A FAST.Farm and MATLAB/Simulink interface for wind farm control design

Coen-Jan Smits[1], **Jean Gonzalez Silva**[1], **Valentin Chabaud**[2] and **Riccardo Ferrari**[1]

[1] Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
[2] SINTEF Energy Research, 7034 Trondheim, Norway

E-mail: `c.j.c.smits@student.tudelft.nl`, `J.GonzalezSilva@tudelft.nl`, `valentin.chabaud@sintef.no`

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 controls. FAST.Farm is a recently developed multi-physics engineering tool for modeling wind farm performance by solving the aero-hydro-servoelastic dynamics of each individual turbine. The capabilities of FAST.Farm for control design purposes can be extended through a co-simulation with MATLAB/Simulink. Therefore, a MATLAB/Simulink interface with FAST.Farm has been developed. The creation and operation of this interface are explained in this paper. This interface supports developing and testing advanced closed-loop control at the wind turbine and wind farm levels.

1. Introduction

The development of wind farm controllers progresses, 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 farm performance [1, 2]. 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 [3, 4]. 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. FLORIS [5], 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 SOWFA [6] estimate the flow field in detail by resolving the governing equations (Navier-Stokes) in three-dimensional (3-D) space. However, high-fidelity tools are impractical in use because of the high computation time. FAST.Farm [7] 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 [8, 9]. Therefore, FAST.Farm shows to be a promising tool for designing wind farm controllers and simulating wind farm behavior [10].

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 [11]. An SC-DLL is used in conjunction with individual wind turbine controllers designed in the style of the DISCON DLL [12]. The SC-DLL and DISCON DLL communicate by exchanging the avrSWAP matrix [13]. This matrix contains simulation results from the OpenFAST instances and control variables from the SC-DLL. 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 academic control community 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. Communication between FAST.Farm and MATLAB/Simulink could be realized by exchanging the avrSWAP matrix.

The contribution of this paper 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 exchanging the avrSWAP matrix between FAST.Farm and MATLAB/Simulink by linking the individual turbine controllers in the FAST.Farm tool to MATLAB through a 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.

The structure of this paper is as follows. First, the setup of the FAST.Farm and MATLAB/Simulink interface is detailed in section 2. Next, section 3 describes the outcome of a case study that shows the operation of the interface. Finally, section 4 concludes the paper.

2. MATLAB/Simulink and FAST.Farm interface

MATLAB and Simulink are both developed by Mathworks, which is specialized in developing mathematical computing software [14]. MATLAB is primarily used for analyzing and visualizing data, as well as developing algorithms, while Simulink is a graphical simulation environment used for designing dynamical systems. Simulink can be used as an add-on to MATLAB. The MATLAB/Simulink and FAST.Farm interface is realized by linking the individual turbine controllers in the FAST.Farm tool to MATLAB through an MPI and MEX functions. Figure 1 depicts the calling sequence of the interface of a wind farm in FAST.Farm with three wind turbines. On the side of FAST.Farm, an MPI is connected to the individual turbine controllers of 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. 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 calls the MPI interface through the use of MEX functions, and therefore messages are exchanged between the turbine controllers and MATLAB. The message that is sent back and forth between FAST.Farm and MATLAB is stacked in the so-called avrSWAP matrix [13]. This matrix contains simulation results and control variables of each wind turbine in vectors. The avrSWAP matrix is a generally known matrix by FAST.Farm and OpenFAST. 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 remaining of this section is as follows. Section 2.1 explains the creation of the MPI interface. The setup of the MEX functions in MATLAB is outlined in section 2.2. Last, section 2.3 explains the expansion to Simulink.

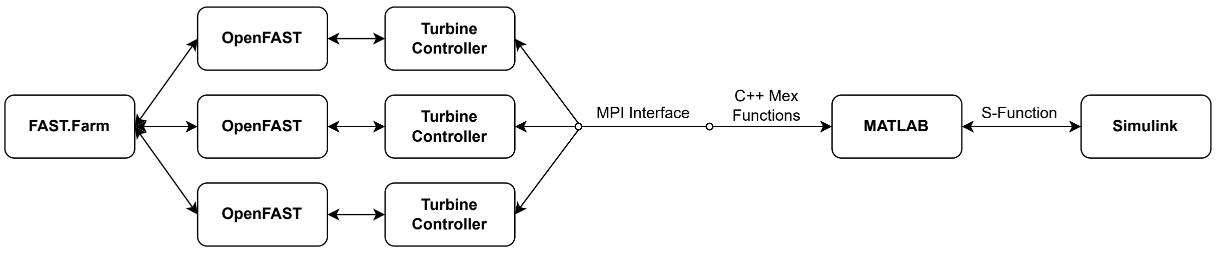


Figure 1: Calling sequence between FAST.Farm and MATLAB/Simulink.

2.1. MPI interface in FAST.Farm

FAST.Farm can exchange messages with other programs through an MPI interface. Herein, the so-called message is the avrSWAP matrix. Initially, the DLLs of the turbine controllers in each OpenFAST instance receive controller commands from the SC in FAST.Farm. The SC-DLL of FAST.Farm is essentially identical to the super controller available in SOWFA [11]. The SC-DLL is used in conjunction with individual wind turbine controllers designed in the style of the DISCON DLL (originally used in Bladed) [12]. Instead of using an SC-DLL, the DLLs of the internal turbine controller are modified in order to link them to an MPI. 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 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 [15] or ROSCO [16], or offers the possibility to directly link MATLAB/Simulink to OpenFAST. 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, which is the case in this paper, the turbine controller has to be defined inside MATLAB/Simulink. This provides the opportunity to design the turbine controller yourself.

2.2. MATLAB Interface with MEX Functions
A MATLAB script has been developed with four MEX functions to establish a connection between MATLAB and the MPI framework. Hence, information can be exchanged between MATLAB and the OpenFAST instances. The four MEX functions are named *Initialise*, *Receive*, *Send*, and *Stop*. Figure 2 depicts the calling sequence of the MEX functions in the MATLAB/Simulink environment.

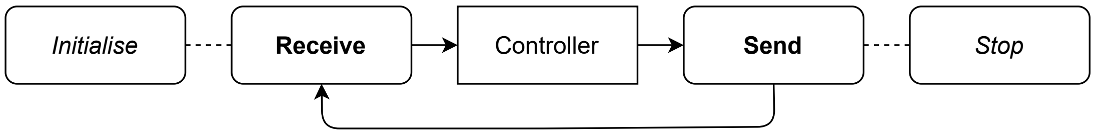


Figure 2: Calling sequence of MEX functions in MATLAB/Simulink.

- 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 in order to exchange messages between the MPI 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 for one (random) turbine. This function ensures that FAST.Farm has performed a new simulation time-step before it retrieves the matrix.
- 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 same 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 the MATLAB script 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.

2.3. Simulink interface
Simulink offers the possibility to visualize controllers in a graphical block diagram. This makes Simulink a more intuitive tool for designing 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.

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
```

receiving, processing, and sending data. 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, are created in Simulink to facilitate the exchange of the avrSWAP matrix between MATLAB and Simulink. The Receive and Send block contain the same eponymous MEX functions. See Figure 3 for the visualization in Simulink.

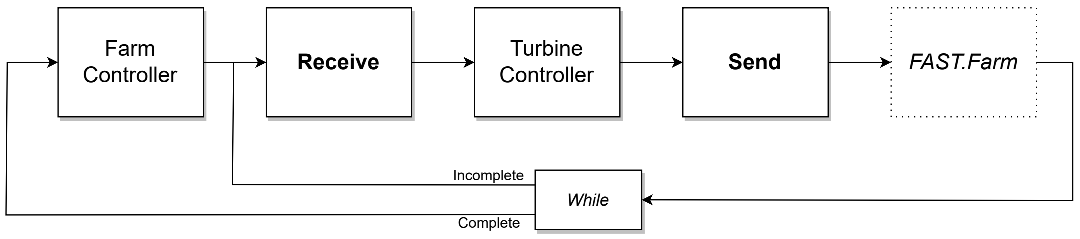


Figure 3: Block diagram in Simulink of FAST.Farm and Simulink interface.

3. Case Study

Three test cases have been set up to show the operation of the MATLAB/Simulink and FAST.Farm interface. These test cases verify the implementation of the MATLAB/Simulink interface in FAST.Farm. Figure 4 illustrates the setup of the wind farm used in this case study. In section 3.1, three wind turbines are simulated in FAST.Farm with and without the MATLAB interface to investigate the communication time between MATLAB and FAST.Farm. In section 3.2, an Active Power Controller (APC) and a yaw controller have been implemented in MATLAB and simulated in FAST.Farm through the MATLAB/Simulink interface in order to validate the functionality of the interface. The APC is based on power tracking controllers [17, 18]. The APC responds to grid requirements through the control of wind farm power output. The yaw controller purposely misaligns the rotor of the first (most upstream) turbine with respect to the incoming flow in order to deflect downstream wakes from downstream turbines.

3.1. Communication Time Experiment

Three 10 MW wind turbines are simulated in FAST.Farm with and without the interface to MATLAB to compare the simulation time, see Table 1. 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 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. Such a simulation of three aligned wind turbines in the low-fidelity tool FLORIS would be in the order of seconds on a desktop PC, and in the high-fidelity tool SOWFA in the order of days on a cluster of 10^2 CPU's [19].

Table 1: Simulation 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

3.2. Controlled Wind Farm Simulations
A wind farm consisting of three wind turbines is simulated considering the scenario of high wake interaction, see Figure 4. Two controllers are simulated to evaluate the functionality of the interface: an APC and a yaw controller. With the APC, the wind farm responds to grid requirements through control of farm power output. With the yaw controller, the wakes of the upstream turbine are deflected from the downstream turbines. Figure 5 visualizes the turbine-level and farm-level controller interface between MATLAB and FAST.Farm. The APC and yaw controller are implemented in the FAST.Farm and MATLAB interface according to this structure. First, section 3.2.1 provides a brief explanation of the APC, accompanied by the presentation of its simulation results. Moving on to section 3.2.2, the yaw controller is briefly explained, and its simulation results are presented. The simulation results of both controllers are discussed in Section 3.2.3.

3.2.1. Power Tracking Controller In this section, we briefly present the implementation of the wind turbine and wind farm controllers for power tracking purposes designed by the authors in

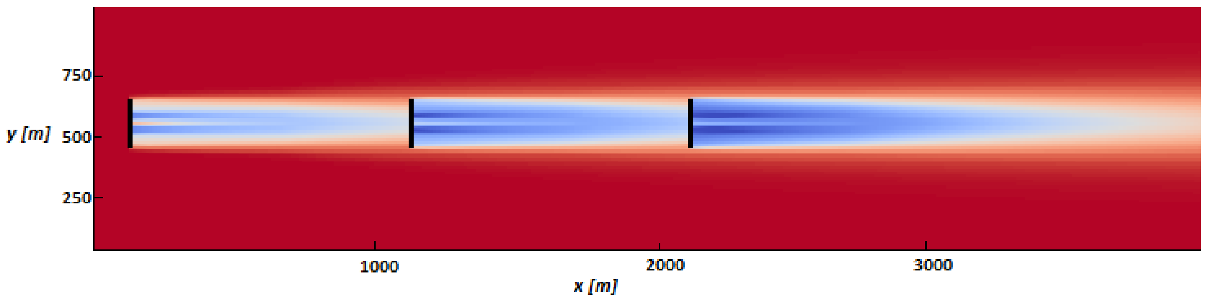


Figure 4: An instantaneous horizontal slice of flow output taken from FAST.Farm.

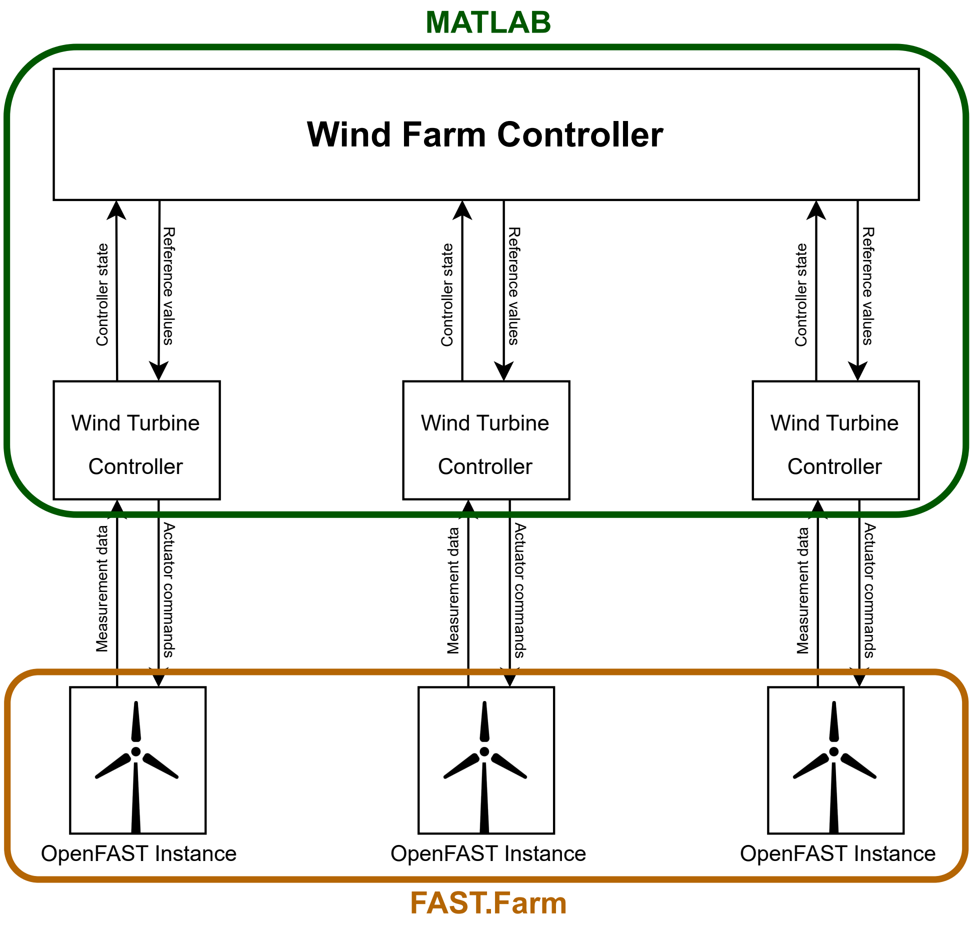


Figure 5: Block diagram visualizing the controller interface between MATLAB and FAST.Farm.

Silva et al. [20] [4]. At the turbine level, the controller receives the current blade pitch angle generator torque, and rotor speed as input from FAST.Farm. Subsequently, the controller determines the updated blade pitch angle and generator torque in order to match the power output with the reference signal. Two operational modes can be distinguished in the power tracking controller using the input signals from FAST.Farm. The controller either forces the 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.

At farm level, the wind farm power tracking controllers can be used, for example, for thrust balancing, wake-loss compensation, load-limiting, and turbulent wind fluctuation compensation [4, 21]. The wind farm controller tested in this paper uniformly distributes the power references over the number of turbines in the farm to follow a power command signal provided by the system operator. The time-varying Automatic Generation Control (AGC) signal is used as the power reference signal [17]. The AGC signal maintains constant for the first 300 seconds to allow time for the wakes to develop and propagate during the simulation. After 300 seconds the AGC signal varies over time. Both, the wind farm and wind turbine controllers are designed within MATLAB. See Algorithm 1 for the implementation of this structure in MATLAB. Figure 6 shows the simulation results in FAST.Farm.

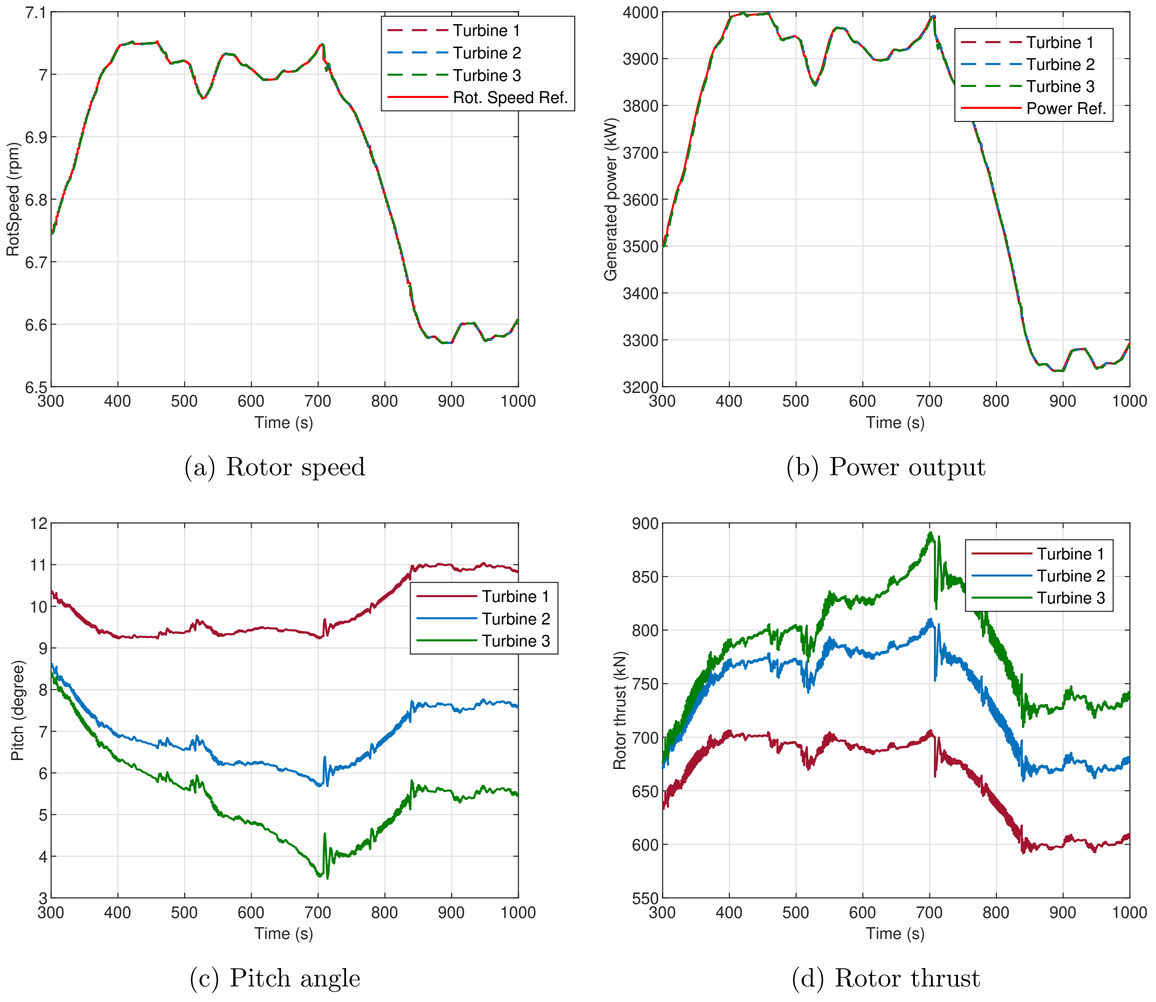


Figure 6: Simulation results of APC simulated in FAST.Farm.

3.2.2. Yaw Controller Yaw control focuses on purposely misaligning the rotor of upstream turbines with respect to the incoming flow. By misaligning the rotor, the downstream wake is deflected from downstream turbines, see Figure 7. The operation of the FAST.Farm and MATLAB/Simulink interface is validated by obtaining the expected simulation results of the yaw controller in FAST.Farm. The yaw controller is implemented within the turbine-level loop and the yaw angle references are determined by the farm-level controller. The yaw angle is controlled indirectly by controlling the yaw rate in the avrSWAP matrix. At the turbine level, the current yaw angle is an output of FAST.Farm. Subsequently, the turbine-level yaw controller in MATLAB compares this measured yaw angle with the reference yaw angle. Based on the deviation between those two angles, the yaw rate is commanded to steer the nacelle and minimize the deviation. In this case study, greedy control is used and the first upstream turbine is yawed 30 degrees from MATLAB with respect to its initial condition. The other turbines are not yawed with respect to the direction of the incoming flow. Figure 8 shows the simulation results in FAST.Farm. The first 300 seconds of the simulation are omitted, as some time is needed to start up the turbines and develop and propagate the wakes.

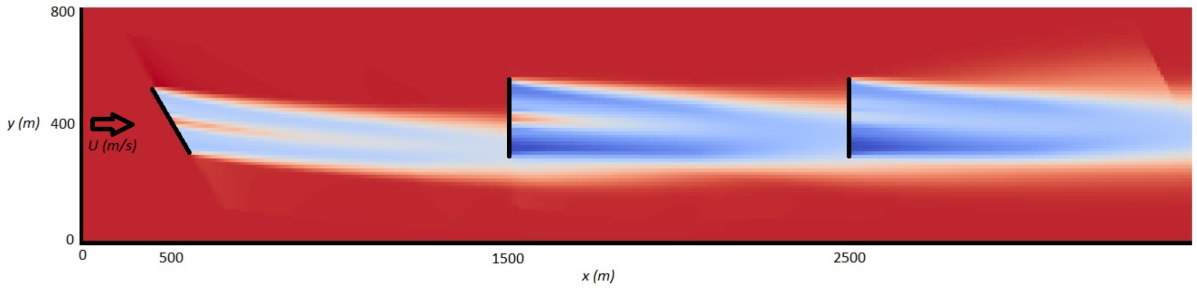
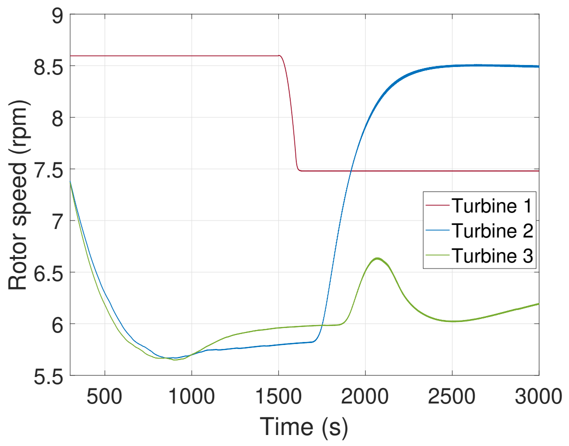
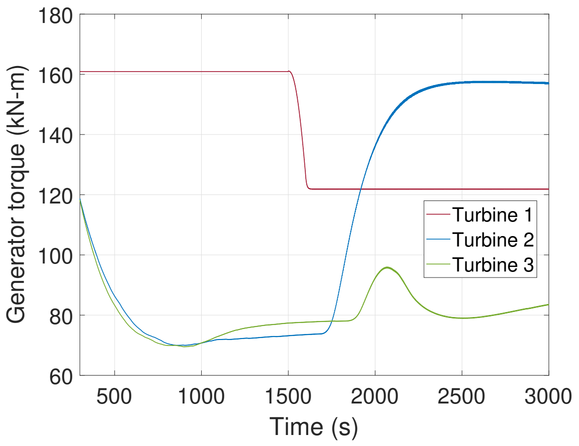


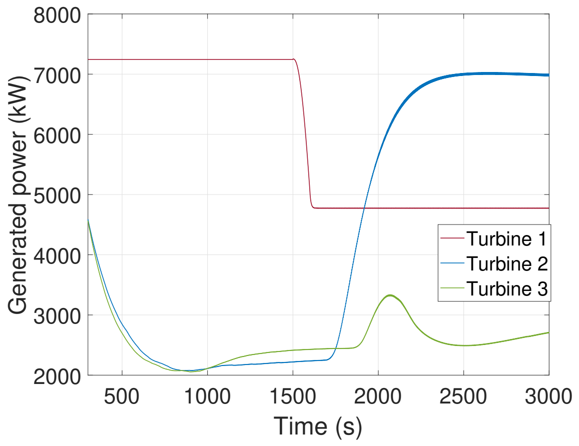
Figure 7: An instantaneous horizontal slice of flow output taken from FAST.Farm. The first (most left) turbine is yawed at a 30-degree angle with respect to the incoming flow field.



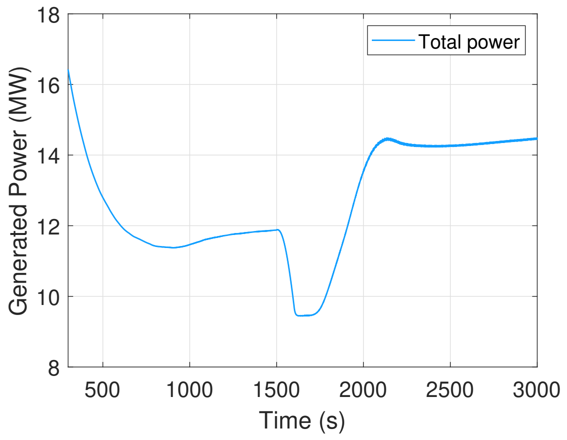
(a) Rotor speed



(b) Generator torque



(c) Generated power



(d) Total power output

Figure 8: Simulation results of yaw controller simulated in FAST.Farm.

3.2.3. Discussion The results illustrated in Figure 6 and 8 demonstrate the interaction between the wind turbine & farm controller in MATLAB and FAST.Farm. The APC in MATLAB determines the optimal control variables based on both the AGC signal and the synchronized simulation output from FAST.Farm. Consequently, these control commands are transmitted to FAST.Farm by exchanging the avrSWAP matrix. The results in Figure 6 demonstrate the expected power tracking results. The yaw controller considers the reference yaw angles and simulation output from FAST.Farm for determining the control output. Figure 8 shows the output of FAST.Farm given a desired yaw reference. Figure 8 shows that the first upstream turbine deflects the wakes from downstream turbines. These simulations confirm the exchange of the avrSWAP matrix between FAST.Farm and MATLAB. As expected, the total farm power increases reaching a steady state after steering the wake of the upstream turbine. Therefore, these results validate the operation of the FAST.Farm and MATLAB/Simulink interface. For these simulations, an inflow wind speed of 10 m/s is used.

4. Conclusion

In this paper the first, to the best of the author's knowledge, step in creating a full-purpose MATLAB/Simulink interface in FAST.Farm was presented, see GitHub [22]. The capabilities of FAST.Farm for control design purposes have been extended through a co-simulation with MATLAB/Simulink. The case study in this paper shows that MATLAB/Simulink can be used as an add-on to FAST.Farm. Control variables and simulation results can be transmitted between FAST.Farm and MATLAB/Simulink by exchanging the avrSWAP matrix. The FAST.Farm simulation results of the APC and the yaw controller validate the operation of the interface. Controllers can be created at both wind farm and turbine levels. This interface supports the development and testing of advanced closed-loop control at the wind farm level. In addition, the interface barely affects the simulation time. Future work will focus on further implementing closed-loop controllers at the wind farm level, as in [4, 23]. The interface could be linked to the Simulink wind farm control platform (Simulink-wfc-platform) from the FarmConners project [24] to extend its capabilities. Moreover, using wind farm models, such as FLORIS in the Simulink-wfc-platform and predictive controllers [25–27] could be explored.

Code availability

The MATLAB/Simulink interface used in this paper can be found at GitHub [22].

Acknowledgments

The authors would like to acknowledge the WATEREYE project (grant no. 851207). This project has received funding from the European Union Horizon 2020 research and innovation program under the call H2020-LC-SC3-2019-RES-TwoStages.

References

[1] Pao L Y and Johnson K E 2009 *American Control Conference* 2076–2089
[2] Kheirabadi A C and Nagamune R 2019 *Journal of Wind Engineering and Industrial Aerodynamics* **192** 45–73
[3] Doekemeijer B M, van der Hoek D and van Wingerden J W 2020 *Renewable Energy* **156** 719–730
[4] Silva J G, Ferrari R and van Wingerden J W 2023 *Renewable Energy* **203** 421–433
[5] NREL 2022 FLORIS. version 3.2 Accessed: 2022-11-06 URL `https://www.nrel.gov/wind/floris.html`
[6] NREL 2022 SOWFA Accessed: 2022-11-20 URL `https://www.nrel.gov/wind/nwtc/sowfa.html`
[7] NREL 2023 FAST.Farm GitHub `https://github.com/OpenFAST/openfast` Accessed: 2023-02-11
[8] Jonkman J M, Annoni J, Hayman G, Jonkman B and Purkayastha A 2017 *35th Wind Energy Symposium* 0454
[9] NREL 2022 FAST.Farm documentation Accessed: 2022-10-04 URL `https://openfast.readthedocs.io/en/dev/source/user/fast.farm`
[10] Kretschmer M, Jonkman J, Pettas V and Cheng P W 2021 *Wind Energy Science* **6** 1247–1262

[11] Fleming P, Gebraad P, van Wingerden J W, Lee S, Churchfield M, Scholbrock A, Michalakes J, Johnson K and Moriarty P 2013 *Wind Energy Conference and Exhibition* **3** 1561–70
[12] DNV-GL 2022 Bladed Accessed: 2022-11-25 URL `https://www.dnvgl.com/energy`
[13] NREL 2023 Extended bladed interface Accessed: 2023-01-13 URL `https://openfast.readthedocs.io/en/main/source/user/servodyn/ExtendedBladedInterface.html`
[14] Inc T M 2022 MATLAB Accessed: 2022-09-01 URL `https://nl.mathworks.com/`
[15] Meng F, Lio W H and Barlas T 2020 *Journal of Physics: Conference Series* **1618** 022009
[16] NREL 2021 ROSCO. Version 2.4.1 Accessed: 2023-02-27 URL `https://github.com/NREL/rosco`
[17] Fleming P, Aho J, Gebraad P, Pao L and Zhang Y 2016 *American Control Conference (ACC)* 1413–1420
[18] van Wingerden J W, Pao L, Aho J and Fleming P 2017 *IFAC-PapersOnLine* **50** 4484 – 4491 20th IFAC World Congress
[19] Boersma S, Doekemeijer B M, Gebraad P M O, Fleming P A, Annoni J, Scholbrock A K, Frederik J A and van Wingerden J W 2017 *American Control Conference (ACC)* 1–18
[20] Silva J G, Doekemeijer B M, Ferrari R and van Wingerden J W 2022 *Journal of Physics: Conference Series* **2265** 022056
[21] Merz K, Chabaud V, Garcia-Rosa P B and Kolle K 2021 *Journal of Physics: Conference Series* **2018** 012026
[22] Smits C J, Chabaud V, Silva J G and Ferrari R 2023 FAST.Farm and MATLAB/Simulink interface Accessed: 2023-03-08 URL `https://github.com/ValentinChb/FASTFarm2Simulink`
[23] Doekemeijer B M, van der Hoek D and van Wingerden J W 2020 *Renewable Energy* **156** 719 – 730
[24] Eguinoa I, Göçmen T, Garcia-Rosa P B, Das K, Petrović V, Kölle K, Manjock A, Koivisto M J and Smailes M 2021 *Advanced Control for Applications* **3** 1–28
[25] Shapiro C R, Bauweraerts P, Meyers J, Meneveau C and Gayme D F 2017 *Wind Energy* **20** 1261–1275
[26] Boersma S, Doekemeijer B, Siniscalchi-Minna S and van Wingerden J W 2019 *Renewable Energy* **134** 639 – 652
[27] Silva J G, Ferrari R and van Wingerden J W 2022 *IEEE 61st Conference on Decision and Control (CDC)* 3110–3115