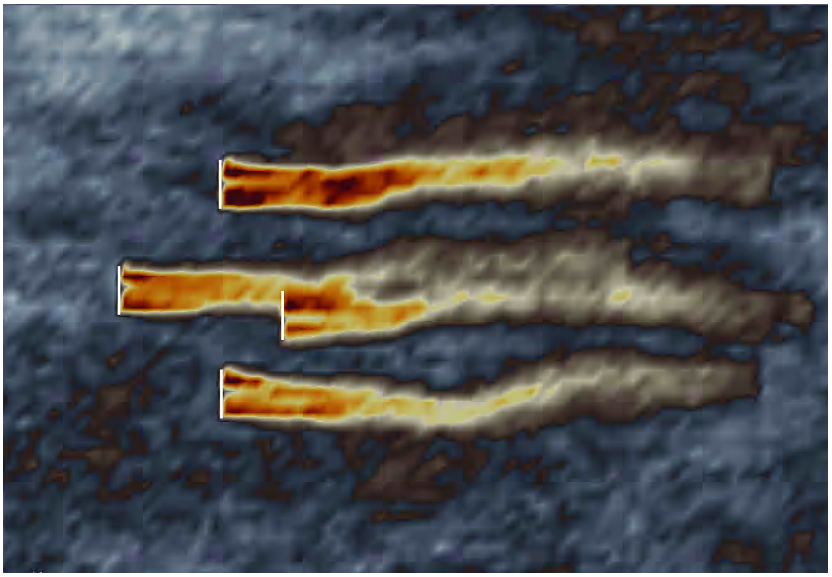


Figure 3-4: Wake merging of closely spaced rotors [6].

Low- and High-Resolution Domain for Wake Dynamics

The low-resolution domain represents the space in which the simulation of the farm is executed. Turbines may potentially reside over the whole low-resolution domain. Smaller high-resolution domains are situated around each wind turbine in the farm. The high-resolution domain is placed on top of the low-resolution domain. Every wind turbine has their own (separate) high resolution domain. High-resolution domains have (in general) a higher sampling rate than the low-resolution domain, see figure 3-3. A higher sampling rate results in more accurate wind turbine behavior and wake dynamic calculations. In theory the whole simulation space could be covered by the high-resolution domain, but this would unnecessary increase the simulation time. Figure 3-5 represents the structure for the low- and high-resolution domains. The domains can be modified accordingly, based on the desired dimensions. In this thesis the high-resolution and low-resolution spatial nodes are equally spaced. In addition, the high-resolution domain has the lowest time-step. The size of the low- and high-resolution domain is calculated with the following formulas [6]:

$$X = (NX - 1)dX \tag{3-1}$$

$$Y = (NY - 1)dY \tag{3-2}$$

$$Z = (NZ - 1)dZ \tag{3-3}$$

NX/NY/NZ stands for the number (no dimension) of low- and high-resolution spatial nodes in X/Y/Z direction for wind data interpolation. dX/dY/dZ stand for the pacing, in [m], of low- and high-resolution spatial nodes in X/Y/Z direction for wind data interpolation.

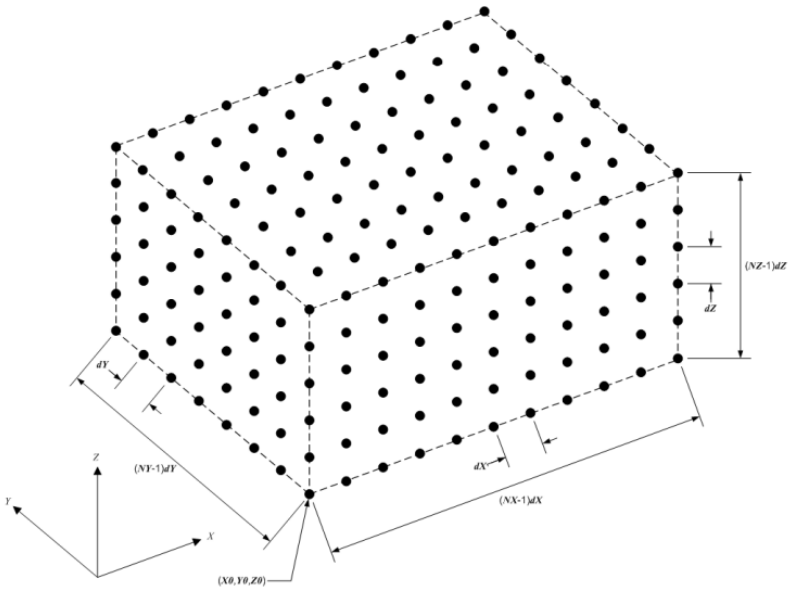


Figure 3-5: Structured 3D grid for the low- or high-resolution domains in FAST.Farm [6].

3-3 FAST.Farm Compared to SOWFA and FLORIS

FAST.Farm is one of several wind turbine/farm simulation tools, see Table 2-1 for an overview. Simulation tools can be distinguished based on the level of fidelity. FAST.Farm can be placed in the group of medium-fidelity simulation tools. FAST.Farm simulations are in the order of minutes. SOWFA and FLow Redirection and Induction in Steady State (FLORIS) are two widely used (flow field) simulation tools used for designing and testing wind farm controllers. Figure 3-6 shows the differences in fidelity and computation time between SOWFA, FAST.Farm, and FLORIS for a simulation of three turbines.

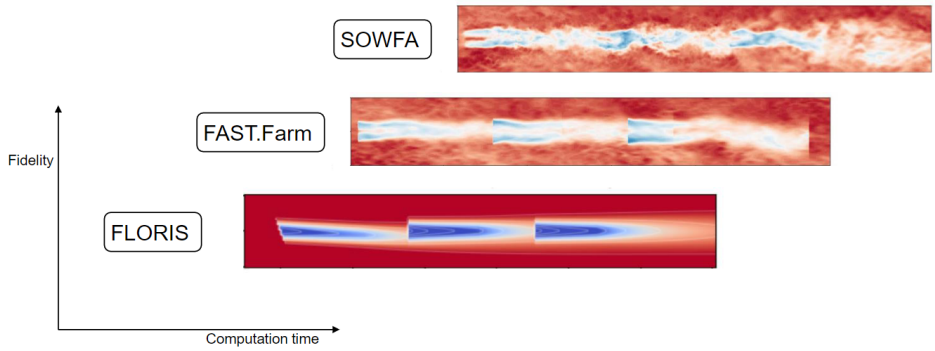


Figure 3-6: Differences between level of fidelity and computation time between SOWFA, FAST.Farm and FLORIS.

SOWFA SOWFA is a high-fidelity simulation tool based on the Computational Fluid Dynamics (CFD) software OpenFOAM. SOWFA performs Large-Eddy Simulation (LES) for predicting the flow field [60]. SOWFA is able to predict the flow field after a wind turbine with high accuracy. But, long computation time is a drawback of this simulation tool. SOWFA simulations are often used as validation data [53, 66]. SOWFA resolves the governing (NS) equations in 3-Dimension (3D), this results in more veracious simulation results than in FAST.Farm. SOWFA should be able to simulate the flow field more accurate than FAST.Farm, but lacks the possibilities to include tower movements. Lastly, SOWFA simulations are in the order of days.

FLORIS FLORIS is a static low-fidelity parametric flow field simulation tool used for predicting steady-state wake locations, flow velocities, and turbine power outputs [17]. A downside of FLORIS is that plots generated with FLORIS do not show wake propagation. FLow Redirection and Induction Dynamics (FLORIDyn) extends the FLORIS model by modelling simple wake propagation [67]. FLORIS can be used in wind farm controllers for, e.g., calculating optimal yaw angles for wind farm power optimization [15]. FLORIS simulations are in the order of seconds.

3-4 Current Method of Controller Design in FAST.Farm

The current procedure of wind farm control synthesis in FAST.Farm is by constructing wind farm controllers in Fortran language. A SC has to be written inside a Fortran Dynamic Link Library (DLL) and placed inside the FAST.Farm input file. In addition, a DISCON style Fortran DLL has to be written for the OpenFAST instances. The SC DLL contains the farm-level controller. The OpenFAST DISCON style DLL contains the turbine-level controller. The SC-DLL determines references values for the DISCON DLL. Subsequently, the DISCON DLL determines actuator commands for the turbine. See Figure 3-7 for a visualization about the the communication between the SC-DLL and the OpenFAST instances. A drawback of constructing a wind farm controller in a Fortran DLL is that Fortran is a compiled language made for performance and lacks flexibility and interactivity compared with MATLAB/Simulink. The DISCON and SC-DLLs have to be recompiled for every modification on the farm or turbine controllers.

The SC-DLL communicates with the individual OpenFAST modules by exchanging the avrSWAP matrix. This is, as far as I know, the only possible way of communication between the OpenFAST DISCON DLLs and other DLLs or programs. By exchanging the avrSWAP matrix with MATLAB/Simulink, a FAST.Farm and MATLAB/Simulink interface can be realised. MATLAB/Simulink should than be coupled to the DISCON DLLs of each OpenFAST module in FAST.Farm. This would exclude the use of the SC-DLL in FAST.Farm. The next chapter discusses the creation of such a FAST.Farm and MATLAB/Simulink interface.

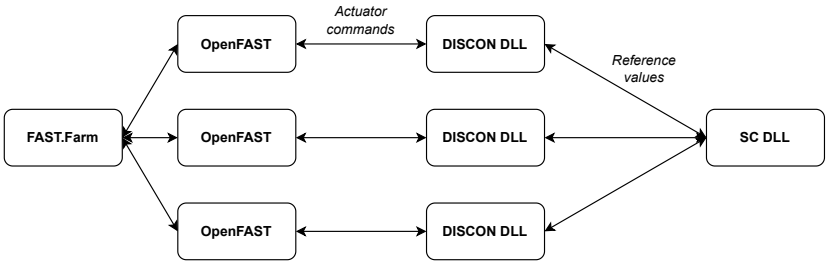


Figure 3-7: Calling sequence between FAST.Farm, the DISCON DLLs and the SC-DLL.

3-5 Summary

This chapter has outlined the principles FAST.Farm. FAST.Farm consists of several modules interconnected by the FAST.Farm driver. All those modules, e.g. wake dynamics and turbine properties, can be adjusted accordingly by modifying the appropriate input files. FAST.Farm seems to be a promising simulation tool for wind farm control design, but is lacking the flexibility for controller design in comparison with MATLAB and Simulink. The next chapter explains the creation of an interface between FAST.Farm and MATLAB/Simulink.

Chapter 4

FAST.Farm and MATLAB/Simulink Interface

4-1 Introduction

This chapter explains the creation and operation of the Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) and MATLAB/Simulink interface. The interface is realized by linking the individual Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST) instances in the FAST.Farm tool to MATLAB through an Message Passing Interface (MPI) and MATLAB Executable (MEX) functions.

4-2 MATLAB and Simulink

MATLAB and Simulink are both developed by Mathworks. Mathworks is specialized in developing mathematical computing software [68]. MATLAB is primarily used for analyzing and visualizing data, it can be used as well for developing algorithms. Simulink is a graphical programming environment used for designing, simulating, and analyzing dynamical systems. Simulink can be used as an add-on to MATLAB. Simulink offers the possibility to visualize controllers in a graphical block diagram. This makes Simulink a more intuitive tool for designing controllers than MATLAB.

4-3 Setup of Interface

FAST.Farm does not offer the possibility to directly be linked to MATLAB. First, on the side of FAST.Farm, an MPI is connected to each OpenFAST instance. There exists one OpenFAST instance for each wind turbine simulated in FAST.Farm. An MPI is used to exchange messages between multiple computers or programs running a parallel program across

distributed memory. This MPI connection facilitates the exchange of messages between the turbine controllers in FAST.Farm and other programs. Subsequently, on the MATLAB side, MEX functions are used to enable communication with external programs, by linking to a C/C++ or Fortran shared library. In this way, MATLAB can call the MPI interface through the use of MEX functions, and therefore messages can be exchanged between the OpenFAST instances and MATLAB. In addition, S-Functions can be used to expand the interface to Simulink. Figure 4-1 depicts the calling sequence of the interface of a wind farm in FAST.Farm with three wind turbines. The message that is sent back and forth between FAST.Farm and MATLAB is stacked in the so-called avrSWAP matrix [61]. This matrix contains all the measurement data and control variables of each wind turbine. Having access to this structure information, MATLAB can read and modify the matrix accordingly and subsequently send back the matrix with the updated control variables to FAST.Farm. The amount of wind turbines simulated in FAST.Farm is limited by the number of cores in your computer. Every turbine is subjected to one core for running the simulation. The next subsections explain the operation of the MPI interface, the MATLAB interface with MEX Functions, the Simulink interface, and the avrSWAP matrix.

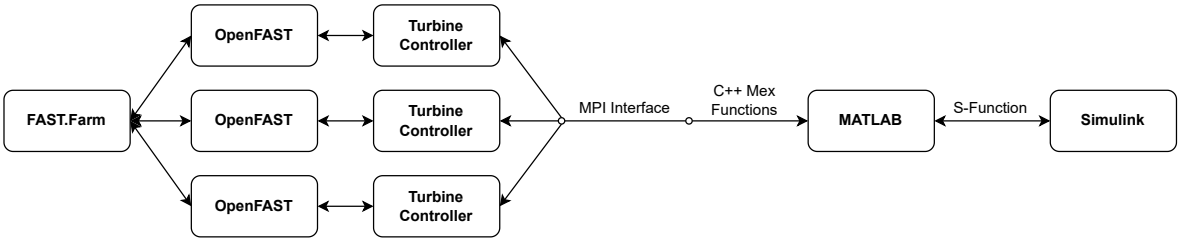


Figure 4-1: Calling sequence between FAST.Farm and MATLAB/Simulink.

4-3-1 MPI Interface

FAST.Farm can exchange messages with other programs through an MPI interface. Initially, the Dynamic Link Library (DLL) of the turbine controllers in each OpenFAST instance receive controller commands from the Super Controller (SC) in FAST.Farm. The SC-DLL of FAST.Farm is essentially identical to the super controller available in SOWFA [22]. The SC-DLL is used in conjunction with individual wind turbine controllers designed in the style of the DISCON DLL (originally used in Bladed) [23]. The SC-DLL and the DISCON DLLs communicate by exchanging the avrSWAP matrix. Instead of using an SC-DLL, the DLLs of the internal turbine controller have been modified in order to link them to an MPI. In this way, the turbine controllers can exchange the avrSWAP matrix with other programs. Then, FAST.Farm can exchange measurement data and controller commands with MATLAB/Simulink through the MPI. The interface is designed in a scalable way preserving the parallelized structure of FAST.farm. The MPI interface is not limited to a predefined number of turbine connections. The MPI interface supports the calling sequence of FAST.Farm in which FAST.Farm only accepts updated messages after completing the previous time-step. The MPI interface supports both the implementation of wind turbine and wind farm controllers. In general, a wind farm controller determines reference yaw angles and power set-points for all the wind turbines in a wind farm. Consequently, wind turbine controllers

compute the associated generator torque and blade-pitch angles. When it comes to turbine-level controls, the interface supports to use a standard Bladed-Style DLL like DTUWEC [69] or ROSCO [70], or offers the possibility to directly link MATLAB/Simulink to OpenFAST, see Figure 4-2. Using a Bladed-Style DLL turbine controller, e.g. DTUWEC, means that only a farm-level controller can be designed inside MATLAB/Simulink as the turbine controller (DTUWEC) only accepts reference points, e.g. power set-points. By directly linking MATLAB/Simulink to OpenFAST, the turbine controller has to be defined inside MATLAB/Simulink. This provides the opportunity to design the turbine controller yourself, but would slightly increase the computation time as MATLAB is slower than Fortran compiled code.

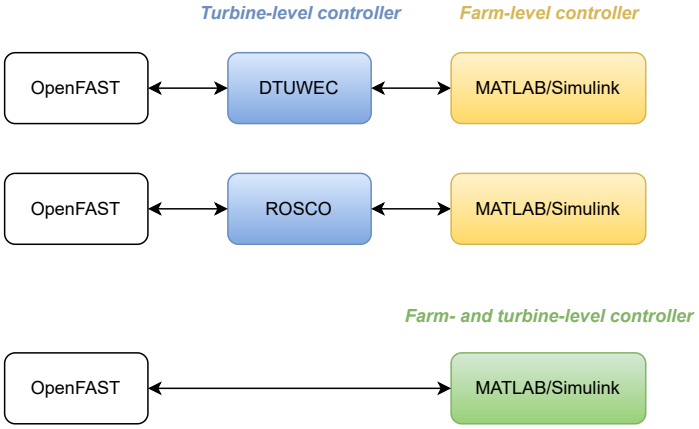


Figure 4-2: Calling sequence between OpenFAST and MATLAB. The interface supports to either use a internal controller like DTUWEC or ROSCO, or offers the possibility to link OpenFAST directly to MATLAB.

4-3-2 MATLAB Interface with Mex Functions

On the side of MATLAB, four MEX functions have been written to establish a connection between MATLAB and the MPI. Hence, information can be exchanged between MATLAB and the OpenFAST instances. The four MEX functions are named *Initialise*, *Receive*, *Send*, and *Stop*. Figure 4-3 depicts the calling sequence of the MEX functions in the MATLAB/Simulink environment.

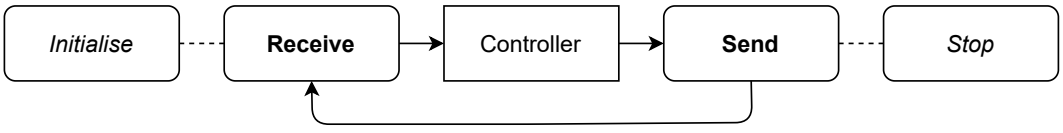


Figure 4-3: Calling sequence of MEX functions in MATLAB/Simulink.

- Initialise: This function is needed to initiate the connection between the MPI server and

MATLAB. The purpose of this function is to connect the right internal ports (inside the computer) in order to exchange messages between the MPI (client) server and MATLAB.

- Receive: By calling this function the avrSWAP matrix is retrieved from FAST.Farm to MATLAB/Simulink. The avrSWAP matrix is only updated by FAST.Farm for one turbine. FAST.Farm releases the avrSWAP matrix once one of the turbine simulations (cores) has been finished. In this way, the function only receives a signal (avrSWAP matrix) after FAST.Farm has performed a new simulation time-step. For every simulation time-step the function has to be called equal to the number of simulated turbines.

- Send: By calling this function the updated avrSWAP matrix is sent back to FAST.Farm. FAST.Farm waits for the updated matrix before it performs a new simulation time-step for the indicated turbine.

- Stop: The stop function terminates the connection between the MPI server and MATLAB once the total run time has been reached. This function is needed to prevent memory leakage.

The structure in MATLAB contains a farm-level loop and a turbine-level loop, see Algorithm 1. This algorithm offers the possibility to separately include a turbine-level controller and a farm-level controller at the indicated lines and preserves the parallel computing features of FAST.Farm. In general, wind farm controllers have a lower sampling rate than wind turbine controllers. Within a farm-level time-step, turbine signals are updated and released in parallel independent from each other. At the wind turbine level, FAST.Farm and MATLAB/Simulink perform a co-simulation in which both programs simultaneously update measurements or control variables for different turbines in any order. The *while incomplete* loop only updates the control variables for one turbine during every iteration. Once all turbines have reached the next farm-level time-step (incomplete = false), the farm-level loop proceeds. A farm-level time-step consists of 80 turbine-level time-steps in the case of a farm-level time-step of 2.0 seconds and a turbine-level time-step of 0.025 seconds. This means that, in the case of a farm with three turbines, during every wind farm loop 240 turbine-level loops are run. After the total run time, the algorithm terminates the connection.

Algorithm 1 FAST.Farm and MATLAB interface

```
Initialise
while true do
    incomplete = true
    % Update farm-level control here
    while incomplete do
        Receive
        % Update turbine-level control here
        Send
    end while
end while
Stop
```

4-3-3 Simulink Interface

Simulink offers the possibility to visualize controllers in a graphical block diagram. This makes Simulink a more intuitive tool for designing wind farm controllers than MATLAB. Messages that are received in MATLAB can be redirected to Simulink with the use of System-Functions (S-Functions). An S-Functions is a computer language description of a Simulink block written in MATLAB, C, C++ or Fortran. S-Functions can be used to dynamically link subroutines, e.g. receiving, processing, and sending data. The MATLAB execution engine can automatically load and execute S-Functions. S-Function blocks are the only Simulink blocks that accept MEX functions as input (source code). The MATLAB Function block is not able to process MEX functions. C++ MEX based S-Functions can be run in Simulink within a level 2 S-Function block. Two level 2 S-Function blocks, Receive and Send, have been constructed in Simulink in order to exchange the avrSWAP matrix between FAST.Farm and Simulink. See Figure 4-4 for the visualization in Simulink. A wind farm controller can be created in between the Receive and Send function.

Figure 4-4: Visualization Send and Receive block in Simulink.

Algorithm 2 visualises the working of the FAST.Farm and Simulink interface. Fist, a connection has to be initiated between the MPI server and MATLAB, similar as with the MATLAB interface. Once MATLAB and FAST.Farm are connected, the Simulink model with the Receive and Send function can be run. The Receive and Send functions in Simulink contain the same MEX functions as in MATLAB. So, the connection is initiated in MATLAB, and the turbine-level and farm-level loop are run in Simulink. Keep in mind that the loop in Simulink should respect the *'while incomplete'* and *'while true'* structure explained in Algorithm 1.

Algorithm 2 FAST.Farm and Simulink interface.

```
Initialise
sim(Simulink_Interface)
Stop
```

4-3-4 avrSWAP Matrix

The avrSWAP matrix contains simulation results from the FAST.Farm simulation, predefined parameters from the OpenFAST instances, and control variables from the MAT-

LAB/Simulink/Fortran controller. Table 4-1 shows the information that is wrapped in the avrSWAP matrix. See [61] for a more detailed explanation. Not every element (column) in the matrix is used, some elements are left open for future purpose. The simulation results and control variables of a turbine are all stored in a vector. These vectors stacked together form the avrSWAP matrix. So, every row (vector) of the matrix represent one turbine. Simulation results of the FAST.Farm simulation are written to the avrSWAP matrix, indicated by $-->$. MATLAB/Simulink can use this data to determine new optimal control variables. $<--$ Indicate the columns where MATLAB/Simulink can update the control variables for FAST.Farm. FAST.Farm uses these columns as input for the next time-step. Some columns are used by both, indicated by $<->$. MATLAB/Simulink and FAST.Farm can read and write to these columns. Not all the available information is needed for determining updated control variables, nor is it needed to define all the available control variables. Control variables that are not defined inside MATLAB/Simulink remain zero. Note: some parameters in the avrSWAP matrix, e.g. minimum/maximum pitch angle, have to be defined in the OpenFAST input files.

Table 4-1: Description of the columns of the avrSWAP matrix.

Column	$--$	**Description**
1	$-->$	Status flag: 0 first call, 1 next steps, -1 final call (-)
2	$-->$	Current time (sec)
3	$-->$	Communication interval (sec)
4	$-->$	Blade 1 pitch angle (rad)
5	$-->$	Below-rated pitch angle set-point (rad)
6	$-->$	Minimum pitch angle (rad)
7	$-->$	Maximum pitch angle (rad)
8	$-->$	Minimum pitch rate (most negative value allowed) (rad/s)
9	$-->$	Maximum pitch rate (rad/s)
10	$-->$	0 = pitch position actuator, 1 = pitch rate actuator (-)
11	$-->$	Current demanded pitch angle (rad)
12	$-->$	Current demanded pitch rate (rad/s)
13	$-->$	Demanded power (W)
14	$-->$	Measured shaft power (W)
15	$-->$	Measured electrical power output (W)
16	$-->$	Optimal mode gain (Nm/(rad/s)^2)
17	$-->$	Minimum generator speed (rad/s)
18	$-->$	Optimal mode maximum speed (rad/s)
19	$-->$	Demanded generator speed above rated (rad/s)
20	$-->$	Measured generator speed (rad/s)
21	$-->$	Measured rotor speed (rad/s)
22	$-->$	Demanded generator torque above rated (Nm)
23	$-->$	Measured generator torque (Nm)
24	$-->$	Measured yaw error (rad)
25	$-->$	Start of below-rated torque-speed look-up table
26	$-->$	No. of points in torque-speed look-up table (-)

Continued on next page

Table 4-1 – *Continued from previous page*

Column	−−	Description
27	−− >	Hub wind speed (m/s)
28	−− >	Pitch control: 0 = collective, 1 = individual (-)
29	−− >	Yaw control: 0 = yaw rate, 1 = yaw torque (-)
30	−− >	Blade 1 root out-of-plane bending moment (Nm)
31	−− >	Blade 2 root out-of-plane bending moment (Nm)
32	−− >	Blade 3 root out-of-plane bending moment (Nm)
33	−− >	Blade 2 pitch angle (rad)
34	−− >	Blade 3 pitch angle (rad)
35	< −−	Generator contactor (-)
36	< − >	Shaft brake status (-)
37	−− >	Nacelle yaw angle from North (rad)
41	< −−	Demanded yaw actuator torque (Nm)
45	< −−	Demanded pitch angle (Collective pitch) (rad)
47	< −−	Demanded generator torque (Nm)
48	< −−	Demanded nacelle yaw rate (rad/s)
49	−− >	Maximum number characters in "MESSAGE" argument (-)
50	−− >	Number of characters in the "INFILE" argument (-)
51	−− >	Number of characters in the "OUTNAME" argument (-)
53	−− >	Tower top fore-aft acceleration (m/s^2)
54	−− >	Tower top side-to-side acceleration (m/s^2)
55	< −−	UNUSED: Pitch override
56	< −−	UNUSED: Torque override
60	−− >	Rotor azimuth angle (rad)
61	−− >	Number of blades (-)
62	−− >	Maximum number of values returned for logging (-)
63	< −−	Number logging channels
64	−− >	Maximum number of characters returned in "OUTNAME" (-)
65	< −−	Number of variables returned for logging
69	−− >	Blade 1 root in-plane bending moment (Nm)
70	−− >	Blade 2 root in-plane bending moment (Nm)
71	−− >	Blade 3 root in-plane bending moment (Nm)
73	−− >	Rotating hub My (GL co-ords) (Nm)
74	−− >	Rotating hub Mz (GL co-ords) (Nm)
75	−− >	Fixed hub My (GL co-ords) (Nm)
76	−− >	Fixed hub Mz (GL co-ords) (Nm)
77	−− >	Yaw bearing My (GL co-ords) (Nm)
78	−− >	Yaw bearing Mz (GL co-ords) (Nm)
82	−− >	Nacelle roll acceleration (rad/s^2)
83	−− >	Nacelle nodding acceleration (rad/s^2)
84	−− >	Nacelle yaw acceleration (rad/s^2)

4-4 Working of Interface

Exchanging the avrSWAP matrix with MATLAB/Simulink is the key feature of this interface. Figure 4-5 shows the FAST.Farm submodel hierarchy with the MATLAB/Simulink connection included. This interface can be used to design wind turbine-level and wind farm-level controllers in MATLAB and Simulink. As depicted in Algorithm 1, the farm-level loop runs once every low-resolution time-step of FAST.Farm. The turbine-level loop runs every module time-step of OpenFAST. The sampling rate can be calculated by dividing the low-resolution time-step of FAST.Farm by the module time-step of OpenFAST. Off course, the settings of FAST.Farm highly affect the outcome of the simulations. Keep in mind that the avrSWAP matrix is the only message that is transmitted between FAST.Farm and MATLAB/Simulink, this means that controller commands have to be based on the information available in the avrSWAP matrix. Technically speaking, the real-time output of FAST.Farm could be used as input to MATLAB. But, this increases the computation time and is not an elegant programming method.

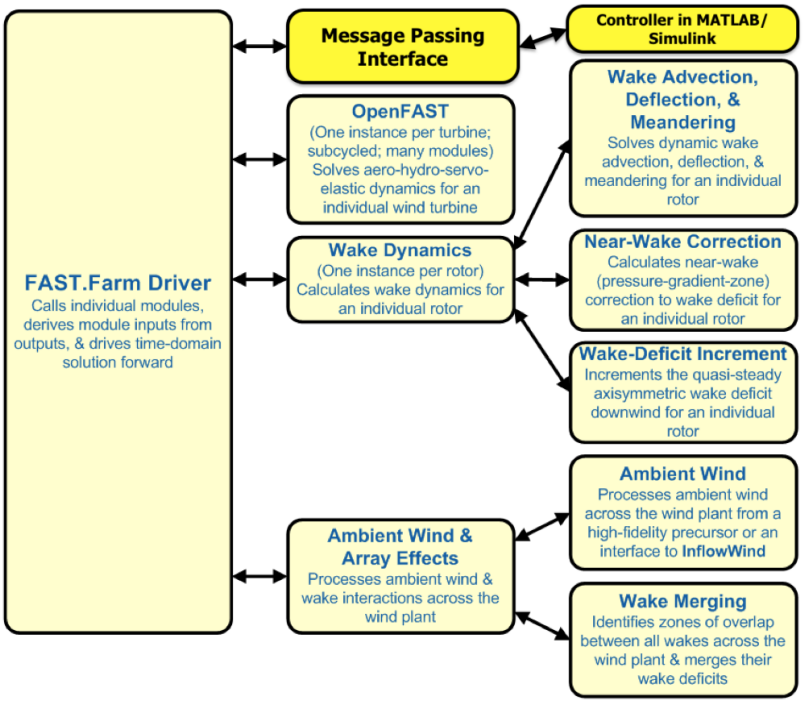


Figure 4-5: FAST.Farm submodel hierarchy with link to MATLAB/Simulink.

4-5 Summary

This chapter has given an overview of the creation and operation of the FAST.Farm and MATLAB/Simulink interface. This interface extends the controller design capabilities of FAST.Farm. The interface can be downloaded from GitHub [24]. The next chapter describes the wind turbine and wind farm controllers that were implemented and tested in FAST.Farm.

Chapter 5

Wind Farm Controller Design and Implementation

5-1 Introduction

This chapter outlines the design and implementation of wind farm controllers inside the Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) and MATLAB/Simulink interface. Wind farm controllers consist of a turbine-level loop and a farm-level loop. A yaw controller and an Active Power Controller (APC) have been implemented to show the possibilities of the FAST.Farm and MATLAB interface.

5-2 Controller Design and Implementation

Wind farm controllers could be designed on farm-level or on turbine-level. On farm level a wind farm controller determines reference values, e.g. power references, for all the turbines in a farm. On turbine level a controller determines actuator commands, e.g. blade pitch angle and generator torque. The current industry standard are greedy controllers, these controllers only determine optimal wind turbine actuator commands on turbine level without taking into account the aerodynamic coupling with other turbines in a farm. However, research has demonstrated that applying wind farm controllers could increase the performance of wind farms [13, 14]. Current research on wind farm control aims to develop controllers which take into account the wake interactions within a wind farm and the performance of individual turbines [15, 16]. FAST.Farm can be used to design and test wind farm controllers. The next sections explain the implementation of wind farm controllers in FAST.Farm through the MATLAB interface. First, a simple yaw controller is designed to validate the working of the interface. Second, an APC is implemented to show the possibilities of the interface.

5-2-1 Yaw Controller

Yaw control can be placed in the group of Wake Redirection Control (WRC). Yaw control focuses on purposely misalign the rotor of the first (most upstream) turbine with respect to the incoming flow. By misaligning the rotor, the downstream wake could deflect from downstream turbines. The operation of the FAST.Farm and MATLAB/Simulink interface can be validated by controlling the yaw angle in MATLAB and plotting the results of FAST.Farm. A small wind farm consisting of three turbines is simulated in FAST.Farm. In MATLAB a simple yaw controller is designed which controls the yaw angle(s) of the first (and second) most upstream turbine(s) in the farm. This yaw controller is implemented within the turbine-level loop. The yaw angle references are set in the farm-level loop. The simulation results of FAST.Farm are plotted in ParaView to show the behavior of the turbines.

FAST.Farm lacks the possibility to directly control the yaw angle. The avrSWAP matrix does not have a control variable that is used by FAST.Farm for controlling the yaw angle. However, the yaw angle can be indirectly controlled by controlling the yaw rate. The 'Nacelle yaw angle from north' is measured by FAST.Farm and updated in the avrSWAP matrix in column 37. Subsequently, MATLAB/Simulink receives this yaw angle and compares it with the desired yaw angle. Based on the deviation between those two angles the yaw rate steers the nacelle in order to minimize the deviation. A threshold is set in order to prevent the turbine from constantly yawing. Algorithm 3 describes the implementation of a yaw controller in the FAST.Farm and MATLAB interface. In this example the first turbine is 30 degrees yawed with respect to the nacelle yaw angle from north. During every *while incomplete* loop the received avrSWAP matrix is updated for only one turbine, namely turbine iT. So, during one *while incomplete* iteration the controller only updates the control variables for one turbine.

The yaw controller outlined in Algorithm 3 is implemented in a Fortran Dynamic Link Library (DLL) and in the MATLAB interface. The working of the MATLAB interface can be fully validated by comparing the FAST.Farm simulation results of both yaw controllers. In addition, this gives the opportunity to compare the computation times of a yaw controller implemented in a Fortran DLL versus the MATLAB interface. No generator torque control loop is implemented in this controller, a constant generator toque is applied after 80 seconds. Because, the aim of this controller is to validate the working of the MATLAB interface, not to implement advanced controllers. In this controller the first turbine starts yawing after 300 seconds, the second turbine starts yawing after 600 seconds.

5-2-2 Active Power Controller

An APC with a turbine level control loop and a farm level control loop is implemented in the MATLAB interface. The APC ensures a stable power supply to the grid in order to improve the quality of wind energy. In addition, APC algorithms could have among others thrust force balancing and load limiting as secondary objectives. The APC implemented in this section is based on the power tracking controller designed by the authors in Silva et al. [16, 25]. See their paper for the technical details, e.g. stability analysis, of the power tracking controller. A wind farm consisting of three wind turbines is simulated considering the scenario of high wake interaction. With the implemented APC, the wind farm responds to grid requirements through the control of wind farm power output. On farm level the

Algorithm 3 Yaw controller

```
Initialise
while true do % Farm-level loop
    incomplete = true
    if time > 300 then
        Desired_Yaw_Angles = [0.52 0 0] % (rad)
    else
        Desired_Yaw_Angles = [0 0 0] % (rad)
    end if
    while incomplete do % Turbine-level loop
        Receive avrSWAP
        Yaw_Angle = avrSWAP(iT,37) % (rad)
        Yaw_Error = Yaw_Angle - Desired_Yaw_Angle(iT)
        if Yaw_Error > Threshold then
            Yaw_Rate = 0.052 % (rad/s)
        else if Yaw_Error < -Threshold then
            Yaw_Rate = -0.052 % (rad/s)
        else
            Yaw_Rate = 0.0 % (rad/s)
        end if
        avrSWAP(iT,48) = Yaw_Rate % (rad/s)
        Send avrSWAP
    end while
end while
Stop
```

wind farm controller determines power output references for every turbine in the farm. On turbine level the turbine controllers determine actuator commands based on the reference values from the farm controller. Figure 5-1 visualizes the interaction between a farm and turbine controller. The following paragraphs describe the working and implementation of the turbine-level and farm-level control loop. In general, the turbine-level control loop has a higher sampling rate than the farm-level control loop.

Turbine-level control loop

At wind turbine level the power tracking controller receives signals from both, FAST.Farm and the wind farm controller, see Figure 5-1 for a visualization. On one side, FAST.Farm sends the current blade pitch angle, generator torque, and rotor speed to the wind farm controller. On the other side, the wind turbine controller receives the reference signals, power reference in this case, from the wind farm controller. Subsequently, the wind turbine controller determines the updated blade pitch angle and generator torque in order to match the

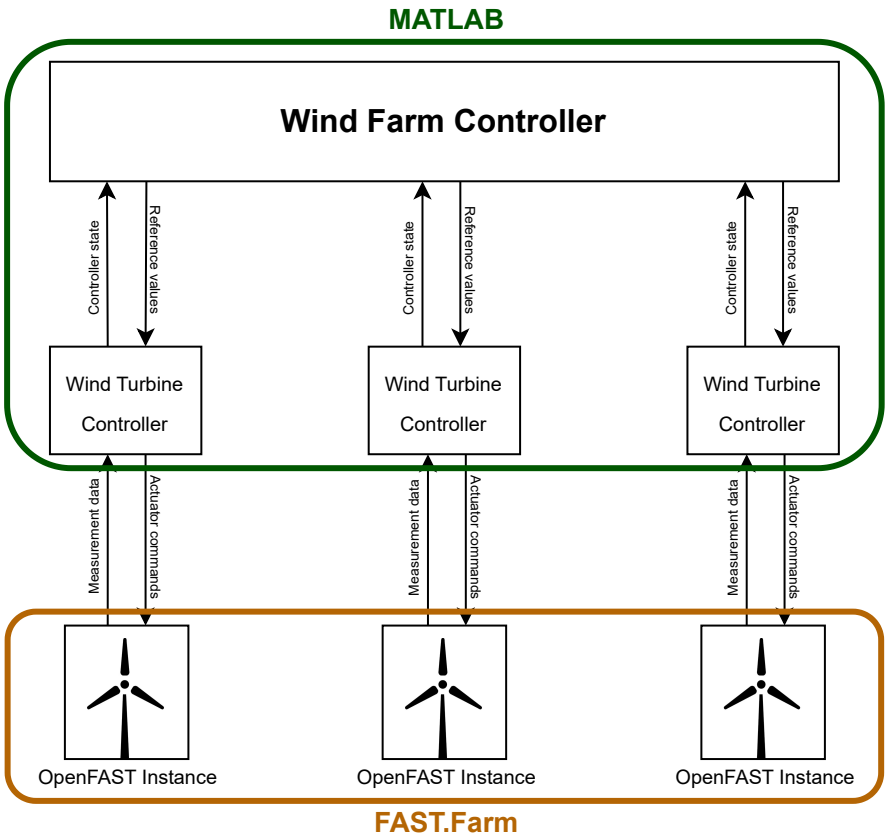


Figure 5-1: Block diagram visualizing the controller interface between MATLAB and FAST.Farm.

power output with the reference signal.

Two operational modes can be distinguished in the power tracking controller. The controller either forces a turbine to track the power reference or operates in greedy mode when the power reference exceeds the maximum available power in the wind. In the power tracking mode, a pitch controller regulates the rotor speed to a rotor speed reference while greedy torque control is applied. In the greedy mode, the main objective is to extract the maximum available power from the wind limited by electro-mechanical constraints. The controller starts operating once the wind speed exceeds the cut-in wind speed (power curve region two). The power tracking algorithm is based on the KNU2 algorithm in Kim et al. [71].

Farm-level control loop

The wind farm controller distributes the power reference over the number of turbines in the farm to follow a power command signal provided by the system operator. The farm controller uniformly distributes the power command signal, provided that all turbines are able to meet the power references from the farm controller. Downstream wake effects could

reduce the power generation of downstream turbines. This could lead to so-called 'turbine saturation'. Turbine saturation occurs once the individual available power in the wind is lower than the individual demanded power. Turbines switch to greedy mode when the power reference is higher than the maximum available power in the wind. When (downstream) turbines are saturated, the wind farm controller redistributes the power reference signals in order to comply with the power command signal from the system operator. This would result in higher power reference values for more upstream turbines. The wind farm power reference signal could be the power demand from the grid. The Automatic Generation Control (AGC) signal is used as farm power reference signal [72]. The AGC signal lasts for 1000 seconds but starts at 300 seconds to allow time for the wakes to develop and propagate during the simulation. See Figure 5-2 for the AGC signal. This AGC signal exceeds the power capacity of a single turbine.

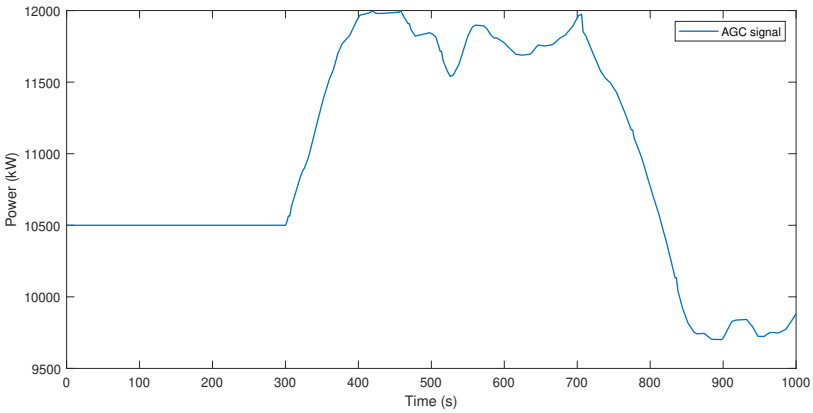


Figure 5-2: Automatic Generation Control signal.

Implementation

The APC was implemented in Algorithm 1, this resulted in Algorithm 4. During every turbine-level time-step the algorithm keeps track of the generated power. The blade pitch angle and generator torque are controlled in the turbine-level loop in order to track the power reference. The power references are updated once every farm-level time-step. See GitHub [24] for the MATLAB scripts.

Algorithm 4 Active power controller

```
Initialise
while true do % Farm-level loop
    incomplete = true
    Power_Setpoint = AGC
    if Turbine saturated then
        Redistribute power references
    end if
    while incomplete do % Turbine-level loop
        Receive avrSWAP
        Power tracking algorithm
        Torque controller
        Blade pitch controller
        Send avrSWAP
    end while
end while
Stop
```

5-3 Summary

This chapter has outlined the design and implementation of a yaw controller and an APC. A yaw controller has been designed in order to validate the working of the FAST.Farm and MATLAB interface. The yaw controller yaws the most upstream turbine of a small wind farm in order to demonstrate the exchange of the avrSWAP matrix between FAST.Farm and MATLAB. The APC proposed in this chapter is simulated in the next chapter to show the possibilities of FAST.Farm with respect to SOWFA. Both controllers consist of a turbine-level loop and a farm-level loop. This structure respects the programming interface of FAST.Farm. See GitHub [73] for the MATLAB scripts of these controllers. The next chapter shows the simulation results of these controllers simulated in FAST.Farm.

Chapter 6

Results of FAST.Farm and SOWFA Simulations

6-1 Introduction

This chapter shows the simulation results generated by Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) and Simulator fOr Wind Farm Applications (SOWFA). These simulation results validate the functionality of the interface. The controllers outlined in chapter 5 have been simulated in FAST.Farm with the MATLAB and FAST.Farm interface.

6-2 Communication Time Comparison

Three 10 MW wind turbines have been simulated in FAST.Farm with and without the use of the FAST.Farm and MATLAB interface in order to compare the computation time, see Table 6-1. The simulations have a simulation time of 600 seconds. No wind farm controller is implemented in this comparison. Therefore, the difference in simulation time indicates the communication time between FAST.Farm and MATLAB. Table 6-1 shows that the interface adds no significant overhead. The simulation time is of the order of real time, depending on the chosen turbine-level time-step. The difference between these two simulations lies in the fact that in the MATLAB interface simulation the avrSWAP matrix is exchanged between MATLAB and FAST.Farm. Values may differ from machine to machine. Such a simulation of three aligned wind turbines in FLORIS would be in the order of seconds on a desktop PC, and in SOWFA in the order of days on a cluster of 10^2 CPU's [38].

The total real-time is the real-time the computer needed to run the simulation. The total CPU time is the length of time that the central processing unit takes to process the simulation. The difference between total and simulation CPU time lies in the fact that the program needs to start and run the model before it can run the simulation. But, this has

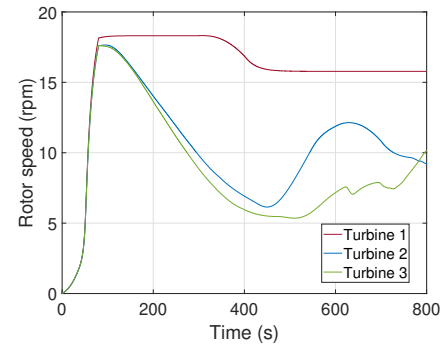
almost no impact on the total CPU time. The time ratio is calculated by dividing the simulation time by the total CPU time. The bigger this number, the faster your computer. A ratio above one would imply that FAST.Farm can simulate an event in less time than it would take in real life.

Table 6-1: Computation time of FAST.Farm and the FAST.Farm & MATLAB interface.

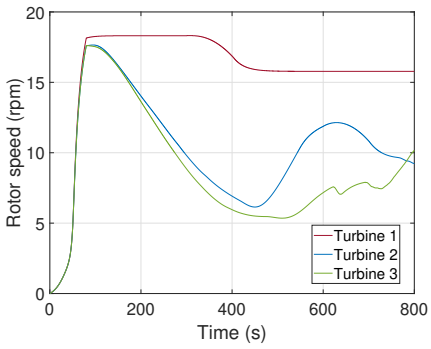
	FAST.Farm	MATLAB interface
Total Real Time	3.06 min	3.24 min
Total CPU Time	28.9 min	29.0 min
Simulation CPU Time	28.8 min	29.0 min
Simulated Time	10.0 min	10.0 min
Time Ratio (Sim/CPU)	0.347	0.345

6-3 FAST.Farm Simulation Results of Yaw Controller

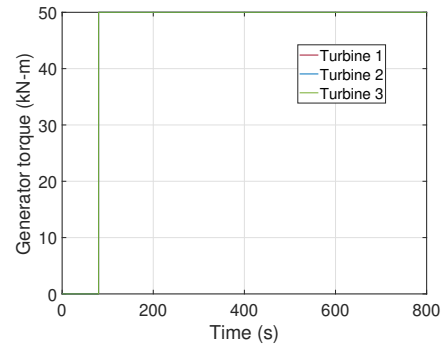
In this section three aligned 10 Megawatts (MW) wind turbines (DTU 10-MW) have been simulated in FAST.Farm. In the first experiment the turbines are spaced 500 meters apart, with an inflow wind speed of 12 m/s. The aim of the first experiment is to compare the computation time between a Fortran compiled controller and a MATLAB compiled controller. The yaw controller outlined in section 5-2-1 is implemented in Fortran DLLs and in the MATLAB interface. The simulation results of the yaw controllers are plotted in figure 6-1. The yaw controller yaws the first and second most upstream turbines at a 30-degree angle with respect to the incoming wind. The first turbine starts yawing after 300 seconds, the second turbine starts yawing after 600 seconds. As supposed, the simulation results of the yaw controller designed in the MATLAB interface matches the simulation results of the yaw controller implemented in Fortran language DLLs. Both controllers follow the same control logic. These results validate the exchange of the avrSWAP matrix between FAST.Farm and MATLAB. In the second experiment the turbines are spaced 1000 meters apart, with an inflow wind speed of 10 m/s. In addition, the yaw controller is enhanced with the optimum torque control law. A first-order low-pass filter has been implemented to filter the generator speed. The aim of this experiment is to show the behavior of a wake steering controller in FAST.Farm. The controller is only implemented in the MATLAB interface. The simulation results of the second experiment are plotted in Figure 6-2. Only the first most upstream turbine starts yawing at a 30-degree angle after 1400 seconds. By yawing the first turbine the overall power generation of the small wind farm increased. The simulation time has been extended with respect to the first experiment in order to reach a steady state before yawing. Figure 6-3 visualizes the wake steering simulation results with ParaView. This experiment shows that FAST.Farm can be used for simulating wake steering algorithms. Next subsections zoom in on the simulation results and computation time.



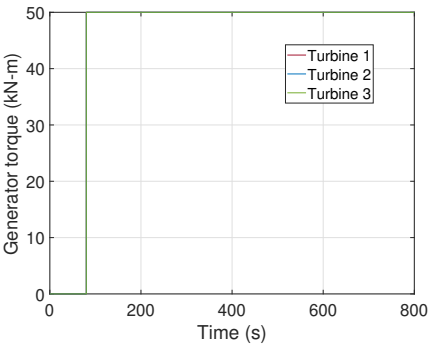
(a) Rotor speed Fortran



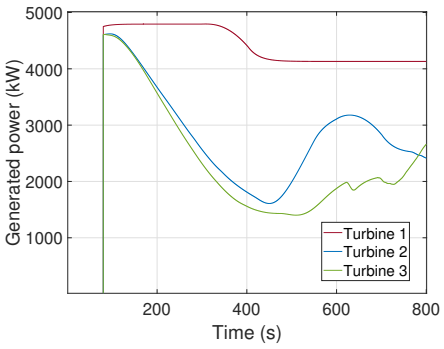
(b) Rotor speed MATLAB



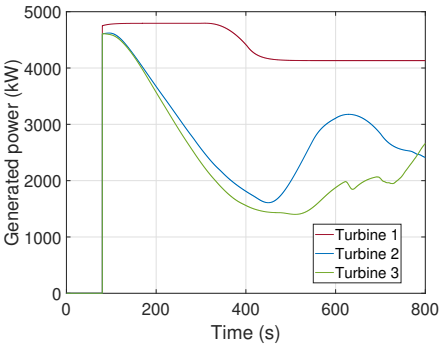
(c) Generator torque Fortran



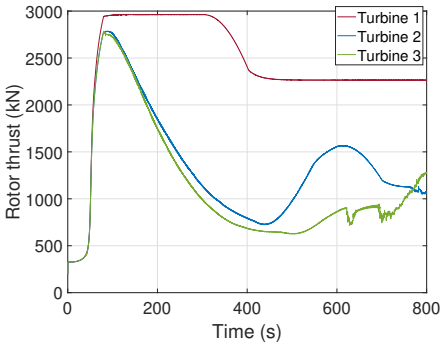
(d) Generator torque MATLAB



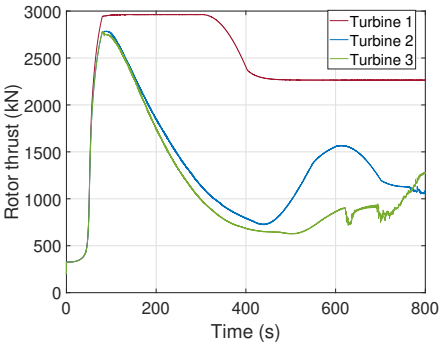
(e) Power generation Fortran



(f) Power generation MATLAB

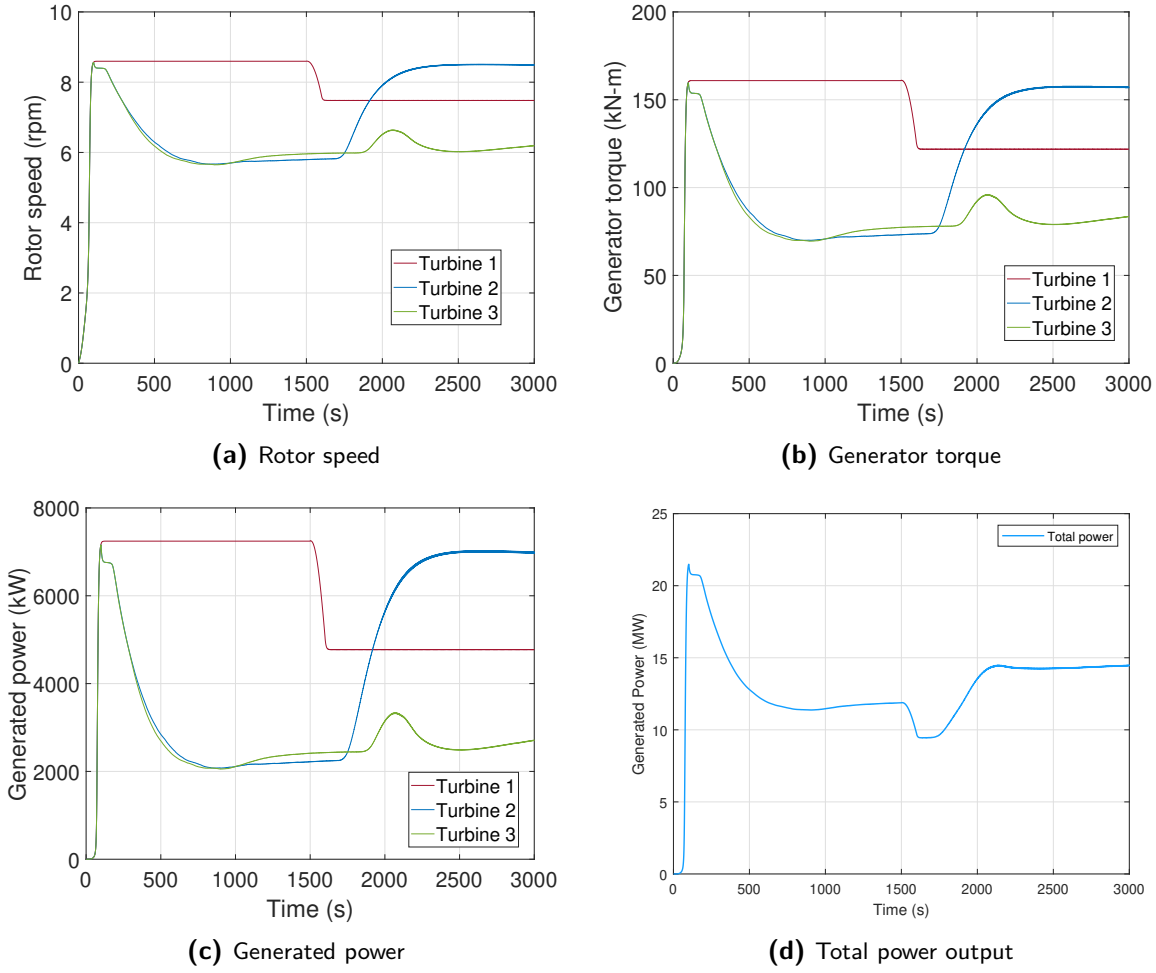


(g) Rotor thrust Fortran



(h) Rotor thrust MATLAB

Figure 6-1: FAST.Farm simulation results of yaw controller implemented in a Fortran DLL and in the MATLAB interface.



(a) Rotor speed

(b) Generator torque

(c) Generated power

(d) Total power output

Figure 6-2: FAST.Farm simulation results of yaw controller with optimum torque control.

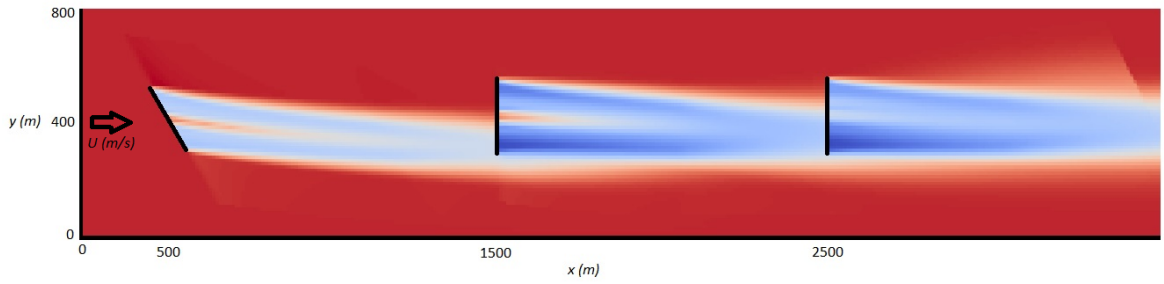


Figure 6-3: An instantaneous horizontal slice of flow output taken from FAST.Farm. The first (most left) turbine is yawed at a 30-degree angle with respect to the incoming flow field.

6-3-1 Implementation of Yaw Controller in MATLAB and Fortran DLL

Yaw Controller in Fortran DLL

A Super Controller (SC) Dynamic Link Library (DLL) has been created in Fortran language containing a simple yaw controller which dispatches yaw reference angles to the Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST) instances. The OpenFAST instances contain the so-called DISCON DLL. Subsequently, in this DISCON DLL a yaw-rate controller has been designed in order to control the yaw angle. The DISCON DLL determines the yaw rate based on the desired yaw angles from the SC-DLL. So, two Fortran language DLLs have been created in order to implement the yaw controller.

Yaw Controller in MATLAB

Exactly the same SC has been created in the MATLAB interface. The MATLAB interface contains a farm-level loop and a turbine-level loop. The farm-level loop contains the same controller logic as the Fortran SC-DLL. The turbine level loop contains the same controller logic as the Fortran DISCON DLL. MATLAB and Fortran show exactly the same results, this validates the exchange of the avrSWAP matrix between MATLAB and FAST.Farm.

6-3-2 Computation Time Fortran DLL vs MATLAB Interface

Fortran is a compiled language made for performance, in contradiction to MATLAB. This means that wind farm controllers simulated in a Fortran DLL should have a lower computation time with respect to wind farm controllers simulated in the MATLAB interface. Table 6-2 shows the computation times of the yaw controller implemented in both a Fortran DLL and the MATLAB interface. Values may differ from machine to machine. This table shows that the yaw controllers have the same computation time. The designed FAST.Farm and MATLAB interface does not increase the total computation time in this situation. More mathematical advanced optimization based controllers will most probably increase the computation time of simulations more when using the MATLAB interface compared to using Fortran DLLs. The goal of the MATLAB/Simulink interface was to avoid designing controllers in Fortran language DLLs, no advanced controllers were designed in nor compared with a Fortran DLL.

Table 6-2: Computation time Fortran DLL vs MATLAB interface.

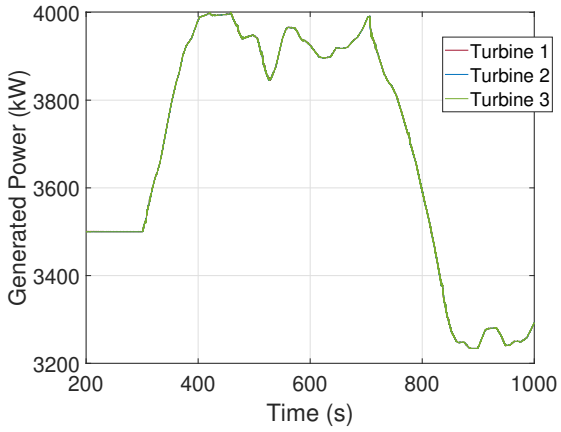
	Fortran DLL	MATLAB interface
Total Real Time	0.243 hours	0.248 hours
Total CPU Time	1.337 hours	1.338 hours
Simulation CPU Time	1.337 hours	1.338 hours
Simulated Time	10 min	10 min
Time Ratio (Sim/CPU)	0.166	0.166

6-4 FAST.Farm Simulation Results of Active Power Controller

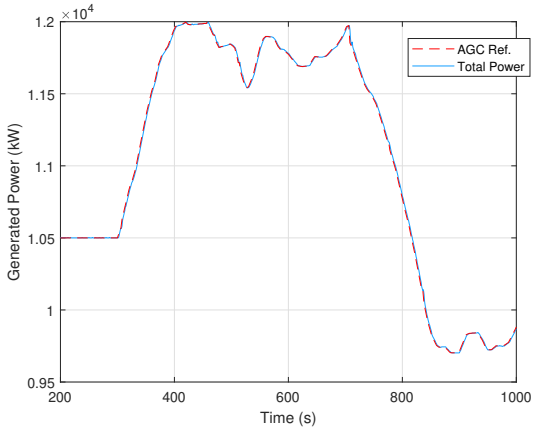
This section shows the simulation results of the Active Power Controller (APC) introduced in chapter 5. The wind farm setup consists of three 10 MW wind turbines (DTU 10-MW) spaced 500 meters apart. The APC is simulated in FAST.Farm with the use of the MATLAB interface. Figure 6-4 shows the simulation results with a steady inflow wind speed of 12 m/s. No turbine saturation occurs in this situation. Figure 6-5 shows the simulation results with a steady inflow wind speed of 10 m/s. In this situation turbine saturation occurs with the second and third turbine. Therefore, the wind farm controller redistributes the power references. In the saturation scenario, the rotor thrust forces on all turbines are (much) higher. Lower blade pitch angles (blades that are more faced into the wind) result in higher thrust forces on the turbines. Moreover, turbine one and two are boosted, by means of lower blade pitch angles, to compensate for the loss of power generation of turbine three. This is not an optimal situation due to the relatively high rotor thrust forces. Thrust force balancing and thrust force limiting could be used to lower the overall thrust forces while maintaining the same power output. These results validate as well the operation of the interface. By tracking the reference Automatic Generation Control (AGC) signal, the interface demonstrates that the avrSWAP matrix is exchanged between FAST.Farm and MATLAB. The rotor speed, generator power, collective blade pitch angle, and rotor thrust give a clear summary of the behavior of the wind farm. The first 200 seconds of the simulation are omitted, as some time is needed to start up the turbines and develop and propagate the wakes. The direction of the inflow wind field does not change over time. Table 6-3 shows the computation time for these simulations. This controller is not implemented in a Fortran DLL, direct comparison in computation time is not possible. The next subsections outline the settings of FAST.Farm. For a detailed overview of all the variables and parameters used in these simulations see GitHub [73].

Table 6-3: Computation time for APC simulated in FAST.Farm with MATLAB interface.

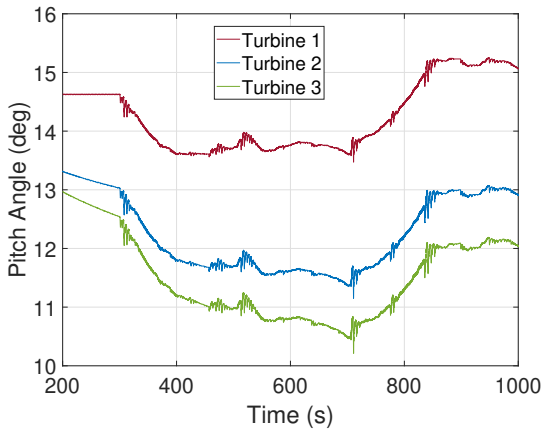
	Active Power Controller
Total Real Time	0.305 hours
Total CPU Time	1.695 hours
Simulation CPU Time	1.695 hours
Simulated Time	0.278 hours
Time Ratio (Sim/CPU)	0.164



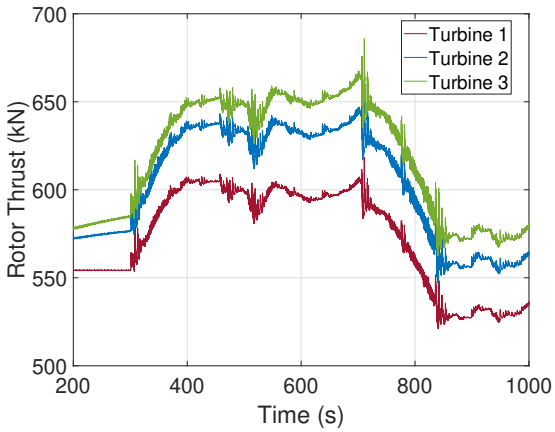
(a) Generated power



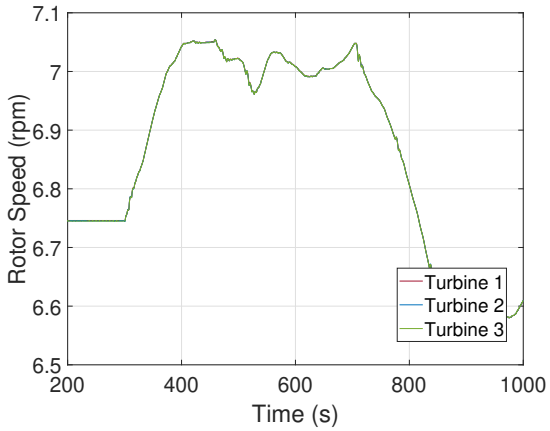
(b) Total power output



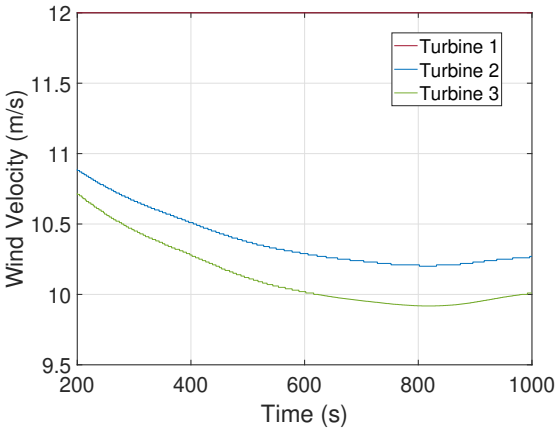
(c) Pitch angle



(d) Rotor thrust

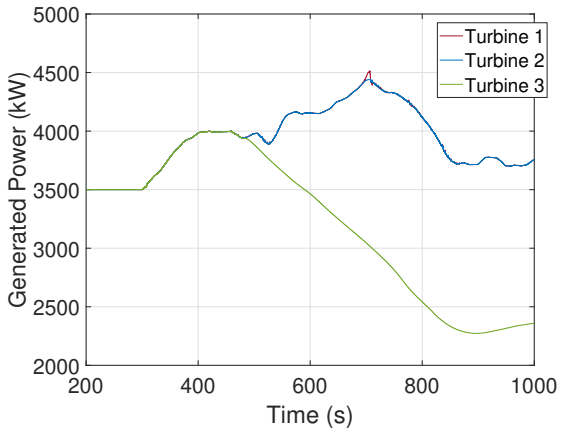


(e) Rotor speed

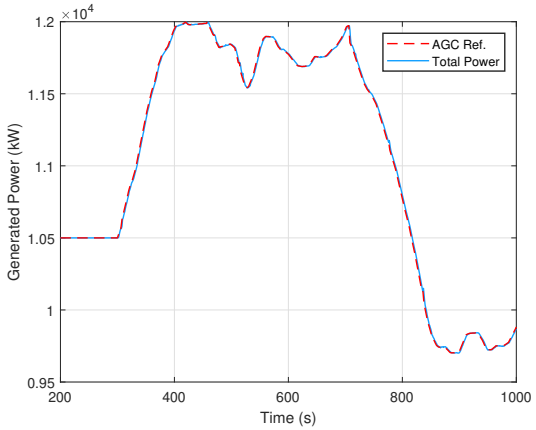


(f) Wind speed

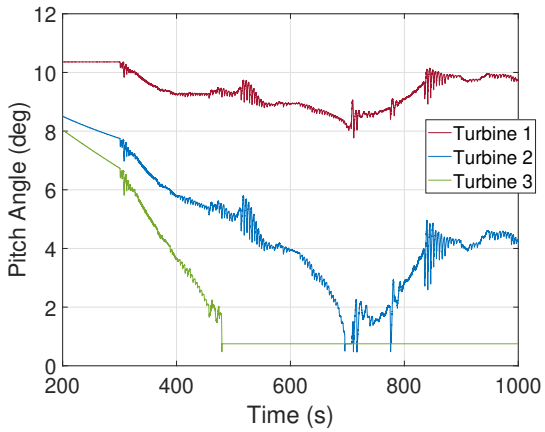
Figure 6-4: Simulation results of APC simulated in FAST.Farm with an steady inflow wind speed of 12 m/s.



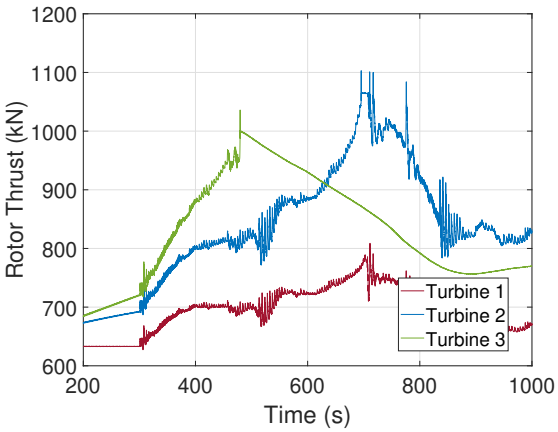
(a) Generated power



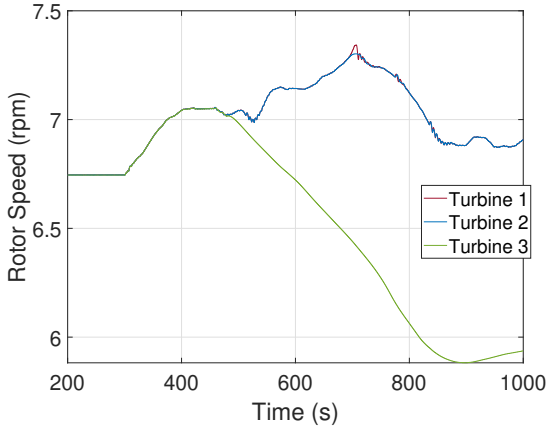
(b) Total power output



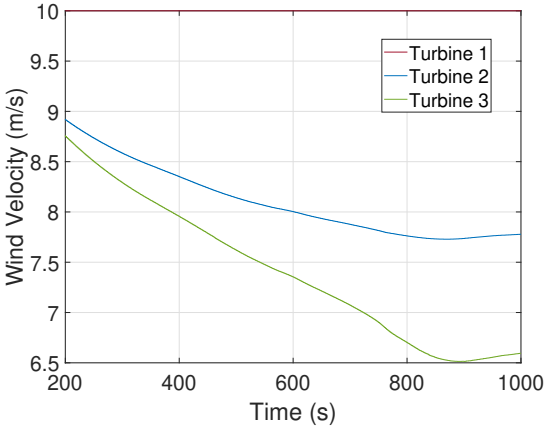
(c) Pitch angle



(d) Rotor thrust



(e) Rotor speed



(f) Wind speed

Figure 6-5: Simulation results of APC simulated in FAST.Farm with an steady inflow wind speed of 10 m/s.

6-4-1 Simulation Domain Settings in FAST.Farm

Table 6-4 states the simulation domain settings of FAST.Farm applied during the simulation of the APC. These settings were selected in order to balance the need for accurate simulations while restraining computation time. The FAST.Farm simulation domain consist of a low- and high-resolution domain. Both domains have their own specific simulation time-step and spacing between spatial nodes in X, Y, and Z direction. Three wind turbines are spaced 500 meters apart, see Figure 6-6 for a visualization of the wind turbines in the low- and high-resolution domain. The low resolution domain spans the whole area. The high-resolution domain covers a rectangle with sides of 340 by 500 meters with the center of the hub as middle-point of the rectangle. The time-step for low- and high-resolution wind data interpolation can be approximated with the following formulas:

$$DT\_Low \leq \frac{C_{\text{Meander}} D^{\text{Wake}}}{10 V_{\text{Hub}}} \tag{6-1}$$

$$DT\_High \leq \frac{1}{2 f_{\max}} \tag{6-2}$$

where C_{Meander} stands for the spatial filter model for wake meandering, this value (no dimension) is by default 1.9 [6]. D_{Wake} can be approximated as the diameter of the rotor, the rotor has a diameter of 178 meters. V_{Hub} is the mean wind speed at hub height, which is 10 m/s. $f_{\max}$ represents the maximum excitation frequency, which is around 1 Hz [74]. This means that *DT_Low* and *DT_High* should be, respectively, under 3 and 0.5 seconds. DT is the OpenFAST module time-step [7]. This time-step is often smaller than *DT_High*. DT of OpenFAST should be low enough that it does not influence the simulation output, but not lower than needed. In general, DT should be lower than 0.1 second. Finding the right DT is a process of trial and error. Note that DT and *DT_High* should be an integer multiple of *DT_Low*, see Figure 3-3. The spacing of low- and high-resolution spatial nodes in X, Y, and Z direction for wind data interpolation can be determined with the following formulas:

$$DS\_Low \leq \frac{C_{\text{Meander}} D^{\text{Wake}} V_{\text{Hub}}}{150} = \frac{DT\_Low V_{\text{Hub}}^2}{15} \tag{6-3}$$

$$DS\_High \leq c_{\max} \tag{6-4}$$

$c_{\max}$ denotes the maximum blade chord length of the DTU 10-MW turbine, which is 6 meters. This means that *DS_Low* and *DS_High* should be, respectively, under 20 and 6 meters. However, *DS_High* could be increased to 10, according to National Renewable Energy Laboratory (NREL) [6]. Simulation results indicate no observable variation in output between a *DS_High* of 5 or 10.

Note that the mentioned formulas only provide upper bounds. The simulation output would be more accurate (but also more computational expensive) if *DT_Low*, *DT_High*, *DS_Low*, *DS_High*, and DT are decreased with respect to the upper bounds. In the input files of FAST.Farm and the OpenFAST modules many more simulation variables and parameters can be modified, e.g. the wind turbine properties. For a detailed overview of the simulation domain settings in FAST.Farm see GitHub [73].

Table 6-4: FAST.Farm simulation settings.

Parameter	Description	Value
DT_Fast	Time-step in OpenFAST module (s)	0.05
DT_High	FAST.Farm high resolution domain time-step (s)	0.2
DT_Low	FAST.Farm high resolution domain time-step (s)	2.0
NX_Low	Number of low-resolution spatial nodes in X direction (-)	200
NY_Low	Number of low-resolution spatial nodes in Y direction (-)	100
NZ_Low	Number of low-resolution spatial nodes in Z direction (-)	65
NX_High	Number of high-resolution spatial nodes in X direction (-)	51
NY_High	Number of high-resolution spatial nodes in Y direction (-)	35
NZ_High	Number of high-resolution spatial nodes in Z direction (-)	25
dX_Low	Spacing of low-resolution spatial nodes in X direction (m)	10
dY_Low	Spacing of low-resolution spatial nodes in Y direction (m)	10
dZ_Low	Spacing of low-resolution spatial nodes in Z direction (m)	10
dX_High	Spacing of high-resolution spatial nodes in X direction (m)	10
dY_High	Spacing of high-resolution spatial nodes in Y direction (m)	10
dZ_High	Spacing of high-resolution spatial nodes in Z direction (m)	10

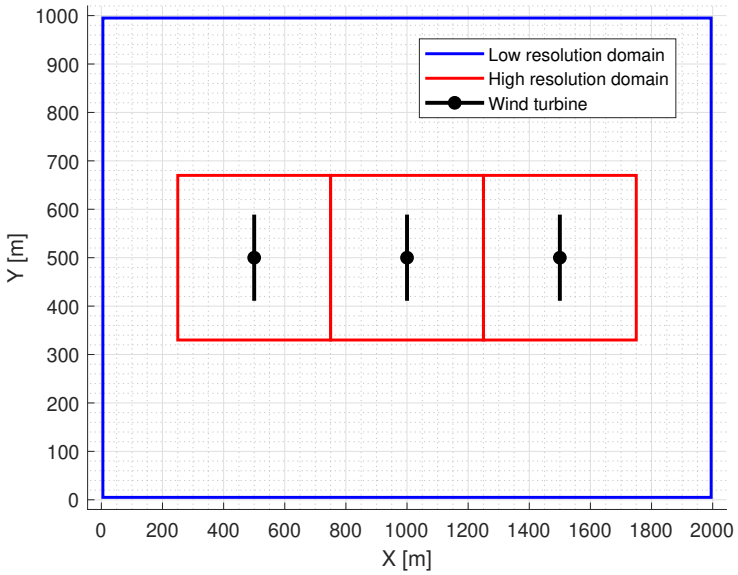


Figure 6-6: A top 2D view (X-Y) of the domains and the wind farm.

6-4-2 Wake Propagation Settings in FAST.Farm

The wake dynamic parameters affect wake propagation simulation in FAST.Farm. The number of wake planes, *NumPlanes*; radial increment of radial finite-difference grid, *dr*; and number of radii in the radial finite-difference grid, *NumRadii*, have to be defined carefully in the FAST.Farm input file [6] in order to realize accurate wake propagation in FAST.Farm. See Table 6-5 for the wake dynamic parameters. In addition, the cut-off frequency of the

low-pass time-filter f_c could be increased to 0.1 Hz, recommended by NREL [6]. The default value of f_c seems to be to low. For accurate computations about the wake deficits, the number of radii should be set so that the diameter of the wake planes is large relative to the rotor diameter. This relationship is captured in the following formula:

$$NumRadii \geq \frac{3D^{\text{Rotor}}}{2dr} + 1 \tag{6-5}$$

where D^{Rotor}, in [m], stands for the diameter of the rotor. dr sets the radial increment, it determines the space, in [m], between the radial nodes in the wake plane. It is suggested that $dr \leq c_{\max}$ (maximum turbine chord length). *NumPlanes* should be set high enough so that the wake planes propagate a sufficient distance downstream. The following formula is suggested for determining the number of wake planes:

$$NumPlanes \geq \frac{x_{\text{dist}}}{DT\_LowV} \tag{6-6}$$

x_{dist} represents the distance, in [m], until the wake deficit decays away, this is typically between 10 and 20 times the rotor diameter. *DT_Low* is the low-resolution time step. V stands for the average convection speed of the wake, in [m/s]. V can be approximated as:

$$V = V_{\text{Hub}}(1 - \frac{a}{2}) \tag{6-7}$$

V_{Hub} is the mean hub-height wind speed. a stands for the time- and spatial-temporal-average of the axial induction at the rotor disk. a is between 1/3 and 0, depending on the rated wind speed. V is expected to be around 9 m/s (V_{Hub} = 10 m/s and a = 0.167). This results into 200 wakes planes, if we set x_{dist} to be 20 times the rotor diameter (178 meters) and DT_LOW to 2.0 seconds. dr is set equal to the maximum turbine chord length of the DTU 10-MW, which is 6 meters. This implies that *NumRadii* has to be at least 46.

The above mentioned wake dynamic parameters, see Table 6-5, have to be adjusted case specific. Furthermore, the FAST.Farm input file contains additional parameters related to wake effects. These additional parameters, e.g. 'C_HWKDfl_O: calibrated parameter in the correction for wake deflection', were predefined in the FAST.Farm input file [6]. These parameters were calibrated on other SOWFA simulations, see the papers of Jonkman and Doubrawa [64, 65]. Calibrating all the (wake dynamic) parameters on SOWFA simulations is not in the scope of this research. Besides, not only do wake dynamic parameters affect wake propagation, the developing of wakes is also influenced by the wind turbine model. For a detailed overview of the wake propagation settings in FAST.Farm see GitHub [73].

Table 6-5: Wake dynamic parameters in FAST.Farm.

Parameter	Description	Value
dr	Radial increment of radial finite-difference grid (m)	6
NumRadii	Number of radii in the radial finite-difference grid (m)	50
NumPlanes	Number of wake planes (-)	200
f_c	Cut-off (corner) frequency of the low-pass time-filter (Hz)	0.1

6-5 Comparison Between FAST.Farm and SOWFA Simulations

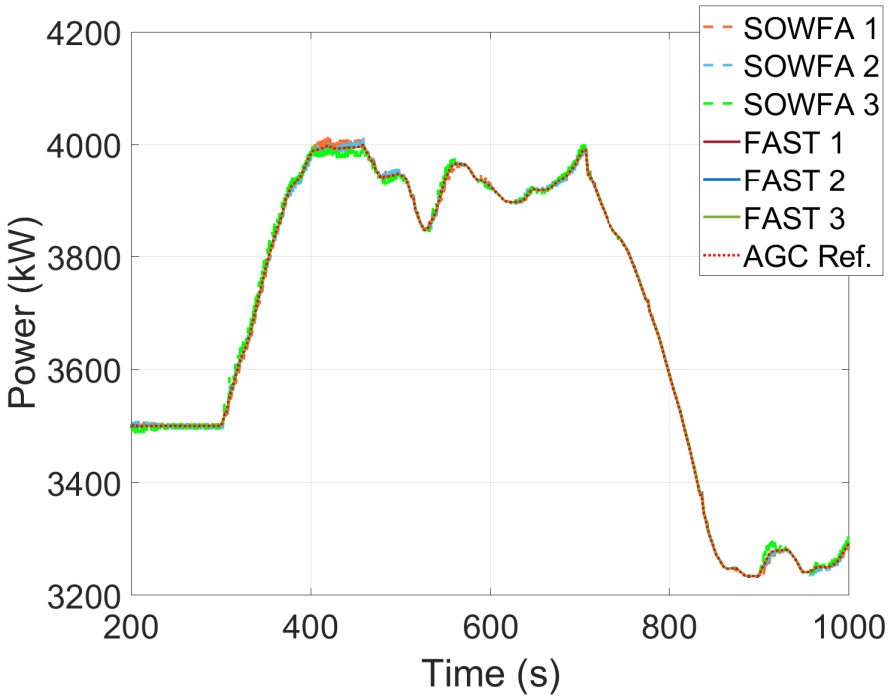
SOWFA is a high-fidelity simulation tool used for simulating wind farms. SOWFA simulations are often used as validation data [53, 66]. The APC simulation results illustrated in the previous section have been compared with the simulation results of the same controller simulated in SOWFA. Figure 6-7 shows the simulation results of SOWFA and FAST.Farm. A steady inflow wind speed of 12 m/s is applied during the simulations. The simulation results do not exactly correspond, but show the same trend. The goal of the APC is to track the reference signal, in both simulation tools the power output matches the reference signal. The differences, in e.g. pitch angle, seem to depend on the differences in the mathematical model behind both tools. The SOWFA simulations show a bit more oscillation in the power generation, thrust forces, and pitch angles. This is because in SOWFA the tower of the turbine influences the wind flow around the turbine. In FAST.Farm the tower is omitted from the simulation. In both tools the Actuator Line Model (ALM) is used as internal turbine model.

6-5-1 Settings of SOWFA

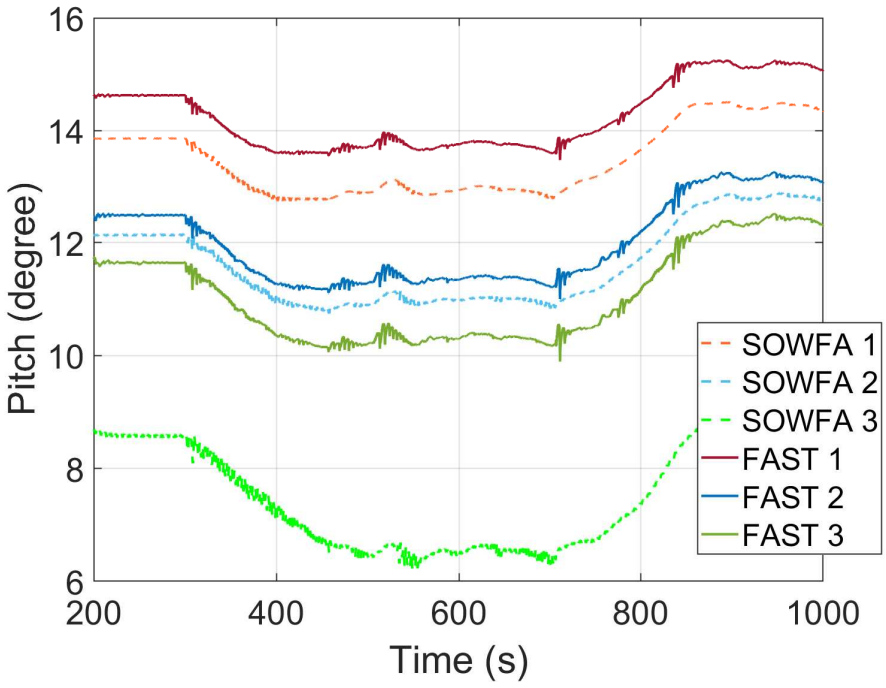
Table 6-6 shows the settings of SOWFA used during the simulation. SOWFA and FAST.Farm employ distinct mathematical models for estimating wind farm performance. So, simulation domain and wake dynamic parameters in SOWFA differ from FAST.Farm. SOWFA does not have different timescale ranges. The lower-resolution domain time-step, higher-resolution domain time-step, OpenFAST time-step, and controller-level time-steps are executed at the same frequency. Even the turbine-level and farm-level loop are executed during every iteration (although, different timescales for the turbine-level and farm-level loop could be manually programmed in the controller script). In addition, the simulation domain in SOWFA consists of four different sized domains, whereas FAST.Farm has only two different sized domains. In SOWFA the difference in domains is based on the spacing between the spatial nodes in X, Y, and Z direction. In FAST.Farm the difference in domains is based on the spacing between the spatial nodes and on the low- and high-resolution domain time-step. For this comparison the turbine-level loop and farm-level loop inside the controller (see Figure 5-1) are executed at the same sampling frequency in both tools. This means that DT_SOWFA and DT_FAST are equal. In addition, the size of the low- and high-resolution domains in SOWFA resemble the domains in FAST.Farm.

Table 6-6: SOWFA simulation settings.

Parameter	Description	Value
DT_SOWFA	Time-step of SOWFA (s)	0.05
D_Low	Spacing of rough grid size (m)	10
D_MedLow	Spacing of rough grid size (m)	5
D_MedHigh	Spacing of rough grid size (m)	2.5
D_High	Spacing of fine grid size (m)	1.25
KinVisc	Kinematic air viscosity (m^2/s)	1.0E-5

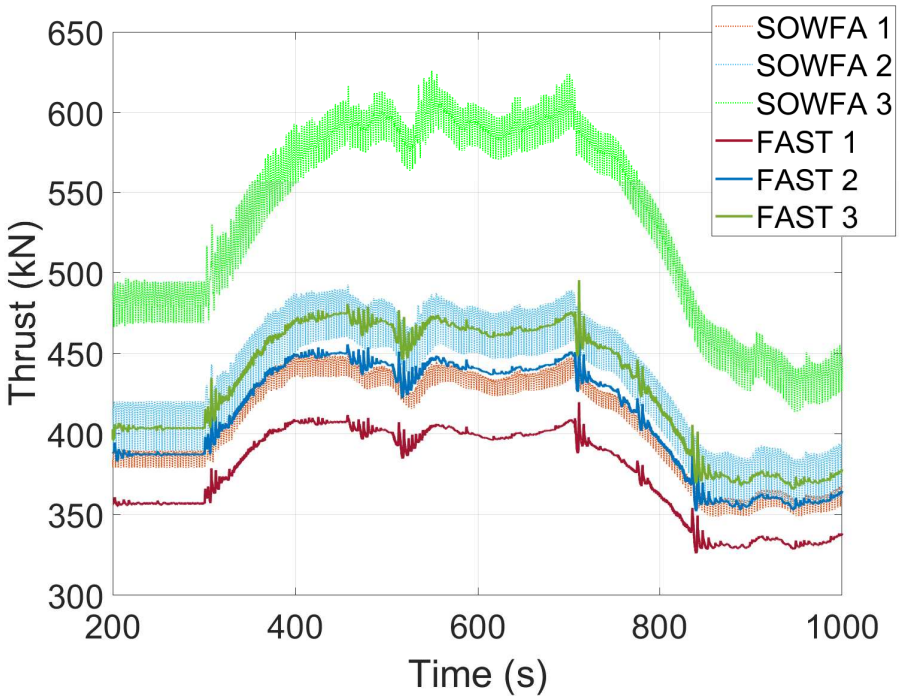


(a) Generated power



(b) Pitch angle

Figure 6-7: Simulation results APC simulated in SOWFA and FAST.Farm.



(c) Rotor thrust

Figure 6-7: Simulation results APC simulated in SOWFA and FAST.Farm.

6-5-2 Computation Time Comparison

The computation time for SOWFA simulations are in the order of days. The computation time for this SOWFA simulation was around 24 hours on a cluster of 40 computers (processors). FAST.Farm simulations would have a computation time in the order of minutes. The computation time for this FAST.Farm simulations was around 17 minutes on a normal desktop computer. The computation time between those two programs differs greatly, although the simulation time is the same. Keep in mind that the computation time depends greatly on the device used for execution the calculations. In addition, the settings in both programs, e.g. the time-step and spacing of spatial nodes, also influence the computation time. The difference in computation time lies in the fact that SOWFA resolves the governing equations on a 3-Dimension (3D) scale, FAST.Farm uses 2-Dimension (2D) Navier-Stokes (NS) equations to solve the flow field dynamics. FAST.Farm is preferred in terms of computation time, but lacks the level of fidelity of SOWFA.

6-5-3 Discussion About Differences in Simulation Results

Figure 6-7 shows the simulation results of the APC simulated in SOWFA and FAST.Farm. SOWFA simulation results are used for validation purpose. The pitch angles, thrust forces, and rotor speeds of the FAST.Farm simulation do not exactly comply with the simulation results from SOWFA. However, the trend in both graphs is the same. Both (identical) controllers realize the same power generation. As the objective of the APC is realized in both

simulations, the differences in pitch angles, thrust forces, and rotor speeds depends on the differences between the underlying mathematical models of SOWFA and FAST.Farm. The mathematical models of SOWFA and FAST.Farm differ in the following respects:

- **Nature of Simulation Tool:** FAST.Farm and SOWFA have distinct underlying mathematical models. SOWFA resolves the governing equations for estimating the flow field on a 3D scale, FAST.Farm uses 2D NS equations to solve the flow field dynamics. Consequently, this would lead to different flow field simulations, thereby affecting the response of the controller and the wind turbines. Wake estimations become less accurate further downstream the farm in FAST.Farm simulations. This is seen, in Figure 6-7, in the increasing deviation of the blade pitch angle and thrust force of turbine three with respect to the deviation of turbine one and two.
- **Different Calibrated Turbine and Flow Model:** The ALM is used as turbine model in SOWFA. The turbine model in FAST.Farm is calibrated on SOWFA simulations using the ALM. But, the calibrated ALM in FAST.Farm is calibrated on a different (but comparable) set of SOWFA simulations [64, 65]. This results in slightly different behavior of the turbine models in both simulation tools. Just like the turbine model, the flow dynamics of FAST.Farm are also calibrated on SOWFA simulations. The SOWFA simulations used for calibrating the flow dynamics of FAST.Farm differ from the simulation conducted in this thesis. The biggest difference is that the flow dynamics of FAST.Farm are calibrated on turbulent wind inflow simulations, while in this thesis steady wind inflow is used.
- **Inflow Wind Module** In both simulation tools a steady inflow wind speed of 12 m/s is applied. However, the wind inflow in SOWFA is in 3D (VTK files), the wind inflow in FAST.Farm is in 2D. This difference in inflow flow field could affect the simulation results. However, more experiments are needed in order to validate or reject this claim.

6-6 Summary

This chapter has visualized the simulation results of different wind farm controllers simulated in FAST.Farm and SOWFA. The MATLAB interface seems a promising tool in terms of computation time. The MATLAB interface does not increase the total computation time of a simple yaw controller with respect to the same controller implemented in Fortran language DLLs. Moreover, instead of waiting 24 hours for a 10-minute simulation in SOWFA, FAST.Farm can simulate the same event in 15 minutes. This will speed up the design process. The simulation results of the APC in FAST.Farm and SOWFA show that FAST.Farm and SOWFA behave slightly different. This was already expected as both tools work differently. However, both tools show the same trend in the simulation results. The next chapter discusses the observations made in this thesis.

Chapter 7

Discussion

7-1 Introduction

This chapter discusses the main findings of this thesis. The Fatigue, Aerodynamics, Structures, and Turbulence Farm tool (FAST.Farm) and MATLAB/Simulink interface supports the development and testing of advanced control at the wind farm level. However, Simulator fOr Wind Farm Applications (SOWFA) is still preferred as validation tool.

7-2 Discussion

The aim of this thesis was first, the development of a FAST.Farm and MATLAB/Simulink interface, and second, a comparison of the FAST.Farm and SOWFA simulation results of an Active Power Controller (APC). The working of the interface has been validated in chapter 6. However, this interface can still be improved. For example, a dashboard could make the ease of use of the interface clearer. In addition, this interface could be 'interfaced' with other tools/GitHubs in order to increase the applicability. FAST.Farm seems not to be able to replace SOWFA. But, by using FAST.Farm for designing controllers, less iterations with SOWFA are required. This has the potential to significantly reduce the design process time, as the computational time of FAST.Farm is 1,000 times less than that of SOWFA. The next subsections explain the ease of use of the interface and the features of FAST.Farm.

7-2-1 Ease of Use of Interface

The interface could be improved in terms of ease of use. On the FAST.Farm side, different input files exist for all the different modules of FAST.Farm and Open Fatigue, Aerodynamics, Structures, and Turbulence tool (OpenFAST). In addition, several intermediate steps have to be taken, e.g. writing a (*.bat*) file for starting the simulation, in order to simulate a controller inside FAST.Farm. The interface of FAST.Farm and OpenFAST itself is also not easy in use.

ServoDyn, Elastody, and Aerodyn are examples of input files for OpenFAST [7], for every wind turbine in FAST.Farm input parameters have to be defined for these files. On top of that, new versions of FAST.Farm (and OpenFAST) are released regularly which sometimes disable old features. FAST.Farm and OpenFAST have been developed by National Renewable Energy Laboratory (NREL), they are responsible for the ease of use of these programs.

The MATLAB/Simulink interface is more intuitive in use than FAST.Farm, as shown in Algorithms 1 and 2. However, several programs have to be installed in order to use the interface, see GitHub [24] for a detailed explanation. Controllers can be constructed on the designated lines in the interface. These controllers can use all the information provided by the avrSWAP matrix. To facilitate a connection between FAST.Farm and MATLAB, both programs have to be connected to the same internal port through the Message Passing Interface (MPI) and MATLAB Executable (MEX) functions. First MATLAB has to 'open' a port, then FAST.Farm can connect to that port. This process can be automated by writing a (*.bat*) file which automatically starts the programs sequentially. The interface could be improved by integrating the input files of FAST.Farm and OpenFAST in MATLAB/Simulink. In this way all parameters can be defined in one script.

7-2-2 Features FAST.Farm vs SOWFA

FAST.Farm and SOWFA have distinct underlying mathematical models resulting in (slightly) different outputs for the same simulation setup. FAST.Farm offers the possibilities to incorporate more information about the tower dynamics in wind farm simulations. This could lead to more accurate turbine structural load estimations compared to SOWFA. However, SOWFA provides more precise estimations of the flow field throughout the simulation domain. Flow field estimations highly affect the simulation results of the wind turbines. Overall, this makes the simulation output of SOWFA more reliable than FAST.Farm. Having access to more information about the structural behavior of turbines does not necessarily result in more accurate/robust wind farm controllers [75]. Further research is needed to improve flow field estimations in FAST.Farm. The simulation results in section 6.5 support this assumption. On the other hand, the computation time for a 10-minute simulation in FAST.Farm is around 15 minutes, while this would be around 24 hours in SOWFA. This makes FAST.Farm an interesting tool in the (early phase of the) design process of wind farm controllers.

7-3 Recommendations for Future Work

The construction of the FAST.Farm and MATLAB/Simulink interface has simplified the implementation of wind farm controllers in FAST.Farm. Controllers that were already constructed in MATLAB/Simulink can be easily implemented and simulated in FAST.Farm. Previously, FAST.Farm was barely used at the TU Delft for designing and testing controllers. This thesis work has paved the way for using FAST.Farm for designing and testing novel controllers. Possible future research subjects could focus on:

- Extending the APC outlined in chapter 5. This controller has been designed and tested in SOWFA solely. FAST.Farm provides the option to present additional information regarding structural loads. With this extra information the controller can be

improved. For example, tower vibrations could cause oscillations that were not simulated in SOWFA. With FAST.Farm the APC could be designed more robust. However, dozens of promising wind farm controller have already been proposed in literature [39]. I believe future research should focus more on field testing novel wind farm controllers [41].

- Designing and implementing closed loop wind farm controllers in the FAST.Farm and MATLAB/Simulink interface. The development of wind farm controllers progresses, as the number and size of wind farms increase. FAST.Farm can be used in the design phase prior to validation in SOWFA.
- Enhancing FAST.Farm. The simulation results in Chapter 6 show that the estimations of the flow field dynamics in FAST.Farm can be improved. I would recommend to calibrate the flow field dynamics in the FAST.Farm input file on data from field test, besides the validation on large-eddy simulations [64]. Multiple sets of calibrated parameters could be constructed for different real-life scenarios.
- Designing a dashboard for the FAST.Farm and MATLAB/Simulink interface. The ease of use of the interface could be more simplified. Although designing controllers in the MATLAB/Simulink interface is much more intuitive than in Fortran Dynamic Link Libraries (DLL), The interface between FAST.Farm and the OpenFAST instances is cluttered.
- Linking this interface to the FarmConners Simulink-wfc-platform [76]. The FarmConners market showcases can be used to evaluate wind farm control algorithms and strategies for the current market landscape and possible future scenarios. By integrating the FAST.Farm and MATLAB/Simulink interface in the FarmConners platform, we can ensure greater exposure to the interface, allowing more people to benefit from it.

7-4 Summary

The development of the FAST.Farm and MATLAB/Simulink interface has led to the emergence of several intriguing research topics for further investigation. Although the ease of use of the interface could be improved, the interface simplifies the design and implementation of wind farm controllers in FAST.Farm. The next chapter concludes this thesis report.

Chapter 8

Conclusion

8-1 Conclusion

The aim of this thesis was to develop an interface between FAST.Farm and MATLAB/Simulink. In this thesis report the first, to the best of the author's knowledge, step in creating a full-purpose MATLAB/Simulink interface in FAST.Farm was presented. This interface can be downloaded from GitHub [24]. The capabilities of FAST.Farm for control design purposes have been extended through a co-simulation with MATLAB/Simulink. Controller commands and measurement data can be exchanged between FAST.Farm and MATLAB/Simulink by exchanging the avrSWAP matrix. Consequently, in MATLAB/Simulink controllers can be designed at both wind farm and turbine levels. This interface greatly simplifies the design and implementation of wind farm controllers in FAST.Farm.

The simulation results of the Yaw controller validate the functionality of the interface. The case study of the Active Power Controller (APC) demonstrates that MATLAB/Simulink can be used as an add-on to FAST.Farm. However, a comparison of FAST.Farm and SOWFA APC simulation results reveal that FAST.Farm could deviate in flow field estimations compared to SOWFA. These deviations can be attributed partially to variations in the underlying mathematical models and partly because FAST.Farm has been calibrated on a different set of SOWFA simulations [64]. On the other hand, the computation time for a 10-minute simulation in FAST.Farm is around 15 minutes, while it would take approximately 24 hours in SOWFA. This makes FAST.Farm an interesting tool in the (early phase of the) design process of wind farm controllers.

This interface supports the development and testing of advanced closed-loop controllers at the wind farm level. Computation time comparison experiments indicate that this interface does not affect the total computation time of FAST.Farm. However, it should be noted that computationally demanding controllers will likely increase the computation time of simulations more when using the MATLAB/Simulink interface compared to using Fortran dynamic link libraries. Because Fortran compiled code tends to be faster than MATLAB/Simulink compiled code.

Future work could focus on further implementation of closed-loop controllers at the wind farm level, as demonstrated in [15, 16]. The interface could be linked to the Simulink-wfc-platform from the FarmConners project [76] to extend its capabilities. Moreover, using wind farm models, such as FLORIS [76] and predictive controllers [77–79] could be explored.

Bibliography

[1] W.Tong, "Wind power generation and wind turbine design," *Wit Press*, 2010.

[2] M. Perry, J. McAlorum, G. Fusiek, P. Niewczas, I. Mckeeman, and T. Rubert, "Crack monitoring of operational wind turbine foundations," *Sensors*, 2017.

[3] C. B. Hasager, L. Rasmussen, A. Peña, L. E. Jensen, and P.-E. Réthoré, "Wind farm wake: The horns rev photo case," *Energies, 6, 696-716*, 2013.

[4] D. S. Zalkind and L. Y. Pao, "The fatigue load effects of yaw control for wind plants," *American Control Conference*, 2016.

[5] J. A. Frederik, B. M. Doekemeijer, S. P. Mulders, and J.-W. van Wingerden, "The helix approach: Using dynamic individual pitch control to enhance wake mixing in wind farms," *Wind Energy, Volume 23*, 2020.

[6] NREL, "FAST.Farm." https://openfast.readthedocs.io/en/dev/source/user/fast.farm, 2022. Accessed: 2022-05-06.

[7] NREL, "OpenFAST documentation," *National Renewable Energy Laboratory (NREL)*, 2022.

[8] M. Meinshausen, J. Lewis, C. McGlade, J. Gütschow, Z. Nicholls, R. Burdon, L. Cozzi, and B. Hackmann, "Realization of paris agreement pledges may limit warming just below 2 c," *Nature*, vol. 604, no. 7905, pp. 304–309, 2022.

[9] J. Skea, P. Shukla, and S. Kilkis, *Climate Change 2022: Mitigation of Climate Change*. Cambridge University Press, Cambridge (MA), USA, 2022.

[10] Global Wind Energy Council, "Global wind report 2022." https://gwec.net/global-wind-report-2022/, 2022. Accessed: 2022-05-12.

[11] I. Komusanac, G. Brindley, D. Fraile, and L. Ramirez, "Wind energy in europe, 2020 statistics and the outlook for 2021-2025." https://windeurope.org/, 2021. Accessed: 2022-05-10.

[12] "Kabinet verdubbelt productie windenergie op zee." https://www.rijksoverheid.nl/actueel/nieuws/2022/03/18/kabinet-verdubbelt-productie-windenergie-op-zee. Accessed: 2022-05-05.

[13] L. Y. Pao and K. E. Johnson, "A tutorial on the dynamics and control of wind turbines and wind farms," *American Control Conference*, pp. 2076–2089, 2009.

[14] A. C. Kheirabadi and R. Nagamune, "A quantitative review of wind farm control with the objective of wind farm power maximization," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 192, pp. 45–73, 2019.

[15] B. M. Doekemeijer, D. van der Hoek, and J. W. van Wingerden, "Closed-loop model-based wind farm control using FLORIS under time-varying inflow conditions," *Renewable Energy*, vol. 156, pp. 719 – 730, 2020.

[16] J. G. Silva, R. Ferrari, and J.-W. van Wingerden, "Wind farm control for wake-loss compensation, thrust balancing and load-limiting of turbines," *Renewable Energy*, vol. 203, pp. 421–433, 2023.

[17] NREL, "FLORIS. version 3.2." https://www.nrel.gov/wind/floris.html, 2022. Accessed: 2022-05-20.

[18] NREL, "SOWFA." https://www.nrel.gov/wind/nwtc/sowfa.html, 2022. Accessed: 2022-05-05.

[19] NREL, "FAST.Farm github." https://github.com/OpenFAST/openfast, 2023. Accessed: 2023-02-11.

[20] J. M. Jonkman, J. Annoni, G. Hayman, B. Jonkman, and A. Purkayastha, "Development of FAST.Farm: A new multi-physics engineering tool for wind-farm design and analysis," *35th Wind Energy Symposium*, p. 0454, 2017.

[21] M. Kretschmer, J. Jonkman, V. Pettas, and P. W. Cheng, "Fast. farm load validation for single wake situations at alpha ventus," *Wind Energy Science*, vol. 6, pp. 1247–1262, 2021.

[22] P. Fleming, P. Gebraad, J. W. van Wingerden, S. Lee, M. Churchfield, A. Scholbrock, J. Michalakes, K. Johnson, and P. Moriarty, "SOWFA super-controller: A high-fidelity tool for evaluating wind plant control approaches," *Wind Energy Conference and Exhibition*, vol. 3, pp. 1561–70, 2013.

[23] DNV-GL, "Bladed." https://www.dnvgl.com/energy, 2022. Accessed: 2022-09-12.

[24] C.-J. Smits, V. Chabaud, J. G. Silva, and R. Ferrari, "FAST.Farm and MATLAB/Simulink interface." https://github.com/ValentinChb/FASTFarm2Simulink, 2023. Accessed: 2023-05-01.

[25] J. G. Silva, B. M. Doekemeijer, R. Ferrari, and J.-W. van Wingerden, "Active power control of wind farms: an instantaneous approach on waked conditions," *Journal of Physics: Conference Series*, vol. 2265, no. 2, p. 022056, 2022.

[26] C.-J. Smits, J. Gonzalez Silva, V. Chabaud, and R. Ferrari, "A FAST.Farm and MATLAB/Simulink interface for wind farm control design," *EERA DeepWind*, 2023.

[27] W. Tong, *Wind power generation and wind turbine design*. Wit Press, 2010.

[28] M. Miller, *The Multi-Objective Design of Flatback Wind Turbine Airfoils*. PhD thesis, Carleton University, 2016.

[29] F. Bianchi, H. D. Battista, and R. Mantz, "Wind turbine control systems; principles, modelling and gain scheduling design," *Springer-Verlag London*, 2007.

[30] R. Nash, R. Nouri, and A. Vasel-Be-Hagh, "Wind turbine wake control strategies: A review and concept proposal," *Energy Conversion and Management*, vol. 245, p. 114581, 2021.

[31] A. Gupta, M. A. Rotea, M. Chetan, M. S. Sakib, and D. T. Griffith, "Effect of wind turbine size on load reduction with active flow control," *Journal of Physics: Conference Series*, vol. 2265, p. 032093, may 2022.

[32] "World's largest, most powerful wind turbine stands complete." https://www.offshorewind.biz/2021/11/12/worlds-largest-most-powerful-wind-turbine-stands-complete/. Accessed: 2022-06-12.

[33] A. Betz, *Introduction to the Theory of Flow Machines*. Pergamon Press, 1966.

[34] F. Bianchi, H. D. Battista, and R. Mantz, *Wind Turbine Control Systems; Principles, modelling and gain scheduling design*. Springer Verlag London, 2007.

[35] M. Hansen, *Aerodynamics of wind turbines, third edition*. Routledge, 2015.

[36] G. Freebury, "Determining equivalent damage loading for full-scale wind turbine blade fatigue tests," *19th American Society of Mechanical Engineers (ASME) Wind Energy Symposium*, 2000.

[37] L. Martinez Tossas, E. Branlard, and J. Jonkman, "Wind turbine wakes under high thrust coefficients," *APS Division of Fluid Dynamics Meeting Abstracts*, 2020.

[38] S. Boersma, B. Doekemeijer, P. Gebraad, P. Fleming, J. Annoni, A. Scholbrock, J. Frederik, and J. W. van Wingerden, "A tutorial on control-oriented modeling and control of wind farms," *American Control Conference (ACC)*, 2017.

[39] L. E. Andersson, O. Anaya-Lara, J. O. Tande, K. O. Merz, and L. Imsland, "Wind farm control - part i: A review on control system concepts and structures," *IET Renew. Power Gener. 15, 2085–2108*, 2021.

[40] T. Knudsen, T. Bak, and M. Svenstrup, "Survey of wind farm control—power and fatigue optimization," *Wind Energy*, 2015.

[41] J. Wingerden, P. Fleming, T. Göçmen, I. Eguinoa, B. Doekemeijer, K. Dykes, M. Lawson, E. Simley, J. King, D. Astrain, M. Iribas, C. Bottasso, J. Meyers, S. Raach, K. Kölle, and G. Giebel, "Expert elicitation on wind farm control," *Journal of Physics: Conference Series*, vol. 1618, p. 022025, 09 2020.

[42] A. C. Kheirabadi and R. Nagamune, "Modeling and power optimization of floating offshore wind farms with yaw and induction-based turbine repositioning," *American Control Conference (ACC)*, pp. 5458–5463, 2019.

[43] N. Saadallah and E. Randeberg, "Dynamic repositioning in floating wind farms," *NORCE Norwegian Research Centre*, 2020.

[44] J. Annoni, P. M. O. Gebraad, A. Scholbrock, P. A. Fleming, and J. W. van Wingerden, "Analysis of axial-induction-based wind plant control using an engineering and a high-order wind plant model," *Wind Energy*, 2016.

[45] P. Gebraad, P. Fleming, and J. van Wingerden, "Comparison of actuation methods for wake control in wind plants," *American Control Conference*, 2015.

[46] P. M. O. Gebraad, J. J. Thomas, A. Ning, P. A. Fleming, and K. Dykes, "Maximization of the annual energy production of wind power plants by optimization of layout and yaw-based wake control," *Wind Energy*, 2016.

[47] K. Kimura, Y. Tanabe, Y. Matsuo, and M. Iida, "Forced wake meandering for rapid recovery of velocity deficits in a wind turbine wake," *AIAA Scitech Forum*, 2019.

[48] I. Katic, J. Hojstrup, and N. O. Jensen, "A simple model for cluster efficiency," *EWEC*, 1986.

[49] S. Frandsen, R. Barthelmie, S. Pryor, O. Rathmann, S. Larsen, J. Højstrup, and M. Thøgersen, "Analytical modelling of wind speed deficit in large wind farms," *Wind Energy*, vol. 9, pp. 39 – 53, 04 2006.

[50] P. M. O. Gebraad, P. A. Fleming, and J. W. van Wingerden, "Wind turbine wake estimation and control using floridyn, a control-oriented dynamic wind plant model," *American Control Conference (ACC)*, 2015.

[51] G. C. Larsen, H. A. Madsen, F. Bingoel, J. Mann, S. Ott, J. N. Sorensen, V. Okulov, N. Troldborg, N. M. Nielsen, K. Thomsen, K. Larsen, T. J. Larsen, and R. Mikkelsen, "Dynamic wake meandering modeling," *Riso National Lab., DTU, Roskilde (Denmark). Wind Energy Dept.*, 2007.

[52] J. F. Ainslie, "Calculating the flow field in the wake of wind turbines," *Journal of Wind Engineering and Industrial Aerodynamics*, 1988.

[53] S. Boersma, P. M. O. Gebraad, M. Vali, B. M. Doekemeijer, and J. W. van Wingerden, "A control-oriented dynamic wind farm flow model: Wfsim," *Torque*, 2016.

[54] J. Meyers, "Large eddy simulations of large wind-turbine arrays in the atmospheric boundary layer," *Aerospace Sciences Meeting*, 2010.

[55] H. Özdemir, M. C. Versteeg, and A. J. Brand, "Improvements in ecn wake model," *ICOWES conference*, 2013.

[56] L. A. Martinez-Tossas, M. J. Churchfield, and S. Leonardi, "Large eddy simulations of the flow past wind turbines: actuator line and disk modeling," *Wind Energy*, 2014.

[57] F. Campagnolo, A. Molder, J. Schreiber, and C. L. Bottasso, "Comparison of analytical wake models with wind tunnel data," *J. Phys.: Conf. Ser. 1256 012006*, 2019.

[58] A. Peña, P.-E. Réthoré, and M. P. van der Laan, "On the application of the jensen wake model using a turbulence-dependent wake decay coefficient: the sexbierum case," *Wind Energy*, vol. 19, no. 4, pp. 763–776, 2016.

[59] S. Boersma, M. Vali, M. Kühn, and J. W. van Wingerden, "Quasi linear parameter varying modeling for wind farm control using the 2d navier-stokes equations," *American Control Conference (ACC)*, 2016.

[60] M. Churchfield, S. Lee, and P. Moriarty, "Overview of the simulator for wind farm application (SOWFA)," *National Renewable Energy Laboratory (NREL)*, 2012.

[61] NREL, "Extended bladed interface." https://openfast.readthedocs.io/en/main/source/user/servodyn/ExtendedBladedInterface.html, 2023. Accessed: 2022-09-22.

[62] P. Fleming, P. Gebraad, M. Churchfield, S. Lee, K. Johnson, J. Michalakes, J. W. van Wingerden, and P. Moriarty, "SOWFA + super controller user's manual," *National Renewable Energy Laboratory (NREL)*, 2013.

[63] K. Shaler, J. Jonkman, P. Doubrawa Moreira, and N. Hamilton, "Fast. farm response to varying wind inflow techniques," tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2019.

[64] J. Jonkman, P. Doubrawa, N. Hamilton, J. Annoni, and P. Fleming, "Validation of FAST.Farm against large-eddy simulations," *Journal of Physics: Conference Series*, vol. 1037, p. 062005, jun 2018.

[65] P. Doubrawa, J. R. Annoni, and J. M. Jonkman, "Optimization-based calibration of fast. farm parameters against large-eddy simulations," in *2018 Wind Energy Symposium*, p. 0512, 2018.

[66] M. J. van den Broek and J.-W. van Wingerden, "Dynamic flow modelling for model-predictive wind farm control," *Journal of Physics: Conference Series*, vol. 1618, p. 022023, sep 2020.

[67] P. M. O. Gebraad and J. W. van Wingerden, "A control-oriented dynamic model for wakes in wind plants," *Journal of Physics: Conference Series*, vol. 524, p. 012186, jun 2014.

[68] The Mathworks Inc., "MATLAB." https://nl.mathworks.com/, 2022. Accessed: 2022-05-01.

[69] F. Meng, W. H. Lio, and T. Barlas, "Dtuwec: an open-source dtu wind energy controller with advanced industrial features," *Journal of Physics: Conference Series*, vol. 1618, no. 2, p. 022009, 2020.

[70] NREL, "ROSCO. Version 2.4.1." https://github.com/NREL/rosco, 2021. Accessed: 2023-02-27.

[71] K. Kim, H. Kim, C. Kim, I. Paek, C. Bottasso, and F. Campagnolo, "Design and validation of demanded power point tracking control algorithm of wind turbine," *Int. J. of Precision Engineering and Manufacturing-Green Technology*, vol. 5, pp. 387–400, 07 2018.

[72] P. Fleming, J. Aho, P. Gebraad, L. Pao, and Y. Zhang, "Computational fluid dynamics simulation study of active power control in wind plants," *American Control Conference (ACC)*, pp. 1413–1420, 2016.

[73] C.-J. Smits, "GitHub Coen-Jan with files from thesis work." https://github.com/coenjan/Files-MSC-Thesis-.git, 2023. Accessed: 2023-05-26.

[74] L. Prendergast, K. Gavin, and P. Doherty, "An investigation into the effect of scour on the natural frequency of an offshore wind turbine," *Ocean Engineering*, vol. 101, 06 2015.

[75] K. Shaler and J. Jonkman, "Fast. farm development and validation of structural load prediction against large eddy simulations," *Wind Energy*, vol. 24, no. 5, pp. 428–449, 2021.

[76] I. Eguinoa, T. Göçmen, P. B. Garcia-Rosa, K. Das, V. Petrović, K. Kölle, A. Manjock, M. J. Koivisto, and M. Smailes, "Wind farm flow control oriented to electricity markets and grid integration: Initial perspective analysis," *Advanced Control for Applications*, vol. 3, no. 3, pp. 1–28, 2021.

[77] C. R. Shapiro, P. Bauweraerts, J. Meyers, C. Meneveau, and D. F. Gayme, "Model-based receding horizon control of wind farms for secondary frequency regulation," *Wind Energy*, vol. 20, no. 7, pp. 1261–1275, 2017.

[78] S. Boersma, B. Doekemeijer, S. Siniscalchi-Minna, and J. W. van Wingerden, "A constrained wind farm controller providing secondary frequency regulation: An les study," *Renewable Energy*, vol. 134, pp. 639 – 652, 2019.

[79] J. G. Silva, R. Ferrari, and J.-W. van Wingerden, "Convex model predictive control for down-regulation strategies in wind turbines," *IEEE 61st Conference on Decision and Control (CDC)*, pp. 3110–3115, 2022.

Glossary

List of Acronyms

2D 2-Dimension

3D 3-Dimension

AFC Active Flow Control

AGC Automatic Generation Control

AIC Axial Induction Control

ALM Actuator Line Model

APC Active Power Controller

AWAE Ambient Wind & Array Effects

CFD Computational Fluid Dynamics

DEL Damage Equivalent Load

DLL Dynamic Link Library

DNS Direct Numerical Simulation

DWM Dynamic Wake Meandering

FAST.Farm Fatigue, Aerodynamics, Structures, and Turbulence Farm tool

FLORIDyn FLow Redirection and Induction Dynamics

FLORIS FLow Redirection and Induction in Steady State

LES Large-Eddy Simulation

MEX MATLAB Executable

MPI Message Passing Interface

MW Megawatts

NREL National Renewable Energy Laboratory

NS Navier-Stokes

OpenFAST Open Fatigue, Aerodynamics, Structures, and Turbulence tool

S-Functions System-Functions

SOWFA	Simulator fOr Wind Farm Applications
SC	Super Controller
TSR	Tip Speed Ratio
WM	Wake Mixing
WRC	Wake Redirection Control

List of Symbols

Abbreviations

γ	Yaw Angle
λ	Tip Speed Ratio
ω	Angular Velocity
ρ	Air Density
τ_{g}	Generator Torque
θ	Blade Pitch Angle
C_{P}	Power Coefficient
C_{T}	Thrust force Coefficient
c_{max}	Maximum Turbine Chord Length
D^{Rotor}	Diameter of rotor
P_{w}	Maximum Kinetic Power in Flow
U_{r}	Wind Velocity at Rotor
U_{∞}	Free-stream Wind Velocity
V_{Hub}	Hub-Height Wind Speed
x_{dist}	Distance Until Wake Deficit Decays Away
A	Rotor Surface
a	Axial Induction Factor
dr	Radial Increment
P	Power
r	Length of a Blade
U	Flow Velocity
V	Average Convection Speed of the Wake