# Wind turbine control

## Open-source software for control education, standardization and compilation

Mulders, S. P.; Zaaijer, M. B.; Bos, R.; Van Wingerden, J. W.

**DOI**
10.1088/1742-6596/1452/1/012010

**Publication date**
2020

**Document Version**
Final published version

**Published in**
Journal of Physics: Conference Series

# Wind turbine control: open-source software for control education, standardization and compilation

# Wind turbine control: open-source software for control education, standardization and compilation

**S P Mulders[1], M B Zaaijer[2], R Bos[3] and J W van Wingerden[1]**

[1] Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
[2] Aerodynamics, Wind Energy, Flight Performance & Propulsion, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands
[3] Eneco, Marten Meesweg 5, 3068 AV Rotterdam, The Netherlands

E-mail: S.P.Mulders@tudelft.nl, M.B.Zaayer@tudelft.nl, J.W.vanWingerden@tudelft.nl

**Abstract.** Standardized, easy to use, and preferably open-source research software is an important aspect in supporting and solidifying the wind turbine community. To this end, three contributions in the form of open-source software projects are presented in this paper. First, a community-driven wind turbine baseline controller, the Delft Research Controller (DRC), is presented. The DRC is applicable to high-fidelity simulation software that uses the DISCON controller interface. The controller distinguishes itself by the variety of available control and estimation implementations, its ease of use, and the universal applicability to wind turbine models. Secondly, in the wake of the DRC, the SimulinkDRC graphical controller design and compilation environment has been developed. Users having access to Simulink can benefit from the convenient way of controller development the tool provides. Finally, the FASTTool has been developed for educational purposes, by focusing on the graphical aspect of wind turbine (controller) design. The tool simplifies interaction with the advanced FAST simulation software, by comprehensive visualizations and analysis tools. This paper demonstrates and describes the functionality of all three software projects.

## 1. Introduction

Wind turbine control is a non-trivial task, and generally requires expert control knowledge and software skills for design, tuning and implementation. For this reason, wind energy research groups from the Delft University of Technology (TU Delft) develop and actively maintain various open-source, free, and publicly available wind turbine control oriented software projects. The following three open-source projects are outlined in this paper:

*Delft Research Controller (DRC).* The DRC is an open-source and community-driven wind turbine baseline controller. The development of the controller is driven by the notion of wind energy research groups from various disciplines often using self-developed baseline implementations and tunings, complicating the evaluation and comparison of new control algorithms. To solve this problem, the DRC provides an open, modular and fully adaptable baseline wind turbine controller to the scientific community. New control implementations are easily added to the existing baseline controller, and in this way, convenient assessments of the proposed algorithms is possible. Because of the open character and modular set-up, scientists

are able to collaborate and contribute in making continuous improvements to the code. The DRC is being developed in Fortran and uses the Bladed-style DISCON controller interface. The compiled controller is configured by a single control settings parameter file, and can work with any wind turbine model and simulation software using the DISCON interface. Baseline parameter files are supplied for the NREL 5-MW and DTU 10-MW reference wind turbines.

*SimulinkDRC.* In the wake of the DRC, a graphical controller design and compilation environment has been developed in Simulink, and is called SimulinkDRC. For engineers having access to Simulink, this tool provides an easy and convenient way of controller development.

*FASTTool.* The FASTTool has been developed for educational purposes in wind turbine design. FASTTool is a graphical user interface (GUI) for NREL's aeroelastic simulation code FAST. The tool is centered around a three-dimensional animated wind turbine plot, which dynamically adapts to the defined design inputs. FASTTool provides users with convenient and insightful tools to tune controllers and assess the performance of the design.

All software is released under the MIT license – a free software license – at the following location:

`https://github.com/TUDelft-DataDrivenControl`

The organization of this paper is as follows. Section 2 describes the philosophy, functionality and the working principles of the DRC baseline wind turbine controller. In Section 3, SimulinkDRC is presented, opening possibilities for graphical controller design in Simulink. Finally, Section 4 presents FASTTool: An educational and graphical interface for NREL's high-fidelity wind turbine simulation software FAST.

## 2. DRC: an open-source and community-driven baseline controller

The existence of reference models and baseline load cases is a crucial aspect in the scientific community, as it allows for convenient and fair evaluation of proposed innovations. In the wind turbine community, the National Renewable Energy Laboratory (NREL) 5-MW baseline wind turbine is a fictive, but fully defined reference wind turbine (RWT) model [1], and is actively used in the scientific field. To accommodate the next step in enlarging the size and rated power of offshore wind turbines, the Technical University of Denmark (DTU) provides a 10-MW RWT model [2].

Besides reference models, wind turbine simulation software is also largely standardized. The industry standard, commercial and certified high-fidelity wind turbine simulation package is Bladed by DNV GL [3]. On the other hand, an open-source aeroelastic package for simulating horizontal-axis wind turbines is (Open)FAST, which is actively developed and maintained by NREL.

In contrast to the previously mentioned reference models and simulation software, no clear choice of a baseline wind turbine controller currently exists that is easy to use, modular and extendable. Wind energy research groups from various disciplines generally use self-developed baseline control implementations and tunings, of which the source code is rarely available. This negatively impacts the ability to compare results from different research projects or groups. It has to be noted that NREL provides an open-source controller for its NREL 5-MW reference wind turbine [1]. However, this controller is limited in functionality and inconvenient to extend or interchange between distinct wind turbine models as functionality, turbine parameters and controller tunings are hard-coded in a single source file. DTU also provides an internally developed controller for their DTU 10-MW RWT [4] of which the source code is available, but the development is not driven by a broad community.

To this end, the Data Driven Control wind energy research group from Delft University of Technology started the initiative to develop an open-source and community-driven wind turbine baseline controller. A design specification was that the controller should be generally applicable to all turbine models defined in simulation software that uses the Bladed-style DISCON controller interface [5], such as OpenFAST, Bladed or HAWC2. Also, a convenient way of configuring the controller should be present, without editing the source code and thus the need for recompilation. With these goals in mind, the foundations of a baseline wind turbine controller have recently been laid out, and is dubbed the Delft Research Controller (DRC) [6]. For consistency with OpenFAST, the DRC is being developed in the Fortran programming language (free-form) [7]. The modular and open character allows scientists to collaborate and contribute in making continuous improvements to the code. The DRC is provided with a toolbox consisting of regularly used (control) functions and filters to allow for rapid development and implementation of new contributions.

The main contribution of this section is to provide a comprehensive description of the working principles and functionality of the DRC, and is organized as follows. In Section 2.1, an overview of the DRC is given, after which in Section 2.2 the built-in filter and function modules are described. A wind speed estimator, described in Section 2.3, is included to provide below-rated, closed-loop, tip-speed ratio tracking capabilities. The components in the before mentioned modules are used to make up the baseline torque, (individual) pitch and yaw controllers, described in Section 2.4. Section 2.5 describes fatigue load reduction control strategies, and Section 2.6 outlines the incorporated yaw control implementations.

### *2.1. Overview and description of the DRC*

This section gives a general overview and description of the DRC controller architecture and philosophy. The overall DRC working principle is presented in Figure 1. The DRC is set up such that only a single parameter file `DISCON.IN` for each wind turbine model is required to define the complete control system. This feature removes the need for repetitive recompilation of the controller under a change in control settings. Baseline controller parameter configuration files

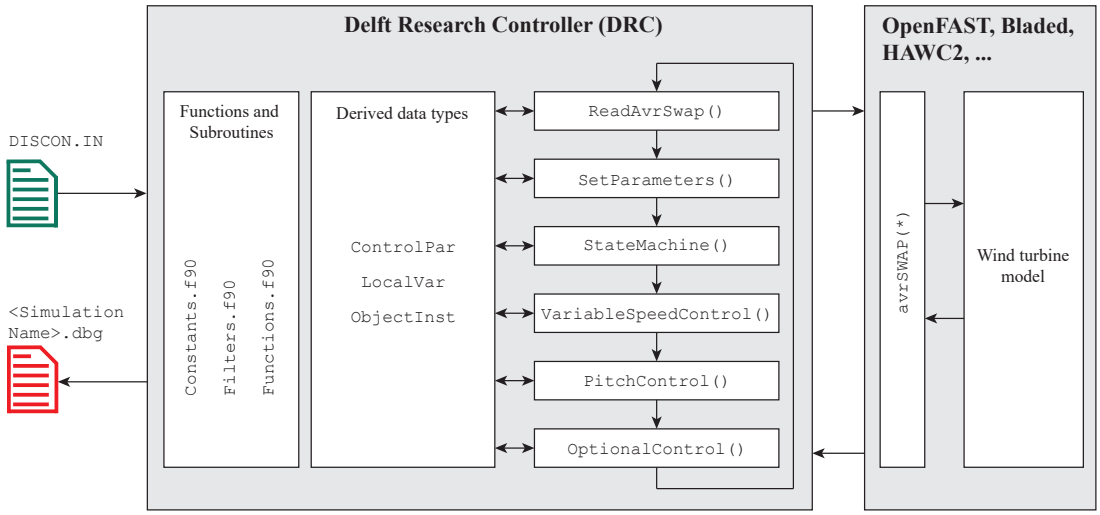



Figure 1: A schematic of the DRC architecture for wind turbine control. The controller exchanges data using the so-called the DISCON external controller interface via the `avrSWAP`-array. The DRC is completely parameterized by a single configuration file, and writes debug information to a log file when desired.

are supplied for the NREL 5-MW and DTU 10-MW reference wind turbines, and the source code is publicly available under terms of the MIT License [8].

The DRC consists of multiple modules containing commonly used functions and subroutines, and uses derived data types to store parameters and variables in a centralized manner. Function calls are executed in a fixed sequence during each control iteration. The DRC reads the control in- and output `avrSWAP`-array [5] and performs value assertions. Next, before calling any controller, the state-machine determines the state of the turbine, and this information is used by the controllers to perform corresponding control actions. To enable the Variable-Speed Variable-Pitch (VSVP) control strategy, torque and pitch control subroutines are implemented. Optional controllers are executed after the two before-mentioned controllers.

### *2.2. Filters and functions modules*

An overview of the included filters and functions are described in this section. The DRC comes with a collection of frequently used filters. The filters included are discretized using the bilinear transformation, also known as Tustin's method [9]. The filters are not bound to a predefined sampling time, as this variable is taken as an input from the simulation. All functions return a single real filtered output signal, with a default unity steady-state gain. The following filters are included:

- *First/second-order low-pass filter.* Pass signals with frequencies lower than the cross-over frequency, but attenuate signal components above this frequency.
- *First-order high-pass filter.* Passes signals with frequencies higher than the cut-in frequency, but attenuates signal components below this frequency.
- *Notch filter.* Passes most of the frequencies but attenuates in a very specific interval.
- *Inverted-notch filter with decreasing slopes.* Amplifies a very specific frequency region, and provides extra attenuation of frequencies outside this domain.

The DRC controller is supplied with the following frequently used functions:

- *Value/signal saturation.* Saturates a given input signal to a upper and lower value.
- *Signal rate limiter.* A signal rate limiter with respect to time.
- *Proportional-Integral (PI) controller.* An object-based PI-controller with anti-integrator wind-up and signal saturation capabilities.
- *1D-interpolation.* A one-dimensional interpolation function taking a one-dimensional table. The x-data should be monotonically increasing.

### *2.3. Wind speed estimation*

The wind speed measurement from the anemometer is influenced by induction, as it is often located downwind of the rotor at the back of the nacelle. Therefore, the measurements are often considered unreliable for use in control implementations [10], and only serve indication purposes. Moreover, the sensor only measures the wind speed at the center of the rotor swept area, while the wind speed varies spatially over the rotor surface [11]. Although not perfect, to keep the rotor operating at maximum power coefficient tracking, torque control is often implemented as the optimal-mode gain, multiplied by the rotor or generator speed squared [12]. However, this control scheme assumes perfect model representation and knowledge about the rotor aerodynamic behavior. In real world scenarios, the model and actual rotor show increasing inconsistencies over time, as a result of wear, tear, fouling, icing and manufacturing imperfections [13]. For this reason, often a wind speed estimator is employed to provide closed-loop tip-speed ratio tracking capabilities, eliminating the need for perfect a priori knowledge of aerodynamic characteristics.

Different types of rotor effective wind speed estimators have been proposed [14], like the power balance estimator [15], and the (extended) Kalman filter [10, 16]. More advanced

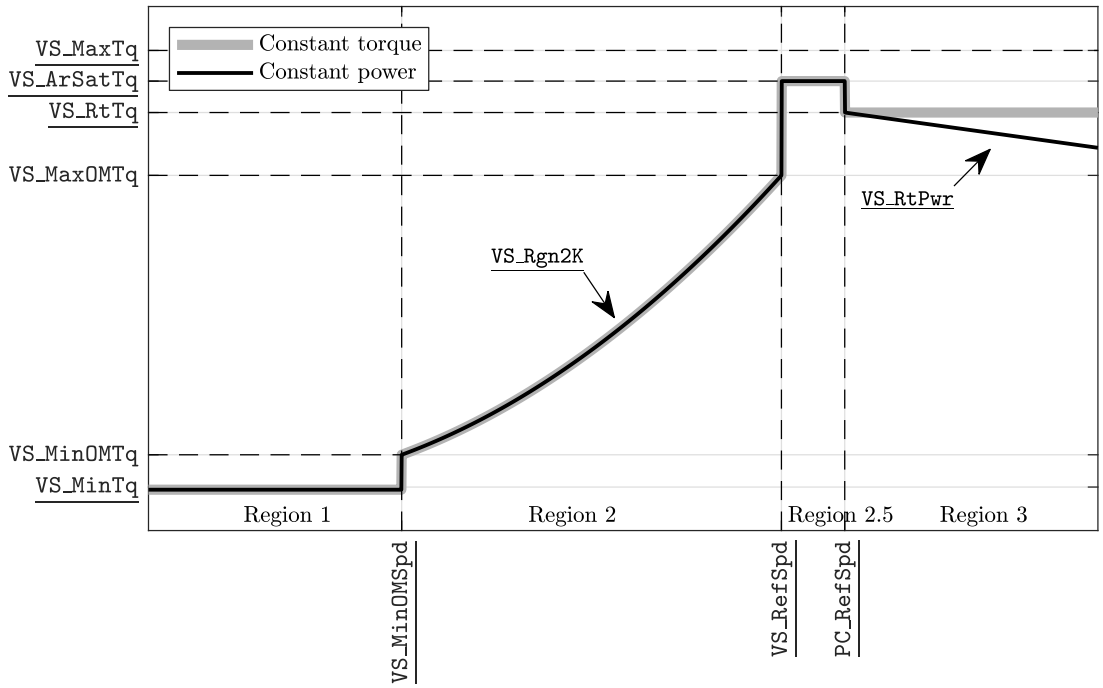



Figure 2: Torque control strategies implemented in the DRC. All variables regarding torque control are indicated by their respective names present in the control parameter file.

implementations are proposed for estimation of horizontal and vertical, misalignments and shears [17, 18].

Another rotor effective wind speed estimation technique, is the immersion and invariance (I&I) estimator [19], inspired by the eponymous identification method described in [20]. The technique assumes the rotor speed and applied generator torque being available as measured signals. The technique shows satisfactory estimation results, is conveniently implemented, and is therefore included in the DRC.

*2.4. State-machines, and baseline pitch and torque control*

This section presents the concept of state-machines, and the pitch and torque control implementations included in the DRC. The purpose of the global state-machine is to determine the operational state of the wind turbine, and is included as a subroutine in the function module. Inside this function, separate state-machines are included for pitch and torque control. The operational state is determined by comparing measured turbine quantities and control signals to settings defined in the controller configuration file. Figure 2 gives an overview of the different wind turbine operating regions, indicating internal controller variables and configuration parameters.

The pitch and torque controllers use a common filtered generator speed measurement, to calculate the error from the reference set point. At the end and beginning of regions 1 and 2.5, two PI torque controllers are implemented to regulate the rotor speed towards the optimal below-rated torque path, and to above-rated operating conditions. As shown in Figure 2, the maximum torque controller saturation can be set independently to facilitate less frequent switches between torque and pitch control. The cut-in speed for below-rated torque control (Region 2) can be defined, as well as the optimal mode-gain for tracking the maximum rotor power coefficient $C_{\mathrm{p,max}}$. The baseline pitch controller is implemented as a gain-scheduled PI-controller, and the

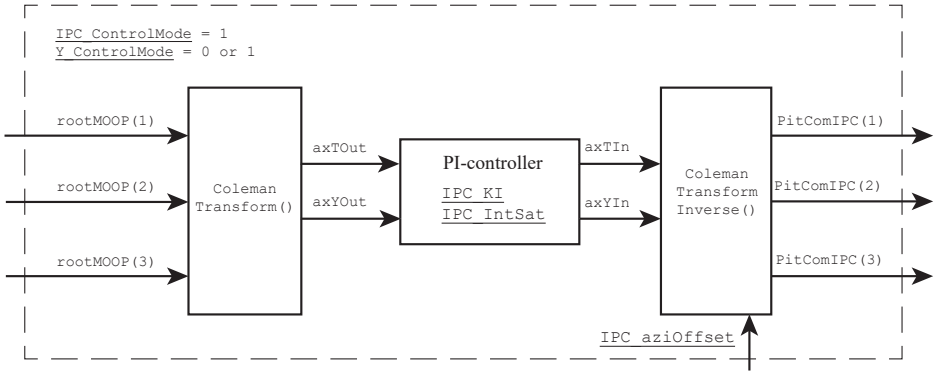



Figure 3: Individual pitch control for blade fatigue load reductions. The out-of-plane blade root moments are transformed in a tilt- and yaw-axis by a 1P Coleman transformation. After PI-control, the resulting pitch angles are transformed back by an inverse Coleman transformation to obtain IPC pitch signals to mitigate 1P fatigue loadings.

gain information is defined in the parameter file as a function of pitch angles.

For power regulation during above-rated operation, the torque controller can be configured to either deliver a constant torque signal, or actively change the torque signal to obtain a constant power output. For constant power tracking, the torque signal varies subject to the measured generator speed and the electrical power set point.

### *2.5. Fatigue load control*

Fatigue load reduction capabilities are available in the DRC, in the form of individual pitch control (IPC) for periodic blade load reductions, and active tower fore-aft damping. The two implementations are respectively described in this section.

*Individual Pitch Control.* IPC reduces out-of-plane blade oscillations causing fatigue, by adding contributions to the individual pitch control signals. A schematic IPC implementation overview is presented in Figure 3. The measured blade root out-of-plane moments, together with the rotor azimuth angle, are taken as input to the forward Coleman (or multiblade coordinate (MBC)) transformation [21], resulting in non-rotating rotor tilt- and yaw-moments. The IPC implementation in the DRC allows for attenuation of the 1P blade load harmonic, or the combined 1P+2P periodic loads. A phase offset can be added to the azimuth angle in the reverse transformation, which turns out to be crucial for practical IPC implementations [22].

*Tower fore-aft damping.* Tower fore-aft oscillations are naturally lightly damped by aerodynamic damping [23]. To further enhance damping of fore-aft oscillations, an active control strategy can be implemented. Active fore-aft damping uses an integrated nacelle acceleration signal, which is added to the collective pitch signal [12]. The acceleration signal is possibly additionally filtered by a notch filter to prevent unwanted actuation at, e.g., the blade passing frequency.

### *2.6. Yaw control*

To maximize energy extraction from the wind, the rotor axis of a wind turbine needs to be aligned with the dominating wind direction. Because the wind flow direction changes over time, a yaw system is required to keep the orientation of a wind turbine aligned with the wind direction

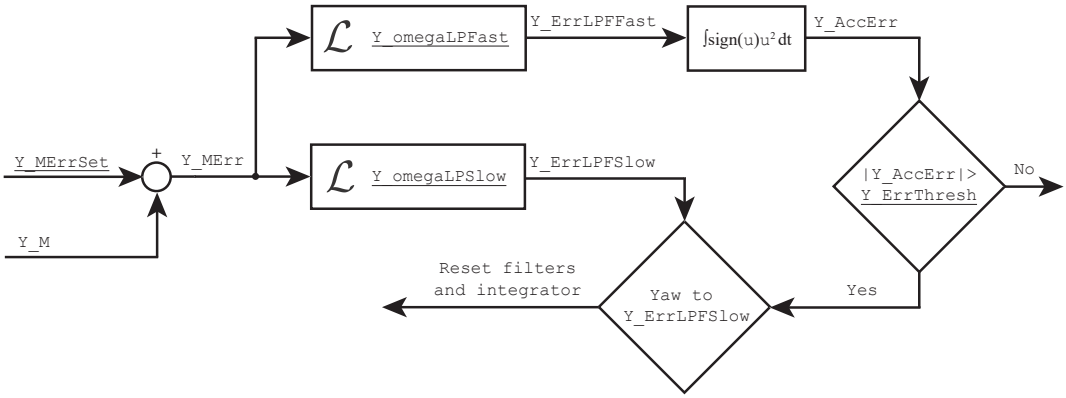



Figure 4: Yaw rate control uses the error between the misalignment set point and the measured misalignment with the dominating wind direction to intermittently perform yaw manoeuvres.

to capture as much energy as possible [24, 11]. Yawing the wind turbine nacelle and rotor on the support structure can be achieved in different ways, for example, by active yaw and free yaw-by-IPC implementations. Both implementations are included in the DRC, and are respectively described in this section.

*Yaw-rate control.* The yaw-rate control implementation does not provide continuous alignment, but intermittently aligns the turbine nacelle when a predefined threshold is exceeded. The implementation adapted from [25] and schematically depicted in Figure 4, and is slightly adjusted to allow for yaw-angle offsets. The yaw-rate controller uses measurements from a wind vane located downwind, i.e., seen from upwind the vane is positioned behind the rotor and tower. The wind vane measures the nacelle yaw-misalignment with respect to the dominating wind direction, but does not give information on the absolute nacelle orientation. Yaw motors with a fixed yaw-rate are used for yaw movements.

*Yaw-by-IPC.* Besides of the common fatigue load reduction implementation, IPC can also be configured to act in a yaw-by-IPC set-up. Figure 5 shows a schematic overview of the

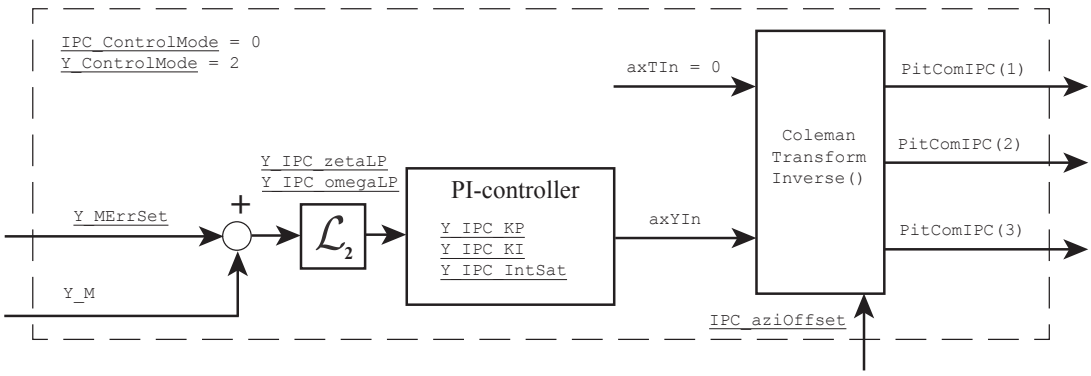



Figure 5: An IPC yaw control implementation for a wind turbine where the nacelle is mounted on the tower, in a free-damped fashion. The non-rotating tilt pitch angle is nullified, and the yaw angle is actively controlled by the error between the set point and measured yaw misalignment.

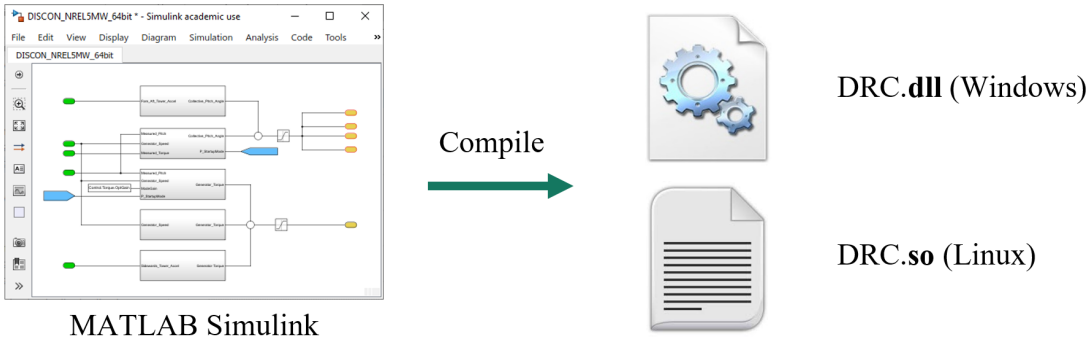



Figure 6: SimulinkDRC allows to graphically design a wind turbine controller in Simulink. Compilation result in a dynamic library with the DISCON controller interface, which is compatible with a wide variety of wind turbine simulation software.

implementation. With yaw-by-IPC, a 1P contribution is added to the pitch signals resulting in a yaw moment over the entire rotor to actively regulate or track a yaw misalignment set point. Normally, this type of control is present in downwind wind turbines, where the nacelle is mounted in a free-damped fashion on the tower support structure [26, 27]. In the DRC, either the fatigue load reduction or the yaw-by-IPC controller is active. Simultaneous operation is possible by separating frequency activity, however, such an implementation requires in-depth knowledge and careful tuning [26].

## 3. SimulinkDRC: graphical controller design and compilation

The SimulinkDRC package provides a more convenient and graphical controller design environment in MATLAB Simulink [28]. All built-in Simulink objects and functions can be used, and compilation results into dynamic library files: *.dll* for Windows, and *.so* for Linux. The compiled controller uses the same DISCON controller interface as the DRC. The implementation includes custom code to compile a 32-bit dynamic library using a 64-bit version of MATLAB Simulink, which is helpful for use with older 32-bit versions of Bladed and FAST.

The tool has proven to be insightful in the development phase of new control algorithms. At the time of writing, a lightweight controller for the NREL 5-MW reference turbine is included. However, as not every engineer or scientist has access to Simulink, implementation of (novel) control algorithms is prioritized in the open-source DRC written in Fortran. The next development goal is to bring the controller functionality of SimulinkDRC on par with the Fortran-based DRC, sharing the functionality and support for external controller parameter files.

## 4. FASTTool: an educational GUI for FAST

The FASTTool was developed in the wake of another educational project at Delft University of Technology (TU Delft), where the idea was to teach an online course through *gamification*. Students would play a game in which they can design and test their own wind turbine through various levels, each with a new challenge to overcome (e.g., wind shear, setting the right cut-out wind speed, etc.). There are plenty of examples of such games with a good educational value. For instance, playing SimCity teaches the essentials of urban planning, Poly Bridge teaches about statics and truss structures, and Kerbal Space Program teaches about rocketry and orbital mechanics. In fact, Kerbal Space Program was so successful in this that NASA began to actively contribute to the game as a means of public outreach and getting young people excited for spaceflight. The strength of these games, in terms of teaching a subject, lies

in the fact that playing them does not feel as a chore and that there is good educational value in trying and failing a level. Although FASTTool was not developed with gamification in mind, the relatively simple user interface and many graphical elements were designed to lower the learning curve of the software and make it more enjoyable to use. This enables students to spend most of their time on substantive aspects of wind turbine design, while limiting distractions from the, often complex, capabilities of commercially available simulation and analysis software.

The strength of many wind turbine aeroelastic tools lies in the analysis and not in the user-friendliness per se, which hampers the usability in educational courses. This aspect results in students often losing themselves in the vast array of options. In FASTTool, this problem was solved by centering the graphical user interface (GUI) around an animated three-dimensional plot of the wind turbine. Changes to the geometry are immediately visible on screen, which provides students with an immediate sanity check, but also gives the feeling of creating something new. The performance of the turbine can be checked by a quick power curve calculation (e.g., to see the impact of rotor diameter), as well as through a full time series analysis by FAST, for which the input files are generated by the tool.

Summarizing, FASTTool is a wind turbine design, assessment and simulation tool. It is used in a master-level course on wind turbine design, in which students construct a turbine and assess its performance, dynamics and limit states. The design starts with a choice for system-level parameters and then focuses on the rotor, drivetrain, tower and controller design. As the name of the software already suggests, the simulation back-end is based on NREL's FAST v8.16 (Fatigue, Aerodynamics, Structures, and Turbulence), which is a high-fidelity open-source wind turbine simulation software package [29]. The software is to date still under active development, and updated regularly based on new insights and feedback from students. The tool is publicly available at no cost as an open-source repository [30]. This section gives a high-level overview of the FASTTool, and is organized as follows: Section 4.1 outlines the capabilities of the graphical user interface, and Section 4.2 demonstrates the back-end simulation and control environment.

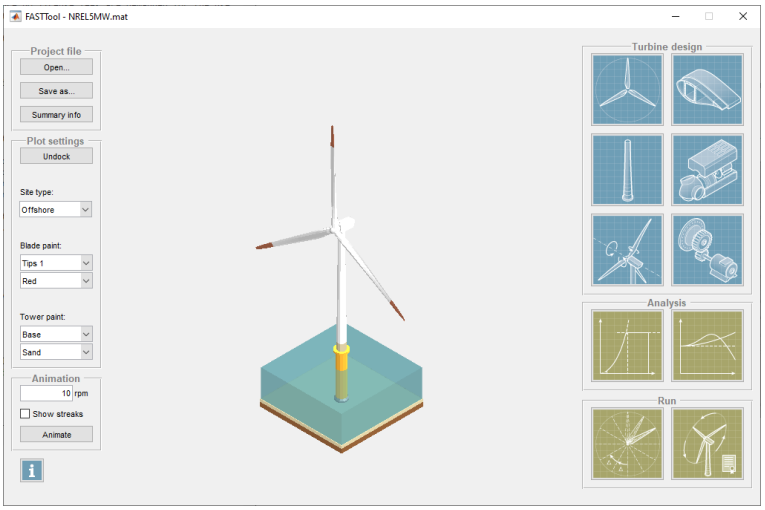



Figure 7: The main window of FASTTool. The wind turbine plotted in the center of the screen is animated and adapts to the current turbine design. The turbine's visual appearance can be changed by the options on the left-hand side of the screen. The design – in terms of blade, tower, nacelle, drivetrain, and controller – is altered by the blue-colored options on the right. Analysis and simulation functionality – steady-state operating curves, modal analysis, linearization and simulation – is included under the yellow-colored buttons.

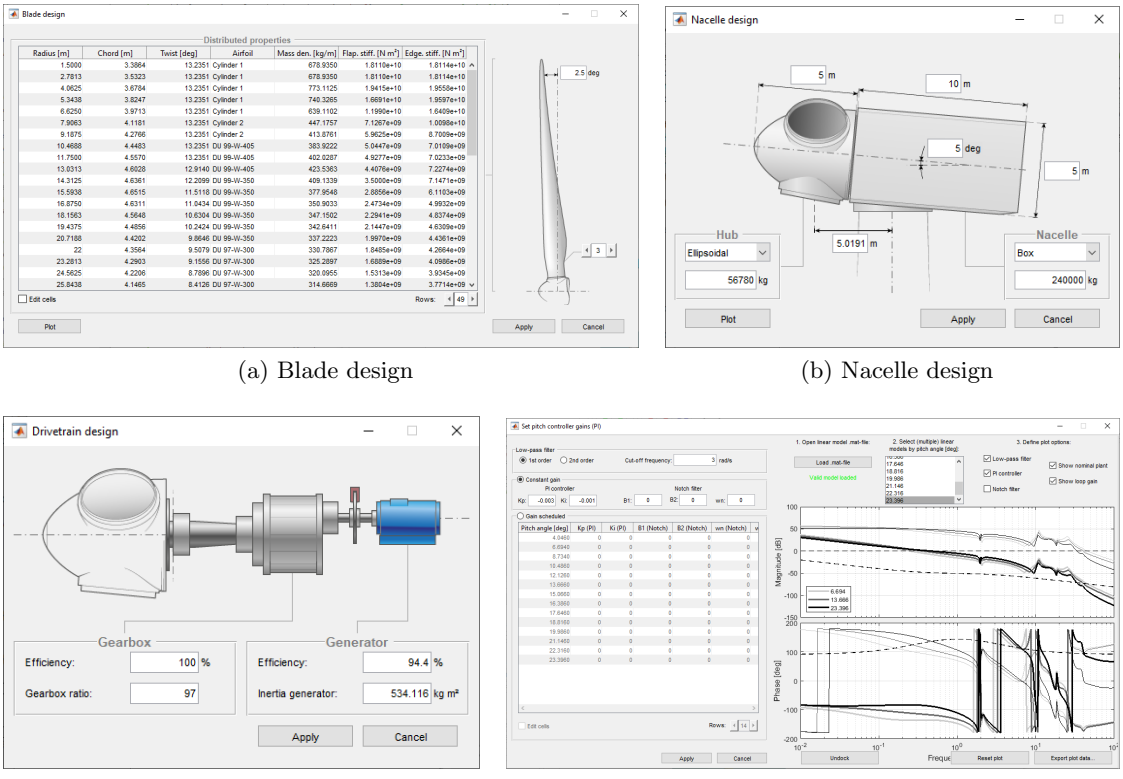
### *4.1. MATLAB-based graphical user interface*

The costs for industry standard wind turbine simulation software are often prohibitive for use in educational courses, considering the amount of students involved. For this reason, FASTTool is developed in MATLAB/Simulink [28], in conjunction with the publicly available FAST simulation code. The choice of software is convenient for an academic environment, since no license fees are demanded for the use of FAST, while often students have access to and experience with MATLAB/Simulink. Although MATLAB and Simulink require a license, the employed environment provides flexibility and insight for both the end-user and developer. The MATLAB scripts and the Simulink model of the tool can be edited by more expert users to add or change functionality and to enable other inputs and outputs.

Figure 7 presents the main window of the FASTTool, which gives access to the following functionality:

(i) A three-dimensional animated wind turbine visualization, adapting to the current design defined by parameters in the graphical user interfaces for blade, tower, nacelle, drivetrain and controller design.
(ii) Determination of steady-state performance, calculation of turbine natural frequencies, and visualization in a Campbell diagram.
(iii) Linearization of the non-linear turbine dynamics at the controller-defined operational path.
(iv) High-fidelity simulation of various load cases, by a wide variety of wind profiles.

Each item in the above-given enumeration is discussed in the remaining paragraphs of this section.

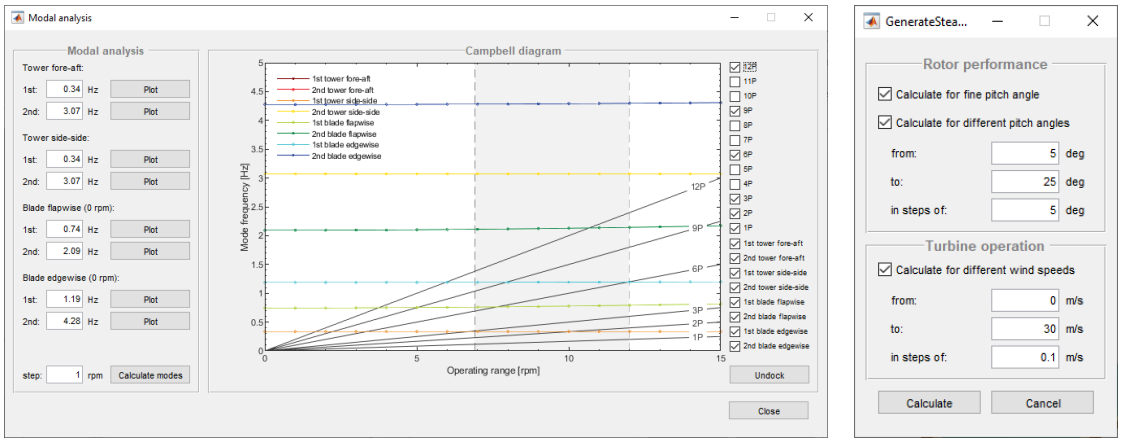



(a) Blade design (b) Nacelle design

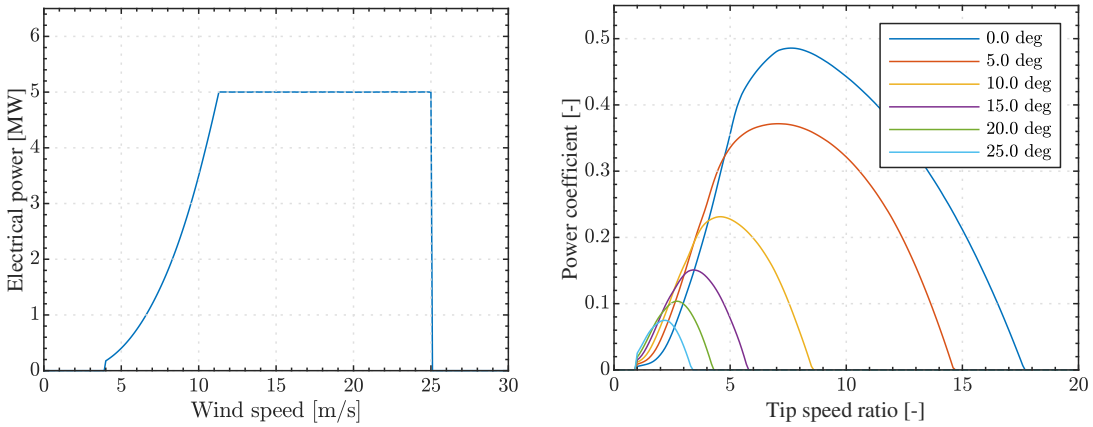
(c) Nacelle design (d) Controller design

Figure 8: Different design modules of the FASTTool. The GUI provides a convenient way of changing the blade geometry, nacelle sizing, and drivetrain parameters. An extensive interface is available for controller design by loop-shaping techniques using standard filters.

*1. Structural, drivetrain and controller design.* The various graphical interfaces for structural, drivetrain and controller design are presented in Figure 8. Because FASTTool is built for educational purposes, the software is supplied with the NREL 5-MW reference wind turbine [1] as a MATLAB-style `.mat`-file, and student designs are based on scalings of this turbine. The blade design window in Figure 8a provides functionality to radially specify the blade geometry and structural properties by defining the chord, twist and airfoil for each node, as well as the mass density, flap- and edgewise stiffness. A similar interface is provided for tower design. The user can also edit airfoil properties or add new airfoils. Figures 8b and 8c respectively present the nacelle and drivetrain design options: Parameters size the nacelle, and define the drivetrain by efficiencies, the gearbox ratio, and the generator inertia. To easily check for mistakes in geometric data inputs, the blade, tower and turbine-nacelle configuration are assessed with graphical visualizations. Finally, the controller design component is shown in Figure 8d. The controller design section allows to visually tune the pitch controller by loop shaping the system's frequency responses. Loop-shaping is performed by tuning standard PI, low-pass and




(a) Modal analysis of turbine structural components (b) Rotor performance




(c) Operational power curve (d) Steady-state rotor power coefficients

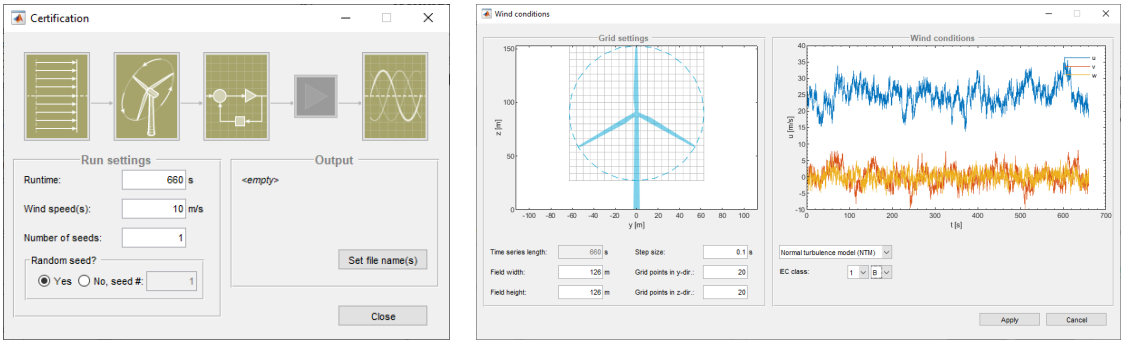
Figure 9: Modal analysis tools for calculating the tower and blade, first and second natural frequencies, and visualizing the results in a Campbell diagram. The diagram helps to identify problematic interactions with the variable turbine rotational frequency in the below-rated region. Furthermore, the operational path and steady-state rotor performance mappings are calculated based on the blade design, rotor configuration, and control strategy.

notch control modules. A comparison can be made between a fixed-gain, and a gain-scheduled controller, to advocate the performance advantages of a variable-gain control implementation. Several other control parameters, such as for the feed forward partial load torque control and for the braking action, can also be changed by the user.

*2. Steady-state rotor performance and modal analysis.* As a result of the structural geometry and mass properties, Figure 9a shows that a modal analysis can be performed on the tower fore-aft and side-side modes, along with the blade flap- and edgewise modes. The natural frequencies are determined with `BModes`, which is also developed by NREL as part of the FAST suite of tools [31]. To analyze whether these structural modes interfere with the varying rotational $n$P harmonics, a Campbell diagram is plotted on the right hand side of the window. Figure 9b shows the configuration window for steady-state performance calculations. Steady-state mappings can be calculated at a predefined range of pitch angles, based on the blade and rotor configuration, and with use of included blade-element momentum (BEM) code. The overall turbine operational behavior as a function of wind speed is shown in Figure 9c, whereas Figure 9d shows the result of a the rotor power coefficient calculation as a function of pitch angle and tip-speed ratio (TSR). The figures are generally used to find the maximum power coefficient, rated wind speed and best pitch angle setting for partial load operation.

*3. Linearization.* A wide arsenal of powerful mathematical and frequency domain techniques is available for linear controller design. For this reason, high-fidelity non-linear models are generally linearized at operating points of interest. At the time of writing, FAST includes linearization functionality, however, unlike previous versions, it lacks the capability of finding operational trim conditions. The latter mentioned aspect is included in FASTTool. After finding the trim points for a predefined range of wind speeds, the input values are provided to an open-loop fixed-time FAST simulation that linearizes the model. The linear model is stored in a `.mat`-file, which is needed to support the controller design.

*4. Wind load cases and simulation.* When the wind turbine design is completed, FASTTool provides the opportunity to run certification simulations, as shown in Figure 10a. This means that as in Figure 10b, first a desired wind field is selected and dimensioned. The user can choose various wind conditions, such as steady wind, stepped wind speed changes, a normal or extreme




(a) Certification simulation (b) Wind profile design

Figure 10: Certification and wind field design windows. The certification section allows to define the total simulation time and mean wind speed of the high-fidelity FAST run. The wind field design window offers among others the selection of steady, stepped, or turbulent wind profiles. Turbulent wind files are generated using NREL's TurbSim [32].

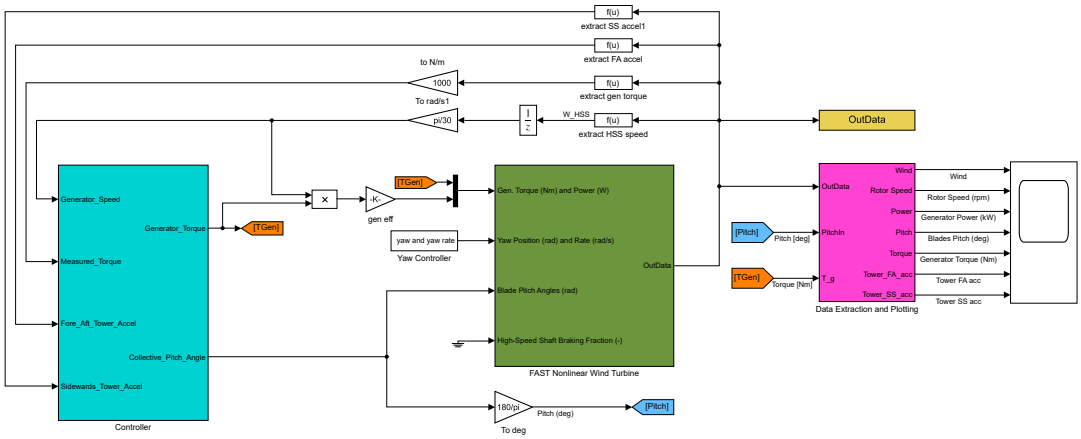



Figure 11: Simulink implementation of the FAST wind turbine simulation code (green block) and the controller (cyan block). Data extraction and manipulation (magenta block) for plotting purposes is handled on the right-hand side, and allows for convenient evaluation of simulation results, resulting from the turbine design and controller tuning.

turbulence model, an extreme operating gust. The more complex wind fields are generated by NREL's TurbSim [32]. Wind profiles can be set for assessment of the behaviour of the design and the controller, or to run a load case according to the IEC 61400-1 standard [33]. Various turbine conditions can be chosen, such as power production, grid loss, normal or emergency shutdown and idling, supporting IEC load case assessment. Then, when the output filename is defined, a certification simulation is initiated. For this, FASTTool takes the user-defined turbine design parameters and generates the corresponding FAST input files, after which it starts a high-fidelity non-linear FAST simulation, implemented using an S-Function in Simulink. To avoid an overload of information, a small (but relevant) selection of the vast amount of signal outputs is made available to the user; an experienced user can extend the list of outputs. The next section outlines the FAST Simulink simulation and controller environment.

### *4.2. Simulink-based controller and simulation environment*

FAST has the ability to either run as a compiled standalone application on Windows and Linux, or have the FAST dynamic library being called by a Simulink S-Function. As shown in Figure 11, the latter mentioned implementation is employed by FASTTool, as it provides a convenient and insightful development environment. During a simulation run, the built-in controller of FAST is disabled and the controller is provided by Simulink blocks, configured with information from the different interfaces. The Simulink implementation offers course participants, who want to gain a deeper understanding of wind turbine simulation and control, an accessible way of doing so. Experienced user can even change the controller and can for instance add active yaw control.

## 5. Conclusions

Three software projects are discussed in this paper. First a community-driven wind turbine baseline controller is presented, applicable to high-fidelity simulation software that uses the DISCON controller interface. The controller aims in being the reference controller for evaluation of new control algorithms. The controller architecture is such that it can be used for any wind turbine model. A single parameter file configures the controller, which abandons the need for recompilation under a change in controller settings. Because of the modular set-up, the existing baseline control implementations are easily replaced, which enables for convenient

comparison, reproducibility, and evaluation of new algorithms. Second, a Simulink tool for convenient graphical design and compilation of a turbine controller is demonstrated. Finally, FASTTool is showcased, which is a graphical user interface for NREL's aeroelastic simulation code FAST for educational purposes in wind turbine and controller design. FASTTool provides people new to the field with insights in the design process, by visualizing changes in a three-dimensional turbine visualization, adapting to the current design. The software has options for quick sanity checks, and can generate FAST input files to run high-fidelity simulations based on the turbine design.

With the aim of supporting, standardizing and solidifying the wind turbine (research) community, all software is open-source and publicly available at an online repository. The repositories are regularly updated, and users are invited to provide feedback and contribute to the projects.

## References

[1] J. Jonkman et al. *Definition of a 5-MW reference wind turbine for offshore system development*. Tech. rep. Golden, Colorado: National Renewable Energy Laboratory (NREL), 2009.

[2] C. Bak et al. "Design and performance of a 10 MW wind turbine". In: *Wind Energy* (2013).

[3] DNV-GL. *Bladed*. `https://www.dnvgl.com/energy/generation/software/bladed/index.html`. [Online; accessed 14-November-2017]. 2017.

[4] M. H. Hansen and L. C. Henriksen. "Basic DTU wind energy controller". In: *DTU Wind Energy* (2013).

[5] Garrad Hassan & Partners Ltd. *Bladed User Manual*. Version 4.2. 2011.

[6] S. P. Mulders and J. W. van Wingerden. "Delft Research Controller: an open-source and community-driven wind turbine baseline controller". In: *Journal of Physics: Conference Series*. Vol. 1037. 3. IOP Publishing. 2018.

[7] T. M. Lahey and T. Ellis. *Fortran 90 programming*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[8] S. P. Mulders and J. W. van Wingerden. *Delft Research Controller (DRC)*. `https://github.com/TUDelft-DataDrivenControl/DRC_Fortran`. 2019.

[9] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[10] K. Z. Østergaard, P. Brath, and J. Stoustrup. "Estimation of effective wind speed". In: *Journal of Physics: Conference Series*. Vol. 75. Bristol, United Kingdom: IOP Publishing, 2007.

[11] F. D. Bianchi, H. De Battista, and R. J. Mantz. *Wind turbine control systems: principles, modelling and gain scheduling design*. Springer Science & Business Media, 2006.

[12] E. A. Bossanyi. "The design of closed loop controllers for wind turbines". In: *Wind Energy* 3.3 (2000), pp. 149–163.

[13] E. Hau. *Wind turbines: fundamentals, technologies, application, economics*. Berlin, Germany: Springer Science & Business Media, 2013.

[14] M. N. Soltani et al. "Estimation of rotor effective wind speed: A comparison". In: *IEEE Transactions on Control Systems Technology* 21.4 (2013), pp. 1155 –1167.

[15] E. van der Hooft and T. van Engelen. "Estimated wind speed feed forward control for wind turbine operation optimisation". In: *European Wind Energy Conference (EWEC)*. ECN-RX–04-126. ECN. London, United Kingdom, Nov. 2004.

[16] T. Knudsen, T. Bak, and M. Soltani. "Prediction models for wind speed at turbine locations in a wind farm". In: *Wind Energy* 14.7 (2011). DOI: 10.1002/we.491.

[17] K. Selvam et al. "Feedback-feedforward individual pitch control for wind turbine load reduction". In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 19.1 (2009), pp. 72–91.

[18] M. Bertelè et al. "Wind inflow observation from load harmonics". In: *Wind Energy Science* 2.2 (2017), pp. 615–640. DOI: 10.5194/wes-2-615-2017.

[19] R. Ortega, F. Mancilla David, and F. Jaramillo. "A globally convergent wind speed estimator for wind turbine systems". In: *International Journal of Adaptive Control and Signal Processing* 27.5 (2013). DOI: 10.1002/acs.2319.

[20] X Liu et al. "Identification of nonlinearly parameterized nonlinear models: application to mass balance systems". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC)*. Conference on Decision and Control (CDC). 2009, pp. 4682–4685. DOI: 10.1109/CDC.2009.5399817.

[21] G. Bir. "Multi-blade coordinate transformation and its application to wind turbine analysis". In: *46th AIAA aerospace sciences meeting and exhibit* (2008).

[22] S. P. Mulders et al. "Analysis and optimal individual pitch control decoupling by inclusion of an azimuth offset in the multiblade coordinate transformation". In: *Wind Energy* 22.3 (2019), pp. 341–359. DOI: 10.1002/we.2289.

[23] T. Burton et al. *Wind energy handbook*. Chichester, United Kingdom: John Wiley & Sons, 2001.

[24] J. F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind energy explained: theory, design and application*. John Wiley & Sons, 2010.

[25] K. Kragh and P. Fleming. "Rotor Speed Dependent Yaw Control of Wind Turbines Based on Empirical Data". In: *AIAA Aerospace Sciences Meeting* AIAA 2012.1018 (2012).

[26] E van Solingen et al. "Control design for a two-bladed downwind teeterless damped free-yaw wind turbine". In: *Mechatronics* 36 (2016), pp. 77–96.

[27] V. Schorbach and P. Dalhoff. "Two bladed wind turbines: antiquated or supposed to be resurrected". In: *Proceedings of the EWEA Conference*. 2012.

[28] MathWorks. *MATLAB / Simulink*. https://www.mathworks.com. 2019.

[29] NWTC Information Portal. *FAST v8.16*. https://nwtc.nrel.gov/FAST8. [Online; accessed 27-August-2019]. 2019.

[30] R. Bos et al. *FASTTool*. https://github.com/TUDelft-DataDrivenControl/FASTTool. 2019.

[31] NWTC Information Portal. *BModes*. https://nwtc.nrel.gov/BModes. Sept. 2014.

[32] NWTC Information Portal. *TurbSim*. https://nwtc.nrel.gov/TurbSim. June 2016.

[33] IEC. *IEC 61400-1 third edition 2005-08: Wind turbines - Part 1: Design requirements*. Tech. rep. International Electrotechnical Commission (IEC), 2005.