Next Generation Wind Turbine Simulation

The QBlade Software

QBlade is an advanced multi-physics software designed to facilitate the comprehensive aero-servo-hydro-elastic development, prototyping, simulation, and certification of wind turbines.

This powerful tool enables highly detailed simulations of wind turbine designs, featuring physics models that are more than 30 times faster than real-time. All of this functionality is accessible through an intuitive and user-friendly graphical interface.

QBlade is available as standalone software for both Windows and Linux operating systems, with two distinct releases tailored for non-commercial and commercial applications.

You can utilize QBlade for various purposes, including:

- Wind Turbine Design (HAWT, VAWT and Multi-Rotor)
- Floater Design and Numerical Testing
- Controller Testing and Optimization
- IEC Design Load Certification
- Wind Farm Simulations and Wake Interaction Studies
- Digital Twins in Python or Matlab Environments

v 2.0.7

QBlade Documentation

QBlade [1] is a state-of-the-art multi-physics code covering the complete range of aspects required for the aero-servo-hydro-elastic simulation of horizontal or vertical axis wind turbines. This software, developed since 2010, is implemented as a modular system of highly efficient multi-fidelity aerodynamic, structural dynamic, and hydrodynamic solvers within a modern, object-oriented C++ framework.

Advanced Performance and User-Friendly Interface

We leverage the current computer architecture by thoroughly utilizing CPU (via OpenMP) and GPU (via OpenCL) parallelization techniques for high numerical performance. QBlade is platform-independent software, deployable on workstations or clusters running Windows or Unix based operating systems. The software is equipped with an intuitive graphical user interface that aids users throughout the wind turbine design process. All turbine and simulation details are readily accessible and modifiable within a logical, well-structured, and tested graphical user interface (GUI). Simulation results are presented in dynamic graphs that provide insight into every simulation detail. Simulations and turbine designs are fully rendered in real time to aid in the comprehension and evaluation of our complex multi-physics models. QBlade enables the serialization of complete model data, setup, and results into project files to facilitate simple sharing and collaboration on complex simulation and turbine design projects. The Community Edition of QBlade (QBlade-CE) is freely available under the Academic Public License, while the Enterprise Edition (QBlade-EE) is available under a Commercial License.

Aerodynamics

QBlade uses a highly optimized and thoroughly validated Lifting Line Free Vortex Wake Method for its aerodynamic calculations. Instead of approximating the wake aerodynamics with a steady-state momentum balance (BEM), the rotor wake is explicitly modeled through Lagrangian vortex elements. This results in a more accurate and detailed spatial and temporal representation Marten *et al.*[2] of the rotor induction compared to BEM approaches, and fully resolves the velocity distribution behind the rotor. This allows for the assessment of wind turbine wake interaction, accurately accounts for the aerodynamics of oscillating floating wind turbine structures, and explicitly resolves unsteady vertical axis wind turbine wake dynamics Balduzzi *et al.*[3]. As an alternative with lower computational demand, the aerodynamics of horizontal-axis wind turbines can be simulated using an unsteady polar implementation Madsen *et al.*[4].

Structural Dynamics

The structural dynamics are modeled in a true multi-body formulation. The subcomponents of the multi-body model consist of rigid or flexible nonlinear Euler or Timoshenko beam elements in a corotational formulation. For floating offshore simulations, QBlade uses integrated cable elements in the absolute nodal coordinate formulation (ANCF), which meet the requirements to effectively model the nonlinear dynamics of complex mooring systems.

Hydrodynamics

The hydrodynamic loads on the wind turbine's substructure are calculated either via potential flow theory, the Morison equation-based strip theory, or a user-defined combination of the two. The integrated potential flow approach also includes second order sum- and difference frequency loads obtained from quadratic transfer functions (QTFs). QBlade integrates with potential flow data from common software such as WAMIT, NEMOH, or similar toolboxes.

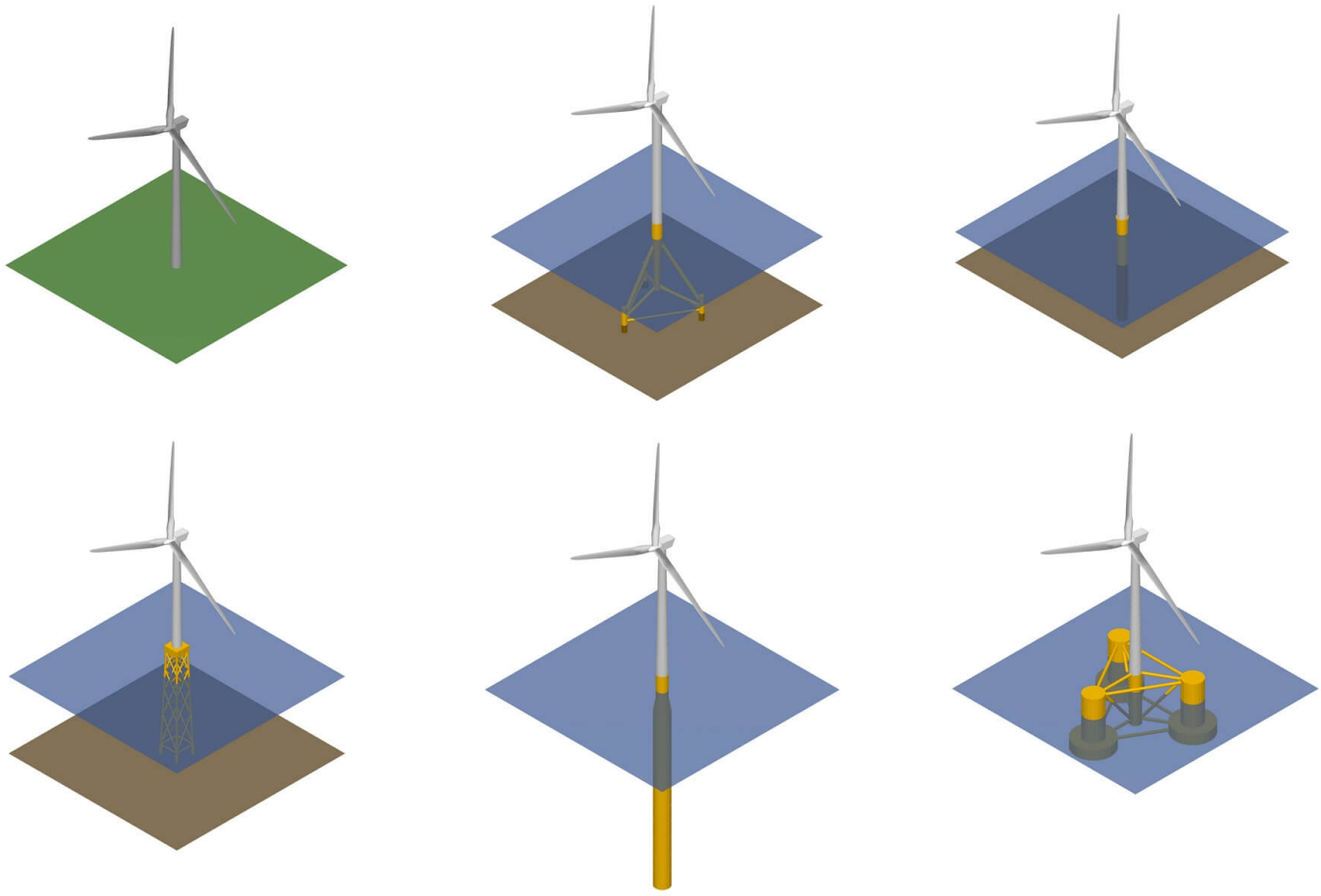


Fig. 1 *Different wind turbine types modelled with QBlade.*

- QBlade Documentation
- Theory Guide
    - Aerodynamics

[1] D. Marten. *QBlade: A Modern Tool for the Aeroelastic Simulation of Wind Turbines*. PhD thesis, TU Berlin, 2019. doi:10.14279/depositonce-10646.

[2] D. Marten, C. O. Paschereit, X. Huang, M. Meinke, W. Schröder, J. Müller, and K. Oberleithner. Predicting wind turbine wake breakdown using a free vortex wake code. *AIAA Journal*, 58(11):4672–4685, 2020. URL: https://doi.org/10.2514/1.J058308.

[3] Francesco Balduzzi, David Marten, Alessandro Bianchini, Jernej Drofelnik, Lorenzo Ferrari, Michele Sergio Campobasso, Georgios Pechlivanoglou, Christian Navid Nayeri, Giovanni Ferrara, and Christian Oliver Paschereit. Three-Dimensional Aerodynamic Analysis of a Darrieus Wind Turbine Blade Using Computational Fluid Dynamics and Lifting Line Theory. *Journal of Engineering for Gas Turbines and Power*, 140(2):022602, 2017. doi:10.1115/1.4037750.

[4] H. A. Madsen, T. J. Larsen, G. R. Pirrung, A. Li, and F. Zahle. Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact. *Wind Energy Science*, 5(1):1–27 URL: https://wes.copernicus.org/articles/5/1/2020/, doi:10.5194/wes-5-1-2020.

Theory Guide

Aerodynamics

- Blade Element Momentum Method
- Dynamic Wake Meandering Model
- Lifting Line Free Vortex Wake
- Dynamic Stall
- Secondary Effects

Hydrodynamics

- Linear Potential Flow Theory
- Morison Equation
- Hydrostatic Forces

Structural Dynamics

- Multi Body Beam Formulation
- Aero-Elastic Coupling
- Structural Data Tables

Environmental Conditions

- Soil
- Waves
- Wind

Blade Element Momentum Method

In QBlade the aerodynamic forces acting on a rotor can be modeled either using a steady Blade Element Momentum (BEM) or a with a more advanced, time resolved unsteady BEM (UBEM) which is enhanced by several correctional models. The theory interlinks the actuator disc theory and the blade element theory and it was first introduced by Glauert[1]. Despite its simplicity, the BEM method allows for an accurate representation of the steady aerodynamic loads that act on the rotor of a wind turbine, provided certain model assumptions are not violated.

Momentum Theory

Under the assumptions of a steady, incompressible and axisymmetric inflow of an inviscid fluid the actuator disc theory may be applied. The rotor plane is treated as an actuator disk that causes a uniform pressure drop over the rotor area while the flow velocity varies continuously through the disk. In its simplest form, the actuator disc theory assumes that the velocity through the rotor plane does not contain a tangential component and the pressures far up- and downstream of the rotor are equal to the ambient pressure. These assumptions allow for the calculation of the rotor performance (power and thrust) and the velocity in the rotor plane by invoking the conservation of mass and momentum (see Branlard[2]). The introduction of the induction factor a allows for the expression of the velocity in the rotor plane as function of the incoming velocity u_0:

$$u = (1 - a)u_\infty.$$

The rotor performance coefficients for power and thrust may also be expressed as a function of the axial induction factor a:

$$C_T = 4a(1 - a),$$

$$C_P = 4a(1 - a)^2.$$

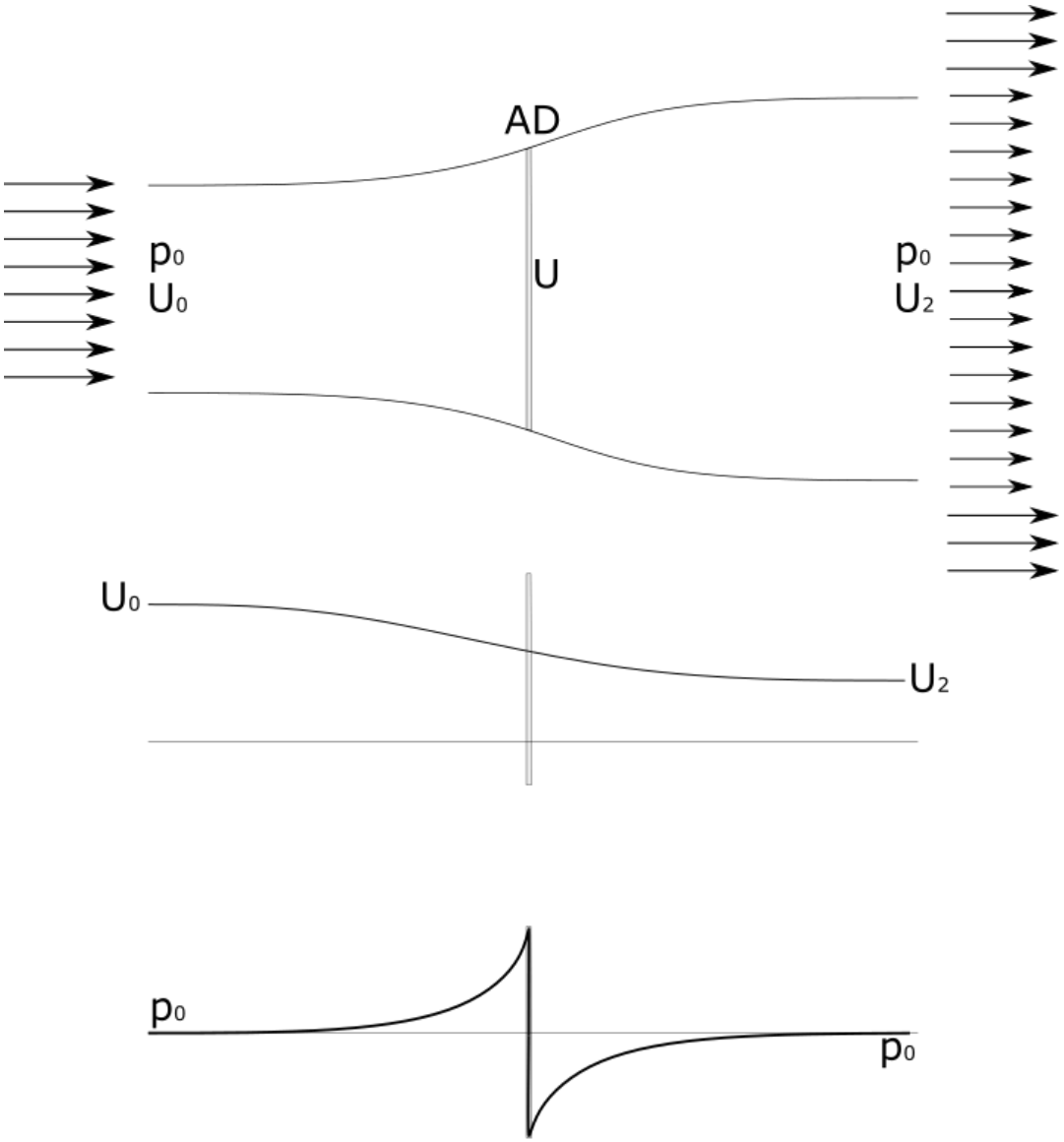


Fig. 2 *1D momentum theory, pressure and velocity evolution.*

Blade Element Theory

The blade element theory allows for the computation of the loads acting on a rotor based on the geometric and aerodynamic properties of individual spanwise blade sections. The blade is divided into a discrete number of radially distributed sections. The loads on each section are calculated under the assumption that the flow there is locally two-dimensional and in the plane of the airfoil section. This allows for the use of two-dimensional lift, drag and moment coefficients together with the relative flow velocity to determine sectional airfoil forces.

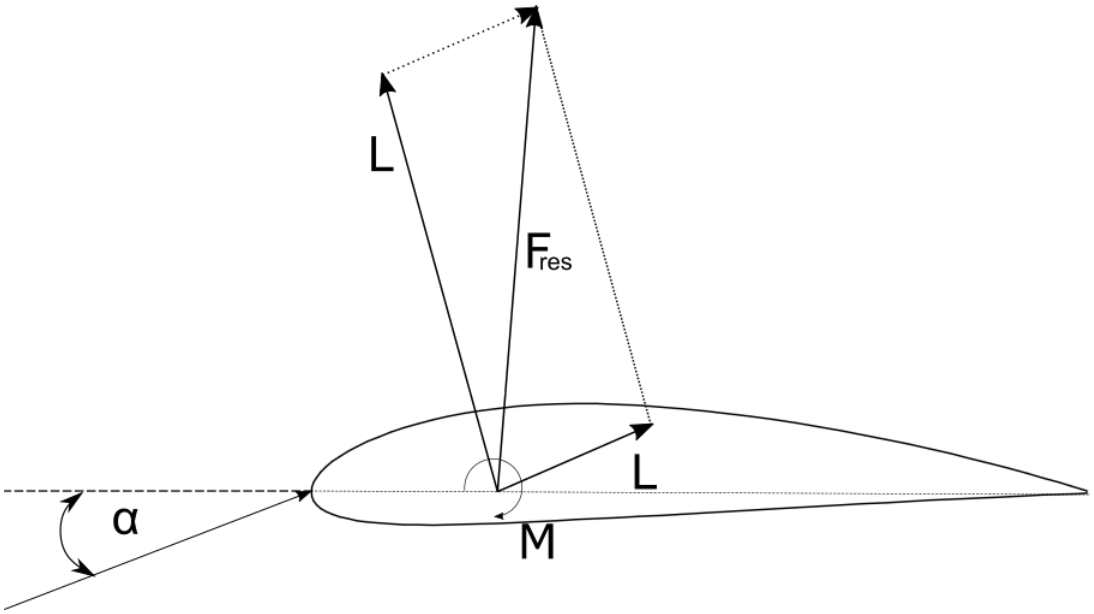


Fig. 3 *2D forces on an airfoil.*

Classical Blade Element Momentum Theory

The blade element momentum theory combines the 1D actuator disc theory with the blade element theory. For practical reasons, the stream tube theory is applied to radial annuli that match the discretization of the blade elements. Both theories allow for the expression of the blade forces within an annular segment as a function of streamtube geometric properties and the axial and tangential induction factors a and a', respectively. A solution method is applied which iteratively finds the values of a and a' which satisfy both theories.

Corrections

Due to the two dimensional nature of the BEM theory, three dimensional effects are not accounted for by the classical BEM. This leads to large deviations of flow quantities compared with measured turbine data, particularly in regions where strong changes in the blade circulation occur. To improve the accuracy of the BEM results, two correctional methods are implemented into QBlade:

- Prandtl Tip Loss Factor (see Glauert[1]);
- 3D Correction (see H. Snel, R. Houwink, W. J. Piers[3]);

Unsteady Blade Element Momentum Theory

Although the classical BEM method provides good estimates of the annual energy production, it is incapable of accounting for unsteady phenomena like the atmospheric boundary layer, turbulence or the tower influence. These unsteady phenomena make the position of each blade at a certain time necessary. Hence, non-rotating coordinate systems are placed at the bottom of the tower ar nacelle. Furthermore, a coordinate system is attached to the rotating shaft and each blade. The instantaneous velocity seen by each blade can now be determined and be accounted for in the

calculation of the flow angle (Tavares da Silva and Donadon[4]). As the classical BEM is only valid for steady flow cases and provides induced velocities corresponding to this specific state, a dynamic inflow model is used to introduce a time lag to the sectional rotor induction (see Glauert[1] or Henriksen *et al.*[5]).

Polar Grid

The polar-grid has been developed by (Madsen *et al.*[6]) to consider azimuthal variations of the axial induction caused by the azimuthal dependence of blade loadings. Within the approach, the annular rings of the actuator disc theory are divided into stationary azimuthal sections. Each point on the azimuthal grid is associated with a local induction factor, based on the local instantaneous velocity. The latter is approximated by the induced velocity of the neighboring two blades and weighted by their azimuthal distance (Behrens de Luna *et al.*[7]).

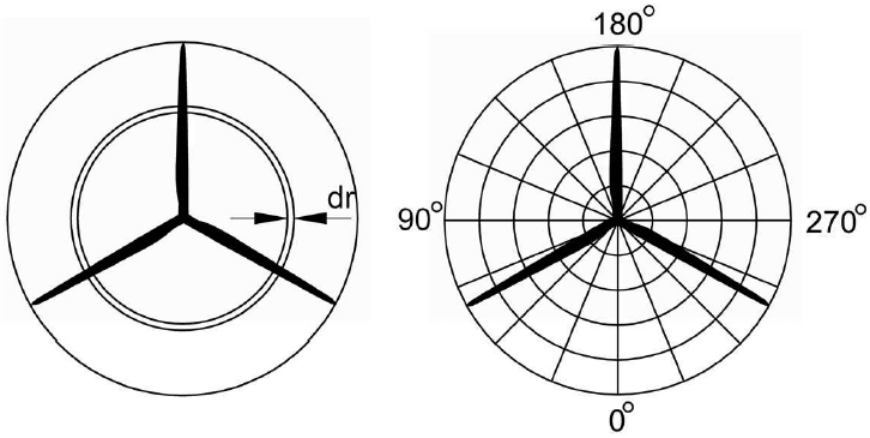


Fig. 4 *Classical BEM approach (left) and polar grid with azimuthal sub elements (right), taken from Madsen et al.*[6])

[1] (1, 2, 3) H. Glauert. *Airplane Propellers*, chapter Aerodynamic Theory, pages 169–360. Springer Berlin Heidelberg, 1935. doi:10.1007/978-3-642-91487-4_3.

[2] Emmanuel Branlard. *Wind Turbine Aerodynamics and Vorticity-Based Methods*. Volume 7. Springer International Publishing AG, 01 2017. ISBN 978-3-319-55163-0. doi:10.1007/978-3-319-55164-7.

[3] H. Snel, R. Houwink, W. J. Piers. Sectional Prediction of 3D Effects for Separated Flow on Rotating Blades. In *Proc. European Community Wind Energy Conference*. Lübeck - Travemünde, 1992.

[4] Claudio Tavares da Silva and Mauricio Donadon. Unsteady blade element-momentum method including returning wake effects. *Journal of Aerospace Technology and Management*, 5:, 03 2013. doi:10.5028/jatm.v5i1.163.

[5] L.C. Henriksen, M.H. Hansen, and Niels Poulsen. A simplified dynamic inflow model and its effect on the performance of free mean wind speed estimation. *Wind Energy*, 16:, 09 2012. doi:10.1002/we.1548.

[6] (1, 2) Helge Madsen, Torben Larsen, Georg Pirrung, Ang Li, and Frederik Zahle. Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact. *Wind Energy Science*, 5:1–27, 01 2020. doi:10.5194/wes-5-1-2020.

[7] R. Behrens de Luna, D. Marten, T. , Barlas, S.G. Horcas, N. Ramos-Garcia, A. Li, and C.O. Paschereit. Comparison of different fidelity aerodynamic solvers on the IEA 10MW turbine including novel tip extension geometries, Journal of Physics Conference Series [accepted]. *Journal of Physics: Conference Series*, 2022.

Dynamic Wake Meandering Model

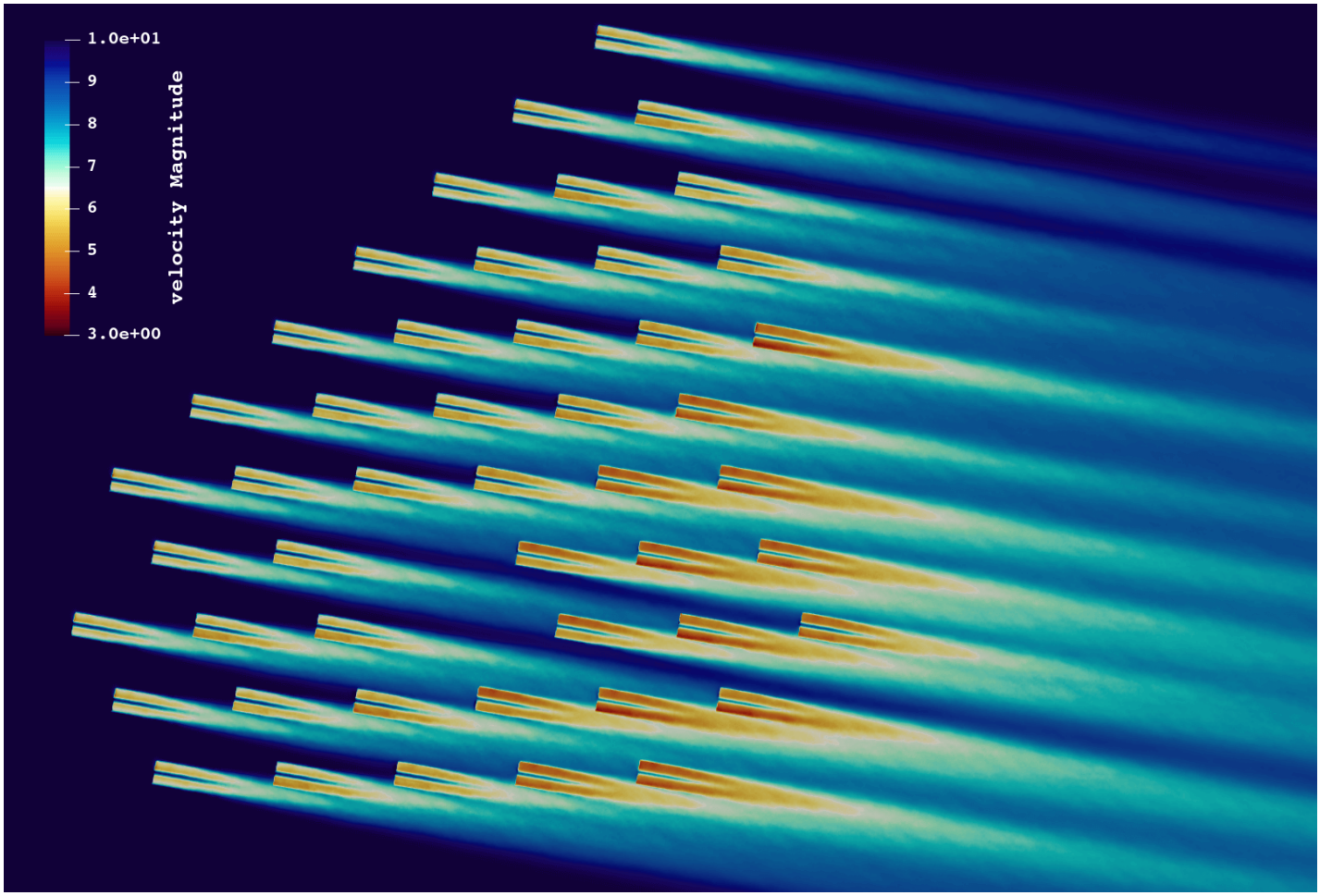


Fig. 5 *Exemplary application of the DWM model to the simulation of the Lillgrund Wind Farm in QBlade.*

The Dynamic Wake Meandering (DWM) model in QBlade represents an approach to simulating the wake behind a wind turbine rotor, when its aerodynamics are simulated with the unsteady BEM theory. The DWM model captures the essential dynamics of wind turbine wakes, providing a robust framework for analyzing how wakes influence the operation and loads on downstream turbines. The DWM model is able to simulate wake behavior quickly and accurately, making it an ideal tool for investigations into wind park layout or control optimization.

The DWM model simulates the meandering motion of the wake generated by a wind turbine. Unlike static wake models, which assume a steady-state flow, the DWM model accounts for the time-dependent nature of wakes, incorporating both the advection and diffusion of wake structures over time. This dynamic approach allows for a realistic representation of wake effects, including:

- **Meandering Motion**: The turbulence induced lateral and vertical movement of the wake as it propagates downstream.
- **Wake Expansion**: The spreading of the wake caused by turbulent mixing with the ambient wind flow.

- **Velocity Deficits**: The reduction in wind speed within the wake, which affects downstream turbines.
- **Turbulence Intensification**: The increase in turbulence levels within the wake, impacting the loading on downstream turbines.

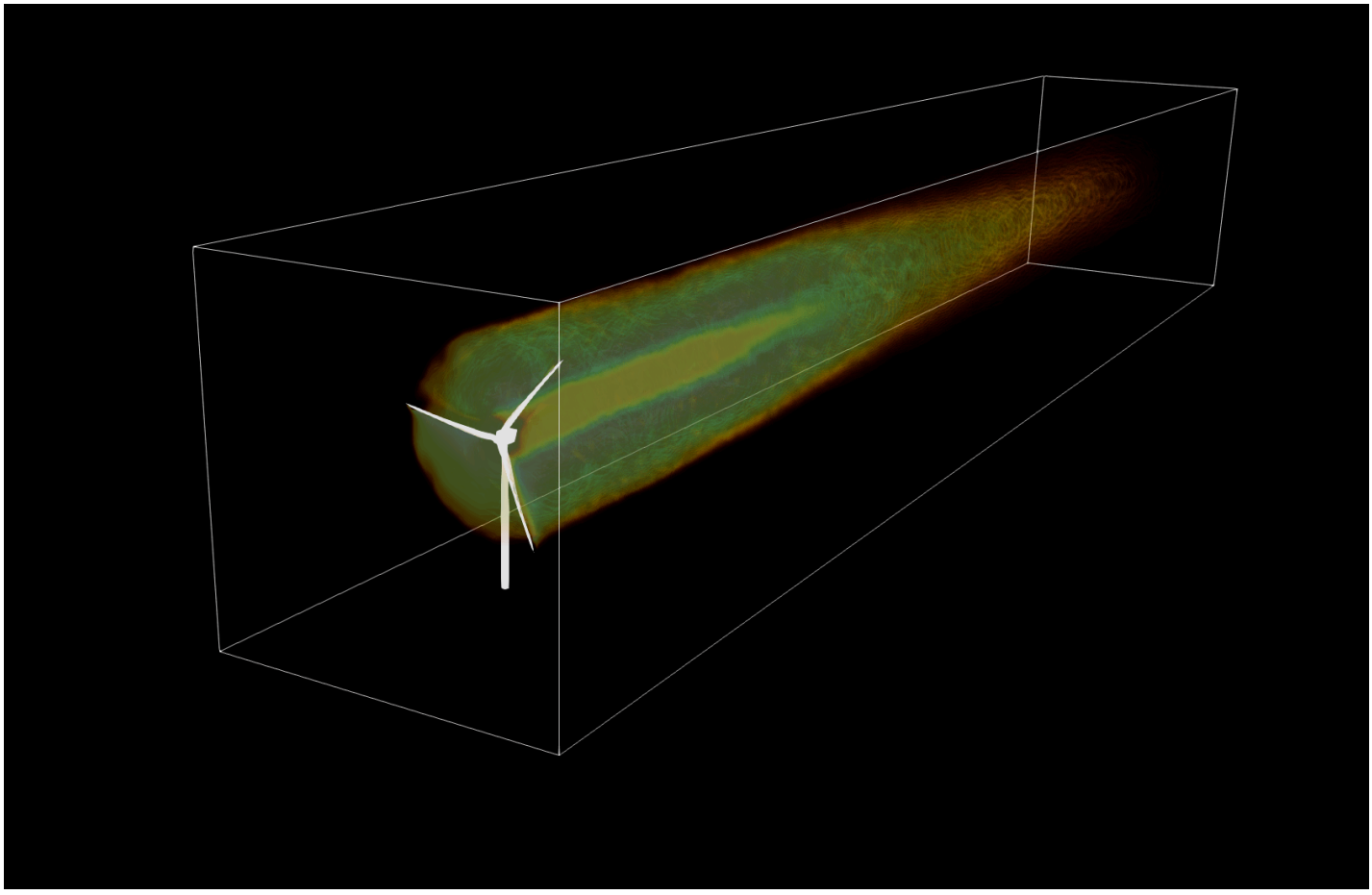


Fig. 6 *3D Visualization of the wake velocity deficit evaluated with the DWM model.*

Info

This section will be expanded in the near future...

Lifting Line Free Vortex Wake

In QBlade the aerodynamic forces acting on a rotor can be modeled using the Lifting Line Free Vortex Wake method (LLFVW). Similar to the Blade Element Momentum Method, in the LLFVW model the blade forces are calculated using two dimensional sectional airfoil polar data. The main difference is that the rotor wake, shed from the blades, is explicitly resolved. This is a large improvement over the commonly used Blade Element Momentum Method style approaches, which necessitate the introduction of a large number of empirical corrections into the simulated system. Modeling the wake dynamics explicitly avoids the dependency on such correction models and often lead to more physically sound results. Simulation results are improved especially in cases where the assumptions of the Blade Element Momentum Method are violated. These include unsteady operation, large blade deformations and high tip speed ratios where the turbulent wake state is approached. Such conditions become more and more prevalent with the ongoing trend towards larger rotor sizes and offshore floating wind turbines.

Overview of LLFVW Theory

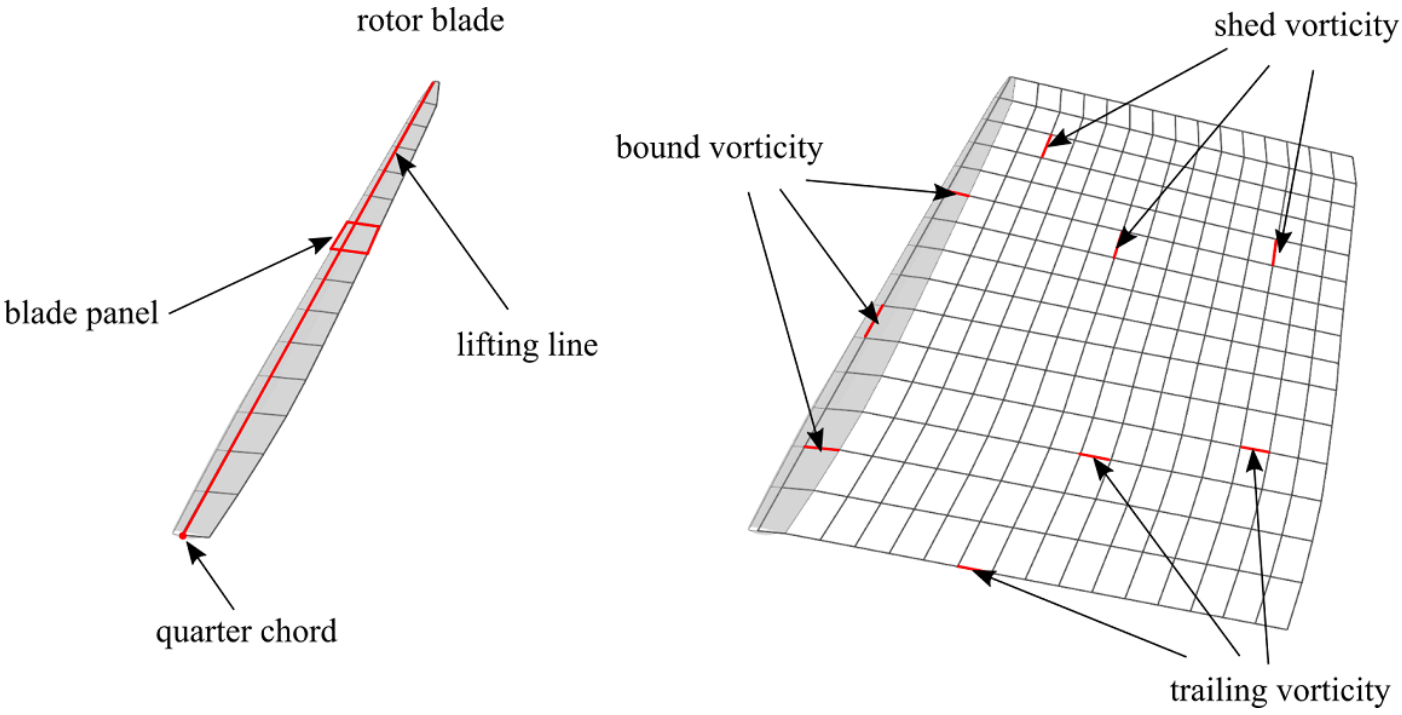


Fig. 7 *Basic elements of the blade and wake model inside the LLFVW algorithm.*

The rotor is represented by a lifting line, located at the quarter chord position of the 2D airfoil sections (see Fig. 7). Each blade panel is represented by a vortex ring which consists of four straight vortex filaments. The circulation of the bound vortex lines, forming the lifting line, is calculated from the relative inflow velocity and the lift and drag coefficients that are obtained from tabulated data. The sectional circulation $\partial\Gamma$ is calculated according to the Kutta-Joukowski theorem:

$$\partial F_L(\alpha) = \rho V_{rel} \times \partial \Gamma,$$

where ∂F_L is the sectional lift force and ρ is the fluid density. The relative velocity V_{rel} is obtained from a simple vector addition of the free stream velocity V_∞, the blade motion V_{mot} and the induced velocity V_{ind}, which is calculated from the contribution of all vortex elements on the blade and in the wake through the Biot-Savart equation:

$$V_{ind} = -\frac{1}{4\pi} \int \Gamma \frac{\vec{r} \times d\vec{l}}{r^3}.$$

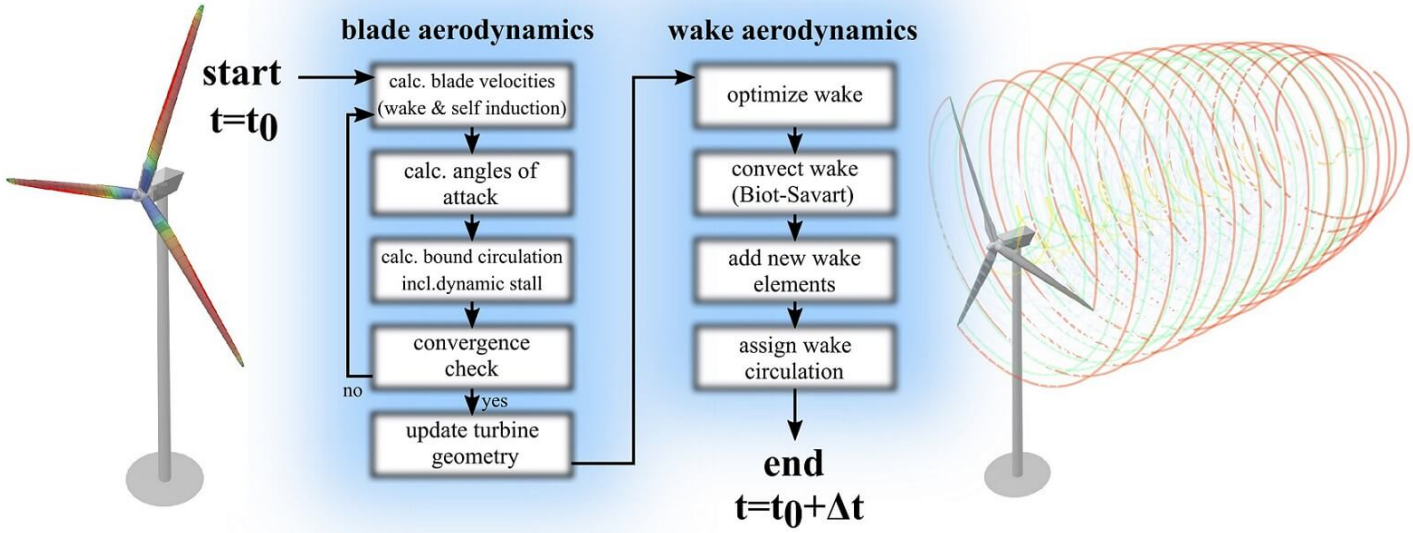


Fig. 8 *Flowchart for a single timestep of the aerodynamic calculations in QBlade*

At the beginning of each time step, the circulation distribution along the blade is calculated. This is carried out with an iterative procedure which ensures that the forces predicted by the Kutta-Joukoswki theorem and the blade element theorem coincide. During the iteration only the bound vorticity distribution is updated, while the induction of the wake elements on the blade is only evaluated once. After convergence is obtained, the rotor rotation is advanced for a single time step. All free wake vortex elements are convected with the local inflow and local induced velocity. After the wake convection step, new vortex elements are created between the trailing edge of each blade panel and the last row of wake vortices that were convected away from the trailing edge. As a last step, the circulation is computed and assigned to the new released vortex lines through the Kutta condition:

$$\Gamma_{trail} = \frac{\partial \Gamma_{bound}}{\partial x} \Delta x.$$

$$\Gamma_{shed} = \frac{\partial \Gamma_{bound}}{\partial t} \Delta t.$$

Wake Lattice and Connectivity

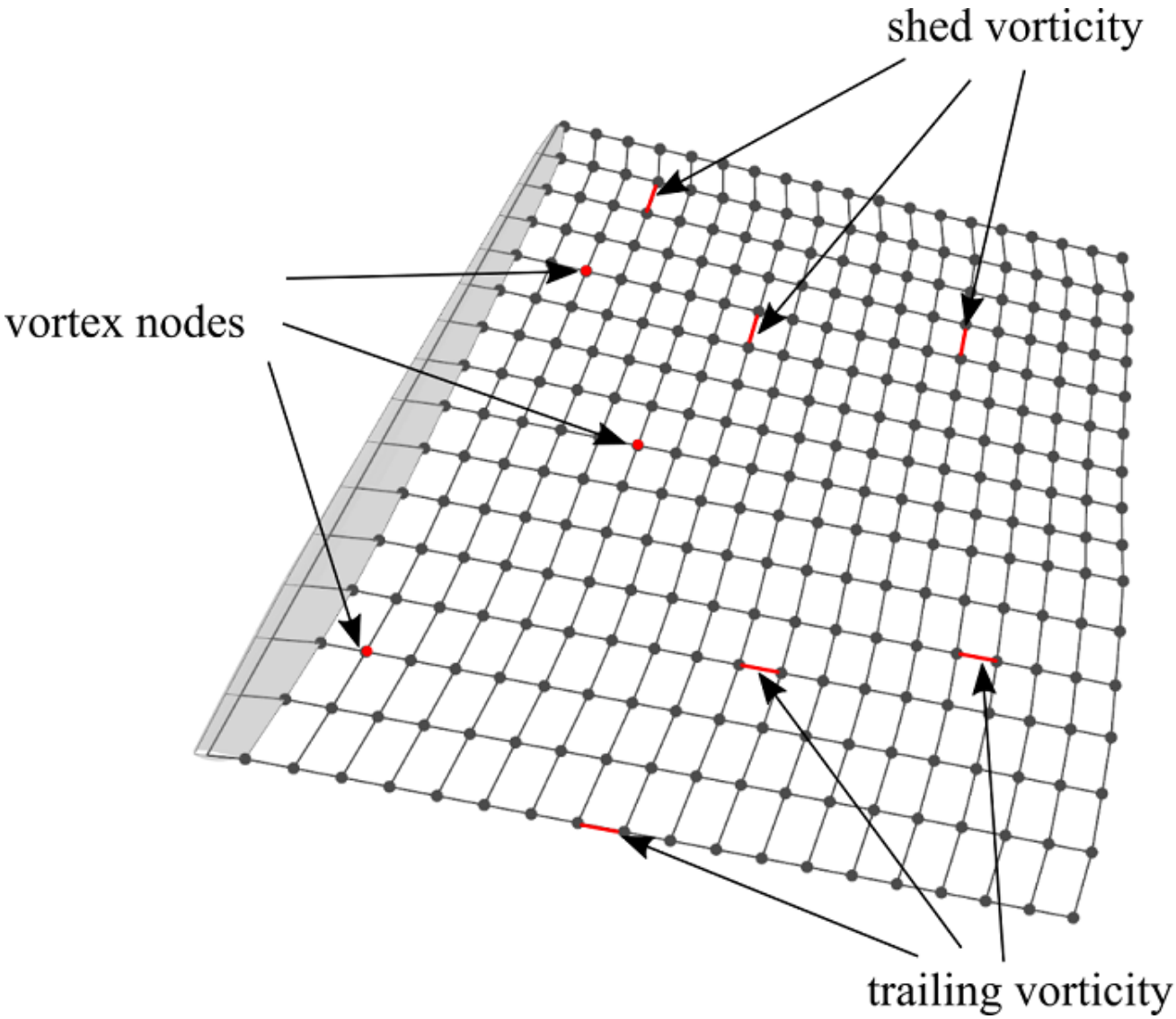


Fig. 9 *Visualization of the wake lattice structure with wake nodes and filaments.*

Fig. 9 shows the wake lattice structure. Shed- and trailing vortices are interconnected via common vortex nodes. During the free wake convection step the evolution of the wake is evaluated by advancing the positions of the vortex nodes in time. Each newly created vortex node is attached to at least one shed and one trailing vortex filament, thus the total number of vortex nodes is approximately half the number of vortex filaments. Consequently, the Biot-Savart equation has to be evaluated around:

$$N_{nodes} \cdot N_{vortices} \approx \frac{N_{2vortices}}{2}$$

times for a fully populated (assuming that no vortex elements have been removed) infinite wake lattice. Compared to a vortex particle discretization, where no inter-connectivity exists, this a reduction in computational cost by a factor of 2, due to the inter-connectivity of the wake la o facilitate strategies that reduce the number of free vortices within the wake a method to remove individual vortices from the wake mesh has been implemented whereby vortex filaments are

detached from their corresponding nodes. A check is performed during every step of the simulation that removes isolated vortex nodes which are not attached to any vortex filament. The more vortices have been removed from the wake lattice, the lower the aforementioned leverage of the interconnections.

Vortex Core Desingularization

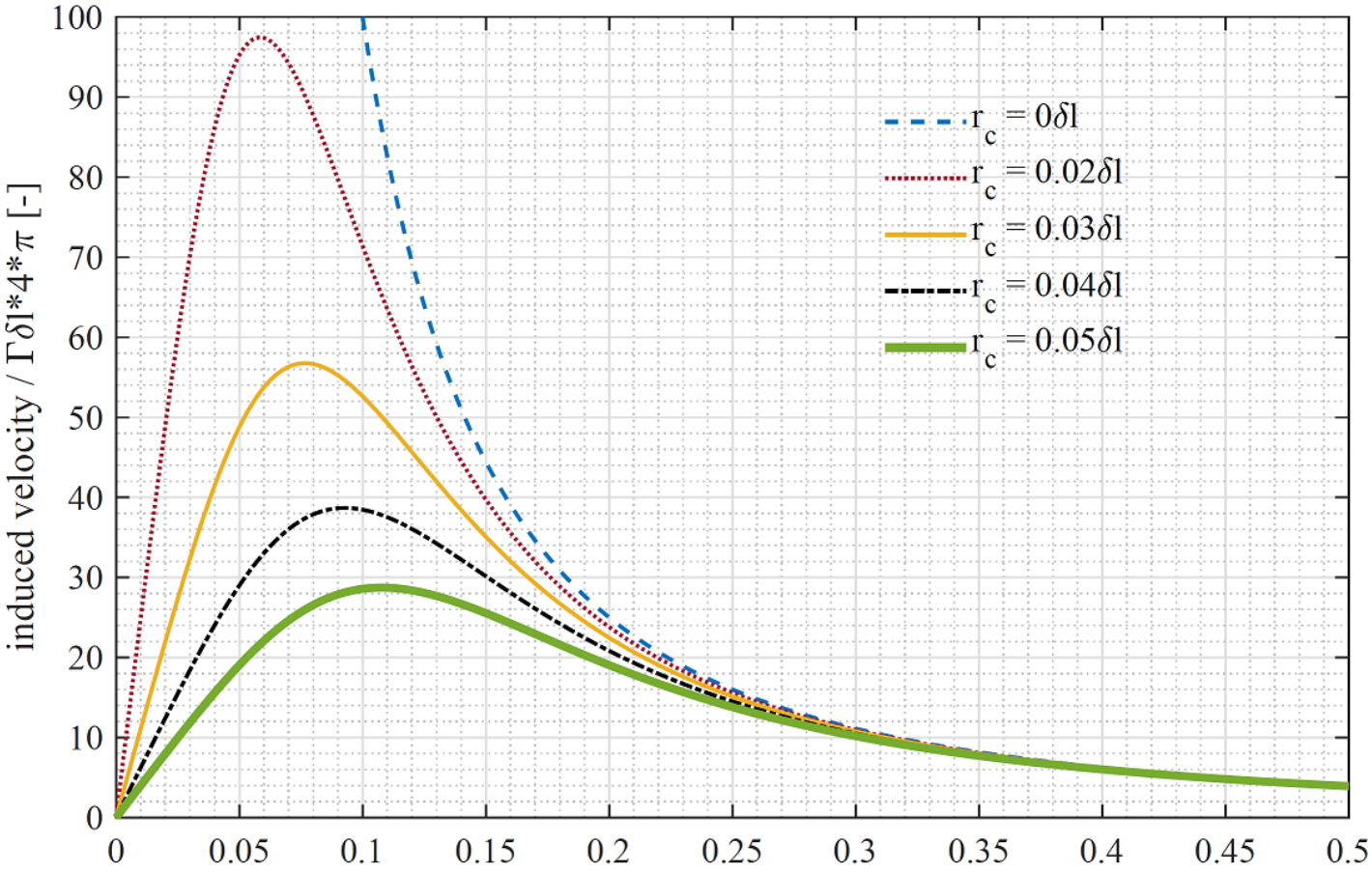


Fig. 10 *Velocity distribution around the vortex core.*

The Biot-Savart equation exhibits a singularity at the core where $\vec{r} = 0$ (Fig. 10). To prevent this singularity from affecting the stability of the simulation and also to model the viscous core of the bound and free vortices more accurately, a model for a viscous vortex core was implemented. Many different models that describe the tangential velocity distribution around the core exist, such as the Rankine, Lamb-Oseen or Ramasay and Leishman models (see Hommes *et al.*[1]). In QBlade a simple cut-off radius is used, which is added to the denominator of this equation in the form of r_{2c}, and ensures that the induced velocity smoothly approaches zero in the vicinity of the core. This is a computationally efficient implementation as the viscous core modeling is directly implemented in the calculation of the induced velocity. For other vortex models a viscous parameter needs to be evaluated from the relative vortex positions in addition to the Biot-Savart equation. This has a severe effect on the simulation performance, as the evaluation of the viscous parameter is carried $N_{2vortices}/2$ times per time step. When shed from the trailing edge of the blade, a vortex is with an initial core-size r_c (a value of around 10% of local chord is proposed from experienc core-size is updated every time step according to:

$$r_c = r_0 + \sqrt{\frac{4a\delta_v \nu \Delta t}{1+\epsilon}}.$$

where $a = 1.25643$ is a constant, δ_v is the turbulent viscosity coefficient (a value depending on rotor size, see Sant[2]), ν is the kinematic viscosity and Δt the time step size. The strain rate of the vortex filament is computed as:

$$\epsilon = \frac{\Delta l}{l}.$$

The desingularized Biot-Savart equation then becomes:

$$V_{ind} = -\frac{1}{4\pi} \int \Gamma \frac{\vec{r} \times \partial \vec{l}}{r^3 + r_{2c}}.$$

[1] T. Hommes, J. Bosschers, and H. W.M. Hoeijmakers. Evaluation of the radial pressure distribution of vortex models and comparison with experimental data. *Journal of Physics: Conference Series*, 656(1):7–11, 2015. doi:10.1088/1742-6596/656/1/012182.

[2] Tonio Sant. *Improving BEM - based Aerodynamic Models in Wind Turbine Design Codes Improving BEM - based Aerodynamic Models Tonio Sant Tonio Sant Improving BEM-based Aerodynamic Models in Wind Turbine Design Codes*. TU Delft, 2007. ISBN 978-99932-0-483-1.

Dynamic Stall

- OYE Model
- IAG Model
- Gormont-Berg Model
- ATEFlap Model

OYE Model

In QBlade dynamic stall may be modeled in unsteady Lifting Line Free Vortex Wake or Blade Element Momentum Method simulations by using the dynamic stall model proposed by Oye [1]. It should be noted that this model only captures the dynamics of separated flow. The additional attached flow dynamics due to airfoil wake memory effects are captured intrinsically by the Lifting Line Free Vortex Wake model. In its implementation in QBlade the Oye dynamic stall model is only applied within the angle of attack range of -50° to 50°.

In Oye's work the dynamic stall is modeled with th help of a separation function f. It is used to calculate the dynamic lift Cl_{dyn} in the following way:

$$Cl_{dyn} = fCl_{att} + (1 - f)Cl_{sep}.$$

Where Cl_{att} is the fully attached inviscid lift contribution and Cl_{sep} the fully separated lift contribution. To solve the equation above, it is applied to steady conditions. The result is:

$$Cl^{st} = f^{st}Cl_{statt} + (1 - f^{st})Cl_{stsep},$$

where the superscript st refers to steady conditions. Comparing the equations above, $Cl_{att} = Cl_{statt}$ and $Cl_{sep} = Cl_{stsep}$. Cl^{st} is obtained by reading in static polar data and Cl_{statt} is obtained by extrapolating the linear part of the lift curve to the required angle of attack. Following Hansen [2], f^{st} can be calculated in the following way:

$$f^{st} = \left(2\sqrt{\frac{Cl^{st}}{Cl_{statt}}}\right)^2.$$

The value of f^{st} is limited to be between 0 and 1. Now f is assumed to return to the static value f^{st} as follows:

$$\frac{df}{dt} = \frac{f^{st} - f}{\tau}.$$

Integrating the equation above allows the determination of the dynamic behavior of $f(t)$:

$$f(t) = f^{st}(t) + \left(f(t - \Delta t) - f^{st}(t)\right)e^{\frac{-\Delta t}{\tau}}.$$

τ is a time constant, defined as:

$$\tau = \frac{A\frac{c}{2}}{V_{rel}},$$

where A is a parameter (typically around 8), c is the airfoil chord and V_{rel} the relative velocity at the airfoil section. After f has been evaluated, Cl_{sep} can be obtained from:

$$Cl_{sep} = \frac{Cl^{st} - f^{st} Cl_{att}}{1 - f^{st}}.$$

Cl_{att} is gained by extrapolation of the linear lift curve.

Now, all variables have been determined and the dynamic lift Cl_{dyn} can be computed.

In QBlade, the Oye dynamic stall model also determines a dynamically changing drag coefficient Cd_{dyn}. After Bergami, the dynamic drag Cd_{dyn} is evaluated as:

$$Cd_{dyn} = Cd^{st} + (Cd^{st} - Cd_{st0})(0.5(\sqrt{f^{st}} - \sqrt{f})) - 0.25(f - f^{st}).$$

In this equation Cd_{st0} is the drag at 0 degree angle of attack.

[1] Stig Oye. Dynamic stall simulated as time lag of separation. 1991.

[2] Morten Hartvig Hansen, Mac Gaunaa, and Helge Aagaard Madsen. A Beddoes-Leishman type dynamic stall model in state-space and indicial formulation. Technical Report, DTU, 2004.

IAG Model

The recently developed (2020) IAG dynamic stall model, [1], [2] can be chosen to model the dynamic stall in QBlade.

The implementation in QBlade considers the first order model in indicial formulation presented in [2].

[1] G. Bangga, T. Lutz, and M. Arnold. An improved second-order dynamic stall model for wind turbine airfoils. *Wind Energy Science*, 5(3):1037–1058, 2020. URL: https://wes.copernicus.org/articles/5/1037/2020/, doi:10.5194/wes-5-1037-2020.

[2] (1, 2) Galih Bangga, Steven Parkinson, and William Collier. Development and validation of the iag dynamic stall model in state-space representation for wind turbine airfoils. *Energies*, 2023. URL: https://www.mdpi.com/1996-1073/16/10/3994, doi:10.3390/en16103994.

Gormont-Berg Model

For the simulation of dynamic stall on VAWT, especially at low tip speed ratios, where the angle of attack variations can be very large, QBlade includes Berg's [1] modification of Gormonts [2] dynamic stall model.

[1] D. E. Berg. Improved double-multiple streamtube model for the darrieus-type vertical-axis wind turbine. 1 1983. URL: https://www.osti.gov/biblio/6279384, doi:.

[2] R. E. Gormont. A mathematical model of unsteady aerodynamics and radial flow for application to helicopter rotors. 1 1973. URL: https://apps.dtic.mil/sti/citations/AD0767240, doi:.

ATEFlap Model

To account for dynamic stall and unsteady aerodynamics the ATEFlap [1] model for 2D airfoil behavior has been integrated to be used with Lifting Line Free Vortex Wake simulations.

Note that the ATEFlap model has been specifically modified to be used with Lifting Line Free Vortex Wake simulations [2], it is currently not advised to use the ATEFlap model in simulations that use the Blade Element Momentum Method aerodynamic model.

The ATEFlap unsteady aerodynamics model consists of mainly two parts; an attached or potential flow model, as proposed by Bergami and Gaunaa [1], and the classical Beddoes-Leishman dynamic stall model with a custom formulation for vortex lift, as presented by Hansen and Gaunaa [3]. The implemented ATEFlap model also accounts for unsteady lift contribution of active trailing edge flap deflections. In its implementation in QBlade the ATEFlap dynamic stall model is only applied within the angle of attack range of -50° to 50°.

Decomposed Polar Data

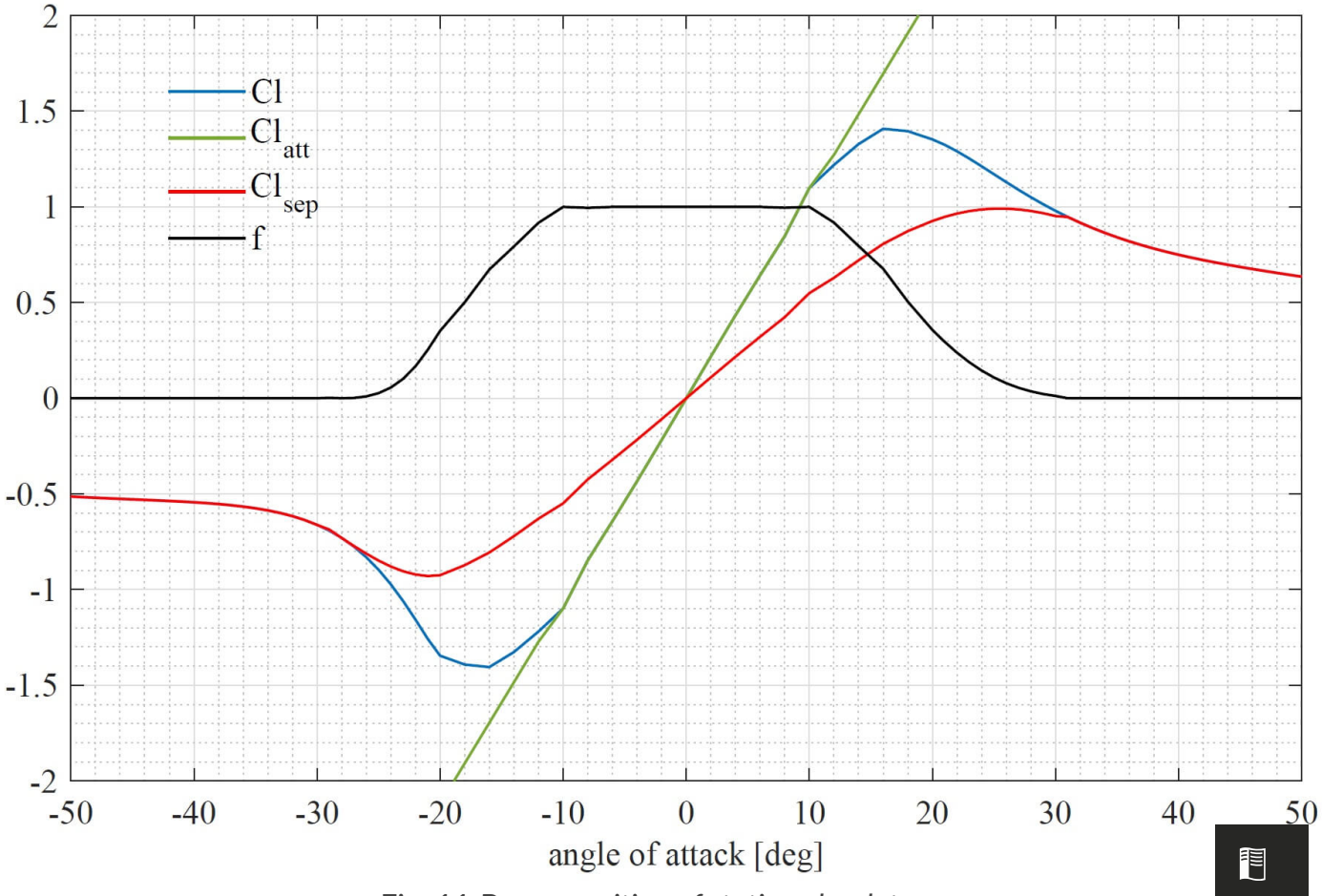


Fig. 11 *Decomposition of static polar data.*

The unsteady aerodynamics model is based on a decomposition of the static, two dimensional lift Cl_{st} into a fully attached Cl_{att} and a fully separated Cl_{sep} contribution. This is shown in Fig. 11. The contributions of the attached and the separated lift to the static lift are described by the following equation:

$$Cl_{st} = fCl_{att} + (1 - f)Cl_{sep},$$

where f represents the separation function.

A module to perform the decomposition of polar data has been integrated with QBlade's airfoil data pre-processor (see Polar Decomposition).

Lift: Attached Flow Contribution

The potential flow model accounts for non-circulatory (added mass) effects, and circulatory lift which includes wake memory effects that play a role in the linear attached flow region. The added mass term models the forces due to the reaction of the fluid to the airfoil motion and the motion of its trailing edge flap:

$$Cl^{nc} = \pi \frac{b_{hc}}{V_\infty} \dot{\alpha}^{str} + \frac{F_{dydxLE}}{\pi} \frac{b_{hc}}{V_\infty} \dot{\beta}.$$

b_{hc} denotes the half chord length of the airfoil, V_∞ the free stream velocity and $\dot{\alpha}_{str}$ the pitch rate due to torsional deformation, F_{dxdyLE} is the deflection shape integral, a geometrical property depending on the airfoils shape (See Gaunaa's work in [4]) and $\dot{\beta}$ the flap deflection rate.

The quasi steady lift component is the steady lift that is generated by the airfoil at the current angle of attack α_{qs} and the current flap deflection β_{qs} obtained from the relative airfoil motion and free-stream velocity, but without the influence of shed wake vorticity:

$$Cl^{qs} = Cl^{att}(\alpha_{qs}, \beta_{qs}).$$

The circulatory component of the lift coefficient is obtained by evaluating the attached lift coefficient at the effective angle of attack α_{eff} with the effective flap angle β_{eff} (see [1] for details).

$$Cl^{circ} = Cl^{att}(\alpha_{eff}, \beta_{eff}).$$

α_{eff} is the angle of attack that arises of the airfoil interaction with its wake. Wake memory effects are caused by the influence of the trailing and shed vorticity in the wake on the quasi-steady angle of attack. Originally the ATEFlap model was formulated for BEM codes, so the downwash of the wake (which also causes the wake memory effect) is modeled with an effective angle of attack tha computed via step responses that are described by exponential indicial response functions [1]. In

QBlade's implementation, the effective angle of attack is directly obtained as the induction from the free vortex wake formulation is already considered in the evaluation of the on-blade velocities.

It should be noted that the quasi steady angle of attack, which does not include the effect of wake vorticity, is not known in the free vortex wake formulation of QBlade. As the quasi steady angle α_{qs} is needed for a later evaluation of the induced drag contribution it is computed by calculating the isolated contribution of the wake vorticity on the angle of attack, denoted as α_{shed}, separately. α_{shed} is obtained by considering the induction of the total shed vorticity in the vicinity of the blade, up to 8 chord lengths away from the trailing edge. As the dynamic stall model is formulated for an isolated two-dimensional airfoil, it is necessary to limit the vortices that are involved in the evaluation of α_{shed} to those in the vicinity of the blade to exclude the significant influence of the total shed vorticity from all previous time steps on the global flow field (this is especially important for VAWT simulations where the shed vorticity has a major contribution to the total induction field around the rotor). α_{shed} is then used to calculate the quasi steady angle of attack from the effective angle of attack.

$$\alpha_{qs} = \alpha_{eff} - \alpha_{shed}.$$

This extra treatment is necessary because the common unsteady aerodynamics models are formulated for BEM codes and use indicial functions. In QBlade, these functions are replaced by the free vortex wake model.

Lift: Separated Flow Contribution

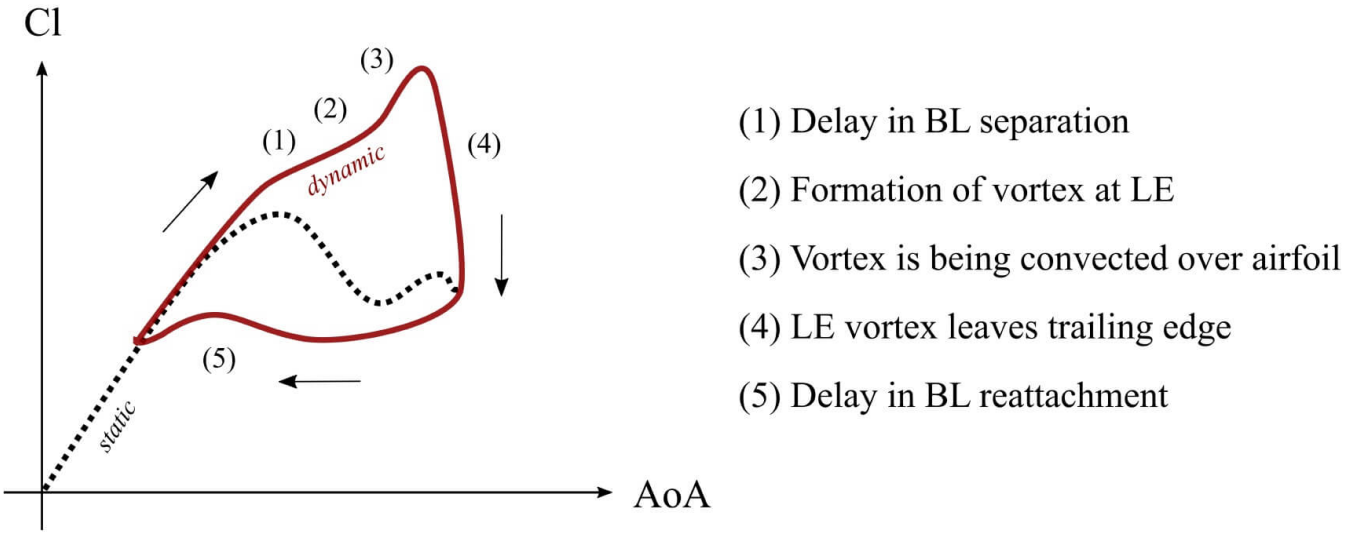


Fig. 12 *The dynamic stall hysteresis loop.*

The implementation of the Beddoes-Leishman dynamic stall model follows the procedure explained in [1]. A schematic representation of the dynamic stall loop is shown in Fig. 12.

In QBlade, the dynamic stall effect is modeled by means of three contributions. The first contribution is the lagged potential lift (leading edge pressure time lag), obtained via a low pass filter function with the pressure time lag constant τ_p:

$$\dot{Cl}^{lag} = -\frac{V_\infty}{b_{hc}}\frac{1}{\tau_p}Cl^{lag} + \frac{V_\infty}{b_{hc}}\frac{1}{\tau_p}Cl^{pot}.$$

In the equation above, Cl^{pot} represents the potential lift from the attached flow contribution:

$$Cl^{pot} = Cl^{circ} + Cl^{nc}.$$

Using the lagged potential lift Cl^{lag} the second contribution can be calculated, namely the intermediate separation function. In this contribution, a separation function is calculated from the static separation function f (obtained via the polar decomposition) in combination with an equivalent angle of attack α_* and flap angle β_* that are obtained with the help of Cl^{lag}:

$$\alpha^* = \frac{Cl^{lag}}{\frac{\partial Cl}{\partial \alpha}} + \alpha_0,$$
$$\beta^* = \frac{Cl^{lag} - Cl_{lag\beta=0}}{\frac{\partial Cl}{\partial \beta}}.$$

With the help of α^* and β^*, the third contribution can be calculated: the dynamic separation function. In this contribution, the dynamic separation function f^{dyn} is calculated by passing the intermediate separation function $f(\alpha^*, \beta^*)$ through a low pass filter with the boundary layer lag constant τ_f:

$$\dot{f}^{dyn} = -\frac{V_\infty}{b_{hc}}\frac{1}{\tau_f}f^{dyn} + \frac{V_\infty}{b_{hc}}\frac{1}{\tau_f}f(\alpha^*, \beta^*),$$

The dynamic circulatory lift $Cl^{circ,dyn}$ is then obtained by multiplying the dynamic separation function f^{dyn} with the fully attached Cl^{att} and the fully separated Cl^{sep} lift contributions that were obtained from the polar decomposition:

$$Cl^{circ,dyn} = Cl^{att}(\alpha_{eff}, \beta_{eff})f^{dyn} + Cl^{sep}(\alpha_{eff}, \beta_{eff})(1 - f^{dyn}).$$

Within the ATEFlap formulation for separated flow a term for modeling the vortex lift is included:

$$C_v = Cl^{circ,dyn}(1 - \frac{(1 + \sqrt{f^{dyn}})^2}{4}).$$

However, it was found, especially when simulating VAWT with large fluctuations in angle of attack, that this term is prone to large fluctuations, often causing unrealistically large values for the total dynamic lift coefficient. Thus, in favor of robustness, it was decided to exclude this term from the calculation of total lift. The total lift, including the attached and separated flow contribution, but excluding the vortex lift, then equals:

$$Cl^{dyn} = Cl^{circ,dyn} + Cl^{nc}.$$

Drag

The dynamic drag is evaluated from four contributions. These are: first, the steady drag at the effective angle of attack and the effective flap angle:

$$Cd^{eff} = Cd(\alpha_{eff}, \beta_{eff}).$$

Second, the drag induced from shed wake vorticity. It is obtained using the quasi steady angle of attack:

$$Cd_{ind} = Cl^{circ,dyn}(\alpha_{qs} - \alpha_{eff}).$$

The third contribution is the induced drag contribution from the flap deflection. It is calculated according to:

$$Cd_{\beta ind} = Cl^{circ,dyn} \cdot \frac{\frac{\partial Cl}{\partial \beta}}{\frac{\partial Cl}{\partial \alpha}} (\beta^{st} - \beta^{eff}) f^{dyn}.$$

The last contribution is the drag change caused through the separation delay:

$$Cd_{find} = (Cd^{eff} - Cd(\alpha_0)) \left[\frac{\left(1 - \sqrt{f^{dyn}(\alpha^*, \beta^*)}\right)^2}{4} - \frac{\left(1 + \sqrt{f^{st}(\alpha^*, \beta^*)}\right)^2}{4} \right].$$

The total drag is then computed as the sum of these contributions:

$$Cd = Cd^{eff} + Cd_{ind} + Cd_{\beta ind} + Cd_{find}.$$

[1] (1, 2, 3, 4, 5) Leonardo Bergami and Mac Gaunaa. ATEFlap Aerodynamic Model. Technical Report, DTU, 2012.

[2] Juliane Wendler, David Marten, George Pechlivanoglou, Christian Oliver Paschereit, ar Christian Navid Nayeri. An Unsteady Aerodynamics Model for Lifting Line Free Vortex Simulations of HAWT and VAWT in QBlade. In *Proceedings of ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition*. 2016.

[3] Morten Hartvig Hansen, Mac Gaunaa, and Helge Aagaard Madsen. A Beddoes-Leishman type dynamic stall model in state-space and indicial formulation. Technical Report, DTU, 2004.

[4] Mac Gaunaa. Unsteady two-dimensional potential flow model for thin variable geometry airfoils. *Wind Energ.*, 13(December 2005):167–192, 2010. URL: https://doi.org/10.1002/we.377.

Secondary Effects

- Tower Influence
- Ground Effect
- Himmelskamp Effect

Tower Influence

A tower shadow model, based on the work of Bak (Moriarty and Hansen[1]) is implemented in QBlade. This model is based on a superposition of the analytical solution for potential flow around a cylinder and a model for the downwind wake behind a cylinder, based on a tower drag coefficient.

For the potential flow around the cylinder contribution, the local velocity components U_{local} and V_{local} are affected by normalized velocity factors:

$$U_{\text{local}} = u \cdot U_\infty,$$
$$V_{\text{local}} = v \cdot U_\infty.$$

In the above equations, U_∞ is the free stream velocity and u and v are given by:

$$u = 1 - \frac{(x+0.1)^2 - y^2}{\left((x+0.1)^2 + y^2\right)^2} + \frac{C_d}{2\pi}\frac{x+0.1}{(x+0.1)^2+y^2},$$
$$v = 2\frac{(x+0.1)y}{\left((x+0.1)^2 + y^2\right)^2} + \frac{C_d}{2\pi}\frac{y}{(x+0.1)^2+y^2}.$$

In these equations, x and y are the upwind and crosswind distances normalized by the tower radius at the relevant height. C_d is the drag coefficient of the the tower.

In addition, the tower produces a wake deficit in the downstream direction. The deficit inside the location of the wake is given by:

$$U_{\text{local}} = (1 - u_{\text{wake}}) \cdot U_\infty,$$

where u_{wake} is given by:

$$u_{\text{wake}} = \frac{C_d}{\sqrt{d}}\cos^2\left(\frac{\pi}{2}\frac{y}{\sqrt{d}}\right) \qquad \text{for} \quad |y| \leq \sqrt{d}.$$

In the above equation, $d = \sqrt{x^2+y^2}$ is the non-dimensional radial distance from the evaluation point to the tower center. The wake width is assumed to be $\sqrt{d}$.

The tower shadow model only affects velocity components that are normal to the tower centerline; the z-component of the velocity, parallel to the tower centerline, remains unaffected. The to[illegible] shadow model is only used when the z-component of the evaluation point is smaller or equa[illegible] tower height. An example of the tower shadow velocity deficit is shown in Fig. 13.

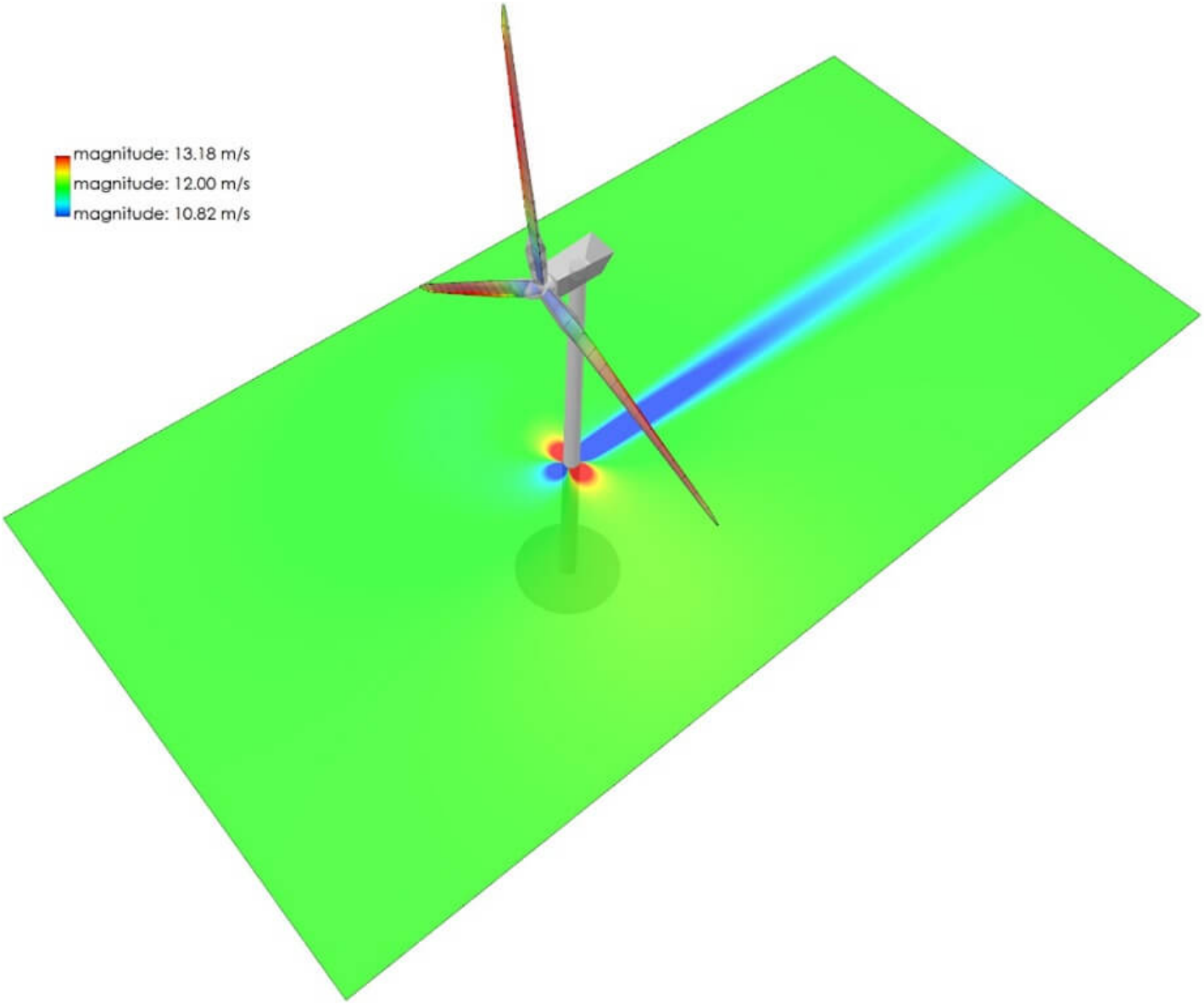


Fig. 13 *Visualization of the tower shadow model; showing velocity magnitude.*

An application of the tower model, including a comparison to CFD simulations and experimental data is found in the work of Klein *et al.*[2].

[1] Moriarty and Hansen. AeroDyn Theory Manual. *Renewable Energy*, 15(January):500–36313, 2005. URL: http://www.nrel.gov/docs/fy05osti/36881.pdf.

[2] Annette Claudia Klein, Sirko Bartholomay, David Marten, Thorsten Lutz, George Pechlivanoglou, Christian Navid Nayeri, Christian Oliver Paschereit, and Ewald Krämer. About the suitability of different numerical methods to reproduce model wind turbine measurements in a wind tunnel with a high blockage ratio. *Wind Energy Science*, 3(1):439–460, 2018. doi:10.5194/wes-3-439-2018.

Ground Effect

Ground effects can be modelled with the Lifting Line Free Vortex Wake method by mirroring all vortex elements, bound and free, at the ground plane (Leishman[1]). A mirror image (see Fig. 14) of all bound and free vortices is created at every time step using the ground as a symmetry plane. Such a treatment doubles the number of times that the Biot-Savart equation is calculated and thereby doubles the computational time needed for the evaluation of the convection step.

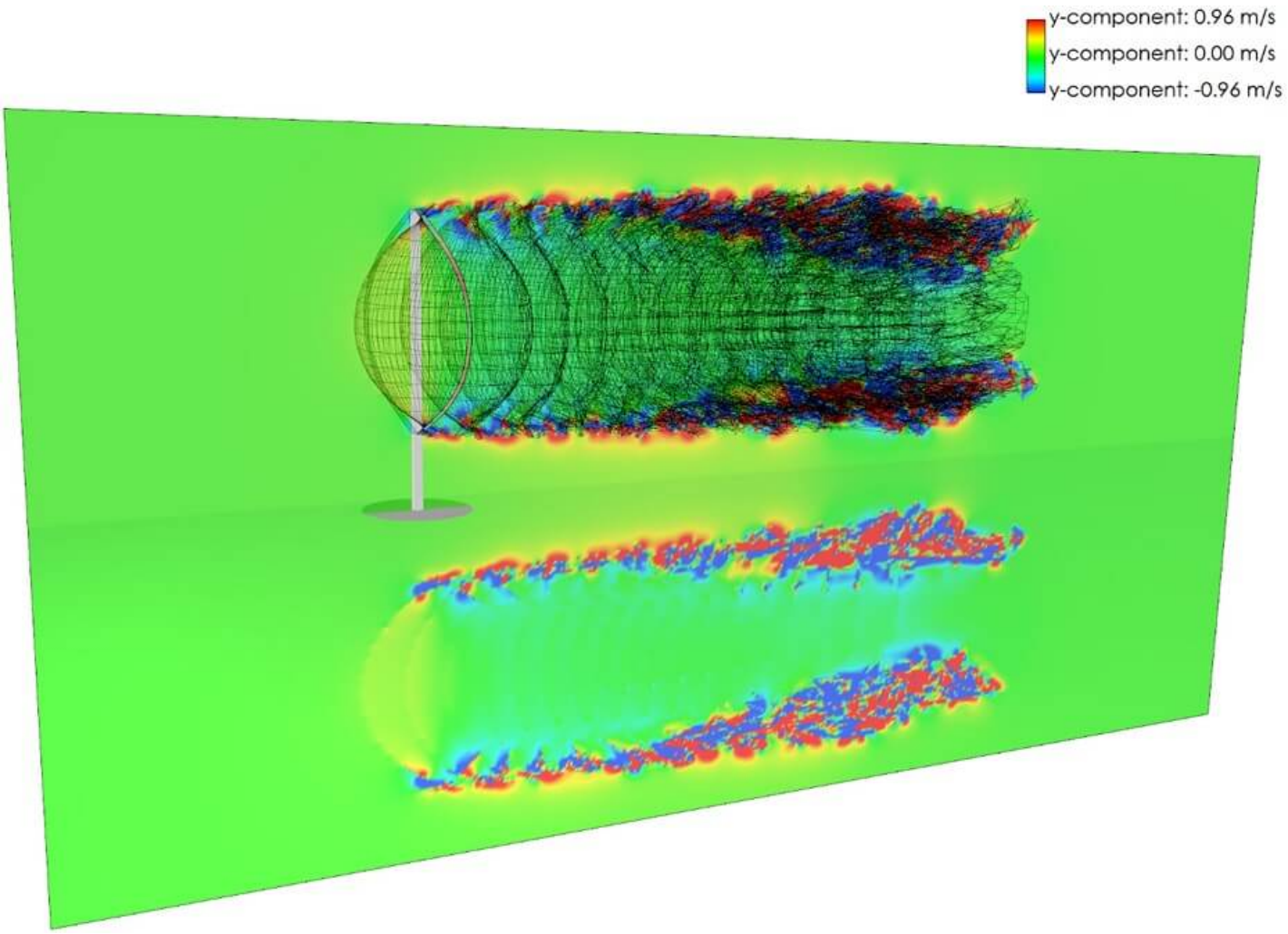


Fig. 14 *Modeling of ground effect through mirroring of the wake.*

[1] J. G. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge University Press, 2000.

Himmelskamp Effect

The maximum lift coefficient of profiles on a rotating rotor blade is significantly higher than the maximum lift coefficient of the same profile measured on a stationary blade. The centrifugal force accelerates the boundary layer radially. This results in a thinner boundary layer in which stall is delayed. At the same time, air flowing radially, in a rotating reference system, generates a Coriolis force opposite to the rotational direction of the rotor. This force is opposing the rise in pressure of the profile's suction side and delays the stall. This effect, called stall delay or Himmelskamp effect, can be taken into account by modifying the two dimensional polar data. For the affected profiles the stalled region will shift to higher angles of attack. With a viscous-inviscid interaction method, Snel investigated the flow around a rotating rotor blade and developed a semi-empirical formula to correct 2D polar data (Snel *et al.*[1]). According to Snel, only the lift but not the drag coefficient, needs to be modified. The Himmelskamp effect, can be modeled in QBlade using Snel's correction:

$$Cl_{3D} = Cl_{2D} + \frac{3.1\lambda^2}{1+\lambda^2} g \left(\frac{c}{r}\right)^2 \left(\frac{\partial Cl}{\partial \alpha}(\alpha - \alpha_0) - Cl_{2D}\right).$$

Where Cl_{2D} is the 2D lift polar data, λ is the local tip speed ratio, α (α_0) is the angle of attack (at 0 deg) and c is the chordlength. g is a blending factor: $g = 1$ for $0 < \alpha < 30$;
$g = 0.5(1 + \cos(6\alpha - 180))$ for $30 < \alpha < 60$ and $g = 0$ for $60 < \alpha < 360$. If this correction is activated for a simulation it is applied on the unmodified, tabulated 2D airfoil data at every timestep.

[1] H. Snel, R. Houwink, and W. J. Piers. Sectional Prediction of 3D Effects for Separated Flow on Rotating Blades. 1993.

Linear Potential Flow Theory

QBlade is capable of calculating hydrodynamic forces on submerged elements due to floater motion and waves by making use of potential flow theory. This allows the full interaction between the three-dimensional floater geometry and the wave field to be accounted for.

Potential flow and boundary conditions

Here it is assumed that the flow field is irrotational. This allows the flow field to be described in terms of a scalar potential ϕ, which satisfies Laplace's differential equation $\nabla^2\phi = 0$. The velocity field $\vec{v}$ can be expressed as the gradient of this potential:

$$\vec{v} = \nabla\phi = \frac{\partial\phi}{\partial x}\vec{e}_x + \frac{\partial\phi}{\partial y}\vec{e}_y + \frac{\partial\phi}{\partial z}\vec{e}_z \, .$$

Three boundary conditions are require to specify fully the problem [1]. The first enforces continuity at the free surface. This is linearised (hence *linear* potential flow) to give:

$$\frac{\partial\phi}{\partial z} - \frac{\omega^2}{g}\phi = 0 \quad \text{on} \quad z = 0 \, ,$$

where g is the acceleration due to gravity and ω is the discrete frequency being analysed. The second enforces the boundary condition on the sea bottom:

$$\begin{aligned} \nabla\phi \to 0 \ \text{as} \ z \to -\infty \qquad & \text{Infinite depth} \\ \frac{\partial\phi}{\partial z} = 0 \ \text{on} \ z = -h \qquad & \text{Finite depth } h \, . \end{aligned}$$

Finally, the Sommerfeld condition states that wave energy associated with disturbance due to the body is radiated in all directions. A few assumptions are outlined here for the following discussion:

- The floater undergoes negligibly small motions away from the equilibrium position.
- Solutions are assumed to be harmonic $\hat{\phi} = \phi\, e^{i\omega t}$.
- The floater is submerged in a fluid with density $\rho \, \mathrm{kg\,m^{-3}}$.

Equations of Motion

The motion of a generic floating body in dimension j: $x_j(t)$ can be modelled with the equat to Cummins[2]:

$$(M_{ij} + A_{\infty ij})\ddot{x}_j(t) + \int_{t-\infty} K_{ij}(t-\tau)\dot{x}_j(\tau)\,d\tau + C_{ij}x_j(t) = F_{wj}(t) - F_{ej}(x, \dot{x}, t)\ .$$

In the above expression the indices i and j represent the degree of freedom of motion and the acting force, respectively. In this equation:

- M_{ij} represents the inertia of the floater
- $A_{\infty ij}$ is the added mass matrix (see Radiation Forces below)
- K_{ij} is the radiation damping matrix (see Radiation Forces below)
- C_{ij} is the hydrostatic stiffness matrix (see Hydrostatic Forces)
- F_{wj} are the forces due to waves (see Excitation forces below)
- F_{mj} are external forces due to moorings

These terms shall be described in the following sections.

Radiation Forces

The motion of the floater in an undisturbed field generates waves which radiate away from the body. Newton's third law dictates that the force required to set this disturbance in motion gives rise to an equal and opposite force on the body. This acts in the form of a pressure disturbance. An example of this is shown in the following figure for a motion in the surge direction.

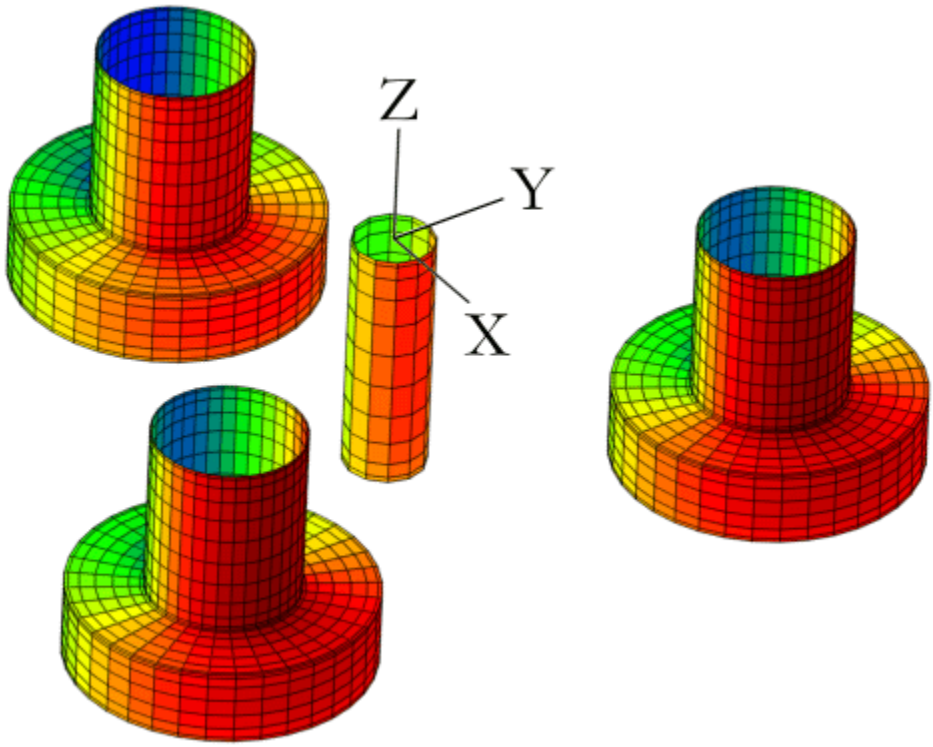


Fig. 15 *Disturbance potential ϕ_r of a triple-spar geometry for a surge motion.*

The disturbance potential ϕ_r due to this motion is sought. The body is taken as being impermeable which implies that the local velocity (gradient of the potential) must be equivalent to the local body

motion:

$$\frac{\partial \phi_r}{\partial n} = \nabla \phi \cdot \vec{n} = \vec{u} \, ,$$

where $\vec{n}$ is the normal vector at the surface. The solution for ϕ_r on the geometry represents the hydrodynamic pressure acting. This is integrated over the floater surface S_b to give the total forcing due to motion (note that this is complex as a harmonic solution has been assumed):

$$A_{ij} - \frac{i}{\omega} B_{ij} = \rho \iint_{S_b} n_i \, \phi_{r,j} \, dS \, ,$$

where A_{ij} and B_{ij} are referred to as the *added mass* and *radiation damping* matrices. It is important to note that the terms above are calculated in a frequency domain analysis. The added mass matrix can be taken directly from the frequency domain analysis as:

$$A_{\infty ij} = \lim_{\omega \to \infty} A_{ij}(\omega) \, ,$$

where it is important that a sufficiently high analysis frequency ω is taken to ensure that $A_{ij}(\omega)$ has converged. An impulsive motion in any direction generates a force which is time-varying, this is accounted for with the time convolution in the equations of motion above. The time convolution kernel is referred to as the *impulse response function*, or IRF and is calculated as:

$$K_{ij}(t) = \frac{2}{\pi} \int_{\infty 0} \omega A_{ij}(\omega) \sin \omega t \, d\omega = \frac{2}{\pi} \int_{\infty 0} B_{ij}(\omega) \cos \omega t \, d\omega \, .$$

In practise the second form is used due to its easier numerical integration.

The arrays for $A_{ij}(\omega)$ and $B_{ij}(\omega)$ can be imported into QBlade in NEMOH, WAMIT, or BEMUse formats. This integration is carried out numerically with a frequency step size Δ_ω.

$$K_{ij}(t) = \frac{2}{\pi} \int_{\infty 0} B_{ij}(\omega) \cos \omega t \, d\omega \approx \frac{2}{\pi} \sum_{n=1}^{n=\omega_{max}} \Delta_\omega B_{ij}(n\Delta_\omega) \cos tn\Delta_\omega \, .$$

The decay of K_{ij} implies that the time convolution can be truncated to a finite time T. The radiation force F^r can be calculated by carrying out a time convolution numerically with timestep Δ_t:

$$F_{rj}(t) = \int_{tt-T} K_{ij}(t-\tau) \dot{x}_j(\tau) \, d\tau \approx \sum_{i=1}^{i=T/\Delta_t} \Delta_t K_{ij}(i\Delta_t) \dot{x}_j(t - i\Delta_t) \, .$$

Excitation forces

The boundary condition on the surface of the floater causes incoming waves to be reflected away. As with the radiation forces, this gives rise to a disturbance potential ϕ_d and a corresponding force X_j

which acts on the floater. The Haskind relations allows these forces to be expressed in terms of the radiation potential:

$$X_j = -i\omega\rho \iint_{S_b} \left(n_i \phi_0 - \phi_{r,j} \frac{\partial \phi_0}{\partial n} \right) dS \, ,$$

where ϕ_0 is the potential of the incoming wave. An IRF for this is calculated as:

$$H_{ij}(t) = \frac{1}{2\pi} \int_{\infty - \infty} X_j(\omega) e^{i\omega t} \, d\omega \, .$$

This is numerically integrated with a frequency step size Δ_ω. The lower limit of the integral indicates that the excitation IRF is non-causal, which means the chosen input is not the cause for the output. In this context, the non-causality may be explained by the fact that the incident wave hits the body and exerts a wave force before the wave reaches the chosen reference point for the body (usually located at the geometrical center), see Falnes[3]:.

As with the radiation forces, the time-domain excitation forces F^e are calculated with a time convolution with the IRF given above:

$$F_{ej}(t) = \int_{\infty - \infty} H_{ij}(\tau) \zeta(x_0, y_0, t - \tau) \, d\tau \, .$$

where $\zeta(x_0, y_0, t)$ is the wave elevation at the reference position (x_0, y_0) during time t.

Since the non-causality of the excitation IRF means $H_{ij}(t) \neq 0$ for $(t) < 0$, future wave information is required before the waves reach the neutral reference position of the floater, see Falnes[3]. This integral is again calculated numerically over a truncated time period T with the time step dT.

[1] J.N. Newman. *Marine Hydrodynamics*. MIT Press, 2018. ISBN 9780262534826.

[2] W.E. Cummins. The impulse response function and ship motions. *Shiffstechnik*, pages 101–109, 1961.

[3] (1, 2) J. Falnes. On non-causal impulse response functions related to propagating water waves. *Applied Ocean Research*, 17(6):379–389, 1995. doi:https://doi.org/10.1016/S0141-1187(96)00007-7.

Morison Equation

QBlade also offers the possibility to model hydrodynamic loads on slender cylindrical bodies with a full Morison equation approach. This is especially useful for calculating distributed hydrodynamic loads on the members and allow substructure flexibility. QBlade considers two types of Morison forces on cylindrical elements: normal forces that act at the center of the element and axial forces that act at the ends of the element.

Normal Morison Force on a Cylindrical Element

The hydrodynamic normal forces on a cylindrical element are given by Faltinsen[1]

$$F_{nM} = \rho\pi\left(\frac{D}{2} + R_{MG}\right)^2 L\left((C_{na} + C_{np})\dot{u}^n - C_{na}\ddot{X}^n\right) + \frac{1}{2}\rho\,(D + R_{MG})LC_{nd}\left(u^n - \dot{X}^n\right)\left|u^n - \dot{X}^n\right|.$$

In this equation,

- F_{nM} is the normal Morison force,
- ρ is the water density,
- D and L are the diameter and length of the cylinder element, respectively,
- R_{MG} is the marine growth thickness,
- C_{na} is the normal added mass coefficient,
- C_{np} is the normal dynamic pressure coefficient,
- C_{nd} is the normal drag coefficient,
- u^n is the resulting normal flow velocity,
- $\dot{X}^n$ is the resulting normal velocity of the center of the cylinder element.

The structural model of QBlade naturally handles the vector direction of the local normal force components depending on the element location and orientation (see Multi Body Beam Formulation). The normal force is assumed to be constant over the submerged part of the cylinder and is integrated along the submerged length. For this evaluation the local instantaneous wave elevation is obtained at every timestep to calculate the current submerged fraction of the cylinder. It is recommended to subdivide cylindrical elements that are close to the water surface into smaller sub-elements to increase the model accuracy (see Modeling Considerations for Morison Elements)

Axial Morison Force on a Cylindrical Element

Axial forces can also be applied to the ends of a cylindrical element. In QBlade, the hydrodynamic forces are calculated using following equation:

$$F_{axM} = \rho\frac{2\pi}{3}\left(\frac{D}{2}+R_{MG}\right)^3 C_{axa}(\dot{u}^{ax}-\ddot{X}^{ax}) + C_{axp}p_{axdyn}\pi\left(\frac{D}{2}+R_{MG}\right)^2 + \frac{1}{2}\rho\pi(D+R_{MG})^2 C_{axd}\left(u^{ax}-\dot{X}^{ax}\right)\left|u^{ax}-\dot{X}^{ax}\right|.$$

In this equation,

- F_{axM} is the axial Morison force,
- ρ is the water density,
- D and L are the diameter and length of the cylinder element respectively,
- R_{MG} is the marine growth thickness,
- C_{axa} is the axial added mass coefficient,
- C_{axp} is the axial dynamic pressure coefficient,
- C_{axd} is the axial drag coefficient, along the ith axis,
- p_{axdyn} is the axial dynamic pressure,
- u^{ax} is the resulting axial flow velocity,
- $\dot{X}^{ax}$ is the resulting axial velocity of the center of the cylinder end.

As with F_{nM}, the structural model in QBlade always applies the axial loads in the local frame of reference considering the orientation of the cylinder.

The axial force calculation is only performed at the ends of cylindrical elements that are not obstructed, or overlapped, by the ends of other cylindrical elements which share the same node. Example: If a 'thinner' cylindrical element D_{thin} is connected to a 'thicker' element D_{thick} via a common node the axial force is evaluated at the end of the 'thick' element only.

In such a case the effective volume at the 'thicker' element would be calculated as:

$$V = \frac{2\pi}{3}\left(\left(\frac{D_{thick}}{2}+R_{MG}\right)^3 - \left(\frac{D_{thin}}{2}+R_{MG}\right)^3\right),$$

and the effective area would be calculated as:

$$A = \pi\left(\left(\frac{D_{thick}}{2}+R_{MG}\right)^2 - \left(\frac{D_{thin}}{2}+R_{MG}\right)^2\right).$$

McCamy-Fuchs Correction

For cylindrical elements that have a diameter to wavelength ratio larger than 0.2, the diffraction forces become relevant and the pure Morison equation is no longer accurate, see Faltinsen[1]. The diffraction effects can be accounted for with the Morison equation formulation by introducing the MacCamy-Fuchs Correction (MCFC). The original MCFC is formulated in terms of the normal force per unit length on the cylinder. It can be recast so that the inertia part of the Morison equation is affected, see the IEC standard [2] for reference. Following an approach presented in the USFOS reality engineering guide [3], QBlade changes the local water particle acceleration's magnitude and phase to affect the Morison element. The original modifications include the calculation of Bessel functions. To speed up calculations, an more efficiently calculable approximation proposed in USFOS Reality Engineering[3] is used for the modification of the particle acceleration's magnitude and phase. The equivalent particle acceleration amplitude is given by:

$$\dot{u}_{eq} = \dot{u} \cdot \min\left(\frac{1.05 \tanh\left(2\pi \frac{d}{\lambda}\right)}{\left(\mathrm{abs}\left(\pi \frac{D}{\lambda} - 0.2\right)^{2.2} + 1\right)^{0.85}}, 1 \right).$$

In this equation, $\dot{u}$ is the water particle acceleration amplitude, d is the water depth, D is the diameter of the element and λ is the wavelength. The phase shift of the acceleration is given by:

$$\alpha^{\mathrm{phase}} = \frac{\pi}{180}\left(-\frac{450}{8}\left(\pi \frac{D}{\lambda} - 2\right) - \frac{75}{\left(\pi \frac{D}{\lambda} + 0.5\right)^2} \right).$$

Since it affects the incoming water particle acceleration and phase, this implementation of the MCFC can be easily extended to irregular wave spectra. In this case, the correction is done on each wave train that makes up the wave spectrum and avoids using frequency dependent added mass coefficients in the Morison equation formulation.

Modeling Considerations for Morison Elements

In QBlade, each cylindrical element can be divided into sub-elements. Morison forces acting on these elements are evaluated and applied at each time step. Setting the hydrodynamic coefficients to 0 effectively disables their contribution in the calculation of the Morison forces. This way, it possible to include for example the hydrodynamic drag only. To determine if a sub-element is partially or fully submerged, the wave elevation is required. Wave kinematics are also required to calculate u and $\dot{u}$ in the equations above. There are three possibilities in QBlade to do this. These options are presented in Fig. 16.

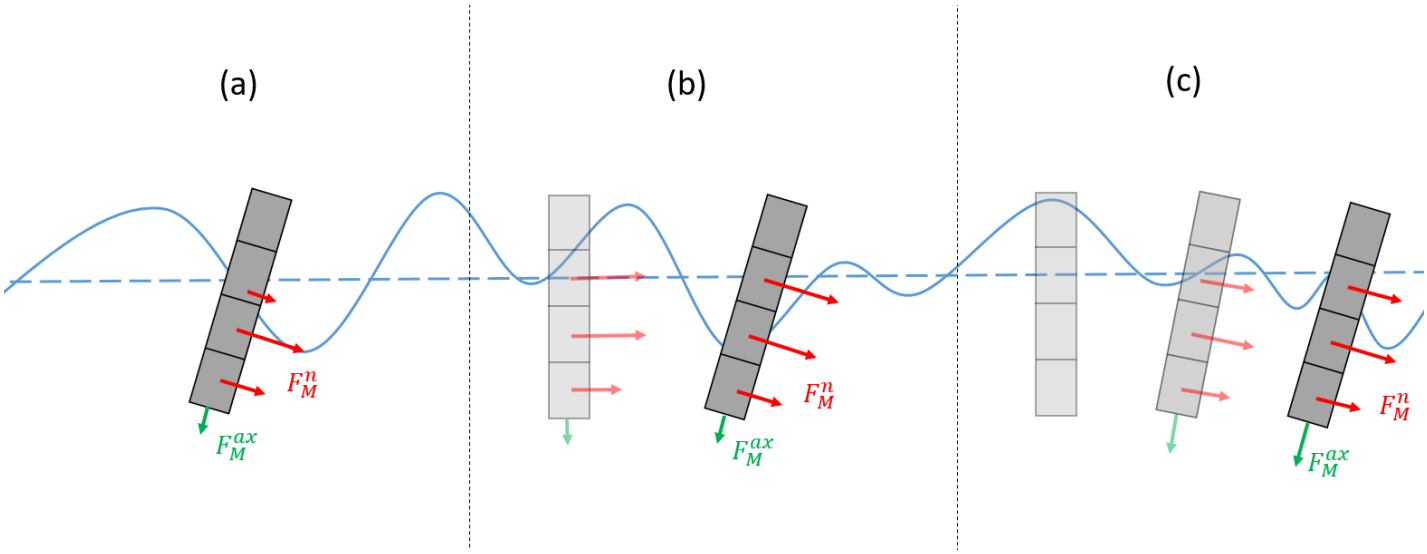


Fig. 16 *Options in QBlade to consider the wave elevation and kinematics. (a) local instantaneous values; (b) values at the initial undisplaced position; (c) values at a low-passed position of the element.*

The first option shown in Fig. 16 (a) is the wave kinematics and elevation at the local instantaneous position of the cylinder. In this example, the cylinder has been divided into four sub-elements. The lower two are fully submerged and one sub-element is partially submerged. The second option in Fig. 16 (b) is using the wave elevation and kinematics at the initial position of the sub-element. This option allows a coherent theoretical assumption of small oscillations around a steady positon when Morison forces are used in conjunction with a linear potential flow model (see Linear Potential Flow Theory). The third option uses the wave elevation and kinematics at a low-passed position of the sub-element (Fig. 16 (c)). This allows for an assumption of small oscillations around a steady state for an element that as drifted from its initial position due to an acting force such as and aerodynamic thrust or sea current hydrodynamic force.

[1] (1, 2) O. M. Faltinsen. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press, 1993.

[2] IEC61400-3-1 Standard. IEC 61400-3-1: Wind Energy Generation Systems - Part 3-1: Design Requirements for fixed offshore wind turbines. Standard, International Electrotechnical Commission, Geneva, Switzerland, 2019.

[3] (1, 2) USFOS Reality Engineering. USFOS Theory, Description of use and Verification Manual. https://www.usfos.no/manuals/usfos/theory/documents/Usfos_Hydrodynamics.pdf, 2021. [Online; accessed 2021-11-21].

Hydrostatic Forces

Of the numerous forces acting on the substructure of an offshore wind turbine, amongst the most important are those due to buoyancy. It shall be assumed that the displaced fluid has density ρ [kgm $^{-3}$] and is within a gravity field with acceleration g [ms $^{-2}$].

Archmides' Principle

This states that the magnitude of the buoyancy force $\vec{F}_b$ is equivalent to the weight of the fluid volume that the body displaces V_d. The total force can be calculated as:

$$\vec{F}_b = \iint_A p(z)\vec{n}\, dA = \iiint_V div(p)\, dV = \rho g V_d \,,$$

where $p(z)$ is the pressure (normal stress) as a function of depth, $\vec{n}$ is the normal to the surface and A and V are the submerged surface area and displaced volume of the body, respectively. The force acts at the centre of buoyancy, which is the centre of gravity of V_d. This is shown for a simple geometry in Fig. 17.

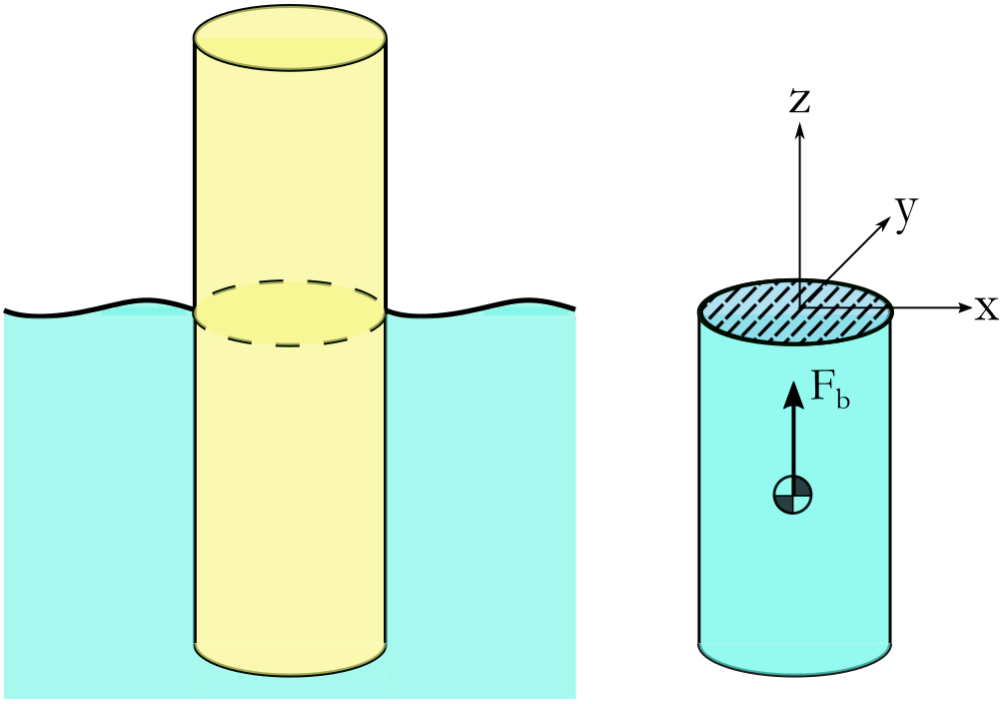


Fig. 17 *Floating body (left) and the corresponding displaced volume (right).*

The motion and orientation of the body and the local surface elevation causes the displaced volume to change with time, this correspondingly changes the magnitude and direction of $\vec{F}_b$. Depen n the spatial distribution of V_d, this can also induce a buoyancy moment on the body. There ar main approaches for calculating these loads.

Discrete Surface Buoyancy Calculation

In this approach the submerged surface A is discretised non-overlapping surface elements and the pressure acting over the surface is numerically integrated in order to arrive at the total buoyancy force. The hydrostatic equation is used to determine the pressure as a function of depth:

$$p(z) = p_0 - \rho g z,$$

where p_0 is the atmospheric pressure at the free surface. As pressure is a normal stress, this force always acts in the direction normal to the surface $\vec{n}$.

Discrete Volume Buoyancy Calculation

In this approach the displaced volume V_d is discretised into non-overlapping volume elements. The buoyancy force acting at the centre of buoyancy of each element are then integrated to get $\vec{F}_b$. This is practically equivalent to the discrete surface approach, however integration is carried out over volume elements.

In QBlade the volume buoyancy calculation approach is used. In general all offshore substructures in QBlade are composed of cylindrical elements. There are two methods implemented how the buoancy of a cylinder is calculated in QBlade.

In the simple approach the intersection between the cylinder centerline and the local sea elevation is evaluated. Based on this intersection and the cylinder endpoints it is estimated wether the cylinder is not, fully or only partially submerged. The partially submerged part of the cylinder is estimated from the submerged part of the centerline. The submerged height is treated as constant around the cylinder circumference, regardless of its orientation. The buoancy force then acts on the midpoint of the submerged part of the centerline.

In the advanced buoancy approach in QBlade each cylinder is discretized into multiple prismatic elements (the element number is a user input, 100 is the suggested default value). For each of these elements the centerline approximation is carried out and then all forces are summed up and an equivalent force acting point is evaluated.

Hydrostatic Stiffness Matrix

It shall be assumed that the surface elevation is everywhere approximately constant. This can be valid if the wavelength of the sea state is much larger than the dimension of the body. Expressing the motion of the body away from the equilibrium position as a simple translation + rotation: $d\vec{x} = [\delta_x, \delta_y, \delta_z, \theta_x, \theta_y, \theta_z]$ (surge, sway, heave, roll, pitch, yaw), the buoyancy force can be expressed as:

$$\vec{F}_b = \vec{F}_c - Cd\vec{x} \quad \text{where: } \vec{F}_c = \begin{bmatrix} 0 \\ 0 \\ \rho g V_d \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{ and: } C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{33} & C_{34} & C_{35} & 0 \\ 0 & 0 & C_{43} & C_{44} & C_{45} & C_{46} \\ 0 & 0 & C_{53} & C_{54} & C_{55} & C_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$\vec{F}_c$ is the vector representing the buoyancy force acting on the body in equilibrium. The matrix C is referred to as the hydrostatic stiffness matrix and is composed of a set of integrals over the waterplane area S, shown as the hatched area in Fig. 17:

$$\begin{aligned} C_{33} &= \rho g \iint_S dS & C_{34} &= C_{43} = \rho g \iint_S y\, dS \\ C_{44} &= \rho g \iint_S y^2\, dS + \rho g V_d z_b & C_{35} &= C_{53} = -\rho g \iint_S x\, dS \\ C_{55} &= \rho g \iint_S x^2\, dS + \rho g V_d z_b & C_{45} &= C_{54} = -\rho g \iint_S xy\, dS \\ C_{46} &= -\rho g V_d x_b & C_{56} &= -\rho g V_d y_b \end{aligned},$$

where $\vec{B} = [x_b, y_b, z_b]$ is the deflected position of the centre of buoyancy. In practise many of these terms are zero due to symmetries of the body. It can be seen that the hydrostatic loads assume that the waterplane area does not significantly change. It is for this reason that the above expression is only valid for small rotations and translations. When the force acting on the body due to gravity is taken into account, it can be seen that the V_d terms in the C_{ij} represent restoring buoyancy moments which act to stabilise the body position. For a detailed overview of stability of floating bodies, the reader is referred to the book by Newman [1].

[1] J.N. Newman. *Marine Hydrodynamics*. MIT Press, 2018. ISBN 9780262534826.

Multi Body Beam Formulation

The structural model in QBlade is based on the FEA module of the open source multi-physics engine Project Chrono Tasora *et al.*[1]. Project Chrono is based on a platform independent design Projectchrono.org[2], which is developed in the C++ language as an object-oriented library.

For the integration into QBlade, the Chrono::Engine module is employed. Chrono::Engine is the core module of Project::Chrono, it contains functionality for setting up and solving physical systems containing Newtonian dynamics and finite elements. The SparseLU solver of the EIGEN C++ template library Tuxfamily[3] is used as a solver for the finite element problem. A dynamic link library, containing the Chrono module, has been compiled from Project Chrono's GIT repository. The relevant header files of Project Chrono and the EIGEN library are included, and the Chrono DLL is linked to QBlade's source code. This enables the definition of the physical system and the finite elements and grants access to the solver to perform time domain simulation of structural dynamics inside QBlade.

Element and Multi-Body Formulation

The structural turbine model in the QBlade-Chrono coupling consists of Euler Bernoulli beams in a co-rotational formulation Negrut[4]. In the co-rotational formulation (see Fig. 18), a floating coordinate system is attached to each deformable beam element. The overall motion of a beam element is then the addition of the rigid body (translation and rotation) undergone by the floating coordinate system and a smaller strain deformation, expressed in the floating frame of reference. The global tangent stiffness matrix in Project Chrono's implementation is formulated in a way to include terms for geometric stiffness.

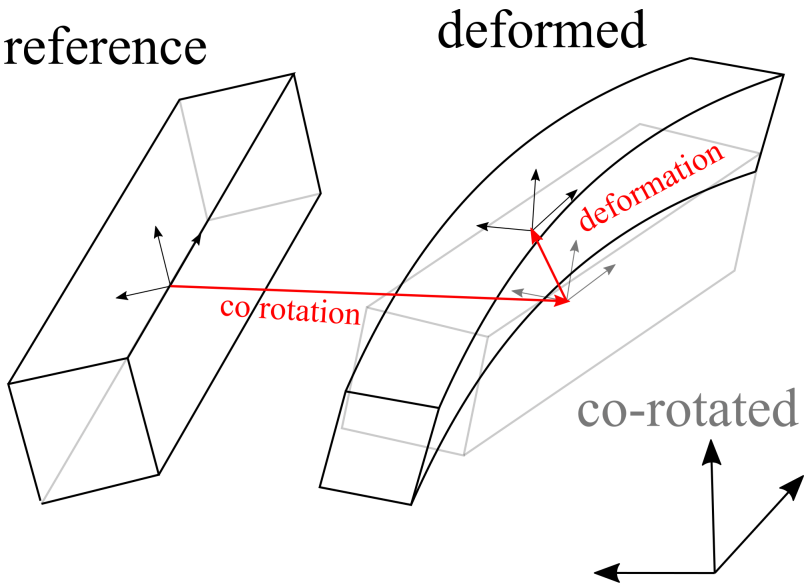


Fig. 18 *Visualization of the co-rotational beam approach.*

In QBlade's implementation of the structural model, the complete turbine structure is divided up into body objects. A body object contains an array for its structural nodes, an array for its structural beam elements, a unique identifier and several functions to access forces, torques, positions, velocities, accelerations and deflections. For a common HAWT, one body is created for each blade and one body for the tower.

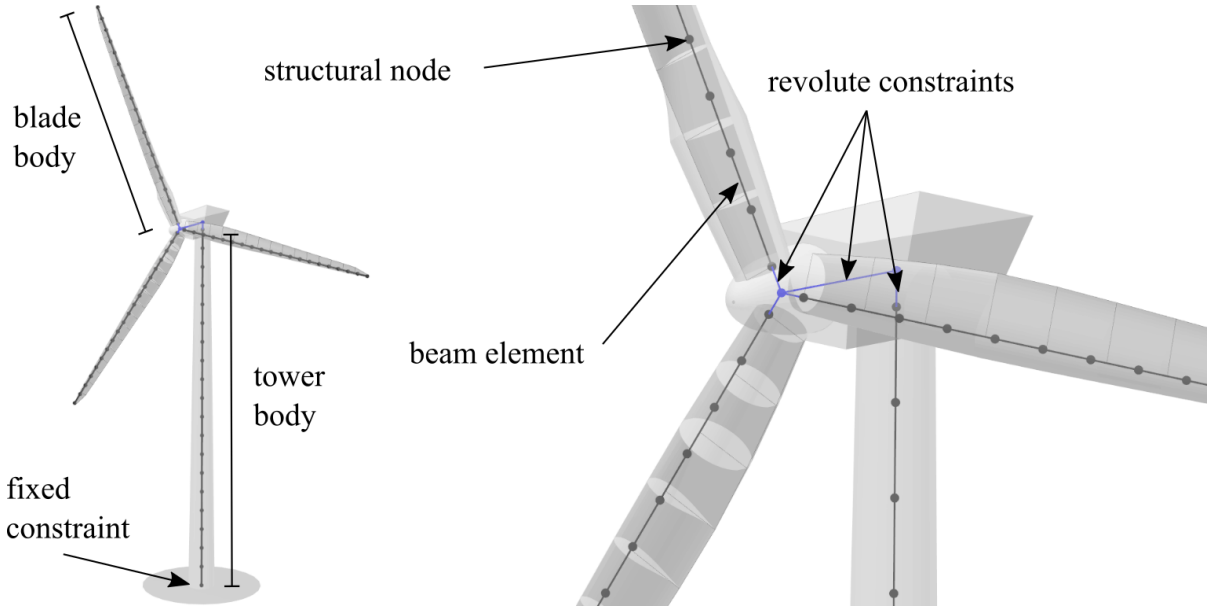


Fig. 19 *Visualization of the co-rotational beam approach.*

After the bodies have been created, they are assembled using joints or constraints (see Fig. 19). The tower is fixed to the ground by constraining all six DoF of the bottom tower node. A spring and damper may be defined at the ground to include foundation and soil dynamics. Using a revolute constraint, a free yaw node is connected to the tower top. Another revolute constraint then connects the hub node to the yaw node. Lastly, the blades are fixed around the hub node with revolute constraints, allowing them to rotate around the pitch axis. After the assembly of the bodies is completed, actuators are added to the revolute constraints. These actuators are used to yaw or pitch the turbine, based on controller signals and to model the generator. Actuators are implemented as engine type constraints. At these engine type constraints, either a rotational angle, a rotational speed or a torque can be applied. This functionality is used to prescribe pitch angles at the pitch constraints, yaw angles at the yaw constraint and the generator torque at the shaft constraint. Furthermore, if no controller is used within a simulation, a constant rotational speed is prescribed for the main shaft to operate the turbine at a constant rotational speed.

Time Integrators and Solver for the Structural Dynamics Simulation

Various factors influence the overall contribution of the structural model to the total computational cost of an aeroelastic simulation. The size of the problem matrix is proportional to the numb degrees of freedom that the system contains. Each main component (blades, struts, tower) o assembled turbine can be discretized with an arbitrary number of structural nodes, where each node

adds 6 degrees of freedom to the system matrix. Clearly, the total contribution of the structural model evaluations to the overall computational cost scales with the time step size of the structural evaluations. Due to the loose coupling method that is being employed in QBlade, the time step size can be set independently of that of the aerodynamic calculations.

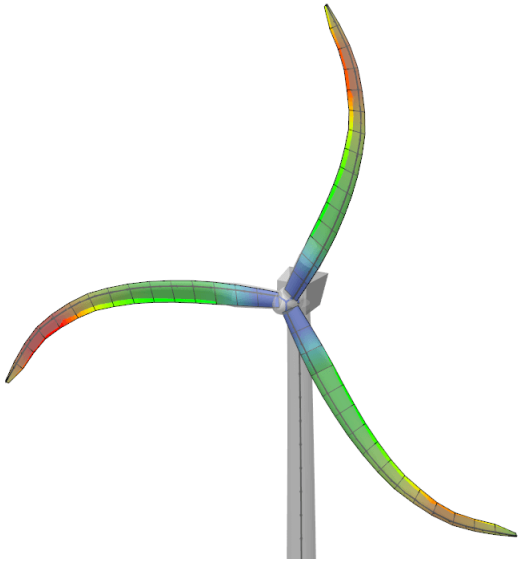


Fig. 20 *Large blade deformations caused by inertial forces during rotor ramp-up.*

In the Chrono library, the multi-body FEA problem is formulated as a *Differential Variational Inequality* (DVI) problem. At each time step of the structural simulation, the DVI problem is solved using the EIGEN SparseLU solver, which is included in the EIGEN C++ template library Tuxfamily[3]. The structural simulation is then advanced using a time integrator of choice. Several different time integrators (Tasora[5]) are available in Chrono. However, only the iterative HHT (*Hilber-Hughes-Taylor formulation*) has proven its usability within the current integration of Chrono in QBlade. While other, non-iterative, integrators suffer from constraint drifts or require very small timesteps to yield reasonable results, the HHT integrator shows good performance for structural time steps in the range of up to 5 degree azimuthal rotor increments.

[1] Alessandro Tasora, Hammad Mazhar Radu Serban2, Arman Pazouki, Daniel Melanz, Jonathan Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. Chrono: An Open Source Multi-physics Dynamics Engine. *Tasora, A., Serban, R., Mazhar, H., Pazouki, A., Melanz, D., Fleischmann, J., ... Negrut, D. (2016). Chrono: An Open Source Multi-physics Dynamics Engine. High Performance Computing in Science and Engineering, 19–49. doi:10.1007/978-3-319-40361-8_2*, 2016. doi:10.1007/978-3-319-40361-8_2.

[2] Projectchrono.org. Project Chrono - An Open-Source Physics Engine. 2019. URL: https://projectchrono.org/about/ (visited on 2019-07-12).

[3] (*1*, *2*) Tuxfamily. EIGEN - a C++ template library for linear algebry: matrices, vectors, numerical solvers and related algorithms. URL: http://eigen.tuxfamily.org (visited on 2019-07-12).

[4] Dan Negrut. Co-rotational Formulation in Chrono Antonio Recuero Chrono. pages 1–10, 2016. doi:.

[5] Alessandro Tasora. Time integration in Chrono::Engine. pages 1–44, 2017. doi:.

Aero-Elastic Coupling

A loose coupling approach is employed for the aero-elastic co-simulation in QBlade. Fig. 21 shows the flowchart for one complete aeroelastic time step:

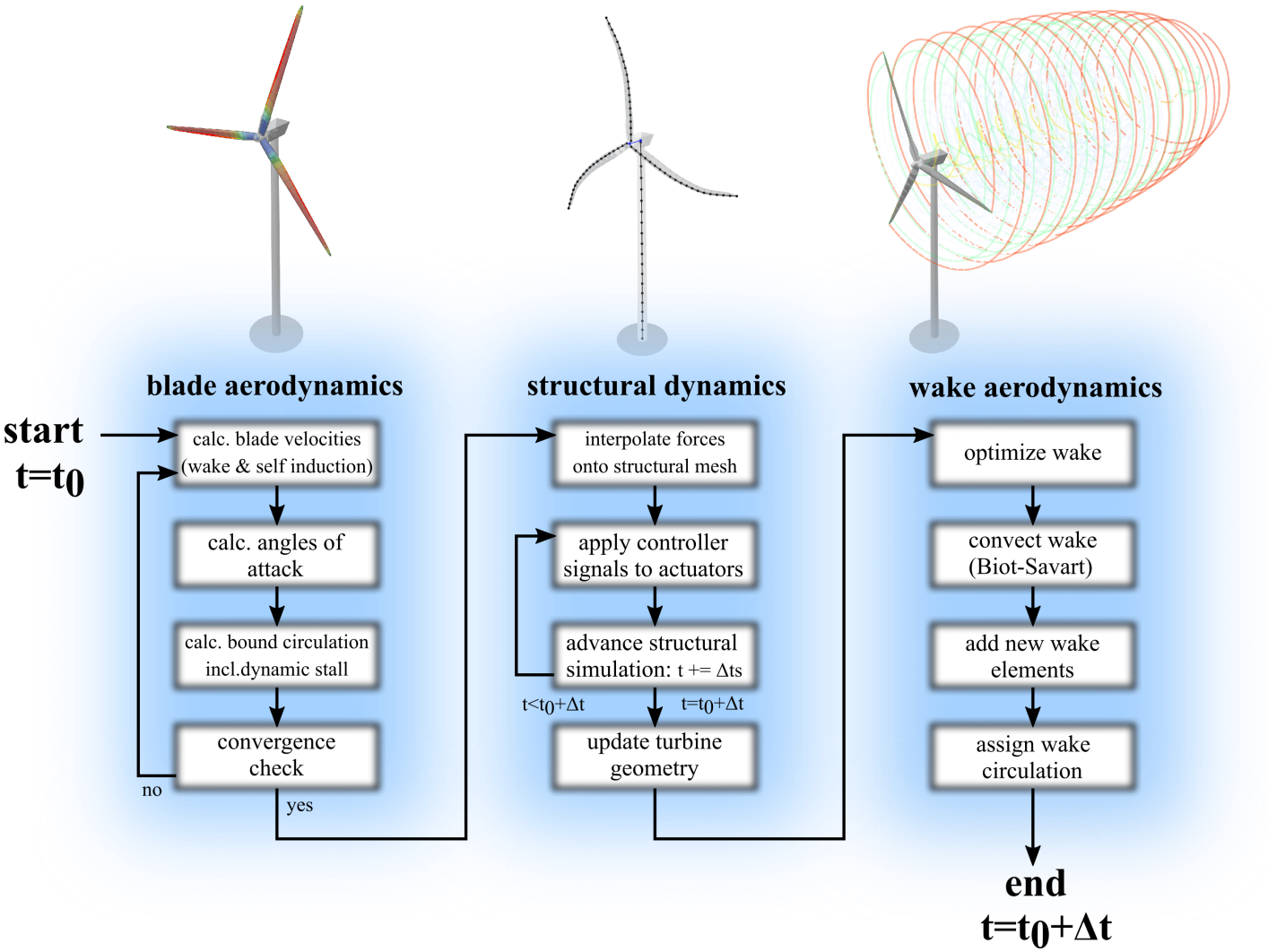


Fig. 21 *Flowchart for one time step of the aeroelastic model in QBlade.*

- The simulation starts at the time $t = t_0$. First, the iteration to find a converged circulation distribution for the bound vorticity of the rotor blade is performed. In this evaluation, the expensive step of calculating the wake induced velocities for the rotor panels is only carried out once at the beginning of the iteration, as the wake shape remains fixed during the iteration. During this iteration, the calculation of the blade panels lift and drag characteristics, the unsteady aerodynamics model (see Dynamic Stall) and other corrections, such as Snel's correction (see Himmelskamp Effect), are applied. Once a converged solution for the bound circulation is obtained, the blade aerodynamics calculations are finished.
- The blade panel forces and moments, resulting from the associated airfoils lift-, drag- and [illegible]ent characteristics, are interpolated from the aerodynamic discretization (panels) onto the structural dynamics discretization (beam elements). Furthermore, the controller signals are applied onto the

actuators. The simulation is now advanced with the structural time step Δt_s. If the current simulation time has reached $t = t_0 + \Delta t$, the structural dynamics simulation is finished. If $t < t_0 + \Delta t$ the structural simulation is incremented again with Δt_s, applying new controller signals and rotating the aerodynamic forces and moments [1] with the incremental rotor advancement. This is repeated until the simulation time has reached $t = t_0 + \Delta t$. Now, the current positions of all beam elements and rigid bodies are interpolated back onto the aerodynamic mesh and the aerodynamic model is advanced onto its final position for this time step.

- In the last step, the wake is updated. The wake discretization is optimized by removing or lumping wake elements (see Lifting Line Free Vortex Wake). Now, the wake is updated by evaluating the self-induced wake velocities at the wake nodes, updating the vortex core sizes and advancing the wake element positions with Δt using one of the implemented time integrators (see Multi Body Beam Formulation). The gap that now exists between the new wake positions and the advanced rotor blade positions is 'filled' with new shed and trailing wake elements. Finally, the circulation of the newly created wake elements is assigned using the Kutta condition.

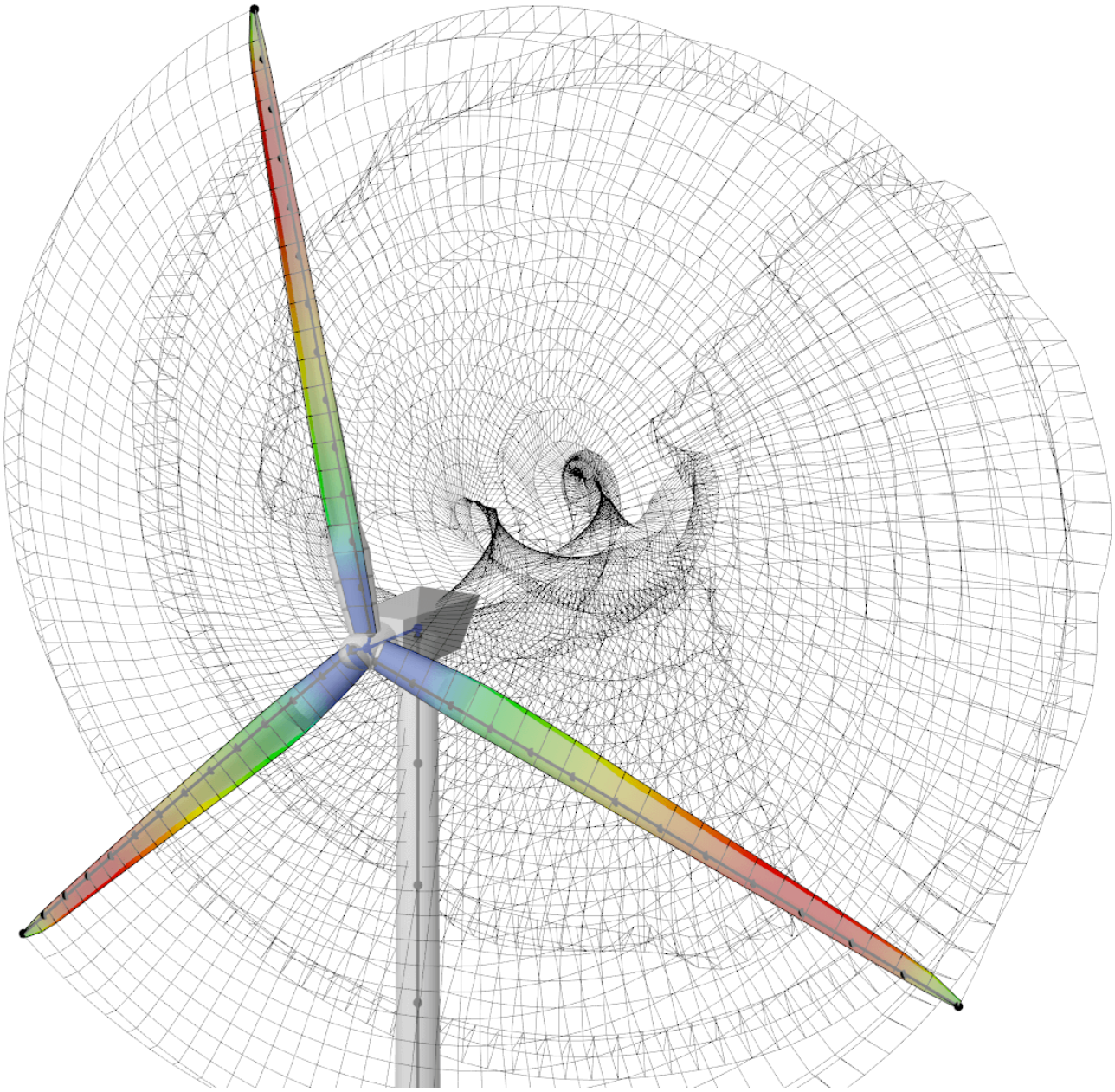


Fig. 22 *Visualization of the coupled, aero-elastic model.*

[1] These forces remain constant during the structural sub time steps.

Structural Data Tables

Col. Nr.	Name	Explanation	Unit
1	Length	Curved length distance from the first body node normalized by the body length	-
2	Mass density	Mass per unit length	kg/m
3	Bend. stiff. X	Bending Stiffness around X (EI_{xx})	Nm^2
4	Bend. stiff. Y	Bending Stiffness around Y (EI_{yy})	Nm^2
5	Axial stiff.	Longitudinal Stiffness (EA)	N
6	Tors. stiff.	Torsional Stiffness (GJ)	Nm^2
7	Shear stiff.	Shear Stiffness (GA) (not used with Euler beams)	N
8	Str. pitch	Structural pitch angle between reference X axis and elastic X axis	deg
9	Shear factor X	Shear factor for force in principal bending axis X	-
10	Shear factor Y	Shear factor for force in principal bending axis Y	-
11	Radius of gyration X	Norm. radius of inertia corresponding to a rotation around the elastic axis X	%chord
12	Radius of gyration Y	Norm. radius of inertia corresponding to a rotation around the elastic axis Y	%chord
13	Center of mass X	Norm. center of mass position X	%chord
14	Center of mass Y	Norm. center of mass position Y	%chord
15	Center of elast. X	Norm. center of elasticity position X	%chord
16	Center of elast. Y	Norm. center of elasticity position Y	%chord
17	Center of shear X	Norm. center of shear position X	%chord
18	Center of shear Y	Norm. center of shear position Y	%chord
19	Diameter	Cross section diameter	m
20	Drag	Drag coefficient for aerodynamic drag	-

Soil

In both onshore and fixed bottom offshore installations, a wind turbine is fixed in position through a suitable foundation. A variety of foundations are available depending on the site conditions, material choices, desired interaction and dynamics. A generalized model is implemented within QBlade in order to be able to simulate accurately not only pile foundations but also other foundations of interest. This model makes use of a distributed spring model with nonlinear spring coefficients as illustrated in Fig. 23. The spring coefficients are specified with linear segments which represent the restoring force as a function of displacement, these are commonly referred to as p-y curves [1]. For simplicity, the springs shown in this figure provide a restoring force in a single direction, however the model allows specification of nonlinear restoring forces in all translational and rotational degrees of freedom.

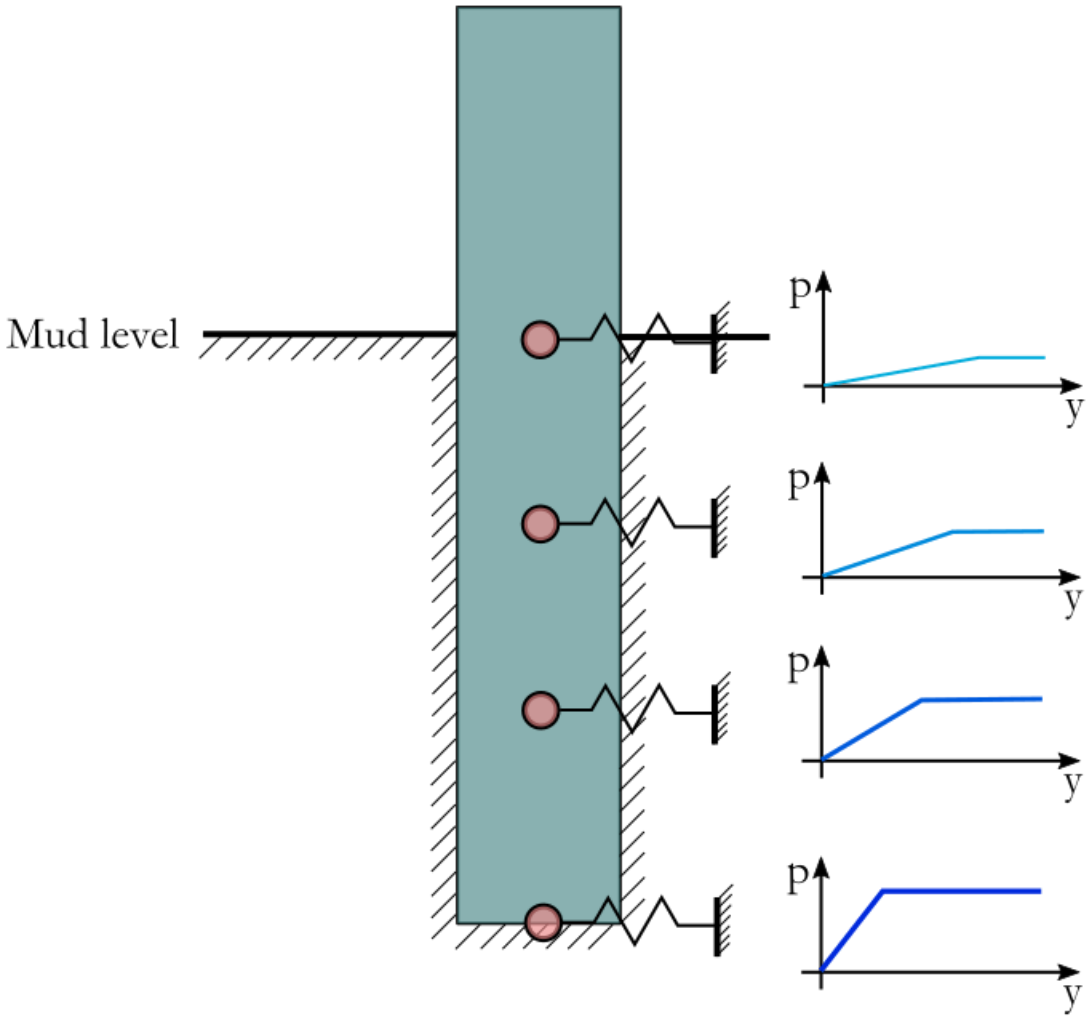


Fig. 23 *P-y curves for specification of nonlinear spring coefficients. The dynamics of the foundati is case shall be captured with four lateral springs.*

This generalized approach furthermore allows the structure of the foundation to be discretely treated, capturing the full structural dynamics of the foundation. This influences greatly the natural frequencies of the structure.

[1] R. Salgado. *The Engineering of Foundations*. McGraw-Hill Education, 2006. ISBN 0072500581.

Linear Wave Theory

Linear wave theory (also referred to as Airy waves in the literature) may be applied when the sea water is assumed to be incompressible, inviscid and the fluid motion is irrotational. In addition to this is it assumed that the wave steepness is very small. Subsequently, a velocity potential can be used to describe the velocity vector (u,v,w) of a water particle at an arbitrary position (x,y,z) at any time (t) Energy[1]. Following these assumptions, the linear wave model in QBlade is implemented through the following set of equations. The velocity potential for finite depth may be expressed as

$$\Theta(x,y,z,t) = -\frac{gA}{\omega}\frac{cosh(k(z+h))}{cosh(kh)}cos(\omega t - kx)$$

where,

- Θ is the velocity potential,
- g is the graviational acceleration,
- A is the wave amplitude,
- ω is the ciruclar frequency,
- h is the water depth,
- t is the time,
- x, y, z are Cartesian coordinates,
- k is the wave number.

The wave number k may be be expressed through the approximation of the dispersion relationship Guo[2]

$$k = \frac{\omega^2}{g}\left(1 - e^{-\omega\sqrt{\frac{h}{g}}^{\frac{5}{2}}}\right)^{-\frac{2}{5}}.$$

For infinite water depth the velocity potential is defined as:

$$\Theta(x,y,z,t) = -\frac{gA}{\omega}e^{kz}cos(\omega t - kx).$$

Regular Waves

The wave elevation of a regular wave train at any position of the free surface at a given time t is defined by

$$\zeta(x,y,t) = Asin(kX - \omega t)$$

where,

- ζ is the wave elevation,
- X is the ordinate in wave direction defined as $X = xcos(\Theta) + ysin(\Theta)$.

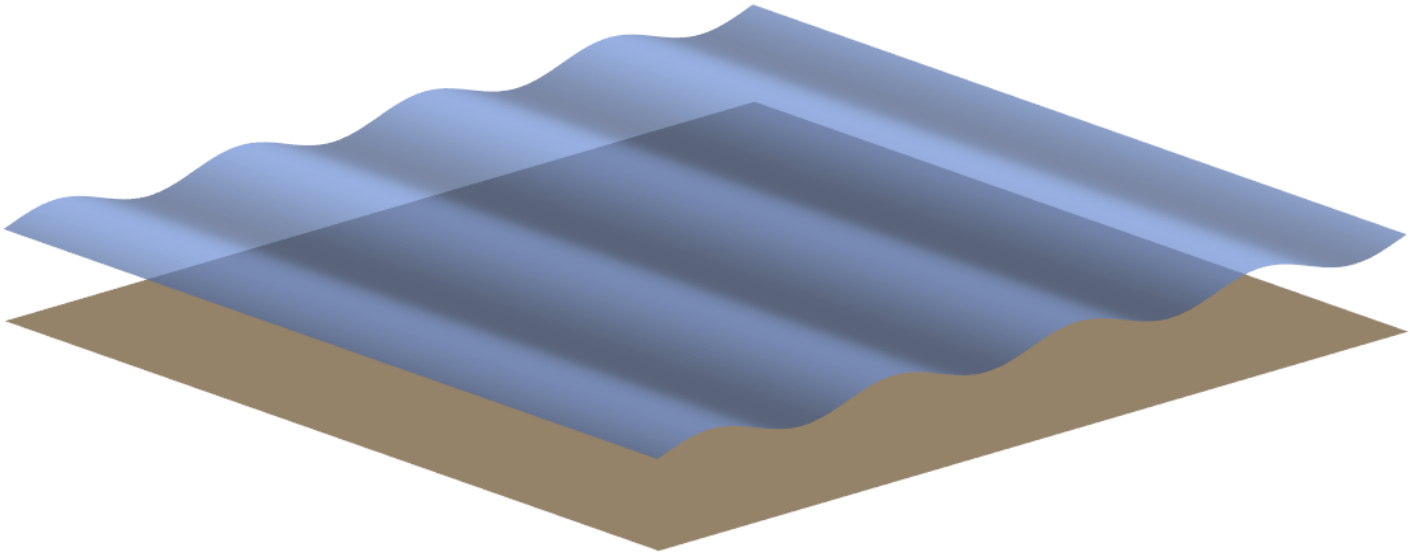


Fig. 24 *Regular wave field created in QBlade*

Unidirectional Irregular Waves

Irregular wave fields in QBlade belong to a short-term description of the sea, meaning that the significant wave height and the mean wave period are assumed to be constant over the considered time Faltinsen[3]. The irregular wave field is created through a linear superposition of N different linear waves trains. Each one of these wave trains is described by an amplitude A_j, a circular frequency ω_j and a phase ϵ_j $(0 - 2\pi)$. Hence, the wave elevation of an irregular wave field propagating along one direction (x-axis in this case) can be expressed through the following equation.

$$\zeta(x, y, t) = \sum_{j}^{N_j} A_j sin(k_j x - \omega_j t + \epsilon_j)$$

in which N_j is the number of wave trains.

The amplitude A_j is to be determined by discretizing a wave spectrum $S(\omega)$ into bins corresponding to a frequency range $\Delta\omega$ Faltinsen[3].

$$\frac{1}{2} A_{2j} = S(\omega_j)\Delta\omega$$

Two of the most commonly used wave spectra are the Pierson-Moskowitz (PM), also known as C-spectrum, and the JONSWAP (JS) spectrum. The former was originally proposed for fully de

sea and the latter extends the PM-spectrum to include developing sea states Nestegård *et al.*[4]. The spectra are described by the following equations.

$$S_{PM} = \frac{5}{16}\left(\frac{f}{f_p}\right)^{-5} H_{2s} T_p e^{-\frac{5}{4}\left(\frac{f}{f_p}\right)^{-4}}$$

and

$$S_{JS} = A_\gamma S_{PM}(\omega)\gamma^{e^{-\frac{1}{2}\left(\frac{f-f_p}{\sigma f_p}\right)^2}}$$

where,

- T_p is the peak period,
- f_p is the peak frequency,
- H_s is the significant wave height,
- A_γ is a normalizing factor,
- γ is the peak shape parameter
- σ is the spectra width parameter

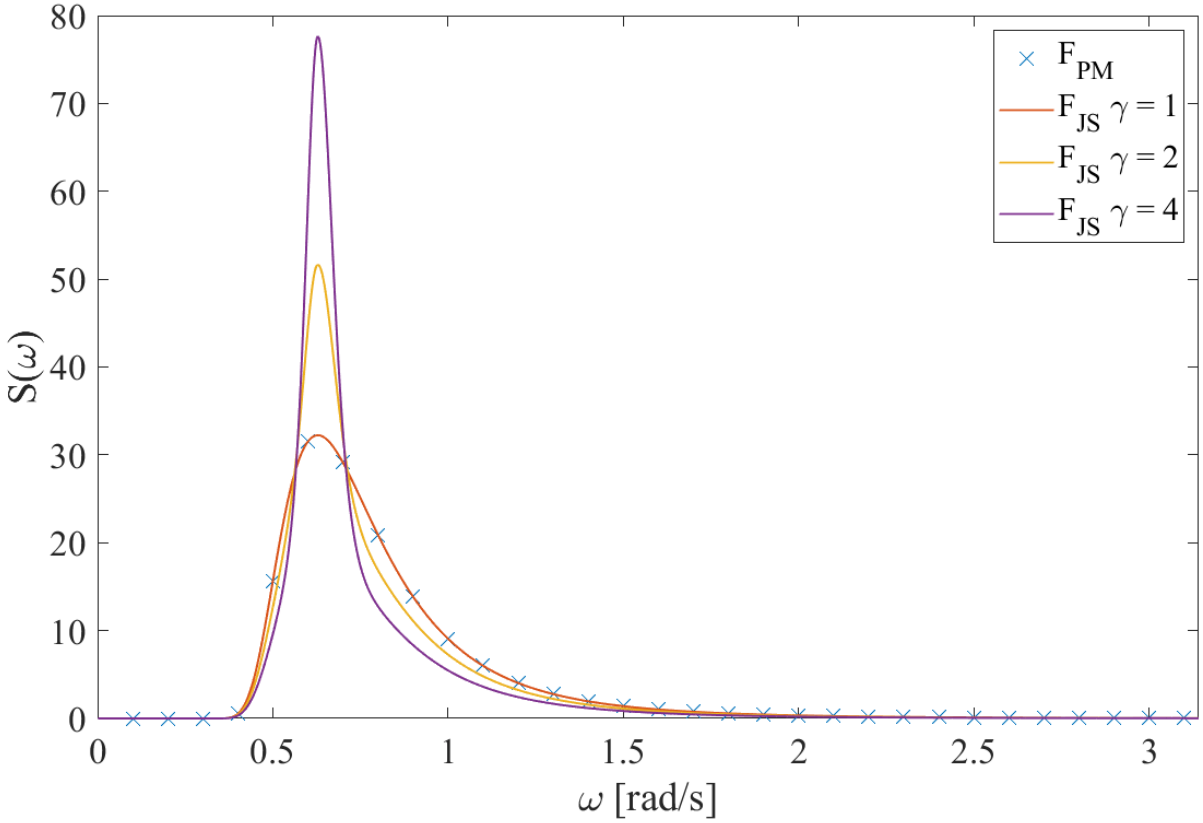


Fig. 25 *Person-Moskowitz an JONSWAP spectra with different peak shape parameters* γ

As visible in Fig. 25, the JONSWAP spectrum is a modification of the PM-spectrum by A_γ a normalizing factor, γ the peak shape parameter and σ the spectra width parameter Branlard[5].

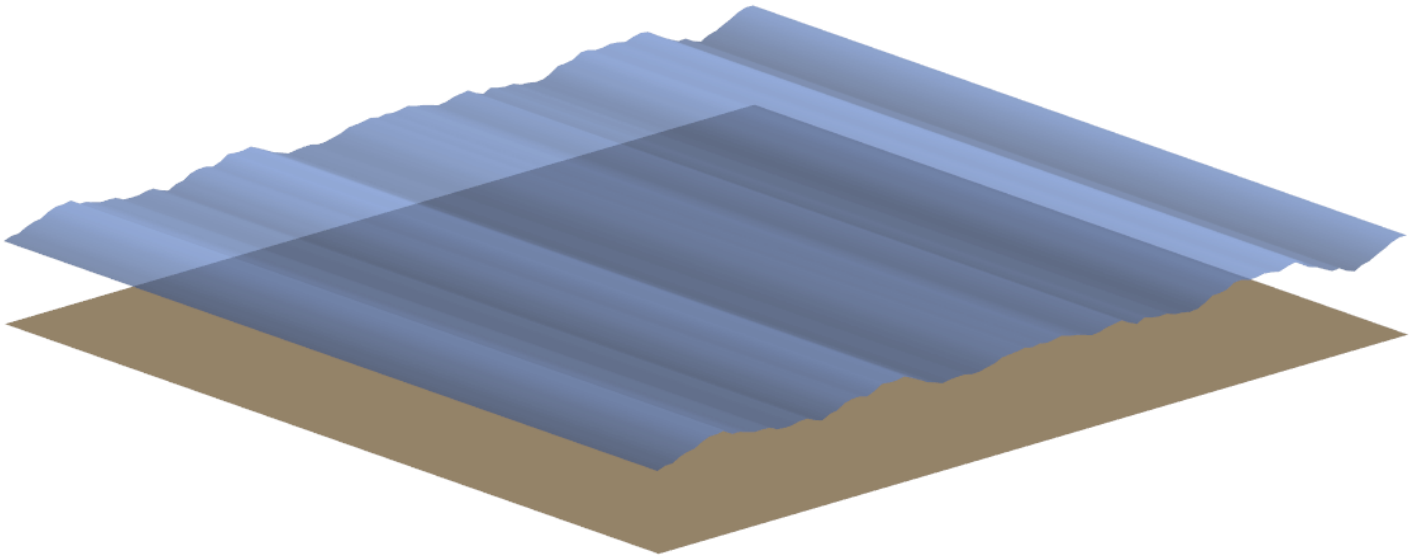


Fig. 26 *Irregular wave field created in QBlade*

Multidirectional Irregular Waves

A uni-directional wave spectrum $S(\omega)$ may be augmented through a directional function $D(\Theta)$ in order to create a multi-directional wave field Faltinsen[3]

$$S(\omega\Theta) = S(\omega)D(\Theta)$$

where Θ is the wave angle. The directional spectrum $D(\Theta)$ is implemented in QBlade as defined in OrcaFlex[6]

$$D(\Theta) = C(s)cos^{s}(\Theta - \Theta_p)$$

$C(s)$ is a normalizing constant that is defined as

$$C(s) = \frac{\Gamma(\frac{s}{2}+1)}{\sqrt{(\pi)}\Gamma(\frac{s}{2}+\frac{1}{2})}.$$

In this equation,

- s is the spreading exponent,
- Θ_p is the principal wave direction,

When the directional spectrum is added, the equation of the wave elevation needs to be advanced by another summation term over the number of directions of the wave trains. Subsequently, the wave amplitude needs to be extended by the wave direction

$$\zeta(x,y,t) = \sum_{i}^{N_i}\sum_{j}^{N_j} A_{ij}sin(k_i X_j - \omega_i t + \epsilon_{ij}),$$

$$\frac{1}{2}A_{2ij} = S(\omega_j, \Theta_j)\Delta\omega\Delta\Theta.$$

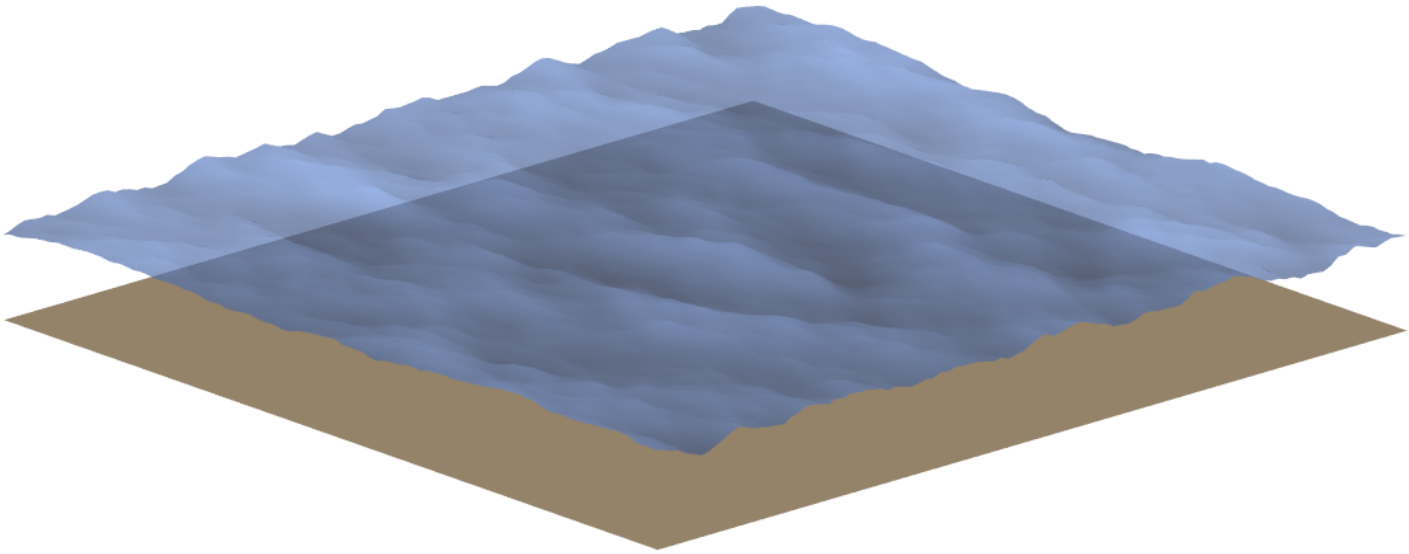


Fig. 27 *Irregular, multi-directional wave field created in QBlade*

[1] DNV GL - Energy. Theory Manual Bladed. 2014. [Version 4.6].

[2] Junke Guo. Simple and explicit solution of wave dispersion equation. *Coastal Engineering*, 45:71–74, 2002.

[3] (1, 2, 3) O. M. Faltinsen. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press, 1993.

[4] Arne Nestegård, Marit Ronæss, Øistein Hagen, Knut Ronold, and Elzbieta Maria Bitner-Gregersen. On environmental conditions and environmental loads. In *DNV Recommended Practice DNV-RP-C205*. 05 2006.

[5] E. Branlard. Generation of time series from a spectrum. Technical Report, Risø DTU National Laboratory for Sustainable Energy, 2010.

[6] OrcaFlex. Directional spread spectrum. https://www.orcina.com/webhelp/OrcaFlex/Content/html/Wavetheory.htm#WaveSpreadingTheory, 2021.

Wave Kinematics

The velocity and acceleration profile over the water depth may be derived from the velocity potentials (finite and infinite depth). For simplicity, the distinction between uni- and multi-directional wave fields is neglected in this section. In the case of a uni-directional wave field, the first summation term becomes redundant. In the case of infinite depth (for most waves of interest this represents a depth greater than 100m), the velocity profiles are defined by:

$$V_x = \sum_i^{N_i} \sum_j^{N_j} A_{ij} \omega_i cos(\Theta_j) E_c(z) sin(k_i X_j - \omega_i t + \epsilon_{ij},$$

$$V_y = \sum_i^{N_i} \sum_j^{N_j} A_{ij} \omega_i sin(\Theta_j) E_c(z) sin(k_i X_j - \omega_i t + \epsilon_{ij},$$

$$V_z = \sum_i^{N_i} \sum_j^{N_j} -A_{ij} \omega_i E_s(z) cos(k_i X_j - \omega_i t + \epsilon_{ij}.$$

Hence, the acceleration may be derived:

$$a_x = \sum_i^{N_i} \sum_j^{N_j} -A_{ij} \omega_{2i} cos(\Theta_j) E_c(z) cos(k_i X_j - \omega_i t + \epsilon_{ij},$$

$$a_y = \sum_i^{N_i} \sum_j^{N_j} -A_{ij} \omega_{2i} sin(\Theta_j) E_c(z) cos(k_i X_j - \omega_i t + \epsilon_{ij},$$

$$a_z = \sum_i^{N_i} \sum_j^{N_j} -A_{ij} \omega_{2i} E_s(z) sin(k_i X_j - \omega_i t + \epsilon_{ij}.$$

E_c and E_s are depth scaling factors that depending on the case are defined as:

$$E_c(z) = \frac{cosh(k_i(z+h))}{sinh(k_i h)}$$

$$E_s(z) = \frac{sinh(k_i(z+h))}{sinh(k_i h)}$$

for finite depth and:

$$E(z) = e^{k_i z}$$

for infinite depth.

Kinematic Stretching

Linear wave theory only provides information about the water kinematics at and below mean sea level (MSL). If the velocity or acceleration within points above MSL is of interest (i.e. in a wave crest), extrapolation or stretching methods become necessary Nestegård *et al.*[1].

In the literature, several wave stretching methods have been introduced Nestegård *et al.*[1], OrcaFlex[2], FRyDoM[3]. Their general approach is to model the respective scaling factor $E(z)$ (introduced in Wave Kinematics) for points above MSL $(z > 0)$ by stretching or extrapolating its values. In the following, the three methods that have been implemented into QBlade are introduced briefly. For further information the reader is referred to Nestegård *et al.*[1], OrcaFlex[2], FRyDoM[3].

Vertical Stretching

This method assumes that all points above MSL equal the kinematic conditions at MSL $(E(z) = 0)$. E below MSL is left unchanged

Extrapolation Stretching

This method extrapolates $E(z)$ above MSL linearly by using its gradient along the z-axis. Again, $E(z)$ below MSL is left unchanged.

Wheeler Stretching

This method modifies $E(z)$ so it always is stretched (or contracted) to the instantaneous wave elevation ($z = \zeta$). This is done by replacing z with a scaling factor z' that modifies z linearly so the following statement is always valid $h < z' < 0$, where h Is the water depth.

[1] (1, 2, 3) Arne Nestegård, Marit Ronæss, Øistein Hagen, Knut Ronold, and Elzbieta Maria Bitner-Gregersen. On environmental conditions and environmental loads. In *DNV Recommended Practice DNV-RP-C205*. 05 2006.

[2] (1, 2) OrcaFlex. Kinematic stretching. https://www.orcina.com/webhelp/OrcaFlex/Content/html/Waves,Kinematicstretching.htm, 2021.

[3] (1, 2) FRyDoM. Kinematic stretching. http://theory.frydom.org/src/environment/waves/linear_wave_theory/wave_stretching.html, 2021.

Currents

Three different types of time constant currents may be defined in QBlade. An overview of the available types of currents is given below. Further details can be found in Energy[1].

- **Near-Surface Currents**: The velocity profile of a near-surface current varies linearly with depth from a specified velocity at the sea surface to zero at the reference depth, which is defined by the user.

- **Sub-Surface Currents**: The sub-surface current velocity follows a power law profile. The implementation in QBlade is of the following form:

$$u_{cs}(z) = \left[\left(\frac{z+h}{h}\right)^{\alpha}\right] u_{s0}(z=0)$$

where,

- $u_{cs}(z=0)$ is the velocity at the sea surface,
- α is the power law exponent (deault value is $\alpha = 0.14$),
- h is the water depth,
- z is $0 \geq z \geq -h$.

- **Near-Shore Currents**: The near-shore current is defined as a uniform velocity profile independent of the depth

Any combination of these types of currents (together with waves) may be included within a QBlade simulation. In all cases, the velocities at each evaluation point are calculated as a superposition of all contributions from waves and currents. A complete hydroelastic representation of the turbine also requires the consideration of fluid-structure interaction. This topic is covered in Sections Linear Potential Flow Theory and Morison Equation.

[1] DNV GL - Energy. Theory Manual Bladed. 2014. [Version 4.6].

User's Guide

General GUI Functionality

- GUI Overview
- General Settings
- Opengl Light Settings
- Graph Functionality
    - Graph Context Menu
    - Variables & Styles and Axes
    - Curve Styles
    - Graph Layout

Data Structure, Import & Export

- Object Hierarchy and Data Structure
- Project Serialization
- Data Objects Import and Export

Coordinate Systems

- Global Coordinate System
- Local Body Coordinate Systems
    - Local Blade Coordinate System
    - Local Tower Coordinate System
- Local Sensor Coordinate Systems

HAWT, VAWT and PROP Modes

- Setting the Design Mode
- HAWT Mode, VAWT Mode and PROP Mode

Airfoil Generation and Import

- Airfoil Generation Overview
- Creating Standard Airfoils
- Importing Airfoils

- Airfoil Editing Options
- Airfoil Transformations
- Exporting Airfoils

Airfoil Analysis with XFoil

- Airfoil Analysis Overview
- Carrying out an XFoil Analysis
    - XFoil Batch Analysis
- Operational Point Analysis
- Importing Polar Data
- Exporting Polar Data

360° Polar Extrapolation

- Polar Extrapolation Overview
- Viterna Extrapolation
- Montgomery Extrapolation
- Polar Decomposition
- Dynamic Polar Sets
- Import and Export of 360 Polars

Aerodynamic Blade Design

- Blade Design Overview
- Basic Blade Design
    - Multi Polar Blade Definition
- Advanced Blade Design
    - Active Elements
    - Blade Damage
- Importing and Exporting Blade Definitions
- Blade definition ASCII File

Steady BEM Simulation

- BEM Analysis Overview
- Rotor BEM
- Characteristic BEM

- Turbine BEM

Wind Turbine Modeling

Controller Modeling

Substructure Modeling

Wind Turbine Simulation

Multi Turbine Simulation

Windfield Generation

Wave Generation

Design Load Case Generation

- Design Load Cases Overview
- DLC Object Generation (in GUI)
    - Template
    - Turbine Data
    - DLC Parameter Range
    - Wind Model
    - Turbulent Grid Parameters
    - Environmental Vars
    - General Sim Settings
    - Rotational Speed Setting
    - Simulation Event(Fault) Settings
    - Structural Sim Settings
    - Modal Analysis Settings
    - Stored Sim Data
    - Offshore DLC Generation in the GUI
- Exporting DLC Definitions
- DLC Definition via Spreadsheets
- DLC Generation via Spreadsheets
    - Importing DLC's from a Spreadsheet
    - Exporting DLC's from a Spreadsheet

Command Line Interface (CLI)

- CLI Overview
- CLI Functionality
- Sample CLI Call to Start a Batch Run

Software in Loop Interface (SIL)

- Software in Loop (SIL) Overview
- Quick Start with the SIL Interface in Python
- Interface Function Definitions
- Interface Function Documentation
- Python Example: Running the QBlade Library
- Python Example: Definition of the QBladeLibrary Class
- Matlab Example: Running the QBlade Library
- Matlab Example: Definition of the QBladeLibrary Class

Changelog

GUI Overview

A stand out feature of QBlade is the seamless integration of all functionality in a sophisticated and intuitive graphical user interface. The main concepts and terms of QBlades user interface are briefly described in the following image.

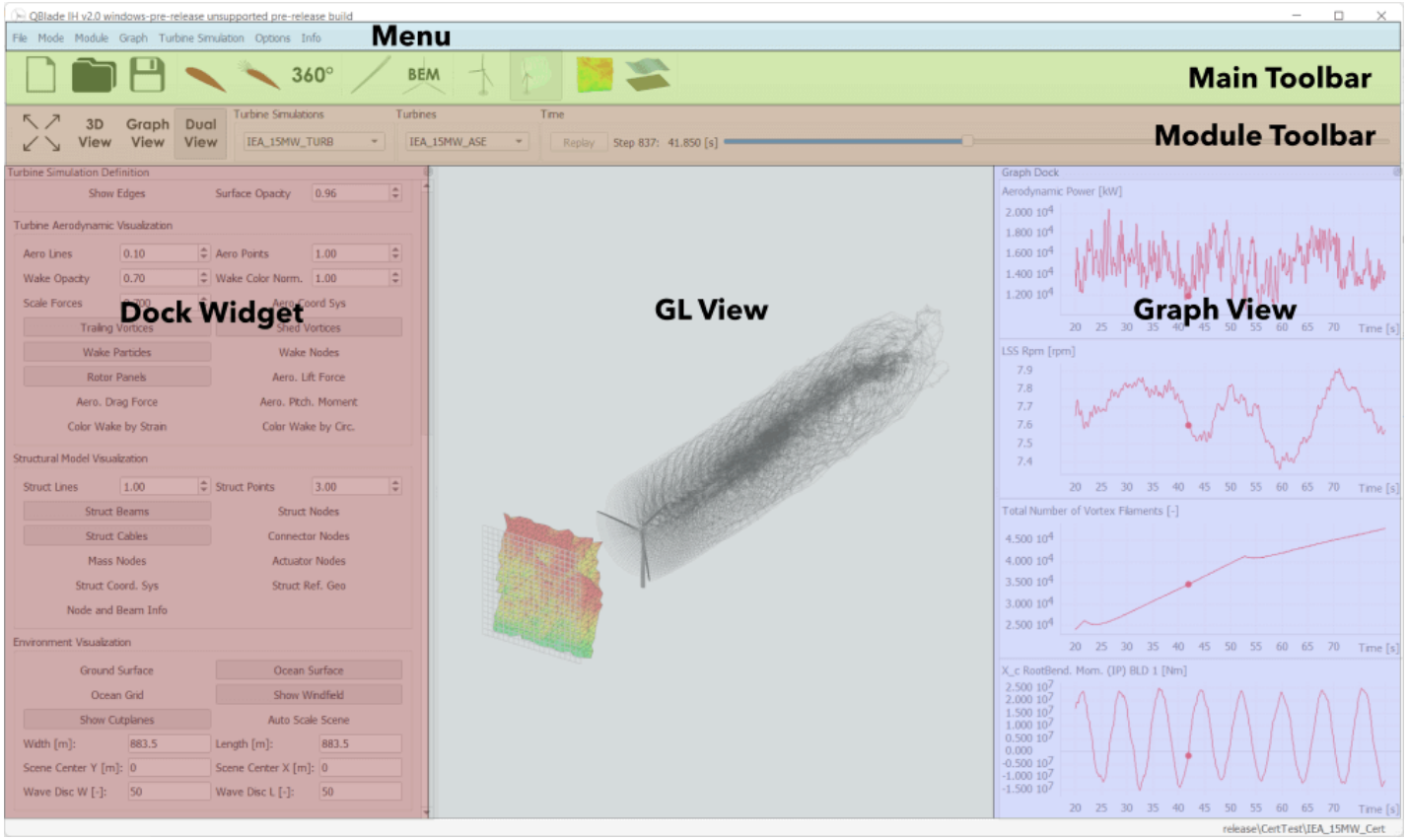


Fig. 32 *Overview of QBlades user interface.*

- The **Menu** allows to change between HAWT and VAWT mode and enables access to GUI customization and export functionality.
- The **Main Toolbar** is used to switch to quickly open or save project and to switch between QBlade's different modules.
- The **Module Toolbar** changes, depending on which module is currently selected. The main functionality of this Toolbar is to select data objects through the shown *Combo Boxes* and to switch between the *Graph View*, *Gl View* or *Dual View* modes
- The **Dock Widget** changes, depending on which module is currently selected. Here data objects can be created, edited or removed and simulation can be started. Furthermore it contains module specific functionality.
- The **GL View** is available for some modules and shows a rendered representation of the simulation or data object.
- The **Graph View** is used to plot simulation results or object data in QBlade's custom 2D G class.

General Settings

the general settings dialog (see Fig. 33) can be found in the top *Menu* under *Options->General Gui Appearance*.

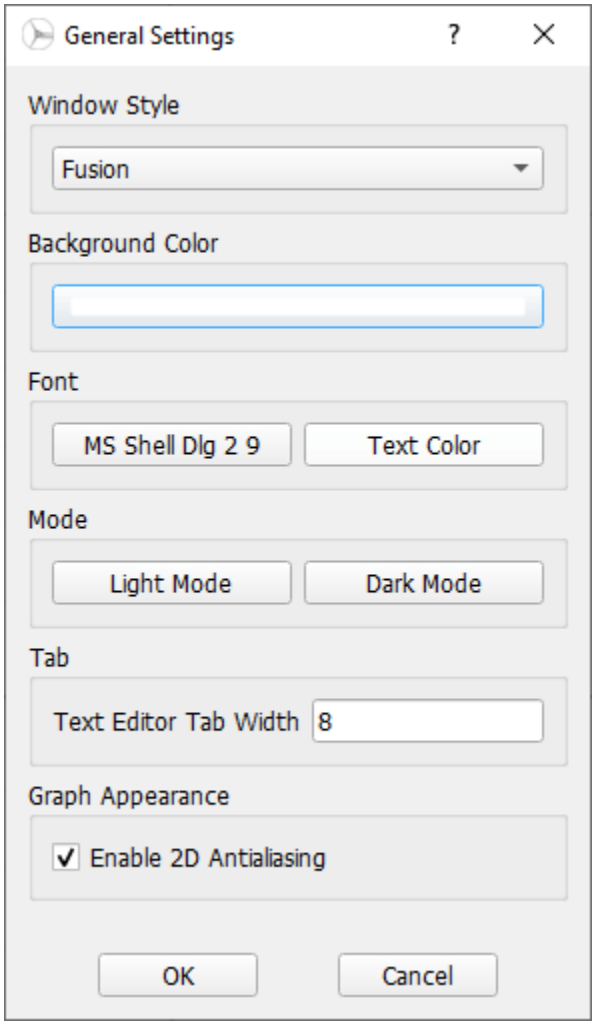


Fig. 33 *The general settings dialog.*

In this dialog the global font size and font type can be changed. Furthermore the overall wind style and several other global visualization options can be set.

Opengl Light Settings

The OpenGl Light Settings Dialog (see Fig. 34) can be found in the top *Menu* under *Options->OpenGl Light Settings*.

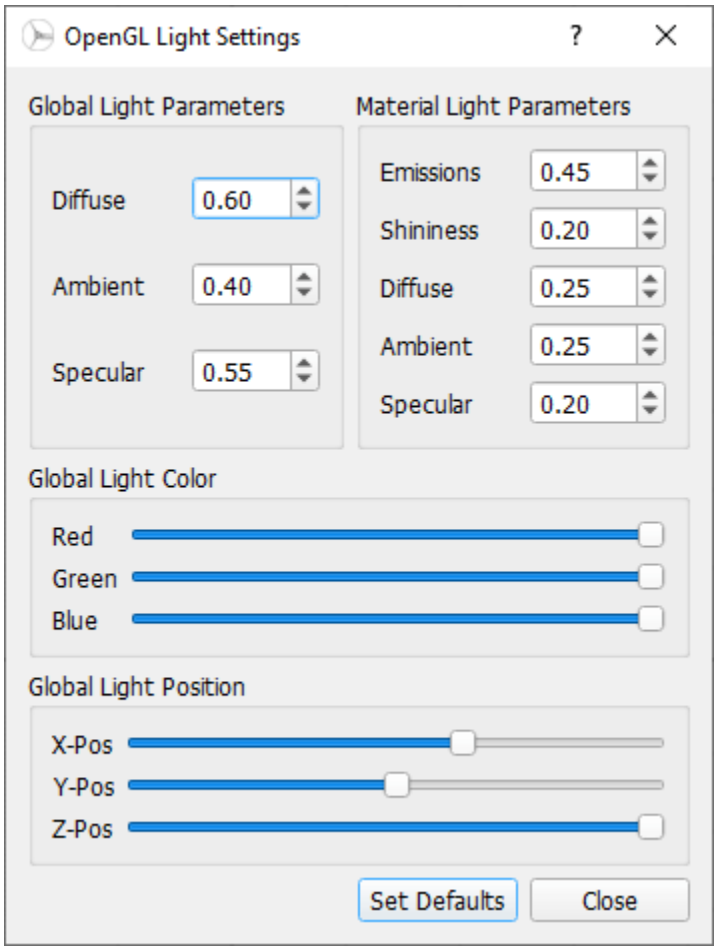


Fig. 34 *The opengl light settings dialog.*

This dialog allows to change the global illumination and light settings for all scenes that are rendered in 3D using opengl.

Graph Functionality

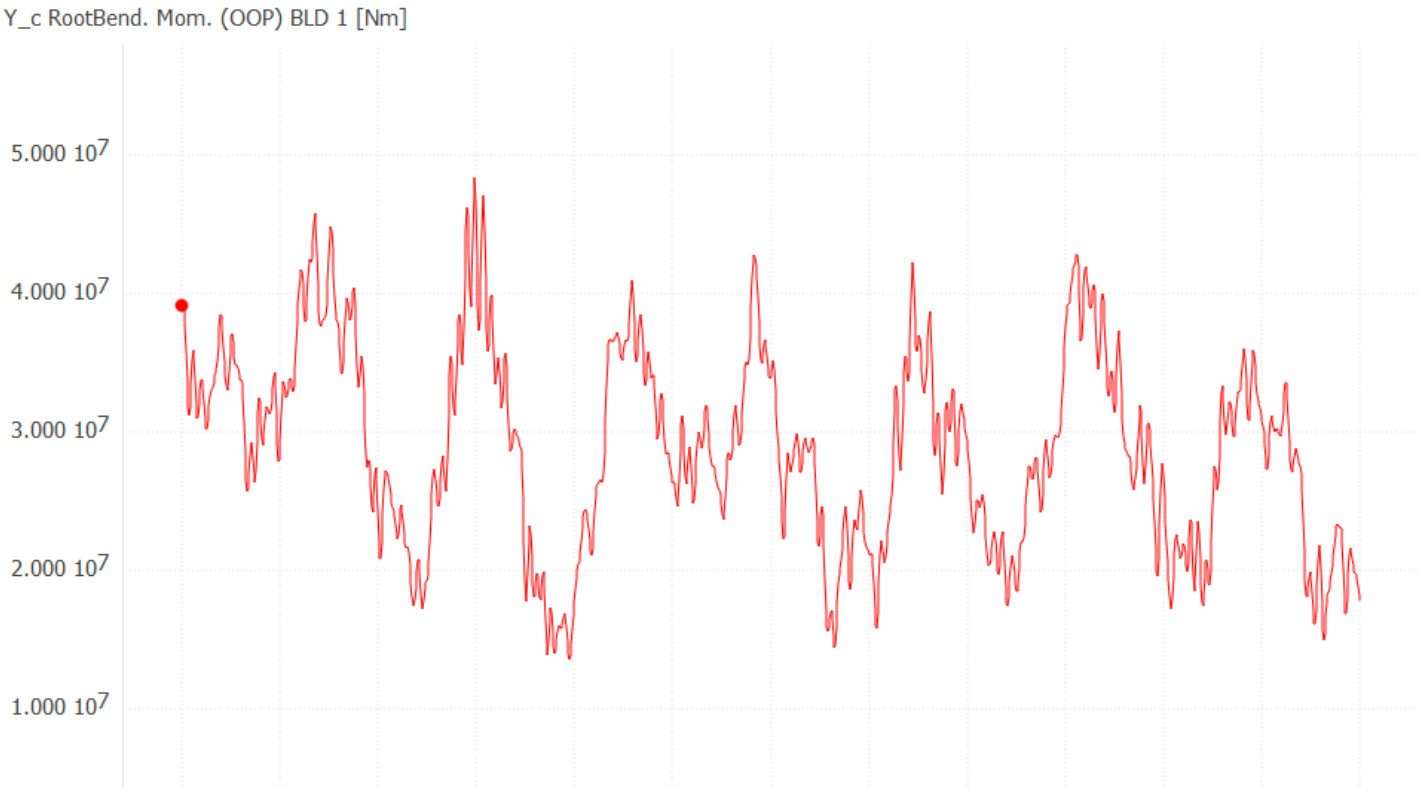


Fig. 35 *A QBlade Graph.*

Each of the modules in QBlade is equipped with custom graphs that are used to plot various parameters of a simulation or a data object. The graphs in QBlade can be considered the main tool for analyzing and exporting results in QBlade. Each graph can be customized individually. When closing a QBlade session all graph customizations are stored and are reloaded when QBlade is started again. The user can interact with a graph in the following ways:

- **Pan**: Hold down the left mouse key to pan the graph
- **Zoom**: Use the mouse wheel to zoom a graph. Holding down the *Y* or *X* key on the keybord allows to only zoom the x- or y-axis.

Graph Context Menu

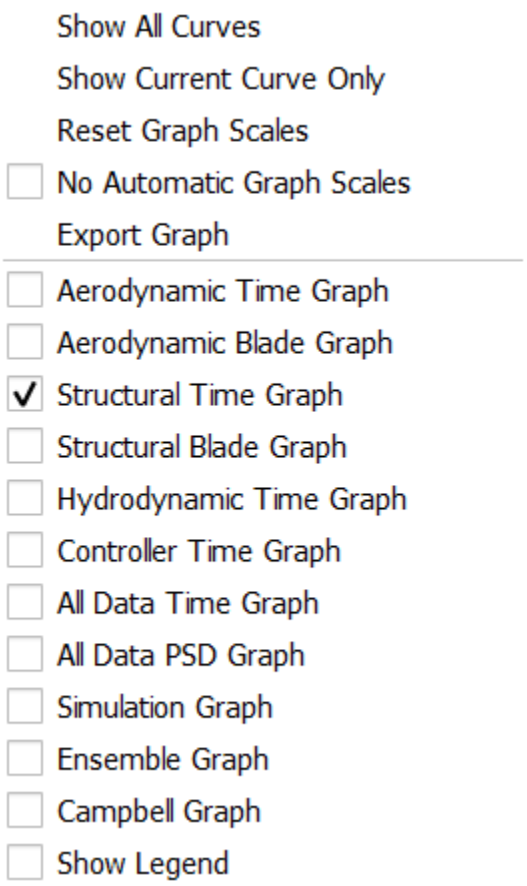


Fig. 36 *The Graph Context Menu.*

The *Graph Context Menu* contains two sections, separated by a horizontal line. The bottom section can be used to change the *Graph Type*. Depending on the module several different graph types are available. The top part of the *Graph Context Menu* is the same for all modules. The functionality is described below:

- **Show All Curves**: All data objects or simulations are displayed in the graph.
- **Show Current Curve Only**: Only the currently selected object curve is shown in the graph, all other object curves are hidden.
- **Reset Graph Scales**: The graph scales are adapted to best fit the currently shown curves
- **No Automatic Scaling**: Disables the automatic graph scaling, the graph scaling is frozen until this option is deselected again.
- **Export Graph**: The curves displayed in the currently selected graph are exported to a `.txt` file.

Variables & Styles and Axes

The *Variables* menu can be opened by a left mouse double click on any graph. The *Styles and Axes* menu is found by clicking on the *Styles and Axes* tab in the graph menu.

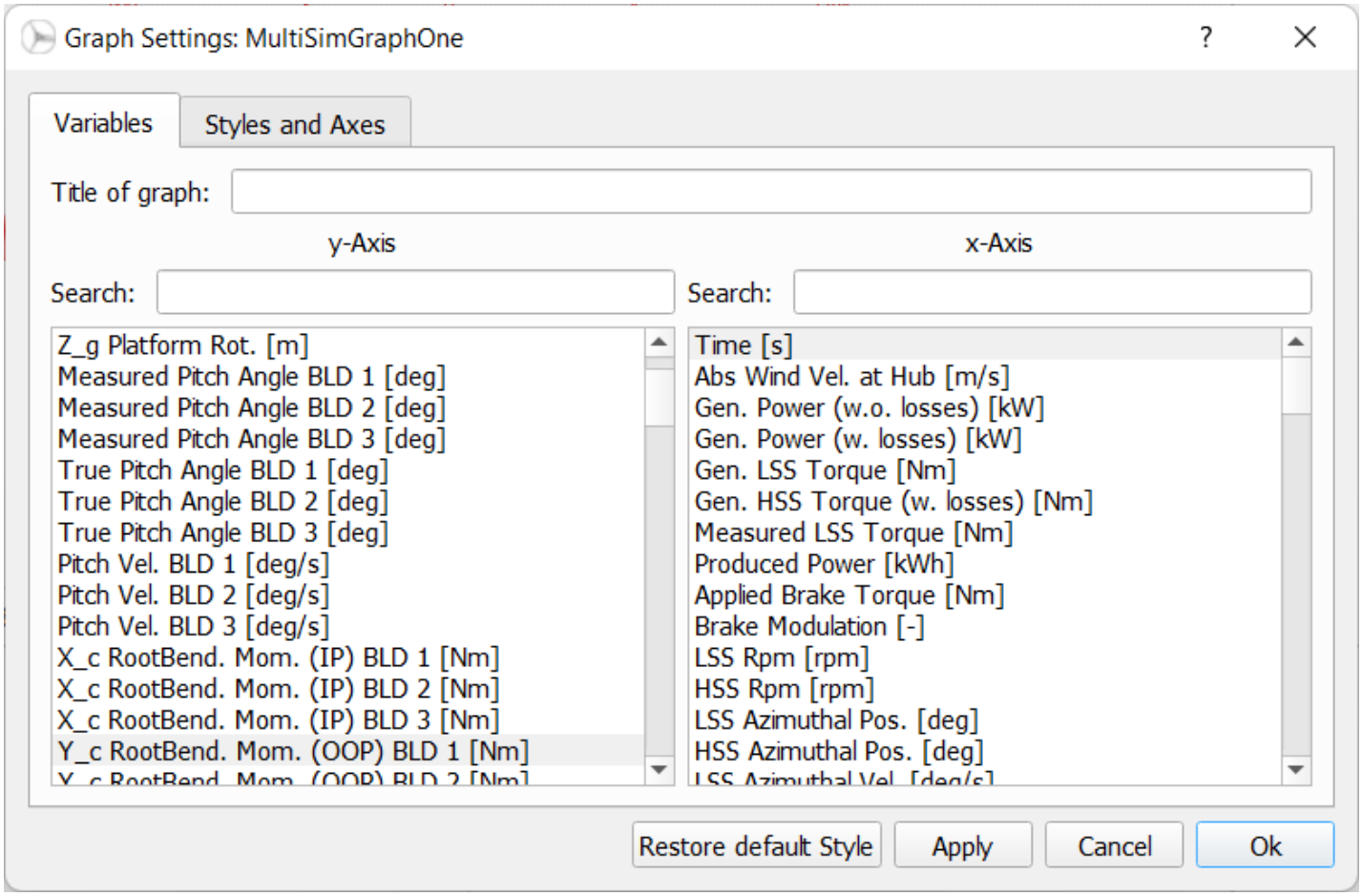


Fig. 37 *The Graph Variables Menu.*

The main functionality of the *Variables* menu is to select the parameter that is currently plotted. A variable can be selected for the x- and the y-axis. The *Search* edit can be used to search for a parameter in the graphs parameter list.

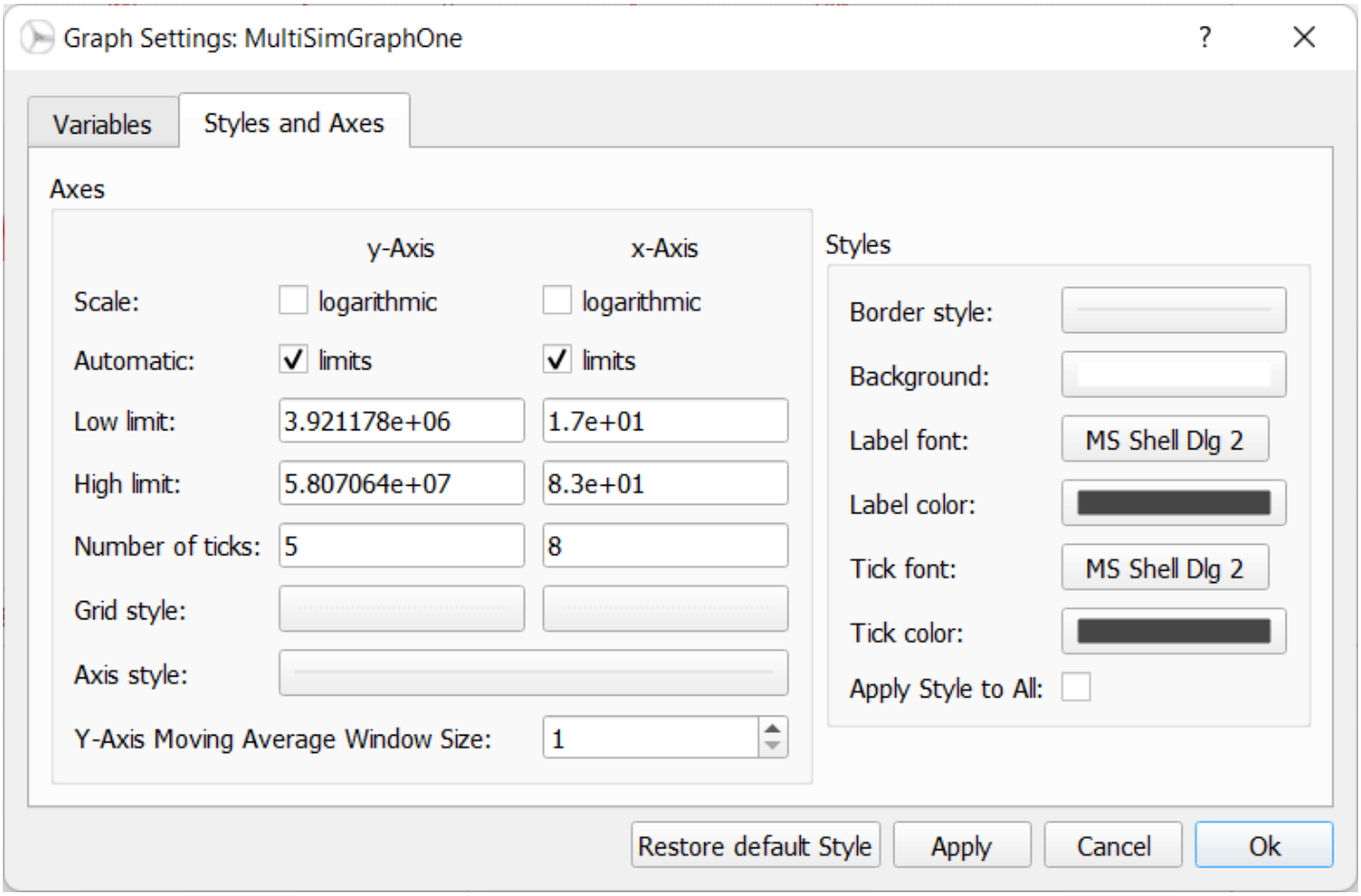


Fig. 38 *The Graph Styles Menu.*

The *Styles and Axes* menu can be used to customize the graph appearance and the graph limits. Furthermore a moving average window size can be defined in this menu that is applied to the currently plotted parameter.

Curve Styles

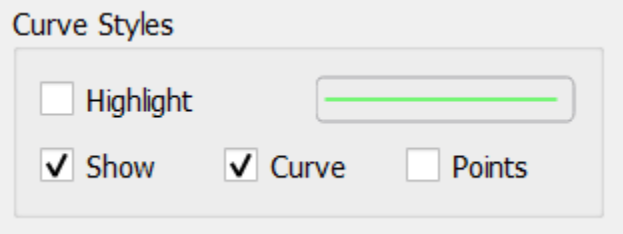


Fig. 39 *The Curve Styles Menu.*

When in the *Graph View* of a module the *Curve Styles* box is visible in the *Dock Widget*. The *Curve Styles* menu is used to set the appearance of the data curve of an object. By clicking on the colored line box the curve color, curve style and curve width can be changed by the user. Furthermore, the following options are avaliable:

- **Highlight**: If this checkbox is ticked the currently selected object will be highlighted by in g the width of the associated curve.
- **Show**: This checkbox toggles the visibility of the curve.

- **Curve**: This toggles if the curve is diaplyed.
- **Points**: This toggles if the individual data points are displayed.

Graph Layout

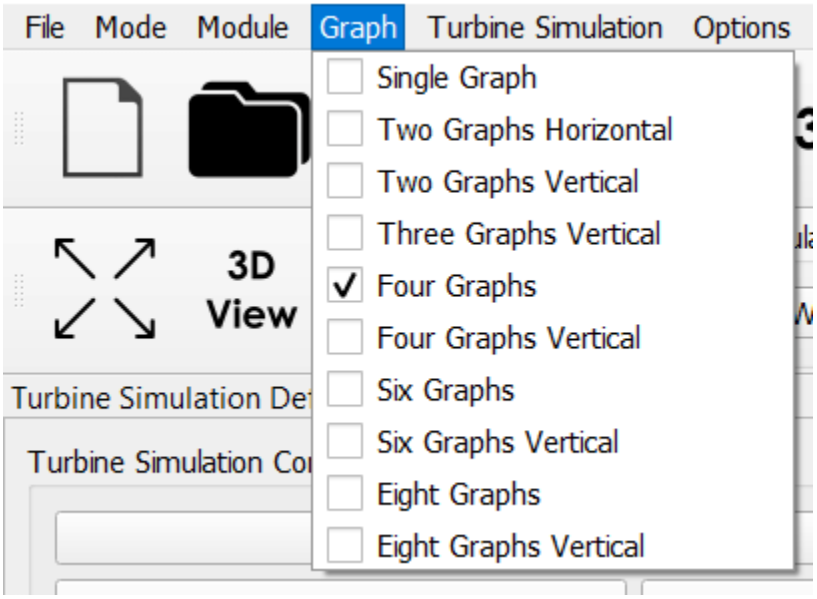


Fig. 40 *The Graph Layout Menu.*

The *Graph Layout Menu* can be accessed from the *Menu*. For each module an individual graph layout can be selected. The user can choose to display one, two, three, four, six or eight graphs in two layout options. When multiple graphs are displayed in QBlade each graph can be of a different *Graph Type* and can be configured with an individual appearance.

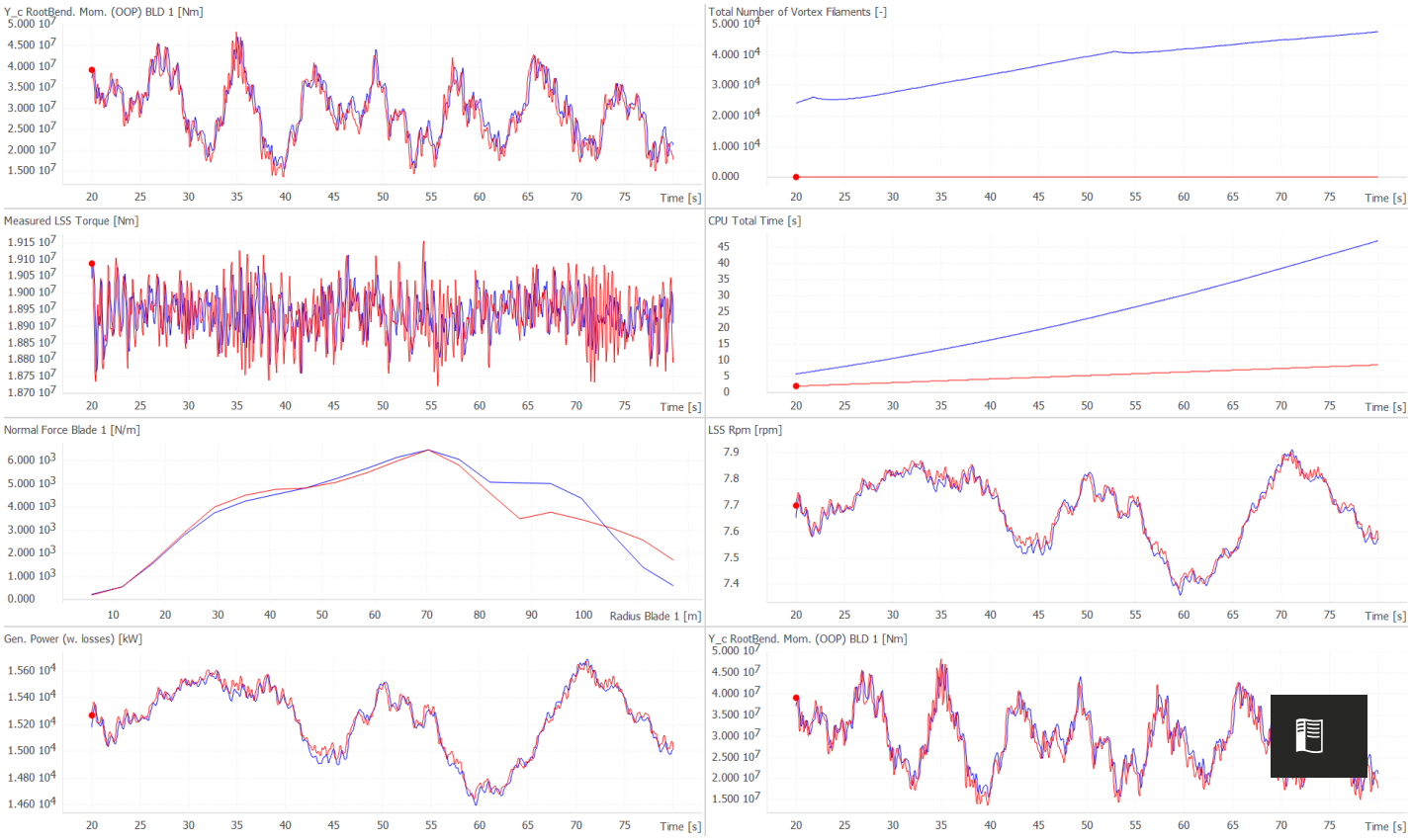


Fig. 41 *The 'Eight Graphs Vertical' Layout.*

Object Hierarchy and Data Structure

In the back-end of QBlade all simulation objects and all data is organized in a specific hierarchy, representing the 'building blocks' of a completer aero-servo-hydro-elastic wind turbine simulation. The data structure and object hierarchy of QBlade is shown in the following figure.

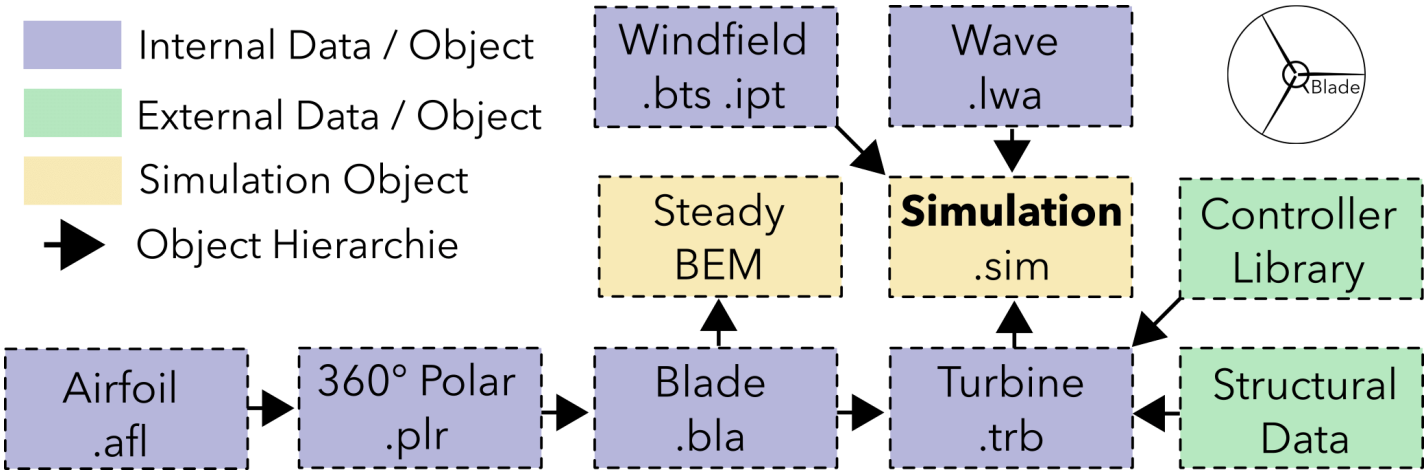


Fig. 42 *Overview of QBlades data structure.*

One important thing to note is that when a data object is deleted from QBlades database all objects associated with it, that are below in the object hierarchy, are automatically removed from the database. As an example: If a polar object is deleted all associated rotor blades are deleted and thus all associated simulations as well!

Project Serialization

All data and objects in QBlade can be serialized into the binary QBlade Project file formnat `.qpr`. QBlade project files allow easy sharing or saving of projects and simulation results.

Data Objects Import and Export

In QBlade several data objects exists. An aero-servo-hydro-elastic simulation definition is made up of several data objects (see Fig. 42). These objects are:

- **Simulation definition**: The simulation definition objects defines boundary conditions, discretization and simulation length and other parameters (`.sim` files)
- **Turbine definition**: The turbine definition object defines the aero-servo-hydro elastic model of the turbine that is simulated (`.trb` files)
- **Blade definition**: The blade definition contains the aerodynamic definition of the rotor blade (`.bld` files)

- **Polar**: The polar object contains the airfoil coefficients (`.plr` files)
- **Airfoil**: The airfoil object contains the outer contour of the airfoil (`.afl` files)
- **Windfield**: The windfield object contains the time resolved data of the inflow (`.bts` , `.hht` or `.inp` files)
- **Wavefield**: The wavefield contains the time resolved information of the seastate (`.lwa` files)

One main concept in QBlade is the ability to easily create and edit the use data objects. In the GUI objects can easily be created within custom dialogs. These objects can be exported into ASCII format and edited, or completely created, outside of QBlade and imported with ease.

User's Guide

General GUI Functionality

- GUI Overview
- General Settings
- Opengl Light Settings
- Graph Functionality
    - Graph Context Menu
    - Variables & Styles and Axes
    - Curve Styles
    - Graph Layout

Data Structure, Import & Export

- Object Hierarchy and Data Structure
- Project Serialization
- Data Objects Import and Export

Coordinate Systems

- Global Coordinate System
- Local Body Coordinate Systems
    - Local Blade Coordinate System
    - Local Tower Coordinate System
- Local Sensor Coordinate Systems

HAWT, VAWT and PROP Modes

- Setting the Design Mode
- HAWT Mode, VAWT Mode and PROP Mode

Airfoil Generation and Import

- Airfoil Generation Overview
- Creating Standard Airfoils
- Importing Airfoils

- Airfoil Editing Options
- Airfoil Transformations
- Exporting Airfoils

Airfoil Analysis with XFoil

- Airfoil Analysis Overview
- Carrying out an XFoil Analysis
    - XFoil Batch Analysis
- Operational Point Analysis
- Importing Polar Data
- Exporting Polar Data

360° Polar Extrapolation

- Polar Extrapolation Overview
- Viterna Extrapolation
- Montgomery Extrapolation
- Polar Decomposition
- Dynamic Polar Sets
- Import and Export of 360 Polars

Aerodynamic Blade Design

- Blade Design Overview
- Basic Blade Design
    - Multi Polar Blade Definition
- Advanced Blade Design
    - Active Elements
    - Blade Damage
- Importing and Exporting Blade Definitions
- Blade definition ASCII File

Steady BEM Simulation

- BEM Analysis Overview
- Rotor BEM
- Characteristic BEM

- Turbine BEM

Wind Turbine Modeling

Controller Modeling

Substructure Modeling

Wind Turbine Simulation

Multi Turbine Simulation

Windfield Generation

Wave Generation

Design Load Case Generation

- Design Load Cases Overview
- DLC Object Generation (in GUI)
    - Template
    - Turbine Data
    - DLC Parameter Range
    - Wind Model
    - Turbulent Grid Parameters
    - Environmental Vars
    - General Sim Settings
    - Rotational Speed Setting
    - Simulation Event(Fault) Settings
    - Structural Sim Settings
    - Modal Analysis Settings
    - Stored Sim Data
    - Offshore DLC Generation in the GUI
- Exporting DLC Definitions
- DLC Definition via Spreadsheets
- DLC Generation via Spreadsheets
    - Importing DLC's from a Spreadsheet
    - Exporting DLC's from a Spreadsheet

Command Line Interface (CLI)

- CLI Overview
- CLI Functionality
- Sample CLI Call to Start a Batch Run

Software in Loop Interface (SIL)

- Software in Loop (SIL) Overview
- Quick Start with the SIL Interface in Python
- Interface Function Definitions
- Interface Function Documentation
- Python Example: Running the QBlade Library
- Python Example: Definition of the QBladeLibrary Class
- Matlab Example: Running the QBlade Library
- Matlab Example: Definition of the QBladeLibrary Class

Changelog

Airfoil Analysis Overview

Once the desired blade section profiles have been selected in the Airfoil Generation Overview, the aerodynamic data for these profiles must be generated. This is accomplished within the airfoil creation module. This module is shown in the main toolbar in Fig. 53.

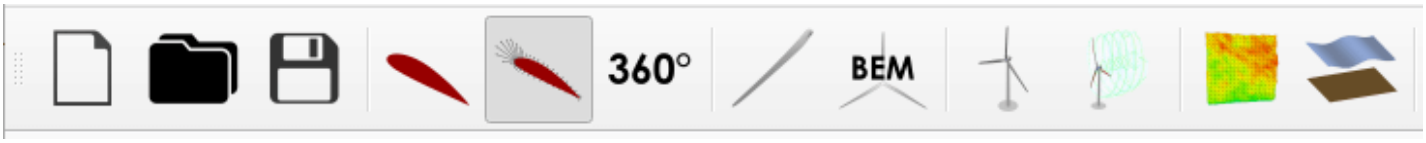


Fig. 53 *The airfoil analysis module is represented by the dynamic foil symbol in the QBlade main tool bar.*

The aerodynamic models within QBlade calculate local aerodynamic properties using airfoil look-up tables. These tables store the relevant aerodynamic quantities such as airfoil lift, drag and moment coefficients as a function of angle of attack α. There are two ways to generate these tables in QBlade. These two options are described below.

Generally the method used to generate the airfoil polars, whether experimental or numerical, only delivers reliable or repeatable values within a certain α range. For this reason, the polars generated or imported here may only correspond to a disjoint range: $\alpha \in [-180^o, 180^o]$. Extrapolating this range to ensure continuity of aerodynamic coefficients is the topic of the Polar Extrapolation Overview page.

Carrying out an XFoil Analysis

It is possible for the user to carry out an analysis using the 2D airfoil solver XFoil [1] directly within QBlade. This solver is linked to QBlade such that no preprocessing is required and the airfoil coordinates generated in the Airfoil Generation Overview are automatically prepared for analysis. An XFoil analysis can be carried out by generating a polar definition by selecting a new analysis from the polar control panel. The dialogue which then appears is shown in Fig. 54.

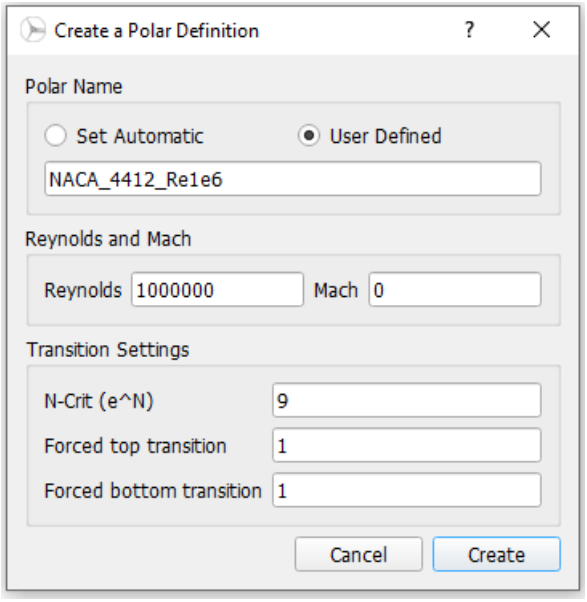


Fig. 54 *The XFoil polar creation dialogue.*

This requires the input of a range of parameters:

- **Polar Name**: This is the name stored for the generated polar.
- **Reynolds**: The Reynolds number of the analysis dictates the Reynolds number at which the airfoil is operating.
- **Mach**: The Mach number of the analysis. XFoil is also capable of treating transonic flow. For most wind energy applications however the flow around the airfoil is assumed to be incompressible.
- **N-Crit**: This is a parameter of the e^n model XFoil uses to predict when the boundary layer over the airfoil freely transitions from laminar to turbulent flow. [2]
- **Forced top transition**: If the e^n model is to be ignored on the suction side of the airfoil, this parameter gives the position of boundary layer transition (as a fraction of chord length).
- **Forced bottom transition**: If the e^n model is to be ignored on the pressure side of the airfoil, this parameter gives the position of boundary layer transition (as a fraction of chord length).

A range of advanced XFoil parameter settings can also be found in the *Polar* menu option under *XFoil Parameter Settings*. Once a polar object has been created, the operational points for the analysis can be selected. This is accomplished in the *Analysis Settings* module widget. The user must specify the start, end and delta values for α. Subsequently, the analysis is executed by clicking on the *Start Analysis* button. A progress bar will display the state of completion of the analysis.

XFoil Batch Analysis

For any given blade design, the different airfoil sections of a blade will be exposed to a range of operating conditions. For this reason, it can be beneficial to carry out an airfoil analysis for a range of Reynolds numbers for each of the airfoils that is used in the particular blade design. To streamline this process, QBlade features an XFoil batch option. The creation dialogue for this is shown in Fig. 55.

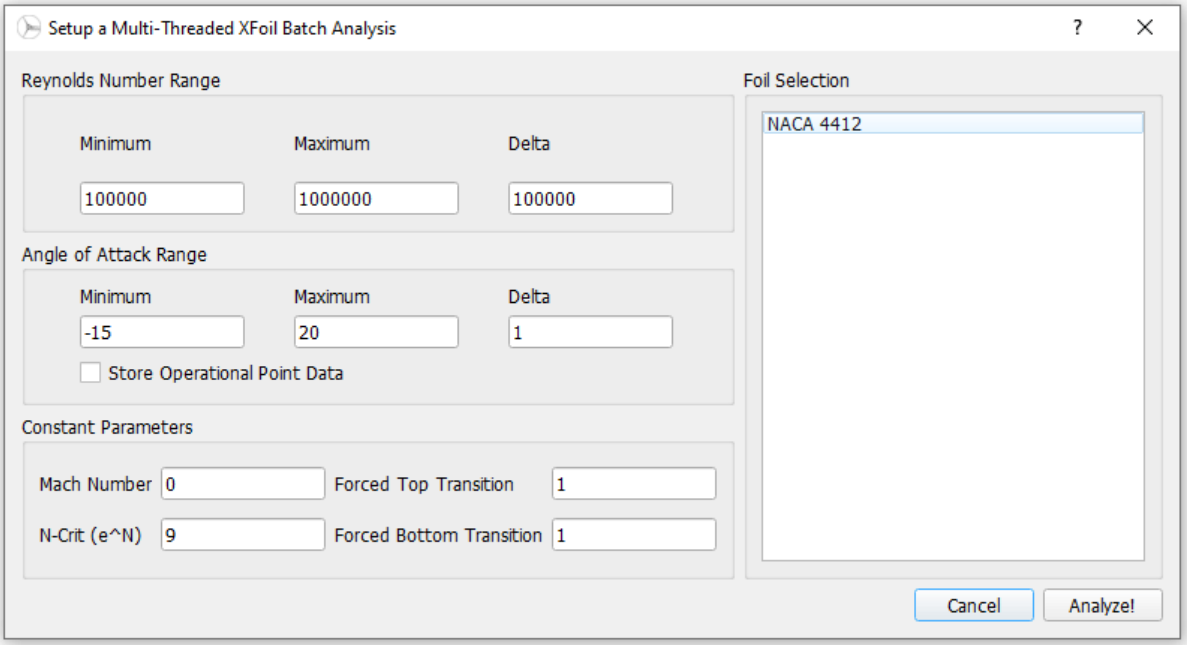


Fig. 55 *The XFoil batch creation dialogue.*

The parameter options are as described above and the batch calculation is executed by clicking on the *Analyze!* button.

Operational Point Analysis

Open completion of the XFoil analysis, a detailed aerodynamic description of the flow over the airfoil at each of the selected operational points (OpPoint) is available. These parameters can be conveniently viewed in the graphics interface. Three options are available for data visualization:

- **Polar Graph:** Shows changes of global aerodynamic parameters for each OpPoint.
- **OpPoint Graph:** Shows local aerodynamic quantities as a function of the position on the airfoil.
- **Aifoil visualization:** Provide a visualization of flow features superimposed onto the airfoil profile.

An example output for an airfoil is given in Fig. 56.

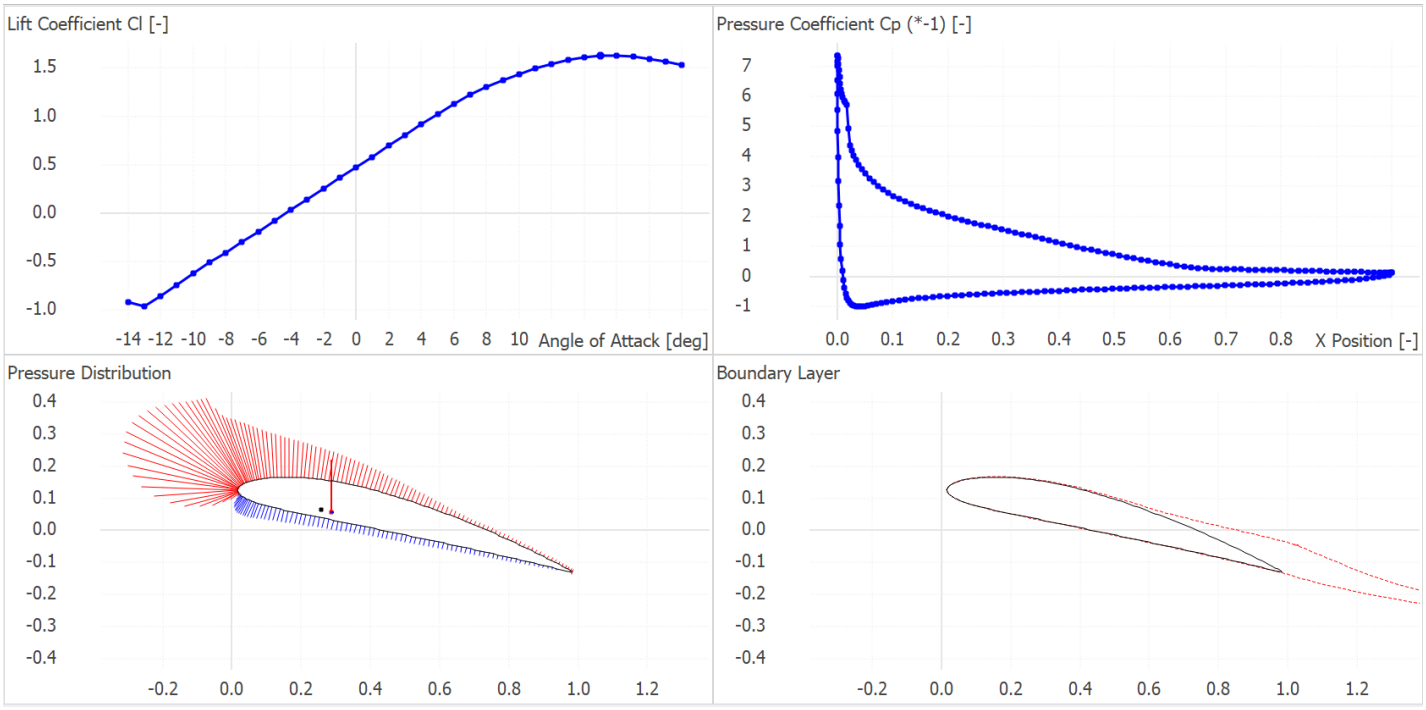


Fig. 56 *Operational point data from an XFoil Analysis. Top left: Polar plot. Top right: OpPoint plot. Bottom plots: Airfoil visualizations.*

Importing Polar Data

Airfoil aerodynamic data can also be imported within the airfoil analysis module by selecting this option in the *Polar* menu.

- **Plain Text**: The file needs to contain somewhere in the body an array with at least three columns containing: [α, C_L, C_D, (C_M)].
- **XFOIL**: This is a filetype generated by the XFoil solver which contains numerous additional aerodynamic parameters for the airfoil.
- **NREL (Aerodyn v.13)**: The Aerodyn v.13 `.dat` format.
- **QBlade**: The polar file `.plr` format of QBlade (see Import and Export of 360 Polars).

It should again be emphasized that polars for the entire α range are required for an analysis, as such polar import is more practical within the Polar Extrapolation Overview.

Exporting Polar Data

Airfoil polar data generated within the airfoil creation module can be exported for each airfoil either as an XFoil file or as an NREL (Aerodyn file simply be selecting the *Export Data* option from the *Polar* menu. The option is also available to export all generated airfoil data by selecting *Export ALL*.

[1] M. Drela. Xfoil: an analysis and design system for low reynolds number airfoils. In *Conference on Reynolds Number Airfoil Aerodynamics*. Notre Dame, Ind (USA), 1989.

[2] M. Drela and M.B. Giles. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA*, 1987.

Airfoil Analysis Overview

Once the desired blade section profiles have been selected in the Airfoil Generation Overview, the aerodynamic data for these profiles must be generated. This is accomplished within the airfoil creation module. This module is shown in the main toolbar in Fig. 53.

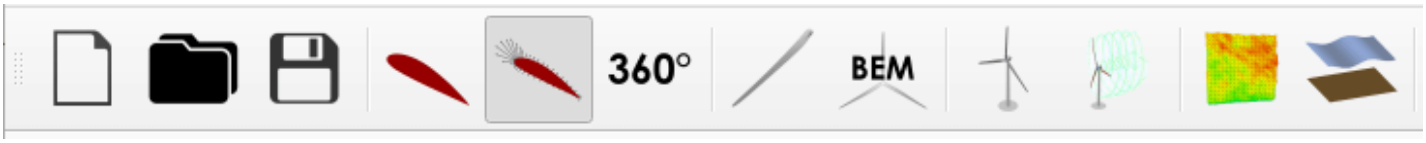


Fig. 53 *The airfoil analysis module is represented by the dynamic foil symbol in the QBlade main tool bar.*

The aerodynamic models within QBlade calculate local aerodynamic properties using airfoil look-up tables. These tables store the relevant aerodynamic quantities such as airfoil lift, drag and moment coefficients as a function of angle of attack α. There are two ways to generate these tables in QBlade. These two options are described below.

Generally the method used to generate the airfoil polars, whether experimental or numerical, only delivers reliable or repeatable values within a certain α range. For this reason, the polars generated or imported here may only correspond to a disjoint range: $\alpha \in [-180^o, 180^o]$. Extrapolating this range to ensure continuity of aerodynamic coefficients is the topic of the Polar Extrapolation Overview page.

Carrying out an XFoil Analysis

It is possible for the user to carry out an analysis using the 2D airfoil solver XFoil [1] directly within QBlade. This solver is linked to QBlade such that no preprocessing is required and the airfoil coordinates generated in the Airfoil Generation Overview are automatically prepared for analysis. An XFoil analysis can be carried out by generating a polar definition by selecting a new analysis from the polar control panel. The dialogue which then appears is shown in Fig. 54.

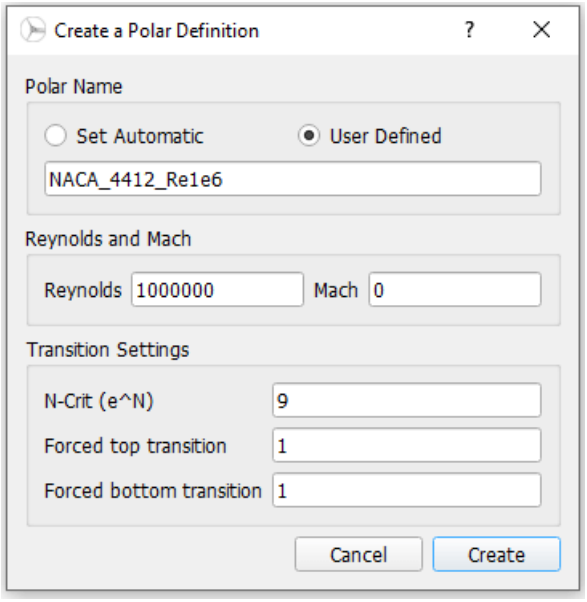


Fig. 54 *The XFoil polar creation dialogue.*

This requires the input of a range of parameters:

- **Polar Name**: This is the name stored for the generated polar.
- **Reynolds**: The Reynolds number of the analysis dictates the Reynolds number at which the airfoil is operating.
- **Mach**: The Mach number of the analysis. XFoil is also capable of treating transonic flow. For most wind energy applications however the flow around the airfoil is assumed to be incompressible.
- **N-Crit**: This is a parameter of the e^n model XFoil uses to predict when the boundary layer over the airfoil freely transitions from laminar to turbulent flow. [2]
- **Forced top transition**: If the e^n model is to be ignored on the suction side of the airfoil, this parameter gives the position of boundary layer transition (as a fraction of chord length).
- **Forced bottom transition**: If the e^n model is to be ignored on the pressure side of the airfoil, this parameter gives the position of boundary layer transition (as a fraction of chord length).

A range of advanced XFoil parameter settings can also be found in the *Polar* menu option under *XFoil Parameter Settings*. Once a polar object has been created, the operational points for the analysis can be selected. This is accomplished in the *Analysis Settings* module widget. The user must specify the start, end and delta values for α. Subsequently, the analysis is executed by clicking on the *Start Analysis* button. A progress bar will display the state of completion of the analysis.

XFoil Batch Analysis

For any given blade design, the different airfoil sections of a blade will be exposed to a range of operating conditions. For this reason, it can be beneficial to carry out an airfoil analysis for a range of Reynolds numbers for each of the airfoils that is used in the particular blade design. To streamline this process, QBlade features an XFoil batch option. The creation dialogue for this is shown in Fig. 55.

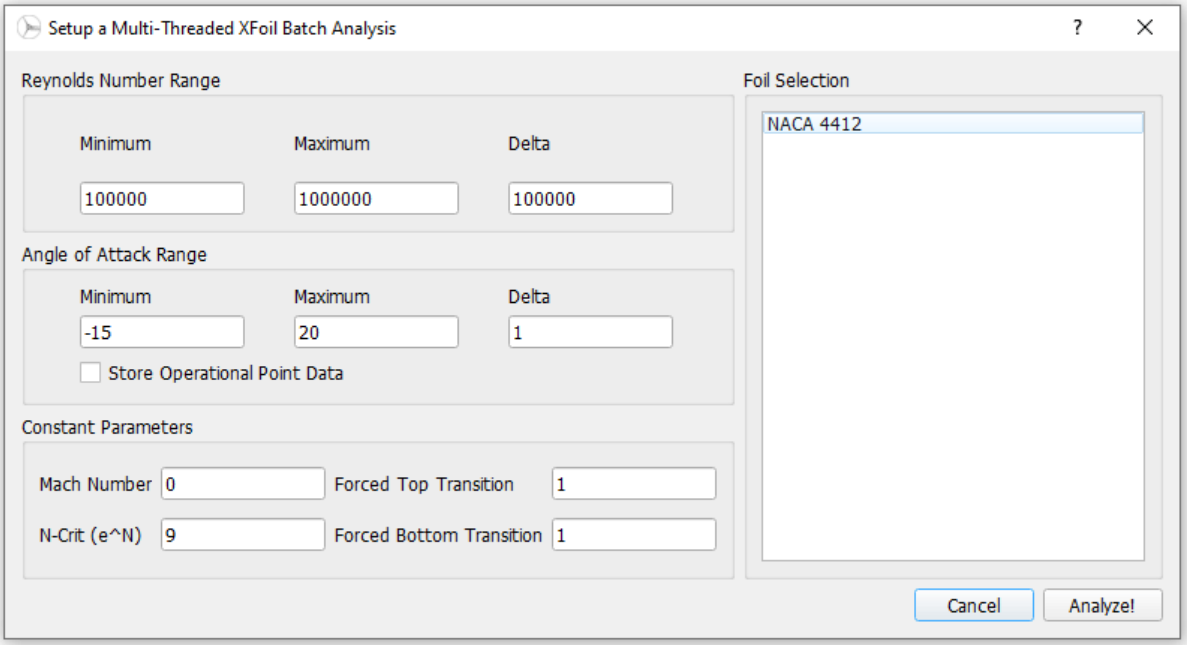


Fig. 55 *The XFoil batch creation dialogue.*

The parameter options are as described above and the batch calculation is executed by clicking on the *Analyze!* button.

Operational Point Analysis

Open completion of the XFoil analysis, a detailed aerodynamic description of the flow over the airfoil at each of the selected operational points (OpPoint) is available. These parameters can be conveniently viewed in the graphics interface. Three options are available for data visualization:

- **Polar Graph:** Shows changes of global aerodynamic parameters for each OpPoint.
- **OpPoint Graph:** Shows local aerodynamic quantities as a function of the position on the airfoil.
- **Aifoil visualization:** Provide a visualization of flow features superimposed onto the airfoil profile.

An example output for an airfoil is given in Fig. 56.

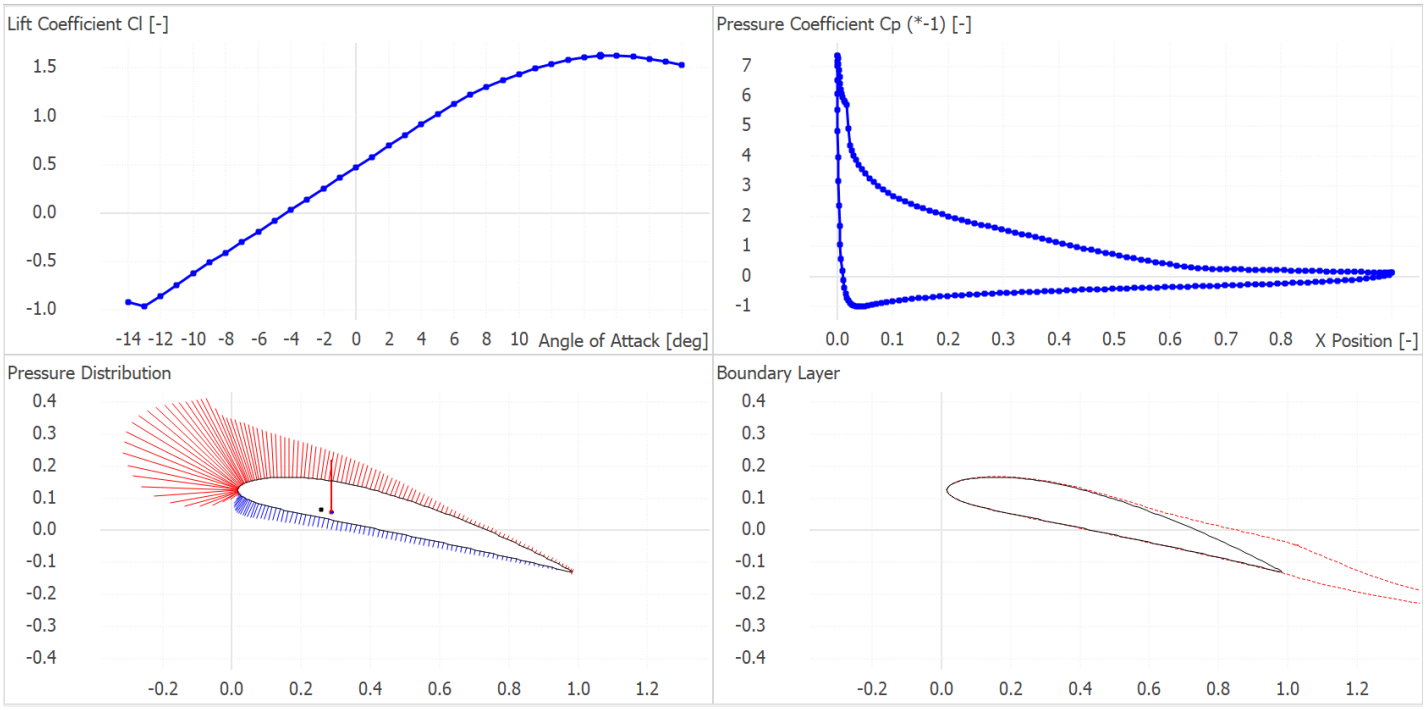


Fig. 56 *Operational point data from an XFoil Analysis. Top left: Polar plot. Top right: OpPoint plot. Bottom plots: Airfoil visualizations.*

Importing Polar Data

Airfoil aerodynamic data can also be imported within the airfoil analysis module by selecting this option in the *Polar* menu.

- **Plain Text**: The file needs to contain somewhere in the body an array with at least three columns containing: [α, C_L, C_D, (C_M)].
- **XFOIL**: This is a filetype generated by the XFoil solver which contains numerous additional aerodynamic parameters for the airfoil.
- **NREL (Aerodyn v.13)**: The Aerodyn v.13 `.dat` format.
- **QBlade**: The polar file `.plr` format of QBlade (see Import and Export of 360 Polars).

It should again be emphasized that polars for the entire α range are required for an analysis, as such polar import is more practical within the Polar Extrapolation Overview.

Exporting Polar Data

Airfoil polar data generated within the airfoil creation module can be exported for each airfoil either as an XFoil file or as an NREL (Aerodyn file simply be selecting the *Export Data* option from the *Polar* menu. The option is also available to export all generated airfoil data by selecting *Export ALL*.

[1] M. Drela. Xfoil: an analysis and design system for low reynolds number airfoils. In *Conference on Reynolds Number Airfoil Aerodynamics*. Notre Dame, Ind (USA), 1989.

[2] M. Drela and M.B. Giles. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA*, 1987.

Polar Extrapolation Overview

The numerical models used by QBlade to calculate aerodynamic properties and forces require knowledge of the airfoil sectional properties. It is often the case that the polar data for an airfoil (which has previously been either defined or imported- see Airfoil Generation Overview) is only available for a certain range of angles of attack α, these we shall refer to as *partial polars*. It often occurs that as a result of the turbine architecture, geometry, operational state or other factors that α values outside of this range are experienced by the airfoils of the turbine blade. For this reason, the blade creation module of QBlade requires that polars are defined for the full 360 ^{o} α range. It is therefore practical to have a methodology for extrapolating the partial polars. This is possible within QBlade using the polar extrapolation module. The module button in the main toolbar for this is shown in Fig. 57.

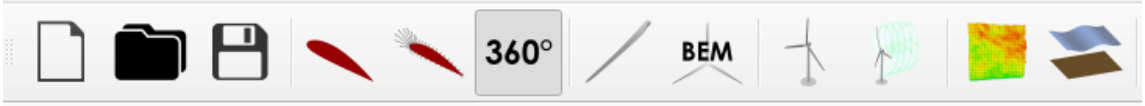


Fig. 57 *The polar extrapolation module is represented by the 360 ^{o} symbol in the QBlade main tool bar.*

A range of options for generating or defining 360 ^{o} polars are available in the polar extrapolation module. These are described in the following sections.

Viterna Extrapolation

The first option available to extrapolate airfoil polar data is the Viterna method. [1]. The generated extrapolation is visualized in the graph section to allow for optimization. The following parameters can be tuned to improve the behavior of the polar:

- **Range of original polar:** This determine which α range of the polar is used for the interpolation.
- **CD90:** Specifies the value of the drag coefficient at $\alpha = 90^{o}$. This also influences the lift coefficient behavior.
- **St+, St-:** These specify the positive and negative stall α for the airfoil, respectively.

Upon generation the tuning parameters are specified such that they correspond to those recommended in Viterna [1]. Once the polar is found to be suitable, this can be stored by clicking on the *Save* button. A full visualization of an airfoil extrapolation with the Viterna method is shown in Fig. 58.

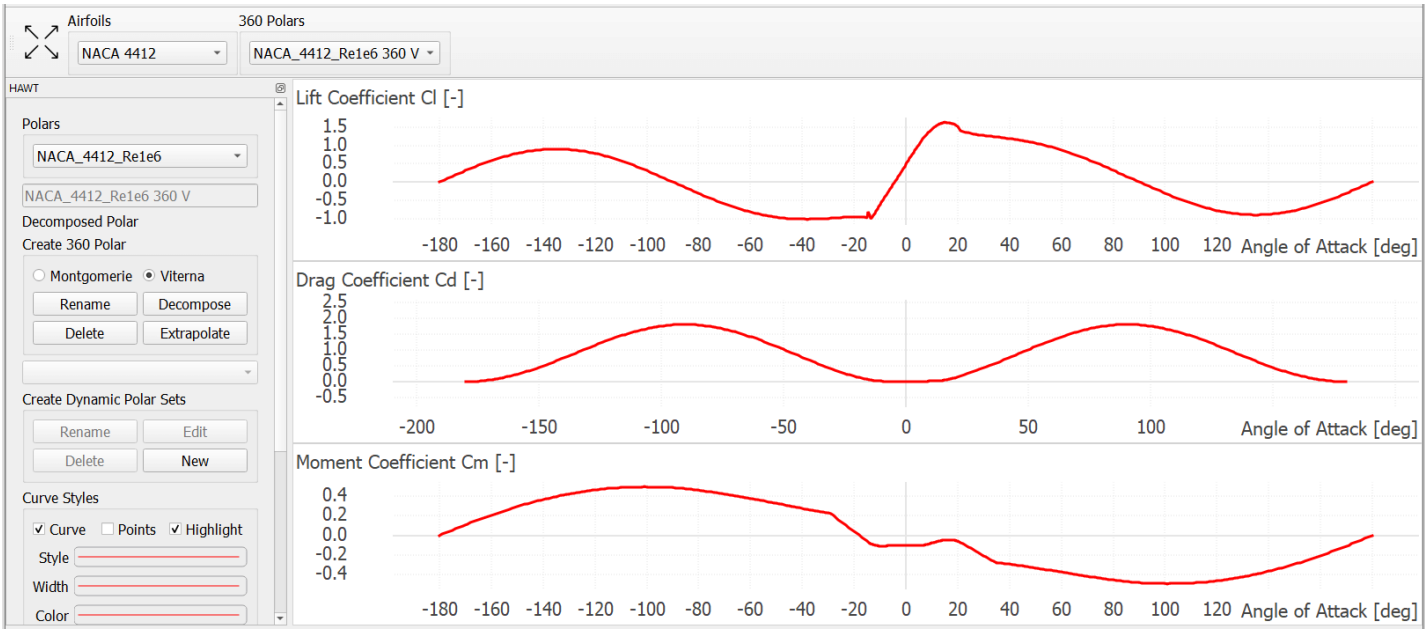


Fig. 58 *An airfoil extrapolation carried out using the Viterna method in QBlade.*

Montgomery Extrapolation

The second option available to extrapolate airfoil polar data is the Montgomery method [2]. This method is based on the assumption that the airfoil acts aerodynamically as a flat plat for high values of α. The generated extrapolation is visualized in the graph section to allow for optimization. The following parameters can be tuned to improve the behavior of the polar:

- **A+/B+:** These are curve parameters corresponding to the positive α range.
- **A-/B-:** These are curve parameters corresponding to the negative α range.
- **Slope:** This is a matching parameter which ensures continuity of the curve slope.
- **CD90:** Specifies the value of the drag coefficient at $\alpha = 90^o$. This also influences the lift coefficient behavior.

Upon generation the tuning parameters are specified such that they correspond to those recommended in Montgomerie [2]. Once the polar is found to be suitable, this can be stored by clicking on the *Save* button.

Polar Decomposition

In the case that the ATEFlap Model dynamic stall model is to be applied for a simulation, then a decomposition of the airfoil must be carried out. This separates the the airfoil coefficients into fully attached and fully separated regimes, which are applied together with kinematic data to calculate the unsteady lift, drag or moment coefficients. In QBlade this decomposition is automatically performed during the polar extrapolation - or can be applied as a post processing to already extrapolated polars. Once the extrapolation has been carried out, the parameters of the decomposition can be visualized. The parameters are include:

- **Attached Lift Coefficient:** The value of the lift coefficient for attached flows.
- **Detached Lift Coefficient:** The value of the lift coefficient for detached flows and dynamic airfoils near the detachment point.
- **f function:** The function which determines the fraction of the aforementioned lift contributions over the range of angles of attack.

$$Cl_{st} = fCl_{att} + (1 - f)Cl_{sep},$$

A plot of these parameters, as generated for a NACA 4412 profile are shown in Fig. 59.

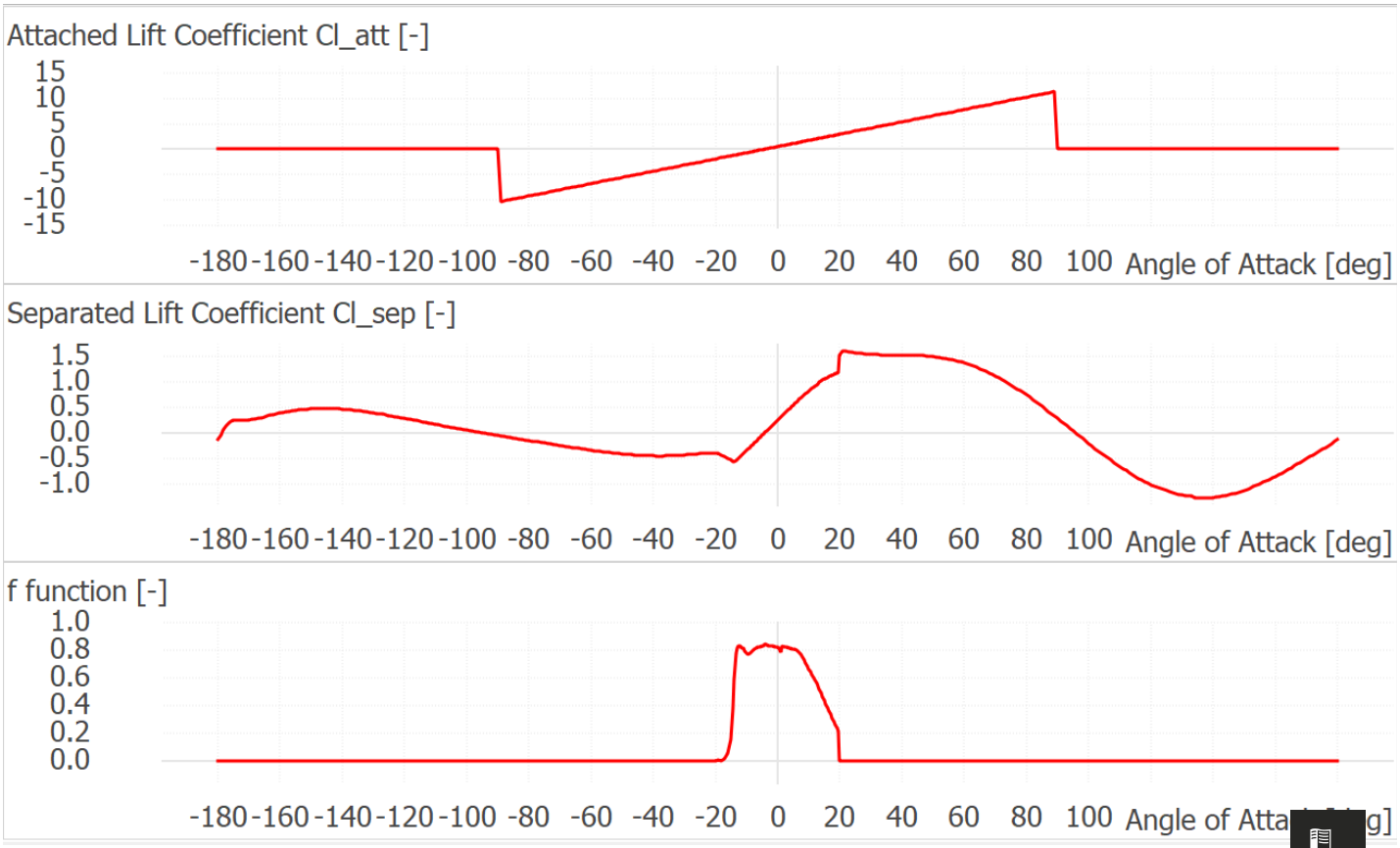


Fig. 59 *Dynamics airfoil parameters for a NACA 4412 polar decomposition carried out in QBlade.*

Dynamic Polar Sets

In QBlade dynamic polar sets can be used to model the changing states of flow control devices, such as trailing edge flaps. Dynamic polar sets allow to store polars that represents a series of states. For example: for a flap each state would correspond to a certain flap deflection angle. For each state multiple polars, covering a range of Reynolds numbers, may be stored. A dynamic polar set can then be assigned to an active element in the Blade Design Module (see Active Elements). The different states can then later be switched by Wind Turbine Controllers or the **Actuator Control Options Panel**, see Fig. 60.

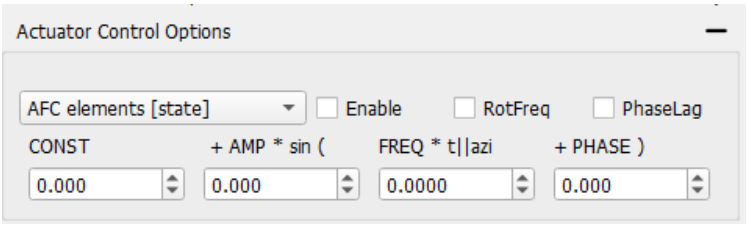


Fig. 60 *The actuator control panel, found in the Dock of the Simulation Module*

Fig. 61 shows the dialog to create dynamic polar sets.

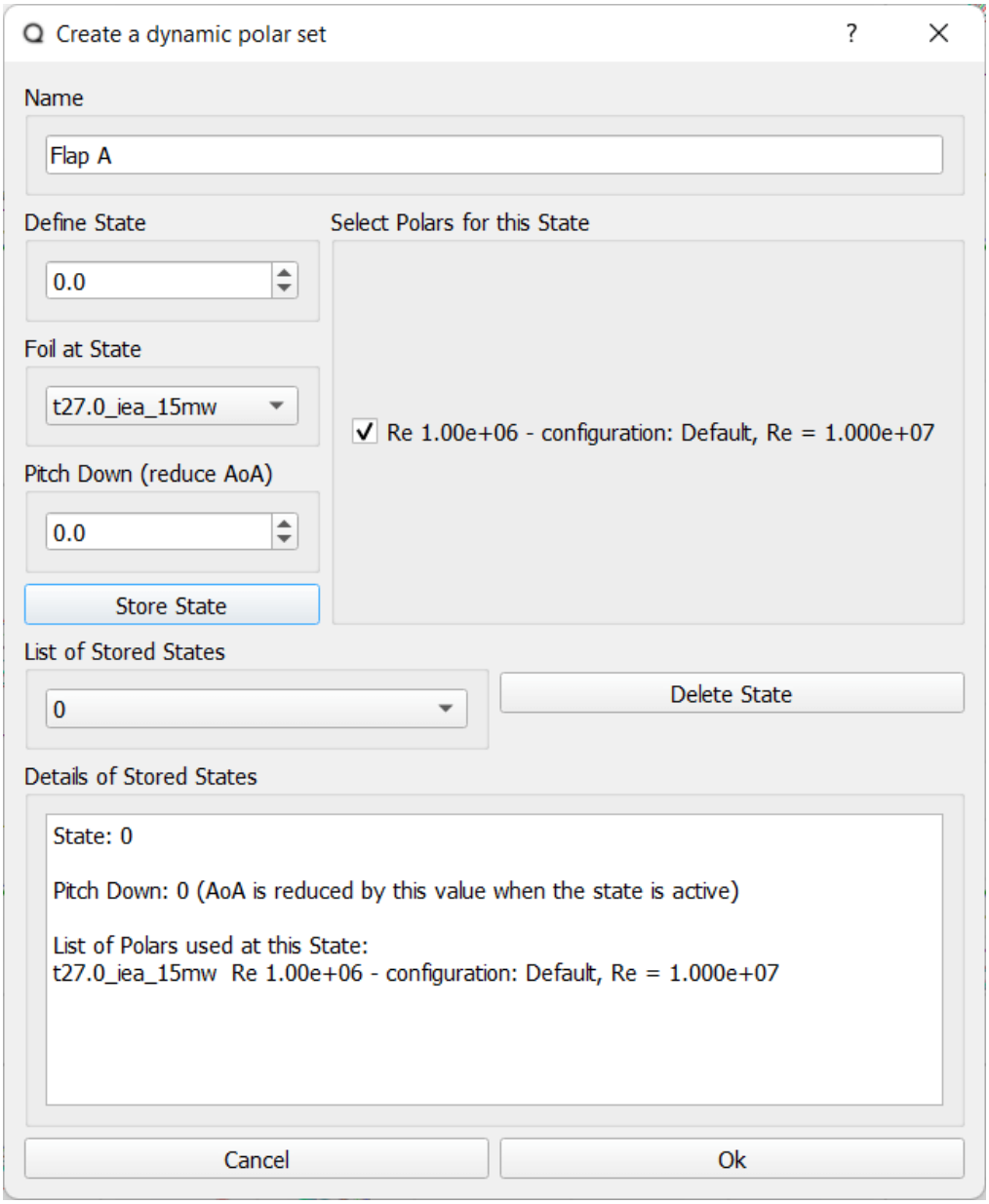


Fig. 61 *The dynamic polar set creator dialog.*

Import and Export of 360 Polars

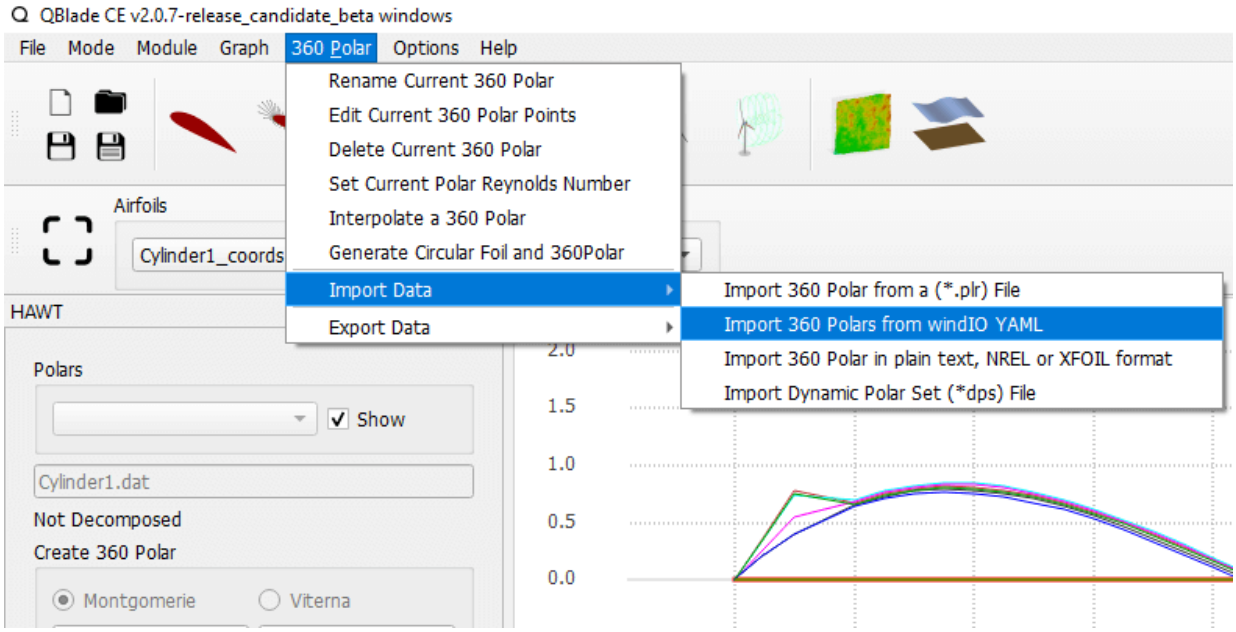


Fig. 62 *Import Options for 360 Polars*

QBlade allows to import and export 360 Polar objects in a series of formats. The import/export options are located in the menu item *360 Polar*, see Fig. 62. QBlade is currently capable of importing and exporting 360 Polars in the following formats:

- **Plain text format:** These include either AeroDyn V13 files or XFoil formats (see Airfoil Generation Overview)
- **Multi Re polar file:** The `.plr` format is the main format to import or export polar data. It can also be used to import polar data that is defined over a range of Reynolds numbers.
- **WindIO Yaml:** Imports all polars from a WindIO Yaml turbine definition file (import only)

When a `.plr` file is imported and no airfoil, as specified in the file, exists an airfoil with the thickness as defined with the parameter *THICKNESS* is automatically created during loading. An exemplary `.plr` file is shown below:

Listing 1 : An exemplary QBlade multi-RE polar file

```
-------------------------------------QBlade Multi RE Polar File-------------------------------------------------
Generated with : QBlade CE v 2.0 windows-pre-release
Archive Format: 310001
Time : 21:36:42
Date : 14.06.2022

-------------------------------------Object Names-----------------------------------------------------------------
t17.0_nre_5mw_Polar                    POLARNAME          - the polar name
t17.0_nre_5mw                          FOILNAME           - the airfoil name to which the polar(s) belong

-------------------------------------Parameters-------------------------------------------------------------------
17.0                                   THICKNESS          - the name of the blade
0                                      ISDECOMPOSED       - is the polar decomposed (add Cl_Sep, Cl_att and f_st columns)
REYNOLDS           1.0000E+06         - the list of Reynolds numbers for the imported polars

-------------------------------------Polar Data-------------------------------------------------------------------
AOA                CL                 CD                 CM
-180.000000        0.000000           0.019800           0.000000
-175.000000        0.374000           0.034100           0.188000
-170.000000        0.749000           0.095500           0.377000
-160.000000        0.659000           0.280700           0.274700
-155.000000        0.736000           0.391900           0.313000
-150.000000        0.783000           0.508600           0.342800
-145.000000        0.803000           0.626700           0.365400
-140.000000        0.798000           0.742700           0.382000
-135.000000        0.771000           0.853700           0.393500
```

-130.000000	0.724000	0.957400	0.400700
-125.000000	0.660000	1.051900	0.404200
-120.000000	0.581000	1.135500	0.404700
-115.000000	0.491000	1.207000	0.402500
-110.000000	0.390000	1.265600	0.398100
-105.000000	0.282000	1.310400	0.391800
-100.000000	0.169000	1.341000	0.383800
-95.000000	0.052000	1.357200	0.374300
-90.000000	-0.067000	1.358700	0.363600
-85.000000	-0.184000	1.345600	0.351700
-80.000000	-0.299000	1.318100	0.338800
-75.000000	-0.409000	1.276500	0.324800
-70.000000	-0.512000	1.221200	0.309900
-65.000000	-0.606000	1.153200	0.294000
-60.000000	-0.689000	1.073100	0.277200
-55.000000	-0.759000	0.982200	0.259500
-50.000000	-0.814000	0.882000	0.240900
-45.000000	-0.850000	0.774200	0.221200
-40.000000	-0.866000	0.661000	0.200600
-35.000000	-0.860000	0.545100	0.178900
-30.000000	-0.829000	0.429500	0.156300
-25.000000	-0.853000	0.307100	0.115600
-24.000000	-0.870000	0.281400	0.104000
-23.000000	-0.890000	0.255600	0.091600
-22.000000	-0.911000	0.229700	0.078500
-21.000000	-0.934000	0.204000	0.064900
-20.000000	-0.958000	0.178500	0.050800
-19.000000	-0.982000	0.153400	0.036400
-18.000000	-1.005000	0.128800	0.021800
-17.000000	-1.082000	0.103700	0.012900
-16.000000	-1.113000	0.078600	-0.002800
-15.000000	-1.105000	0.053500	-0.025100
-14.000000	-1.078000	0.028300	-0.041900
-13.500000	-1.053000	0.015800	-0.052100
-13.000000	-1.015000	0.015100	-0.061000
-12.000000	-0.904000	0.013400	-0.070700
-11.000000	-0.807000	0.012100	-0.072200
-10.000000	-0.711000	0.011100	-0.073400
-9.000000	-0.595000	0.009900	-0.077200
-8.000000	-0.478000	0.009100	-0.080700
-7.000000	-0.375000	0.008600	-0.082500
-6.000000	-0.264000	0.008200	-0.083200
-5.000000	-0.151000	0.007900	-0.084100
-4.000000	-0.017000	0.007200	-0.086900
-3.000000	0.088000	0.006400	-0.091200
-2.000000	0.213000	0.005400	-0.094600
-1.000000	0.328000	0.005200	-0.097100
0.000000	0.442000	0.005200	-0.101400
1.000000	0.556000	0.005200	-0.107600
2.000000	0.670000	0.005300	-0.112600
3.000000	0.784000	0.005300	-0.115700
4.000000	0.898000	0.005400	-0.119900
5.000000	1.011000	0.005800	-0.124000
6.000000	1.103000	0.009100	-0.123400
7.000000	1.181000	0.011300	-0.118400
8.000000	1.257000	0.012400	-0.116300
8.500000	1.293000	0.013000	-0.116300
9.000000	1.326000	0.013600	-0.116000
9.500000	1.356000	0.014300	-0.115400
10.000000	1.382000	0.015000	-0.114900
10.500000	1.400000	0.026700	-0.114500
11.000000	1.415000	0.038300	-0.114300
11.500000	1.425000	0.049800	-0.114700
12.000000	1.434000	0.061300	-0.115800
12.500000	1.443000	0.072700	-0.116500
13.000000	1.451000	0.084100	-0.115300
13.500000	1.453000	0.095400	-0.113100
14.000000	1.448000	0.106500	-0.111200
14.500000	1.444000	0.117600	-0.110100
15.000000	1.445000	0.128700	-0.110300
15.500000	1.447000	0.139800	-0.110900
16.000000	1.448000	0.150900	-0.111400
16.500000	1.444000	0.161900	-0.111100
17.000000	1.438000	0.172800	-0.109700
17.500000	1.439000	0.183700	-0.107900
18.000000	1.448000	0.194700	-0.108000

```
18.500000          1.452000          0.205700          -0.109000
19.000000          1.448000          0.216500          -0.108600
19.500000          1.438000          0.227200          -0.107700
20.000000          1.428000          0.237900          -0.109900
21.000000          1.401000          0.259000          -0.116900
22.000000          1.359000          0.279900          -0.119000
23.000000          1.300000          0.300400          -0.123500
24.000000          1.220000          0.320400          -0.139300
25.000000          1.168000          0.337700          -0.144000
26.000000          1.116000          0.355400          -0.148600
28.000000          1.015000          0.391600          -0.157700
30.000000          0.926000          0.429400          -0.166800
32.000000          0.855000          0.469000          -0.175900
35.000000          0.800000          0.532400          -0.189700
40.000000          0.804000          0.645200          -0.212600
45.000000          0.793000          0.757300          -0.234400
50.000000          0.763000          0.866400          -0.255300
55.000000          0.717000          0.970800          -0.275100
60.000000          0.656000          1.069300          -0.293900
65.000000          0.582000          1.160600          -0.311700
70.000000          0.495000          1.243800          -0.328500
75.000000          0.398000          1.317800          -0.344400
80.000000          0.291000          1.380900          -0.359300
85.000000          0.176000          1.430400          -0.373100
90.000000          0.053000          1.456500          -0.385800
95.000000          -0.074000         1.453300          -0.397300
100.000000         -0.199000         1.434500          -0.407500
105.000000         -0.321000         1.400400          -0.416200
110.000000         -0.436000         1.351200          -0.423100
115.000000         -0.543000         1.287400          -0.428000
120.000000         -0.640000         1.209900          -0.430600
125.000000         -0.723000         1.119600          -0.430400
130.000000         -0.790000         1.017900          -0.427000
135.000000         -0.840000         0.906400          -0.419600
140.000000         -0.868000         0.787100          -0.407700
145.000000         -0.872000         0.662700          -0.390300
150.000000         -0.850000         0.536300          -0.366500
155.000000         -0.798000         0.411600          -0.334900
160.000000         -0.714000         0.293100          -0.294200
170.000000         -0.749000         0.097100          -0.377100
175.000000         -0.374000         0.033400          -0.187900
180.000000         0.000000          0.019800          0.000000
```


[1] (1, 2) L. A. Viterna and D. C. Janetzke. Theoretical and Experimental Power From Large Horizontal-Axis Wind Turbines. Technical Report, NASA, 1982. doi:10.2172/6763041.

[2] (1, 2) B Montgomerie. Methods for Root Effects, Tip Effects and Extending the Angle of Attack Range to 6180, With Application to Aerodynamics for Blades on Wind Turbines and Propellers. Technical Report, FOI, Swedish Defence Research Agency, 2004.

Blade Design Overview

Once the airfoil polars have been created or imported (see Airfoil Analysis Overview and Polar Extrapolation Overview), they can be used in an aerodynan module in QBlade, click on the blade symbol in the main toolbar, as shown in Fig. 63.

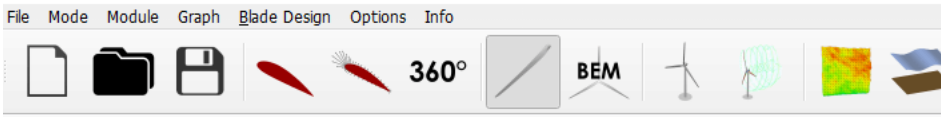


Fig. 63 *The aerodynamic blade design module is represented by the blade symbol in the QBlade main to*

An overview of the blade design module is shown in Fig. 64. The module is split into three main parts. On the left side, the controls dock allows the user t The latter is located in the middle of the module and allows the user have a interactive 3D representation of the current blade design. The right side of th aerodynamic quantities along the blade. The user can interact with these graphs in the same manner as with all the other graphs within QBlade (see GUI

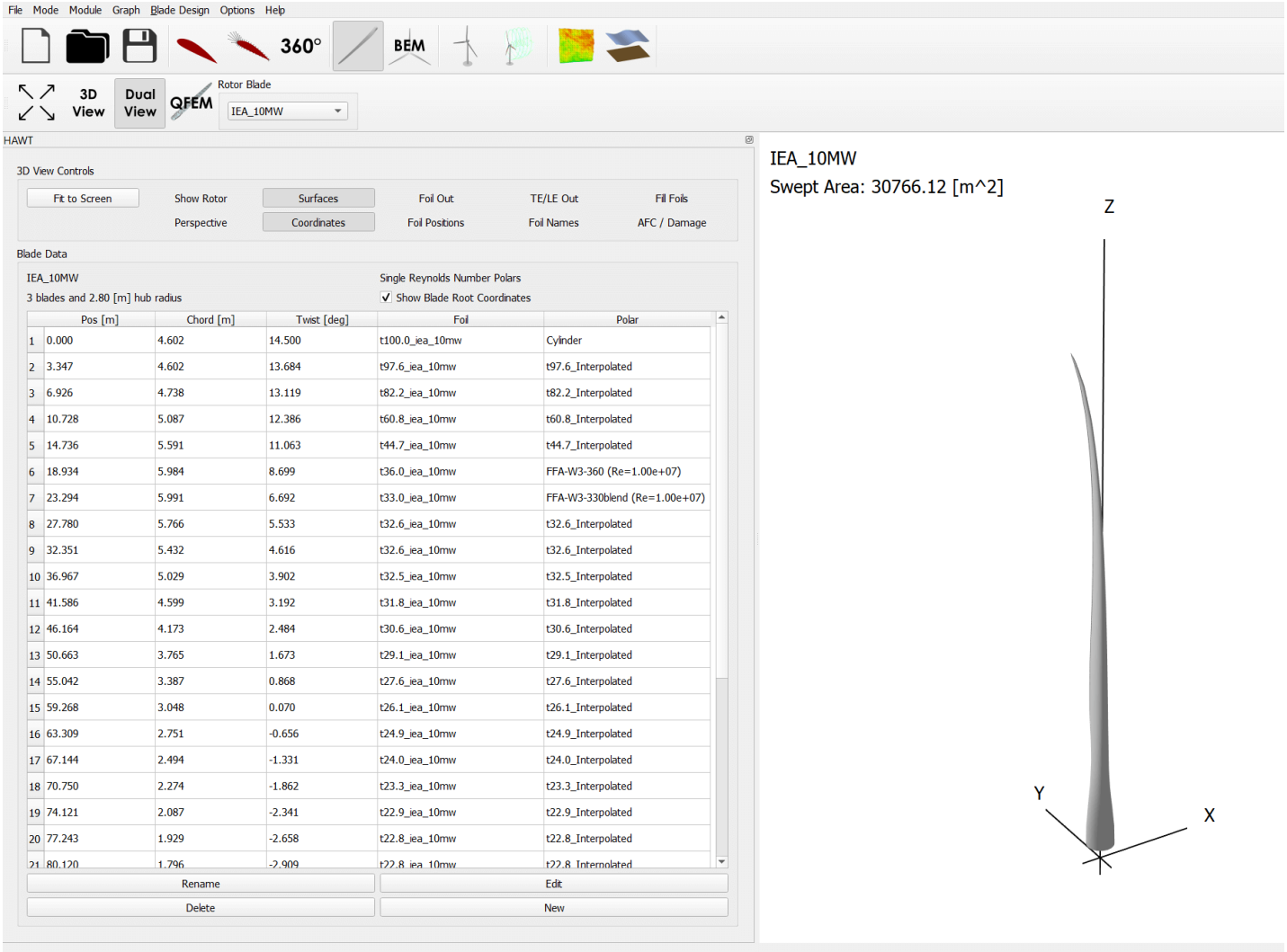


Fig. 64 *Overview of the blade design module in QBlade. The control dock is located on the left, the interactive 3D view in the cen*

Basic Blade Design

In QBlade, the aerodynamics of a blade are defined by splitting the blade into several stations. This is shown in Fig. 65. The aerodynamic parameters are d these values between stations (i.e. over a blade element).

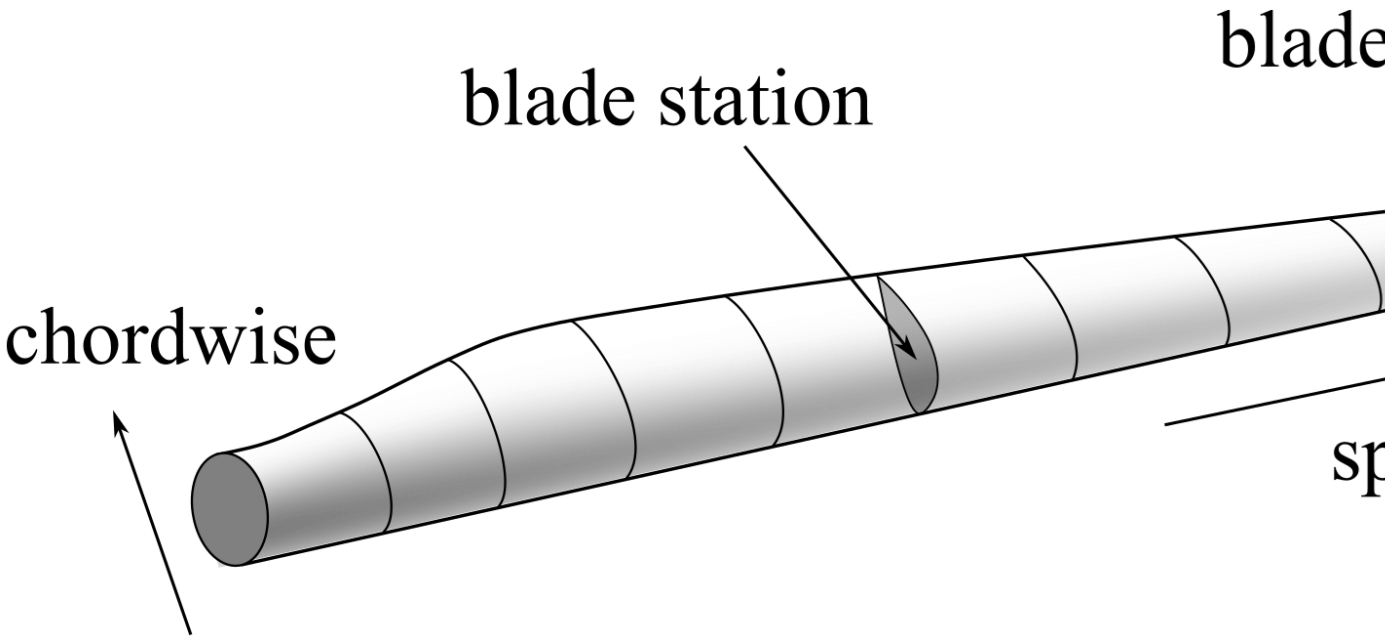


Fig. 65 *Aerodynamic blade definition in QBlade is done in stations. Values are linearly interpolated along the elements b*

Each blade station is defined by a series of aerodynamic properties. The global blade/rotor parameters are the number of blades and the hub radius. The c aerodynamic blade properties. The columns of the basic blade properties are:

- **Pos [m]** is the position of the station along the blade pitch axes (in m). It can be given in blade root coordinates or in hub coordinates.
- **Chord [m]** is the local chord length of the blade station (in m).
- **Twist [deg]** the local twist of the blade station (in deg).
- **Foil** is the airfoil object used for that station (see Airfoil Generation Overview).
- **Polar** is the particular polar used for this station (linked to the airfoil object).

In addition to the manual blade definition option, QBlade allows for some automated setups to speed up the blade design. It has the option to automatica the airfoil. It also offers the option to do a blade shape optimization so that the twist angle is optimal for a given tip speed ratio. In addition the chord dist and Schmitz (see Gasch and Twele[1] for details). Finally, the blade design can also be scaled to another size using different scaling methods. These include

Multi Polar Blade Definition

The extrapolated airfoil polar data used in the blade definition is always linked to an airfoil object. Several different airfoil polars can be linked to one airfo can also contain multiple polars for one airfoil object e.g. for different Reynolds numbers. If the blade shall have a multi-polar definition, then these need t the corresponding option has to be enabled in the blade design dock (see Fig. 66).

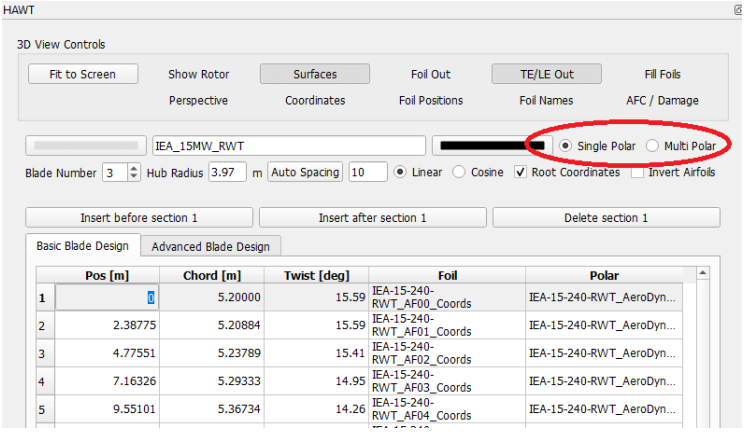


Fig. 66 *The multi Reynolds number polar option in the blade designer dock.*

Advanced Blade Design

The advanced blade design table can be accessed through the *Advanced Blade Design Tabl*, as shown in Fig. 67.

The columns of the advanced blade property table are:

- **Position [m]** is the position of the station along the blade pitch axis (in m). It should match the position given in the basic blade properties.
- **X (IP) Offset [m]** is an additional offset of the blade station in the global y-direction. This is the in-plane direction.
- **Y (OOP) Offset [m]** is an additional offset of the blade station in the global x-direction. This is the out-of-plane direction.
- **T Axis [%c]** is the position of the thread axis as a percentage of the local chord. It is used to define the axis at which the station is rotated and also to d
  Coordinate Systems).

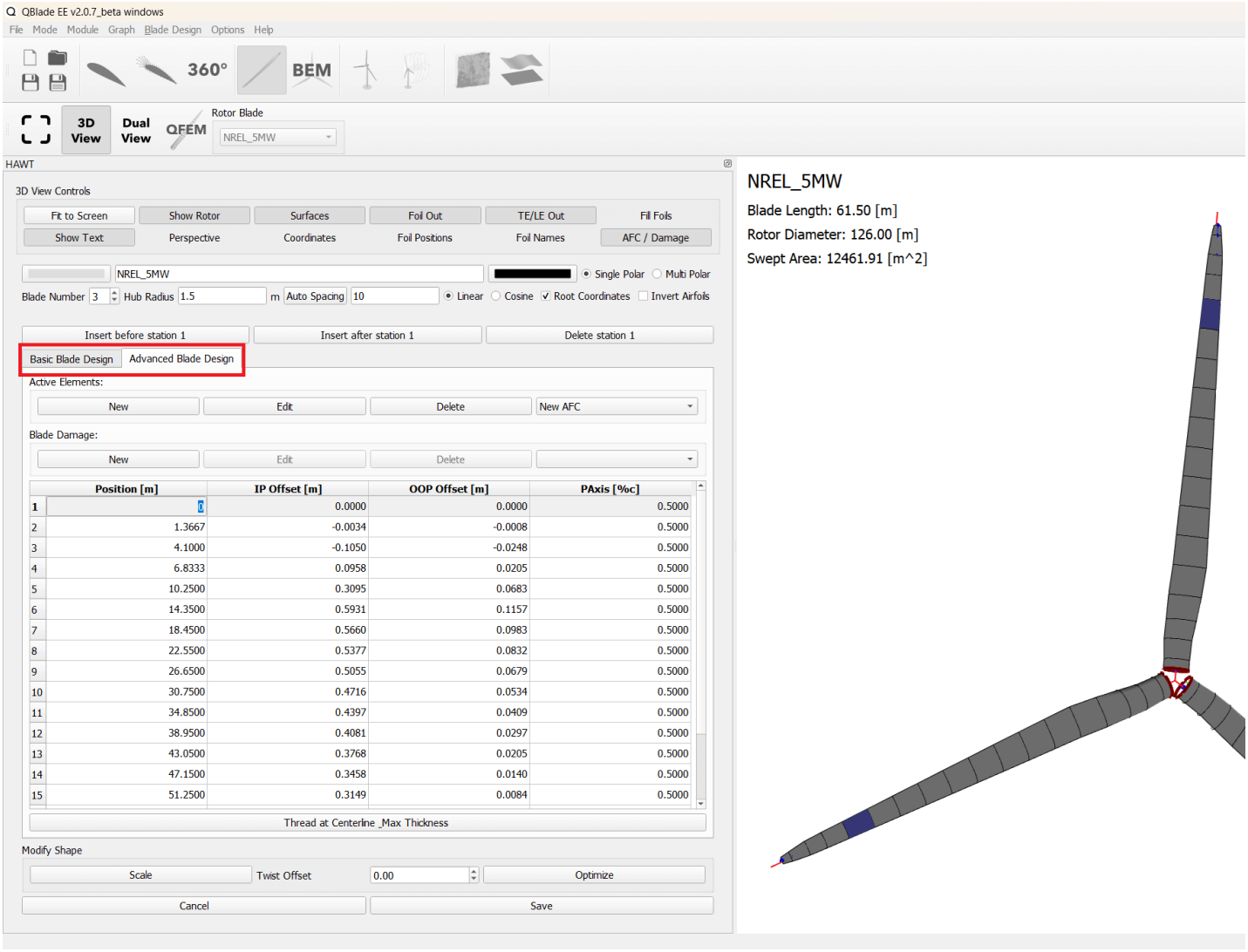


Fig. 67 *The Advanced Blade Design tab*

Active Elements

QBlade can model and add active elements, such as active trailing edge flaps, to a blade definition. This is done in the advanced blade design tab, as show
one ore more Dynamic Polar Sets must have been previously defined in the Polar Extrapolation Module (see: Polar Extrapolation Overview). Each active e
dynamic polar set can be chosen at each blade station. Linear interpolation is used between the two Dynamic Polar Sets. An active element is always app

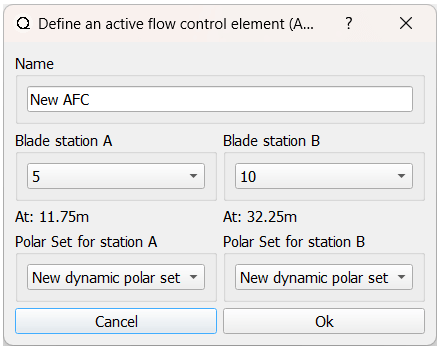


Fig. 68 *The active element dialog.*

Blade Damage

A blade damage can be added to a blade definition. This feature is intended to model *damaged* blade elements (for instance leading edge erosion effects) t
blade stations of individual blades. The airfoil polar for the damaged blade station must be created previously in the Polar Extrapolation Overview. Similar
two stations. These can have different airfoils and polars, even multi-polar sets. QBlade will interpolate along the blade between the two stations with the
damage can be assigned to an individual blade and can thus be used to model an aerodynamic imbalance.

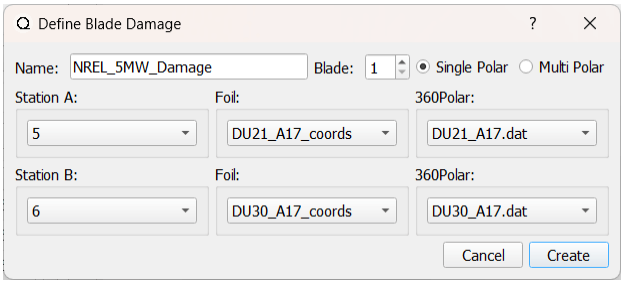


Fig. 69 *The blade damage dialog.*

Importing and Exporting Blade Definitions

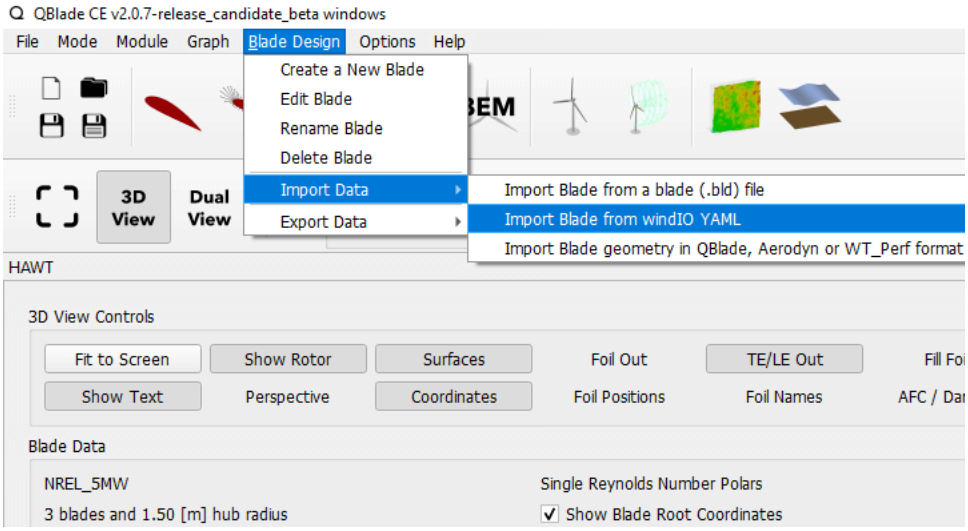


Fig. 70 *Blade Definition Import Options*

QBlade allows to import and export blade definitions in a series of formats. The import/export options are located in the menu item *Blade Design*, see Fig.
in the following formats:

- QBlade blade format (`.bld`),
- Blade geometry in QBlade, AeroDyn and WT_perf format,
- AeroDyn V15 full blade definition,
- WindIO Yaml Definition

QBlade is currently capable of exporting blade definitions in the following formats:

- QBlade blade definition ASCII format (`.bld`),
- Full blade definition in AeroDyn V13 format,
- 3D blade geometry in STL or `.txt` format.

Blade definition ASCII File

When a blade is exported into the `.bld` format, the associated 360 polar (`.plr`) and airfoil (`.afl`) files are automatically created. An exemplary `.bld` fi

```
-------------------------------------QBlade Blade Definition File-------------------------------------------------
Generated with : QBlade IH v2.0.2_alpha windows
Archive Format: 310002
Time : 12:05:50
Date : 29.06.2022

-----------------------------------Object Name-------------------------------------------------------------------
NREL_5MW                              OBJECTNAME         - the name of the blade object
```

```
---------------------------------------Parameters-------------------------------------------------------------
HAWT                                   ROTORTYPE          - the rotor type
3                                      NUMBLADES          - number of blades

---------------------------------------Blade Data-------------------------------------------------------------
POS [m]             CHORD [m]          TWIST [deg]        OFFSET_X [m]       OFFSET_Y [m]       TAXIS [-]          POLAR_FILE
1.5000              3.5420             0.0000             0.0000             0.0000             0.5000             t100.0_nre_5mw_Cylinder_1_s
2.8674              3.5420             13.3080            0.0027             0.0006             0.5000             t100.0_nre_5mw_Cylinder_1_s
5.5992              3.8540             13.3080            0.1057             0.0250             0.5000             t100.0_nre_5mw_Cylinder_1_s
8.3289              4.1670             13.3080            0.2499             0.0591             0.5000             t90.0_nre_5mw_Cylinder_2_se
11.7402             4.5570             13.3080            0.4586             0.1085             0.5000             t40.0_nre_5mw_DU40_airfoil_
15.8399             4.6520             11.4845            0.5696             0.1157             0.5000             t35.0_nre_5mw_DU35_airfoil_
19.9410             4.4580             10.1649            0.5485             0.0983             0.5000             t35.0_nre_5mw_DU35_airfoil_
24.0421             4.2490             9.0132             0.5246             0.0832             0.5000             t30.0_nre_5mw_DU30_airfoil_
28.1432             4.0070             7.7970             0.4962             0.0679             0.5000             t25.0_nre_5mw_DU25_airfoil_
32.2443             3.7480             6.5457             0.4654             0.0534             0.5000             t25.0_nre_5mw_DU25_airfoil_
36.3454             3.5020             5.3623             0.4358             0.0409             0.5000             t21.0_nre_5mw_DU21_airfoil_
40.4464             3.2560             4.1890             0.4059             0.0297             0.5000             t21.0_nre_5mw_DU21_airfoil_
44.5475             3.0100             3.1256             0.3757             0.0205             0.5000             t17.0_nre_5mw_NA64_A17_airf
48.6486             2.7640             2.3193             0.3452             0.0140             0.5000             t17.0_nre_5mw_NA64_A17_airf
52.7497             2.5180             1.5261             0.3146             0.0084             0.5000             t17.0_nre_5mw_NA64_A17_airf
56.1676             2.3130             0.8629             0.2891             0.0044             0.5000             t17.0_nre_5mw_NA64_A17_airf
58.9013             2.0860             0.3699             0.2607             0.0017             0.5000             t17.0_nre_5mw_NA64_A17_airf
61.6338             1.4190             0.1059             0.1774             0.0003             0.5000             t17.0_nre_5mw_NA64_A17_airf
63.0000             0.9610             0.0000             0.1201             0.0000             0.5000             t17.0_nre_5mw_NA64_A17_airf
```

[1] R. Gasch and J. Twele. *Windkraftanlagen: Grundlagen, Entwurf, Planung und Betrieb*. Vieweg+Teubner Verlag, 4th edtion edition, 2005. ISBN 978-3-322-99446-2

Blade Design Overview

Once the airfoil polars have been created or imported (see Airfoil Analysis Overview and Polar Extrapolation Overview), they can be used in an aerodynan module in QBlade, click on the blade symbol in the main toolbar, as shown in Fig. 63.

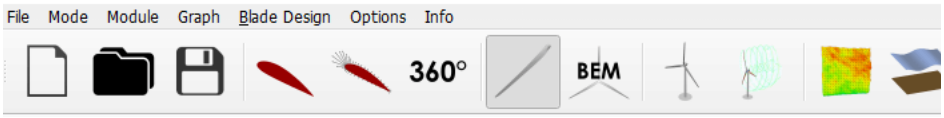


Fig. 63 *The aerodynamic blade design module is represented by the blade symbol in the QBlade main to*

An overview of the blade design module is shown in Fig. 64. The module is split into three main parts. On the left side, the controls dock allows the user t The latter is located in the middle of the module and allows the user have a interactive 3D representation of the current blade design. The right side of th aerodynamic quantities along the blade. The user can interact with these graphs in the same manner as with all the other graphs within QBlade (see GUI

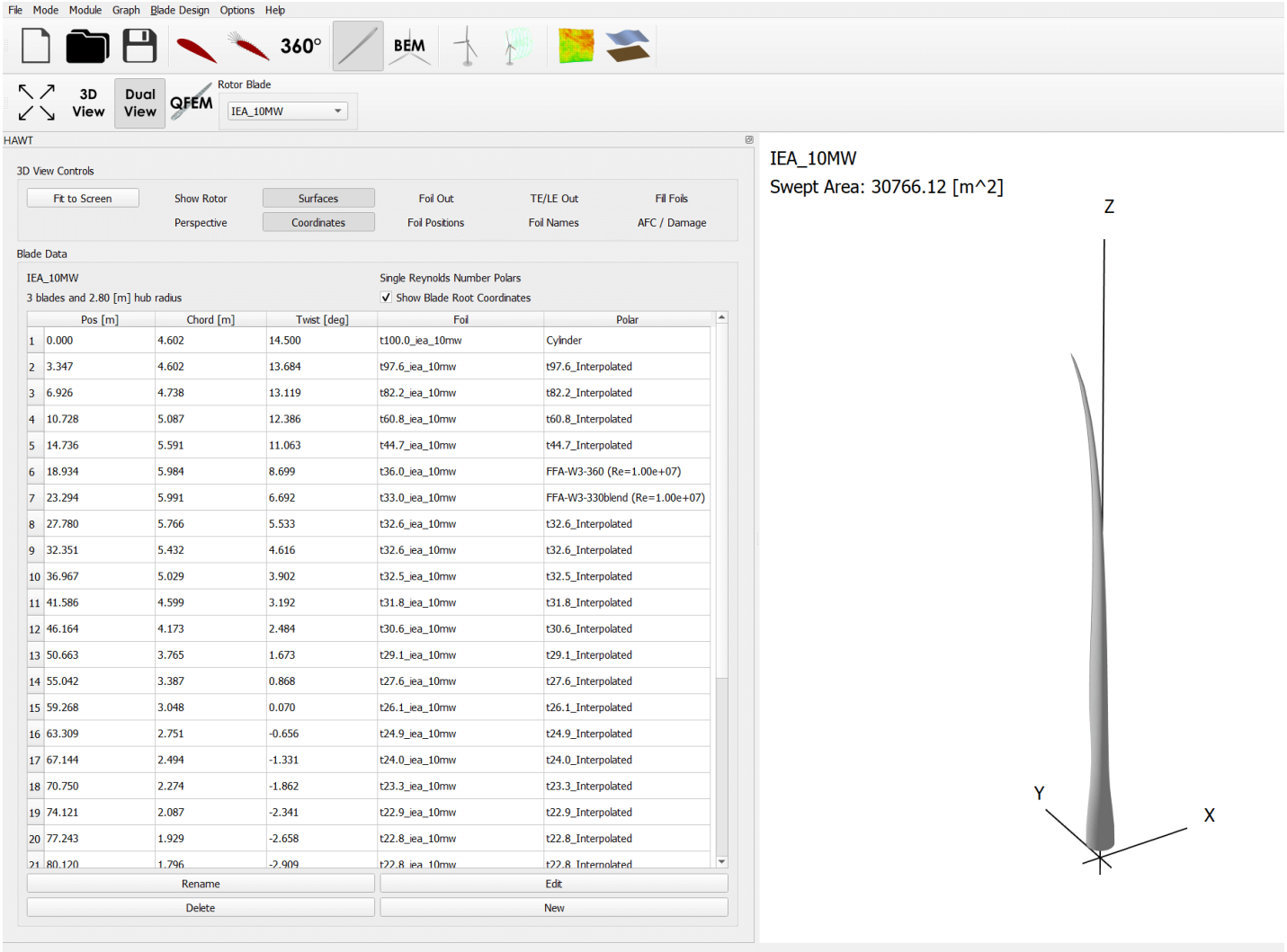


Fig. 64 *Overview of the blade design module in QBlade. The control dock is located on the left, the interactive 3D view in the cen*

Basic Blade Design

In QBlade, the aerodynamics of a blade are defined by splitting the blade into several stations. This is shown in Fig. 65. The aerodynamic parameters are d these values between stations (i.e. over a blade element).

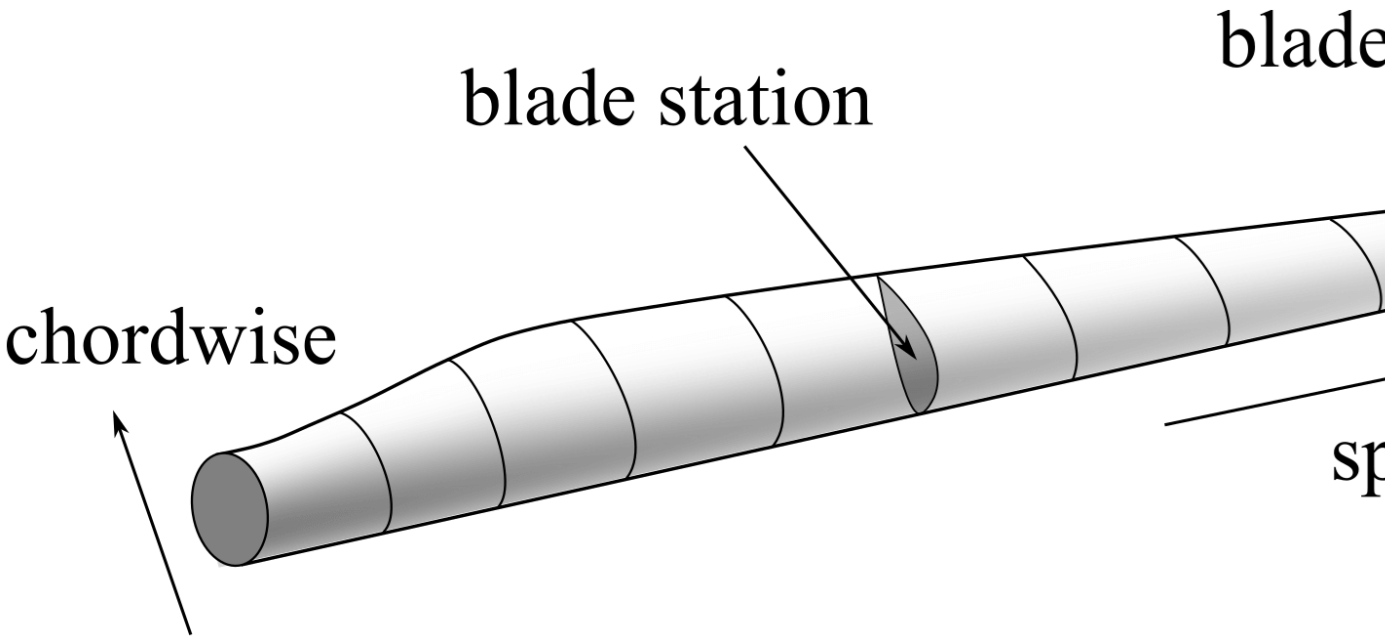


Fig. 65 *Aerodynamic blade definition in QBlade is done in stations. Values are linearly interpolated along the elements b*

Each blade station is defined by a series of aerodynamic properties. The global blade/rotor parameters are the number of blades and the hub radius. The c aerodynamic blade properties. The columns of the basic blade properties are:

- **Pos [m]** is the position of the station along the blade pitch axes (in m). It can be given in blade root coordinates or in hub coordinates.
- **Chord [m]** is the local chord length of the blade station (in m).
- **Twist [deg]** the local twist of the blade station (in deg).
- **Foil** is the airfoil object used for that station (see Airfoil Generation Overview).
- **Polar** is the particular polar used for this station (linked to the airfoil object).

In addition to the manual blade definition option, QBlade allows for some automated setups to speed up the blade design. It has the option to automatica the airfoil. It also offers the option to do a blade shape optimization so that the twist angle is optimal for a given tip speed ratio. In addition the chord dist and Schmitz (see Gasch and Twele[1] for details). Finally, the blade design can also be scaled to another size using different scaling methods. These include

Multi Polar Blade Definition

The extrapolated airfoil polar data used in the blade definition is always linked to an airfoil object. Several different airfoil polars can be linked to one airfo can also contain multiple polars for one airfoil object e.g. for different Reynolds numbers. If the blade shall have a multi-polar definition, then these need t the corresponding option has to be enabled in the blade design dock (see Fig. 66).

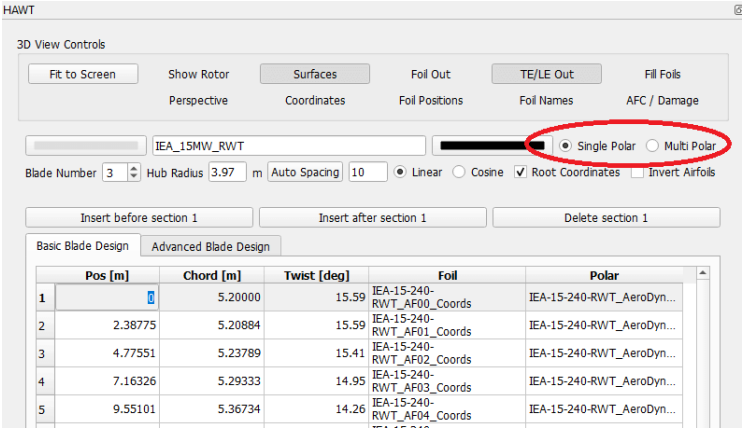


Fig. 66 *The multi Reynolds number polar option in the blade designer dock.*

Advanced Blade Design

The advanced blade design table can be accessed through the *Advanced Blade Design Tabl*, as shown in Fig. 67.

The columns of the advanced blade property table are:

- **Position [m]** is the position of the station along the blade pitch axis (in m). It should match the position given in the basic blade properties.
- **X (IP) Offset [m]** is an additional offset of the blade station in the global y-direction. This is the in-plane direction.
- **Y (OOP) Offset [m]** is an additional offset of the blade station in the global x-direction. This is the out-of-plane direction.
- **T Axis [%c]** is the position of the thread axis as a percentage of the local chord. It is used to define the axis at which the station is rotated and also to d
  Coordinate Systems).

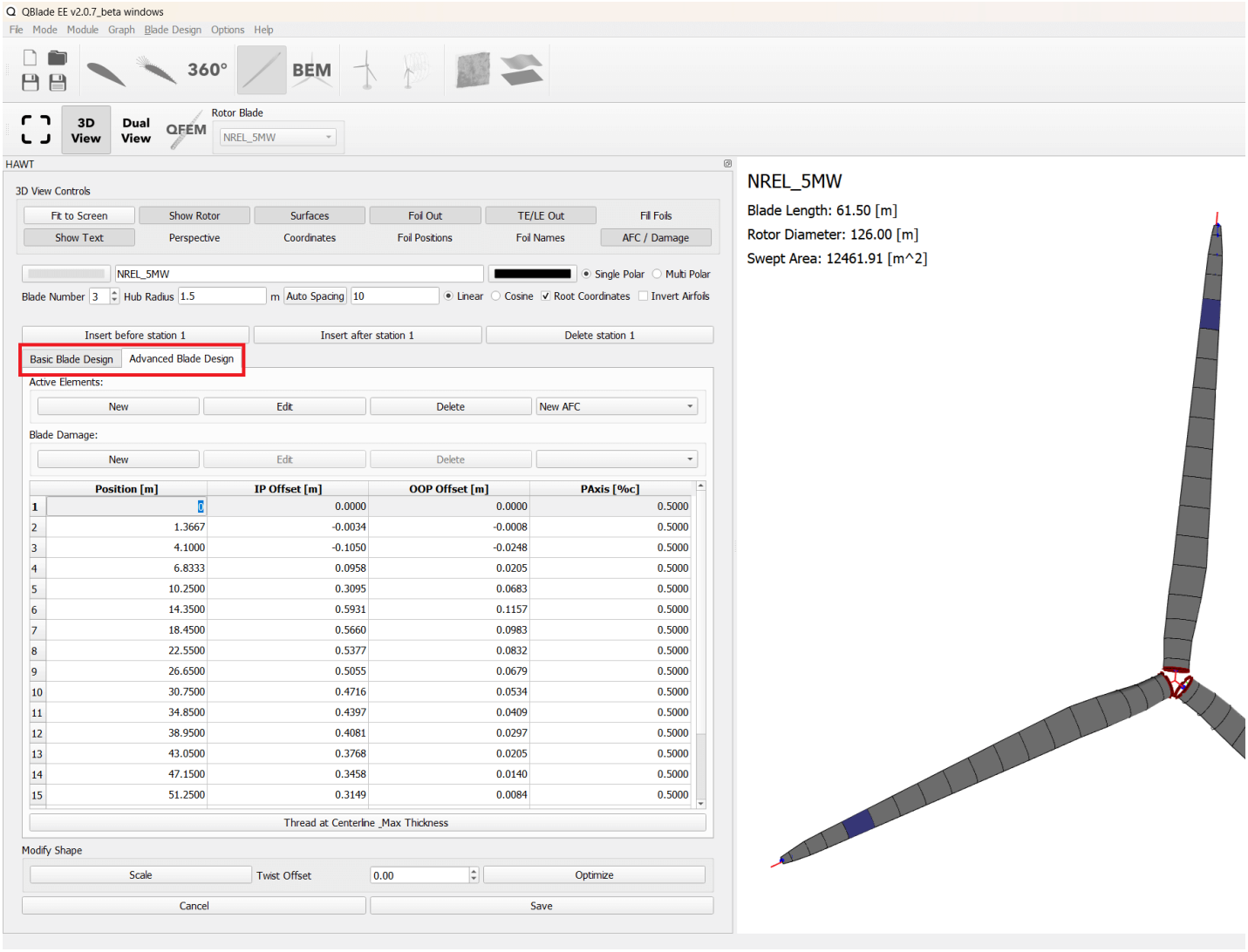


Fig. 67 *The Advanced Blade Design tab*

Active Elements

QBlade can model and add active elements, such as active trailing edge flaps, to a blade definition. This is done in the advanced blade design tab, as show
one ore more Dynamic Polar Sets must have been previously defined in the Polar Extrapolation Module (see: Polar Extrapolation Overview). Each active e
dynamic polar set can be chosen at each blade station. Linear interpolation is used between the two Dynamic Polar Sets. An active element is always app

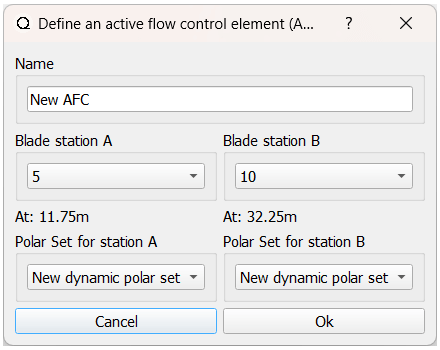


Fig. 68 *The active element dialog.*

Blade Damage

A blade damage can be added to a blade definition. This feature is intended to model *damaged* blade elements (for instance leading edge erosion effects) t
blade stations of individual blades. The airfoil polar for the damaged blade station must be created previously in the Polar Extrapolation Overview. Similar
two stations. These can have different airfoils and polars, even multi-polar sets. QBlade will interpolate along the blade between the two stations with the
damage can be assigned to an individual blade and can thus be used to model an aerodynamic imbalance.

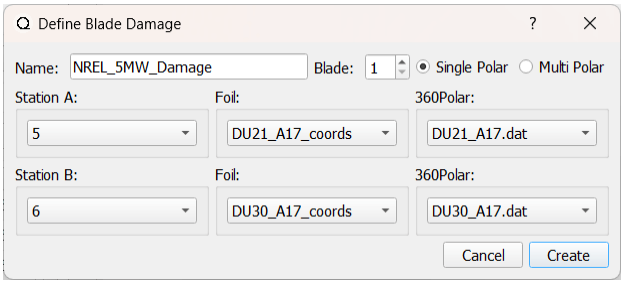


Fig. 69 *The blade damage dialog.*

Importing and Exporting Blade Definitions

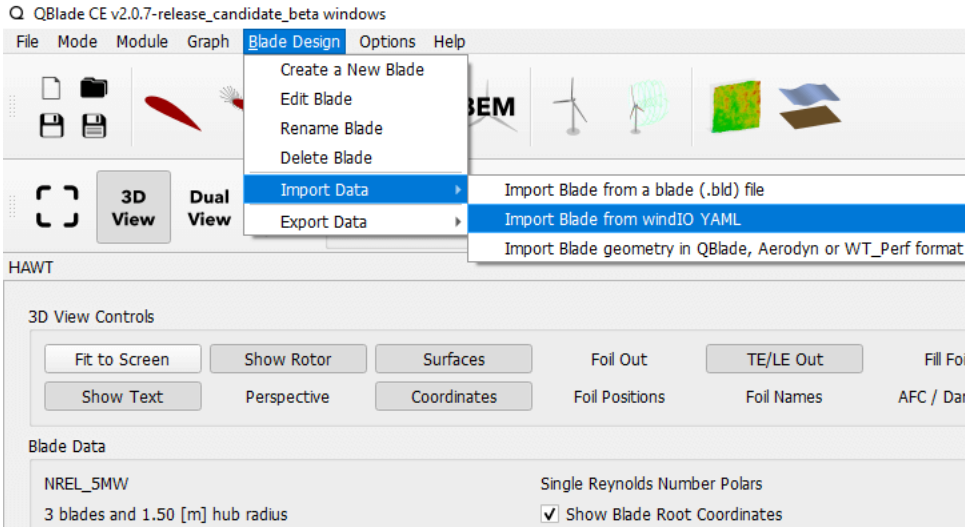


Fig. 70 *Blade Definition Import Options*

QBlade allows to import and export blade definitions in a series of formats. The import/export options are located in the menu item *Blade Design*, see Fig.
in the following formats:

- QBlade blade format (`.bld`),
- Blade geometry in QBlade, AeroDyn and WT_perf format,
- AeroDyn V15 full blade definition,
- WindIO Yaml Definition

QBlade is currently capable of exporting blade definitions in the following formats:

- QBlade blade definition ASCII format (`.bld`),
- Full blade definition in AeroDyn V13 format,
- 3D blade geometry in STL or `.txt` format.

Blade definition ASCII File

When a blade is exported into the `.bld` format, the associated 360 polar (`.plr`) and airfoil (`.afl`) files are automatically created. An exemplary `.bld` fi

```
-------------------------------------QBlade Blade Definition File-------------------------------------------------
Generated with : QBlade IH v2.0.2_alpha windows
Archive Format: 310002
Time : 12:05:50
Date : 29.06.2022

-----------------------------------Object Name-------------------------------------------------------------------
NREL_5MW                              OBJECTNAME         - the name of the blade object
```

```
---------------------------------------Parameters-------------------------------------------------------------
HAWT                                   ROTORTYPE          - the rotor type
3                                      NUMBLADES          - number of blades

---------------------------------------Blade Data-------------------------------------------------------------
POS [m]             CHORD [m]          TWIST [deg]        OFFSET_X [m]       OFFSET_Y [m]       TAXIS [-]          POLAR_FILE
1.5000              3.5420             0.0000             0.0000             0.0000             0.5000             t100.0_nre_5mw_Cylinder_1_s
2.8674              3.5420             13.3080            0.0027             0.0006             0.5000             t100.0_nre_5mw_Cylinder_1_s
5.5992              3.8540             13.3080            0.1057             0.0250             0.5000             t100.0_nre_5mw_Cylinder_1_s
8.3289              4.1670             13.3080            0.2499             0.0591             0.5000             t90.0_nre_5mw_Cylinder_2_se
11.7402             4.5570             13.3080            0.4586             0.1085             0.5000             t40.0_nre_5mw_DU40_airfoil_
15.8399             4.6520             11.4845            0.5696             0.1157             0.5000             t35.0_nre_5mw_DU35_airfoil_
19.9410             4.4580             10.1649            0.5485             0.0983             0.5000             t35.0_nre_5mw_DU35_airfoil_
24.0421             4.2490             9.0132             0.5246             0.0832             0.5000             t30.0_nre_5mw_DU30_airfoil_
28.1432             4.0070             7.7970             0.4962             0.0679             0.5000             t25.0_nre_5mw_DU25_airfoil_
32.2443             3.7480             6.5457             0.4654             0.0534             0.5000             t25.0_nre_5mw_DU25_airfoil_
36.3454             3.5020             5.3623             0.4358             0.0409             0.5000             t21.0_nre_5mw_DU21_airfoil_
40.4464             3.2560             4.1890             0.4059             0.0297             0.5000             t21.0_nre_5mw_DU21_airfoil_
44.5475             3.0100             3.1256             0.3757             0.0205             0.5000             t17.0_nre_5mw_NA64_A17_airf
48.6486             2.7640             2.3193             0.3452             0.0140             0.5000             t17.0_nre_5mw_NA64_A17_airf
52.7497             2.5180             1.5261             0.3146             0.0084             0.5000             t17.0_nre_5mw_NA64_A17_airf
56.1676             2.3130             0.8629             0.2891             0.0044             0.5000             t17.0_nre_5mw_NA64_A17_airf
58.9013             2.0860             0.3699             0.2607             0.0017             0.5000             t17.0_nre_5mw_NA64_A17_airf
61.6338             1.4190             0.1059             0.1774             0.0003             0.5000             t17.0_nre_5mw_NA64_A17_airf
63.0000             0.9610             0.0000             0.1201             0.0000             0.5000             t17.0_nre_5mw_NA64_A17_airf
```

[1] R. Gasch and J. Twele. *Windkraftanlagen: Grundlagen, Entwurf, Planung und Betrieb*. Vieweg+Teubner Verlag, 4th edtion edition, 2005. ISBN 978-3-322-99446-2

BEM Analysis Overview

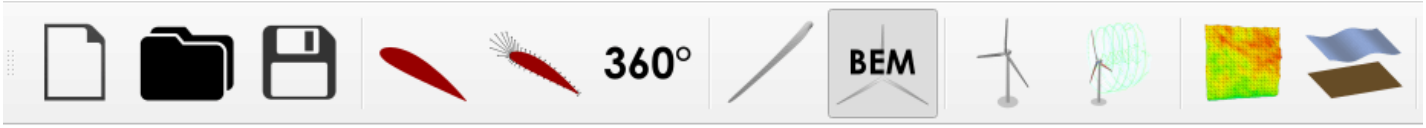


Fig. 71 *The BEM module in QBlade's main tool bar.*

The *Steady BEM Analysis* tool allows the user to run aerodynamic simulations with very low computational expense, short run times and generally accurate preliminary results with regard to rotor performance parameters. The module is very useful for:

- Performing a first evaluation of a blade design,
- Calculating a fast estimate on annual energy production (AEP), or
- Identifying control strategies through parameter studies.

This section gives insight into the three submodules *Rotor BEM*, *Characteristic BEM* and *Turbine BEM*.

Rotor BEM

In the *Rotor BEM* submodule, the user can carry out rotor blade simulations over a range of tip speed ratios (TSRs). A rotor simulation can only be defined when at least one rotor blade is present in the database (see Blade Design Overview). When defining a rotor simulation in the *Analysis Settings* box, the user can select the desired corrections (Corrections) to the BEM algorithm and the simulation parameters. Once a simulation is defined, the user can select a range of TSRs, and the incremental step for the simulation.

A rotor simulation is always carried out with dimensionless arguments. The freestream velocity is assumed to be uniform and the rotor radius is normalized for the computation. This implies that no dimensional power curve or load distributions (e.g. bending moment) can be computed during a rotor simulation.

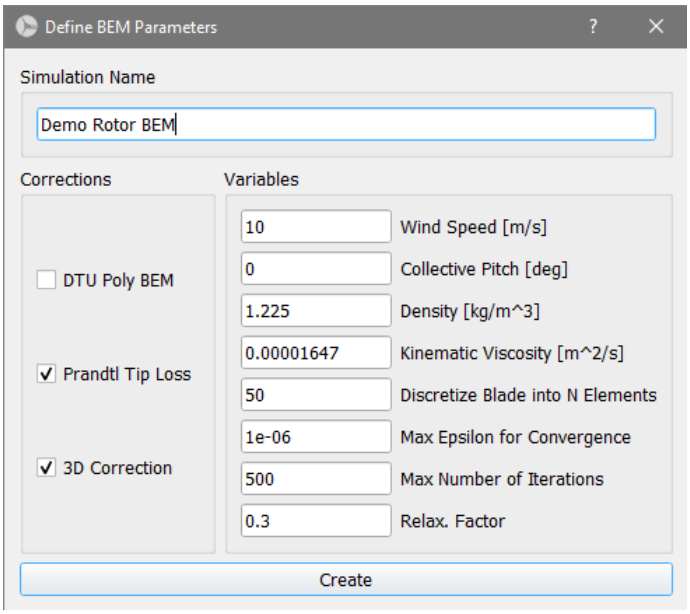


Fig. 72 *Definition of a Rotor BEM simulaton.*

Characteristic BEM

In the *Characteristic BEM* submodule simulations can be carried out over a specified range of windspeeds, rotational speeds and pitch angles. By right-clicking on each graph in the graphics module, the user can specify the plotted variables which are displayed. When the selected windspeed, rotational speed or pitch angle is changed in the top toolbar, the series of curves is changed accordingly. This submodule is of great help when designing custom control strategies for variable rotational speed and/or pitch controlled wind turbine rotors.

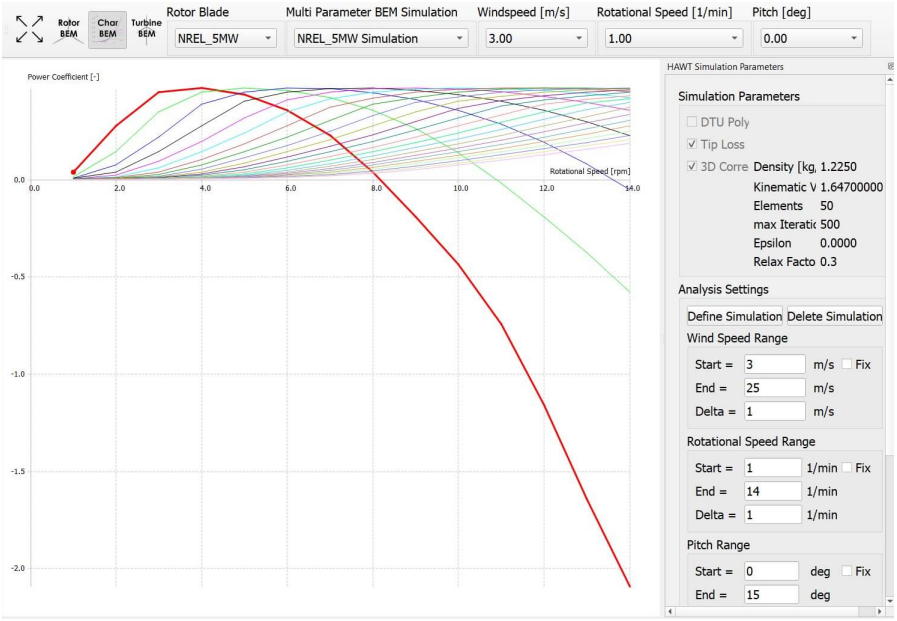


Fig. 73 *Result of a characteristic BEM simulation. The C_p coefficent is plotted over rpm, showing multiple wind speeds curves for a constant pitch angle.*

Turbine BEM

In the *Turbine BEM* submodule, the user can simulate steady state BEM simulations. To define a wind turbine, a rotor blade must be present in the runtime database. In preparation for the simulations, a turbine has to be set up. This requires specification of the turbine type and turbine operational parameters. The turbine type is defined by:

- **Power Regulation**:
    - None (stall): A stall regulated turbine has no pitch control and the power output is limited solely when stall occurs at the rotor. Designing a stall turbine that limits its power to the desired output and at the desired windspeed requires an iterative approach.
    - Pitch limited: Requires the user to specify a nominal power output. When the windspeed that yields the nominal power output is reached, the blades are pitched to reduce the power for higher windspeeds to the nominal output.
    - Prescribed: Allows to set the pitch to an arbitrary value defined in an `.txt` file. This option is useful to match a certain control behavior or for code-to-code comparisons.
- **Transmission**:
    - Single: One stationary rotational speed in which the turbine operates over the whole range of windspeeds.
    - 2-step: Two rotational speeds and a windspeed at which the transmission switches have to be selected.
    - Optimal: A minimum and a maximum value for the rotational speed has to be selected. Additionally, the user selects a desired TSR from which a rotational speed is computed for every given wind speed during the simulation.
    - Prescribed: Allows the user to set the rpm to an arbitrary value defined in an `.txt` file. This option is useful to match a certain control behavior or for code-to-code comparisons.

Further parameters that need to be selected by the user are shown in Fig. 74. At V_{cutin}, the turbine starts and at V_{cutout} the turbine stops operation. To account for power losses that are not of aerodynamical nature but are caused by the efficiency of the generator and the gearbox, a value for fixed losses and a value for variable losses can be selected. The equation in which these losses are implemented is:

$$P_{out} = (1 - k_v)P_0 - P_{fixed},$$

where, k_v is the variable loss factor and P_{fixed} the fixed losses. Finally, a previously defined turbine blade has to be selected.

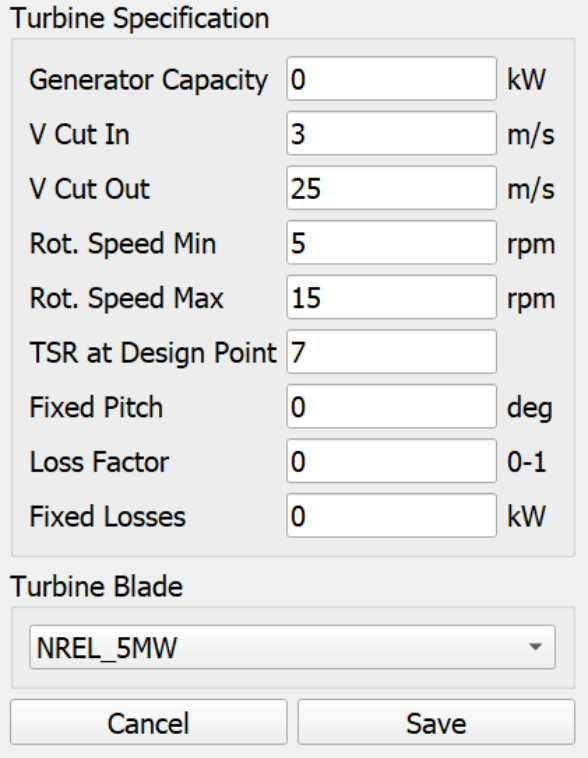


Fig. 74 *Turbine specification dialogue.*

After the turbine has been added to the runtime database, the BEM simulation can be executed identically to the Rotor BEM described above. The simulation is carried out over the specified range of windspeeds with the selected incremental step size.

User's Guide

General GUI Functionality

- GUI Overview
- General Settings
- Opengl Light Settings
- Graph Functionality
    - Graph Context Menu
    - Variables & Styles and Axes
    - Curve Styles
    - Graph Layout

Data Structure, Import & Export

- Object Hierarchy and Data Structure
- Project Serialization
- Data Objects Import and Export

Coordinate Systems

- Global Coordinate System
- Local Body Coordinate Systems
    - Local Blade Coordinate System
    - Local Tower Coordinate System
- Local Sensor Coordinate Systems

HAWT, VAWT and PROP Modes

- Setting the Design Mode
- HAWT Mode, VAWT Mode and PROP Mode

Airfoil Generation and Import

- Airfoil Generation Overview
- Creating Standard Airfoils
- Importing Airfoils

- Airfoil Editing Options
- Airfoil Transformations
- Exporting Airfoils

Airfoil Analysis with XFoil

- Airfoil Analysis Overview
- Carrying out an XFoil Analysis
    - XFoil Batch Analysis
- Operational Point Analysis
- Importing Polar Data
- Exporting Polar Data

360° Polar Extrapolation

- Polar Extrapolation Overview
- Viterna Extrapolation
- Montgomery Extrapolation
- Polar Decomposition
- Dynamic Polar Sets
- Import and Export of 360 Polars

Aerodynamic Blade Design

- Blade Design Overview
- Basic Blade Design
    - Multi Polar Blade Definition
- Advanced Blade Design
    - Active Elements
    - Blade Damage
- Importing and Exporting Blade Definitions
- Blade definition ASCII File

Steady BEM Simulation

- BEM Analysis Overview
- Rotor BEM
- Characteristic BEM

- Turbine BEM

Wind Turbine Modeling

Controller Modeling

Substructure Modeling

Wind Turbine Simulation

Multi Turbine Simulation

Windfield Generation

Wave Generation

Design Load Case Generation

- Design Load Cases Overview
- DLC Object Generation (in GUI)
    - Template
    - Turbine Data
    - DLC Parameter Range
    - Wind Model
    - Turbulent Grid Parameters
    - Environmental Vars
    - General Sim Settings
    - Rotational Speed Setting
    - Simulation Event(Fault) Settings
    - Structural Sim Settings
    - Modal Analysis Settings
    - Stored Sim Data
    - Offshore DLC Generation in the GUI
- Exporting DLC Definitions
- DLC Definition via Spreadsheets
- DLC Generation via Spreadsheets
    - Importing DLC's from a Spreadsheet
    - Exporting DLC's from a Spreadsheet

Command Line Interface (CLI)

- CLI Overview
- CLI Functionality
- Sample CLI Call to Start a Batch Run

Software in Loop Interface (SIL)

- Software in Loop (SIL) Overview
- Quick Start with the SIL Interface in Python
- Interface Function Definitions
- Interface Function Documentation
- Python Example: Running the QBlade Library
- Python Example: Definition of the QBladeLibrary Class
- Matlab Example: Running the QBlade Library
- Matlab Example: Definition of the QBladeLibrary Class

Changelog

Modeling Overview

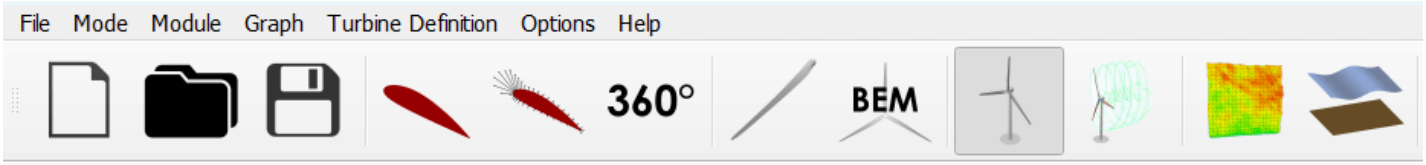


Fig. 75 *The turbine module in the QBlade's main toolbar.*

A new wind turbine can be defined, or an existing turbine can be edited in the **Turbine definition Module** of QBlade. When a new wind turbine is defined a dialog opens where the user can specify in detail how the turbine should be modeled **aerodynamically**, **structurally** and if **controllers** should be included.

The Turbine Definition Dialog

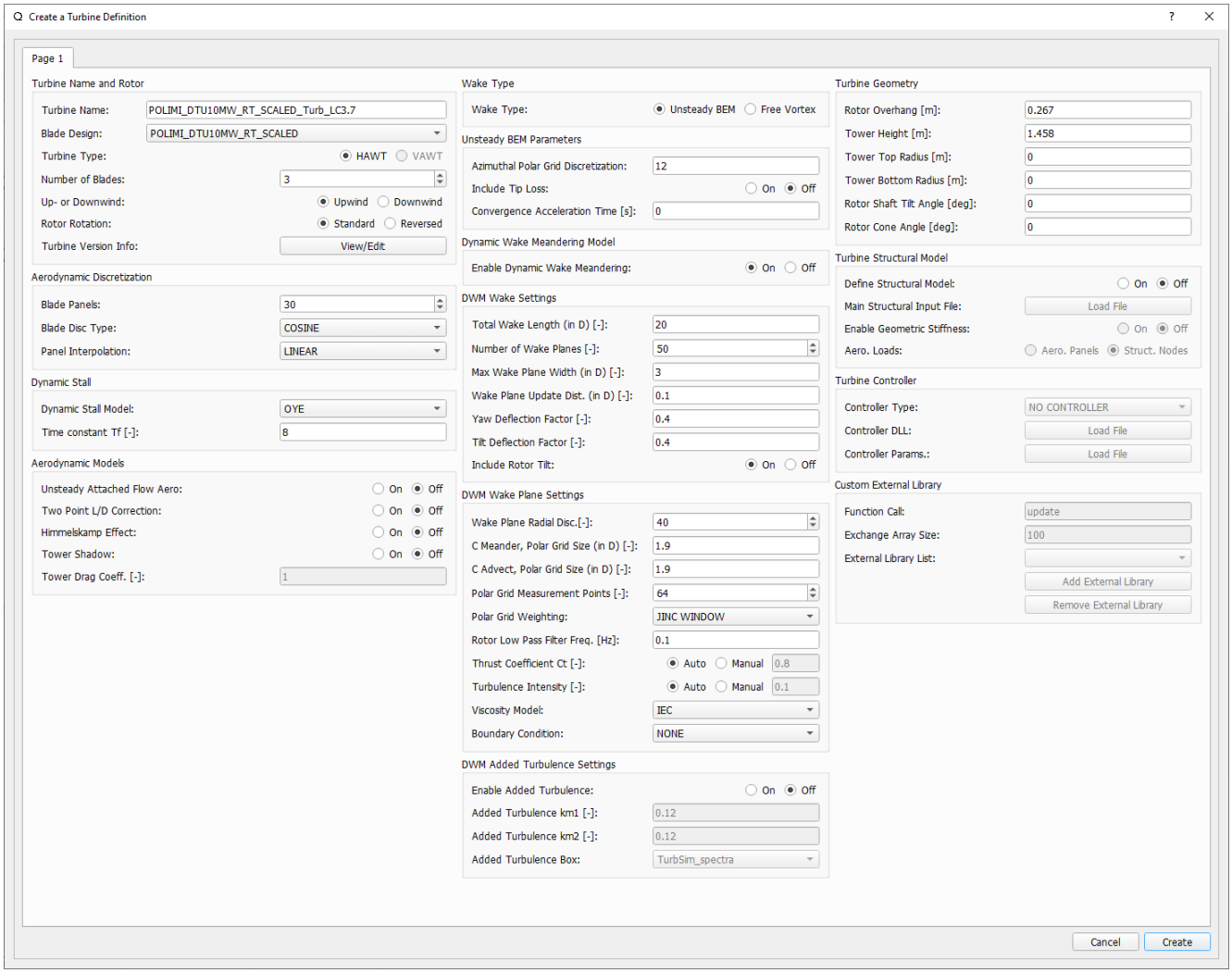


Fig. 76 *The turbine definition dialog showing unsteady BEM, and Dynamic Wake Meandering options (click to enlarge).*

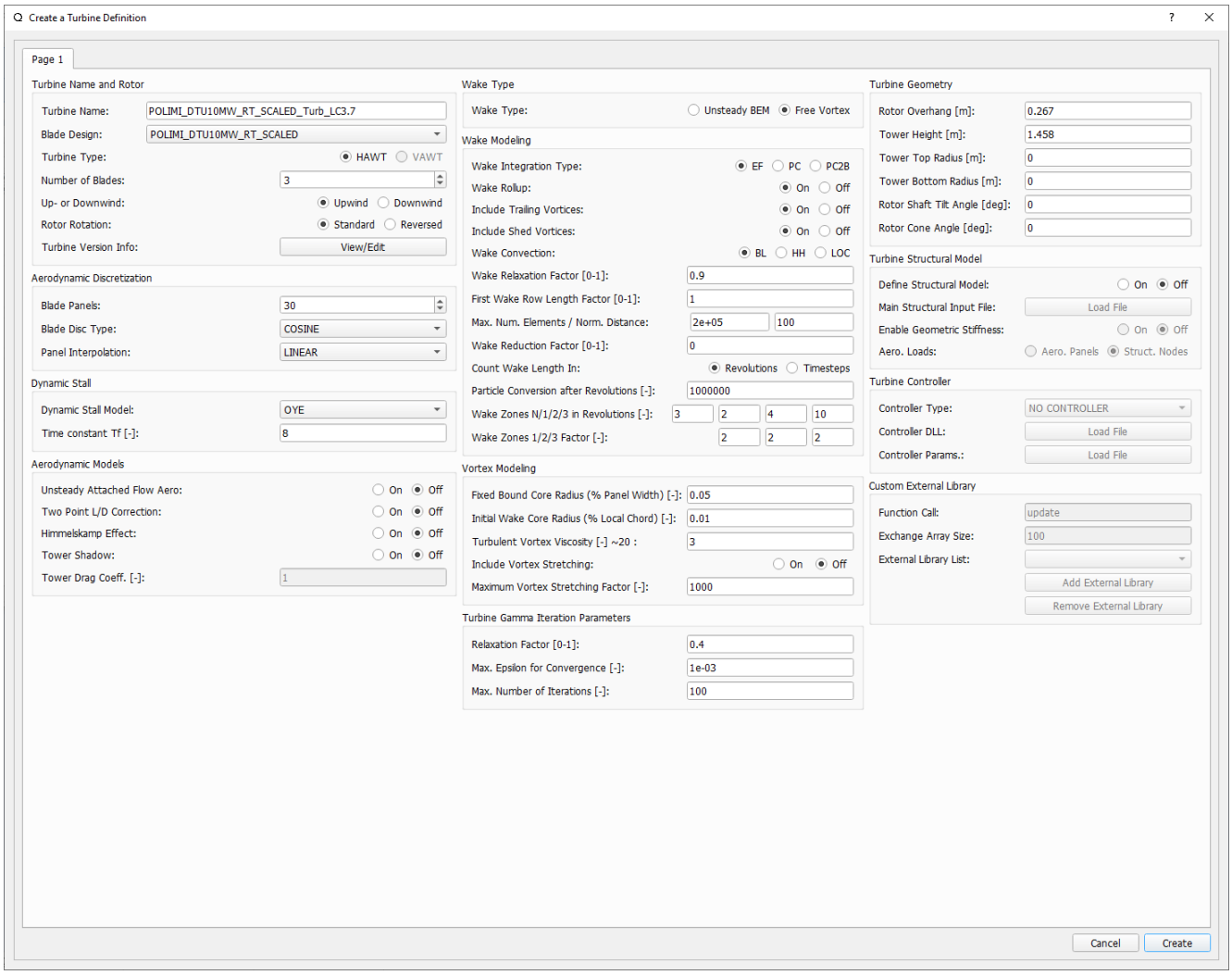


Fig. 77 *The turbine definition dialog showing LLFVW options (click to enlarge).*

In the turbine definition dialog the user can set all details of the wind turbine that should be modeled. In the following all parameters and entries of this dialog are explained.

Aerodynamic only Turbine Definitions

A turbine object can either be defined with or without a structural model definition. If the turbine object is defined without a structural model it is assumed as a rigid model, and only aerodynamic forces are evaluated during a simulation. The only operational modes for a turbine object without a structural model are constant or prescribed operational speed (see Aerodynamic Modeling).

Aeroelastic Turbine Definitions

If a structural model (see Structural Modeling) is added to a turbine object an aeroelastic simulation can be performed. The simulation results then include gravitational, inertia, centrifugal, gyroscopic and aerodynamic forces. For turbine objects with structural model definitions a wind turbine controller may be added to the turbine definition (see Wind Turbine Controllers). If a turbine also contains a definition of a floating or bottom fixed substructure (see Substructure Modeling) also hydrodynamic forces are evaluated during the simulation. All input parameters of a turbine definition

in QBlade are described in the following sections of this documentation, ordered by their appearance in the dialog. If you want to quickly find information about a specific parameter it is suggested to use the **search functionality** of this online documentation.

General Turbine Parameters

Turbine Name and Rotor

- **Turbine Name**: A unique name needs to be assigned to the turbine object.
- **Blade Design**: Choose the aerodynamic blade design that will be used for this turbine object.
- **Turbine Type**: Switch between a HAWT or a VAWT turbine design.
- **Number of Blades**: Sets the numvber of blades for this rotor. This overrides the number of blades that is specified in the *Blade Design*. If the turbine is equipped with a structural model the number of blades is defined in the structural model main input file and this value is not used.
- **Up- or Downwind**: Choose an up- or downwind rotor configuration (only used for HAWT turbine definitions).
- **Rotor Rotation**: Sets the rotor rotation to standard (clockwise) or reversed (counterclockwise).

Turbine Version Info

- **Version Info**: Adds an info string to this turbine object that can be used to annotate changes to the model or revisions of the turbine design.

Aerodynamic Modeling

This section covers all options that are related to the aerodynamic modelling of a turbine design. Covering wake models, dynamic stall models, geometry parameters and blade discretization.

Turbine Geometry

If no structural model is defined for this turbine object the turbine geometry is defined here. If a structural model is defined the geometry is defined within the structural model files.

- **Rotor Overhang**: Sets the rotor overhang of the turbine object (see Fig. 78)
- **Tower Height**: Definition of the tower height.
- **Tower Top/Bottom Radius**: Defined the tower top and bottom radius. A linear interpolation is applied for all tower stations in between.
- **Rotor Shaft Tilt Angle**: Sets the rotor shaft tilt angle (see Fig. 78).
- **Rotor Cone Angle**: Sets the rotor cone angle (see Fig. 78).

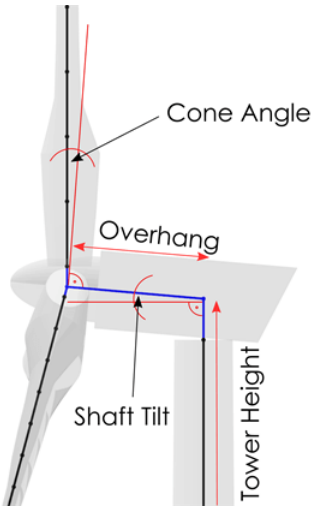


Fig. 78 *Definition of turbine geometry parameters.*

Aerodynamic Discretization

Blade Panels: Here the user can specifiy the number of blade panels and the type of spacing. A **Linear** spacing distribues the panels evenly over the blade length. A **Cosine** spacing results in a finer discretization near the blade ends (root and tip) and a slighly coarser discretization near the blade center. The option **Table** uses the aerodynamic blade definition table as a temnplate for the aerodynamic discretization, thus the user can use this option for a fully customized blade discretization.

Aerodynamic Models

- **Dynamic Stall**: The user can activate the use of a dynamic stall model. The options are: **Off**: No dynamic stall model is used. **OYE**: The OYE dynamic stall model is used, see OYE Model. **ATEF**: The ATEFlap unsteady aerodynamics model is used, see ATEFlap Model.
- **2 Point L/D Eval**: This actives the two point lift and drag evaluation model, proposed by Li *et al.*[1]. The advantage of this two point evaluation is that lift and drag predictions for dihedral or conned wind turbine rotor are improved and the airfoil **pitch rate** is explicitly being taken into account by evaluating the angle of attack at the three-quarter chord point and then applying the aerodynamic coefficients at the quarter-chord point.
- **Himmelskamp Effect**: The correction for the *Himmelskamp* effect can be activated here, see Himmelskamp Effect.
- **Tower Shadow**: The *Tower Shadow Effect* can be avtivated or deactivated here, see Tower Influence.
- **Tower Drag Coeff.**: Sets the drag coefficient that is used to model the *Tower Shadow Effect*. If a structural model is used for this turbine the tower drag coefficient is set in the 20th column of the structural tower data table, see Tower and Torquetube Euler Bernoulli and Timoshenko Datatable.

Wake Type

Here the user can choose between the **Free Vortex Wake** or the **Unsteady BEM** aerodynamic model. The **Unsteady BEM** model can only be used with **HAWT** turbine definitions.

Unsteady BEM

The Blade Element Momentum Method in QBlade is the default modeling option for HAWT (Horizontal Axis Wind Turbines). It has a low computational cost and good accuracy in most cases. The Unsteady BEM cannot be used to model VAWT (Vertical Axis Wind Turbines). To model a VAWT the Free Vortex Wake method must be used.

Unsteady BEM Options

- **Azimuthal Polar Grid Discretization**: The polar grid is discretized into the chosen number of azimuthal sections. A value of 1 is equal to the BEM without a polar grid.
- **Include Tip Loss**: This activates the classical BEM tip loss correction to account for a finite number of blades, see Glauert[2].
- **Convergence Acceleration Time**: The time lag constants in the unsteady BEM implementation are increased by a factor of 20 during the time span entered by the user. This enables a much faster convergence of the unsteady BEM towards a steady operational point.

The theory of the unsteady polar BEM is briefly described in Polar Grid.

Dynamic Wake Meandering Parameters

DWM Wake Settings

- **Total Wake Length (in D) [-]**: This parameter sets the total wake length of the DWM model, normalized by rotor diameter.
- **Number of Wake Planes [-]**: The total number of wake planes that is spread out over the total wake length. If the *Total Wake Length* would be 10 and the *Number of Wake Planes* 20, then the wake planes would be $\frac{10D}{20} = 0.5D$ apart.
- **Max Wake Plane Width (in D) [-]**: This specifies the max. diameter of the wake planes. If *Max Wake Plane Width* = 3, then the wake plane would cover 3 rotor diameters.
- **Wake Plane Update Dist. (in D) [-]**: After each wake plane was propagated by this distance, normalized by rotor diameter, its velocity distribution is updated (evolve step).
- **Yaw Deflection Factor [1/deg]**: Is a parameter for the yaw deflection correction, scaled with yaw error and normalized downstream distance.
- **Tilt Deflection Factor**: Is a parameter for the tilt deflection correction, scaled with tilt error and normalized downstream distance.
- **Include Rotor Tilt**: If deactivated, the rotor tilt error does not cause a vertical deflection of the wake planes.

DWM Wake Plane Settings

- **Wake Plane Radial Disc. [-]**: Specifies with how many points the wake plane is discretized over its width (*Max Wake Plane Width*).
- **C Meander, Polar Grid Size (in D) [-]**: Specifies the size of the polar grid, normalized by rotor diameter, that is used to average velocities at each wake plane to evaluate the meandering (in plane) components during the propagation step.
- **C Advect, Polar Grid Size (in D) [-]**: Specifies the size of the polar grid, normalized by rotor diameter, that is used to average velocities at each wake plane to evaluate the advection (out of plane) component during the propagation step.
- **Polar Grid Measurement Points [-]**: The number of points distributed over the polar grid (for meandering and advection calculation) at which velocities are evaluated during the averaging step.
- **Polar Grid Measurement Points [-]**: The number of points distributed over the polar grid (for meandering and advection calculation) at which velocities are evaluated during the averaging step.
- **Polar Grid Weighting [-]**: Specified the weighting function for the polar grid points, used during velocity averaging.
- **Rotor Low Pass Filter Freq. [Hz]**: The cut-off (corner) frequency f_c of a low pass time filter to obtain rotor conditions (thrust, yaw, etc.), implemented as $x_{lp,t} = x_{lp,t-1} \cdot e^{-2\pi f_c} + (1 - e^{-2\pi f_c}) \cdot x_t$.
- **Thrust Coefficient Ct [-]**: The thrust coefficient can be obtained automatically from the lo or conditions (*auto*) or manually, as a user input (*manual*).

- **Turbulence Intensity [-]**: The turbulence intensity can be obtained automatically from the inflowconditions (*auto*) or manually, as a user input (*manual*).
- **Viscosity Model**: The viscosity model that is used during the wake plane update (evolution) calculations, options are *MADSEN*, *LARSEN*, *IEC*, *KECK*..
- **Boundary Condition**: The boundary condition model that is used to generate the velocity distribution of the rotor fixed wake plane, options are *NONE*, *MADSEN*, *IEC*, *KECK*.

DWM Added Turbulence Settings

A small scale three dimensional turbulence windfield may be used to introduce wake added turbulence into the flowfield. The adde turbulence wind field should have a unit variance and isotropic turbulence. It is introduced into the wake plane velocity field by a weighting factor km:

$$T_{added}(\vec{x},t) = k_m \cdot T_{field}(\vec{x},t)$$

$$k_m(x,r) = |(1-U(x,r))| \cdot k_{m1} + |\frac{\delta U(x,r)}{\delta r}| \cdot k_{m2}$$

- **Enable Added Turbulence**: This activates the added turbulence model
- **Added Turbulence km1 [-]**: A tunable parameter in the formula for the weighting factor km
- **Added Turbulence km2 [-]**: A tunable parameter in the formula for the weighting factor km
- **Added Turbulence Box**: The windfield that is used to provide the turbulence, should be of unit variance and isotropic turbulence.

Info

This section will be expanded in the near future...

Free Vortex Wake

The Lifting Line Free Vortex Wake method in QBlade yields an improved accuracy over the Unsteady BEM method, especially for unsteady operating conditions, such as changing inflow speed or direction or floating wind turbines, that are subjected to wave forces. This increased fidelity however comes at an increased computational cost. Furthermore, the number of settings that are required to setup this method is significantly larger than the BEM settings. All LLFVW modeling options are detailed in the following.

Wake Modelling

- **Wake Integration Type**: This sets the velocity integration method for the wake nodes during the free wake convection step. **EF**: A simple 1st Order Euler Forward integration. **PC**: A 2nd O Predictor Corrector integration method. **PC2B**: A second Order Predictor Corrector Back integration scheme.

- **Wake Rollup**: This actives or deactivates the wake self-induction.
- **Include Trailing/Shed Vortices**: This sets if trailing (streamwise) or shed (spanwise) vortices are generated at the blades trailing edge during every timepstep.
- **Wake Convection**: The user can choose here which free-stream velocity contributes to the total convection velocitzy of the wake nodes. **BL**: The convection velocity is the mean boundary layer velocity (as a function of height). **HH**: The convection velocity is the constant hub-height velocity. **LOC**: The convection velocity is evaluated locally at each wake node position.
- **Wake Relaxation Factor**: This factor can be used to *relax* the wake by blending out the starting vortex. The factor controls how long the wake is allowed to be after a given number of rotor revolutions or timesteps (depending on the **Count Wake Length In** setting). Such as a value of 0.5 allows for a wake length of 5 revolutions after the rotor has undergone 10 revolutions. A factor of 1 deactivaes the blending.
- **First Wake Row Length Factor**: This factor can be used to assign a shortened length to the newly created wake elements at the trailing edge so that the newly created shed vorticity is in closer proximity to the blade. A factor of 1 deactivates the shortening.
- **Max Num. Elements / Norm. Distance**: These two values are used to cut-off the wake after a fixed numbner of vortex elements has been created (Max. Num. Wake Elements) or after a vortex element has reached a distance (normalized by rotor diameter) from the hub that is larger than **Norm. Distance**.
- **Wake Reduction Factor**: This factor *filters* out wake elements that have a circulation smaller than the maximum circulation in the wake multiplied by this factor. In most cases this effectively removes shed vorticity that does not significantly affect the wake induction (see Fig. 79).

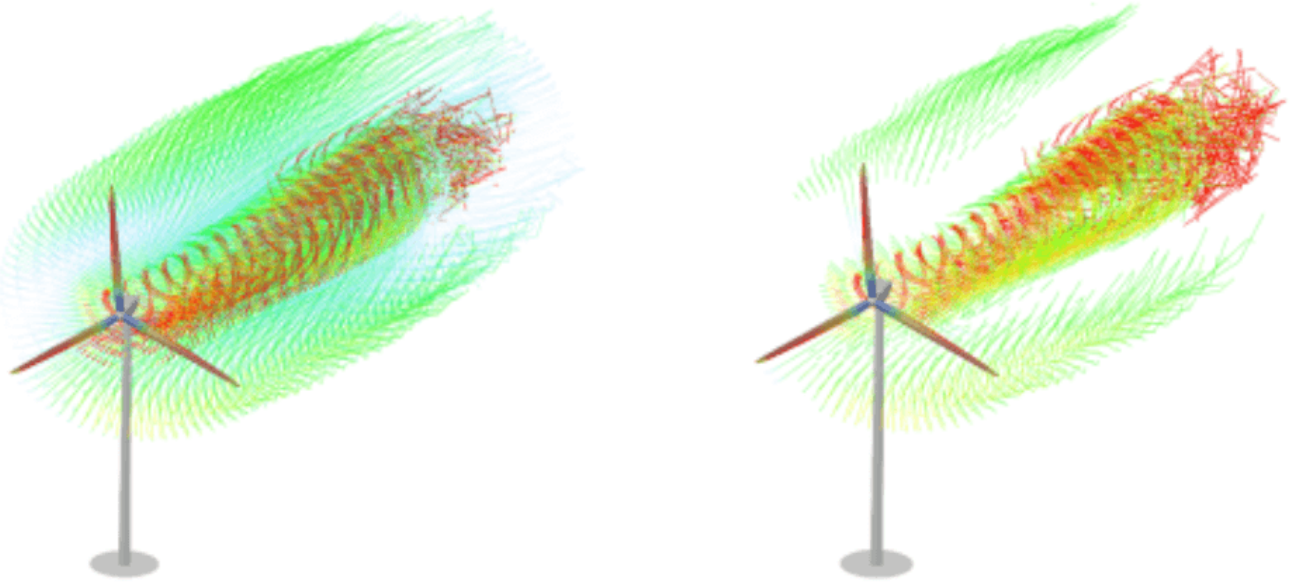


Fig. 79 *Visualization of the wake reduction approach.*

- **Count Wake Length In**: This setting controls how the age of a vortex element is counted. as a number of rotor revolutions, or as a number of timesteps that have passed since the element was created.

- **Particle Conversion after [Revolutions/Timesteps]**: (**Only QBlade-EE**) This setting controls when a vortex filament is converted into a vortex particle. If the vortex elemnt has reached an age (in timesteps or revolutions) equal to this value it is converted into a particle.
- **Wake Zones N/1/2/3 in [Revolutions/Timesteps]**: This setting controls the *length* of the different wake zones. The length is either counted in rotor revolutions or in timesteps, depending on the setting (**Count Wake Length In**). Each wake zone has a successively coarser discretization (depending on the **Wake Zones Factor** settings) to reduce the total number of free wake elements and thereby to speed up the simulation.
- **Wake Zones 1/2/3 factor**: These (integer) factors control by how much the wake is coarsenend in between the different wake zones. A factor of 4 means that when transitioning from one zone to the next 4 wake elements are replaced by a single wake element to coarsen the wake resolution (see :numref:fig-wakezones`.png`).

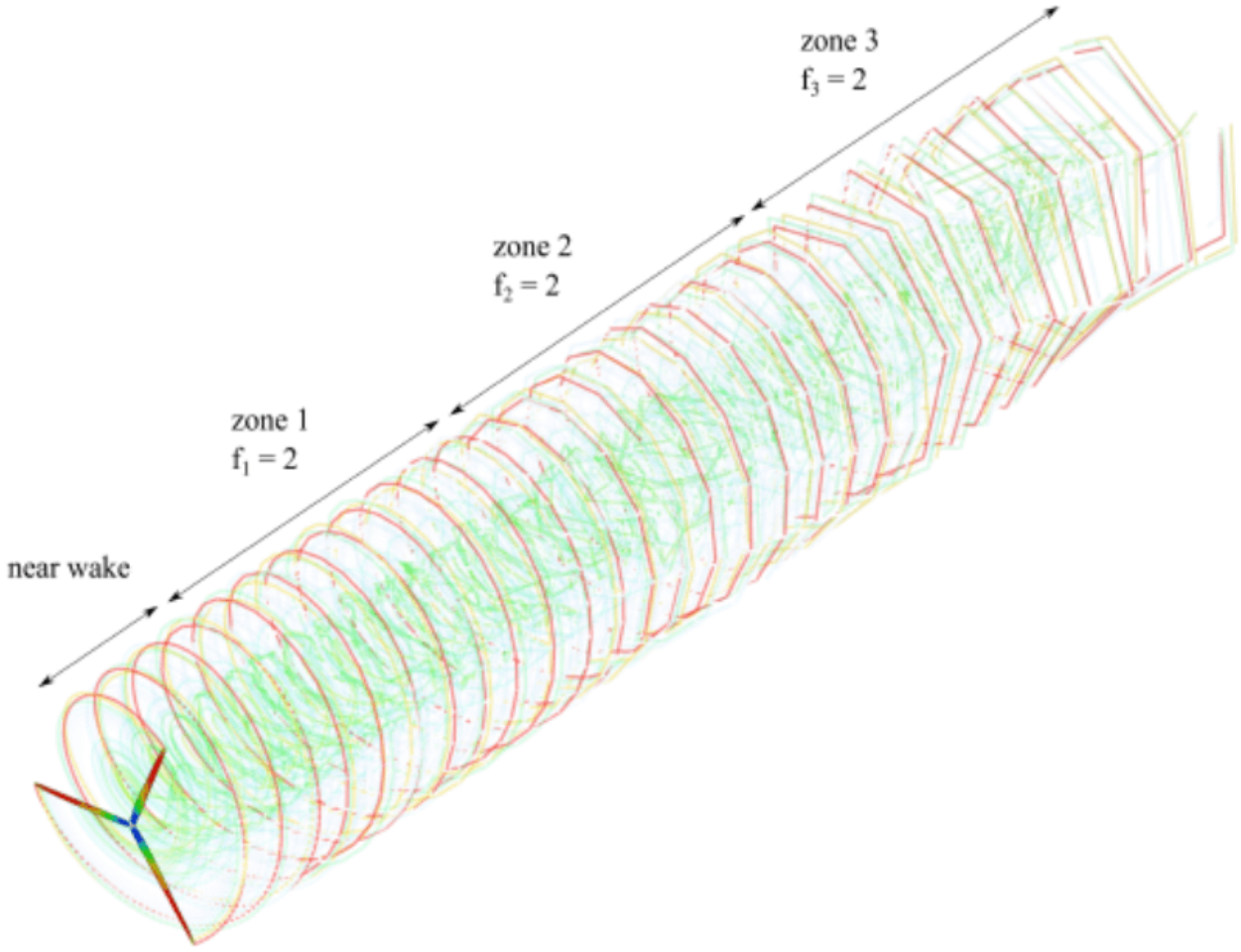


Fig. 80 *Visualization of the wake zoning approach.*

Vortex Modelling

- **Fixed Bound Core Radius (% Chord)**: This sets the fixed core radius of the bound blade vortices. Defined as a fraction of the local blade chord.
- **Initial Wake Core Radius (% Chord)**: This sets the intial core radius of the free vortices that are created at the blades trailing edge. Defined as a fraction of the local blade chord.

- **Turbulent Vortex Viscosity**: This value is used in the vortex core growth model, see Vortex Core Desingularization.
- **Include Vortex Stretching**: This option activates vortex stretching, see Vortex Core Desingularization.
- **Maximum Vortex Stretching Factor**: After the cummulative vortex strain rate has reached a value larger than this factor it is automatically removed from the wake.

Turbine Gamma Iteration Parameters

- **Relaxation Factor**: This relaxation factor is used when the blade circulation is updated during the circulation iteration.
- **Max. Epsilon for Convergence**: The convergence criteria for the blade circulation.
- **Max. Number of Iterations**: The maximum number of blade circulation iterations that will be carried out.

[1] A. Li, M. Gaunaa, G. R. Pirrung, A. Meyer Forsting, and S. G. Horcas. How should the lift and drag forces be calculated from 2-d airfoil data for dihedral or coned wind turbine blades? *Wind Energy Science Discussions*, 2022:1–40, 2022. URL: https://wes.copernicus.org/preprints/wes-2021-163/, doi:10.5194/wes-2021-163.

[2] H. Glauert. *Airplane Propellers*, chapter Aerodynamic Theory, pages 169–360. Springer Berlin Heidelberg, 1935. doi:10.1007/978-3-642-91487-4_3.

Structural Modeling

A structural turbine model in QBlade is defined by a set of ASCII input files that describe the overall dimensions of the turbine and the structural properti stiffness and damping properties that are required by the structural simulation engine to resolve the structural dynamics. In most cases these structural p structural blade layout and can generate the cross-sectional mass, stiffness and damping properties that are required to setup a structural model in QBlac in QBlade can be found in the section: Multi Body Beam Formulation.

A structural model may be loaded into a turbine definition by loading the structural model main file through the open file dialog. When a structural model The dialog that shows the file contents can also be used to quickly change parameters of the structural model, without the need to modify and save the fi *reload* files from the file system, such that changing the string of a blade parameter table doesn't lead to reloading the newly defined filename. Generally, i work with the structural model files outside of QBlade in a text editor.

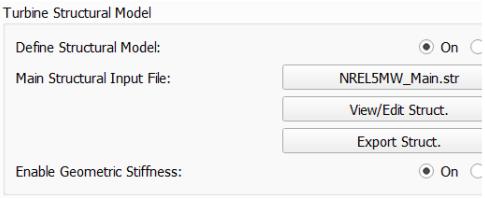


Fig. 81 *The structural model dialog.*

An overview of the file structure of the structural model definition files is shown in Fig. 82. The main input file needs to be loaded through the dialog. It c tower and the blades.

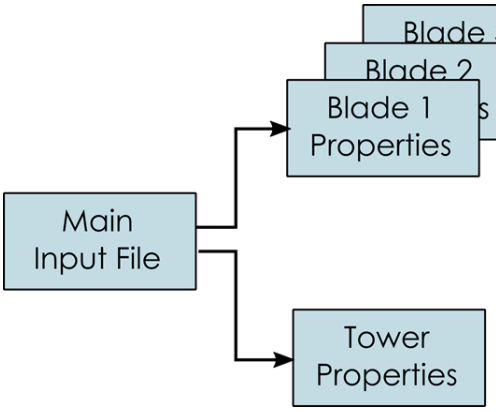


Fig. 82 *The file structure of the structural model input fi*

Main Structural Definition File

Within the main definition file the overall dimensions of the wind turbine are defined. Furthermore, nacelle mass and inertia and drivetrain properties are the detailed structural properties of the blades, tower and the torquetube (for a VAWT turbine). An overview of important parameters that define the turb

Exemplary Main File

An exemplary main structural input file for the NREL 5MW HAWT wind turbine is shown below and will be discussed in more detail in the following:

Listing 2 : Main Input File

```
---------------------- QBLADE STRUCTURAL MODEL INPUT FILE -----------------
NREL 5MW Turbine
------------------------------ CHRONO PARAMETERS -------------------------
0.2             GLBGEOEPS - Global geometry epsilon for node placement

------------------------------ HAWT TURBINE CONFIGURATION ----------------
2.5             PRECONE - Rotor PreCone (deg) (HAWT only)
5               SHFTTILT - Turbine Shaft Tilt (deg) (HAWT only)
5.0191          OVERHANG - Rotor Overhang (m) (HAWT only)
1.96256         TWR2SHFT - Tower to Shaft distance (m) (HAWT only)

------------------------------ MASS AND INERTIA --------------------------
0.0             YAWBRMASS - Yaw Bearing Mass (kg) (HAWT only)
240000          NACMASS - Nacelle Mass (kg) (HAWT only)
1.9             NACCMX - Downwind distance from the tower-top to the nacelle CM (m) (HAWT only)
0.0             NACCMY - Lateral  distance from the tower-top to the nacelle CM (m) (HAWT only)
1.75            NACCMZ - Vertical distance from the tower-top to the nacelle CM (m) (HAWT only)
```

```
2607890         NACYINER - Nacelle Yaw Inertia (kg*m^2) (HAWT only)
56780           HUBMASS - Hub Mass (kg)
115926          HUBINER - Hub Inertia (kg*m^2)

------------------------------ DRIVETRAIN MODEL -------------------------
97              GBRATIO - gearbox ratio (N)
1.0             GBOXEFF - gearbox efficiency (0-1)
1.0             GENEFF - generator efficiency  (0-1)
true            DRTRDOF - model drivetrain dynamics (true / false)
534.116         GENINER - Generator side (HSS) Inertia (kg*m^2)
867637000       DTTORSPR - Drivetrain torsional stiffness (N*m/rad)
6215000         DTTORDMP - Drivetrain torsional damping (N*m*s/rad)

------------------------------ BRAKE MODEL ------------------------------
0               BRKTORQUE - maximum brake torque
0               BRKDEPLOY - brake deploy time (s) (only used with DTU style controllers)
0               BRKDELAY - brake delay time (s) (only used with DTU style controllers)

------------------------------ SENSOR ERRORS ----------------------------
0               ERRORYAW - yaw error (deg) (HAWT only)
0               ERRORPITCH_1 - pitch error blade1 (deg)
0               ERRORPITCH_2 - pitch error blade2 (deg)
0               ERRORPITCH_3 - pitch error blade3 (deg)

------------------------------ BLADES -----------------------------------
3                           NUMBLD - Number of blades
NREL5MW_Blade.str           BLDFILE_1 - Name of file containing properties for blade 1
NREL5MW_Blade.str           BLDFILE_2 - Name of file containing properties for blade 2
NREL5MW_Blade.str           BLDFILE_3 - Name of file containing properties for blade 3

------------------------------ TOWER ------------------------------------
77.6                        TWRHEIGHT - Height of the tower (m)
OC3_Sparbuoy_Tower.str      TWRFILE - Name of file containing properties for the tower
OC3_Sparbuoy_Sub_LPMD.str   SUBFILE  - Name of the substructure file

------------------------------ DATA OUTPUT TYPES ------------------------
true                FOR_OUT - store (local) forces at all chosen locations
true                ROT_OUT - store (local) body rotations at all chosen locations
true                MOM_OUT - store (local) moments at all chosen locations
true                DEF_OUT - store (local) deflections at all chosen locations
true                POS_OUT - store (global) positions at all chosen locations
true                VEL_OUT - store (global) velocities at all chosen locations
true                ACC_OUT - store (global) accelerations at all chosen locations
true                LVE_OUT - store (local) velocities at all chosen locations
true                LAC_OUT - store (local) accelerations at all chosen locations

------------------------------ DATA OUTPUT LOCATIONS --------------------
any number, or zero, user defined positions can be chosen as output locations.
Locations can be assigned at any of the following components: blades, struts, tower
and guy cables. See the following examples for the used nomenclature:

BLD_1_1.0           - exemplary position, blade 1 at 100% normalized radius
BLD_1_0.8           - exemplary position, blade 1 at  80% normalized radius
BLD_1_0.5           - exemplary position, blade 1 at  50% normalized radius
BLD_1_0.4           - exemplary position, blade 1 at  40% normalized radius
BLD_1_0.2           - exemplary position, blade 1 at  20% normalized radius
BLD_1_0.0           - exemplary position, blade 1 at  00% normalized radius

BLD_2_1.0           - exemplary position, blade 2 at 100% normalized radius
BLD_2_0.8           - exemplary position, blade 2 at  80% normalized radius
BLD_2_0.5           - exemplary position, blade 2 at  50% normalized radius
BLD_2_0.4           - exemplary position, blade 2 at  40% normalized radius
BLD_2_0.2           - exemplary position, blade 2 at  20% normalized radius
BLD_2_0.0           - exemplary position, blade 2 at  00% normalized radius

BLD_3_1.0           - exemplary position, blade 3 at 100% normalized radius
BLD_3_0.8           - exemplary position, blade 3 at  80% normalized radius
BLD_3_0.5           - exemplary position, blade 3 at  50% normalized radius
BLD_3_0.4           - exemplary position, blade 3 at  40% normalized radius
BLD_3_0.2           - exemplary position, blade 3 at  20% normalized radius
BLD_3_0.0           - exemplary position, blade 3 at  00% normalized radius

TWR_1.00            - exemplary position, tower at 100% normalized height
TWR_0.90            - exemplary position, tower at  90% normalized height
TWR_0.80            - exemplary position, tower at  80% normalized height
TWR_0.70            - exemplary position, tower at  70% normalized height
TWR_0.60            - exemplary position, tower at  60% normalized height
TWR_0.50            - exemplary position, tower at  50% normalized height
TWR_0.40            - exemplary position, tower at  40% normalized height
TWR_0.30            - exemplary position, tower at  30% normalized height
TWR_0.20            - exemplary position, tower at  20% normalized height
TWR_0.10            - exemplary position, tower at  10% normalized height
TWR_0.00            - exemplary position, tower at   0% normalized height
```

The different sections of the structural model input file will now be briefly discussed.

HAWT Turbine Configuration

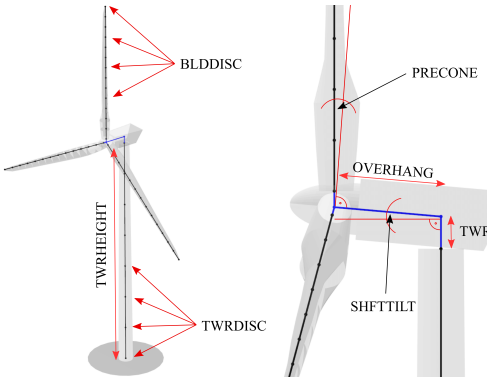


Fig. 83 *Overview of geometrical parameters for a HAWT tu*

Listing 3 : HAWT Turbine Configuration

```
------------------------------ HAWT TURBINE CONFIGURATION ----------------
2.5             PRECONE - Rotor PreCone (deg) (HAWT only)
5               SHFTTILT - Turbine Shaft Tilt (deg) (HAWT only)
5.0191          OVERHANG - Rotor Overhang (m) (HAWT only)
1.96256         TWR2SHFT - Tower to Shaft distance (m) (HAWT only)
```

In this section of the file the main geometrical turbine parameters are defined. These parameters are equivalent to the parameters discussed in Turbine G

Mass and Inertia Parameters

Listing 4 : Mass and inertia parameters

```
------------------------------ MASS AND INERTIA --------------------------
0.0             YAWBRMASS - Yaw Bearing Mass (kg) (HAWT only)
240000          NACMASS - Nacelle Mass (kg) (HAWT only)
1.9             NACCMX - Downwind distance from the tower-top to the nacelle CM (m) (HAWT only)
0.0             NACCMY - Lateral  distance from the tower-top to the nacelle CM (m) (HAWT only)
1.75            NACCMZ - Vertical distance from the tower-top to the nacelle CM (m) (HAWT only)
2607890         NACYINER - Nacelle Yaw Inertia (kg*m^2) (HAWT only)
56780           HUBMASS - Hub Mass (kg)
115926          HUBINER - Hub Inertia (kg*m^2)
```

In this section of the input file mass and inertia properties are assigned to the nacelle and the hub. It should be noted here that the parameter `HUBINER` sh blades as this is explicity included through the finite element model.

Mass and Inertia Parameters Extended

`NACCM`

this (alternative) keyword can be used to set the center of mass of the nacelle in a single line by specifying the x, y and z positions before or after the k

`NACINER`

this (alternative) keyword can be used to define the full inertia matrix of the nacelle (applied at the nacelle CM. Six values can be specified to define th

`HUBINER`

this (alternative) keyword can be used to define the full inertia matrix of the hub (applied at the hub position. Six values can be specified to define the

Nacelle Drag Model

Listing 5 : Nacelle drag

```
------------------------------ NACELLE DRAG ------------------------------
10.0            NACCAX - Downwind distance from the tower-top to the nacelle CD (m) (HAWT only)
0.0             NACCAY - Lateral  distance from the tower-top to the nacelle CD (m) (HAWT only)
1.75            NACCAZ - Vertical distance from the tower-top to the nacelle CD (m) (HAWT only)
15              NACARX - Downwind area of the nacelle (m^2) (HAWT only)
90              NACARY - Lateral  area of the nacelle (m^2) (HAWT only)
60              NACARZ - Vertical area of the nacelle (m^2) (HAWT only)
1.2             NACCDX - Downwind drag coefficient of the nacelle (-) (HAWT only)
1.2             NACCDY - Lateral drag coefficient of the nacelle (-) (HAWT only)
1.2             NACCDZ - Vertical drag coefficient of the nacelle (-) (HAWT only)
```

The nacelle drag model is optional. If no nacelle drag is defined no nacelle drag is applied. The nacelle drag can only be used with HAWT turbine definitio coefficients (NACCD). The total acting nacelle drag force in all directions is then summed up and applied at the center of drag (NACCD).

Drivetrain Parameters

Listing 6 : Drivetrain parameters

```
------------------------------ DRIVETRAIN MODEL --------------------------
97              GBRATIO - gearbox ratio (N)
1.0             GBOXEFF - gearbox efficiency (0-1)
1.0             GENEFF - generator efficiency  (0-1)
true            DRTRDOF - model drivetrain dynamics (true / false)
534.116         GENINER - Generator side (HSS) Inertia (kg*m^2)
867637000       DTTORSPR - Drivetrain torsional stiffness (N*m/rad)
6215000         DTTORDMP - Drivetrain torsional damping (N*m*s/rad)
```

This section of the main input file defined the drive train model. The drive train model in QBlade is a simple 2 mass spring-damper model. An overview is defined the electrical losses within the generator. The drivetrain is parameterized by the main shaft torsional stiffness and damping, a high speed side (HS should be summed up and assigned to the `HUBINER` value.

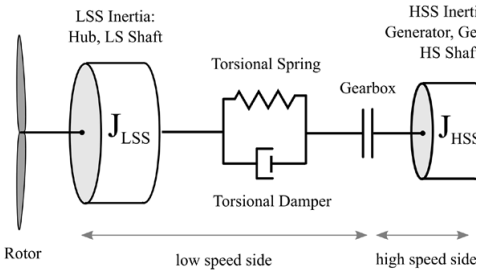


Fig. 84 *An overview of the drivetrain model in QBlade*

Brake Model Parameters

Listing 7 : Brake model parameters

```
------------------------------ BRAKE MODEL -------------------------------
0                       BRKTORQUE - maximum brake torque
0                       BRKDEPLOY - brake deploy time (s)
0                       BRKDELAY - brake delay time (s)
```

The brake in QBlade is defined as shown above. The brake is parameterized with a delay time, a deploy time and a maximum value for the brake torque. A passed the brake is activated and ramped up to the maximum brake torque (`BRKTORQUE`) during the deploy time (`BRKDEPLOY`). An overview of this process

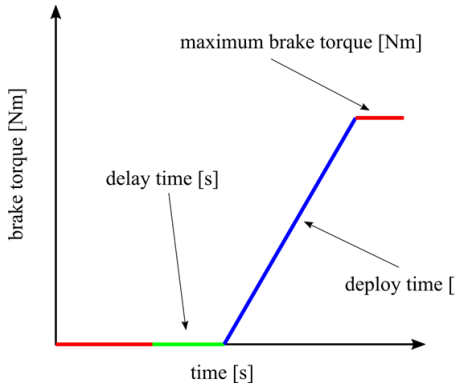


Fig. 85 *An overview of the brake model in QBlade.*

Modeling Sensor Errors

Listing 8 : Sensor errors

```
------------------------------ SENSOR ERRORS -----------------------------
0               ERRORYAW - yaw error (deg) (HAWT only)
0               ERRORPITCH_1 - pitch error blade1 (deg)
0               ERRORPITCH_2 - pitch error blade2 (deg)
0               ERRORPITCH_3 - pitch error blade3 (deg)
```

Sensor errors are defined for each blade pitch bearing sensor and the yaw bearing sensor. These errors are simply added to the corresponding signals as a

Blade Parameters

Listing 9 : Blade parameters

```
------------------------------ BLADES ------------------------------------
3                              NUMBLD - Number of blades
NREL5MW_Blade.str              BLDFILE_1 - Name of file containing properties for blade 1
NREL5MW_Blade.str              BLDFILE_2 - Name of file containing properties for blade 2
NREL5MW_Blade.str              BLDFILE_3 - Name of file containing properties for blade 3
```

The location of the structural data tables for the blades is defined by the keywords shown above. The number of blades is defined by the parameter `NUMBL` blade a keyword `BLDFILE_X` is searched for where the filename of the blade data table is defined. Different blade data tables can be assigned to each indi

Tower Parameters

Listing 10 : Tower parameters

```
------------------------------ TOWER -------------------------------------
77.6                                    TWRHEIGHT - Height of the tower (m)
OC3_Sparbuoy_Tower.str          TWRFILE - Name of file containing properties for the tower
OC3_Sparbuoy_Sub_LPMD.str       SUBFILE  - Name of the substructure file
```

The structural tower data table is defined in a similar fashion as for the blades. The keyword `TWRHEIGHT` defines the absolute height of the tower. The key bottom fixed substructure for offshore wind turbines or to model soil dynamics. If the keyword `SUBFILE` is not defined then the tower will simply be rigidl section: Substructure Modeling.

VAWT Specific Parameters

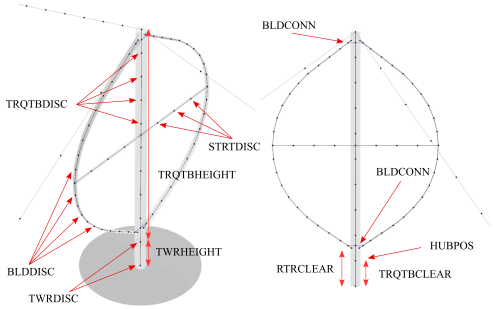


Fig. 86 *Overview of geometrical parameters for a VAWT tu*

Strut Parameters

Listing 11 : Strut parameters

```
------------------------------ STRUTS ------------------------------------
strutF100.dat           STRTFILE_1 - Name of file containing properties for strut1 (if blade has struts)
strutF100.dat           STRTFILE_2 - Name of file containing properties for strut2 (if blade has struts)
```

One structural properties table is defined for each strut. This table is used for the corresponding strut on each blade. So if there are three blades the para

Strut Constraint Table

In some cases, the user may want to specify a special constraint for the connection between the strut and the blade, or the strut and the torquetube, for e torquetube. The `STRUT_BLADE_CONSTRAINTS` and `STRUT_TORQUETUBE_CONSTRAINTS` tables can be used to control the constrained degrees of freedom between corresponding strut. Optionally, by adding an additional 9th column to the table and setting its value to 1, the local coordinate system of the blade or torq

Listing 12 : The STRUT_BLADE_CONSTRAINTS table

```
STRUT_BLADE_CONSTRAINTS
STR_ID  BLD_ID  DoF_X   DoF_Y   DoF_Z   DoF_rX  DoF_rY  DoF_rZ
1       1       1       1       1       1       0       1
1       2       1       1       1       1       0       1
```

This exemplary table models a hinge connection between strut 1 and blade 1 and strut 1 and blade 2, where the rotational degree of freedom around the

Listing 13 : The STRUT_TORQUETUBE_CONSTRAINTS table

```
STRUT_TORQUETUBE_CONSTRAINTS
STR_ID  BLD_ID  DoF_X   DoF_Y   DoF_Z   DoF_rX  DoF_rY  DoF_rZ
1       1       1       1       1       1       0       1
1       2       1       1       1       1       0       1
```

This exemplary table models a hinge connection between strut 1 and the torquetube and strut 1 and the torquetube, where the rotational degree of freec

Tower and Torquetube Parameters

Listing 14 : Tower and torquetube parameters

```
------------------------------ TOWER AND TORQUE TUBE --------------------
20.845                  TWRHEIGHT -  Height of the (fixed - non rotating) tower [m]
tower.dat               TWRFILE - Name of file containing properties for the tower

2.4376                  TRQTBHEIGHT - Height (or length) of the torque tube (the rotating part of the tower) [m]
torquetube.dat          TRQTBFILE - Name of file containing properties for the torque tube

18.427                  TRQTBCLEAR - Clearance of the torque tube, must be <= TWRHEIGHT [m]
18.427                  HUBPOS - Height of the generator hub that is connecting the torque tube with the fixed tower (VAWT only) [m]
2.4376                  TRQTBCONN - Absolute height position, starting after torque tube clearance, of a frictionless bearing that connects the tor

0.5                     BLDCONN - Absolute height position, starting after rotor clearance, of blade of the rigid blade torque tube connection 1 ir
40.853                  BLDCONN - Absolute height position, starting after rotor clearance, of blade of the rigid blade torque tube connection 2 ir

15.635                  RTRCLEAR - Rotor clearance
```

See Fig. 86 for a visual explanation of each parameter.

Cable Parameters

Listing 15 : Cable Parameters

```
------------------------------ BLDDE CABLES (VAWT only) -----------------
cable.dat               CABFILE - file containing the definitions of cables
```

An exemplary cable definition file is shown here: Cable Structural Data File.

Loading Data and Sensor Locations

Listing 16 : Output data definition and sensor locations

```
------------------------------ DATA OUTPUT TYPES -------------------------
true                    FOR_OUT - store forces at all sensor locations
true                    DEF_OUT - store deflections at all sensor locations
true                    POS_OUT - store positions at all sensor locations
true                    VEL_OUT - store velocities at all sensor locations
true                    ACC_OUT - store accelerations at all sensor locations
true                    STR_OUT - store element strain at all sensor locations
true                    AER_OUT - store aerodynamic data at all sensor locations

------------------------------ SENSOR OUTPUT LOCATIONS ------------------
any number, or zero, user defined positions can be chosen as output locations.
Locations can be assigned at any of the following components: blades, struts, tower
and guy cables. See the following examples for the used nomenclature:

BLD_1_1.0               - exemplary position, blade 1 at 100% normalized radius
BLD_1_0.8               - exemplary position, blade 1 at  80% normalized radius
BLD_1_0.5               - exemplary position, blade 1 at  50% normalized radius
BLD_1_0.4               - exemplary position, blade 1 at  40% normalized radius
BLD_1_0.2               - exemplary position, blade 1 at  20% normalized radius
BLD_1_0.0               - exemplary position, blade 1 at  00% normalized radius

BLD_2_1.0               - exemplary position, blade 2 at 100% normalized radius
BLD_2_0.8               - exemplary position, blade 2 at  80% normalized radius
BLD_2_0.5               - exemplary position, blade 2 at  50% normalized radius
BLD_2_0.4               - exemplary position, blade 2 at  40% normalized radius
BLD_2_0.2               - exemplary position, blade 2 at  20% normalized radius
BLD_2_0.0               - exemplary position, blade 2 at  00% normalized radius

BLD_3_1.0               - exemplary position, blade 3 at 100% normalized radius
BLD_3_0.8               - exemplary position, blade 3 at  80% normalized radius
BLD_3_0.5               - exemplary position, blade 3 at  50% normalized radius
```

```
BLD_3_0.4              - exemplary position, blade 3 at  40% normalized radius
BLD_3_0.2              - exemplary position, blade 3 at  20% normalized radius
BLD_3_0.0              - exemplary position, blade 3 at  00% normalized radius

TWR_1.00               - exemplary position, tower at 100% normalized height
TWR_0.90               - exemplary position, tower at  90% normalized height
TWR_0.80               - exemplary position, tower at  80% normalized height
TWR_0.70               - exemplary position, tower at  70% normalized height
TWR_0.60               - exemplary position, tower at  60% normalized height
TWR_0.50               - exemplary position, tower at  50% normalized height
TWR_0.40               - exemplary position, tower at  40% normalized height
TWR_0.30               - exemplary position, tower at  30% normalized height
TWR_0.20               - exemplary position, tower at  20% normalized height
TWR_0.10               - exemplary position, tower at  10% normalized height
TWR_0.00               - exemplary position, tower at   0% normalized height
```

The last part of the main structural input file deals with the definition of loading data and sensor locations. The locations at which the data will be stored a input file:

- `BLD_X_Y` : Stores data for blade X at the normalized curved length position Y
- `STR_X_Y_Z` : Stores data for strut Y of blade X at the normalized curved length position Z
- `TWR_X` : Stores data for the tower at the normalized curved length position X
- `TRQ_X` : Stores data for the torque tube at the normalized curved length position X
- `CAB_X_Y` : Stores data for guy cable X at the normalized curved length position Y

Furthermore data is automatically stored at each inter body connection of the model. Each inter body connection is identified by a combination of two bo model definition. In the following two exemplary auto-generated variable names are shown and explained:

Y I Mom. TRQ - BLD_3 z=29.7m

The moment around the local Y axis at the connection between the torque tube and blade 3, which was defined at a height of 29.7m. This result is giv

X I For. STR_2_2 - BLD_2 z=27.5m

This example defines the local reaction force at the connection between the top strut of blade 2 and blade 2, given for the local X axis of the strut.

Seven different data types can be specified to be stored (true) or not (false) at all locations that are specified or automatically generated. It is recommende memory requirements and size of project and data files generated by QBlade. The different types of data that can be stored for each sensor are:

true FOR_OUT - store forces at all sensor locations true DEF_OUT - store deflections at all sensor locations true POS_OUT - store positions at all sensor sensor locations true STR_OUT - store element strain at all sensor locations true AER_OUT - store aerodynamic data at all sensor locations

The forces and moments that obtained from a structural body are the **internal shear forces and bending moments**. However, the forces and moments giv constraint. For an overview of the coordinate systems / conventions in which the simulation results are stored see the section: Coordinate Systems.

Structural Definition of Bodies

For an aeroelastic wind turbine setup, each body in the multi-body setup is defined by its own structural data table. These datatables contain the crosssec body. The structural bodies that can be defined with structural datatables are: **blades**, **struts**, **tower**, **torquetube**, **cable elements** and the **substructure**. Di briefly explained below:

Euler-Bernoulli Beam

Euler-Bernoulli beams are the most basic type of beams in QBlade. These beams rely on the thin beam theory and thus do not consider shear forces. They displacements.

Timoshenko Beam

Timoshenko beams represent a more advanced beam model in QBlade compared to Euler-Bernoulli beams. These beams incorporate the effects of shear Timoshenko beams are implemented with a corotational formulation to accommodate large displacements and deflections while providing a more accurat

Timoshenko Beam FPM

Timoshenko beams with a Fully Populated Stiffness Matrix (FPM) represent the most sophisticated and versatile beam model in QBlade. Timoshenko FPM particularly important for an accurate modeling of very large blades. The Timoshenko FPM element is reserved to be used excludively to model rotor blad

ANCF Cable Element

The ANCF Cable element in QBlade is used for an efficient simulation of slender, cable like structures such as mooring lines and blade cables. These elem
mooring system configurations or tower guywires (see Cable Structural Data File and Mooring Elements and Ground-Constraints).

Blade, Strut and Tower Structural Data Files

The cross-sectional beam properties of the blade, tower and strut bodies have to be defined in the form of structural data tables. The definition of the tab
Timoshenko FPM Datatable and Tower and Torquetube Euler Bernoulli and Timoshenko Datatable. An exemplary structural blade data table for a Timoshe

Listing 17 : Exemplary blade structural data file for a Timoshenko b

```
0.0024  RAYLEIGHDMP

0       INTPTYPE 0-LINEAR; 1-AKIMA; 2-HERMITE; 3-C2SPLINE
1       BEAMTYPE 0-EULER; 1-TIMOSHENKO; 2-TIMOSHENKO_FPM
1       DISCTYPE 0-LINEAR; 1-COSINE; 2-STRUCT; 3-AERO
60      DISC

ADDMASS_0.50 0.00 - add a point mass at relative position 0.50 with 0.00kg mass

LENFRACT_[-]  MASSD_[kg/m]  EIx_[N.m^2]   EIy_[N.m^2]   EA_[N]        GJ_[N.m^2]    GA_[N]        STRPIT_[deg]  KSX_[-]       KSY_[-]       RGX_[-]
0.0000E+00    7.1502E+02    1.8116E+10    1.8116E+10    9.7300E+09    5.5600E+09    6.9500E+08    0.0000E+00    5.0000E-01    5.0000E-01    3.2931E
3.2520E-03    7.1502E+02    1.8116E+10    1.8116E+10    9.7300E+09    5.5600E+09    6.9500E+08    0.0000E+00    5.0000E-01    5.0000E-01    3.2931E
1.9512E-02    8.1446E+02    1.9418E+10    1.9558E+10    1.0790E+10    5.4300E+09    7.7070E+08    0.0000E+00    5.0000E-01    5.0000E-01    3.2685E
3.5772E-02    7.7991E+02    1.7458E+10    1.9502E+10    1.0067E+10    4.9900E+09    7.1910E+08    0.0000E+00    5.0000E-01    5.0000E-01    3.0601E
5.2033E-02    7.7937E+02    1.5288E+10    1.9782E+10    9.8672E+09    4.6700E+09    7.0480E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.8228E
6.8293E-02    6.2399E+02    1.0783E+10    1.4854E+10    7.6076E+09    3.4700E+09    5.4340E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.6375E
8.4553E-02    4.7421E+02    7.2296E+09    1.0220E+10    5.4908E+09    2.3200E+09    3.9220E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.4658E
1.0081E-01    4.4659E+02    6.3098E+09    9.1448E+09    4.9714E+09    1.9100E+09    3.5510E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.3129E
1.1707E-01    4.2193E+02    5.5286E+09    8.0626E+09    4.4940E+09    1.5700E+09    3.2100E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.1690E
1.3333E-01    4.0237E+02    4.9798E+09    6.8838E+09    4.0348E+09    1.1600E+09    2.8820E+08    0.0000E+00    5.0000E-01    5.0000E-01    2.0504E
1.4959E-01    4.2090E+02    4.9364E+09    7.0098E+09    4.0376E+09    1.0000E+09    2.8840E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.9141E
1.6585E-01    4.4898E+02    4.6914E+09    7.1680E+09    4.1692E+09    8.5600E+08    2.9780E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.7635E
1.8211E-01    4.3897E+02    3.9494E+09    7.2716E+09    4.0824E+09    6.7200E+08    2.9160E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.6368E
1.9837E-01    4.2777E+02    3.3866E+09    7.0812E+09    4.0866E+09    5.4700E+08    2.9190E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.5436E
2.1463E-01    4.0169E+02    2.9344E+09    6.2440E+09    3.6680E+09    4.4900E+08    2.6200E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.4756E
2.3089E-01    3.7157E+02    2.5690E+09    5.0484E+09    3.1472E+09    3.3600E+08    2.2480E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.4153E
2.4715E-01    3.6805E+02    2.3884E+09    4.9490E+09    3.0114E+09    3.1100E+08    2.1510E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.3776E
2.6341E-01    3.6496E+02    2.2722E+09    4.8076E+09    2.8826E+09    2.9200E+08    2.0590E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.3583E
2.9593E-01    3.5737E+02    2.0496E+09    4.5010E+09    2.6138E+09    2.6100E+08    1.8670E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.3211E
3.2846E-01    3.4754E+02    1.8284E+09    4.2434E+09    2.3576E+09    2.2900E+08    1.6840E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.2843E
3.6098E-01    3.3910E+02    1.5890E+09    3.9956E+09    2.1462E+09    2.0100E+08    1.5330E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.2363E
3.9350E-01    3.3050E+02    1.3619E+09    3.7506E+09    1.9446E+09    1.7400E+08    1.3890E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.1868E
4.2602E-01    3.1040E+02    1.1024E+09    3.4468E+09    1.6324E+09    1.4400E+08    1.1660E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.1139E
4.5854E-01    3.0238E+02    8.7584E+08    3.1388E+09    1.4322E+09    1.2000E+08    1.0230E+08    0.0000E+00    5.0000E-01    5.0000E-01    1.0343E
4.9106E-01    2.7734E+02    6.8124E+08    2.7342E+09    1.1687E+09    8.1200E+07    8.3480E+07    0.0000E+00    5.0000E-01    5.0000E-01    9.6993E
5.2358E-01    2.6666E+02    5.3466E+08    2.5550E+09    1.0475E+09    6.9100E+07    7.4820E+07    0.0000E+00    5.0000E-01    5.0000E-01    9.0303E
5.5610E-01    2.5451E+02    4.0894E+08    2.3338E+09    9.2302E+08    5.7500E+07    6.5930E+07    0.0000E+00    5.0000E-01    5.0000E-01    8.3338E
5.8862E-01    2.3236E+02    3.1458E+08    1.8284E+09    7.6076E+08    4.5900E+07    5.4340E+07    0.0000E+00    5.0000E-01    5.0000E-01    7.9830E
6.2114E-01    2.1094E+02    2.3870E+08    1.5848E+09    6.4806E+08    3.6000E+07    4.6290E+07    0.0000E+00    5.0000E-01    5.0000E-01    7.6068E
6.5366E-01    1.8894E+02    1.7584E+08    1.3234E+09    5.3970E+08    2.7400E+07    3.8550E+07    0.0000E+00    5.0000E-01    5.0000E-01    7.2179E
6.8618E-01    1.7387E+02    1.2601E+08    1.1837E+09    5.3116E+08    2.0900E+07    3.7940E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6939E
7.1870E-01    1.6262E+02    1.0725E+08    1.0202E+09    4.6004E+08    1.8500E+07    3.2860E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6508E
7.5122E-01    1.4632E+02    9.0874E+07    7.9786E+08    3.7576E+08    1.6300E+07    2.6840E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6749E
7.8374E-01    1.3644E+02    7.6314E+07    7.0966E+08    3.2886E+08    1.4500E+07    2.3490E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6198E
8.1626E-01    1.1296E+02    6.1054E+07    5.1814E+08    2.4402E+08    9.0700E+06    1.7430E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6835E
8.4878E-01    1.0403E+02    4.9476E+07    4.5486E+08    2.1154E+08    8.0600E+06    1.5110E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.6071E
8.8130E-01    9.5044E+01    3.9354E+07    3.9508E+08    1.8158E+08    7.0800E+06    1.2970E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.5143E
8.9756E-01    8.7412E+01    3.4664E+07    3.5378E+08    1.6030E+08    6.0900E+06    1.1450E+07    0.0000E+00    5.0000E-01    5.0000E-01    6.5499E
9.1382E-01    7.6781E+01    3.0408E+07    3.0478E+08    1.0923E+08    5.7500E+06    7.8020E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.7897E
9.3008E-01    7.2427E+01    2.6516E+07    2.8140E+08    1.0009E+08    5.3300E+06    7.1490E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.8201E
9.3821E-01    6.9786E+01    2.3842E+07    2.6166E+08    9.2246E+07    4.9400E+06    6.5890E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.8860E
9.4634E-01    6.2494E+01    1.9628E+07    1.5876E+08    6.3224E+07    4.2400E+06    4.5160E+06    0.0000E+00    5.0000E-01    5.0000E-01    7.0184E
9.5447E-01    5.8886E+01    1.6002E+07    1.3789E+08    5.3326E+07    3.6600E+06    3.8090E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.9485E
9.6260E-01    5.5273E+01    1.2830E+07    1.1879E+08    4.4534E+07    3.1300E+06    3.1810E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.8804E
9.7073E-01    5.1724E+01    1.0080E+07    1.0163E+08    3.6904E+07    2.6400E+06    2.6360E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.8277E
9.7886E-01    4.8253E+01    7.5502E+06    8.5064E+07    2.9918E+07    2.1700E+06    2.1370E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.6807E
9.8699E-01    4.3884E+01    4.6004E+06    6.4260E+07    2.1308E+07    1.5800E+06    1.5220E+06    0.0000E+00    5.0000E-01    5.0000E-01    6.1430E
9.9512E-01    1.2062E+01    2.5004E+05    6.6094E+06    4.8496E+06    2.5000E+05    3.4640E+05    0.0000E+00    5.0000E-01    5.0000E-01    5.4262E
1.0000E+00    1.0867E+01    1.6996E+05    5.0106E+06    3.5294E+06    1.9000E+05    2.5210E+05    0.0000E+00    5.0000E-01    5.0000E-01    4.4641E


RGBCOLOR
R       G       B
220     220     220
```

The keyword `RAYLEIGHDMP` : defines a stiffness proportional Rayleigh damping coefficient (see Damping of Structural Bodies). The parameters `STIFFTUNER` multiplication by this factor. The keyword `RGBCOLOR` defines the rgb values that are used to color the structural body during the 3D visualization.

The keyword `INTPTYPE` controls the interpolation of the cross-sectional quantities between the user specified data table and the structural ele Op

The keyword `BEAMTYPE` sets the type of structural beam, based on which the structural datatable is interpreted. Options are: 0-EULER; 1-TIMOSHENKO; (see Blade and Strut Euler Bernoulli and Timoshenko Datatable and Blade and Strut Timoshenko FPM Datatable)

The keyword `DISCTYPE` controls the discretization type of the structural body. Options are: 0-LINEAR; 1-COSINE; 2-STRUCT; 3-AERO. LINEAR is the sta COSINE is a cosine distribution based on the number of nodes specified by the keyword `<num> DISC`. STRUCT discretizes the structural body based on th aerodynamic blade design.

The keyword `<num> DISC` controls the number of structural nodes that are distributed over the length of the body:

The keyword `ADDMASS_<pos>` can be used to add a mass at the normalized position <*pos*>. `ADDMASS_<pos>` can be followed by up to 7 numeric values (at le mass of 10kg at the normalized position of 0.2. The following numbers assign the rotational inertia in local body coordinates: *Ixx = 1*, *Iyy = 2*, *Izz = 3*, *Ixy = 4*

Blade and Strut Euler Bernoulli and Timoshenko Datatable

The following table gives an overview of the entries of the structural data table for blades and struts. All entries reserved for modeling the shear stiffness

Table 1 Blade / Strut Cross Sectional Beam Properties for Euler-Bernoulli Beams

Col. Nr.	Name	Explanation	Unit
1	Length	Norm. curved length	-
2	Mass density	Mass per unit length	kg/m
3	Bend. stiff. X	Bending Stiffness around X_{ce} (EI_{xx})	Nm^2
4	Bend. stiff. Y	Bending Stiffness around Y_{ce} (EI_{yy})	Nm^2
5	Axial stiff.	Longitudinal Stiffness (EA)	N
6	Tors. stiff.	Torsional Stiffness (GJ)	Nm^2
7	Shear stiff.	Shear Stiffness (GA) (not used with Euler beams)	N
8	Str. pitch	Structural pitch angle between reference X and X_{ce} axis	deg
9	Shear factor X	Shear factor for force in principal bending axis X_{ce}	-
10	Shear factor Y	Shear factor for force in principal bending axis Y_{ce}	-
11	Radius of gyration X	Norm. radius of inertia corresponding to a rotation around X_{ce}	%chord
12	Radius of gyration Y	Norm. radius of inertia corresponding to a rotation around Y_{ce}	%chord
13	Center of mass X	Norm. center of mass position X	%chord
14	Center of mass Y	Norm. center of mass position Y	%chord
15	Center of elast. X	Norm. center of elasticity position X	%chord
16	Center of elast. Y	Norm. center of elasticity position Y	%chord
17	Center of shear X	Norm. center of shear position X	%chord
18	Center of shear Y	Norm. center of shear position Y	%chord
19	Damping Coefficient	**(optional)** This column allows to assign distributed Rayleigh beta coeff.	-

The radius of gyration r_g is related to the moment of inertia (I_{xx}, or I_{yy}) in the following way:

$$r_{g,x} = \sqrt{\frac{I_{xx}}{m}} = \sqrt{\frac{I_x}{A}}$$

Please not the the radius of gyration in the structural datatable furthermore is normalized by the local chord of the blade.

Blade and Strut Timoshenko FPM Datatable

The following table gives an overview of the entries of the structural data table for blades and struts:

Table 2 Blade / Strut Cross Sectional Beam Properties for Timoshenko FPM Beams

Col. Nr.	Name	Explanation	Unit
1	Length	Norm. curved length	-
2	Beam offset X	Offset in local x-direction (norm with c)	-
3	Beam offset Y	Offset in local y-direction (norm with c)	-
4	Pitch	Structural pitch, applied to matrix	deg
5	K11	(1,1) entry for the stiffness matrix	N
6	K12	(1,2) entry for the stiffness matrix	N
7	K13	(1,3) entry for the stiffness matrix	N
8	K14	(1,4) entry for the stiffness matrix	Nm
9	K15	(1,5) entry for the stiffness matrix	Nm
10	K16	(1,6) entry for the stiffness matrix	Nm
11	K22	(2,2) entry for the stiffness matrix	N
12	K23	(2,3) entry for the stiffness matrix	N
13	K24	(2,4) entry for the stiffness matrix	N
14	K25	(2,5) entry for the stiffness matrix	N
15	K26	(2,6) entry for the stiffness matrix	N
16	K33	(3,3) entry for the stiffness matrix	N
17	K34	(3,4) entry for the stiffness matrix	Nm
18	K35	(3,5) entry for the stiffness matrix	Nm
29	K36	(3,6) entry for the stiffness matrix	Nm
20	K44	(4,4) entry for the stiffness matrix	Nm^2
21	K45	(4,5) entry for the stiffness matrix	Nm^2
22	K46	(4,6) entry for the stiffness matrix	Nm^2
23	K55	(5,5) entry for the stiffness matrix	Nm^2
24	K56	(5,6) entry for the stiffness matrix	Nm^2
25	K66	(6,6) entry for the stiffness matrix	Nm^2
26	M11	(1,1) entry for the mass matrix	kg
27	M12	(1,2) entry for the mass matrix	kg
28	M13	(1,3) entry for the mass matrix	kg
29	M14	(1,4) entry for the mass matrix	kgm
30	M15	(1,5) entry for the mass matrix	kgm
31	M16	(1,6) entry for the mass matrix	kgm
32	M22	(2,2) entry for the mass matrix	kg
33	M23	(2,3) entry for the mass matrix	kg
34	M24	(2,4) entry for the mass matrix	kg
35	M25	(2,5) entry for the mass matrix	kg
36	M26	(2,6) entry for the mass matrix	kg
37	M33	(3,3) entry for the mass matrix	kg
38	M34	(3,4) entry for the mass matrix	kgm
39	M35	(3,5) entry for the mass matrix	kgm
40	M36	(3,6) entry for the mass matrix	kgm

Col. Nr.	Name	Explanation	Unit
41	M44	(4,4) entry for the mass matrix	kgm^2
42	M45	(4,5) entry for the mass matrix	kgm^2
43	M46	(4,6) entry for the mass matrix	kgm^2
44	M55	(5,5) entry for the mass matrix	kgm^2
45	M56	(5,6) entry for the mass matrix	kgm^2
46	M66	(6,6) entry for the mass matrix	kgm^2

Tower and Torquetube Euler Bernoulli and Timoshenko Datatable

The following table gives an overview of the entries of the structural data table:

Table 3 Tower / Torquetube Cross Sectional Beam Properties

Col. Nr.	Name	Explanation	Unit
1	Length	Norm. curved length	-
2	Mass density	Mass per unit length	kg/m
3	Bend. stiff. X	Bending Stiffness around X_{ce} (EI_{xx})	Nm^2
4	Bend. stiff. Y	Bending Stiffness around Y_{ce} (EI_{yy})	Nm^2
5	Axial stiff.	Longitudinal Stiffness (EA)	N
6	Tors. stiff.	Torsional Stiffness (GJ)	Nm^2
7	Shear stiff.	Shear Stiffness (GA) (not used with Euler beams)	N
8	Str. pitch	Structural pitch angle between reference X and X_{ce} axis	deg
9	Shear factor X	Shear factor for force in principal bending axis X_{ce}	-
10	Shear factor Y	Shear factor for force in principal bending axis Y_{ce}	-
11	Radius of gyration X	Norm. radius of inertia corresponding to a rotation around X_{ce}	%chord
12	Radius of gyration Y	Norm. radius of inertia corresponding to a rotation around Y_{ce}	%chord
13	Center of mass X	Norm. center of mass position X	%chord
14	Center of mass Y	Norm. center of mass position Y	%chord
15	Center of elast. X	Norm. center of elasticity position X	%chord
16	Center of elast. Y	Norm. center of elasticity position Y	%chord
17	Center of shear X	Norm. center of shear position X	%chord
18	Center of shear Y	Norm. center of shear position Y	%chord
19	Diameter	Cross section diameter	m
20	Drag	**(optional)** Drag coefficient for aerodynamic drag	-
21	Damping Coefficient	**(optional)** This column allows to assign distributed Rayleigh beta coeff.	-

The radius of gyration r_g is related to the moment of inertia (I_{xx}, or I_{yy}) in the following way:

$$r_{g,x} = \sqrt{\frac{I_{xx}}{m}} = \sqrt{\frac{I_x}{A}}$$

Please not the the radius of gyration in the structural datatable furthermore is normalized by the local diameter of the tower or torquetube.

Cable Structural Data File

Listing 18 : Exemplary cable definition data file

```
----------------------------CABLE DATA-------------------------------

CABELEMENTS
CabID   MASS_[kg/m]     EIy_[N.m^2]     EA_[N]          DAMP_[-]        DIA_[m]
1       1.574300E+00    6.755490E+02    4.222260E+07    0.002           0.016
2       9.048000E-01    1.964547E+02    2.182830E+07    0.002           0.012

CABMEMBERS
ID      CONN_1          CONN_2          Tension[N]      CabID   Drag    ElmDsc  Name
1       STR_1_1_0.0     STR_1_1_1.0     70000           1       0       2       B1StrutBot
2       STR_2_1_0.0     STR_2_1_1.0     70000           1       0       2       B1StrutTop
3       STR_1_1_1.0     TRQ_0.9631      15000           2       0.99    2       B1TieRod3
4       STR_2_1_1.0     TRQ_0.2839      15000           2       0.99    2       B1TieRod1
5       STR_1_2_0.0     STR_1_2_1.0     70000           1       0       2       B2StrutBot
6       STR_2_2_0.0     STR_2_2_1.0     70000           1       0       2       B2StrutTop
7       STR_1_2_1.0     TRQ_0.9631      15000           2       0.99    2       B2TieRod3
8       STR_2_2_1.0     TRQ_0.2839      15000           2       0.99    2       B2TieRod1
9       STR_1_3_0.0     STR_1_3_1.0     70000           1       0       2       B2StrutBot
10      STR_2_3_0.0     STR_2_3_1.0     70000           1       0       2       B2StrutTop
11      STR_1_3_1.0     TRQ_0.9631      15000           2       0.99    2       B2TieRod3
12      STR_2_3_1.0     TRQ_0.2839      15000           2       0.99    2       B2TieRod1
```

Cables can be defined between blades (BLD), struts (STR), the tower (TWR), torquetube (TRQ) or the ground (GRD).

`CABDAMP`

In some cases, if the alpha damping coefficient of a cable element (CABELEMENTS) is too large, a simulation can become unstable. Therefore, be defa damping of mooring lines and guy cables, the keyword `CABDAMP` must be set to true.

Listing 19 : Activating axial cable damping for all CABELEMENTS be setting the

```
true CABDAMP
```

Damping of Structural Bodies

Two different damping models exists, which can be used to define the damping properties of a structural body.

Isotropic Rayleigh Damping

A single Rayleigh damping coefficient can be set for each structural data table by using the keyword `RAYLEIGHDMP`. This keyword defined the *stiffness prop* structural body:

$$C = beta * K,$$

where C the stiffness matrix. The Rayleigh damping $beta$ coefficient is related to the fraction of critical damping Xi as:

$\zeta = \beta * \pi * f$, or

$\beta = \frac{\zeta}{\pi * f}$.

Rayleigh damping is not constant, but varies with frequency. Typically, Rayleigh damping is set for the first natural frequency of a component. Optionally, and Strut Euler Bernoulli and Timoshenko Datatable).

Anisotropic Rayleigh Damping

For a more detailed definition of the damping properties of a structural body the anisotropic damping model is recommended. This damping model allows body. The anisotropic damping of a body is defined by at least four parameters (and an optional fifth parameter), followed by the keyword `RAYLEIGHDMP_AN`

Listing 20 : Exemplary definition of anisotropic damping properties in a struc

```
0.004048 0.003153 0.00027325 0.00000 0.00000 RAYLEIGHDMP_ANISO
```

The five parameters are related to the anisotropic damping in the following way:

- **1:** The stiffness proportional β Rayleigh damping coefficient for bending about the local y-axis (flapwise) or shear along the local x-axis.
- **2:** The stiffness proportional β Rayleigh damping coefficient for bending about the local x-axis (edgewise) or shear along the local y-axis.

- **3:** The stiffness proportional β Rayleigh damping coefficient for bending about the local z-axis (torsion).
- **4:** The stiffness proportional β Rayleigh damping coefficient along the local z-axis (elongation).
- **5:** (optional) A mass proportional α Rayleigh damping coefficient, applied to all degrees of freedom (0.00 as default).

In the same way as the isotropic stiffness proportional Rayleigh damping coefficients, the Rayleigh damping $beta$ coefficients are related to the fraction o

$\zeta = \beta * \pi * f$, or

$\beta = \frac{\zeta}{\pi * f}$.

Cross Sectional Coordinate Systems

The local cross-sectional coordinate system for the definition of the blade and strut structural data table is shown in Fig. 87.

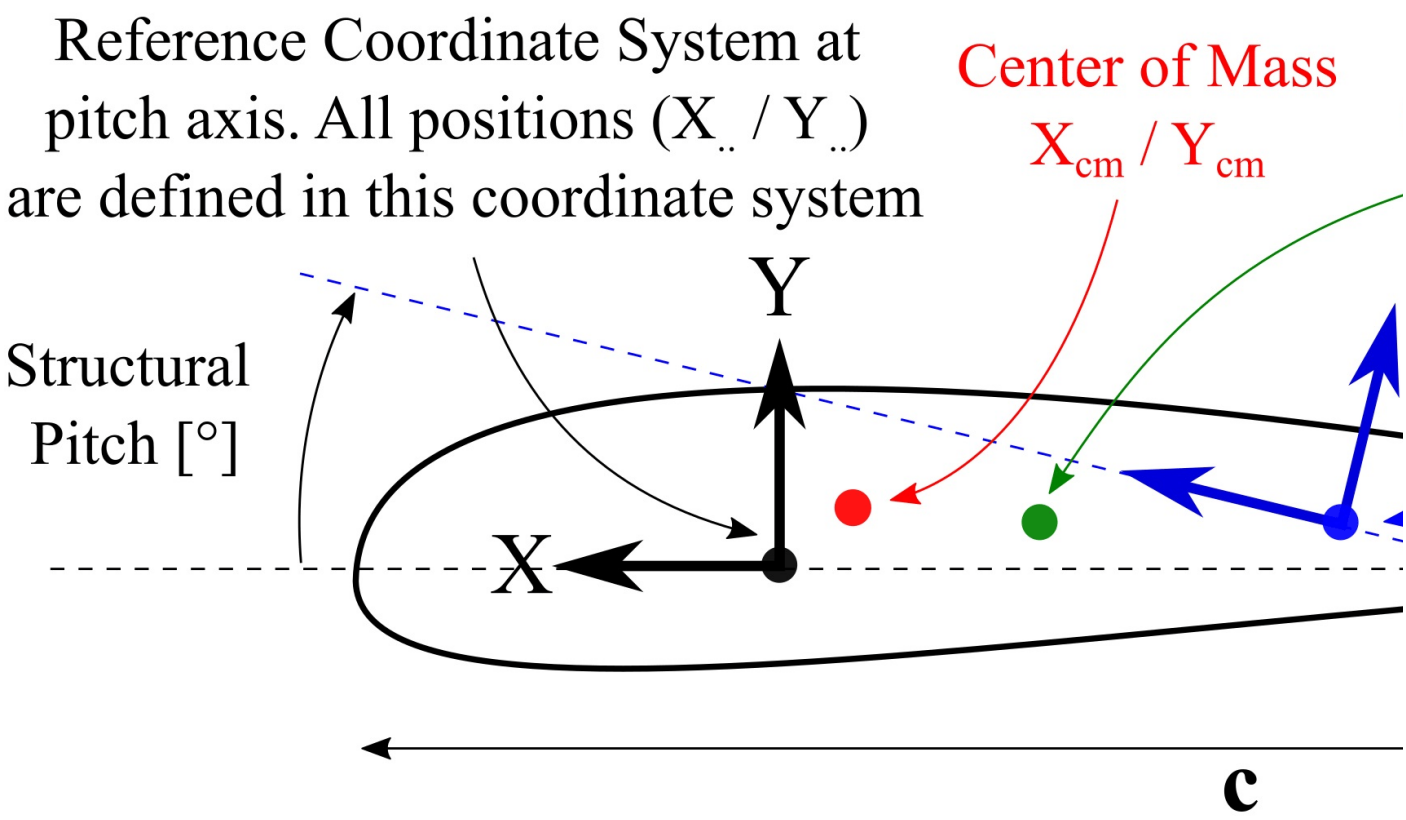


Fig. 87 *Visualization of the local coordinate system that is used to define the cross sectiona*

Please note: this cross sectional coordinate system is **ONLY** used for the definition of the **blade** and **strut** sectional structural properties! This cross sectio Blade Coordinate System), which is used to report the loading data of the blade or strut. The local blade (and strut) coordinate system, used to report load sectional coordinate system shown in Fig. 87 in the following way:

- the **local blade X-axis** points in the direction of the **cross sectional Y-axis**.
- the **local blade Y-Axis** points in the direction **opposite the cross sectional X-axis**.
- the **local blade Z-Axis** points along the blade **principal axis** towards the blade tip.

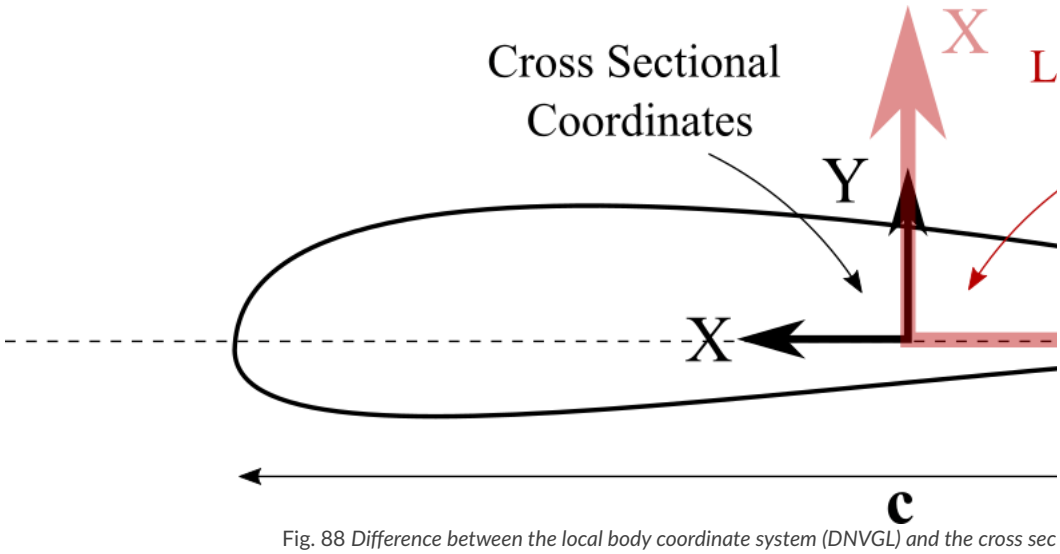


Fig. 88 *Difference between the local body coordinate system (DNVGL) and the cross sec*

For all other structural bodies (tower, torquetube, substructure) the coordinate system in which the cross sectional structural properties are defined coinc

[1] DNVGL. Guideline for the Certification of Wind Turbines. 2010.

Marine Hydrokinetic Turbines

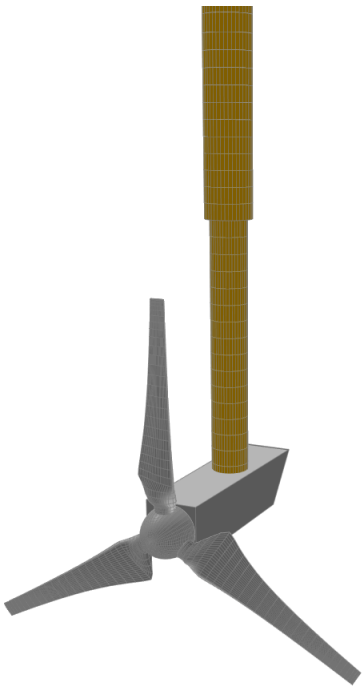


Fig. 89 *A marine hydrokinetic turbine designed in QBlade.*

QBlade also provides the capability to model Marine Hydrokinetic (MHK) Turbines that are operating in water instead of air. When changing from air to water factors such as buoyancy, added mass, and inertia forces become increasingly important. Properly and consistently modeling these effects requires following a specific set of steps which are presented in the sections below.

Simulation Settings for MHK Turbines

MHK turbines are treated as if the were *onshore* turbines within the modeling framework. This involves substituting the density and viscosity values of air with those corresponding to water to ensure accurate physical properties throughout the model.

- In the simulation settings, Turbine Environment set the Installation to **Onshore**.
- Change the **Air Density** value to the respective value for water.
- Change the **Kinematic Viscosity Air** value to the value for water.

Blade and Tower Model Settings for MHK Turbines

To assign **Added Mass** and **Dynamic Pressure** coefficients to the Blade, Strut and Tower Stru
Data Files, specific keywords can simply be added to the structural data table files associate

these components. Additionally, it's possible to activate **Buoyancy** for these components using a dedicated keyword. Below are the relevant keywords explained:

The keyword `ADDEDMASSCOEFF` is employed to assign an added mass coefficient to the blade. This coefficient becomes valuable when modeling marine hydrokinetic turbines (MHK), contributing to the calculation of hydrodynamic inertia force and added-mass force in a Morison-style approach.

For added mass calculations, the blade sections are approximated as flat plates, while the tower and torquetube are assumed to have cylindrical shapes.

Listing 21 : Exemplary Added Mass Keyword use

```
1.2 ADDMASSCOEFF
```

The keyword `DYNPRESSURECOEFF` is used to assign a dynamic pressure coefficient to the blade. This coefficient plays a role in modeling marine hydrokinetic turbines (MHK) and contributes to the Morison-style hydrodynamic inertia force calculation.

Listing 22 : Exemplary Dynamic Pressure Coefficient Keyword use

```
1.2 DYNPRESSURECOEFF
```

The keyword `ISBUOYANCY` activates buoyancy calculations for the structure to which it is applied. This accounts for the buoyancy force acting on the cross-sectional area of a blade or tower section.

Listing 23 : Exemplary Buoyancy Keyword use

```
true ISBUOYANCY
```

Substructure Model Settings for MHK Turbines

In the Substructure Modeling section, **Morison Coefficients** and **Buoyancy** can be assigned to various components of the substructure, following the usual procedure (see Morison Equation (Strip Theory) Modelling). In this context the whole substructure (and turbine) is considered to be *submerged* in water. During the simulation of an *onshore* installed turbine, these coefficients will be used in conjunction with the **Air Density** value that was previously replaced with the value for water (see Simulation Settings for MHK Turbines).

Turbine Definition ASCII File

Turbine objects can be exported as a QBlade project file (`.qpr`) or into the text based `.trb` format. When a turbine object is exported into the `.trb` for (`.bld`) file is automatically created. Furthermore, the structural definition files (if the turbine has a structural model) and the controller parameters are als folder structure See an exemplary `.trb` file below:

```
----------------------------------------QBlade Turbine Definition File-------------------------------------------
Generated with : QBlade IH v2.0.2_alpha windows
Archive Format: 310002
Time : 12:07:32
Date : 29.06.2022

---------------------------------------Object Name---------------------------------------------------------------
NREL_5MW_Servo_OC3                      OBJECTNAME          - the name of the turbine object

---------------------------------------Rotor Definition----------------------------------------------------------
Aero/NREL_5MW.bld                       BLADEFILE           - the path of the blade file that is used in this turbine definition
0                                       TURBTYPE            - the turbine type (0 = HAWT or 1 = VAWT)
3                                       NUMBLADES           - the number of blades (a Structural Model overrides this value)
0                                       ROTORCONFIG         - the rotor configuration (0 = UPWIND or 1 = DOWNWIND)
0                                       ROTATIONALDIR       - the direction of rotor rotation (0 = STANDARD or 1 = REVERSED)
1                                       DISCTYPE            - type of rotor discretization (0 = from Bladetable, 1 = linear, 2 = cosine)
20                                      NUMPANELS           - the number of aerodynamic panels per blade (unused if DISCTYPE = 0)

---------------------------------------Turbine Geometry Parameters-----------------------------------------------
These values are only used if no Structural Model is defined for this Turbine, in case of a Structural Model the geometry is defined in the Structu
10.5000                                 OVERHANG            - the rotor overhang [m] (HAWT only)
0.0000                                  SHAFTTILT           - the shaft tilt angle [deg] (HAWT only)
0.0000                                  ROTORCONE           - the rotor cone angle [deg] (HAWT only)
0.0000                                  CLEARANCE           - the rotor clearance to ground [m] (VAWT only)
0.0000                                  XTILT               - the rotor x-tilt angle [deg] (VAWT only)
0.0000                                  YTILT               - the rotor y-tilt angle [deg] (VAWT only)
126.0000                                TOWERHEIGHT         - the tower heigh [m]
1.8000                                  TOWERTOPRAD         - the tower top radius [m]
2.5200                                  TOWERBOTRAD         - the tower bottom radius [m]

---------------------------------------Aerodynamic Models--------------------------------------------------------
0                                       DYNSTALLTYPE        - the dynamic stall model: 0 = none; 1 = OYE; 2 = GORMONT-BERG or 3 = ATEFLAP
8                                       TF_OYE              - Tf constant for the OYE dynamic stall model
6                                       AM_GB               - Am constant for the GORMONT-BERG dynamic stall model
3                                       TF_ATE              - Tf constant for the ATEFLAP dynamic stall model
2                                       TP_ATE              - Tp constant for the ATEFLAP dynamic stall model
1                                       2PLIFTDRAG          - include the 2 point lift drag correction? (0 = OFF or 1 = ON)
0                                       HIMMELSKAMP         - include the Himmelskamp Stall delay? (0 = OFF or 1 = ON) (HAWT only)
0                                       TOWERSHADOW         - include the tower shadow effect (0 = OFF or 1 = ON)
1                                       TOWERDRAG           - the tower drag coefficient [-] (if a Structural Model is used the tower drag is defir

---------------------------------------Wake Type-----------------------------------------------------------------
1                                       WAKETYPE            - the wake type: 0 = free vortex wake; 1 = unsteady BEM (unsteady BEM is only available

---------------------------------------Vortex Wake Parameters----------------------------------------------------
Only used if waketype = 0
0                                       WAKEINTTYPE         - the wake integration type: 0 = EF; 1 = PC; 2 = PC2B
1                                       WAKEROLLUP          - calculate wake self-induction (0 = OFF or 1 = ON)
1                                       TRAILINGVORT        - include trailing vortex elements (0 = OFF or 1 = ON)
1                                       SHEDVORT            - include shed vortex elements (0 = OFF or 1 = ON)
0                                       CONVECTIONTYPE      - the wake convection type (0 = BL, 1 = HH, 2 = LOC)
1.00                                    WAKERELAXATION      - the wake relaxation factor [0-1]
1.00                                    FIRSTWAKEROW        - first wake row length [-]
200000                                  MAXWAKESIZE         - the maximum number of wake elements [-]
100                                     MAXWAKEDIST         - the maxmimum wake distance from the rotor plane (normalized by dia) [-]
0.00100                                 WAKEREDUCTION       - the wake reduction factor [-]
0                                       WAKELENGTHTYPE      - the wake length type (0 = counted in rotor revolutions, 1 = counted in time steps)
1000000.00                              CONVERSIONLENGTH    - the wake conversion length (to particles) [-]
0.50                                    NEARWAKELENGTH      - the near wake length [-]
2.00                                    ZONE1LENGTH         - the wake zone 1 length [-]
4.00                                    ZONE2LENGTH         - the wake zone 2 length [-]
6.00                                    ZONE3LENGTH         - the wake zone 3 length [-]
2                                       ZONE1FACTOR         - the wake zone 1 factor (integer!) [-]
2                                       ZONE2FACTOR         - the wake zone 2 factor (integer!) [-]
2                                       ZONE3FACTOR         - the wake zone 3 factor (integer!) [-]

---------------------------------------Vortex Core Parameters----------------------------------------------------
Only used if waketype = 0
1.00                                    BOUNDCORERADIUS     - the fixed core radius of the bound blade vortex (fraction of local chord) [0-1]
0.05                                    WAKECORERADIUS      - the intial core radius of the free wake vortex (fraction of local chord) [0-1]
800.00                                  VORTEXVISCOSITY     - the turbulent vortex viscosity
0                                       VORTEXSTRAIN        - calculate vortex strain 0 = OFF, 1 = ON
20                                      MAXSTRAIN           - the maximum element strain, before elements are removed from the wake [-]

---------------------------------------Gamma Iteration Parameters------------------------------------------------
Only used if waketype = 0
0.40                                    GAMMARELAXATION     - the relaxation factor used in the gamma (circulation) iteration [0-1]
0.00100                                 GAMMAEPSILON        - the relative gamma (circulation) convergence criteria
```

```
100                                     GAMMAITERATIONS    - the maximum number of gamma (circulation) iterations (integer!) [-]

---------------------------------------Unsteady BEM Parameters----------------------------------------------------
12                                      POLARDISC          - the polar discretization for the unsteady BEM (integer!) [-]
0                                       BEMTIPLOSS         - use BEM tip loss factor, 0 = OFF, 1 = ON
0.00                                    BEMSPEEDUP         - initial BEM convergence acceleration time [s]

---------------------------------------Structural Model-----------------------------------------------------------
Structure/OC3_Sparbuoy_Main_LPMD.str    STRUCTURALFILE     - the input file for the structural model (leave blank if unused)
0                                       GEOMSTIFFNESS      - enable geometric stiffness, 0 = OFF, 1 = ON

---------------------------------------Turbine Controller---------------------------------------------------------
3                                       CONTROLLERTYPE     - the type of turbine controller 0 = none, 1 = BLADED, 2 = DTU, 3 = TUB
TUBCon_1.3.9_64Bit                      CONTROLLERFILE     - the controller file name, WITHOUT file ending (.dll or .so ) - leave blank if unused
Control/TUBCon_Params_V1.3.9_NREL5MW.xml PARAMETERFILE     - the controller parameter file name (leave blank if unused)
```

Multi Rotor Turbine Assembly

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

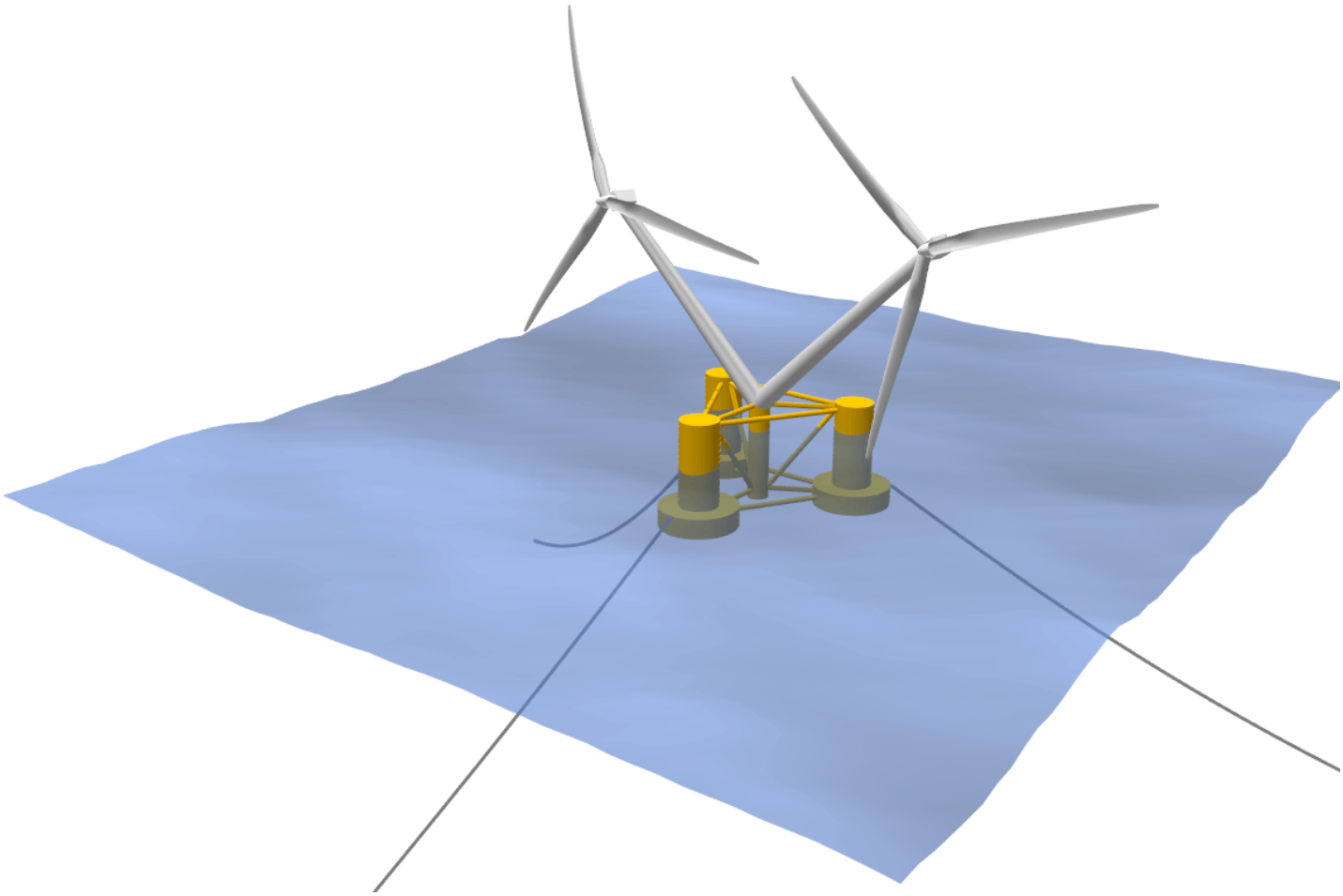


Fig. 90 *Visualization of a Multi-Rotor Turbine Assembly.*

A turbine with multiple rotors may be defined in the dialog *Menu->Turbine Definition->Create a Multi-Rotor Turbine Assembly*. A Multi-Rotor Assembly requires the definition of a common substructure, see Substructure Overview. The common substructure definition must then contain multiple transition pieces (**TP_INTERFACE_POS**) that may have different orientations (**TP_ORIENTATION**). If a common substructure is loaded into the Multi Turbine Assembly dialog (see Fig. 91) a turbine from QBlades database (that contains its own structural definition and controller) can be assigned to each of these transition pieces. Multiple transition pieces can be defined by adding **_2, _3, _4, ..., _N** to the respective keywords, such as **TP_INTERFACE_POS_2** and **TP_ORIENTATION_2**, where the appendix **_1** can optionally be omitted for the first transition piece and all associated keywords. For any additional transition piece **TP_INTERFACE_POS_X**, the following keywords may be defined:

- **REF_COG_POS_X**
- **REF_HYDRO_POS_X**
- **TP_ORIENTATION_X**
- **SUB_MASS_X**
- **SUB_HYDROADDEDMASS_X**
- **SUB_HYDROSTIFFNESS_X**
- **SUB_HYDRODAMPING_X**
- **SUB_HYDROQUADDAMPING_X**
- **SUB_HYDROCONSTFORCE_X**
- **POT_RAD_FILE_X**
- **POT_EXC_FILE_X**
- **POT_SUM_FILE_X**
- **POT_DIFF_FILE_X**

To connect to a specific transition piece in the **SUBCONSTRAINTS** table simply use the number **X** of that transition piece in the **TrPID** column.

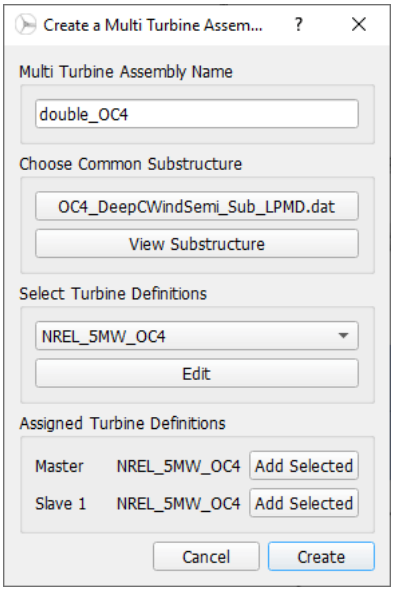


Fig. 91 *The Multi-Rotor Assembly Dialog.*

Multi Rotor Turbine Assembly ASCII File

A Multi Turbine Assembly can be exported or imported in the `.mta` format. The file content points towards the common substructure file and towards th turbine (`.trb`) files that are used in the multi-rotor assembly. See an exemplary `.mta` file below:

```
--------------------------------------QBlade Multi Turbine Assembly Definition File------------------------------
Generated with : QBlade IH v2.0.2_alpha windows
Archive Format: 310003
Time : 18:25:50
Date : 04.07.2022

---------------------------------------Object Name-----------------------------------------------------------
double_OC4                             OBJECTNAME         - the name of the multi-rotor turbine object

---------------------------------------Assembly Definition---------------------------------------------------
OC4_DeepCWindSemi_Sub_LPMD.dat         SUBSTRUCTURE       - the path of the common substructure file that is used in this multi turbine assembly
NREL_5MW_OC4.trb                       MASTER             - the master turbine of the assembly
NREL_5MW_OC4.trb                       SLAVE_1            - the slave turbine(s) of the assembly
```

Wind Turbine Controllers

QBlade allows the integration of standard wind turbine controllers to perform aero-servo-hydro-elastic simulations. This is realized via the coupling to a dynamic link library (.dll) which is called every time step by QBlade to update control actions performed on the wind turbine. This is shown in Fig. 92.

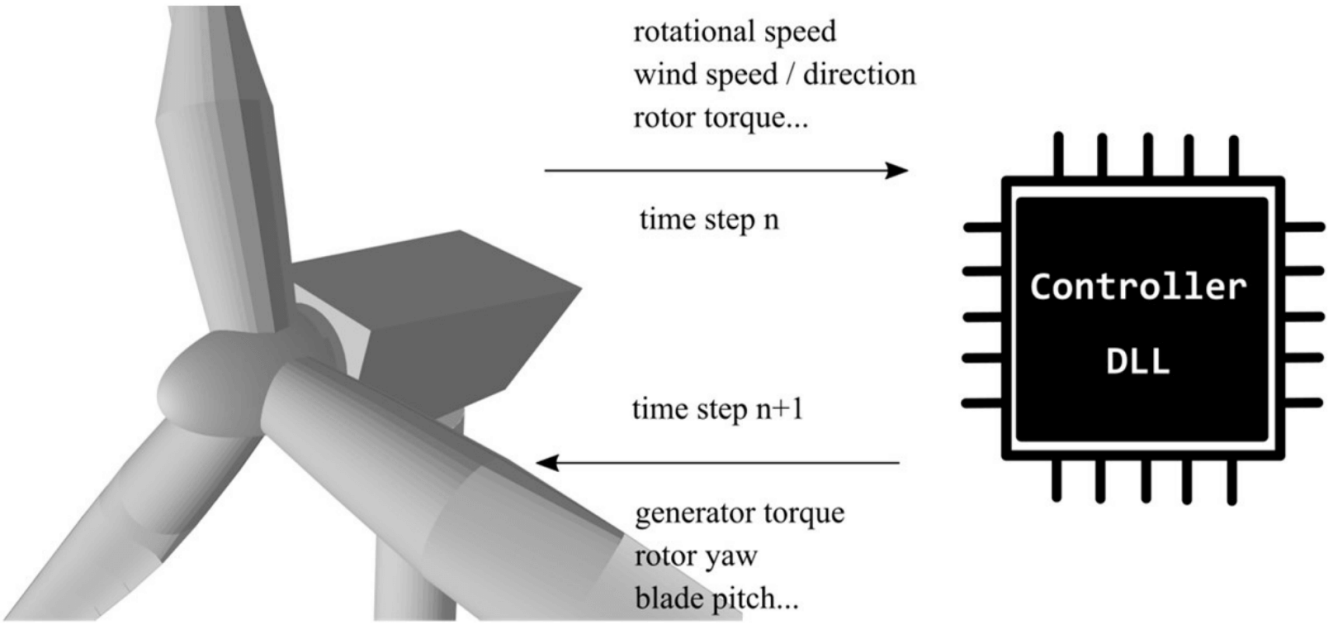


Fig. 92 *Controller interaction with QBlade: For each time step, QBlade passes the sensor input and retrieves the control actions demanded from the controller in a predetermined swap array.*

Several commonly used controller interfaces are compatible with QBlade. The different types of controller interfaces that QBlade is compatible with are:

- **Bladed** interface: the function *DISCON* is called
- **DTU** interface: the function *update_regulation* is called
- **TUB** interface: the function *TUBController* is called

Since QBlade is compiled as a 64bit software it is **only possible to call 64bit compiled controller libraries from QBlade**. Open source examples of these three formats are available online. An example for the Bladed-style controller is the NREL[1] controller.

An example for a DTU-style controller is the DTU[2] controller. An example for the TUB-Style controller is the TUB Controller presented in Perez-Becker *et al.*[3]. The QBlade release contains pre-compiled ROSCO, DTU and TUB Controllers.

For these commonly used interfaces the data that is passed between the controller and the simulation in QBlade and its position in the swap array that is used for this communication is already predefined. This means that there are fixed array positions for data such as torque, rpm, pitch angles and tower top accelerations. The specific data that is communicated and its position depends on the controller interface definition and is different between the BLADED-style, DTU-style and TUB-style interfaces.

External Library Interface

QBlade also allows to integrate multiple custom dynamic libraries into a simulation. The purpose of a custom library could be the control of an active damping device, the control of a mooring cable length or any other active system which influence can be modeled by the application of a force, moment, a change in mass, pitch torque or a change in mooring line length.

The structure of the External Library Interface consists of an arbitrarily named *update()* function that is automatically called during every timestep of the simulation (after rampup has been completed). A swap array of floats of arbitrary size is passed to the *update()* function and retrieved from the update function. The content of the swap array can be freely assigned by the user from all variables that are evaluated from a simulation. In addition the user can also define to what actions should be performed on the turbine, based on the content of the swap array. A minimum working example for the source code of the controller (with fucus only on the function definitions) is shown in the section Example for a custom controller library in C.

In Passing Custom Data to an External Library and Passing External Library Data to the Turbine it is explained how the communication with a controller can be customized.

Adding a Controller or an External Library to a Turbine Definition

A controller library can be included in a turbine definition by selecting the library in the dialog shown in Fig. 93. Depending on which controller interface type is used the appropriate option has to be selected. Furthermore, a controller parameter file has to be selected and loaded by the user. The parameter file is stored in the QBlade project and can be edited by the user once loaded. This if for example useful for setting up an identical turbine with a modified controller parameter (i.e. for control parameter tuning). Controller parameter files edited within QBlade can also be exported to ASCII format.

Any number of custom external libraries can be loaded in the lower part of the dialog shown in Fig. 93. When loading a custom library the user also has to specify the function name that should be called by QBlade as well as the swap array size that will be used for communication between the library and QBlade. Each external library received an integer identifier, which is later used to pass and receive data to and from its swap array.

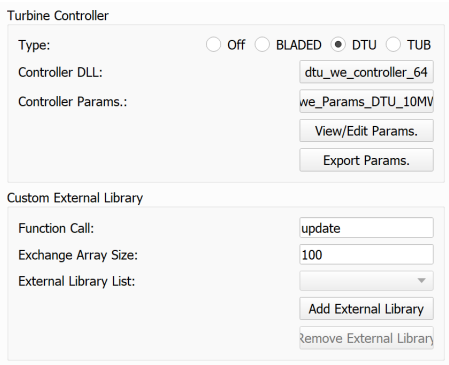


Fig. 93 *The controller dialog.*

Below is an example for the *Turbine Controllers* and *External Libraries* sections in a Turbine Definition ASCII File (.trb). For a predefined controller the controller type, the controller file and its parameter file have to be defined. The three parameters that need to be passed for the external library are the file name (*type2_dll*), the function name, in this case *update2* and the swap array size.

Listing 24 : customDll.cpp

```
---------------------------------------Turbine Controller---------------------------------------------------------------
3                                     CONTROLLERTYPE     - the type of turbine controller 0 = none, 1 = BLADED, 2 = DTU, 3 = TUB
TUBCon_1.3.9_64Bit                    CONTROLLERFILE     - the controller file name, WITHOUT file ending (.dll or .so ) - leave blank if unused
Control/TUBCon_Params_V1.3.9_NREL5MW.xml PARAMETERFILE      - the controller parameter file name (leave blank if unused)

---------------------------------------External Libraries---------------------------------------------------------------
type2_dll                             LIBFILE_1          - the library file name, WITHOUT file ending (.dll or .so )
update2                               LIBFUNCTION_1      - the libary function name that should be called every timestep
100                                   LIBARRAYSIZE_1     - the library swap array size for data exchange
```

IMPORTANT:

- The controller library (.dll or .so) must always be located in the folder path .*ControllerFiles* relative to the QBlade executable.
- QBlade is a 64bit code, so the controller dlls must also be compiled for 64bit use.

Passing Custom Data to a Controller

In addition to the standard predefined sensor information that is passed between QBlade and the controller, the user can add additional sensors to the swap array. This is useful for the development of specialized controllers that rely on unconventional sensorial input. Depending on the controller format chosen, different positions of the swap array will be unoccupied.

IMPORTANT: The user should know which array entries are unused before adding custom sensors. Otherwise, using this option will lead to unwanted turbine behavior!

To add custom sensors to the swap array of a *Predefined Controller*, the following table needs to be added to the controller parameter file, the substructure file or the structural model main file (up to the user to decide what is most convenient). The search order in the files is:

1. structural main file
2. substructure file
3. controller parameter stream

The box below shows an exemplary **CONTROLLER_IN** table. The first column contains the swap array position and the second column the variable name in quotation marks. In the example below the table assigns the variable *Time [s]* to swap array position [0], the variable *Timestep [s]* to array position [1]

and the variable *X_l Acc. BLD_1 pos 1.000* to array position [10]. Note that the full variable name, as shown in any of QBlade's graphs must be included in the table. If the variable name does not exists (or the data is not stored as part of the simulation) no value is passed to the swap array at the designated position. Also note that the data defined in this table overwrites the standard data that is usually passed to the predefined controller interface.

Listing 25 : CONTROLLER_IN Table

```
CONTROLLER_IN
0  "Time [s]"
1  "Timestep [s]"
10 "X_l Acc. BLD_1 pos 1.000"
```

Passing Custom Data to an External Library

Passing custom data to an external library is the same process as passing data to a predefined controller, only the keyword for the table changes. In this case we are passing data to the external controller *1*, indicated by the keyword *EXTERNAL_1_IN*. To pass to the second external controller you would use the keyword *EXTERNAL_2_IN*.

Listing 26 : EXTERNAL_1_IN Table

```
EXTERNAL_1_IN
0 "Time [s]"
1 "Timestep [s]"
```

Note that the output of the desired sensor should be enabled in the Main Structural Definition File. Otherwise, only zeros will be passed to the controller.

Passing Custom Controller Data to the Turbine

Swap array data from predefined controller interfaces is automatically applied to perform specific control actions, such as the application of generator torque on the model or control of the blade pitch. The functionality presented hereafter allows to *wire* data from the controller swap array to specific actions performed on the turbine.

This functionality is very similar to the function `SetExternalAction()`, from QBlade's Software in Loop Interface (SIL), that is described in the Interface Function Documentation. It allows to assign forces, moments, masses and other actions to a turbine in a highly flexible manner.

Again, a table is defined to gather the controller data from the controller swap array. The keyword for this table is **CONTROLLER_OUT**. The table has 7 columns. An examplary table is shown below:

Listing 27 : CONTROLLER_OUT Table

```
CONTROLLER_OUT
50      SETLENGTH       MOO_1   1.0     X       true
50      SETLENGTH       MOO_2   1.0     X       true
```

Below an overview of the six columns is given:

- **1:** The swap array **DATA** that will be applied
- **2:** The **ACTION** that will be performed, based on the data
- **3:** The **ID** at which the action will be performed
- **4:** The **POSITION** at which the action will be performed
- **5:** The **DIRECTION** in which the action is being performed.
- **6:** The **ISLOCAL** whether the direction is defined in local element coordinates or in global coordinates

The different columns are now further defined:

DATA

In this column the swapArray index is selected from which data will be used to perform an action.

Action

Different actions can be performed, these are:

- ADDMASS: adds mass of **DATA** to a location, in [kg]
- ADDFORCE: adds a force of **DATA** to a location, in [N]
- ADDTORQUE: adds a torque of **DATA** to a location, in [Nm]

- SETLENGTH: sets the delta Length of **DATA** of a cable, in [m]
- SETAFC: sets the state of **DATA** of an AFC element [-]
- SETTORQUE: sets the generator torque of **DATA**, in [Nm]
- SETYAW: sets the yaw angle of **DATA**, in [deg]
- SETPITCH: sets the pitch angle of **DATA** for BLD_X, in [deg]
- SETBRAKE: sets the brake modulation of **DATA** [0-1]

ID

The ID is used to identify a certain turbine component, possible IDs and actions that can be performed on them are shown below:

- CAB_<X>: applies the action to the guycable with ID <X>. Actions on cables are: SETLENGTH, ADDMASS, ADDFORCE
- MOO_<X>: applies the action to the mooring line with ID <X>. Actions on moorings are: SETLENGTH, ADDMASS, ADDFORCE
- SMOO_<X>: applies the action to the shared mooring line with ID <X>. Actions on moorings are: SETLENGTH, ADDMASS, ADDFORCE
- TRQ: applies the action to the torquetube. Actions on the torquetube are: ADDFORCE, ADDTORQUE, ADDMASS
- BLD_<X>: applies the action to blade <X>. Actions on the blades are: ADDFORCE, ADDTORQUE, ADDMASS
- STR_<X>_<Y>: applies the action to strut <X> of blade <Y>. Actions on the struts are: ADDFORCE, ADDTORQUE, ADDMASS
- AFC_<X>_<Y>: applies the action to AFC <X> of blade <Y>. Actions on the AFC elements are: SETAFC
- SUB_<X>: applies the action to the substructure element with ID <X>. Actions on the substructure elements are: ADDFORCE, ADDTORQUE, ADDMASS
- JNT_<X>: applies the action to the substructure joint with ID <X>. Actions on the substructure joints are: ADDFORCE, ADDTORQUE, ADDMASS
- HUB: applies the action to the free LSS hub node. Actions on the hub node are: ADDFORCE, ADDTORQUE, ADDMASS
- HUBFIXED: applies the action to the fixed non-rotating hub node. Actions on the hub node are: DDFORCE, ADDTORQUE, ADDMASS

POSITION

Sets the normalized position [0-1] at which the mass, force or torque is applied. Only has an effect on elements, not on nodes.

DIRECTION

Specifies the direction along which the force or torque is applied, options are “X”, “Y”, “Z”.

ISLOCAL

Specifies sets whether the direction is defined in global or local (element or node) coordinates.

Passing External Library Data to the Turbine

Passing custom data from an external library library to the turbine is the same as passing this data from a predefined controller with the exception that th keyword of the table changes to **EXTERNAL_<num>_OUT**, where **<num>** is to be replaced by the library integer ID.

Listing 28 : EXTERNAL_1_OUT Table

```
EXTERNAL_1_OUT
50      SETLENGTH       MOO_1   1.0     X       true
50      SETLENGTH       MOO_2   1.0     X       true
```

Example for a custom controller library in C

The example below shows the source code of a simple external controller library in C-language. Remember that this library should be compiled as **64bit** t be compatible with QBlade.

Listing 29 : customDll.cpp

```
1  #include <stdio.h>
2
3  bool firstCall = true;
4  double value;
5  char message_out[1000];
6
7  //this should be the function that QBlade calls at every timestep. The function name can be assigned
8  //in QBlade turbine setup dialog or in the respective section of the .trb file
9  extern "C" void __declspec(dllexport) __cdecl update(float *avrSwap){
10
11     if (firstCall){
12         //this is an example how the external controller could be initialized
13         sprintf(message_out,"First call, do some initialization things! Timestep = %f",avrSwap[1]);
14         firstCall = false;
15     }
16
17     //this is an example how some value is computed from the data in the swap array and then
18     //returned in the same swap array at position [50]
```

```
    sprintf(message_out,"Successive call, do some calculation things! Time = %f",avrSwap[0]);
    avrSwap[50] = avrSwap[0]*(-1.0);

}

//this function should have the same name as the function above with "_message" appended to it
//if this function is defined QBlade calls it automatically to print the output that "update"
//passes to the message_out variable
extern "C" void __declspec(dllexport) __cdecl update_message(char *message){

    sprintf(message,message_out);
}
```


[1] NREL. ROSCO. https://github.com/nrel/rosco, 2022. [Online; accessed 2022-05-09].

[2] DTU. DTUWEC. https://gitlab.windenergy.dtu.dk/OpenLAC/BasicDTUController, 2022. [Online; accessed 2022-05-09].

[3] S. Perez-Becker, D. Marten, C. N. Nayeri, and C. O. Paschereit. Implementation and Validation of an Advanced Wind Energy Controller in Aero-Servo-Elastic Simulations Using the Lifting Line Free Vortex Wake Model. *Energies*, 14(3):783, 2021. URL: https://www.mdpi.com/1996-1073/14/3/783, doi:10.3390/en14030783.

Substructure Overview

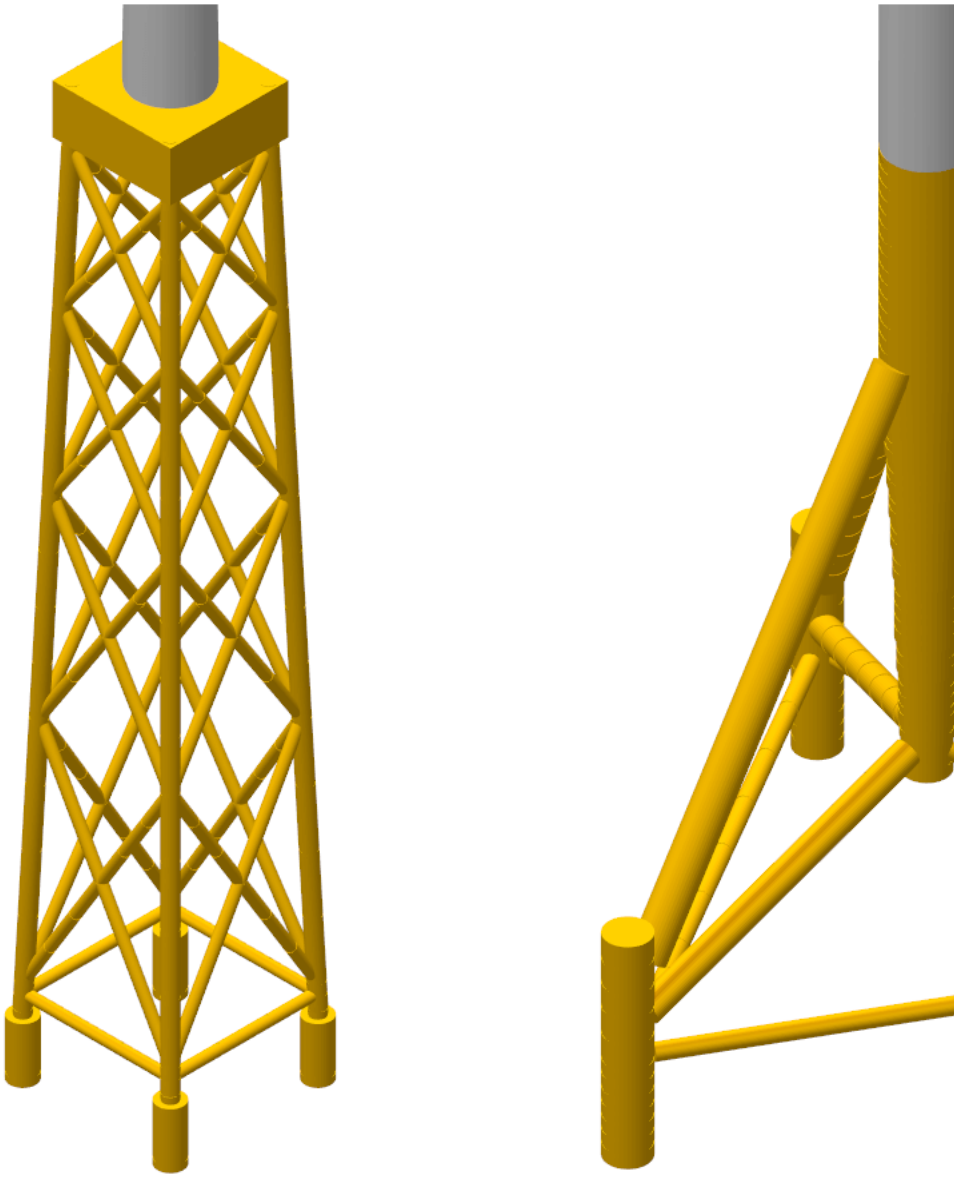


Fig. 94 *Three diffe*

Optionally, a substructure definition can be added to the structural model of a wind turbine in QBlade. The substructure definition can fulfill several differ

- **Lattice towers:** The substructure can be used to model the turbine tower with greater freedom as the standard tower file, which can only model tu
- **Floaters and moorings**: The substructure can be used to define the floater of a floating offshore wind turbine (FOWT). This requires that the floate which keeps the floater place.
- **Soil modeling:** In the substructure file a p-y curve can be defined to model nonlinear soil dynamics, both onshore or offshore.
- **Multi-rotor assemblies:** The substructure definition is also used to define the substructure of a multi-rotor assembly, with an arbitrary number of r

To add a substructure to a turbine definition you need to add the filename of the substructure definition followed by the keyword `SUBFILE` anywhere wit

It is possible to connect the substructure either to the tower bottom, torquetube bottom or directly to the rotor nacelle assembly (RNA) of a turbine defin `TWRFILE` keyword the transition piece is connected to the rotor nacelle assembly (RNA).

Modeling Options for an Offshore Substructure

When it comes to modeling the hydrodynamics and structural dynamics of an offshore substructure, there are several options to consider. This is

Substructure Mass and Inertia

The substructure is typically composed of interconnected members and joints (see Substructure Topology). These members can be modeled as either flexi complete substructure's mass can be represented using a 6x6 mass matrix, known as *lumped mass* (see Lumped Mass, Inertia and Hydrodynamic Forces).

Substructure Buoyancy

Similar to mass and inertia, substructure buoyancy can be modeled explicitly or in a lumped linearized manner. Explicit modeling involves enabling the buc approach, a 6x6 hydrodynamic stiffness matrix is defined to represent the substructure's buoyancy restoring forces and moments (see Lumped Mass, Iner

Substructure Hydrodynamics

The hydrodynamic forces acting on a substructure can be evaluated by means of linear potential flow theory or the Morison equation. When the Morison applied locally to each member. Alternatively, a linear potential flow solver (such as WAMIT, NEMOH, or ANSYS AQWA) can generate databases represer mass matrices (see Lumped Mass, Inertia and Hydrodynamic Forces).

Different Scenarios

It is possible to combine different approaches mentioned above. A common mix involves using linear potential flow hydrodynamics along with the Moriso explicit buoyancy modeling), and the substructure members should be modeled as flexible elements. By carefully selecting and combining these modeling

Keywords and Tables

As with the other structural definition files, the substructure is defined by a series of keywords that are recognized by QBlade when creating the turbine.

A parameter is defined by its value followed by the parameter *Keyword*:

- <*Value*> `KEYWORD`, for parameters defined by a single values.

```
Value   KEYWORD
```

A table is identified by its *Keyword* and the row and column count of the subsequent ASCII values, which need to separated by *space(s)* or *tab(s)*. An examp

- `KEYWORD` <new line> <*Header*> <new line> <*Values*> for parameters defined by a table. The <*Header*> <new line> part is only optional and can be omit

```
KEYWORD
Header1         Header2         Header3         ...
Value(1,1)      Value(1,2)      Value(1,3)      ...
Value(2,1)      Value(2,2)      Value(2,3)      ...
...             ...             ...             ...
```

There is no particular order in which these keywords and the associated data tables should be placed. The only exception is when defining tables. When a

Miscellaneous Substructure Parameters

The following keywords can be used to define different properties and modeling options for the substructure.

`ISFLOATING`

A flag that determines if the substructure is floating of bottom-fixed. If the structure is bottom-fixed the joint coordinates (see `SUBJOINTS` below) are a

`WATERDEPTH`

Sets the design water depth of the substructure, this value is only used for visualization of the turbine and the identification of flooded members durir

`WATERDENSITY`

Sets the water density to calculate the mass of the flooded members. Note that this water density is only for the turbine setup and is not used during

`SEABEDDISC`

Sets the sub-discretization length for mooring lines in contact with the seabed, in [m]. A value of 1 means that when a mooring line element is in conta

`CONSTRAINEDFLOATER`

A flag that if set to true constrains the floater. A constrained floater can be subjected to a prescribed motion via a *Prescribed Motion File* (see

`BUOYANCYTUNER`

A multiplication factor that affects the calculation of the explicit buoyancy forces. Buoyancy caused by the linear hydrodynamic stiffness matrix is not

`ADVANCEDBUOYANCY`

An option to use an advanced discretization technique to calculate the explicit buoyancy of partially submerged members, especially useful if non-vert integer number (a value of 100 is suggested).

`STATICBUOYANCY`

An optional flag that controls for which sea level the explicit buoyancy is calculated in QBlade. If set to true, the buoyancy is considering only the mea database (`USE_EXCITATION`) it is recommended to enable the `STATICBUOYANCY` option since the hydrodynamic forces due to a change in wave elevation for.

`MARINEGROWTH`

A table that allows the user to define different types of marine growth that is present in the members. In QBlade, marine growth is simulated as an add growth.

```
MARINEGROWTH
ID      Thickn  Density
1       0.1     1100
```

`TRANSITIONBLOCK`

Adds a rectangle between the substructure and the tower base. It is used just for visualization purposes.

```
TRANSITIONBLOCK
WIDTH   LENGTH  HEIGHT
12      12      4
```

`TRANSITIONCYLINDER`

Adds a cylinder between the substructure and the tower base. It is used just for visualization purposes.

```
TRANSITIONCYLINDER
HEIGHT  DIAMETER
0.5     6.5
```

`RGBCOLOR`

Defines the color of the complete substructure. It is used just for visualization purposes.

```
RGBCOLOR
Red     Green   Blue
255     200     15
```

Substructure Topology

In general, a substructure consists of **Members** that are defined between **Joints**. A **Member** is a cylindrical or rectangular element that connects two **Node** defined between the same joint(s), these members are rigidly connected through the common joints (see Fig. 95).

Fig. 95 *Three c*

Substructure Joints

Joints are defined via the `SUBJOINTS` table. A joint is defined by its position and optionally by its orientation. In most cases it is sufficient to define only th
mass using the `ADDMASS_<JntID>` keyword.

Fig. 96 *Thre*

`SUBJOINTS`

Defines a table that is used to place spatial joints that help define the members of the substructure. Each row of the table defines one joint and has fo

The table is structured as follows:

```
SUBJOINTS
JntID JntX[m]     JntY[m]     JntZ[m]     X1      Y1      Z1      X2      Y2      Z2
 1     0.00000     0.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
 2     0.00000     0.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
 3    14.43376    25.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 4    14.43376    25.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 5   -28.86751     0.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 6   -28.86751     0.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 7    14.43376   -25.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 8    14.43376   -25.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 9    14.43375    25.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
10   -28.86750     0.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
11    14.43375   -25.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
12     9.23760    22.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
13   -23.67130     3.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
14   -23.67130    -3.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
15     9.23760   -22.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
16    14.43375   -19.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
17    14.43375    19.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
18     4.04145    19.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
19   -18.47520     6.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
20   -18.47520    -6.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
```

`SUBJOINTS` **(orientation defined by y- and y-axes)**

Defines a table that is used to place spatial joints that help define the members of the substructure. Each row of the table defines one joint and has fo

The values X1, Y1, Z1, X2, Y2 and Z2 are optional and can be used to define the local coordinate axes of the joint. X1, Y1 and Z1 are defining the vect
joint orientation is X1, Y1, Z1 = (1,0,0) and X2, Y2, Z2 = (0,1,0). If the user wants to define joint orientations they have to be defined for each joint in t

The table is structured as follows:

Listing

```
SUBJOINTS
JntID JntX[m]     JntY[m]     JntZ[m]     X1      Y1      Z1      X2      Y2      Z2
 1     0.00000     0.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
 2     0.00000     0.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
 3    14.43376    25.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 4    14.43376    25.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 5   -28.86751     0.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 6   -28.86751     0.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 7    14.43376   -25.00000   -14.00000    1.00    0.00    0.00    0.00    1.00    0.00
 8    14.43376   -25.00000    12.00000    1.00    0.00    0.00    0.00    1.00    0.00
 9    14.43375    25.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
10   -28.86750     0.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
```

```
11    14.43375   -25.00000   -20.00000    1.00    0.00    0.00    0.00    1.00    0.00
12     9.23760    22.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
13   -23.67130     3.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
14   -23.67130    -3.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
15     9.23760   -22.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
16    14.43375   -19.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
17    14.43375    19.00000    10.00000    1.00    0.00    0.00    0.00    1.00    0.00
18     4.04145    19.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
19   -18.47520     6.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
20   -18.47520    -6.00000   -17.00000    1.00    0.00    0.00    0.00    1.00    0.00
```

`SUBJOINTS` **(orientation defined by Euler angles))**

An alternative way to define the orientation of the substructure joints is to define the orientation of each joint by means of three consecutive Euler ro three columns are also optional, if not defined the orientation of each joint is the same as the global coordinate system.

Listing

```
SUBJOINTS
JntID JntX[m]     JntY[m]     JntZ[m]          RotX[deg]       RotY[deg]       RotZ[deg]
 1     0.00000     0.00000   -20.00000          0.00            0.00            0.00
 2     0.00000     0.00000    10.00000          0.00            0.00            0.00
 3    14.43376    25.00000   -14.00000          0.00            0.00            0.00
 4    14.43376    25.00000    12.00000          0.00            0.00            0.00
 5   -28.86751     0.00000   -14.00000          0.00            0.00            0.00
 6   -28.86751     0.00000    12.00000          0.00            0.00            0.00
 7    14.43376   -25.00000   -14.00000          0.00            0.00            0.00
 8    14.43376   -25.00000    12.00000          0.00            0.00            0.00
 9    14.43375    25.00000   -20.00000          0.00            0.00            0.00
10   -28.86750     0.00000   -20.00000          0.00            0.00            0.00
11    14.43375   -25.00000   -20.00000          0.00            0.00            0.00
12     9.23760    22.00000    10.00000          0.00            0.00            0.00
13   -23.67130     3.00000    10.00000          0.00            0.00            0.00
14   -23.67130    -3.00000    10.00000          0.00            0.00            0.00
15     9.23760   -22.00000    10.00000          0.00            0.00            0.00
16    14.43375   -19.00000    10.00000          0.00            0.00            0.00
17    14.43375    19.00000    10.00000          0.00            0.00            0.00
18     4.04145    19.00000   -17.00000          0.00            0.00            0.00
19   -18.47520     6.00000   -17.00000          0.00            0.00            0.00
20   -18.47520    -6.00000   -17.00000          0.00            0.00            0.00
```

`JOINTOFFSET`

Defines a table that can be used to apply a global offset to the positions of all `SUBJOINTS`. Note that the offset is only applied to the joints and not the

The table is structured as follows:

```
JOINTOFFSET
XOFF    YOFF    ZOFF
10      0       0
```

`ADDMASS_<JntID>`

can be used to add a mass at a joint *<JntID>*. `ADDMASS_<JntID>` can be followed by up to 7 numeric values (at least one) to assign mass and rotational ir *= 4*, *Ixz = 5*, *Iyz = 6*.

```
ADDMASS_5 10 1 2 3 4 5 6
```

`ADDFORCE_<JntID>`

can be used to add a constant force, defined in the global coordinate system, to a joint *<JntID>*. `ADDFORCE_<JntID>` can be followed by up to 6 numeric defined in the global coordinate system.

```
ADDFORCE__5 10 1 2 3 4 5 6
```

`ADDFORCELOC_<JntID>`

can be used to add a constant force, defined in the local joint coordinate system, to a joint *<JntID>*. `ADDFORCELOC_<JntID>` can be followed by 6 n moments are defined in the local joint coordinate system.

```
ADDFORCELOC_5 10 1 2 3 4 5 6
```

Substructure Elements

Four different types of element exits that can be used to construct the substructure geometry in the `SUBMEMBERS` table. Each element definition, identifiec
rigid elements.

Fig. 97 *A cylindı*

`SUBELEMENTS`

Defines a table that defines flexible cylindrical elements that can be used for the substructure definition. Each row represents one (cylindrical) elemen
structural parameters of the element. The entry placement is very similar to the blade and tower structural element table (see Blade, Strut and Tower S

1. The first entry is used to indicate the ID number of the element (ElemID).
2. The last (20th) entry is used to indicate the Rayleigh damping of the element.

```
SUBELEMENTS
ElemID  MASS_[kg/m]     Eix_[N.m^2]     Eiy_[N.m^2]     EA_[N]          GJ_[N.m^2]      GA_[N]          STRPIT_[deg]    KSX_[-]         KSY_[-]
1       4.7868E+03      6.7007E+13      6.7007E+13      1.2805E+13      5.0380E+13      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
2       1.7668E+04      8.4228E+14      8.4228E+14      4.7263E+13      4.7260E+13      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
3       3.5424E+04      6.7890E+15      6.7890E+15      9.4764E+13      5.1050E+15      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
4       6.8297E+02      5.7201E+11      5.7201E+11      1.8271E+12      4.3010E+11      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
```

`SUBELEMENTSRIGID`

Defines a table that defines rigid elements that will be used for the substructure definition. Each row represents one (cylindrical) element, which is def
below.

```
SUBELEMENTSRIGID
ElemID  BMASSD  DIAMETER
1       1       6.5
2       1       12
3       1       24
4       1       1.6
```

Fig. 98 *A rectangulaı*

`SUBELEMENTS_RECT`

Defines a table that defines rectangular flexible elements that will be used for the substructure definition. Each row represents one (rectangular) eleme and its local y-axis (*YDIM* column 20) need to be specified, instead of the cylindrical diameter. Thus, two additional values are required and the Rayleig (if a `HYDROJOINTCOEFF` is defined for one of the members end nodes).

```
SUBELEMENTS_RECT
ElemID  MASS_[kg/m]     Eix_[N.m^2]     Eiy_[N.m^2]     EA_[N]          GJ_[N.m^2]      GA_[N]          STRPIT_[deg]    KSX_[-]         KSY_[-]
1       4.7868E+03      6.7007E+13      6.7007E+13      1.2805E+13      5.0380E+13      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
2       1.7668E+04      8.4228E+14      8.4228E+14      4.7263E+13      4.7260E+13      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
3       3.5424E+04      6.7890E+15      6.7890E+15      9.4764E+13      5.1050E+15      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
4       6.8297E+02      5.7201E+11      5.7201E+11      1.8271E+12      4.3010E+11      0.0000E+00      0.0000E+00      5.0000E-01      5.0000E
```

`SUBELEMENTSRIGID_RECT`

Defines a table that defines rectangular rigid elements that will be used for the substructure definition. Each row represents one (rectangular) element end faces. When setting up the substructure, one `SUBELEMENTRIGID_RECT` definition can be used for several `SUBMEMBERS` (see below). An exemplary tabl

```
SUBELEMENTSRIGID_RECT
ElemID  BMASSD  XDIM    YDIM    DIA
1       1       2       6       1
2       1       3       1       1
3       1       5       5       1
4       1       4       2       1
```

`STIFFTUNER`

A multiplication factor that affects the stiffness of the flexible elements defined in `SUBELEMENTS`.

`MASSTUNER`

A multiplication factor that affects the mass density of ALL elements defined in `SUBELEMENTS`.

Substructure Members

The members of the substructure are defines within the `SUBMEMBERS` table. Each line in the table generated one element that is defined by an element def growth, flooded area and the discretization for each member.

`SUBMEMBERS`

Defines a table that contains the members that make up the turbine substructure. A member, with the ID **MemID**, is defined between two entries of tl rotate the member around its principal axis. Rotations are entered in degree. Additionally, it can have one Morison force coefficients group (**HyCoID**) c (**FldArea**). The member can be subdivided into smaller elements for a more accurate structural and hydrodynamic evaluation. This is done in the **MemI** (0 = False, 1 = True). Finally, the member can be optionally named for easier recognition in the output tables (**Name**). The last three optional columns c

The keyword table has the following format:

```
SUBMEMBERS
MemID   Jnt1ID  Jnt2ID  ElmID   ElmRot  HyCoID  IsBuoy  MaGrID  FldArea MemDisc Name                    Red     Green   Blue
 1      1       2       1       0       3       1       0       0       2       Main_Column             100     200     100
 2      45      4       2       0       4       1       0       0       2       Upper_Column_1          100     200     100
 3      46      6       2       0       4       1       0       0       2       Upper_Column_2          100     200     100
 4      47      8       2       0       4       1       0       0       2       Upper_Column_3          100     200     100
29      3       45      2       0       4       1       0       0       2       Upper_Column_flooded_1  100     200     100
30      5       46      2       0       4       1       0       0       2       Upper_Column_flooded_2  100     200     100
31      7       47      2       0       4       1       0       0       2       Upper_Column_flooded_3  100     200     100
 5      48      3       3       0       5       1       0       0       2       Base_Column_1           100     200     100
 6      49      5       3       0       5       1       0       0       2       Base_Column_2           100     200     100
 7      50      7       3       0       5       1       0       0       2       Base_Column_3           100     200     100
26      42      48      3       0       5       1       0       0       2       Base_column_flooded_1   100     200     100
27      43      49      3       0       5       1       0       0       2       Base_column_flooded_2   100     200     100
28      44      50      3       0       5       1       0       0       2       Base_column_flooded_3   100     200     100
```

Substructure Constraints

When multiple members are connected to the same joint these members are *rigidly* constrained through this common joint. By using the `SUBCO` `TS` which connects to the turbine structure. The joints can be constrained along any of their degrees of freedom (DoF). Furthermore, it is possible to constrai

`SUBCONSTRAINTS`

Defines the table that defines the constraints between two joints that are not already connected by members, constraints of joints to the ground or to

Each row of the table has 12 entries. The first entry defines the constraint ID number (**CstID**). The next entry define the joint which shall be constraine to the ground by setting the **GrdCon** column to 1. A joint can **either** be constrained to a second joint (**JntCon**), to the transition piece (**TpCon**) point or

The sixth entry specifies that the constraint is realized as a non-linear spring-damper element (defined via an the spring ID number). If no spring or dan translational and three rotational degrees of freedom. For these entries 0 is interpreted as unconstrained (free) and 1 is interpreted as constrained. A s

The coordinate system for these constraints is defined by the type that JointID1 is connected to. If Joint1ID is connected to Joint2ID, or the transition

An exemplary `SUBCONSTRAINTS` table is shown below. In this example all joints in the table are connected directly to the transition piece.

```
SUBCONSTRAINTS
CstID   JntID   JntCon  TpCon   GrdCon  Spring  DoF_X   DoF_Y   DoF_Z   DoF_rX  DoF_rY  DoF_rZ
1       2       0       1       0       0       1       1       1       1       1       1
2       24      0       1       0       0       1       1       1       1       1       1
3       26      0       1       0       0       1       1       1       1       1       1
4       28      0       1       0       0       1       1       1       1       1       1
8       30      0       1       0       0       1       1       1       1       1       1
9       32      0       1       0       0       1       1       1       1       1       1
10      34      0       1       0       0       1       1       1       1       1       1
14      12      0       1       0       0       1       1       1       1       1       1
15      14      0       1       0       0       1       1       1       1       1       1
16      16      0       1       0       0       1       1       1       1       1       1
```

Note that at least one joint of the substructure members `SUBMEMBERS` should be constrained to the transition piece (defined by `TP_INTERFACE_POS`), to

Connections to a Second Transition Piece

A joint can be connected to any created transition piece by entering number of the `TP_INTERFACE_POS_<X>` into the *TpCon* column.

Connections to the Torquetube

When building a floater for a vertical axis wind turbine (VAWT) the user also has the option to connect a joint to the bottom of a rotating torquetu bottom of the second turbine insert -2.

It is also possible to connect a joint to the top of the torquetube of any turbine, to do this subtract 100 from the value inserted in the *TpCon* colum

Connections to the Tower Top

Connections to the tower top are realized in a similar way as connections to the torquetube top. By adding 100 into column *TpCon*. So to connect t

Changes to in SUBCONSTRAINTS QBlade 2.0.7

In QBlade versions prior to 2.0.7 the constraint degrees of freedom were always defined in the coordinate system of the object to which Jnt1ID was cor latter case the coordinate system that is used is the global coordinate system.

The Transition Piece

The transition piece is the reference position in the substructure definition that defines the interface between the turbine definition and the substructure contains a `TWRFILE` the transition piece of the substructure (`TP_INTERFACE_POS`) is automatically connected to the tower bottom. If the substructure conta

transition piece.

`TP_INTERFACE_POS_<X>`

Defines the (x,y,z) coordinates (in m) of the position of the transition piece location of the substructure. It is defined as the point where the substructu

- For floating substructures it is defined in (x,y,z) [m] from the MSL = (0,0,0).
- For bottom fixed substructures, it is defined from the seabed.

Note that the inertia and hydrodynamic reference points (`REF_COG_POS` and `REF_HYDRO_POS`) are always automatically constrained to this point (see Lin `TP_INTERFACE_POS_2`). This allows the user to define additional inertia and hydrodynamic reference points (see Linear Potential Flow Modelling). If a mu turbine 2 and so on.All transition piece points can be constrained to a joint of the substructure in the `SUBCONSTRAINTS` table. The structure of the table

```
TP_INTERFACE_POS
X[m]            Y[m]            Z[m]
0               0               10
```

Note: for the 1st `TP_INTERFACE_POS_<X>` the numbering **_1** can be omitted, so TP1 can be defined by the keyword `TP_INTERFACE_POS` . This is also true f

`TP_ORIENTATION_<X>` **(orientation defined by x- and y -axes)**

Defines the orientation of the tower base or RNA coordinate system which is connected to the `TP_INTERFACE_POS_<X>` by defining its X_t- and Y_t-Axis is not specified the default values are $X_{tp} = (1, 0, 0)$ and $Y_{tp} = (0, 1, 0)$, so the tower base coordinate system is aligned with the global coordinate s

```
TP_ORIENTATION
X[m]            Y[m]            Z[m]
1               0               0
0               1               0
```

`TP_ORIENTATION_<X>` **(orientation defined by Euler angles)**

An alternative way of defining the orientation of the tower base or RNA is to specify the orientation by means of three Euler angles. Starting from the

```
TP_ORIENTATION
Rot_X[deg]      Rot_Y[deg]      Rot_Z[deg]
30              0               0
```

Lumped Mass, Inertia and Hydrodynamic Forces

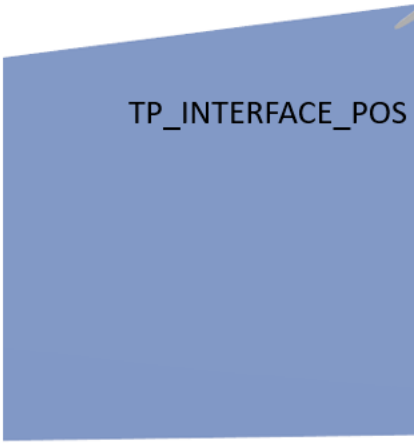


Fig. 100 *Main reference points for the substructure. The inertia reference point* `RI`

For each transition piece multiple reference position exist, which are rigidly constrained with the transition piece. These reference points can be used to a the forces from *linear potential flow* data are applied to the substructure (see Linear Potential Flow Modelling).

`REF_COG_POS_<X>`

defines the (x,y,z) position (in m) of a inertia point of the system (i.e. the center of gravity). It is in this position that the `SUB_MASS` matrix is evaluated. T

```
REF_COG_POS
X[m]            Y[m]            Z[m]
0               0               -13.46
```

`SUB_MASS_<X>`

defines a complete 6 by 6 mass and rotational inertia matrix that is placed in the location defined by the `REF_COG_POS_<X>` keyword. The units are kg f

```
SUB_MASS
1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   6.82700e+09   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   6.82700e+09   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   1.22600e+10
```

`REF_HYDRO_POS_<X>`

defines the (x,y,z) position (in m) of a hydrodynamic evaluation point of the system (i.e. where the lumped hydrodynamic forces are applied). It is in this constrained to the `TP_INTERFACE_POS_<X>` point, so no additional constraints are necessary to attach this point to the substructure. It has the ing

```
REF_HYDRO_POS_1
X[m]            Y[m]            Z[m]
0               0               -10.00
```

SUB_HYDROSTIFFNESS_<X>

defines a complete 6 by 6 stiffness matrix that is evaluated in the location defined by the `REF_HYDRO_POS_<X>` keyword. The units are N/m, N/rad, Nm/

```
SUB_HYDROSTIFFNESS_1
0               0               0               0               0               0
0               0               0               0               0               0
0               0               3.32941e+05     0               0               0
0               0               0               -4.99918e+09    0               0
0               0               0               0               -4.99918e+09    0
0               0               0               0               0               9.834e+07
```

SUB_HYDRODAMPING_<X>

defines a complete 6 by 6 damping matrix that is evaluated in the location defined by the `REF_HYDRO_POS_<X>` keyword. The units are N/(m/s), N/(rad/s

SUB_HYDROQUADDAMPING_<X>

defines a complete 6 by 6 quadratic damping matrix that is evaluated in the location defined by the `REF_HYDRO_POS_<X>` keyword. The units are N/(m/s

SUB_HYDROADDEDMASS_<X>

defines a complete 6 by 6 added mass matrix that is evaluated in the location defined by the `REF_HYDRO_POS_<X>` keyword. The units are kg. This matri

SUB_CONSTFORCE_<X>

applies a constant force (and/or torque) to the `REF_HYDRO_POS_<X>` point. It can be used to e.g. model the constant buoyancy force acting on the floate

```
SUB_HYDROCONSTFORCE_1 //the constant hydrodynamic buoyancy (and other forces,moments)
0               0               8.07081e+07     0               0               0
```

SUB_DISPLACEDVOLUME_<X>

applies a constant force in the global z-direction to the `REF_HYDRO_POS_<X>` point that is calculated based on the displaced water volume given by the u
`SUB_CONSTFORCE_<X>` entries, but can be used without specifying `SUB_CONSTFORCE_<X>`.

Mooring Elements and Ground-Constraints

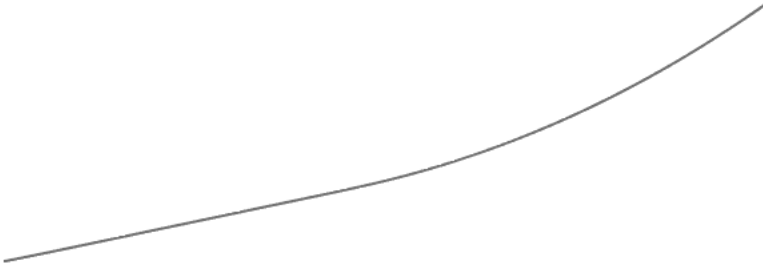


Fig. 101 *Moo*

The connection to the ground is handled differently for floating and fixed-bottom substructures. For floating substructures, the anchoring is done via the connection the ground is defined in the `SUBCONSTRAINTS` table. It can be either a rigid connection or a connection via a system of non-linear springs and d

`MOORELEMENTS`

is a table that contains the structural parameters of the flexible cable elements of the substructure such as mooring lines. Each row defines one set of coefficient and a hydrodynamic diameter in [m], which is used during buoyancy and Morison force evaluations.

```
MOORELEMENTS
MooID   MASS_[kg/m]     EIy_[N.m^2]     EA_[N]          DAMP_[-]        DIA_[m]
1       1.086306E+02    6.148892E+08    7.536117E+08    0.001           0.077
2       2.013616E+02    4.234759E+08    8.513517E+08    0.001           0.137
```

Optionally, a nonlinear axial stiffness for a *MOORELEMENT* may be defined by a nonlinear data table. In this case, the data table identifier, preceded by

```
MOORELEMENTS
MooID   MASS_[kg/m]     EIy_[N.m^2]     EA_[N]          DAMP_[-]        DIA_[m]
1       1.086306E+02    6.148892E+08    #NLDATA1        0.001           0.077
2       2.013616E+02    4.234759E+08    #NLDATA2        0.001           0.137
```

`CABDAMP`

In some cases, if the alpha damping coefficient of a mooring line (or cable) element is too large, a simulation can become unstable. Therefore, be defau

Listing 57 : Activati

```
true CABDAMP
```

`MOORMEMBERS`

is a table that contains the information of the cable members (such as the mooring lines). Each row defines one cable member and has 10 entries. The

- With the keyword `JNT_<ID>`, where <ID> represents the ID of the joint. This way, the cable is connected directly to a existing joint.
- With the keyword `FLT_<XPos>_<YPos>_<ZPos>`, where <XPos>_<YPos>_<ZPos> represent the global (x,y,z) coordinates of the connection point (in m

- With the keyword `GRD_<XPos>_<YPos>`, where <XPos>_<YPos> represent the global (x,y) (in m) coordinates of an anchor point which is located at th

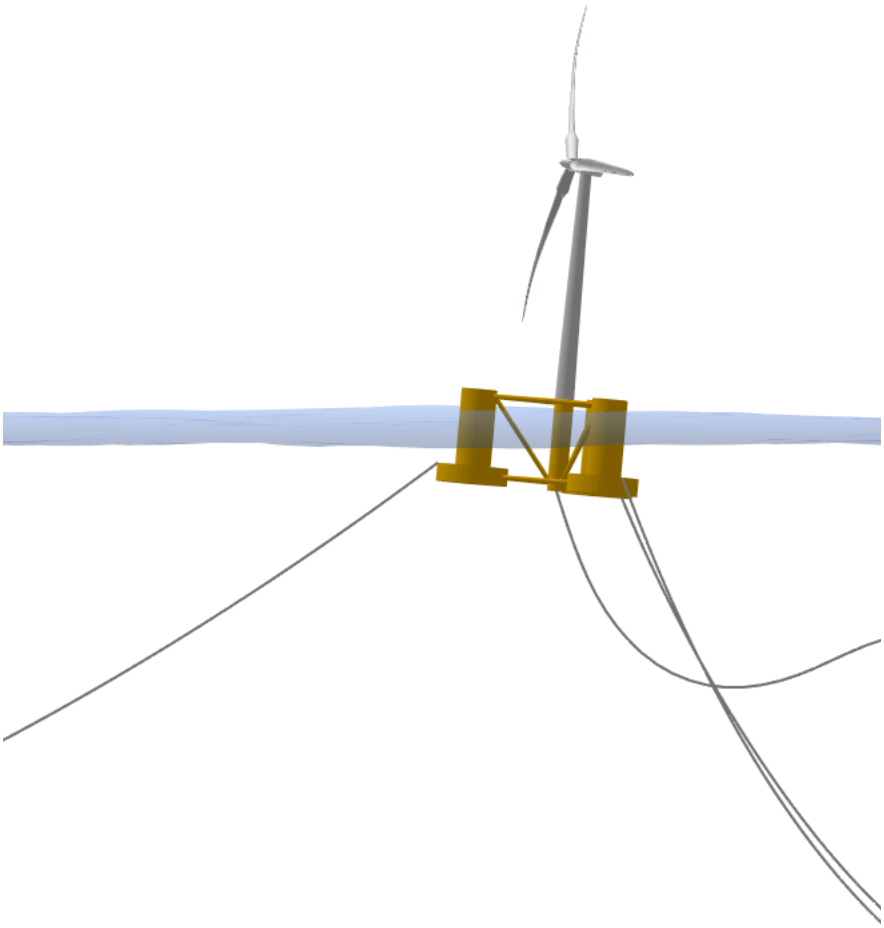
The fourth entry is the length of the cable (in m). The fifth entry is the ID number of the cable element defined in `MOORELEMENTS`. The sixth entry is the marine growth element used for this cable (see `MARINEGROWTH`). The ninth entry is the number of discretization nodes used to discretize the cable and t

```
MOORMEMBERS
ID      CONN_1                          CONN_2                Len.[m] MoorID  HyCoID  IsBuoy  MaGrID  ElmDsc  Name
1       FLT_-40.868_0.0_-14.0           GRD_-837.6_0          835.5   1       1       1       0       30      Mooring1
2       FLT_20.434_35.393_-14.0         GRD_418.8_725.4       835.5   1       1       1       0       30      Mooring2
3       FLT_20.434_-35.393_-14.0        GRD_418.8_-725.4      835.5   1       1       1       0       30      Mooring3
```

Mooring Element Lineloads

`MOORLOADS`

is a table that allows to add buoyancy loads or additional weight to a cable member defined in the `MOORMEMBERS` table. The first column is the cable me Axis. A positive load is pointing upwards and a negative load is pointing downwards.



```
MOORLOADS
ID      Start[m]        End [m]         Force [N/m]
1       150             180             2000
3       520             550             2000
```

Nonlinear Spring and Damper Constraints

`NLSPRINGDAMPERS`

is a table that defines one or more non-linear spring-damper systems for connecting the substructure to the ground, or for the interconnection of two substructure members or joints. Furthermore, in the `SUBCONSTRAINTS` table the nonlinear springs, or dampers may be assigned to constrain any or all de

Each row in the `NLSPRINGDAMPERS` table represents a spring-damper system and has 2N + 2 entries, where N is the number of points on the definition t modeled. There are two options: 'spring' and 'damp'. This affects the way the coefficients in the following entries are interpreted.

- If 'spring' is selected, then QBlade expects the definition table to consists of displacement or rotation (in m or rad) and stiffness (in N/m or Nm/rad
- If 'damp' is selected, then QBlade expects the definition table to consist of velocity (in m/s or rad/s) and damping (in N(m/s) or Nm/(rad/s)) entries.

When a spring or damper is used to constrain two joints its nonlinear definition always acts as a rotational spring or damper along the rotational DOF' spring to constrain translational DOF's.

The following 2N entries represent the additional lookup table entries for the non-linear spring/damper system. The order is x_1/v_1, $K/D(x_1/v_1)$; x_2

```
NLSPRINGDAMPERS
ElemID  Type    Coefficient & Displacement/Velocity Sets (for NL springs, dampers)
1       spring  1.000   1.160E+06
2       spring  1.000   9.000E+06
3       spring  1.000   2.090E+07
4       spring  1.000   3.560E+07
```

Optionally, the force/displacement or force/velocity relationship can also be specified though a Nonlinear Data Tables. In this case, the data table ID, p

Listing

```
NLSPRINGDAMPERS
ElemID  Type    Coefficient & Displacement/Velocity Sets (for NL springs, dampers)
1       spring  #NLDATA1
2       spring  #NLDATA2
3       spring  #NLDATA3
4       spring  #NLDATA4
```

`SPRINGDAMPK`

is an optional proportionality constant to add a damping value to the spring elements. If this keyword is used, then all of the spring elements defined i 'damp' elements defined in `NLSPRINGDAMPERS`.

Changes to in NLSPRINGDAMPERS QBlade 2.0.7

From QBlade 2.0.7, the `NLSPINGDAMPERS` table has been slightly modified. In Previous versions the coefficient for $x = 0$ was specified by the user in colu This implies that in order to continue to use a `NLSPINGDAMPERS` table from a QBlade version prior 2.0.7 the 3rd column has to be removed. The old table

Listi

```
NLSPRINGDAMPERS
ElemID  Type    Coefficient (for x = 0) Coefficient & Displacement/Velocity Sets (for NL springs, dampers)
1       spring  0.000E+00       1.000   1.160E+06
2       spring  0.000E+00       1.000   9.000E+06
3       spring  0.000E+00       1.000   2.090E+07
4       spring  0.000E+00       1.000   3.560E+07
```

Nonlinear Data Tables

`NLDATA<X>`

is a table that defines a nonlinear data distribution. The type as which the data is interpreted, depends on where the data table is used. Currently, nonl relationships of spring (or damper) elements (see Nonlinear Spring and Damper Constraints).

Each nonlinear data table needs a unique identifier, such as NLDATA1, NLDATA2, NLDATA3, up to NLDATA100. The first column in each data table r x=0 and y=0 and it is not requires to input the 0,0 entry into the table. Other values should be input is ascending order of x.

```
NLDATA1
x_val   y_val
0.001   1.0e9
0.002   2.5e9
0.005   5.0e9
```

Hydrodynamic Modeling of a Substructure

Two options are available in QBlade to model the hydrodynamic forces acting on an offshore substructure: The Morison equation and the linear potential

When modeling the hydrodynamics using the Morison equation the user can distribute hydrodynamic coefficients that act in the normal direction of a sul Morison equation the hydrodynamic forces are *distributed* over the substructure model.

The second option is to model the hydrodynamic forces using a linear potential flow theory generated database. At present, QBlade can interpret hydrody point (`REF_HYDRO_POS_<X>`). So in most cases a substructure modeled with potential flow theory should be modeled using *rigid* elements.

QBlade allows the user to combine elements from the Linear Potential Flow Theory and Morison Equation hydrodynamic models freely. The user should b

A typical mix between the Morison equation and potential flow theory is to have all hydrodynamic forces be evaluated by a linear potential flow database

Morison Equation (Strip Theory) Modelling

Hydrodynamic coefficients can be assigned to substructure members and joints. Hydrodynamic member coefficients (`HYDROMEMBERCOEFF`) act in the directi

Memb

Fig. 103 H

`HYDROMEMBERCOEFF`

defines a table that contains the hydrodynamic normal coefficients that are used for the **cylindrical** members of the substructure. Each row contains o drag coefficient, the normal added mass coefficient, the normal dynamic pressure coefficient, a flag that enables the MacCamy-Fuchs correction (MCF

```
HYDROMEMBERCOEFF
CoeffID CdN     CaN     CpN     MCFC    CdL (optional)
1       2.0     0.8     1.0     1       0
2       0.63    0.0     0.0     1       0
3       0.56    0.0     0.0     0       0
4       0.61    0.0     0.0     0       0
5       0.68    0.0     0.0     0       0
```

`HYDROMEMBERCOEFF_RECT`

defines a table that contains the hydrodynamic normal coefficients that are used for the **rectangular** members of the substructure. Each row contains direction, the normal added mass coefficient along the members x-direction, the normal dynamic pressure coefficient along the members x-direction, a flag that enables the MacCamy-Fuchs correction (MCFC).

```
HYDROMEMBERCOEFF_RECT
CoeffID CdNx    CaNx    CpNx    CdNy    CaNy    CpNy    MCFC
1       2.0     0.8     1.0     2.0     0.8     1.0     1
2       0.63    0.0     0.0     0.63    0.0     0.0     1
3       0.56    0.0     0.0     0.56    0.0     0.0     0
4       0.61    0.0     0.0     0.61    0.0     0.0     0
5       0.68    0.0     0.0     0.68    0.0     0.0     0
```

HYDROJOINTCOEFF

is a table that defines hydrodynamic axial coefficients that can be placed at specific joints (defined by their ID number) of the substructure that are loc axial drag, added mass and dynamic pressure axial coefficients and is structured as shown below. The hydrodynamic reference volume for a member e reference areas and reference volumes are subtracted from another so that just the area and reference volumes that is exposed to the fluid is consider

```
HYDROJOINTCOEFF
CoeffID JointID CdA     CaA     CpA
1       9       4.8     0.0     0.0
2       10      4.8     0.0     0.0
3       11      4.8     0.0     0.0
4       1       0.0     0.0     0.0
5       3       0.0     0.0     0.0
6       5       0.0     0.0     0.0
7       7       0.0     0.0     0.0
```

In addition to the basic functionalities of the *HYDROJOINTCOEFF* table additional entries in the table can activate a more advanced modeling of memb **Sided Morison Drag** and **High Pass Filtered Morison Drag** can be activated by populating additional columns in the *HYDROJOINTCOEFF* table.

One-Sided Morison Drag

The *One Sided Morison Drag* method modifies the axial drag model used in QBlade by applying the axial drag force solely when the flow normal to the increased stagnation pressure when the flow is the opposite direction of the end face normal vector. This modification more accurately reflects the ph the evaluation of the end face drag force in the following way:

$$F_D = \frac{1}{2} C_{d_A} |v \cdot n| \max(v \cdot n, 0)$$

The one-sided Morison drag evaluation is activated by setting the sixth column of the *HYDROJOINTCOEFF* table to 1. Setting this column to 0 deactiv

Listin

```
HYDROJOINTCOEFF
CoeffID JointID CdA     CaA     CpA     OSD
1       9       4.8     0.0     0.0     1
2       10      4.8     0.0     0.0     1
3       11      4.8     0.0     0.0     0
4       1       0.0     0.0     0.0     0
5       3       0.0     0.0     0.0     0
6       5       0.0     0.0     0.0     1
7       7       0.0     0.0     0.0     1
```

High-Pass Filtered Morison Drag

The *High-Pass Filtered Morison Drag* modification is another method to improve, or fine-tune the low-frequency response of a floating structure. To opt implement varying drag force across different wave frequency bands. This method uses a first-order high pass filter, which is applied to the relative no

$$\tilde{v}_i = C \cdot \tilde{v}_{i-1} + C \cdot (v_i - v_{i-1}),$$

with

$$C = exp(-2\pi f_c \Delta t).$$

The applied filtered drag force $F_{D,f}$ is then evaluated as:

$$F_{D,f} = \alpha F_D + (1 - \alpha)\tilde{F}_D,$$

where α is a scaling factor between 0 and 1, $\tilde{F}_D$ is the drag force evaluated with the filtered normal velocity $\tilde{v}_i$ and F_D the drag force evaluated with

The *High-Pass Filtered Morison Drag* evaluation is activated by setting the seventh and eight column of the *HYDROJOINTCOEFF* tables to the cut-off fre

Listing 68 : T

```
HYDROJOINTCOEFF
CoeffID JointID CdA     CaA     CpA     OSD     f_c     alpha
1       9       4.8     0.0     0.0     1       0.07    0.5
2       10      4.8     0.0     0.0     1       0.07    0.5
3       11      4.8     0.0     0.0     0       0.07    0.5
4       1       0.0     0.0     0.0     0       0.07    0.5
5       3       0.0     0.0     0.0     0       0.07    0.5
6       5       0.0     0.0     0.0     1       0.07    0.5
7       7       0.0     0.0     0.0     1       0.07    0.5
```

An application of this filtered drag evaluation can be found in the works of Wang et al. [1] and Behrens de Luna et al. [2].

WAVEKINEVAL_MOR

is an *optional* flag that controls how the local wave kinematics are used to calculate the Morison forces (see Modeling Considerations for Morison Elen

- 0: local evaluation of wave kinematics (this is the default value if not specified)
- 1: evaluation at the fixed, undisplaced/unrotated initial reference position
- 2: evaluation at a lagged position (controlled by `WAVEKINTAU`).

`WAVEKINEVAL_POT`

is an *optional* flag that control how the local wave kinematics are used to calculate the diffraction and second order forces at potential flow bodies. Th

- 0: local evaluation of wave kinematics
- 1: evaluation at the fixed, undisplaced/unrotated initial reference position (**this is the default value if not specified**)
- 2: evaluation at a lagged position (controlled by `WAVEKINTAU`).

`WAVEKINTAU`

is an *optional* time constant for the first order low-pass filter used to determine lagged position of the Morison/Potential Flow elements (when `WAVEKI`

Linear Potential Flow Modelling

QBlade supports any number of linear potential flow bodies as part of a substructure definition, where each potential flow body is related to a transition p underscore and a number after the keyword. So, for example, if a substructure has two bodies that use the linear potential flow theory, the second body v denoted as `SUB_MASS_2` and so on (see the section: Lumped Mass, Inertia and Hydrodynamic Forces).

The keywords that are used to read in the linear potential flow databases for radiation, excitation, difference and sum frequency loads, or hydrodynamic s

Defining a Potential Flow Body

`REF_HYDRO_POS_<X>`

defines the position at which the potential flow loads (hydro stiffness, radiation, excitation, sum frequency and difference frequency) are applied. This constrained with the transition piece`:code:*TP_INTERFACE_<X>*.

```
REF_HYDRO_POS_1
X[m]            Y[m]            Z[m]
0               0               -10.00
```

`POT_HST_FILE_<X>`

defines the file where the hydrodynamic stiffness matrix is stored from a WAMIT calculation. If this keyword is used then the hydrodynamic stiffness r

`POT_RAD_FILE_<X>`

defines the file where the radiation coefficients for the linear potential flow model are located. The file ending must be included. This determines the f

`POT_EXC_FILE_<X>`

defines the file where the excitation coefficients for the linear potential flow model are located. The file ending must be included. This determines the

`POT_DIFF_FILE_<X>`

defines the file where the second-order difference-frequency wave force coefficients are located. The file ending must be included. This determines th

`DIFF_CUTOFF_L_<X>`

Specifies the low cut-off frequency for the difference QTFs. Intended to speed up the QTF evaluations.

`DIFF_CUTOFF_H_<X>`

Specifies the high cut-off frequency for the difference QTFs. Intended to speed up the QTF evaluations.

`DIFF_INACTIVE_DOF_<X>`

this keyword can be used to define a list of DOFs for which the difference QTF will not be evaluated. Intended to speed up the QTF evaluations.

Listing 70 : The DIFF_INA

```
1 4 DIFF_INACTIVE_DOF
```

`POT_SUM_FILE_<X>`

defines the file where the second-order sum-frequency wave force coefficients are located. The file ending must be included. This determines the form

`SUM_CUTOFF_L_<X>`

Specifies the low cut-off frequency for the sum QTFs. Intended to speed up the QTF evaluations.

`SUM_CUTOFF_H_<X>`

Specifies the high cut-off frequency for the sum QTFs. Intended to speed up the QTF evaluations.

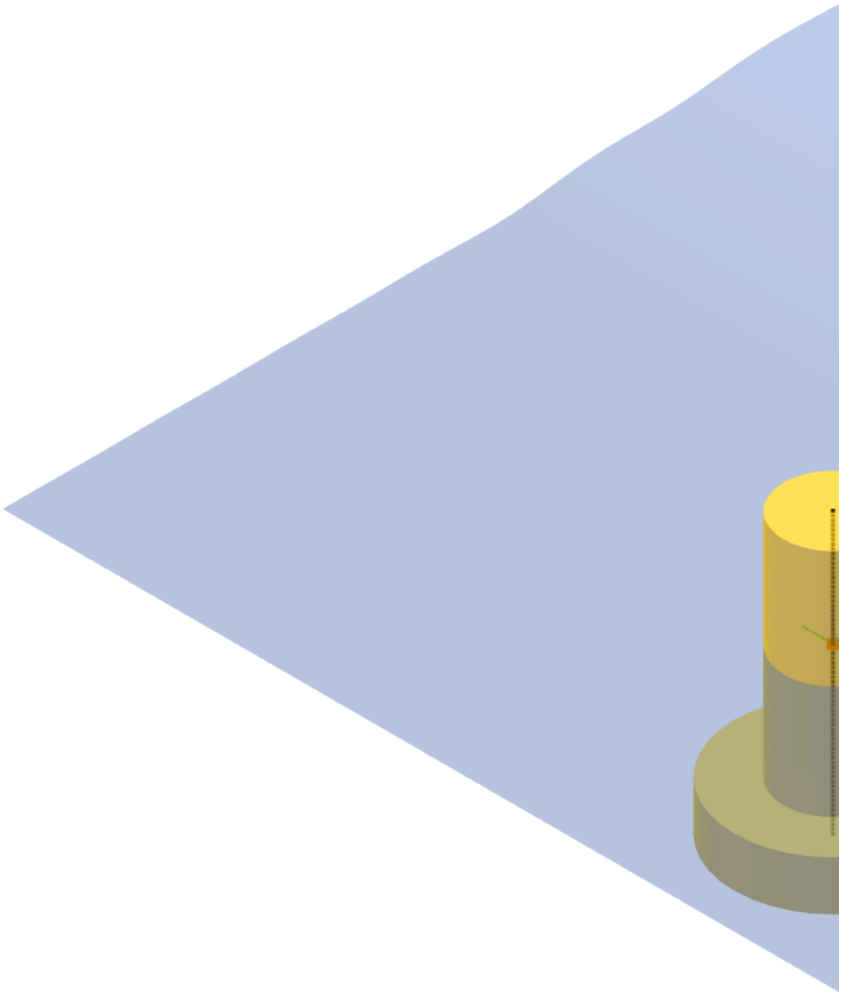
`SUM_INACTIVE_DOF_<X>`

this keyword can be used to define a list of DOFs for which the sum QTF will not be evaluated. Intended to speed up the QTF evaluations.

Listing 71 : The SUM_I

```
1 4 SUM_INACTIVE_DOF
```

NOBODY > 1 Feature



N

QBlade includes the capability to include multiple bodies (NBODY>1 feature of WAMIT) that interact hydrodynamically and mechanically. Each body can

`POT_NBODY_<X>`

specifies the number of hydrodynamic bodies. Each body is associated with 6 DOF's. By default, NBODY is equal to 1. To use NBODY>1, the potentia

```
3 POT_NBODY
```

By default, the hydrodynamic loads are applied at the `REF_HYDRO_POS`. When using multiple hydrodynamic bodies, such $N = 3$, the hydrodynamic forces

`POT_NBODY_NODES_<X>`

specifies the list of joint id's at which the hydrodynamic loads are applied. This keyword can also be used to apply the hydrodynamic loads of a single N

Listing 73 : The POT_NBOD

```
2 4 6 POT_NBODY_NODES
```

Common Potential Flow Keywords

`USE_RADIATION`

is a flag that enables the calculation of the radiation loads on all potential flow bodies. (true or false)

`USE_RAD_ADDMASS`

when this flag is set to true the hydrodynamic added mass matrix is automatically extracted from the potential flow radiation file (if such a file is defin

`DELTA_FREQ_RAD`

is the discretization of the frequencies used for the calculation of the radiation forces (in Hz).

`TRUNC_TIME_RAD`

is the truncation time for the wave radiation kernel calculations (in s).

`USE_EXCITATION`

is a flag that enables the calculation of the excitation loads on all potential flow bodies. (true or false)

`DELTA_FREQ_EXC`

is the discretization of the frequencies used for the calculation of the excitation forces (in Hz).

`DELTA_DIR_EXC`

is the discretization of the directions used for the calculation of multi-directional excitation forces (in degree). The default value is 0.5 degree.

`TRUNC_TIME_EXC`

is the truncation time for the wave excitation kernel calculations (in s).

`DIFF_EVAL_TYPE`

is a flag that controls how the 2nd order difference-frequency loads on all potential flow bodies are evaluated:

- 0 - no difference forces are evaluated
- 1 - difference-frequency loads are evaluated explicitly (full field QTF, high computational demand)
- 2 - the computationally efficient Newman approximation is used for the calculation of difference frequency forces
- 3 - only the mean drift forces are considered

`USE_SUM_FREQS`

is a flag that enables the (full field QTF) calculation of the sum-frequency loads on all potential flow bodies. (true or false)

`UNITLENGTH_WAMIT`

Enables to specify a WAMIT unit length different than 1.0, if not specified 1.0 is the default value.

Sensor Locations and Definitions

The locations at which data is recorded for the substructure is also controlled by keywords. QBlade can generate output for the members defined in the

`SUB_<MemID>_<RelPos>`

is the keyword used for setting an output of a member from the the `SUBMEMBERS` table with the ID number = <MemID> and a relative postion = <RelP
Time Graph and internal structural loads are displayed in the *Structural Time Graph*.

`MOO_<MMemID>_<RelPos>`

is the keyword used for placing a sensor on the cable member with the ID number = <MMemID> and a relative postion = <RelPos>. The relative positi
structural loads are displayed in the *Structural Time Graph*.

`CST_<CstID>`

is the keyword used for placing a sensor at the constraint from the `SUBCONSTRAINTS` table with the ID number = <CstID>. The internal constraint loads
JointID1 and has the axes orientation of the joint (JntCon), transition piece (TpCon) or ground (GrdCon) that JointID1 is connected to (see Substructur

`JNT_<JntID>`

is the keyword used for placing a sensor on a joint from the `SUBJOINTS` table with the ID number = <JntID>. The position and rotation of the joint in al

Exemplary Substructure File

An exemplary substructure file for the OC4 Semi-Submersible floater is shown below. This floater is modeled with rigid cylindrical elements. The hydrody
In this example the members are defined as mass-less (0.0001kg/m) and the total mass is assigned through the 6x6 mass matrix. The total floater hydrody

```
200             WATERDEPTH  //design depth

1025            WATERDENSITY // design density, used for flooded member mass calcs

true            ISFLOATING //if the structure is fixed the joint coordinates are assigned in a coordinate system with O(0,0,0) at the mudline, for

100             ADVANCEDBUOYANCY //using an advanced discretization technique (N must be a square int number) to calculate buoyancy of partially su

0               WAVEKINEVALTYPE // 0 - local evaluation, 1 - eval at fixed ref pos, 2 - eval at lagged position
30              WAVEKINTAU // time constant for the lagged waveKin position evaluation

// potential flow model options, specify RADiation and EXCitation files separately (only RAD if BEMuse), don't forget the file endings, as this ide

true            STATICBUOYANCY // static buoyancy, based on the MSL should be used when using Morison member buoyancy combined with potential flow

radiation.1     POT_RAD_FILE
true            USE_RADIATION
0.05            DELTA_FREQ_RAD
60.0            TRUNC_TIME_RAD

excitation.3    POT_EXC_FILE
true            USE_EXCITATION
0.05            DELTA_FREQ_EXC
0.50            DELTA_DIR_EXC
60.0            TRUNC_TIME_EXC

difference.12d  POT_DIFF_FILE
2               DIFF_EVAL_TYPE  (0-none,1-explicit,2-newman,3-meandrift)

sum.12s         POT_SUM_FILE
false           USE_SUM_FREQS

JOINTOFFSET // these global offsets are only applied to joints (not the TP or cog position)
XPOS    YPOS    ZPOS
0       0       0

MARINEGROWTH
ID      Thickn  Density
1       0.1     1100

//all following positions are defined in (x,y,z) [m]: for floaters: from the neutral point, which is located at MSL (0,0,0); for bottom fixed subst

TP_INTERFACE_POS //the interface position between substructure and tower or RNA
X[m]            Y[m]            Z[m]
0               0               10

REF_COG_POS  //cog reference position, at which the mass matrix is evaluated
X[m]            Y[m]            Z[m]
0               0               -13.46

REF_HYDRO_POS //reference point for the evaluation of linearized hydrodynamic stiffness, damping, quaddamping, addedmass matrices and the constant
X[m]            Y[m]            Z[m]
0               0               0

SUB_MASS //the floater mass matrix is defined at the REF_COG_POS
1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   1.34730e+07   0.00000e+00   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   6.82700e+09   0.00000e+00   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   6.82700e+09   0.00000e+00
0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   1.22600e+10

SUB_HYDROADDEDMASS //the hydrodynamic added mass is defined and applied at the REF_HYDRO_POS
6.3199481E+06   0               -5.4452131E+02  0               -8.4184511E+07  0
0               6.3199122E+06   0               8.4184511E+07   0               2.0423668E+02
-1.8215736E+02  0               1.4673705E+07   0               1.7654785E+03   0
```

```
0               8.4181805E+07   0               7.1983946E+09   0               1.0104395E+04
-8.4190835E+07  0               -8.7227367E+04  0               7.1983290E+09   0
0               6.2468769E+03   0               -3.6169083E+04  0               4.7423470E+09


SUBJOINTS //defined either from MSL (if isFLoating) or from seabed using the designDepth variable (if !isFLoating)
JointID JointX          JointY  JointZ
 1     0.00000     0.00000   -20.00000
 2     0.00000     0.00000    10.00000
 3    14.43376    25.00000   -14.00000
 4    14.43376    25.00000    12.00000
 5   -28.86751     0.00000   -14.00000
 6   -28.86751     0.00000    12.00000
 7    14.43376   -25.00000   -14.00000
 8    14.43376   -25.00000    12.00000
 9    14.43375    25.00000   -20.00000
10   -28.86750     0.00000   -20.00000
11    14.43375   -25.00000   -20.00000
12     9.23760    22.00000    10.00000
13   -23.67130     3.00000    10.00000
14   -23.67130    -3.00000    10.00000
15     9.23760   -22.00000    10.00000
16    14.43375   -19.00000    10.00000
17    14.43375    19.00000    10.00000
18     4.04145    19.00000   -17.00000
19   -18.47520     6.00000   -17.00000
20   -18.47520    -6.00000   -17.00000
21     4.04145   -19.00000   -17.00000
22    14.43375   -13.00000   -17.00000
23    14.43375    13.00000   -17.00000
24     1.62500     2.81500    10.00000
25    11.43376    19.80385    10.00000
26    -3.25000     0.00000    10.00000
27   -22.87000     0.00000    10.00000
28     1.62500    -2.81500    10.00000
29    11.43376   -19.80385    10.00000
30     1.62500     2.81500   -17.00000
31     8.43376    14.60770   -17.00000
32    -3.25000     0.00000   -17.00000
33   -16.87000     0.00000   -17.00000
34     1.62500    -2.81500   -17.00000
35     8.43376   -14.60770   -17.00000
36     1.62500     2.81500   -16.20000
37    11.43376    19.80385     9.13000
38    -3.25000     0.00000   -16.20000
39   -22.87000     0.00000     9.13000
40     1.62500    -2.81500   -16.20000
41    11.43376   -19.80385     9.13000
42    14.43376    25.00000   -19.94000
43   -28.86751     0.00000   -19.94000
44    14.43376   -25.00000   -19.94000
45    14.43376    25.00000   -6.170000
46   -28.86751     0.00000   -6.170000
47    14.43376   -25.00000   -6.170000
48    14.43376    25.00000   -14.89000
49   -28.86751     0.00000   -14.89000
50    14.43376   -25.00000   -14.89000

1.00    STIFFTUNER
1.00    MASSTUNER
1.00    BUOYANCYTUNER

SUBELEMENTSRIGID
ElemID  BMASSD  DIAMETER
1       1       6.5
2       1       12
3       1       24
4       1       1.6

// Heave hydro forces of base columns
HYDROJOINTCOEFF  //hydrodynamic coefficients to be assigned to joints, acting on connected members faces in axial direction, occulation of intercor
CoeffID JointID CdA     CaA     CpA
1       9       4.8     0.0     0.0     // Bottom_Base_Column_1
2       10      4.8     0.0     0.0     // Bottom_Base_Column_2
3       11      4.8     0.0     0.0     // Bottom_Base_Column_3
4       1       0.0     0.0     0.0     // Main_Column
5       3       0.0     0.0     0.0     // Top_Base_Column_1
6       5       0.0     0.0     0.0     // Top_Base_Column_2
7       7       0.0     0.0     0.0     // Top_Base_Column_3


HYDROMEMBERCOEFF //hydrodynamic coefficients to be assigned to rigid or elastic cylindrical members, defined for the normal-to-axis direction of th
CoeffID CdN     CaN     CpN     MCFC
1       2.0     0.8     1.0     0       // Mooring_Lines
2       0.63    0.0     0.0     0       // D_1.6m
3       0.56    0.0     0.0     0       // D_6.5m
4       0.61    0.0     0.0     0       // D_12m
5       0.68    0.0     0.0     0       // D_24m
```

```
SUBCONSTRAINTS //in this version of the OC4 the member nodes are connected directly through the constraints
ID      JntID   Jnt2ID  TrP     Fixed   Spring  DoF_X   DoF_Y   DoF_Z   DoF_rX  DoF_rY  DoF_rZ
1       2       0       1       0       0       1       1       1       1       1       1
2       24      0       1       0       0       1       1       1       1       1       1
3       26      0       1       0       0       1       1       1       1       1       1
4       28      0       1       0       0       1       1       1       1       1       1
8       30      0       1       0       0       1       1       1       1       1       1
9       32      0       1       0       0       1       1       1       1       1       1
10      34      0       1       0       0       1       1       1       1       1       1
14      12      0       1       0       0       1       1       1       1       1       1
15      14      0       1       0       0       1       1       1       1       1       1
16      16      0       1       0       0       1       1       1       1       1       1
20      18      0       1       0       0       1       1       1       1       1       1
21      20      0       1       0       0       1       1       1       1       1       1
22      22      0       1       0       0       1       1       1       1       1       1
26      36      0       1       0       0       1       1       1       1       1       1
27      38      0       1       0       0       1       1       1       1       1       1
28      40      0       1       0       0       1       1       1       1       1       1
29      9       0       1       0       0       1       1       1       1       1       1
30      10      0       1       0       0       1       1       1       1       1       1
31      11      0       1       0       0       1       1       1       1       1       1


SUBMEMBERS
MemID   Jnt1ID  Jnt2ID  ElmID   ElmRot  HyCoID  IsBuoy  MaGrID  FldArea ElmDsc  Name (optional)
 1      1       2       1       0       3       1       0       0       2       Main_Column
 2      45      4       2       0       4       1       0       0       2       Upper_Column_1
 3      46      6       2       0       4       1       0       0       2       Upper_Column_2
 4      47      8       2       0       4       1       0       0       2       Upper_Column_3
29      3       45      2       0       4       1       0       0       2       Upper_Column_flooded_1
30      5       46      2       0       4       1       0       0       2       Upper_Column_flooded_2
31      7       47      2       0       4       1       0       0       2       Upper_Column_flooded_3
 5      48      3       3       0       5       1       0       0       2       Base_Column_1
 6      49      5       3       0       5       1       0       0       2       Base_Column_2
 7      50      7       3       0       5       1       0       0       2       Base_Column_3
26      42      48      3       0       5       1       0       0       2       Base_column_flooded_1
27      43      49      3       0       5       1       0       0       2       Base_column_flooded_2
28      44      50      3       0       5       1       0       0       2       Base_column_flooded_3
23      9       42      3       0       5       1       0       0       2       Base_column_cap_1
24      10      43      3       0       5       1       0       0       2       Base_column_cap_2
25      11      44      3       0       5       1       0       0       2       Base_column_cap_3
 8      12      13      4       0       2       1       0       0       10      Delta_Pontoon_Upper_1
 9      14      15      4       0       2       1       0       0       10      Delta_Pontoon_Upper_2
10      16      17      4       0       2       1       0       0       10      Delta_Pontoon_Upper_3
11      18      19      4       0       2       1       0       0       10      Delta_Pontoon_Lower_1
12      20      21      4       0       2       1       0       0       10      Delta_Pontoon_Lower_2
13      22      23      4       0       2       1       0       0       10      Delta_Pontoon_Lower_3
14      24      25      4       0       2       1       0       0       10      Y_Pontoon_Upper_1
15      26      27      4       0       2       1       0       0       10      Y_Pontoon_Upper_2
16      28      29      4       0       2       1       0       0       10      Y_Pontoon_Upper_3
17      30      31      4       0       2       1       0       0       10      Y_Pontoon_Lower_1
18      32      33      4       0       2       1       0       0       10      Y_Pontoon_Lower_2
19      34      35      4       0       2       1       0       0       10      Y_Pontoon_Lower_3
20      36      37      4       0       2       1       0       0       10      Cross_Brace_1
21      38      39      4       0       2       1       0       0       10      Cross_Brace_2
22      40      41      4       0       2       1       0       0       10      Cross_Brace_3

MOORELEMENTS
MooID   MASS_[kg/m]     EIy_[N.m^2]     EA_[N]          DAMP_[-]        DIA_[m]
1       1.086306E+02    6.148892E+08    7.536117E+08    0.000           0.077

MOORMEMBERS
ID      CONN_1                          CONN_2                  Len.[m] MoorID  HyCoID  IsBuoy  MaGrID  ElmDsc  Name
1       FLT_-40.868_0.0_-14.0           GRD_-837.6_0            835.5   1       1       1       0       30      Mooring1
2       FLT_20.434_35.393_-14.0         GRD_418.8_725.4         835.5   1       1       1       0       30      Mooring2
3       FLT_20.434_-35.393_-14.0        GRD_418.8_-725.4        835.5   1       1       1       0       30      Mooring3

TRANSITIONBLOCK // just for visualization
WIDTH   LENGTH  HEIGHT
0       0       0

TRANSITIONCYLINDER // just for visualization
HEIGHT  DIAMETER
0.5     6.5

RGBCOLOR
R       G       B
255     200     15

MOO_1_0.0
MOO_1_1.0
MOO_2_0.0
MOO_2_1.0
MOO_3_0.0
MOO_3_1.0
```

Substructure File Format Changes from QBlade v2.06b

The section describes the changes that have been made to different parts of the substructure file format from QBlade 2.06b onward. In most cases comp
are very little and can be implemented in a few minutes per file. See a summary of the changes below:

SUBMEMBERS Table

In QBlade versions prior to 2.06b the **SUBMEMBERS** table looked like this:

```
SUBMEMBERS
MemID   Jnt1ID  Jnt2ID  ElmID   RElmID  HyCoID  IsBuoy  MaGrID  FldArea ElmDsc  Name (optional)
 1       1       2       0       1       3       1       0       0       2       Main_Column
 2      45       4       0       2       4       1       0       0       2       Upper_Column_1
 3      46       6       0       2       4       1       0       0       2       Upper_Column_2
 4      47       8       0       2       4       1       0       0       2       Upper_Column_3
29       3      45       0       2       4       1       0       0       2       Upper_Column_flooded_1
30       5      46       0       2       4       1       0       0       2       Upper_Column_flooded_2
31       7      47       0       2       4       1       0       0       2       Upper_Column_flooded_3
 5      48       3       0       3       5       1       0       0       2       Base_Column_1
 6      49       5       0       3       5       1       0       0       2       Base_Column_2
 7      50       7       0       3       5       1       0       0       2       Base_Column_3
26      42      48       0       3       5       1       0       0       2       Base_column_flooded_1
27      43      49       0       3       5       1       0       0       2       Base_column_flooded_2
28      44      50       0       3       5       1       0       0       2       Base_column_flooded_3
23       9      42       0       3       5       1       0       0       2       Base_column_cap_1
24      10      43       0       3       5       1       0       0       2       Base_column_cap_2
25      11      44       0       3       5       1       0       0       2       Base_column_cap_3
```

Pay attention to the columns 4 and 5. In the old version column 4 specified a flexible element ID **ElmID** and column 5 a rigid element ID **RElmID**. In QBlac

Because we did not want to add an extra column for each element type to the **SUBMEMBERS** table all element types can now be assigned in column 4. H
possible anymore since each element requires a unique ID.

Furthermore, for rectangular elements their orientation becomes important (vs cylindrical elements which are unidirectional), so we are using column 5 of

To sum up the changes:

- **Column 4** is now used to define all different element types
- **All elements** (across different types) need a **unique element ID**
- **Column 5** is now used to assign the element rotation

Based on this we can easily convert the old **SUBMEMBERS** table to the new format that is shown below:

```
SUBMEMBERS
MemID   Jnt1ID  Jnt2ID  ElmID   ElmRot  HyCoID  IsBuoy  MaGrID  FldArea ElmDsc  Name (optional)
 1       1       2       1       0       3       1       0       0       2       Main_Column
 2      45       4       2       0       4       1       0       0       2       Upper_Column_1
 3      46       6       2       0       4       1       0       0       2       Upper_Column_2
 4      47       8       2       0       4       1       0       0       2       Upper_Column_3
29       3      45       2       0       4       1       0       0       2       Upper_Column_flooded_1
30       5      46       2       0       4       1       0       0       2       Upper_Column_flooded_2
31       7      47       2       0       4       1       0       0       2       Upper_Column_flooded_3
 5      48       3       3       0       5       1       0       0       2       Base_Column_1
 6      49       5       3       0       5       1       0       0       2       Base_Column_2
 7      50       7       3       0       5       1       0       0       2       Base_Column_3
26      42      48       3       0       5       1       0       0       2       Base_column_flooded_1
27      43      49       3       0       5       1       0       0       2       Base_column_flooded_2
28      44      50       3       0       5       1       0       0       2       Base_column_flooded_3
23       9      42       3       0       5       1       0       0       2       Base_column_cap_1
24      10      43       3       0       5       1       0       0       2       Base_column_cap_2
25      11      44       3       0       5       1       0       0       2       Base_column_cap_3
```

As you can see in the above table we have moved all entries from column 5 of the old format (**RElmID**) to column 4 of the new format (**ElmID**). Since the r

SUBELEMENTS Tables

Each subelement that is defined now requires a unique element ID (the first column of each table) across all element types (**SUBELEMENTS**, **SU**

As an example, such an element definition was possible in older versions of the substructure format:

```
SUBELEMENTSRIGID
ElemID  BMASSD  DIAMETER
1       1       6.5
2       1       12
3       1       24
4       1       1.6

SUBELEMENTS
ElemID  MASS_[kg/m]     Eix_[N.m^2]     Eiy_[N.m^2]     EA_[N]          GJ_[N.m^2]      GA_[N]          STRPIT_[deg]    KSX_[-]         KSY_[-]
1       1.106E+04       2.515E+12       2.515E+12       2.959E+11       1.940E+12       1.141E+11       0.000E+00       5.000E-01       5.000E-
2       2.817E+03       1.819E+11       1.819E+11       7.537E+10       1.403E+11       2.907E+10       0.000E+00       5.000E-01       5.000E-
3       1.481E+03       1.922E+10       1.922E+10       3.963E+10       1.483E+10       1.529E+10       0.000E+00       5.000E-01       5.000E-
4       1.107E+03       1.447E+10       1.447E+10       2.961E+10       1.117E+10       1.142E+10       0.000E+00       5.000E-01       5.000E-
```

This now needs to be change so that every element ID is unique across all element types, in this example the only change is to change the **ElmID** in colum

```
SUBELEMENTSRIGID
ElemID  BMASSD  DIAMETER
1       1       6.5
2       1       12
3       1       24
4       1       1.6

SUBELEMENTS
ElemID  MASS_[kg/m]     Eix_[N.m^2]     Eiy_[N.m^2]     EA_[N]          GJ_[N.m^2]      GA_[N]          STRPIT_[deg]    KSX_[-]         KSY_[-]
5       1.106E+04       2.515E+12       2.515E+12       2.959E+11       1.940E+12       1.141E+11       0.000E+00       5.000E-01       5.000E-
6       2.817E+03       1.819E+11       1.819E+11       7.537E+10       1.403E+11       2.907E+10       0.000E+00       5.000E-01       5.000E-
7       1.481E+03       1.922E+10       1.922E+10       3.963E+10       1.483E+10       1.529E+10       0.000E+00       5.000E-01       5.000E-
8       1.107E+03       1.447E+10       1.447E+10       2.961E+10       1.117E+10       1.142E+10       0.000E+00       5.000E-01       5.000E-
```

MOORELEMENTS Table

The format of the **MOORELEMENTS** table has also been updated to align it more with the other element table formats. Older **MOORELEMENTS** format

The old **MOORELEMENTS** format shown below:

```
MOORELEMENTS
ID      Dens.[kg/m^3]   Area[m^2]       Iyy[m^4]        EMod[N/m^4]     RDp.[-] Dia[m]
1       2.35723E+04     4.6084E-03      3.7601E-03      1.6353E+11      0.015   0.0766
```

has now been updated to:

```
MOORELEMENTS
MooID   MASS_[kg/m]     EIy_[N.m^2]     EA_[N]          DAMP_[-]        DIA_[m]
1       1.086306E+02    6.148892E+08    7.536117E+08    0.015           0.0766
```

As you can see the new format requires one less column and specifies the stiffness and mass per length directly, in the same way as the other element tab

[1] *(1, 2, 3)* Lu Wang, Amy Robertson, Jason Jonkman, and Yi-Hsiang Yu. Oc6 phase i: improvements to the openfast predictions of nonlinear, low-frequency respo

[2] *(1, 2)* R. Behrens de Luna, S. Perez-Becker, J. Saverin, D. Marten, F. Papi, M.-L. Ducasse, F. Bonnefoy, A. Bianchini, and C.-O. Paschereit. Quantifying the impa https://wes.copernicus.org/articles/9/623/2024/, doi:10.5194/wes-9-623-2024.

Simulation Module Overview

Fig. 107 *The simulation module symbol in the QBlade main tool bar.*

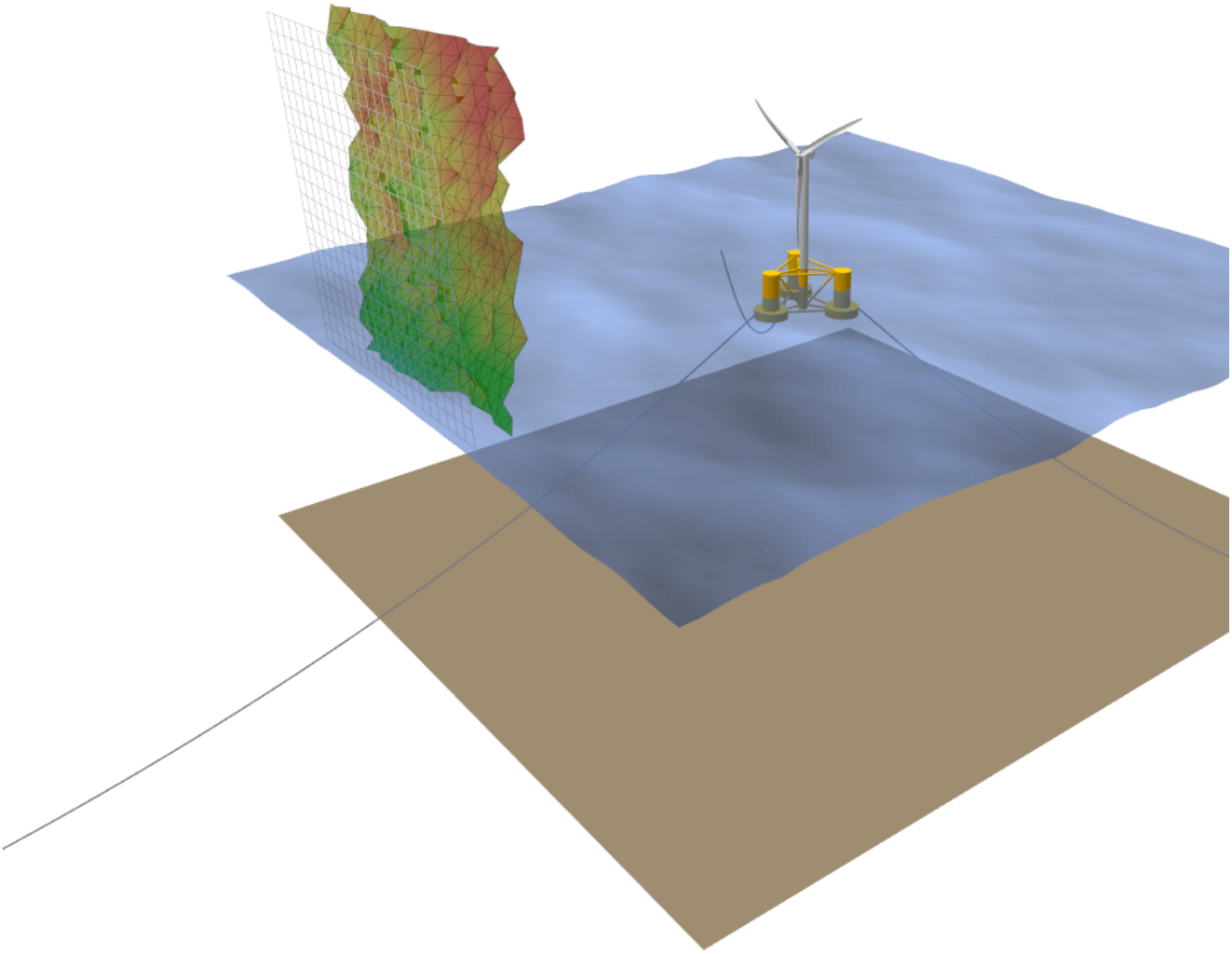


Fig. 108 *Visualization of an aero-hydro-servo-elastic simulation in QBlade.*

To define a simulation of a turbine object several simulation parameters and boundary conditions need to be defined. The following list gives an overview

- Simulation length & time step size
- Structural solver
- Wind boundary conditions
- Wave & current boundary conditions
- Environmental parameters: density, gravity, etc.

Simulations can either be defined through the turbine simulation dialog, or directly by creating or modifying simulation definition (`.sim`) files.

Simulation Definition

Setting up a simulation in QBlade is handled through the dialog shown below:

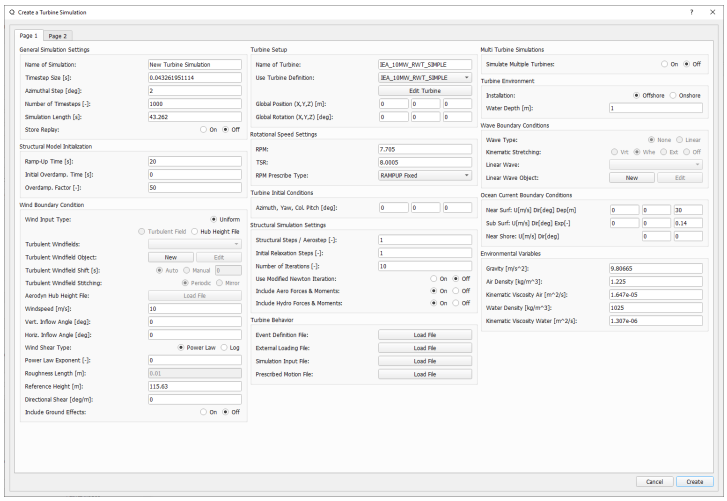


Fig. 109 *Page 1 of the simulation definition dialog (click to enlarge).*

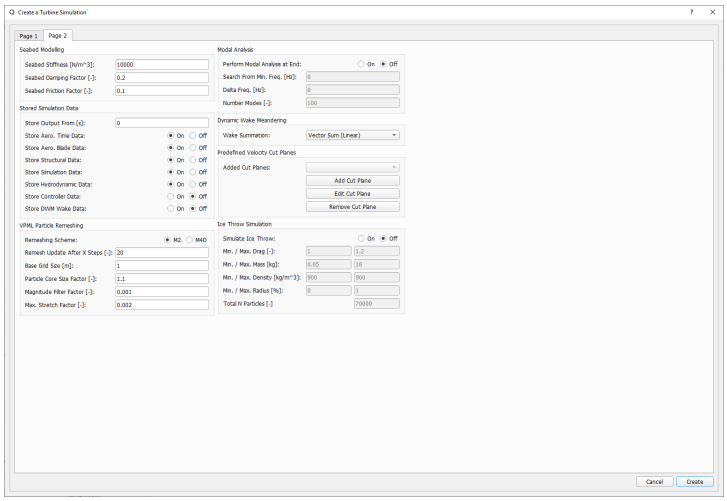


Fig. 110 *Page 2 of the simulation definition dialog (click to enlarge).*

Below, an overview of the different dialog sections and each editable parameters is given. If you are looking for a specific parameter from the dialog it is s find it quickly. Also note that each of the parameters described in the following has an equivalent keyword within the Simulation Definition ASCII File.

General Simulation Settings

- **Name of Simulation**: A unique name to identify the simulation object
- **Timestep Size / Azimuthal Step**: Both values are interconnected through the turbine **RPM** setting
- **Number of Timesteps / Simulation Length**: Both values are interconnected through the **TimestepSize** setting
- **Store Replay**: A replay of the simulation is created by storing all position and wake data for every timestep.

Structural Model Initialization

These field are only editable if the selected turbine object has a structural model definition.

- **Ramp-Up Time**: The (structural) simulation is ramped-up over the specified time. This part of the simulation does not generate data and is not stored.
- **Initial Overdamp. Time**: For the specified time the model Rayleigh damping is increased by the factor **Overdamp. Factor**. If the value is 0 the *overdamp overdamping* of the model.
- **Overdamp. Factor**: During the *overdamping* time all model damping values are increased by this factor.

Wind Boundary Condition

- **Wind Input Type**: Sets the wind input type. Options are Uniform Wind Field, Turbulent Wind Field and Hub Height File.
- **Turbulent Windfields**: In this *ComboBox* all Windfields that are stored in the database are listed.
- **Turbulent Windfield Object**: These buttons allow to generate a new Windfield or to edit an existing one.
- **Turbulent Windfield Shift**: The currently selected Windfield may be shifted in time (to start at another time than t=0s), if *Auto* is chosen the eld freestream velocity to fully immerse the rotor at the start of a simulation..
- **Turbulent Windfield Stitching**: If the simulation time is larger than the Windfield length the Windfield is either repeated (*Periodic*) or mirrored (*Mirror*) a

- **Aerodyn Hub Height File**: This enables to load a hub height file from the file system to be assigned to this simulation.
- **Windspeed**: Set the windspeed if **Wind Input Type** is set to *Uniform*.
- **Vert. Inflow Angle**: Set the vertical inflow angle if **Wind Input Type** is either *Uniform* or *Turbulent Field*.
- **Horiz. Inflow Angle**: Set the horizontal inflow angle if **Wind Input Type** is either *Uniform* or *Turbulent Field*.
- **Wind Shear Type**: Set the windshear type if the wind input type is *Uniform* or the turbulent windfield is imported.
- **Power Law Exponent**: Set the Coefficient for a **Power Law** wind profile.
- **Roughness Length**: Set the **Roughness Length** for a *Log* Profile..
- **Reference Height**: Set the **Reference Height** for the wind profiles.
- **Directional Shear**: Set a directional shear. The shear is assumed to be 0.0 at the **Reference Height**.
- **Include Ground Effects**: Interludes the modelling of **Ground Effects**, see Ground Effect.

Turbine Setup

- **Name of Turbine**: Define a name for the turbine simulation object.
- **Use Turbine Definition**: Use the selected *turbine definition* object from the data base in this simulation object.
- **Global Position (X,Y,Z)**: Set the global position of the wind turbine for this simulation.
- **Global Rotation (X,Y,Z)**: Set the global rotation of the wind turbine for this simulation. In the case of a floating turbine being simulated the global rotat

Rotational Speed Settings

- **RPM / TSR**: Both values are interconnected through the rotor size and current windspeed.

The following options are related to the RPM control for this simulation. For *turbine definition* object without a structural definition the *RPM* is always con *definition* objects with a structural definition the following options are avaliable:

- **Ramp-Up Fixed**: The *RPM* is fixed only during the ramp-up time of the simulation so that when the simulation starts the rotor is operating at the chos governed by the balance of aerodynamic- and generator torque. This is the recommended setting for simulations that contain a *Controller*.
- **Always Fixed**: The *RPM* is fixed for the total duration of the simulation to the chosen *RPM*.
- **Free**: For *ramp-up* and simulation time the rotor rotation is governed by the balance of aerodynamic- and generator torque.

Turbine Initial Conditions

- **Azimuth, Yaw, Col. Pitch**: Sets the initial azimuthal rotor angle, yaw angle and collective pitch angle for the simulation.

Floater Initial Conditions

These edits are only enabled if a wind turbine with a floating substructure is simulated. The initial floater conditions can be used to setup decay tests for s equilibrium position to speed-up initial transients.

- **X, Y, Z Translation**: Sets the initial displacement for the floater.
- **Roll, Pitch, Yaw**: Sets the initial rotation of the floater.

Structural Simulation Settings

- **Structural Steps / Aerostep**: Sets how many structural steps will be evaluated per global timestep. If multiple structural steps are evaluated per global constant.
- **Initial Relaxation Steps**: An initial iterative relaxation is performed, taking into account only gravitational forces.
- **Number of Iterations**: Set the number of iterations for the *iterative* time steppers, such as the **HHT**.
- **Include Aero Forces & Moments**: Toggles if aerodynamic forces are projected onto the structural model definition.
- **Include Hydro Forces & Moments**: Toggles if hydrodynamic forces are projected onto the structural model definition.

Turbine Events and Operation

In this section special events, external loading, prescribed motion and prescribed operation can be defined for a *turbine definition*. Below exemplary files a

- **Event Definition File**: An event is defined by a combination of *Keywords* and values. The following list gives an overview of the available event types. E structural definition. Multiple events may be defined in a single file. The events override any events / control that is returned via the controller exchan
    - **30 FAILGRID**: At time 30 s, the generator moment is set to 0 Nm.
    - **30 SETBRAKE**: At time 30 s, the brake is engaged.
    - **30 1.5 FAILPITCH_1**: At time 30 s, the pitch rate of blade nr. 1 is set to a maximum rate of 1.5 deg/s
    - **30 90 1.5 PITCHTO**: At time 30 s, the collective pitch rate is set to 1.5deg/s until 90 deg are reached.

- **30 90 1.5 YAWTO**: At time 30 s, the yaw rate is set to 1.5deg/s until 90 deg are reached.
- **30 FAILBLADE_1**: At time 30 s, blade nr. 1 is *released* from the hub, by deactivating the respective structural constraint.
- **30 FAILCABLE_1**: At time 30 s, the cable with the IDNr. 1 brakes away from the substructure.

- **External Loading File**: A user defined loading timeseries can be applied to the turbine during simulation via this file format, multiple loading timeseries file is as follows:

Listing 96 : The scheme of an external loading file

```
<SensorName> <localflag>
<time1> <fx1> <fy1> <fz1> <mx1> <my1> <mz1>
<time2> <fx2> <fy2> <fz2> <mx2> <my2> <mz2>
```

Sensor naming is the same as in the main file for the sensor outputs (see Loading Data and Sensor Locations) The local flag (local, global) defined if the loa Local Body Coordinate Systems or Local Sensor Coordinate Systems. QBlade interpolates linearly the loads between time stamps. External load time serie

This exemplary file applies an impulsive load of 1e7 N along the global x-direction to the tower at 50% height. The loads are interpolated in time, so the x drops of to 0 N at 20.2s:

Listing 97 : An exemplary external loading file that applies an impulsive load at 20s to the tower

```
TWR_0.5 false
19.8    0       0       0       0       0       0
20      1e8     0       0       0       0       0
20.2    0       0       0       0       0       0
```

- **Simulation Input File**:

The turbine operation can be prescribed using a file of the following format. *Turbine definition* with or without a structural definition can be subjected to p between time stamps.

Listing 98 : An exemplary simulation input file

```
Time    RPM     Yaw     PitchB1 PitchB2 ...     PitchBN AFC1_B1 AFC2_B2 ...     AFCN_BN
0       1       11      0       0       ...     0       0       0       ...     0
5       2       11      0       0       ...     0       0       0       ...     0
10      4       11      0       5       ...     0       0       0       ...     0
15      7       11      0       10      ...     0       0       0       ...     0
20      11      11      0       17      ...     0       0       0       ...     0
25      12      11      0       27      ...     0       0       0       ...     0
30      13      11      0       40      ...     10      0       0       ...     0
35      12      11      0       40      ...     20      0       0       ...     0
40      11      11      0       40      ...     30      0       0       ...     0
45      11      11      0       40      ...     40      0       0       ...     0
50      11      11      0       40      ...     40      0       0       ...     0
```

- **Prescribed Motion File**

The translation and rotation of the ground, where the tower bottom of the wind turbine is constrained, can be prescribed using a prescribed motion file o to the “ground” to which a bottom fixed turbine is directly connected. If a floating wind turbine is simulated the prescribed motion will only affect elemen line anchors). By using the keyword *CONSTRAINEDFLOATER* in the turbine substructure definition it is also possible to prescribe the translation/rotation c

Listing 99 : An exemplary prescribed motion file

```
Time    TransX  TransY  TransZ  RotX    RotY    RotZ
0       1       11      0       0       0       0
5       2       11      0       0       0       0
10      4       11      0       5       0       0
15      7       11      0       10      0       0
20      11      11      0       17      0       0
25      12      11      0       27      0       0
30      13      11      0       40      10      10
35      12      11      0       40      20      20
40      11      11      0       40      30      30
45      11      11      0       40      40      40
50      11      11      0       40      40      40
```

Multi Turbine Simulations

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

If enabled multiple turbines may be added to a single simulation object and their wake interaction can be evaluated. Find more information in the section

Turbine Environment

- **Installation**: The user can chose between *Offshore* and *Onshore* installation. If *Offshore* is selected the user must also specify the water depth.
- **Water Depth**: Sets the water depth for an offshore simulation.

Wave Boundary Conditions

These edits are only enabled if *Offshore* installation is selected.

- **Wave Type**: Toggles if a linear wave should be included in the simulation.
- **Kinematic Stretching**: Choose the Kinematic Stretching type if a linear wave is selected.
- **Linear Wave**: A wave from QBlades database can be selected.
- **Linear Wave Object**: The currently selected *wave object* can be edited or a new *wave object* can be created.

Ocean Current Boundary Conditions

- **Near Surf: U, Dir, Dep**: Sets velocity, direction and depth parameters for **Near Surface Currents**, see Currents.
- **Sub Surf: U, Dir, Exp**: Sets velocity, direction and exponent parameters for **Sub Surface Currents**, see Currents.
- **Near Shore: U, Dir**: Sets velocity and direction for **Near Shore Currents**, see Currents.

Environmental Variables

The user can set the environmental parameters that are used during the simulation and for the evaluation of several quantities such as *Reynolds Number* c
parameters is shown below:

- **Gravity**
- **Air Density**
- **Kinematic Viscosity (Air)**
- **Water Density**
- **Kinematic Viscosity (Water)**

Seabed Modelling

To prevent the mooring lines from penetrating the seabed, the seabed is modelled as vertically oriented spring/dampers that act on the mooring line elem
implemented is highly similar to the work of Hall[1].

- **Seabed Stiffness**: The spring stiffness coefficient for the seabed model (acting in the vertical direction only).
- **Seabed Damping Factor**: The seabed damping coefficient, as a fraction of the spring stiffness coefficient (acting in the vertical direction only).
- **Seabed Fraction Factor**: The seabed fraction coefficient for the seabed model, as a fraction of the spring stiffness coefficient (acting in the horizontal c

Stored Simulation Data

The uer can choose here to only store a certain type of simulation data (to limit the project file or data export size. Furthermore, the user can choose to st
remove transients from the datasets.

- **Store Output From**: Simulation Data is only stored after the defined simulation time has passed.
- **Store Aero Time Data**: Toggles if this data type is stored. (All data that is shown in the *Aerodynamic Time Graph*).
- **Store Aero Blade Data**: Toggles if this data type is stored. (All data that is shown in the *Aerodynamic Blade Graph*).
- **Store Structural Data**: Toggles if this data type is stored. (All data that is shown in the *Structural Time Graph*).
- **Store Simulation Data**: Toggles if this data type is stored. (All data that is shown in the *Simulation Graph*).
- **Store Hydrodynamic Data**: Toggles if this data type is stored. (All data that is shown in the *Hydrodynamic Time Graph*).
- **Store Controller Data**: Toggles if this data type is stored. (All data that is shown in the *Controller Time Graph*).
- **Store DWM Wake Data**: Toggles if this data type is stored. (All data that is shown in the *DWM Graph*).

VPML Particle Remeshing

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

Free wake filaments may be converted into vortex particles. The following parameters govern the treatment of free vortex particles during a simulation.

- **Remeshing Scheme**
- **Remesh Update After X Steps**
- **Base Grid Size**
- **Particle Core Size Factor**
- **Magnitude Filter Factor**
- **Max. Stretch Factor**

Modal Analysis

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

When activated, a modal analysis is performed at the end of the simulation. This process linearizes the mass, stiffness, and damping matrices around the t
a Jacobian matrix, and geometric stiffness effects are added if enabled in the wind turbine definition. A generalized eigenvalue problem (GEP) is then set u
of the simulation, when the rotor reaches its initial azimuthal position, facilitating the comparison of mode shapes across different operating points, such a

- **Perform Modal Analysis at end**: Toggles if a modal analysis is performed at the end of the simulation.
- **Search From Min. Freq.**: Only modeshapes with an Eigen frequency above this value are stored.
- **Delta Freq.**: Only modeshapes that are spaced apart by this value are stored.
- **Number Modes**: The number of modes (starting from the lowest frequency) that will be stored.

When a model analysis has successfully been conducted, the modeshapes can be inspected in the GUI (see Fig. 112). For this purpose the currently visual
Window (see Fig. 111). In the menu the currently displayed mode can be changed (**Number**). Furthermore, the amplification of the mode shape (**Amp**), anc
relation to the translational modal displacements, can be set (**R.Scale**). Lastly, it is also possible to plot the Real, Imaginary, Magnitude of Phase of the moc

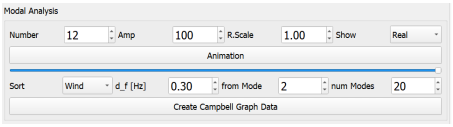


Fig. 111 *The Modal Analysis Dock Window*

By setting up multiple simulations, at different windspeeds or rotational speeds, it is also possible to create a Campbell diagram for a wind turbine. This is

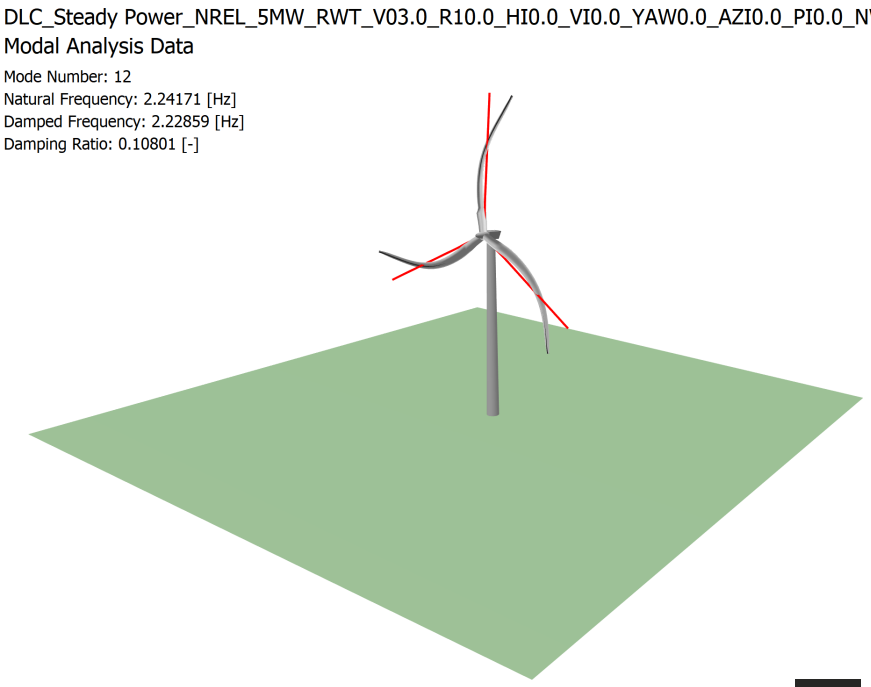


Fig. 112 *Visualization of a modeshape*

Dynamic Wake Meandering

Here the user can choose which wake summation method shall be applied to overlapping wakes of the Dynamic Wake Meandering Model. The different c

- **Dominant Wake**: The combined wake velocity is equal to the velocity of the wake with the largest velocity deficit (induction)
- **Sum of Squares**: The wake induced velocities are summed up by the sum of squares method.
- **Vector Sum**: The wake induced velocities are summed up by linear vector addition.

Ice Throw Simulation

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

A simulation of ice throw, shed from the rotor, can be performed in QBlade, see Lennie and Marten[2]. The following parameters govern the range of the ra
The distribution of *landed* ice particles can then be exported to generate iso-risk contours for the localized individual risk (*LIRA*) of a person being hit by a

- **Simulate Ice Throw**: Toggles if an Ice Throw Simulation is carried out.
- **Min. / Max. Drag**: Sets the range of drag values for the generated ice particles.
- **Min. / Max. Mass**: Set the range of masses for the generated ice particles.
- **Min. / Max. Density**: Set the range of density for the generated ice particles.
- **Min. / Max. Radius**: Set the range of ice particle release positions (in % of rotor radius).
- **Total N Particles**: Set the total number of ice particles that are generated during the simulation. This number will be evenly distributed over all timeste

Simulation Definition ASCII File

Simulation objects can be exported into the text based `.sim` format. When a simulation object is exported into the `.sim` format, the associated turbine exemplary `.sim` file below:

Listing 100 : A simulation definition ASCII file

```
--------------------------------------QBlade Simulation Definition File----------------------------------------
Generated with : QBlade CE v2.0.7-release_candidate_beta windows
Archive Format: 310023
Time : 21:27:48
Date : 15.05.2024

--------------------------------------Object Name--------------------------------------------------------------
OC4_Semi_Test                          OBJECTNAME         - the name of the simulation object

--------------------------------------Simulation Type----------------------------------------------------------
1                                      ISOFFSHORE         - use a number: 0 = onshore; 1 = offshore

--------------------------------------Turbine Parameters-------------------------------------------------------
multiple turbines can be added by adding multiple definitions encapsulated with TURB_X and END_TURB_X, where X must start at 1

TURB_1
    NREL_5MW_OC4_SEMI_RWT/NREL_5MW_OC4_SEMI_RWT.trb TURBFILE           - the turbine definition file that is used for this simulation
    NREL_5MW_OC4_SEMI_RWT              TURBNAME           - the (unique) name of the turbine in the simulation (results will appear under this na
    15.00                              INITIAL_YAW        - the initial turbine yaw in [deg]
    15.00                              INITIAL_PITCH      - the initial collective blade pitch in [deg]
    40.00                              INITIAL_AZIMUTH    - the initial azimuthal rotor angle in [deg]
    1                                  STRSUBSTEP         - the number of structural substeps per timestep (usually 1)
    5                                  RELAXSTEPS         - the number of initial static structural relaxation steps
    0                                  PRESCRIBETYPE      - rotor RPM prescribe type (0 = ramp-up; 1 = whole sim; 2 = no RPM prescibed)
    7.000                              RPMPRESCRIBED      - the prescribed rotor RPM [-]
    5                                  STRITERATIONS      - number of iterations for the time integration (used when integrator is HHT or Euler)
    0                                  MODNEWTONITER      - use the modified newton iteration?
    1                                  INCLUDEAERO        - include aerodynamic forces?
    1                                  INCLUDEHYDRO       - include hydrodynamic forces?
    0.00                               GLOBPOS_X          - the global x-position of the turbine [m]
    0.00                               GLOBPOS_Y          - the global y-position of the turbine [m]
    0.00                               GLOBPOS_Z          - the global z-position of the turbine [m]
    0.00                               GLOBROT_X          - the global x-rotation of the turbine [deg]
    0.00                               GLOBROT_Y          - the global y-rotation of the turbine [deg]
    0.00                               GLOBROT_Z          - the global z-rotation of the turbine [deg]
                                       EVENTFILE          - the file containing fault event definitions (leave blank if unused)
                                       LOADINGFILE        - the loading file name (leave blank if unused)
                                       SIMFILE            - the simulation file name (leave blank if unused)
                                       MOTIONFILE         - the prescribed motion file name (leave blank if unused)
    4.00                               FLOAT_SURGE        - the initial floater surge [m]
    3.00                               FLOAT_SWAY         - the initial floater sway [m]
    6.00                               FLOAT_HEAVE        - the initial floater heave [m]
    7.00                               FLOAT_ROLL         - the initial floater roll [deg]
    5.00                               FLOAT_PITCH        - the initial floater pitch [deg]
    9.00                               FLOAT_YAW          - the initial floater yaw [deg]
END_TURB_1

--------------------------------------Simulation Settings------------------------------------------------------
```

```
0.050000                              TIMESTEP           - the timestep size in [s]
1200                                  NUMTIMESTEPS       - the number of timesteps
10.000                                RAMPUP             - the rampup time for the structural model
5.000                                 ADDDAMP            - the initial time with additional damping
100.000                               ADDDAMPFACTOR      - for the additional damping time this factor is used to increase the damping of all cc
0.000                                 WAKEINTERACTION    - in case of multi-turbine simulation the wake interaction start at? [s]

---------------------------------------Wind Input-----------------------------------------------------------------
0                                     WNDTYPE            - use a number: 0 = steady; 1 = windfield; 2 = hubheight
                                      WNDNAME            - filename of the turbsim input file, mann input file or hubheight file (with extensior
0                                     STITCHINGTYPE      - the windfield stitching type; 0 = periodic; 1 = mirror
true                                  WINDAUTOSHIFT      - the windfield shifting automatically based on rotor diameter [bool]
0.00                                  SHIFTTIME          - the windfield is shifted by this time if WINDAUTOSHIFT = 0
12.00                                 MEANINF            - the mean inflow velocity, overridden if a windfield or hubheight file is use
0.00                                  HORANGLE           - the horizontal inflow angle
0.00                                  VERTANGLE          - the vertical inflow angle
0                                     PROFILETYPE        - the type of wind profile used (0 = Power Law; 1 = Logarithmic)
0.200                                 SHEAREXP           - the shear exponent if using a power law profile, if a windfield is used these values
0.010                                 ROUGHLENGTH        - the roughness length if using a log profile, if a windfield is used these values are
0.00                                  DIRSHEAR           - a value for the directional shear in deg/m
77.60                                 REFHEIGHT          - the reference height, used to contruct the BL profile

---------------------------------------Ocean Depth, Waves and Currents--------------------------------------------
the following parameters only need to be set if ISOFFSHORE = 1
200.00                                WATERDEPTH         - the water depth
New_Wave.lwa                          WAVEFILE           - the path to the wave file, leave blank if unused
1                                     WAVESTRETCHING     - the type of wavestretching, 0 = vertical, 1 = wheeler, 2 = extrapolation, 3 = none
10000.00                              SEABEDSTIFF        - the vertical seabed stiffness [N/m^3]
0.20                                  SEABEDDAMP         - a damping factor for the vertical seabed stiffness evaluation, between 0 and 1 [-]
0.10                                  SEABEDSHEAR        - a factor for the evaluation of shear forces (friction), between 0 and 1 [-]
0.00                                  SURF_CURR_U        - near surface current velocity [m/s]
0.00                                  SURF_CURR_DIR      - near surface current direction [deg]
30.00                                 SURF_CURR_DEPTH    - near surface current depth [m]
0.00                                  SUB_CURR_U         - sub surface current velocity [m/s]
0.00                                  SUB_CURR_DIR       - sub surface current direction [deg]
0.14                                  SUB_CURR_EXP       - sub surface current exponent
0.00                                  SHORE_CURR_U       - near shore (constant) current velocity [m/s]
0.00                                  SHORE_CURR_DIR     - near shore (constant) current direction [deg]

---------------------------------------Global Mooring System------------------------------------------------------
                                      MOORINGSYSTEM      - the path to the global mooring system file, leave blank if unused

---------------------------------------Dynamic Wake Meandering----------------------------------------------------
0                                     DWMSUMTYPE         - the dynamic wake meandering wake summation type: 0 = DOMINANT; 1 = QUADRATIC; 2 = LIN

---------------------------------------Environmental Parameters---------------------------------------------------
1.22500                               DENSITYAIR         - the air density [kg/m^3]
0.000016470                           VISCOSITYAIR       - the air kinematic viscosity
1025.00000                            DENSITYWATER       - the water density [kg/m^3]
0.000001307                           VISCOSITYWATER     - the water kinematic viscosity [m^2/s]
9.806650000                           GRAVITY            - the gravity constant [m/s^2]

---------------------------------------Output Parameters----------------------------------------------------------
20.00000                              STOREFROM          - the simulation stores data from this point in time, in [s]
false                                 STOREREPLAY        - store a replay of the simulation (warning, large memory will be required) [bool]
true                                  STOREAERO          - should the aerodynamic data be stored [bool]
false                                 STOREBLADE         - should the local aerodynamic blade data be stored [bool]
true                                  STORESTRUCT        - should the structural data be stored [bool]
true                                  STORESIM           - should the simulation (performance) data be stored [bool]
true                                  STOREHYDRO         - should the controller data be stored [bool]
false                                 STORECONTROLLER    - should the controller data be stored [bool]
false                                 STOREDWM           - should the dynamic wake meandering (DWM) data be stored [bool]

---------------------------------------Modal Analysis Parameters--------------------------------------------------
false                                 CALCMODAL          - perform a modal analysis (only single turbine simulations) [bool]
0.00000                               MINFREQ            - store Eigenvalues, starting with this frequency
0.00000                               DELTAFREQ          - omit Eigenvalues that are closer spaced than this value
100.00000                             NUMFREQ            - set the number of Eigenmodes and Eigenvalues that will be stored
```

Multi-Threaded Batch Analysis

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

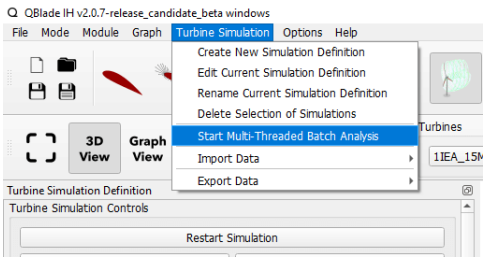


Fig. 113 *The multi-threaded batch menu option.*

Multiple simulations can be evaluated in a parallel batch queue through the dialog *Menu->Turbine Simulation->Multi-Threaded Batch Analysis*. The simulatic choosing the number of parallel threads the batch analysis starts by clicking the *Start Batch* button.

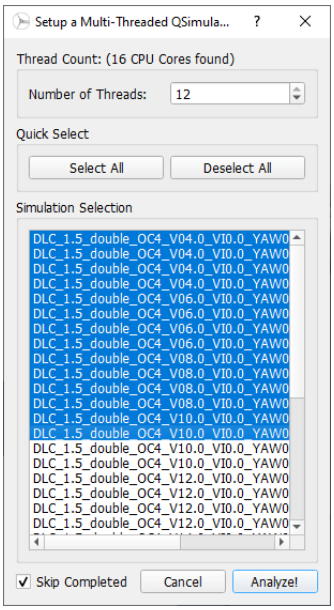


Fig. 114 *The multi-threaded batch analysis dialog.*

Exporting Simulation Results

QBlade can export simulation time data in a range of different formats. The available formats are:

- **QBlade_ASCII**: The QBlade ASCII format. A simple header per data column followed by the data in columns. This format can easily be imported into a
- **HAWC2_ASCII**: The HAWC2 ASCII format. A .sel file that contains the header information and a .dat file that contains the data in ASCII format.
- **HAWC2_BIN**: The HAWC2 binary format. A .sel file that contains the header and a .dat file in binary format that contains the data. Memory efficient.
- **OpenFAST_BIN**: The OpenFAST binary format. A single .outb file that contains header and data, both inbinary format. Memory efficient.

Information on the HAWC2 formats is found in the HOW2HAWC2 User Manual. This format is also compatible with DTU's PDap Software.

Information on the OpenFAST binary format is found in the OpenFAST Documentation.

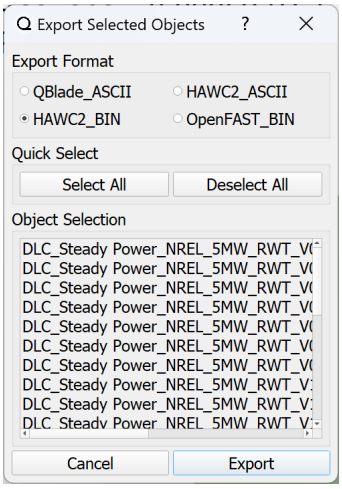


Fig. 115 *The Batch Export dialog*

Global Export Filter

If the user only needs to export a subsection of all data generated by QBlade, the *Global Export Filter* can be employed. It is found in the simulation menu, containing the exact variable names of QBlade, and the order in which the data should be exported. An exemplary list of variable names that can be used

Listing 101 : Exemplary list of variable names for the global export filter

```
Time [s]
Gen. HSS Torque [Nm]
X_c Tip Trl.Def. (OOP) BLD 1 [m]
X_b RootBend. Mom. BLD 1 [Nm]
```

Please note: Once the global filter is set, it is applied each time time-domain data is exported. When a project is stored, the global export filter that is set variable names in the export filter dialog, see Fig. 117.

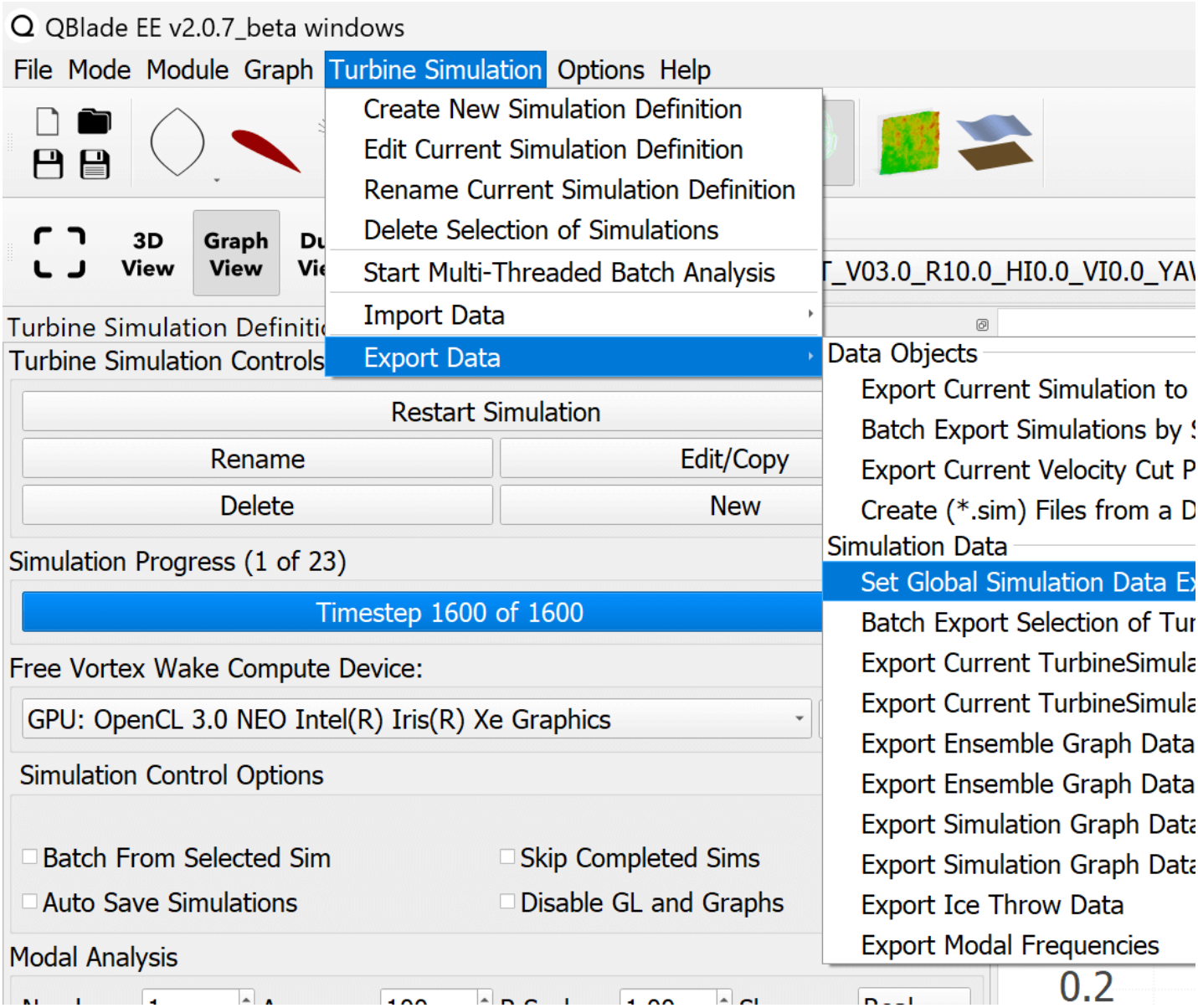


Fig. 116 *The Global Export Filter optioon in the menu*

Fig. 117 *The Global Export Filter optioon in the menu*

[1] M. Hall. Efficient modelling of seabed friction and multi-floater mooring systems in moordyn. *Proceedings of the 12th European Wave and Tidal Energy Conferenc*

[2] S. Lennie, M. Dominin and D. Marten. Development of ice throw model for wind turbine simulation software qblade. *AIAA Scitech 2019 Forum*, 2019. URL: http doi:10.2514/6.2019-1800.

Multi Turbine Simulation Setup

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

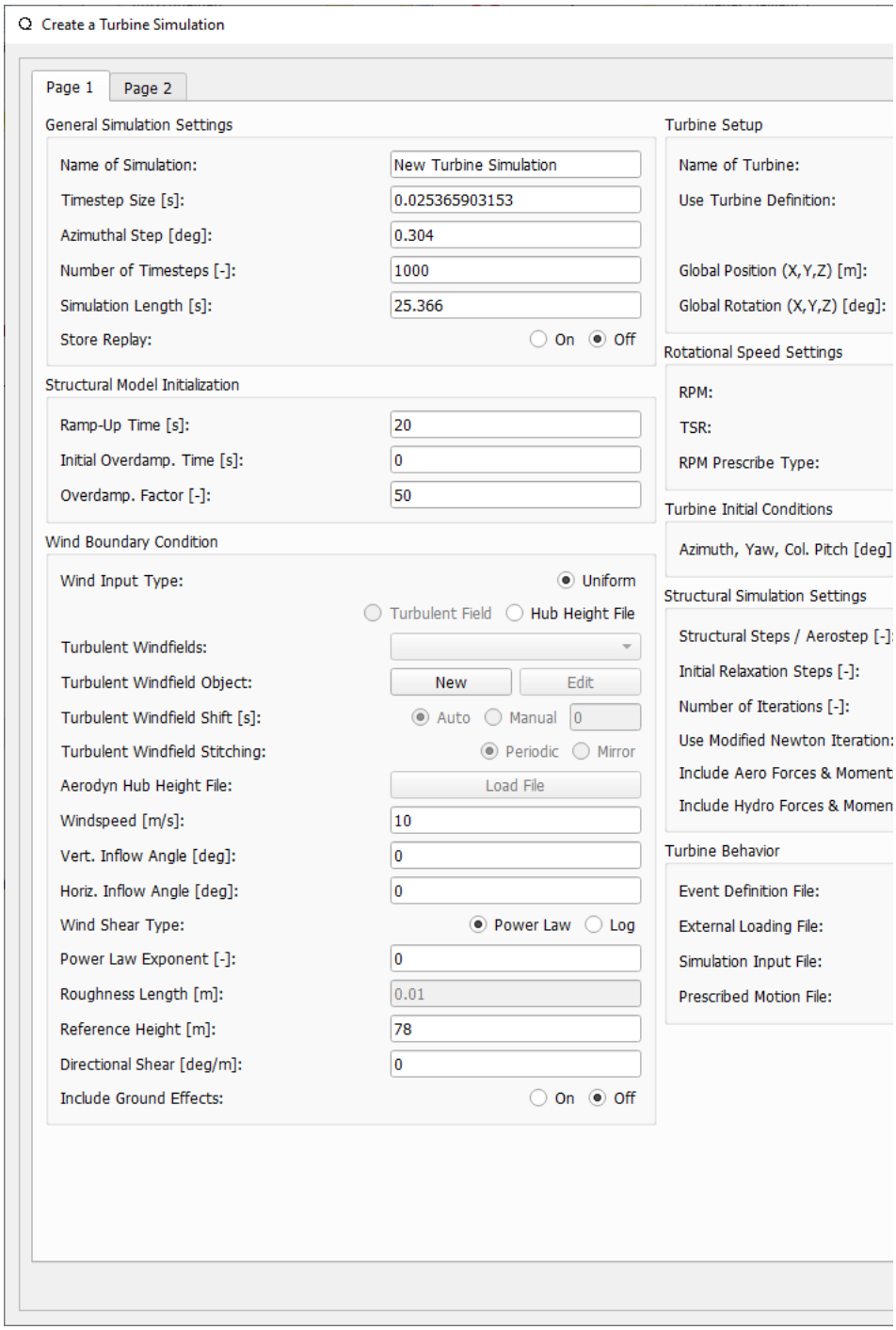


Fig. 128 *The multi-turbine*

To define a simulation containing multiple turbines the user needs to activate the multiple turbines option in the simulation dialog (see :fig:'fig-multi_turbi

Turbines can now be added manually, by clicking **Add**, or automated, through a **Wind Farm Layout File**. If added manually, thourgh the *Add** button, the cu

If a **Wind Farm Layout File** is used turbines can be add automated. This is especially useful when simulating a very large number of turbines. An exemplary

	0	1	2	3	4	5	6	7	8	9	10	11	12
	Turbine		Initial Position						Initial Conditi				
	Name	Turbine Object Nan	X	Y	Z	RX	RY	RZ	Yaw	Pitch	Azimuth	Surge	Sway
	Turb1	NREL_2.3-116	0	-675.397	0	0	0	0	-10	0	0	0	0
	Turb2	NREL_2.3-116	188.3657	-1027.78	0	0	0	0	-10	0	0	0	0
	Turb3	NREL_2.3-116	376.9829	-1365.87	0	0	0	0	-10	0	0	0	0
	Turb4	NREL_2.3-116	179.36	-3.96825	0	0	0	0	-10	0	0	0	0
	Turb5	NREL_2.3-116	372.6171	-342.063	0	0	0	0	-10	0	0	0	0
	Turb6	NREL_2.3-116	565.8743	-680.159	0	0	0	0	-10	0	0	0	0
	Turb7	NREL_2.3-116	749.8286	-1023.02	0	0	0	0	-10	0	0	0	0
	Turb8	NREL_2.3-116	938.4	-1370.63	0	0	0	0	-10	0	0	0	0
	Turb9	NREL_2.3-116	552.2057	338.8889	0	0	0	0	-10	0	0	0	0
	Turb10	NREL_2.3-116	745.4857	5.555556	0	0	0	0	-10	0	0	0	0
	Turb11	NREL_2.3-116	938.72	-337.302	0	0	0	0	-10	0	0	0	0
	Turb12	NREL_2.3-116	1131.931	-684.921	0	0	0	0	-10	0	0	0	0
	Turb13	NREL_2.3-116	1311.223	-1032.54	0	0	0	0	-10	0	0	0	0
	Turb14	NREL_2.3-116	1499.84	-1370.63	0	0	0	0	-10	0	0	0	0
	Turb15	NREL_2.3-116	934.3314	681.746	0	0	0	0	-10	0	0	0	0
	Turb16	NREL_2.3-116	1122.949	343.6508	0	0	0	0	-10	0	0	0	0
	Turb17	NREL_2.3-116	1311.543	0.793651	0	0	0	0	-10	0	0	0	0
	Turb18	NREL_2.3-116	1877.349	-1023.02	0	0	0	0	-10	0	0	0	0
	Turb19	NREL_2.3-116	2065.92	-1370.63	0	0	0	0	-10	0	0	0	0
	Turb20	NREL_2.3-116	1311.817	1024.603	0	0	0	0	-10	0	0	0	0
	Turb21	NREL_2.3-116	1500.434	686.5079	0	0	0	0	-10	0	0	0	0
	Turb22	NREL_2.3-116	1684.389	343.6508	0	0	0	0	-10	0	0	0	0
	Turb23	NREL_2.3-116	1873.006	5.555556	0	0	0	0	-10	0	0	0	0

Fig. 129 *An excel sheet containing the farm la*

Listing 103 : An exemplary

```
Name    Turbine Object Name     X       Y       Z       RX      RY      RZ      Yaw     Pitch   Azimuth Surge   Sway    Heave   Plat.Roll       Pla
Turb1   NREL_2.3-116    0.0     -675.4  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb2   NREL_2.3-116    188.4   -1027.8 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb3   NREL_2.3-116    377.0   -1365.9 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb4   NREL_2.3-116    179.4   -4.0    0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb5   NREL_2.3-116    372.6   -342.1  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb6   NREL_2.3-116    565.9   -680.2  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb7   NREL_2.3-116    749.8   -1023.0 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb8   NREL_2.3-116    938.4   -1370.6 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb9   NREL_2.3-116    552.2   338.9   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb10  NREL_2.3-116    745.5   5.6     0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb11  NREL_2.3-116    938.7   -337.3  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb12  NREL_2.3-116    1131.9  -684.9  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb13  NREL_2.3-116    1311.2  -1032.5 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb14  NREL_2.3-116    1499.8  -1370.6 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb15  NREL_2.3-116    934.3   681.7   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb16  NREL_2.3-116    1122.9  343.7   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb17  NREL_2.3-116    1311.5  0.8     0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb18  NREL_2.3-116    1877.3  -1023.0 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb19  NREL_2.3-116    2065.9  -1370.6 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb20  NREL_2.3-116    1311.8  1024.6  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb21  NREL_2.3-116    1500.4  686.5   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb22  NREL_2.3-116    1684.4  343.7   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb23  NREL_2.3-116    1873.0  5.6     0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb24  NREL_2.3-116    2066.2  -346.8  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb25  NREL_2.3-116    2250.2  -684.9  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb26  NREL_2.3-116    2438.8  -1027.8 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb27  NREL_2.3-116    2627.4  -1365.9 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb28  NREL_2.3-116    1680.0  1367.5  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb29  NREL_2.3-116    1873.3  1024.6  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb30  NREL_2.3-116    2057.2  686.5   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb31  NREL_2.3-116    2250.4  338.9   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb32  NREL_2.3-116    2439.1  5.6     0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb33  NREL_2.3-116    2627.6  -346.8  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb34  NREL_2.3-116    2816.3  -684.9  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb35  NREL_2.3-116    3000.2  -1027.8 0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb36  NREL_2.3-116    2052.9  1710.3  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb37  NREL_2.3-116    2246.1  1362.7  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb38  NREL_2.3-116    2434.7  1029.4  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb39  NREL_2.3-116    2627.9  681.7   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb40  NREL_2.3-116    2811.9  338.9   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb41  NREL_2.3-116    3005.2  5.6     0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb42  NREL_2.3-116    3189.1  -337.3  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb43  NREL_2.3-116    3382.4  -680.2  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb44  NREL_2.3-116    2435.0  2053.2  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb45  NREL_2.3-116    2623.6  1705.6  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb46  NREL_2.3-116    2807.6  1372.2  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb47  NREL_2.3-116    3005.4  1029.4  0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
Turb48  NREL_2.3-116    3170.7  667.5   0.0     0       0       0       -10     0       0       0       0       0       0       0       0       2
```

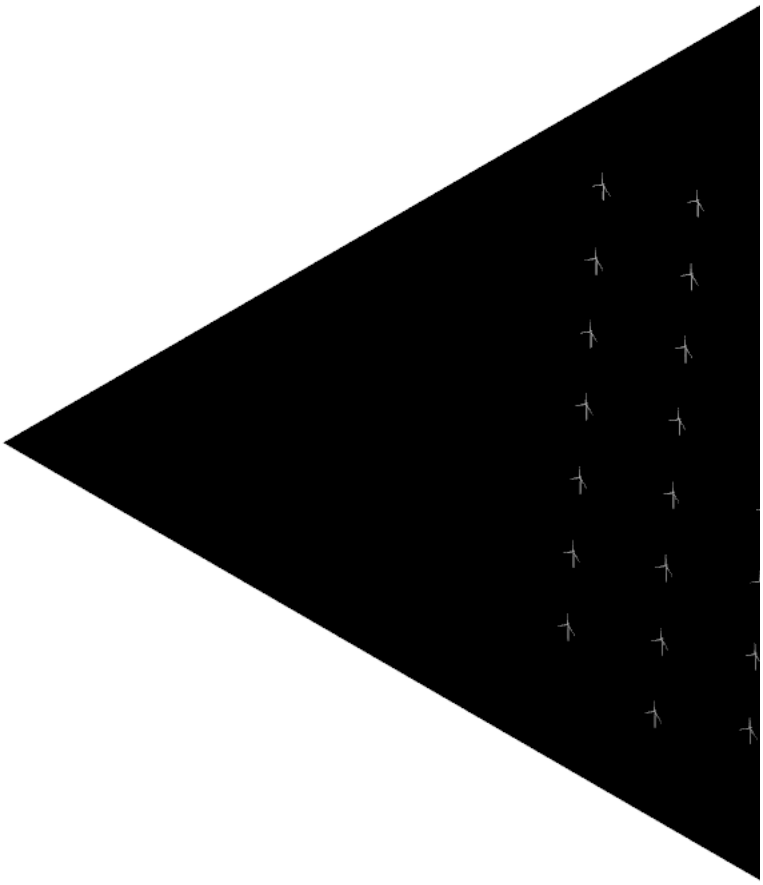


Fig. 130 *The Wind Farm Layout i*

Furthermore, it is also possible to add a **global mooring system** to a multi turbine simulation. More information on this is found in the section Multi Turbir

Multi Turbine Global Mooring System

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

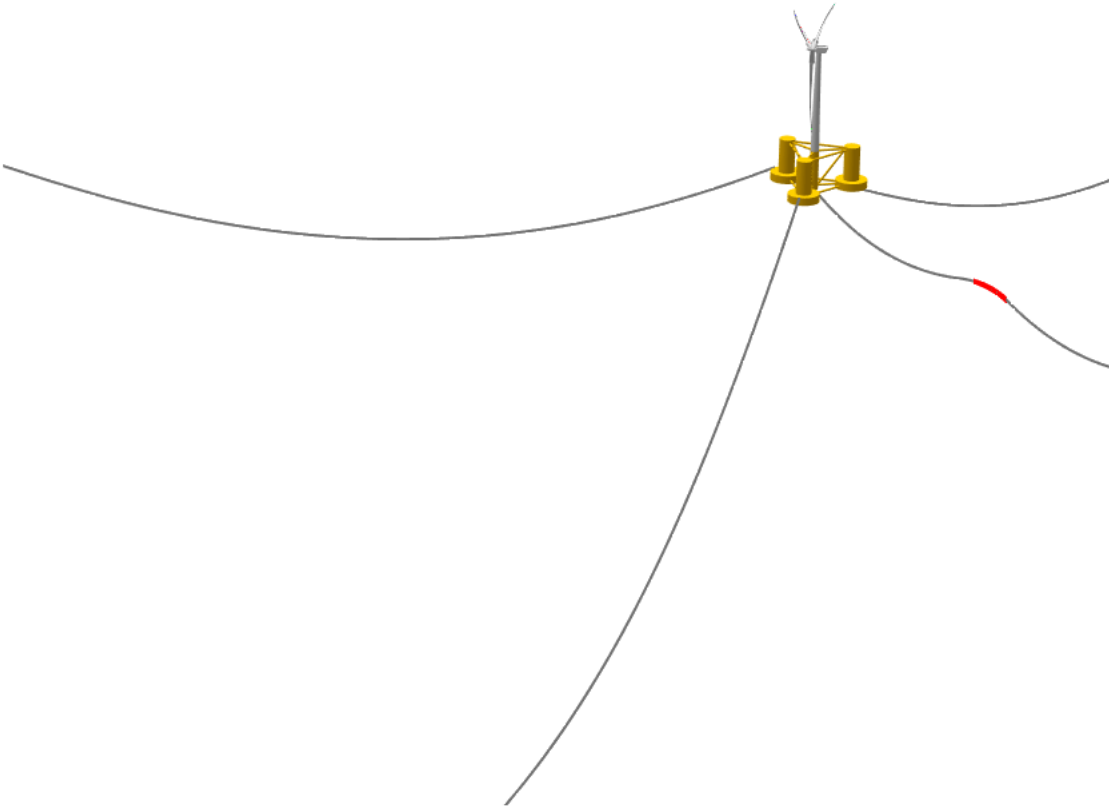


Fig. 131 *A global mooring definition,*

For multi-turbine simulations it is also possible to define a global mooring system. A global mooring system can be defined as an interconnection between **Joint 43 of Turbine 2** (JNT_2_43). In general, the global mooring system definition can contain a **MOORELEMENTS** table, a **MOORMEMBERS** table, and

Furthermore, it is also possible to include **SUBMEMBERS**, **SUBELEMENTS**, **SUBCONSTRAINTS** and **NLSPRINGDAMPERS** and **MOORLOADS** in the sam

The simulation data that is stored from the global mooring system can be viewed in the *Simulation Time Graph*.

The global mooring system definition file shown below is used to setup the mooring conficuration that is shown in Fig. 131.

Listing 104 : A global mooring

```
true    ISFLOATING
100     ADVANCEDBUOYANCY

1.00    STIFFTUNER
1.00    MASSTUNER
1.00    BUOYANCYTUNER

SUBJOINTS
JointID JointX  JointY  JointZ
1       0.00000 0.00000 -10.00000
2       0.00000 0.00000 10.00000
3       0.00000 0.00000 -13.00000

SUBELEMENTSRIGID
ElemID  BMASSD  DIAMETER
1       10000   5.5
2       100000  5.5

HYDROJOINTCOEFF
CoeffID JointID CdA     CaA     CpA
1       2       4.8     1.0     1.0
2       3       4.8     1.0     1.0

SUBMEMBERS
MemID   Jnt1ID  Jnt2ID  ElmID   ElmRot  HyCoID  IsBuoy  MaGrID  FldArea ElmDsc  Name    (optional)
1       1       2       1       0       1       1       0       0       2       Main_Colum
```

```
2       1       3       2       0       1       1       0       0       2       Main_Column2

HYDROMEMBERCOEFF
CoeffID CdN     CaN     CpN     MCFC
1       2.0     0.8     1.0     0

MOORELEMENTS
ID      Dens.[kg/m^3]   Area[m^2]       Iyy[m^4]        EMod[N/m^4]     RDp.[-] Dia[m]
1       2.35723E+04     4.6084E-03      3.7601E-03      1.6353E+11      0.015   0.0766
2       6.35723E+04     4.6084E-03      3.7601E-04      1.6353E+10      0.005   0.0766

MOORMEMBERS
ID      CONN_1          CONN_2  Len.[m] MoorID  HyCoID  IsBuoy  MaGrID  ElmDsc  Name
1       JNT_1_43        JNT_3   270     1       1       1       0       30      Mooring1
2       JNT_2_43        JNT_3   270     1       1       1       0       30      Mooring2
3       JNT_1_1         JNT_2_1 700     2       1       1       0       30      Power

MOORLOADS
3       150     180     16000
3       520     550     16000

RGBCOLOR
255     0       0

------------------------------ DATA OUTPUT TYPES -------------------------
true    FOR_OUT
true    ROT_OUT
true    MOM_OUT
true    DEF_OUT
true    POS_OUT
true    VEL_OUT
true    ACC_OUT
true    LVE_OUT
true    LAC_OUT

------------------------------ SENSORS -----------------------------------
SUB_1_0.5
MOO_1_0.2
```

Multi Turbine Simulation Definition ASCII File

Within the *Simulation Definition ASCII Files* a multi-turbine simulation can be defined in two ways. The first option is to specify the path to the **Farm Layou**

The second options is is to defined multiple turbines by encapsulating each turbine object by *TURB_X* and *END_TURB_X* where *X* is the turbine number st

Listing 105 : A multi turbine s

```
---------------------------------------QBlade Simulation Definition File----------------------------------------
Generated with : QBlade IH v2.0.7-release_candidate_beta windows
Archive Format: 310023
Time : 21:43:28
Date : 15.05.2024

---------------------------------------Object Name--------------------------------------------------------------
New_Turbine_Simulation                  OBJECTNAME          - the name of the simulation object

---------------------------------------Simulation Type----------------------------------------------------------
0                                       ISOFFSHORE          - use a number: 0 = onshore; 1 = offshore

---------------------------------------Turbine Parameters-------------------------------------------------------
NREL_2.3-116/NREL_2.3-116.trb           TURBFILE            - the turbine definition file(s) used in this simulation
farmLayout.xlsx                         FARMLAYOUT          - the farmlayout file (if existing)

---------------------------------------Simulation Settings------------------------------------------------------
0.025366                                TIMESTEP            - the timestep size in [s]
1000                                    NUMTIMESTEPS        - the number of timesteps
20.000                                  RAMPUP              - the rampup time for the structural model
0.000                                   ADDDAMP             - the initial time with additional damping
50.000                                  ADDDAMPFACTOR       - for the additional damping time this factor is used to increase the damping of all co
0.000                                   WAKEINTERACTION     - in case of multi-turbine simulation the wake interaction start at? [s]

---------------------------------------Wind Input---------------------------------------------------------------
0                                       WNDTYPE             - use a number: 0 = steady; 1 = windfield; 2 = hubheight
                                        WNDNAME             - filename of the turbsim input file, mann input file or hubheight file (with extensior
0                                       STITCHINGTYPE       - the windfield stitching type; 0 = periodic; 1 = mirror
true                                    WINDAUTOSHIFT       - the windfield shifting automatically based on rotor diameter [bool]
0.00                                    SHIFTTIME           - the windfield is shifted by this time if WINDAUTOSHIFT = 0
10.00                                   MEANINF             - the mean inflow velocity, overridden if a windfield or hubheight file is use
0.00                                    HORANGLE            - the horizontal inflow angle
0.00                                    VERTANGLE           - the vertical inflow angle
0                                       PROFILETYPE         - the type of wind profile used (0 = Power Law; 1 = Logarithmic)
0.000                                   SHEAREXP            - the shear exponent if using a power law profile, if a windfield is used the
0.010                                   ROUGHLENGTH         - the roughness length if using a log profile, if a windfield is used these v
0.00                                    DIRSHEAR            - a value for the directional shear in deg/m
78.00                                   REFHEIGHT           - the reference height, used to contruct the BL profile
```

```
---------------------------------------Ocean Depth, Waves and Currents-----------------------------------------
the following parameters only need to be set if ISOFFSHORE = 1
1.00                                  WATERDEPTH         - the water depth
                                      WAVEFILE           - the path to the wave file, leave blank if unused
1                                     WAVESTRETCHING     - the type of wavestretching, 0 = vertical, 1 = wheeler, 2 = extrapolation, 3 = none
10000.00                              SEABEDSTIFF        - the vertical seabed stiffness [N/m^3]
0.20                                  SEABEDDAMP         - a damping factor for the vertical seabed stiffness evaluation, between 0 and 1 [-]
0.10                                  SEABEDSHEAR        - a factor for the evaluation of shear forces (friction), between 0 and 1 [-]
0.00                                  SURF_CURR_U        - near surface current velocity [m/s]
0.00                                  SURF_CURR_DIR      - near surface current direction [deg]
30.00                                 SURF_CURR_DEPTH    - near surface current depth [m]
0.00                                  SUB_CURR_U         - sub surface current velocity [m/s]
0.00                                  SUB_CURR_DIR       - sub surface current direction [deg]
0.14                                  SUB_CURR_EXP       - sub surface current exponent
0.00                                  SHORE_CURR_U       - near shore (constant) current velocity [m/s]
0.00                                  SHORE_CURR_DIR     - near shore (constant) current direction [deg]

---------------------------------------Global Mooring System---------------------------------------------------
                                      MOORINGSYSTEM      - the path to the global mooring system file, leave blank if unused

---------------------------------------Dynamic Wake Meandering-------------------------------------------------
2                                     DWMSUMTYPE         - the dynamic wake meandering wake summation type: 0 = DOMINANT; 1 = QUADRATIC; 2 = LIN

---------------------------------------Environmental Parameters------------------------------------------------
1.22500                               DENSITYAIR         - the air density [kg/m^3]
0.000016470                           VISCOSITYAIR       - the air kinematic viscosity
1025.00000                            DENSITYWATER       - the water density [kg/m^3]
0.000001307                           VISCOSITYWATER     - the water kinematic viscosity [m^2/s]
9.806650000                           GRAVITY            - the gravity constant [m/s^2]

---------------------------------------Output Parameters-------------------------------------------------------
0.00000                               STOREFROM          - the simulation stores data from this point in time, in [s]
false                                 STOREREPLAY        - store a replay of the simulation (warning, large memory will be required) [bool]
true                                  STOREAERO          - should the aerodynamic data be stored [bool]
true                                  STOREBLADE         - should the local aerodynamic blade data be stored [bool]
true                                  STORESTRUCT        - should the structural data be stored [bool]
true                                  STORESIM           - should the simulation (performance) data be stored [bool]
true                                  STOREHYDRO         - should the controller data be stored [bool]
false                                 STORECONTROLLER    - should the controller data be stored [bool]
false                                 STOREDWM           - should the dynamic wake meandering (DWM) data be stored [bool]

---------------------------------------Modal Analysis Parameters-----------------------------------------------
false                                 CALCMODAL          - perform a modal analysis (only single turbine simulations) [bool]
0.00000                               MINFREQ            - store Eigenvalues, starting with this frequency
0.00000                               DELTAFREQ          - omit Eigenvalues that are closer spaced than this value
100.00000                             NUMFREQ            - set the number of Eigenmodes and Eigenvalues that will be stored
```

Listing 106 : A multi turbine simulation d

```
---------------------------------------QBlade Simulation Definition File---------------------------------------
Generated with : QBlade IH v2.0.6_beta_dev windows
Archive Format: 310012
Time : 19:16:58
Date : 18.05.2023

---------------------------------------Object Name-------------------------------------------------------------
New_Turbine_Simulation                OBJECTNAME         - the name of the simulation object

---------------------------------------Simulation Type---------------------------------------------------------
1                                     ISOFFSHORE         - use a number: 0 = onshore; 1 = offshore

---------------------------------------Turbine Parameters------------------------------------------------------
multiple turbines can be added by adding multiple definitions encapsulated with TURB_X and END_TURB_X, where X must start at 1

TURB_1
    NREL_5MW_OC4_SEMI_RWT/NREL_5MW_OC4_SEMI_RWT.trb TURBFILE           - the turbine definition file that is used for this simulation
    NREL_5MW_OC4_SEMI_RWT             TURBNAME           - the (unique) name of the turbine in the simulation (results will appear under this na
    0.00                              INITIAL_YAW        - the initial turbine yaw in [deg]
    0.00                              INITIAL_PITCH      - the initial collective blade pitch in [deg]
    0.00                              INITIAL_AZIMUTH    - the initial azimuthal rotor angle in [deg]
    1                                 STRSUBSTEP         - the number of structural substeps per timestep (usually 1)
    5                                 RELAXSTEPS         - the number of initial static structural relaxation steps
    0                                 PRESCRIBETYPE      - rotor RPM prescribe type (0 = ramp-up; 1 = whole sim; 2 = no RPM prescibed)
    4.000                             RPMPRESCRIBED      - the prescribed rotor RPM [-]
    10                                STRITERATIONS      - number of iterations for the time integration (used when integrator is HHT or Euler)
    1                                 MODNEWTONITER      - use the modified newton iteration?
    300.00                            GLOBPOS_X          - the global x-position of the turbine [m]
    0.00                              GLOBPOS_Y          - the global y-position of the turbine [m]
    0.00                              GLOBPOS_Z          - the global z-position of the turbine [m]
    0.00                              GLOBROT_X          - the global x-rotation of the turbine [deg]
    0.00                              GLOBROT_Y          - the global y-rotation of the turbine [deg]
    0.00                              GLOBROT_Z          - the global z-rotation of the turbine [deg]
                                      EVENTFILE          - the file containing fault event definitions (leave blank if unused)
                                      LOADINGFILE        - the loading file name (leave blank if unused)
                                      SIMFILE            - the simulation file name (leave blank if unused)
                                      MOTIONFILE         - the prescribed motion file name (leave blank if unused)
    0.00                              FLOAT_SURGE        - the initial floater surge [m]
```

```
    0.00                                FLOAT_SWAY          - the initial floater sway [m]
    0.00                                FLOAT_HEAVE         - the initial floater heave [m]
    0.00                                FLOAT_ROLL          - the initial floater roll [deg]
    0.00                                FLOAT_PITCH         - the initial floater pitch [deg]
    0.00                                FLOAT_YAW           - the initial floater yaw [deg]
END_TURB_1

TURB_2
    NREL_5MW_OC4_SEMI_RWT-2/NREL_5MW_OC4_SEMI_RWT-2.trb TURBFILE           - the turbine definition file that is used for this simulation
    NREL_5MW_OC4_SEMI_RWT-2             TURBNAME            - the (unique) name of the turbine in the simulation (results will appear under this na
    180.00                              INITIAL_YAW         - the initial turbine yaw in [deg]
    0.00                                INITIAL_PITCH       - the initial collective blade pitch in [deg]
    0.00                                INITIAL_AZIMUTH     - the initial azimuthal rotor angle in [deg]
    1                                   STRSUBSTEP          - the number of structural substeps per timestep (usually 1)
    5                                   RELAXSTEPS          - the number of initial static structural relaxation steps
    0                                   PRESCRIBETYPE       - rotor RPM prescribe type (0 = ramp-up; 1 = whole sim; 2 = no RPM prescibed)
    4.000                               RPMPRESCRIBED       - the prescribed rotor RPM [-]
    10                                  STRITERATIONS       - number of iterations for the time integration (used when integrator is HHT or Euler)
    1                                   MODNEWTONITER       - use the modified newton iteration?
    -300.00                             GLOBPOS_X           - the global x-position of the turbine [m]
    0.00                                GLOBPOS_Y           - the global y-position of the turbine [m]
    0.00                                GLOBPOS_Z           - the global z-position of the turbine [m]
    0.00                                GLOBROT_X           - the global x-rotation of the turbine [deg]
    0.00                                GLOBROT_Y           - the global y-rotation of the turbine [deg]
    0.00                                GLOBROT_Z           - the global z-rotation of the turbine [deg]
                                        EVENTFILE           - the file containing fault event definitions (leave blank if unused)
                                        LOADINGFILE         - the loading file name (leave blank if unused)
                                        SIMFILE             - the simulation file name (leave blank if unused)
                                        MOTIONFILE          - the prescribed motion file name (leave blank if unused)
    0.00                                FLOAT_SURGE         - the initial floater surge [m]
    0.00                                FLOAT_SWAY          - the initial floater sway [m]
    0.00                                FLOAT_HEAVE         - the initial floater heave [m]
    0.00                                FLOAT_ROLL          - the initial floater roll [deg]
    0.00                                FLOAT_PITCH         - the initial floater pitch [deg]
    180.00                              FLOAT_YAW           - the initial floater yaw [deg]
END_TURB_2

---------------------------------------Simulation Settings-----------------------------------------------------
0.050000                                TIMESTEP            - the timestep size in [s]
800                                     NUMTIMESTEPS        - the number of timesteps
20.000                                  RAMPUP              - the rampup time for the structural model
0.000                                   ADDDAMP             - the initial time with additional damping
100.000                                 ADDDAMPFACTOR       - for the additional damping time this factor is used to increase the damping of all co
0.000                                   WAKEINTERACTION     - in case of multi-turbine simulation the wake interaction start at? [s]

---------------------------------------Wind Input--------------------------------------------------------------
0                                       WNDTYPE             - use a number: 0 = steady; 1 = windfield; 2 = hubheight
                                        WNDNAME             - filename of the turbsim input file or hubheight file (with extension), leave blank if
0                                       STITCHINGTYPE       - the windfield stitching type; 0 = periodic; 1 = mirror
1                                       WINDAUTOSHIFT       - the windfield shifting automatically based on rotor diameter; 0 = false; 1 = true
0.00                                    SHIFTTIME           - the windfield is shifted by this time if WINDAUTOSHIFT = 0
10.00                                   MEANINF             - the mean inflow velocity, overridden if a windfield or hubheight file is use
0.00                                    HORANGLE            - the horizontal inflow angle
0.00                                    VERTANGLE           - the vertical inflow angle
0                                       PROFILETYPE         - the type of wind profile used (0 = Power Law; 1 = Logarithmic)
0.000                                   SHEAREXP            - the shear exponent if using a power law profile, if a windfield is used these values
0.010                                   ROUGHLENGTH         - the roughness length if using a log profile, if a windfield is used these values are
0.00                                    DIRSHEAR            - a value for the directional shear in deg/m
77.60                                   REFHEIGHT           - the reference height, used to contruct the BL profile

---------------------------------------Ocean Depth, Waves and Currents-----------------------------------------
the following parameters only need to be set if ISOFFSHORE = 1
200.00                                  WATERDEPTH          - the water depth
New_Wave.lwa                            WAVEFILE            - the path to the wave file, leave blank if unused
1                                       WAVESTRETCHING      - the type of wavestretching, 0 = vertical, 1 = wheeler, 2 = extrapolation, 3 = none
10000.00                                SEABEDSTIFF         - the vertical seabed stiffness [N/m^3]
0.50                                    SEABEDDAMP          - a damping factor for the vertical seabed stiffness evaluation, between 0 and 1 [-]
0.00                                    SEABEDSHEAR         - a factor for the evaluation of shear forces (friction), between 0 and 1 [-]
0.00                                    SURF_CURR_U         - near surface current velocity [m/s]
0.00                                    SURF_CURR_DIR       - near surface current direction [deg]
30.00                                   SURF_CURR_DEPTH     - near surface current depth [m]
0.00                                    SUB_CURR_U          - sub surface current velocity [m/s]
0.00                                    SUB_CURR_DIR        - sub surface current direction [deg]
0.14                                    SUB_CURR_EXP        - sub surface current exponent
0.00                                    SHORE_CURR_U        - near shore (constant) current velocity [m/s]
0.00                                    SHORE_CURR_DIR      - near shore (constant) current direction [deg]

---------------------------------------Global Mooring System---------------------------------------------------
mooring.txt                             MOORINGSYSTEM       - the path to the global mooring system file, leave blank if unused

---------------------------------------Environmental Parameters------------------------------------------------
1.22500                                 DENSITYAIR          - the air density [kg/m^3]
0.000016470                             VISCOSITYAIR        - the air kinematic viscosity
1025.00000                              DENSITYWATER        - the water density [kg/m^3]
0.000001307                             VISCOSITYWATER      - the water kinematic viscosity [m^2/s]
9.806650000                             GRAVITY             - the gravity constant [m/s^2]

---------------------------------------Output Parameters-------------------------------------------------------
0                                       STOREREPLAY         - store a replay of the simulation: 0 = off, 1 = on (warning, large memory will be requ
20.000                                  STOREFROM           - the simulation stores data from this point in time, in [s]
```

```
1                                       STOREAERO          - should the aerodynamic data be stored (0 = OFF; 1 = ON)
0                                       STOREBLADE         - should the local aerodynamic blade data be stored (0 = OFF; 1 = ON)
1                                       STORESTRUCT        - should the structural data be stored (0 = OFF; 1 = ON)
1                                       STORESIM           - should the simulation (performance) data be stored (0 = OFF; 1 = ON)
1                                       STOREHYDRO         - should the controller data be stored (0 = OFF; 1 = ON)
0                                       STORECONTROLLER    - should the controller data be stored (0 = OFF; 1 = ON)
---------------------------------------Modal Analysis Parameters-------------------------------------------------
0                                       CALCMODAL          - perform a modal analysis after the simulation has completed (only for single turbine
0.00000                                 MINFREQ            - store Eigenvalues, starting with this frequency
0.00000                                 DELTAFREQ          - omit Eigenvalues that are closer spaced than this value
```

Wind Field Generator Overview

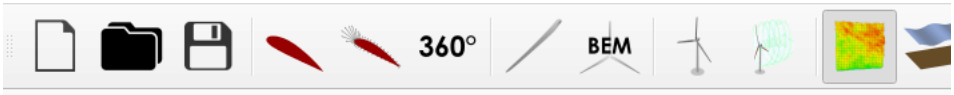


Fig. 132 *The wind field creation symbol in the QBlade main tool bar.*

The Wind Field Generator in QBlade is essential for defining the atmospheric conditions affecting the turbine during simulations. This section provides ar fields, including turbulent, uniform, and hub-height wind fields, each crucial for various simulation scenarios. Below, we describe the options available for settings and parameters necessary to tailor the wind conditions to your specific simulation needs.

Turbulent Wind Field

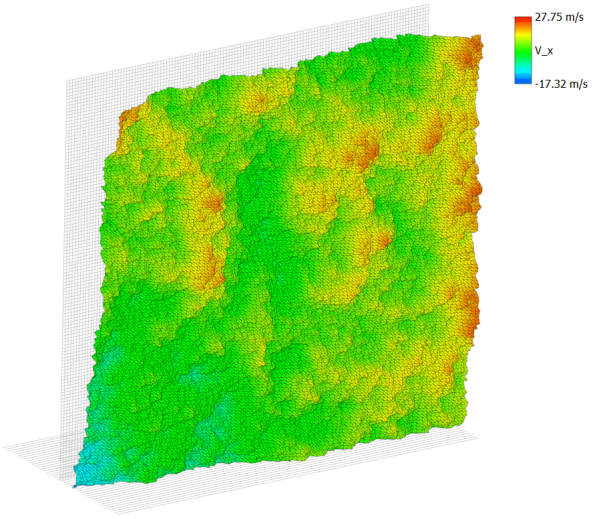


Fig. 133 *A turbulent wind field generated in QBlade.*

This section describes the process of generating a three-dimensional, fully turbulent wind field. Turbulent wind fields are essential for simulating real-worl performance under variable wind speeds and directions.

This can be either generated through the *Wind Input Type* button of the turbine simulation dialogue, as shown in Fig. 138 or by directly generating this wit

Three different options, to generate a three dimensional, fully turbulent wind field exist in QBlade.

- **TurbSim**: Generates the wind field using NREL's TurbSim binary (see B. J. Jonkman [1]).
- **Mann**: Generates the wind field using DTU`s Mann generator (see J. Mann [2]).
- **Veers**: Generates the wind field after the Veers algorithm (see P. S. Veers [3]).

TurbSim Wind Fields

When a new TurbSim wind field is created, a range of parameters must be specified as shown by the wind field generator dialogue in Fig. 134. After these automatically passes the information to the TurbSim program [1]. The TurnSim binary is automatically called by QBlade, and after creation the wind field is a input is required. The input parameters are described in detail in the following section.

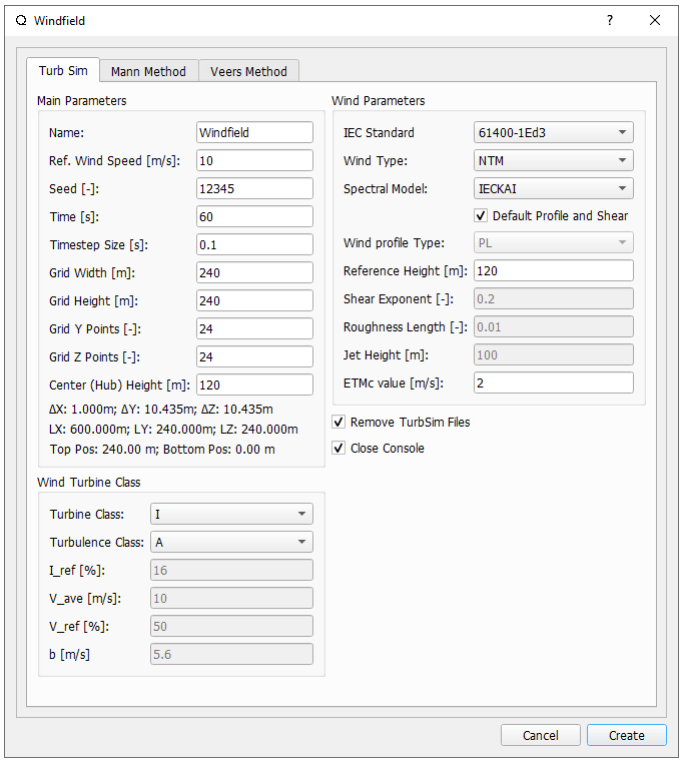


Fig. 134 *Turb sim wind field dialogue in QBlade.*

Main Parameters

These parameters dictate the spatial dimensions of the generated turbulent wind grid. A turbulent *box* is be generated which is then translated throug below) as is consistent with Taylor's hypothesis for a turbulent flow [4].

- **Name**: The name of the wind field object.
- **Ref. Wind Speed**: The reference wind speed of this wind field.
- **Seed**: The random seed used to generate the wind field.
- **Time**: Determines the length of the generated turbulent box.
- **Timestep**: Specifies the discretisation in free stream (x) direction.
- **Grid Width**: Specifies box size in lateral (y) direction.
- **Grid Height**: Specifies box size in vertical (z) direction.
- **Grid Y Points**: Specifies spatial discretisation in y direction.
- **Grid Z Points**: Specifies spatial discretisation in z direction.
- **Center (Hub) Height**: Specifies the vertical position of the box center.

Turbine Class

These determine the turbine class as defined in the IEC 61400 design standard [5].

- **Turbine Class**: Specifies the design turbine class.
- **Turbulence Class**: Specifies the design turbulence class.
- **I_ref**: Specifies the turbulence intensity.
- **V_ref**: Specifies the reference velocity.
- **b**: The *b* parameter, used to calculate the turbulence standard deviation in IEC 61400.

Wind Parameters

These parameters specify the parameters and model inputs required for generation of the turbulent velocity field.

- **IEC Standard**: Specifies the version of the IEC standard, used to generate the wind field.
- **Wind Type**: Specifies the wind field type of the generated wind field.
- **Spectral Model**: Specifies the form of the spectral tensor applied to generate the stochastic velocity fluctuations.
- **Wind Profile Type**: Specifies the model used to represent the atmospheric shear layer.
- **Reference Height**: Specifies the reference height of the aforementioned shear layer model.
- **Shear Exponent**: Specifies the shear exponent of the aforementioned shear layer model (if exponential model chosen).
- **Roughness Length**: Specifies the reference height of the aforementioned shear layer model (if logarithmic model chosen).

- **Jet Height**: Specifies the jet height of the aforementioned shear layer model (if jet model chosen).
- **ETMC value**: Specifies the extreme turbulence model c value (if ETM model chosen).
- **Remove TurbSim Files**: If checked, the TurbSim files generated and subsequently read by QBlade, are automatically deleted.
- **Close Console**: If checked, the console which is called to generate the TurbSim file is automatically closed upon completion of TurbSim file generati

Mann Wind Fields

When a new Mann wind field is created, a range of parameters must be specified as shown by the wind field generator dialogue in Fig. 135. After these h automatically passes the information to DTU's Mann 64bit Turbulence Generator. The Mann binary is automatically called by QBlade, and after creation t additional user input is required. **Please note** that the Mann 64bit generator currently is only available for Windows operating systems. The input paramet

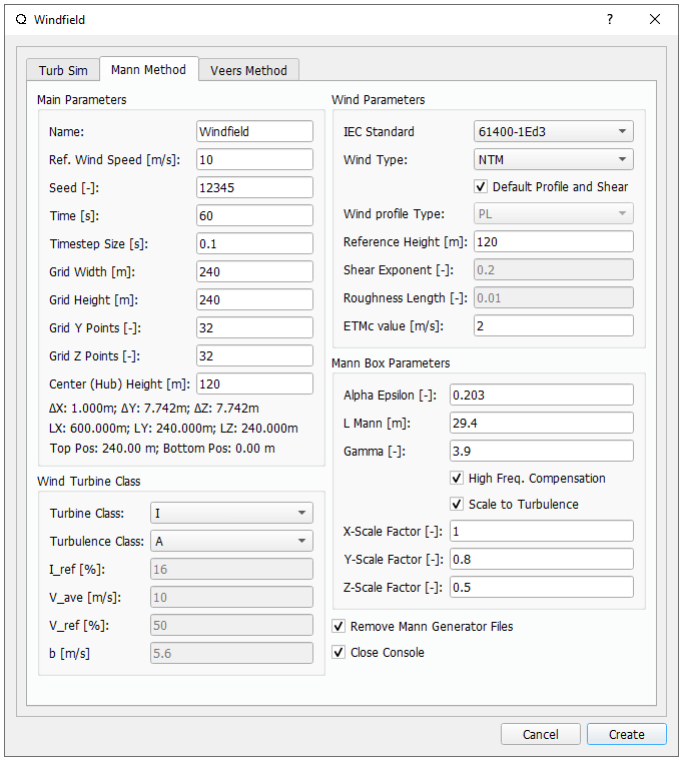


Fig. 135 *Mann wind field dialogue in QBlade.*

Main Parameters

These parameters dictate the spatial dimensions of the generated turbulent wind grid. A turbulent *box* is be generated which is then translated throug below) as is consistent with Taylor's hypothesis for a turbulent flow [4].

- **Name**: The name of the wind field object.
- **Ref. Wind Speed**: The reference wind speed of this wind field.
- **Seed**: The random seed used to generate the wind field.
- **Time**: Determines the length of the generated turbulent box.
- **Timestep**: Specifies the discretisation in free stream (x) direction.
- **Grid Width**: Specifies box size in lateral (y) direction.
- **Grid Height**: Specifies box size in vertical (z) direction.
- **Grid Y Points**: Specifies spatial discretisation in y direction (must be power of 2).
- **Grid Z Points**: Specifies spatial discretisation in z direction (must be power of 2).
- **Center (Hub) Height**: Specifies the vertical position of the box center.

Turbine Class

These determine the turbine class as defined in the IEC 61400 design standard [5].

- **Turbine Class**: Specifies the design turbine class.
- **Turbulence Class**: Specifies the design turbulence class.
- **I_ref**: Specifies the turbulence intensity.
- **V_ref**: Specifies the reference velocity.
- **b**: The *b* parameter, used to calculate the turbulence standard deviation in IEC 61400.

Wind Parameters

These parameters specify the parameters and model inputs required for generation of the turbulent velocity field.

- **IEC Standard**: Specifies the version of the IEC standard, used to generate the wind field.
- **Wind Type**: Specifies the wind field type of the generated wind field.
- **Spectral Model**: Specifies the form of the spectral tensor applied to generate the stochastic velocity fluctuations.
- **Wind Profile Type**: Specifies the model used to represent the atmospheric shear layer.
- **Reference Height**: Specifies the reference height of the aforementioned shear layer model.
- **Shear Exponent**: Specifies the shear exponent of the aforementioned shear layer model (if exponential model chosen).
- **Roughness Length**: Specifies the reference height of the aforementioned shear layer model (if logarithmic model chosen).
- **ETMC value**: Specifies the extreme turbulence model c value (if ETM model chosen).

Mann Box Parameters

- **Alpha Epsilon**: The Mann model $\alpha\epsilon^{\frac{2}{3}}$ parameter.
- **L Mann**: The Mann length scale parameter.
- **Gamma**: The non-dimensional shear distortion parameter.
- **High Freq. Compensation**: If checked: applies the high frequency compensation, so that point velocities represent local anemometer measurement
- **Scale to Turbulence**: If checked: scales the Mann box turbulence to the defined IEC turbulence, multiplied by the parameters described below
- **X-Scale Factor**: Scales the longitudinal turbulence along the x-axis to the IEC turbulence, mutiplied by this value.
- **Y-Scale Factor**: Scales the transversal turbulence along the y-axis to the IEC turbulence, mutiplied by this value.
- **Z-Scale Factor**: Scales the transversal turbulence along the z-axis to the IEC turbulence, mutiplied by this value.

Veers Wind Fields

When a new Veers wind field is created, a range of parameters must be specified as shown by the wind field generator dialogue in Fig. 136. After these ha automatically generates a wind field using the Veers method build into QBlade (see P. Veers [3]). The input parameters are described in detail in the followir

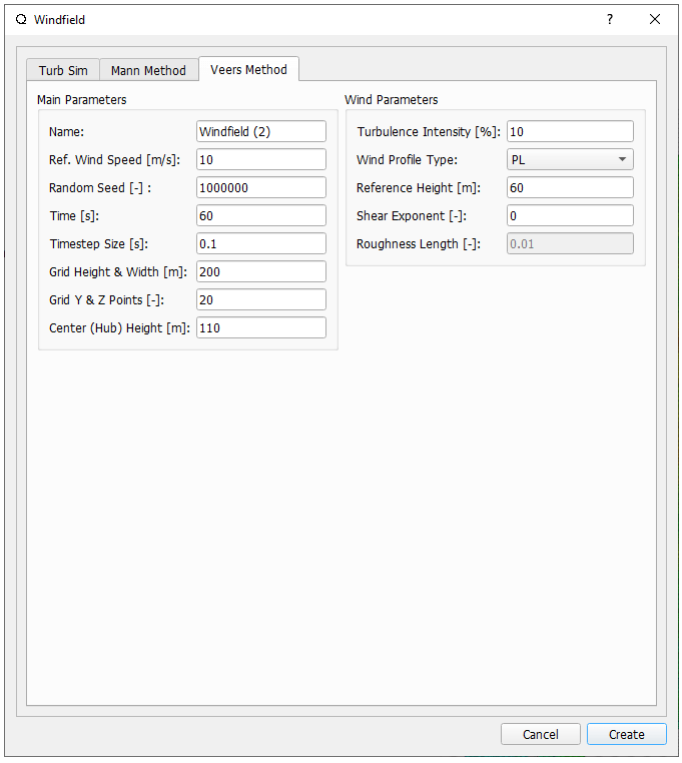


Fig. 136 *Veers wind field dialogue in QBlade.*

Main Parameters

These parameters dictate the spatial dimensions of the generated turbulent wind grid. A turbulent *box* is be generated which is then translated throug below) as is consistent with Taylor's hypothesis for a turbulent flow [4].

- **Name**: The name of the wind field object.
- **Ref. Wind Speed**: The reference wind speed of this wind field.
- **Seed**: The random seed used to generate the wind field.
- **Time**: Determines the length of the generated turbulent box.
- **Timestep**: Specifies the discretisation in free stream (x) direction.
- **Grid Height & Width**: Specifies box size in horizontal (y) and vertical (z) direction.

- **Grid Y & Z Points**: Specifies spatial discretisation in y and z direction
- **Center (Hub) Height**: Specifies the vertical position of the box center.

Wind Parameters

These parameters specify the parameters and model inputs required for generation of the turbulent velocity field.

- **Turbulence Intensity**: The target turbulence intensity.
- **Wind Profile Type**: Specifies the model used to represent the atmospheric shear layer.
- **Reference Height**: Specifies the reference height of the aforementioned shear layer model.
- **Shear Exponent**: Specifies the shear exponent of the aforementioned shear layer model (if exponential model chosen).
- **Roughness Length**: Specifies the reference height of the aforementioned shear layer model (if logarithmic model chosen).

Importing Turbulent Wind Fields

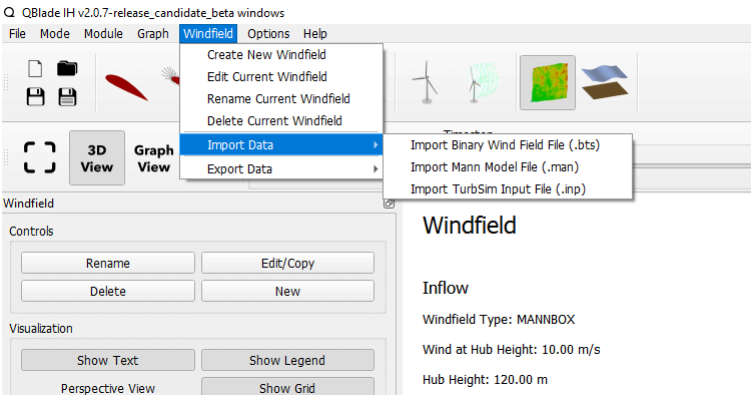


Fig. 137 *Import options in the wind field menu.*

It is also possible to import externally generated three dimensional wind fields into QBlade, see Fig. 137. Wind fields can be imported in three ways:

Binary Wind Field File

A wind field file in binary format (.bts) (see TurbSim Users Guide [1]) can be imported by simply reading the .bts file.

Mann Model File

A Mann box can be imported through the Mann (.man) file format, shown below.

If the parameter **IMPORTBOX** is set to *false*, QBlade will automatically generate a Mann wind box with the parameters specified in the .man file.

If the parameter **IMPORTBOX** is set to *true*, QBlade will search for the (possibly externally generated) files:

- *PREFIX* _u.bin
- *PREFIX* _v.bin
- *PREFIX* _z.bin

An import the velocity components from these binary files directly.

Listing 107 : Exemplary Mann (.man) format file

```
---------------------------------------QBlade Mann Box Definition File-------------------------------------------
Generated with : QBlade EE v2.0.7.4_beta windows
Archive Format: 310024
Time : 13:53:10
Date : 14.07.2024

---------------------------------------Parameters----------------------------------------------------------------
Windfield                             PREFIX            - prefix of the .bin and other files generated
false                                 IMPORTBOX         - false: generate new box from parameters; true: try to find and read .bin files with p

120.000                               HEIGHTBOX         - height of the mann box center in [m]
630.000                               XDIM_BOX          - length of the mann box in [m]
240.000                               YDIM_BOX          - width of the mann box in [m]
240.000                               ZDIM_BOX          - height of the mann box in [m]
631                                   NX_BOX            - number of points along length, must be power of 2 [-]
32                                    NY_BOX            - number of points along width, must be power of 2 [-]
```

```
32                                      NZ_BOX            - number of points along width, must be power of 2 [-]

120.000                                 REFHEIGHT         - reference height for the BL profile in [m]
0                                       PROFILETYPE       - BL profile type: 0- power law; 1 - logarithmic
0.200                                   PROFILEPARAM      - power law exponent or roughness length

0.0660                                  ALPHA_EPSILON     - Mann alpha-epsilon parameter
29.4000                                 L_MANN            - Mann length scale [m]
3.9000                                  GAMMA             - Mann gamma parameter
12345                                   SEED              - turbulent seed

10.000                                  WINDSPEED         - hub-height average wind speed

1                                       IEC_STANDARD      - IEC standard 61400- (1, 2 or 3)
NTM                                     IEC_WINDTYPE      - IEC wind type (NTM, ETM, EWM1, EWM50 or ADDTURB
16.000                                  IEC_IREF          - IEC I_ref value [-]
10.000                                  IEC_VAVE          - IEC V_ave value [m/s]
50.000                                  IEC_VREF          - IEC V_ref value [m/s]
5.600                                   IEC_B             - IEC b value (or a in 61400-2) [m/s]
2.000                                   IEC_ETMC          - IEC ETM c value [m/s]

true                                    TURB_SCALING      - enable turbulent scaling: 0 - OFF; 1 - ON
true                                    HF_CORRECTION     - enable high frequency correction: 0 - OFF; 1 - ON

1.000                                   X_FACTOR          - scaling factor for x-variance
0.800                                   Y_FACTOR          - scaling factor for y-variance
0.500                                   Z_FACTOR          - scaling factor for z-variance
```

TurbSim Input File

A TurbSim input file may be directly opened in QBlade. The input file (.inp) will then automatically be communicated to the TurbSim binary and the corresp

Listing 108 : Exemplary TurbSim Input (.ipt) file

```
!TurbSim Input File. Valid for TurbSim from OpenFAST v2.4.0. Generated with QBlade QBlade IH v2.0.7-release_candidate_beta windows on 15.05.2024 at

---------Runtime Options-----------------------------------
False               Echo            - Echo input data to <RootName>.ech (flag)
12345               RandSeed1       - First random seed  (-2147483648 to 2147483647)
RANLUX              RandSeed2       - Second random seed (-2147483648 to 2147483647) for intrinsic pRNG, or an alternative pRNG: "RanLux" or "RNSNL
False               WrBHHTP         - Output hub-height turbulence parameters in binary form?  (Generates RootName.bin)
False               WrFHHTP         - Output hub-height turbulence parameters in formatted form?  (Generates RootName.dat)
False               WrADHH          - Output hub-height time-series data in AeroDyn form?  (Generates RootName.hh)
True                WrADFF          - Output full-field time-series data in TurbSim/AeroDyn form? (Generates Rootname.bts)
False               WrBLFF          - Output full-field time-series data in BLADED/AeroDyn form?  (Generates RootName.wnd)
False               WrADTWR         - Output tower time-series data? (Generates RootName.twr)
False               WrFMTFF         - Output full-field time-series data in formatted (readable) form?  (Generates RootName.u, RootName.v, RootName
False               WrACT           - Output coherent turbulence time steps in AeroDyn form? (Generates RootName.cts)
True                Clockwise       - Clockwise rotation looking downwind? (used only for full-field binary files - not necessary for AeroDyn)
0                   ScaleIEC        - Scale IEC turbulence models to exact target standard deviation? [0=no additional scaling; 1=use hub scale uni

--------Turbine/Model Specifications-----------------------
24                  NumGrid_Z       - Vertical grid-point matrix dimension
24                  NumGrid_Y       - Horizontal grid-point matrix dimension
0.1000              TimeStep        - Time step [seconds]
63.0000             AnalysisTime    - Length of analysis time series [seconds] (program will add time if necessary: AnalysisTime = MAX(AnalysisTime
63.0000             usableTimeLabel      - Usable length of output time series [seconds] (program will add GridWidth/MeanHHWS seconds)
120.0001            HubHt           - Hub height [m] (should be > 0.5*GridHeight)
240.00              GridHeight      - Grid height [m]
240.00              GridWidth       - Grid width [m] (should be >= 2*(RotorRadius+ShaftLength))
0.0                 VFlowAng        - Vertical mean flow (uptilt) angle [degrees]
0.0                 HFlowAng        - Horizontal mean flow (skew) angle [degrees]

--------Meteorological Boundary Conditions-------------------
"IECKAI"            TurbModel       - Turbulence model ("IECKAI"=Kaimal, "IECVKM"=von Karman, "GP_LLJ", "NWTCUP", "SMOOTH", "WF_UPW", "WF_07D", "WF
"unused"            UserFile        - Name secondary input file for user-defined spectra or time series inputs
"1-ED3"             IECstandard     - Number of IEC 61400-x standard (x=1,2, or 3 with optional 61400-1 edition number (i.e. "1-Ed2") )
"A"                 IECturbc        - IEC turbulence characteristic ("A", "B", "C" or the turbulence intensity in percent) ("KHTEST" option with NW
"NTM"               IEC_WindType    - IEC turbulence type ("NTM"=normal, "xETM"=extreme turbulence, "xEWM1"=extreme 1-year wind, "xEWM50"=extreme 5
2.00                ETMc            - IEC Extreme Turbulence Model "c" parameter [m/s]
default             ProfileType     - Wind profile type ("JET";"LOG"=logarithmic;"PL"=power law;"H2L"=Log law for TIDAL spectral model;"IEC"=PL on
"unused"            ProfileFile -     Name of the file that contains user-defined input profiles
120.00              RefHt           - Height of the reference wind speed [m]
10.00               URef            - Mean (total) wind speed at the reference height [m/s] (or "default" for JET wind profile)
default             ZJetMax         - Jet height [m] (used only for JET wind profile, valid 70-490 m)
default             PLExp           - Power law exponent [-] (or "default")
default             Z0              - Surface roughness length [m] (or "default")

--------Non-IEC Meteorological Boundary Conditions------------
default             Latitude        - Site latitude [degrees] (or "default")
0.05                RICH_NO         - Gradient Richardson number
default             UStar           - Friction or shear velocity [m/s] (or "default")
default             ZI              - Mixing layer depth [m] (or "default")
default             PC_UW           - Hub mean u'w' Reynolds stress (or "default")
default             PC_UV           - Hub mean u'v' Reynolds stress (or "default")
default             PC_VW           - Hub mean v'w' Reynolds stress (or "default")
```

```
--------Spatial Coherence Parameters----------------------------
default             SCMod1          - u-component coherence model ("GENERAL","IEC","API","NONE", or "default")
default             SCMod2          - v-component coherence model ("GENERAL","IEC","API","NONE", or "default")
default             SCMod3          - w-component coherence model ("GENERAL","IEC","API","NONE", or "default")
default             InCDec1         - u-component coherence parameters [-, m^-1] ("a b" in quotes or "default")
default             InCDec2         - v-component coherence parameters [-, m^-1] ("a b" in quotes or "default")
default             InCDec3         - w-component coherence parameters [-, m^-1] ("a b" in quotes or "default")
default             CohExp          - Coherence exponent for general model [-] (or "default")

--------Coherent Turbulence Scaling Parameters-------------------
"path/to/coh_events/eventdata"  CTEventPath     - Name of the path where event data files are located
"Random"            CTEventFile     - Type of event files ("LES", "DNS", or "RANDOM")
true                Randomize       - Randomize the disturbance scale and locations? (true/false)
 1.0                DistScl         - Disturbance scale (ratio of wave height to rotor disk). (Ignored when Randomize = true.)
 0.5                CTLy            - Fractional location of tower centerline from right (looking downwind) to left side of the dataset. (Ignored w
 0.5                CTLz            - Fractional location of hub height from the bottom of the dataset. (Ignored when Randomize = true.)
30.0                CTStartTime     - Minimum start time for coherent structures in RootName.cts [seconds]

================================================
NOTE: Do not add or remove any lines in this file!
================================================
```

Uniform Wind Field

A uniform wind field is specified directly within the *Wind Input Type* of the turbine simulation dialogue, shown in Fig. 138 (see Simulation Module Overvie horizontal inflow angle and directional shear are defined here. In the case that the atmospheric boundary layer is to be modelled, this can be selected wit shear parameters can then by specified (see Wind).

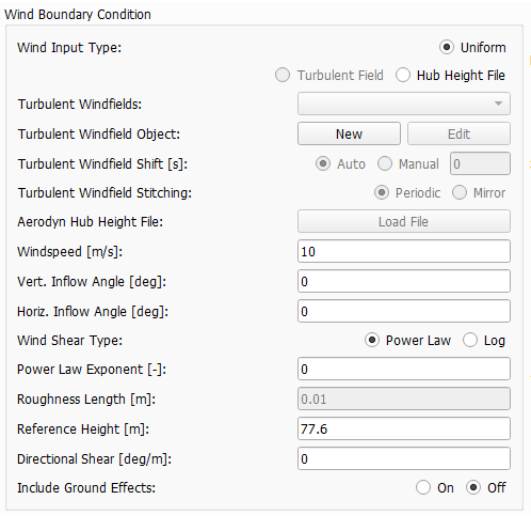


Fig. 138 *Specification of a uniform wind field within the turbine simulation dialogue.*

Hub Height File

The user has more modelling freedom when a hub-height wind file is used. This type of file can either be created manually or by using the IEC wind tool [6] the hub height as a function of time. QBlade interpolates the time between the starting time of the file and the point where the predefined wind velocity specified simulation time exceeds the ending time in the hub-height file, QBlade will create a constant wind field with the parameters from the last entry An exemplary hubheight input file that described an extreme operating gust (EOG) at 20m/s is shown below:

Listing 109 : Exemplary Hub Height Format file

```
Time    Wind    Horiz.  Vert.   LinH.   Vert.   LinV.   Gust
        Speed   Dir     Speed   Shear   Shear   Shear   Speed
0.000   20.000  0.000   0.000   0.000   0.200   0.000   0.000
60.000  20.000  0.000   0.000   0.000   0.200   0.000   0.000
60.100  20.000  0.000   0.000   0.000   0.200   0.000   -0.000
60.200  20.000  0.000   0.000   0.000   0.200   0.000   -0.004
60.300  20.000  0.000   0.000   0.000   0.200   0.000   -0.012
60.400  20.000  0.000   0.000   0.000   0.200   0.000   -0.028
60.500  20.000  0.000   0.000   0.000   0.200   0.000   -0.054
60.600  20.000  0.000   0.000   0.000   0.200   0.000   -0.092
60.700  20.000  0.000   0.000   0.000   0.200   0.000   -0.144
60.800  20.000  0.000   0.000   0.000   0.200   0.000   -0.209
60.900  20.000  0.000   0.000   0.000   0.200   0.000   -0.289
61.000  20.000  0.000   0.000   0.000   0.200   0.000   -0.384
61.100  20.000  0.000   0.000   0.000   0.200   0.000   -0.493
61.200  20.000  0.000   0.000   0.000   0.200   0.000   -0.614
61.300  20.000  0.000   0.000   0.000   0.200   0.000   -0.747
61.400  20.000  0.000   0.000   0.000   0.200   0.000   -0.889
61.500  20.000  0.000   0.000   0.000   0.200   0.000   -1.037
```

```
61.600  20.000  0.000   0.000   0.000   0.200   0.000   -1.188
61.700  20.000  0.000   0.000   0.000   0.200   0.000   -1.338
61.800  20.000  0.000   0.000   0.000   0.200   0.000   -1.485
61.900  20.000  0.000   0.000   0.000   0.200   0.000   -1.622
62.000  20.000  0.000   0.000   0.000   0.200   0.000   -1.748
62.100  20.000  0.000   0.000   0.000   0.200   0.000   -1.856
62.200  20.000  0.000   0.000   0.000   0.200   0.000   -1.944
62.300  20.000  0.000   0.000   0.000   0.200   0.000   -2.007
62.400  20.000  0.000   0.000   0.000   0.200   0.000   -2.041
62.500  20.000  0.000   0.000   0.000   0.200   0.000   -2.043
62.600  20.000  0.000   0.000   0.000   0.200   0.000   -2.011
62.700  20.000  0.000   0.000   0.000   0.200   0.000   -1.942
62.800  20.000  0.000   0.000   0.000   0.200   0.000   -1.834
62.900  20.000  0.000   0.000   0.000   0.200   0.000   -1.686
63.000  20.000  0.000   0.000   0.000   0.200   0.000   -1.498
63.100  20.000  0.000   0.000   0.000   0.200   0.000   -1.271
63.200  20.000  0.000   0.000   0.000   0.200   0.000   -1.005
63.300  20.000  0.000   0.000   0.000   0.200   0.000   -0.703
63.400  20.000  0.000   0.000   0.000   0.200   0.000   -0.366
63.500  20.000  0.000   0.000   0.000   0.200   0.000   0.000
63.600  20.000  0.000   0.000   0.000   0.200   0.000   0.393
63.700  20.000  0.000   0.000   0.000   0.200   0.000   0.807
63.800  20.000  0.000   0.000   0.000   0.200   0.000   1.237
63.900  20.000  0.000   0.000   0.000   0.200   0.000   1.678
64.000  20.000  0.000   0.000   0.000   0.200   0.000   2.124
64.100  20.000  0.000   0.000   0.000   0.200   0.000   2.568
64.200  20.000  0.000   0.000   0.000   0.200   0.000   3.003
64.300  20.000  0.000   0.000   0.000   0.200   0.000   3.425
64.400  20.000  0.000   0.000   0.000   0.200   0.000   3.825
64.500  20.000  0.000   0.000   0.000   0.200   0.000   4.198
64.600  20.000  0.000   0.000   0.000   0.200   0.000   4.539
64.700  20.000  0.000   0.000   0.000   0.200   0.000   4.841
64.800  20.000  0.000   0.000   0.000   0.200   0.000   5.101
64.900  20.000  0.000   0.000   0.000   0.200   0.000   5.314
65.000  20.000  0.000   0.000   0.000   0.200   0.000   5.477
65.100  20.000  0.000   0.000   0.000   0.200   0.000   5.587
65.200  20.000  0.000   0.000   0.000   0.200   0.000   5.642
65.300  20.000  0.000   0.000   0.000   0.200   0.000   5.642
65.400  20.000  0.000   0.000   0.000   0.200   0.000   5.587
65.500  20.000  0.000   0.000   0.000   0.200   0.000   5.477
65.600  20.000  0.000   0.000   0.000   0.200   0.000   5.314
65.700  20.000  0.000   0.000   0.000   0.200   0.000   5.101
65.800  20.000  0.000   0.000   0.000   0.200   0.000   4.841
65.900  20.000  0.000   0.000   0.000   0.200   0.000   4.539
66.000  20.000  0.000   0.000   0.000   0.200   0.000   4.198
66.100  20.000  0.000   0.000   0.000   0.200   0.000   3.825
66.200  20.000  0.000   0.000   0.000   0.200   0.000   3.425
66.300  20.000  0.000   0.000   0.000   0.200   0.000   3.003
66.400  20.000  0.000   0.000   0.000   0.200   0.000   2.568
66.500  20.000  0.000   0.000   0.000   0.200   0.000   2.124
66.600  20.000  0.000   0.000   0.000   0.200   0.000   1.678
66.700  20.000  0.000   0.000   0.000   0.200   0.000   1.237
66.800  20.000  0.000   0.000   0.000   0.200   0.000   0.807
66.900  20.000  0.000   0.000   0.000   0.200   0.000   0.393
67.000  20.000  0.000   0.000   0.000   0.200   0.000   0.000
67.100  20.000  0.000   0.000   0.000   0.200   0.000   -0.366
67.200  20.000  0.000   0.000   0.000   0.200   0.000   -0.703
67.300  20.000  0.000   0.000   0.000   0.200   0.000   -1.005
67.400  20.000  0.000   0.000   0.000   0.200   0.000   -1.271
67.500  20.000  0.000   0.000   0.000   0.200   0.000   -1.498
67.600  20.000  0.000   0.000   0.000   0.200   0.000   -1.686
67.700  20.000  0.000   0.000   0.000   0.200   0.000   -1.834
67.800  20.000  0.000   0.000   0.000   0.200   0.000   -1.942
67.900  20.000  0.000   0.000   0.000   0.200   0.000   -2.011
68.000  20.000  0.000   0.000   0.000   0.200   0.000   -2.043
68.100  20.000  0.000   0.000   0.000   0.200   0.000   -2.041
68.200  20.000  0.000   0.000   0.000   0.200   0.000   -2.007
68.300  20.000  0.000   0.000   0.000   0.200   0.000   -1.944
68.400  20.000  0.000   0.000   0.000   0.200   0.000   -1.856
68.500  20.000  0.000   0.000   0.000   0.200   0.000   -1.748
68.600  20.000  0.000   0.000   0.000   0.200   0.000   -1.622
68.700  20.000  0.000   0.000   0.000   0.200   0.000   -1.485
68.800  20.000  0.000   0.000   0.000   0.200   0.000   -1.338
68.900  20.000  0.000   0.000   0.000   0.200   0.000   -1.188
69.000  20.000  0.000   0.000   0.000   0.200   0.000   -1.037
69.100  20.000  0.000   0.000   0.000   0.200   0.000   -0.889
69.200  20.000  0.000   0.000   0.000   0.200   0.000   -0.747
69.300  20.000  0.000   0.000   0.000   0.200   0.000   -0.614
69.400  20.000  0.000   0.000   0.000   0.200   0.000   -0.493
69.500  20.000  0.000   0.000   0.000   0.200   0.000   -0.384
69.600  20.000  0.000   0.000   0.000   0.200   0.000   -0.289
69.700  20.000  0.000   0.000   0.000   0.200   0.000   -0.209
69.800  20.000  0.000   0.000   0.000   0.200   0.000   -0.144
69.900  20.000  0.000   0.000   0.000   0.200   0.000   -0.092
70.000  20.000  0.000   0.000   0.000   0.200   0.000   -0.054
70.100  20.000  0.000   0.000   0.000   0.200   0.000   -0.028
70.200  20.000  0.000   0.000   0.000   0.200   0.000   -0.012
70.300  20.000  0.000   0.000   0.000   0.200   0.000   -0.004
```

```
70.400  20.000  0.000   0.000   0.000   0.200   0.000   -0.000
70.500  20.000  0.000   0.000   0.000   0.200   0.000   0.000
```


[1] (1, 2, 3) B.J. Jonkman and L. Kilcher. *TurbSim User's Guide Verson 1.06.00*. NREL, 2012. An optional note.

[2] J. Mann. Wind field simulation. *Probabilistic Engineering Mechanics*, 13:269–282, 1998.

[3] (1, 2) Paul S. Veers. Three dimensional wind simulation. 1988.

[4] (1, 2, 3) G.K. Batchelor. *The Theory of Homogeneous Turbulence*. Cambridge University Press, 1953. ISBN 0521041171.

[5] (1, 2) IEC61400-1 Standard. IEC 61400-1:2019 Wind energy generation systems - Part 1: Design requirements. Standard, International Electrotechnical Commiss

[6] NREL. IECWind. https://www.nrel.gov/wind/nwtc/iecwind.html, 2022. [Online; accessed 2022-06-13].

Wave Generator Overview

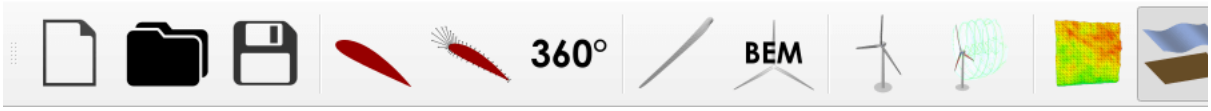


Fig. 139 *The wave generation module in QBlade's main tool bar.*

If an offshore simulation where the consideration of wave excitation is being carried out, it is necessary to provide the information about the sea sate in tl
The field may either consist of a single wave train (regular wave) or multiple, superpositioned regular waves - (irregular waves). Both types may be generat
third possibility is the definition of a prescribed sea state, allowing the user to import externally generated wave fields. The three functionalities are descr
below. The underlying theory implemented in QBlade is described in the Waves section of the theory guide.

Any wave field generated in QBlade requires that a new wave is created by selecting this option in the *Controls* box. This opens the *Linear Waves* dialogue
generation options are displayed. In the *Main Seastate Parameters* box, the wave train is defined (amplitude and frequency). The *Equal Energy Frequency Dis*
user to tune the discretization parameters of the energy spectrum. Finally, the *Equal Energy Directional Discretization* box lets the user define directional pr

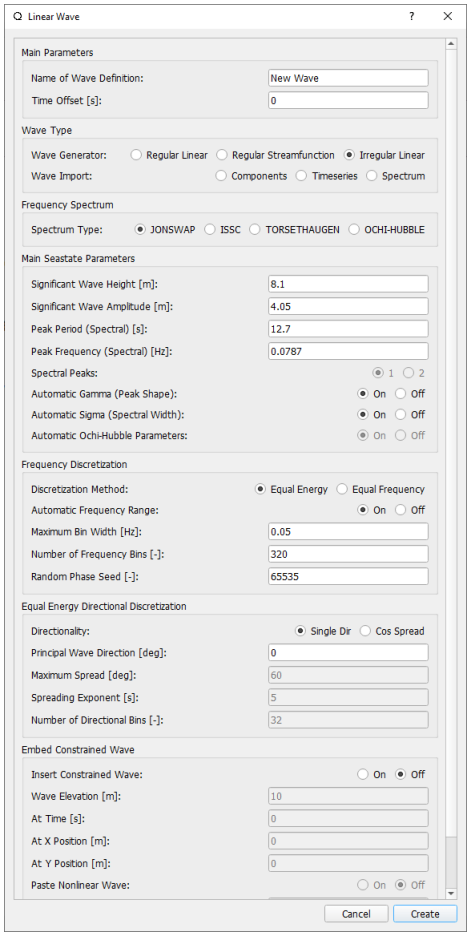


Fig. 140 *The wave generator dialog in QBlade.*

Regular Linear Wave

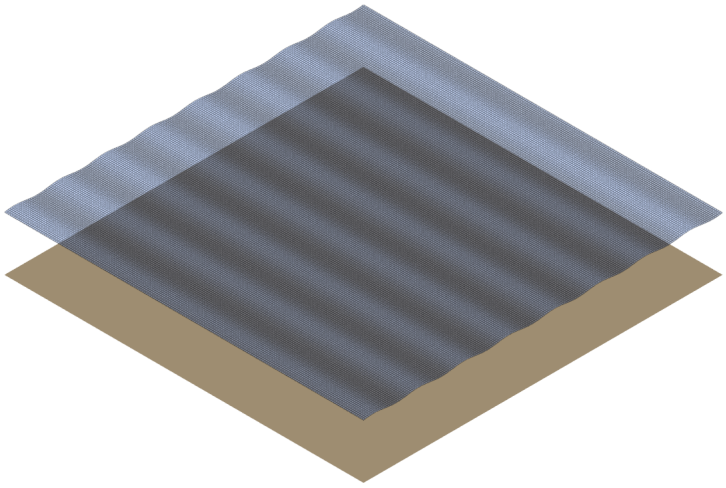


Fig. 141 *Visualization of a regular wave.*

To generate a regular wave, the wave type *Regular Linear* has to be chosen in the *Linear Wave* dialogue. The user now has the option to characterize the si remaining available inputs. These parameters define the shape and direction of an Airy wave (see Linear Wave Theory).

Main Parameters

- **Time Offset**: Time shift of the generated wave signal
- **Significant Wave Height**: Height of wave train to be generated (directly linked to amplitude)
- **Significant Wave Amplitude**: Amplitude of the wave (directly linked to wave height)
- **Peak Period**: Period of the wave (directly linked to wave frequency)
- **Peak Frequency**: Frequency of the wave (directly linked to the wave period)

Equal Energy Directional Discretization

- **Principal Wave Direction**: Incoming wave direction

Regular Nonlinear Wave

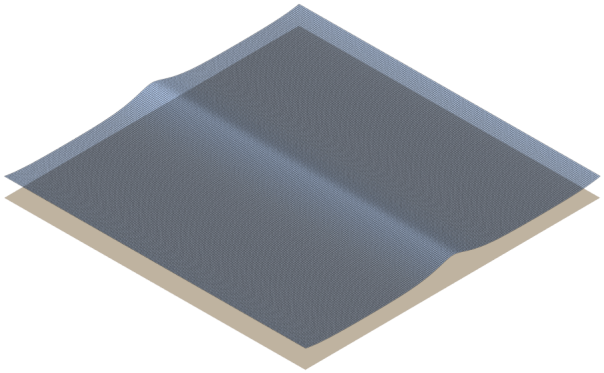


Fig. 142 *Visualization of a regular nonlinear wave.*

To generate a regular nonlinear wave, the wave type *Regular Nonlinear* has to be chosen in the *Linear Wave* dialogue. The user now has the option to chara with the remaining available inputs. These parameters define the shape and direction of a nonlinear Streamfunction Wave. The Streamfunction waves are using the CN-Stream library from LHEAA (see the work of G. Ducrozet et al. [1])

Main Parameters

- **Time Offset**: Time shift of the generated wave signal
- **Significant Wave Height**: Height of wave train to be generated (directly linked to amplitude)
- **Significant Wave Amplitude**: Amplitude of the wave (directly linked to wave height)
- **Peak Period**: Period of the wave (directly linked to wave frequency)
- **Peak Frequency**: Frequency of the wave (directly linked to the wave period)

Equal Energy Directional Discretization

- **Principal Wave Direction**: Incoming wave direction

Please note that nonlinear waves cannot be used to obtain hydrodynamic forces from Linear Potential Flow Theory , but are only suited to model Morisor hydrodynamic forces.

Irregular Linear Wave

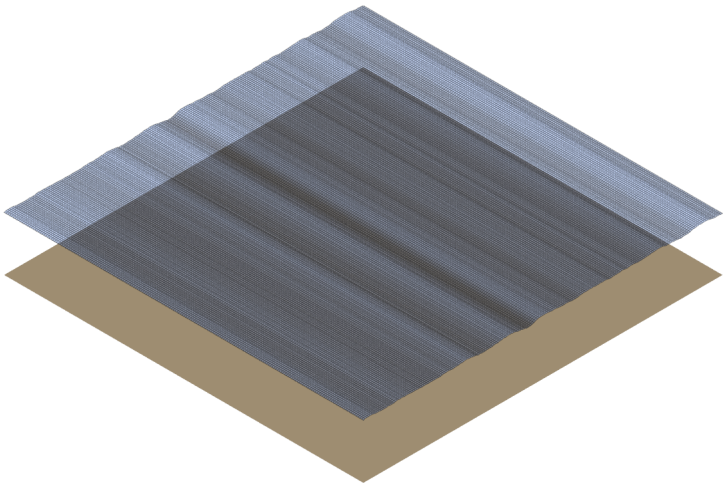


Fig. 143 *Visualization of an irregular wave.*

To generate an irregular wave, the wave type *Irregular Linear* has to be chosen. The user is now given the option to characterize the wave with the remaini addition to the wave train characterization discussed above, spectra discretization options can be specified.

Main Parameters

- **Time Offset**: Time shift of the generated wave signal
- **Significant Wave Height**: Wave height defining shape of the wave spectrum (directly linked to amplitude)
- **Significant Wave Amplitude**: Wave amplitude defining shape of the wave spectrum (directly linked to height)
- **Peak Period**: Peak period of the wave spectrum (directly linked to wave frequency)
- **Peak Frequency**: Peak frequency of the wave spectrum (directly linked to the wave period)
- **Automatic Gamma**: Automatic or manual definition of peak shape factor of the spectrum
- **Automatic Sigma**: Automatic or manual definition of the spectral width parameter

Frequency Discretization

- **Discretization Method**: The options are equal energy or equal frequency discretization of the wave spectrum
- **Maximum Bin Width**: Maximum frequency range of the spectrum discretization.
- **Number of Frequency Bins**: Resolution of frequency discretization of the energy spectrum.
- **Random Phase Seed**: The random seed assigning the wave component phase data.

Equal Energy Directional Discretization

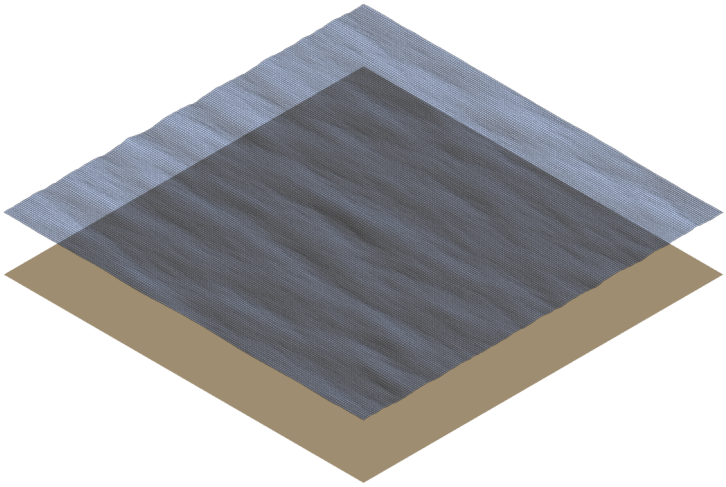


Fig. 144 *Visualization of an irregular multi-directional wave.*

Either a unidirectional irregular wave (Single Dir) or multidirectional wave (Cos Spread) can be created

- **Principal Wave Direction**: Definition of the wave direction (unidirectional spectrum) or of the principal direction of the cosine spectrum.
- **Maximum Spread**: Definition of the width of the cosine spectrum.
- **Spreading Exponent**: Shape defining parameter for the directional spectrum
- **Number of Directional Bins**: Resolution of angular discretization of the directional spectrum.

Embedded Constrained Wave

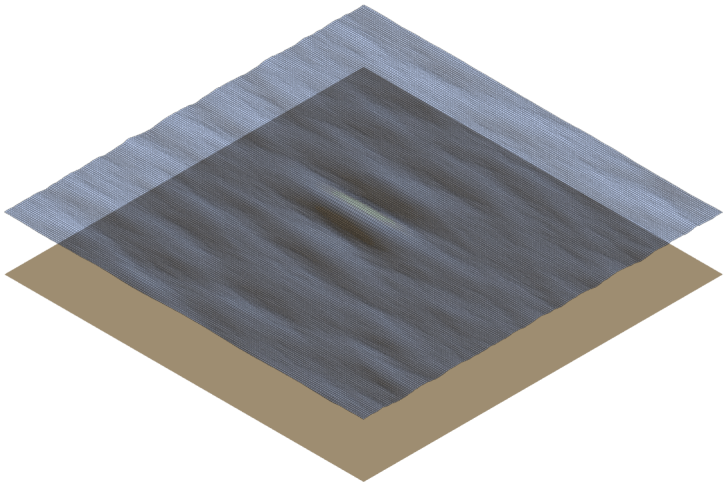


Fig. 145 *A 30m constrained wave embedded in an irregular multi-directional wavefield.*

QBlade also allows to embed a constrained wave into an irregular wavefield. This process is based on the *NewWave* method of Taylor [2] and follows the im
laid out in L. Wang, J. Jonkman, G. Hayman, A. Platt, B. Jonkman, A. Robertson[3]. The main use of this functionality is to reduce the required simulation tim
occurs. The extreme wave that is embedded hereby is conditioned on the underlying wave spectrum and is indistinguishable from a naturally occurring ex

It is highly suggested to use an *Equal Frequency* discretization, with sufficient wave trains when embedding a constrained wave.

- **Wave Elevation**: The elevation of the embedded wave
- **At Time**: The time at which the extreme wave occurs
- **At X Position**: The X position at which the extreme wave occurs
- **At Y Position**: The Y position at which the extreme wave occurs

Furthermore, it is possible to *copy-paste* a nonlinear regular wave over the constrained wave. This process is carried out in a similar way as described in th
[4]. It is only possible to copy-paste a nonlinear wave over a constrained wave in a unidirectional wavefield. The user has to specify the following eter
nonlinear wave:

- **Nonlinear Wave Height**: The wave height of the pasted nonlinear wave
- **Nonlinear Wave Period**: The period of the nonlinear wave

Please note that the *Nonlinear Wave Height* parameter is not the same as the *Wave Elevation* parameter that was specified for the constrained wave. The a the pasted nonlinear wave also depends on other factors, such as the water depth.

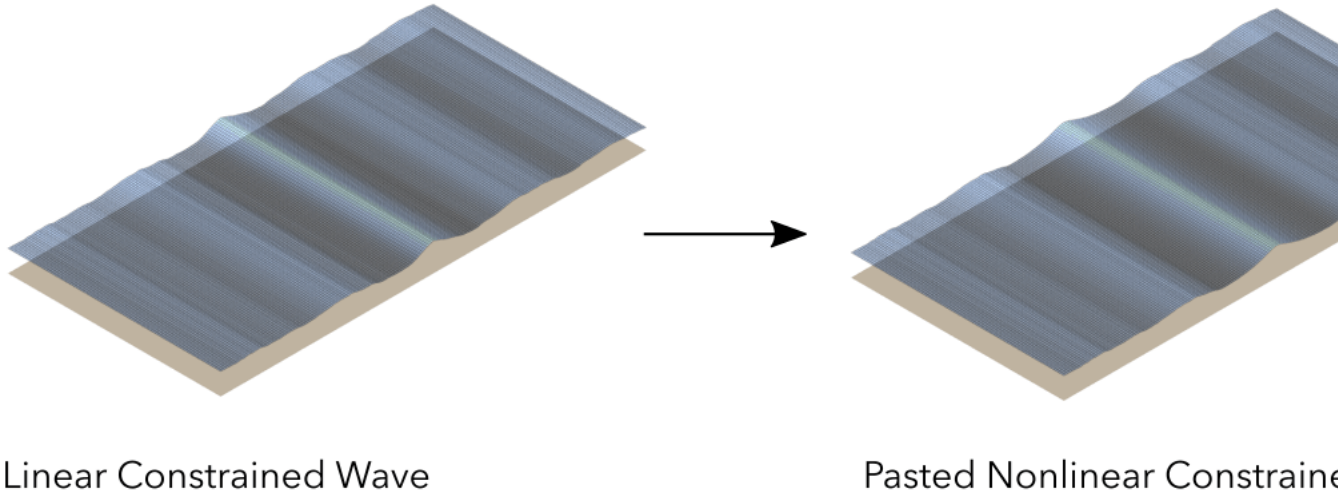


Fig. 146 *An example of a nonlinear regular wave, pasted over a linear constrained wave in an irregular unidirectional wavefield.*

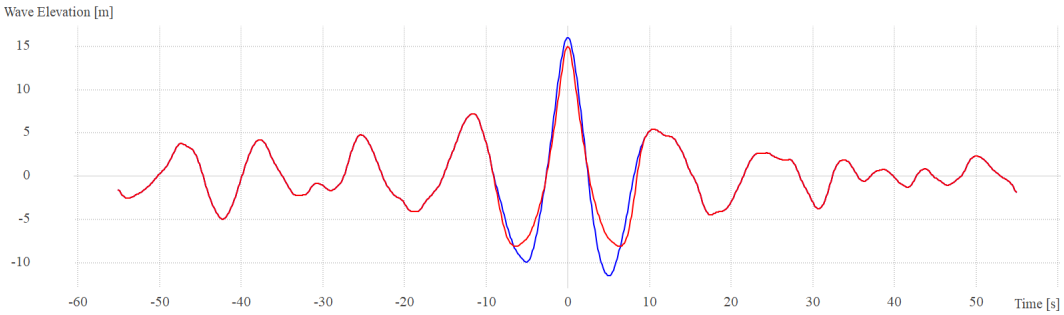


Fig. 147 *Timetrace of wave elevation of the pasted nonlinear regular wave (in red), pasted over a linear constrained wave (in blue).*

Nonlinear wave models in offshore wind turbine simulations offer enhanced accuracy by more realistically representing extreme sea states and complex w technique of pasting nonlinear waves into linear seas enables precise analysis of specific severe wave conditions without requiring extensive simulations t naturally. This leads to more accurate predictions of structural loads, crucial for ensuring the safety and structural integrity of turbines, and facilitates mor and risk assessment under challenging conditions.

Please note that nonlinear waves cannot be used to obtain hydrodynamic forces from Linear Potential Flow Theory , but are only suited to model Morisor hydrodynamic forces.

Import Components

By selecting this option the user can import a wave using wave component data. when this option is selected a button appears *Import Components File* wh import a `.txt` file containing the wave component information. This file must contains frequency [Hz], amplitude [m], phase [deg] and direction [deg] infc in four columns. This data represents the frequency domain information of the wave. This is inverse Fourier-transformed in order to specify a time-series calculated, the button *View Wave File* appears allowing the user to visually check the imported data.

Import Timeseries

By selecting this option the user can import a wave using a time series of the wave height. A discrete Fourier transform (DFT) is applied to the timeseries data in the frequency domain. An inverse Fourier transform (IFT) is then applied to the Fourier coefficient in order to recreate the time-series da et c specified for the DFT which gives the user some control of the wave components that are generated by the DFT. These parameters include:

- **Low Cut-Off Frequency**: The minimum frequency considered in the DFT, below which wave components are discarded (approximately low-pass filteri
- **High Cut-Off Frequency**: The maximum frequency considered in the DFT, above which wave components are discarded (approximately high-pass filte
- **Signal Sampling Rate**: The frequency with which data from the time series is sampled before the DFT is performed. This allows the user to reduce the components that will be generated by the DFT.
- **Amplitude Threshold**: The minimum wave component amplitude allowed after the DFT is performed. This allows the user to filter out wave componer amplitude and thereby helps to reduce the number of generated wave components.

Import and Export Functionality

QBlade allows the user to import and export wave fields either in the four column format described in Import Components or in a `.lwa` format. The `.lwa` parameters necessary to define the time and frequency domain descriptions of a wave. This functionality can be found in the menu toolbar below the *Wa*

Wave Definition ASCII File

An exemplary `.lwa` file is shown below:

Listing 110 : A wave exported in ASCII format

```
----------------------------------------QBlade Wave Definition File-------------------------------------------------
Generated with : QBlade IH v2.0.7-release_candidate_beta windows
Archive Format: 310023
Time : 15:05:06
Date : 15.05.2024

----------------------------------------Object Name-----------------------------------------------------------------
Pasted-Nonlinear-Wave                   OBJECTNAME          - the name of the linear wave definition object

----------------------------------------Main Parameters-------------------------------------------------------------
0.000                                   TIMEOFFSET          - the time offset from t=0s [s]
3                                       WAVETYPE            - wave type: 0=TIMESERIES, 1=COMPONENT, 2=SINGLE, 3=JONSWAP, 4=ISSC, 5=TORSETHAUGEN, 6=
8.100                                   SIGHEIGHT           - the significant wave height (Hs) [m]
12.700                                  PEAKPERIOD          - the peak period (Tp) [s]
true                                    AUTOGAMMA           - use gamma according to IEC (bool): 0 = OFF, 1 = ON (JONSWAP & TORSE only) [bool]
1.000                                   GAMMA               - custom gamma (JONSWAP & TORSE only)
true                                    AUTOSIGMA           - use sigmas according to IEC (JONSWAP & TORSE only) [bool]
0.070                                   SIGMA1              - sigma1 (JONSWAP & TORSE only)
0.090                                   SIGMA2              - sigma1 (JONSWAP & TORSE only)
0                                       DOUBLEPEAK          - if true a double peak TORSETHAUGEN spectrum will be created, if false only a single p
true                                    AUTOORCHI           - automatic OCHI-HUBBLE parameters from significant wave height (OCHI only) [bool]
0.077                                   MODFREQ1            - modal frequency 1, must be "< modalfreq1 * 0.5" (OCHI only)
0.133                                   MODFREQ2            - modal frequency 2, should be larger than 0.096 (OCHI only)
6.804                                   SIGHEIGHT1          - significant height 1, should be larger than height 2 (OCHI only)
4.374                                   SIGHEIGHT2          - significant height 2 (OCHI only)
3.000                                   LAMBDA1             - peak shape 1 (OCHI only)
0.932                                   LAMBDA2             - peak shape 2 (OCHI only)

----------------------------------------Frequency Discretization ----------------------------------------------------
1                                       DISCTYPE            - frequency discretization type: 0 = equal energy; 1 = equal frequency
true                                    AUTOFREQ            - use automatic frequency range (f_in = 0.5*f_p, f_out = 10*f_p) [bool]
0.039                                   FCUTIN              - cut-in frequency
0.787                                   FCUTOUT             - cut-out frequency
0.050                                   MAXFBIN             - maximum freq. bin width [Hz]
3020                                    NUMFREQ             - the number of frequency bins
65535                                   RANDSEED            - the seed for the random phase generator range [0-65535]

----------------------------------------Directional Discretization (Equal Energy)-----------------------------------
0                                       DIRTYPE             - the directional type, 0 = UNIDIRECTIONAL, 1 = COSINESPREAD
0.000                                   DIRMEAN             - mean wave direction [deg]
60.000                                  DIRMAX              - directional spread [deg]
5.000                                   SPREADEXP           - the spreading exponent
32                                      NUMDIR              - the number of directional bins

----------------------------------------Embedded Constrained Wave ---------------------------------------------------
true                                    EMBEDWAVE           - add a constrained wave [bool]
16.00                                   EMBEDELEV           - the wave elevation of the embedded wave [m]
0.00                                    EMBEDTIME           - the time at which the embedded wave occurs [s]
0.00                                    EMBEDXPOS           - the x-position at which the embedded wave occurs [m]
0.00                                    EMBEDYPOS           - the y-position at which the embedded wave occurs [m]
true                                    PASTESTREAM         - paste a streamfunction wave over the embedded linear wave [bool]
23.00                                   SIGHEIGHTSTREAM     - the significant height of the streamfunction wave [m]
12.70                                   PERIODSTREAM        - the period of the stremfunction wave [s]
```

Merged Waves

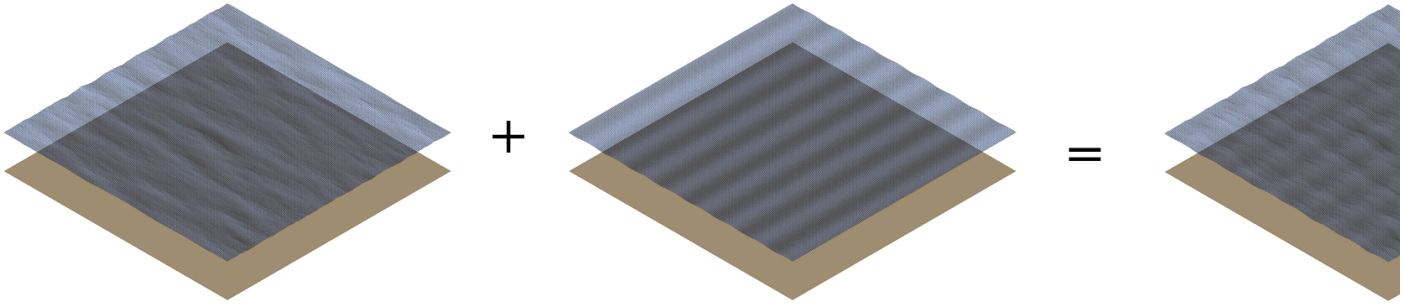


Fig. 148 *Visualization of a new wave merged from an irregular and a regular wave.*

It is also possible to merge two or more linear wave definitions to create a new merged wave. The merged wave is a simple superposition of the wave com waves. The main purpose for this option is to allow the user to generate seastates that are caused both by swell and wind coming from different direction wind) and their direction are known a merged wave can simply be created by merging both wave definitions.

The merge wave dialog is avaiilable from the top menu, shown in Fig. 149.

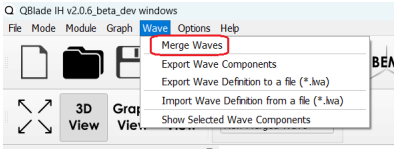


Fig. 149 *The merged wave option in the top wave menu.*

Merged Wave Definition ASCII File

A merged wave definition can also be exported to or imported from a simple ASCII format, that is shown below.

Listing 111 : A merged wave exported in ASCII format

```
--------------------------------------QBlade Wave Definition File-----------------------------------------------
Generated with : QBlade CE v2.0.6_beta_dev windows
Archive Format: 310012
Time : 12:34:36
Date : 18.05.2023

--------------------------------------Object Name---------------------------------------------------------------
New_Merged_Wave                        OBJECTNAME         - the name of the linear wave definition object

--------------------------------------Main Parameters-----------------------------------------------------------
2                                      MERGEDWAVES        - the number of linear waves that are merged in this wave
regular_wave.lwa                       WAVE_1             - the filenames of the waves that are merged
irregular_wave.lwa                     WAVE_2             - the filenames of the waves that are merged
```


[1] Guillaume Ducrozet, Benjamin Bouscasse, Maïté Gouin, Pierre Ferrant, and Félicien Bonnefoy. Cn-stream: open-source library for nonlinear regular waves using s arXiv:1901.10577.

[2] P. H. Taylor. *with the extremesof a Gaussian process*. J. Vib. Acoustics, 1997.

[3] L. Wang, J. Jonkman, G. Hayman, A. Platt, B. Jonkman, A. Robertson. *Recent Hydrodynamic Modeling Enhancements in OpenFAST*. NREL, 2022.

[4] P J Rainey and T R Camp. Constrained non-linear waves for offshore wind turbine design. *Journal of Physics: Conference Series*, 75(1):012067, jul 2007. URL: https 6596/75/1/012067, doi:10.1088/1742-6596/75/1/012067.

Design Load Cases Overview

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

For the certification of a wind turbine, it is essential to assess its lifetime and structural integrity using simulations known as Design Load Cases (DLCs). Th hundred to thousands of DLCs are necessary. Each DLC is characterized by unique boundary conditions, including wind and wave scenarios, and different hundreds of DLCs is inefficient and time-consuming.

QBlade-EE is equipped with a fully featured automatic DLC generator (see Fig. 150), according to the following standards:

- **IEC 61400-1 Ed. 2**
- **IEC 61400-1 Ed. 3**
- **IEC 61400-2 Ed. 2**
- **IEC 61400-3-1 Ed. 1**
- **IEC 61400-3-2 Ed. 2**

There two different ways how DLC's can be generated in QBlade with a high degree of automation. After all DLC simulations have been defined in QBlad

DLC Object Generation (in GUI)

This feature allows to generate a DLC object, which contains the definitions of all simulation timeseries for a specific DLC number from a specific IEC star *Definition ASCII Files*.

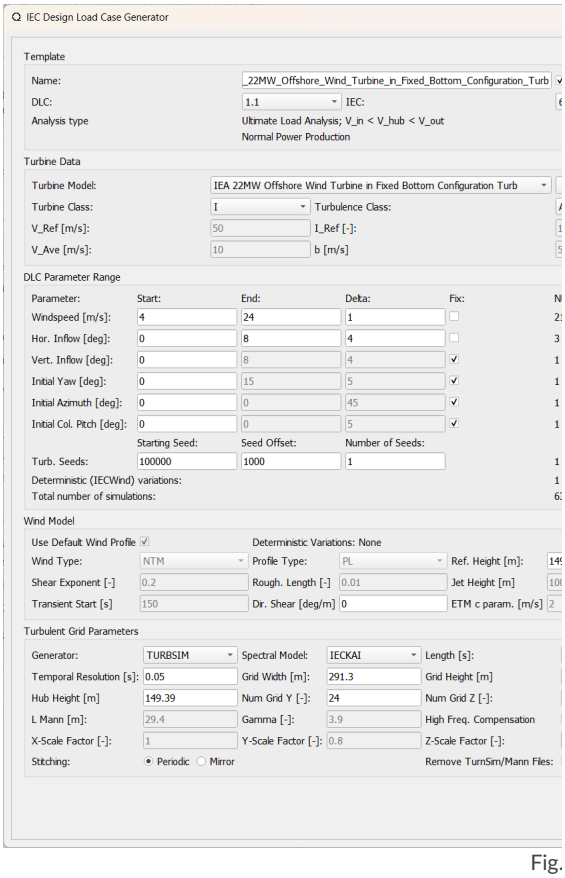


Fig.

Template

The automatic DLC generator can generate DLC's after a user selected standard. In the Template subsection of the dialog the user may choose the DLC, t

Turbine Data

In this section the turbine object, for which the DLC's should be generated, and the *Turbine Class* and *Turbulence Class* can be chosen.

DLC Parameter Range

The user can choose the range and increments of windspeeds, inflow angles and turbine initial conditions that are required for the DLC definition. Depen

Wind Model

In this section the user can choose the wind model for the setup of the DLC. Typically all values are filled out automatically, according to the chosen IEC s

Turbulent Grid Parameters

The turbulent wind fields that are required for some DLC's are generated automatically by TurbSim and imported into QBlade if required. The spatial and t

Environmental Vars

The site specific environmental variables can be defined by the user.

General Sim Settings

Here the user needs to define the duration of the simulation (typically 600s with some added time to remove initial transients) and the timestep of the sir

Rotational Speed Setting

In this section the user can choose how the ramp-up phase should be handled (see Rotational Speed Settings).

Simulation Event(Fault) Settings

If a specific event (such as shut-down, start-up, emergency brake, grid loss, etc.) should be included in the simulation the event definition file can be adde

Structural Sim Settings

These settings concern the structural simulation of the wind turbine and are detailed here: Structural Simulation Settings.

Modal Analysis Settings

A modal analysis can be performed at the end of each DLC, if activated here. As an example, this feature can be used to automatically generate **Campbell**

Stored Sim Data

In this section the user can choose from which timestep and what kind of data should be stored for each generated simulation. Typically the initial transie

Offshore DLC Generation in the GUI

When IEC **61400-3-1** or **61400-3-2** is selected in the *IEC Design Load Case Generator* dialog, the user is asked to choose a *Wave Template* and to provide a template contains information about the the spectrum, spectral discretization etc. and only the key parameters for wave height, wave period or wave dire variations for this DLC (that was previously input into the *Parameter Range* section).

The format of the DLC table that is required is equivalent as described in DLC Generation via Spreadsheets, with the exception that the entries for a few

The table columns that are not required and must be filled out with the keyword *none* are:

3 **Master Simulation** : The simulation template is not required, as all needed data is defined within the *IEC Design Load Case Generator* already. Fill this colu

10 **Hub Height Input File** : The hub height data is automatically generated within QBlade. Fill this column with the keyword *none*.

11 **TurbSim Template** : The TurbSim template file is not required, as all needed data is defined within the *IEC Design Load Case Generator* already. Fill this c

16 **Wave Template** : The Linear Wave template file is not required, as a linear wave object that serves as a template is already defined within the *IEC Desig*

The following columns can be filled with the keyword *auto* so that QBlade will automatically choose the correct values according to the selected IEC stan

17 **Near Surface Current Velocity** : The velocity of the near surface current in [m/s], see Currents. The automatic near surface current velocity is 1% of the

18 **Near Surface Current Direction** : The direction of the near surface current in [°], see Currents. The automatic near surface direction is aligned with the

19 **Near Surface Current Depth** : The depth of the near surface current in [m], see Currents. The automatic depth is 20m.

21 **Sub Surface Current Direction** : The direction of the sub surface current in [°], see Currents. The automatic sub surface current direction is aligned wit

22 **Sub Surface Current Exponent** : The exponent of the sub surface current velocity profile, see Currents. The automatic value for the exponent is 1/7.

Below is an example for such a DLC table, where some entries are replaced with the *none* and *auto* keywords. More information in QBlade DLC tables is f

```
QB_HEXAFLOAT_LC12_s0_ws5_hs1_tp6_mis-30_i0_y0          2200    none    none    5       0       0       0.14    0       none    250     1       6
QB_HEXAFLOAT_LC12_s1_ws5_hs1_tp6_mis30_i0_y0           2200    none    none    5       0       0       0.14    1       none    250     1       6
QB_HEXAFLOAT_LC12_s2_ws5_hs1_tp8_mis-150_i0_y0         2200    none    none    5       0       0       0.14    2       none    250     1       8
QB_HEXAFLOAT_LC12_s3_ws5_hs1_tp8_mis-90_i0_y0          2200    none    none    5       0       0       0.14    3       none    250     1       8
QB_HEXAFLOAT_LC12_s4_ws5_hs1_tp8_mis-30_i0_y0          2200    none    none    5       0       0       0.14    4       none    250     1       8
QB_HEXAFLOAT_LC12_s5_ws5_hs1_tp8_mis30_i0_y0           2200    none    none    5       0       0       0.14    5       none    250     1       8
QB_HEXAFLOAT_LC12_s6_ws5_hs1_tp8_mis90_i0_y0           2200    none    none    5       0       0       0.14    6       none    250     1       8
QB_HEXAFLOAT_LC12_s7_ws5_hs1_tp8_mis150_i0_y0          2200    none    none    5       0       0       0.14    7       none    250     1       8
QB_HEXAFLOAT_LC12_s8_ws5_hs1_tp10_mis-150_i0_y0        2200    none    none    5       0       0       0.14    8       none    250     1       10
QB_HEXAFLOAT_LC12_s9_ws5_hs1_tp10_mis-90_i0_y0         2200    none    none    5       0       0       0.14    9       none    250     1       10
QB_HEXAFLOAT_LC12_s10_ws5_hs1_tp10_mis-30_i0_y0        2200    none    none    5       0       0       0.14    10      none    250     1       10
QB_HEXAFLOAT_LC12_s11_ws5_hs1_tp10_mis30_i0_y0         2200    none    none    5       0       0       0.14    11      none    250     1       10
QB_HEXAFLOAT_LC12_s12_ws5_hs1_tp10_mis90_i0_y0         2200    none    none    5       0       0       0.14    12      none    250     1       10
QB_HEXAFLOAT_LC12_s13_ws5_hs1_tp10_mis150_i0_y0        2200    none    none    5       0       0       0.14    13      none    250     1       10
QB_HEXAFLOAT_LC12_s14_ws5_hs1_tp12_mis-150_i0_y0       2200    none    none    5       0       0       0.14    14      none    250     1       12
QB_HEXAFLOAT_LC12_s15_ws5_hs1_tp12_mis-90_i0_y0        2200    none    none    5       0       0       0.14    15      none    250     1       12
QB_HEXAFLOAT_LC12_s16_ws5_hs1_tp12_mis-30_i0_y0        2200    none    none    5       0       0       0.14    16      none    250     1       12
QB_HEXAFLOAT_LC12_s17_ws5_hs1_tp12_mis30_i0_y0         2200    none    none    5       0       0       0.14    17      none    250     1       12
QB_HEXAFLOAT_LC12_s18_ws5_hs1_tp12_mis90_i0_y0         2200    none    none    5       0       0       0.14    18      none    250     1       12
QB_HEXAFLOAT_LC12_s19_ws5_hs1_tp12_mis150_i0_y0        2200    none    none    5       0       0       0.14    19      none    250     1       12
QB_HEXAFLOAT_LC12_s20_ws5_hs1_tp14_mis30_i0_y0         2200    none    none    5       0       0       0.14    20      none    250     1       14
```

Exporting DLC Definitions

After a *DLC Definition Object* has been defined through the dialog, all individual simulations can be automatically exported as `.sim` files, for an evaluation Export .sim Files from this DLC Definition". To generate Simulation Objects within QBlade press "Create SImulations from this DLC Definition". (see Fig. 1

Fig. 151

DLC Definition via Spreadsheets

Alternatively, to using the GUI based dialog, DLC's may also be generated, based on a spreadsheet software. This gives the user full control over each asp often unique ways.

	1	2	3
	ID	Simulation Length	Master SIM
	Simulation Name	simulation length [s]	.sim
	QB_HEXAFLOAT_LC12_s0_ws5_hs1_tp6_mis-30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s1_ws5_hs1_tp6_mis30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s2_ws5_hs1_tp8_mis-150_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s3_ws5_hs1_tp8_mis-90_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s4_ws5_hs1_tp8_mis-30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s5_ws5_hs1_tp8_mis30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s6_ws5_hs1_tp8_mis90_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s7_ws5_hs1_tp8_mis150_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s8_ws5_hs1_tp10_mis-150_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s9_ws5_hs1_tp10_mis-90_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s10_ws5_hs1_tp10_mis-30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s11_ws5_hs1_tp10_mis30_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s12_ws5_hs1_tp10_mis90_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s13_ws5_hs1_tp10_mis150_i0_y0	2200	Hexafloat_Te
	QB_HEXAFLOAT_LC12_s14_ws5_hs1_tp12_mis-150_i0_y0	220	t_Te
	QB_HEXAFLOAT_LC12_s15_ws5_hs1_tp12_mis-90_i0_y0	220	t_Te

Fig. 152 *Closeup view of a DLC spreadsheet showing the di olu*

The general methodology, when generating DLC's via a spreadsheet, is to define simulation definition (.sim), wind (.inp) and wave (.lwa) template files and *Simulation Definition Objects* or to automatically generate *Simulation Definition ACII Files* from the spreadsheet.

The definition of a single simulation requires 33 entries (columns) in a spreadsheet. The different entries are explained in detail in the following. If an entry

1 **Name** : Each timeseries should have a unique name assigned.

2 **Simulation Length** : The length of the timeseries in [s].

3 **Master Simulation** : The path to a simulation definition template. A relative path based on the spreadsheet location can be used. This needs to be a *Simu*

4 **Events** : The (absolute or relative) path to an event definition file. If no event should be simulation insert the word *none*.

5 **Windspeed** : The windspeed in [m/s].

6 **Horizontal Inflow Angle** : The horizontal inflow angle in [°].

7 **Vertical Inflow Angle** : The vertical inflow angle in [°].

8 **Shear Exponent** : The shear exponent of the power law wind profile.

9 **Turbulence Seed** : The seed that is used by TurbSim for the turbulent windfield generation (if a TurbSim template is defined).

10 **Hub Height Input File** : The (absolute or relative) path of a hub-height wind input file.

11 **TurbSim Template** : The (absolute or relative) path of the TurbSim input file (.inp) that will be used as a template for the generation of turbulent wind fi

12 **Water Depth** : The water depth in [m]. If an onshore turbine is simulated use the value 0.

13 **Significant Height (Hs)** : The significant wave height in [m].

14 **Significant Wave Period (Tp)** : The significant wave period in [s].

15 **Wave Misalignement** : The misalignement between wind and waves in [°]. The wave direction is calculated so that the wave is misaligned from the wir

15 **Wave Seed** : The seed that is used by the wave generator during the generation of wave timeseries from wave spectra.

16 **Wave Template** : The (absolute or relative) path to a Wave Definition ASCII File that is used as a template for the wave generation. Depending on the u

17 **Near Surface Current Velocity** : The velocity of the near surface current in [m/s], see Currents.

18 **Near Surface Current Direction** : The direction of the near surface current in [°], see Currents.

19 **Near Surface Current Depth** : The depth of the near surface current in [m], see Currents.

20 **Sub Surface Current Velocity** : The velocity of the sub surface current in [m/s], see Currents.

21 **Sub Surface Current Direction** : The direction of the sub surface current in [°], see Currents.

22 **Sub Surface Current Exponent** : The exponent of the sub surface current velocity profile, see Currents.

23 **Near Shore Current** : The velocity of the near shore current in [m/s], see Currents.

24 **Near Shore Current Direction** : The direction of the near shore current in [°], see Currents.

25 **Intial Rotor Yaw** : The intial rotor yaw of the turbine at the beginning of the simulation, in [°]

26 **Intial Rotor Azimuth** : The intial rotor azimuthal angle of the turbine at the beginning of the simulation, in [°]

27 **Intial Rotor Pitch** : The intial collective rotor pitch angle at the beginning of the simulation, in [°]

28 **Initial FLoater X Position** : The initial position of the floating wind turbine in X-direction, in [m]

29 **Initial FLoater Y Position** : The initial position of the floating wind turbine in Y-direction, in [m]

30 **Initial FLoater Z Position** : The initial position of the floating wind turbine in Z-direction, in [m]

31 **Initial FLoater X Rotation** : The initial rotation of the floating wind turbine around X, in [°]

32 **Initial FLoater Y Rotation** : The initial rotation of the floating wind turbine around Y, in [°]

33 **Initial FLoater Z Rotation** : The initial rotation of the floating wind turbine around Z, in [°]

DLC Generation via Spreadsheets

Once all DLC's have been defined in the spreadsheet the simulations can either be imported into QBlade or exported as *Simulation Definition ASCII Files*. F

```
QB_HEXAFLOAT_LC12_s0_ws5_hs1_tp6_mis-30_i0_y0          2200    Hexafloat_Template.sim  none    5       0       0       0.14    0       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s1_ws5_hs1_tp6_mis30_i0_y0           2200    Hexafloat_Template.sim  none    5       0       0       0.14    1       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s2_ws5_hs1_tp8_mis-150_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    2       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s3_ws5_hs1_tp8_mis-90_i0_y0          2200    Hexafloat_Template.sim  none    5       0       0       0.14    3       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s4_ws5_hs1_tp8_mis-30_i0_y0          2200    Hexafloat_Template.sim  none    5       0       0       0.14    4       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s5_ws5_hs1_tp8_mis30_i0_y0           2200    Hexafloat_Template.sim  none    5       0       0       0.14    5       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s6_ws5_hs1_tp8_mis90_i0_y0           2200    Hexafloat_Template.sim  none    5       0       0       0.14    6       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s7_ws5_hs1_tp8_mis150_i0_y0          2200    Hexafloat_Template.sim  none    5       0       0       0.14    7       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s8_ws5_hs1_tp10_mis-150_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    8       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s9_ws5_hs1_tp10_mis-90_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    9       DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s10_ws5_hs1_tp10_mis-30_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    10      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s11_ws5_hs1_tp10_mis30_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    11      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s12_ws5_hs1_tp10_mis90_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    12      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s13_ws5_hs1_tp10_mis150_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    13      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s14_ws5_hs1_tp12_mis-150_i0_y0       2200    Hexafloat_Template.sim  none    5       0       0       0.14    14      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s15_ws5_hs1_tp12_mis-90_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    15      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s16_ws5_hs1_tp12_mis-30_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    16      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s17_ws5_hs1_tp12_mis30_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    17      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s18_ws5_hs1_tp12_mis90_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    18      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s19_ws5_hs1_tp12_mis150_i0_y0        2200    Hexafloat_Template.sim  none    5       0       0       0.14    19      DLC1.2_NTM.
QB_HEXAFLOAT_LC12_s20_ws5_hs1_tp14_mis30_i0_y0         2200    Hexafloat_Template.sim  none    5       0       0       0.14    20      DLC1.2_NTM.
```

Now the easiest way to generate all simulations defined in the table above is to place the table and all associated templates (.sim file, .inp file, .lwa file) int

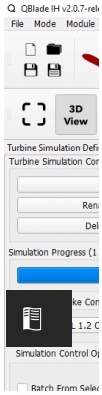
Importing DLC's from a Spreadsheet

To import all simulation defined in a DLC table into QBlade's GUI simply enter the Simulation module and select *Import Simulations from a DLC Table*.

Fig. 1

Exporting DLC's from a Spreadsheet

To export all simulation defined in a DLC table into *Simulation Definition ASCII Files* for batch evaluation in QBlade's CLI (see Sample CLI Call to Start a Bat



CLI Overview

QBlade-EE

This feature is only available in the Enterprise Edition of QBlade.

QBlade-EE can also be executed in a command line interface (CLI). The main purpose of QBlade's CLI is to batch-process large sets of simulations in paral computational overhead from the GUI. When processing large sets of simulations a single instance of QBlade will act as the *thread manager*, in charge of c instances for each simulation that is evaluated and queuing these simulations over all available threads. This approach has the advantage of being highly r crashes for any reason it wont affect any other simulation and the batch run will still complete without the need for user intervention. This section details QBlade CLI and its options.

To start QBlade in CLI mode simply call QBlade's executable from the command line while adding the parameter `-cli` to indicate that instead of starting be operated in CLI mode.

```
QBladeEE -cli
```

If the command above is executed, QBlade prints out the following information on the screen:

```
!!!!! Welcome to QBlade EE v2.0.5.1_alpha windows Command Line Interface (CLI) !!!!!

************************************ OpenCL ******************************************
Available OpenCL Devices:

-d0 : to use CPU: OpenMp (default)
-d1 : to use GPU: OpenCL 2.0 AMD-APP (3380.6) gfx1030
-d2 : to use GPU: OpenCL 3.0 NEO Intel(R) UHD Graphics 770

Specify the OpenCL device by passing -dXX to the CLI, where -d0 is the default
Specify the OpenCL group size by passing -gXX to the CLI, where -g32 is the default
------------------------------------------------------------------------------------------

********************************** Multi Threading ***************************************
Available CPU Cores:

- CPU cores found: 20

Set the parallel threads by passing -tXX to the CLI, where -t1 is the default
------------------------------------------------------------------------------------------

Missing arguments - type 'QBlade -cli help' for a list of options!
```

Since no options have been passed in this example of executing the CLI, the program is exiting without any actions. However, some useful information ha screen. This includes the **OpenCL Devices** as they have been detected by QBlade and the total number of **CPU Cores** that could be found on the machine when evaluating large sets of simulations to inform QBlade which OpenCL device to use and how many simulations should be evaluated in parallel.

The available functionality of the QBlade CLI can be viewed by adding `help` (note that the minus sign is not required when adding the `help` command, v adding `-cli`).

```
QBladeEE -cli help
```

This command prints out an overview of all CLI functionality:

```
************************************ Help ********************************************

!! CLI options can be given in any order !!
!! Multiple files and/or directories can be specified with a single CLI call!!
!! All directories must be input as an ABSOLUTE path !!

List of CLI Options:

help                      - list CLI options
-dX                       - choose compute device X for wake calculations (X must be an integer)
-gX                       - choose opencl group size X for wake calculations (X must be an integer)
-tX                       - choose the number X of paraleel simulation threads used during batch evaluation (X must be an integer)
-oX                       - choose the number X of OMP threads used in each of the parallel simulation threads (X must be an integer)
\directory\simulation.sim - evaluates the simulation definition '\directory\simulation.sim' and save it as *.qpr1
\directory\project.qpr    - evaluates the project '\directory\project.qpr' and save it as .qpr1
\directory                - sets the WORKING_DIR, multiple working directories may be defined
all_sim                   - batch evaluate all .sim files in all WORKING_DIR(s) and saves them as *.qpr1
all_qpr                   - batch evaluate all .qpr files in all WORKING_DIR(s) and save them as *.qpr1
```

```
no_save                 - simulations are not stored after .sim or .qpr files or cutplanes (using post_cut) have been evaluated
remove_wind             - remove the windfield file (.bts) after a simulation definition (.sim) is evaluated
skip                    - skip evaluation of .qpr and .sim files if a .qpr1 file exists or .sim files that were already exported
exp_h2bin               - adds HAWC2BINARY format to auto-export and post-export formats
exp_h2ascii             - adds HAWC2ASCII format to auto-export and post-export formats (only if HAWC2BINARY is not exported)
exp_ascii               - adds ASCII format to auto-export and post-export formats
exp_fastbin             - adds FAST BINARY format to auto-export and post-export formats
exp_modal               - exports modal frequencies (if a modal analysis was performed) during auto-export and post-export
exp_cut_txt             - adds cut-plane txt format to auto-export and post-export formats
exp_cut_vtu             - adds cut-plane vtu format to auto-export and post-export formats
post_exp                - export results and cut-planes from all FINISHED .qpr and .qpr1 files in all WORKING_DIR(s)
flt=\directory\filter.*  - sets a filter file for the generation of all auto-export and post-export formats
dlc=\directory\dlc.*     - create simulations from the dlc table (dlc.*). requires a WORKING_DIR in which the simulations are created
-------------------------------------------------------------------------------------------
```

CLI Functionality

In this section the different CLI options are briefly explained.

`-d`

This parameter is used to specify on which OpenCL device QBlade should be executed. The default value is 0. To execute on the OpenCL device 1, the

`-g`

This parameter specifies the OpenCL work-group size, which has a GPU dependent impact on the OpenCL performance. The work-group size should a default value is 32. To specify a work-group size of 64 you would pass the parameter `-g64`.

`-t`

This parameter sets how many parallel instances of QBlade should be started when evaluating a batch of simulations. The default values is 1. To specif would pass the parameter `-t12`.

`-o`

This parameter sets how many openMp threads should be used for each parallel instance of QB. This is specifically important to fine tune the perform cloud-computing environment.

`c:directorysimulation.sim`

Passing the absolute location of a Simulation Definition ASCII File (*.sim) as one of the parameters adds this simulation definition to the list of simulati Multiple simulation definitions may be added during a single CLI call. Finished simulation definitions are stored as **.qpr1**, to indicate that these are proj been evaluated. Should a simulation fail for any reason the associated project is stored as ***.qpre** instead, to indicate that this is a problematic simulatic

`c:directoryproject.qpr`

Passing the absolute location of a QBlade Project File (*.qpr) adds all simulation definitions within this project to the list of simulations that will be eva may be added during a single CLI call. Finished project files are stored as ***.qpr1**, to indicate that these are project files that have already been evaluate for any reason the associated project is stored as ***.qpre** instead, to indicate that this is a problematic simulation.

`c:directory`

Passing the absolute path of any directory adds this directory to the list of working directories (**WORKING_DIR**). Multiple directories may be added du

`all_sim`

Adding the parameter `all_sim` causes QBlade to add **all *.sim files from all WORKING_DIR(s)** to the list of simulations that will be evaluated.

`all_qpr`

Adding the parameter `all_qpr` causes QBlade to add **all *.qpr files from to all WORKING_DIR(s)** to the list of projects that will be evaluated.

`no_save`

The parameter `no_save` prevents QBlade from automatically storing finished simulations as ***.qpr1** or ***.qpr2** files. Sometimes those files are not explic
results are automatically exported and the user wants to reduce disk memory consumption during very large batch runs.

`remove_wind`

The parameter `remove_wind` removes the binary windfield files (*.bts), that may be automatically generated when a simulation definition file (*.sim) is e
to reduce disk memory usage during very large batch runs.

`skip`

Adding the parameter `skip` causes QBlade to skip the evaluation of a simulation (*.sim) or project (*.qpr) file if an assocated finished project file (*.qpr
results from this simulation have already been exported previously. When using *skip* during post_exp files are only exported if their filename does not

`exp_h2bin`

The parameter `exp_h2bin` adds the HAWC2 binary format to the list of export formats. Whenever a simulation is completed the results of this simula
exported for all specified formats. As default no format is specified, so auto-export if disabled.

`exp_h2ascii`

The parameter `exp_h2ascii` adds the HAWC2 ASCII format to the list of export formats. Whenever a simulation is completed the results of this simul
exported for all specified formats. As default no format is specified, so auto-export if disabled.

`exp_ascii`

The parameter `exp_ascii` adds the ASCII format to the list of export formats. Whenever a simulation is completed the results of this simulation will b
all specified formats. As default no format is specified, so auto-export if disabled.

`exp_fastbin`

The parameter :code:èxp_fastbin`adds the OpenFAST binary format (.outb) to the list of export formats. Whenever a simulation is completed the resul
automatically exported for all specified formats. As default no format is specified, so auto-export if disabled.

`directoryfilter.flt`

Passing the absolute location of a result filter file. All export files that will be generated contain only the variables defined in the filter file. Each line in t
variable name. The variable names in the filter file need to correspond to the exact name of the variable as it is shown in QBlade's graphs.

`exp_modal`

The parameter `exp_modal` exports modal frequencies during auto_exp and post_exp exports. The modal frequencies are only exported if the simulatic
modal analysis. The exported modal frequencies are stored in a file with the appendix *_modal.txt*.

`exp_cut_txt`

The parameter `exp_cut_txt` adds the cut-plane TXT format to the list of export formats. Whenever a cut-plane is evaluated, its velocity field will be au
specified formats. As default no format is specified, so auto-export if disabled.

`exp_cut_vtu`

The parameter `exp_cut_vtu` adds the cut-plane VTU format to the list of export formats. Whenever a cut-plane is evaluated, its velocity field will be a
specified formats. As default no format is specified, so auto-export if disabled.

`post_exp`

The parameter `post_exp` causes QBlade to automatically export the results from all finished project files (*.qpr, *.qpr1, *.qpr2) in all WORKING_DIR(s)
simulations that are already finished when the CLI call is executed and not simulations that are being evaluated during the CLI call. Simulations are exp
have been added to the export format list.

`dlc=directorydlc.*`

This call allows to create simulations from a dlc table definition. It requires the filename of the dlc definition. Furthermore, a WORKING_DIR in which is required.

`dlc=directorydlc.*`

This sets a filter file that is applied during the generation of all auto-export and post-export files.

Sample CLI Call to Start a Batch Run

The following call is an example for a CLI call of QBlade to evaluate and automatically export a batch of simulation definition files located in the folder c:\

```
QBladeEE -cli -d1 -g64 -t12 c:\simulations all_sim exp_h2bin remove_wind skip
```

After this CLI call QBlade will evaluate all simulation definitions (`all_sim`) located in c:\simulations\ over 12 parallel threads (`-t12`). OpenCL device 1 wi work-group size of 64 (`-g64`). The simulation results will automatically be exported to the HAWC2 binary format (`exp_h2bin`). Simulations that have alre previously will be skipped (`skip`) and the automatically generated binary wind fields will be removed after a simulation is finished (`remove_wind`). After ex following info is printed out on the screen:

```
!!!!! Welcome to QBlade EE v2.0.5.1_alpha windows Command Line Interface (CLI) !!!!!

************************************ OpenCL ******************************************
Available OpenCL Devices:

-d0 : to use CPU: OpenMp (default)
-d1 : to use GPU: OpenCL 1.2 CUDA Quadro P6000

Specify the OpenCL device by passing -dXX to the CLI, where -d0 is the default
Specify the OpenCL group size by passing -gXX to the CLI, where -g32 is the default
------------------------------------------------------------------------------------------

********************************** Multi Threading ***************************************
Available CPU Cores:

- CPU cores found: 16

Set the parallel threads by passing -tXX to the CLI, where -t1 is the default
------------------------------------------------------------------------------------------


********************************** Input User Commands ***********************************
1 - OpenCl device:            1
2 - OpenCl groupsize:         64
3 - Parallel Threads:         12
4 - all_sim                   batch evaluate all .sim files in all WORKING_DIR(s) and saves them as .qpr1
5 - WORKING_DIR 0:            c:\simulations
6 - exp_h2bin                 adds HAWC2BINARY format to auto-export and post-export formats
7 - remove_wind               removes bts windfields after auto-generation through .sim files
8 - skip                      skip evauation of .qpr files if a .qpr1 file exists; .sim files that were already exported; cut-planes if a .qpr2 exi

Using OpenCL on  GPU: OpenCL 1.2 CUDA Quadro P6000
------------------------------------------------------------------------------------------


************************ List of sim files that will be evaluated ************************
[1] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y-20.sim
[2] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y0.sim
[3] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y20.sim
[4] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y-20.sim
[5] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y0.sim
[6] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y20.sim
[7] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y-20.sim
[8] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y0.sim
[9] c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y20.sim
[10] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y-20.sim
[11] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y0.sim
[12] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y20.sim
[13] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis0_i0_y-20.sim
[14] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis0_i0_y0.sim
[15] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis0_i0_y20.sim
[16] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis30_i0_y-20.sim
[17] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis30_i0_y0.sim
[18] c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis30_i0_y20.sim
------------------------------------------------------------------------------------------


***************************** Simulation of sim definitions ******************************

...queuing 18 simulations over 12 threads!

...starting evaluation of < Queue Item 1/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y-20.sim ; with ThreadID 0x36e4 at
...starting evaluation of < Queue Item 2/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y0.sim ; with ThreadID 0x1134 at 1
```

```
...starting evaluation of < Queue Item 3/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis-30_i0_y20.sim ; with ThreadID 0x3fc4 at
...starting evaluation of < Queue Item 4/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y-20.sim ; with ThreadID 0x4094 at 1
...starting evaluation of < Queue Item 5/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y0.sim ; with ThreadID 0x11c4 at 16:
...starting evaluation of < Queue Item 6/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis0_i0_y20.sim ; with ThreadID 0x3e10 at 16
...starting evaluation of < Queue Item 7/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y-20.sim ; with ThreadID 0x2d00 at
...starting evaluation of < Queue Item 8/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y0.sim ; with ThreadID 0x3090 at 16
...starting evaluation of < Queue Item 9/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10000_ws31_hs11_tp15_mis30_i0_y20.sim ; with ThreadID 0x3360 at 1
...starting evaluation of < Queue Item 10/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y-20.sim ; with ThreadID 0x1cbc a
...starting evaluation of < Queue Item 11/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y0.sim ; with ThreadID 0xdfc at 1
...starting evaluation of < Queue Item 12/18 > : c:\simulations\QB_HEXAFLOAT_LC63_s10001_ws31_hs11_tp15_mis-30_i0_y20.sim ; with ThreadID 0x2f54 at
```

As can be seen from QBlade's console output an overview of the passed options is given, followed by an overview of the queued simulations before the e themselves starts. The information on the screen will now be updated whenever a simulation instance is finished and a new simulation instance is started of the batch run. After all simulations have been evaluated and exported QBlade will close and return to the command window.

Software in Loop (SIL) Overview

The Software in Loop interface in QBlade provides an easy way of controlling the whole simulation loop of a wind turbine and enable cosimulation within QBlade is compiled as a Dynamic Link Library (.dll, Windows) or as a Shared Object (.so, Unix) and the relevant functionality is exported into an interface.

Through the different functions that are exported the user can explicitly import QBlade projects (.qpr) or Simulation Definition Files (.sim) and then progre is possible to inquire any variable of the simulation and control various aspects of the simulation in response, such as changing the inflow conditions, char generator torque) of the wind turbine. A possible application for the SIL interface is a cosimulation, where the turbine floater can be modeled within a spe the floater, the cosimulation could also model the drivetrain, controller or generator in a more sophisticated way. Another application is controller develop simulation and controlling the turbine actuators in response. When running a multi-turbine simulation within the SIL interface the user may control each s

In general, the high level overview of the SIL interface and the simulation loop, when running the SIL in an external language, looks as follows:

```
loadLibrary()
createInstance()
loadProject()
initializeSimulation()

#start of the simulation loop

for i in range(end):

        ...do something...
        advanceTurbineSimulation()

#end of the simulation loop

storeProject()
closeInstance()
```

Quick Start with the SIL Interface in Python

To test the SIL interface in Python you can simply start the python script `sampleScript.py`, which you find in the folder `PythonInterface` of the QBlade di project file (.qpr) and then runs a simulation loop for 500 timesteps while printing out some results and finally saving the finished simulation as a new proj serve any other particular purpose. Adapt as needed.

Python Examples:

Python Example: Running the QBlade Library. Python Example: Definition of the QBladeLibrary Class.

Matlab Examples:

Matlab Example: Running the QBlade Library. Matlab Example: Definition of the QBladeLibrary Class.

Interface Function Definitions

Listing 112 : QBladeLibInclude.h

```
#all variables and return values are c data types

void setLibraryPath(char *str);

void createInstance(int clDevice, int groupSize);
void loadProject(char *str);
void loadSimDefinition(char *str);
void initializeSimulation();
void runFullSimulation();

void advanceController_at_num(double *vars, int num);
void advanceTurbineSimulation();

void storeProject(char *str);
void exportResults(int type, char *filepath, char* filename, char *filter);
void closeInstance();
void setLogFile(char *str);

void loadTurbulentWindBinary(char *str);
void addTurbulentWind(double windspeed, double refheight, double hubheight, double dimensions, int gridPoints, double length, double dT, char

void setPowerLawWind(double windspeed, double horAngle, double vertAngle, double shearExponent, double referenceHeight);
void setDebugInfo(bool isDebug);
void setUseOpenCl(bool isOpenCl);
```

```
25
26  void setGranularDebug(bool dStr, bool dSim, bool dTurb, bool dCont, bool dSer);
27  void setTimestepSize(double timestep);
28  void setRPMPrescribeType_at_num(int type, int num);
29  void setRampupTime(double time);
30  void setInitialConditions_at_num(double yaw, double pitch, double azimuth, double rpm, int num);
31  void setTurbinePosition_at_num(double x, double y, double z, double rotx, double roty, double rotz, int num);
32  void setControlVars_at_num(double *vars, int num);
33  void setExternalAction(char *action, char *id, double val, double pos, char *dir, bool isLocal, int num);
34
35  void getWindspeed(double posx, double posy, double posz, double *velocity);
36  void getWindspeedArray(double *posx, double *posy, double *posz, double *velx, double *vely, double *velz, int arraySize);
37  void getTowerBottomLoads_at_num(double *loads, int num);
38  void getTurbineOperation_at_num(double *vars, int num);
39  double getCustomData_at_num(char *str, double pos, int num);
40  double getCustomSimulationData(char *str);
```

Interface Function Documentation

In the following, the functionality that is exported from the QBlade dll or shared object is described and the function arguments and return types are give effect for multi turbine simulations.

`void setLibraryPath(char *atr)`

This function sets the location of the QBlade dll or shared object so that the QBlade instance knows about its location. **This function must be called fi** license files.

`void createInstance(int clDevice = 0, int groupSize = 32)`

This function creates a new instance of QBlade. The OpenCL device and the OpenCL group-size can both be specified in the arguments. **Calling this fu**

`void loadProject(char *str)`

This function loads a simulation definition from a QBlade project (.qpr) into the QBlade instance. The file location has to be passed as a *char pointer*. Fi definitions, the first simulation definition of the project file (in alphabetic order) is loaded into the SIL interface.

`void loadSimDefinition(char *str)`

This function loads a simulation definition (.sim) file into the QBlade instance. The (.sim) files are ASCII files and any aspect of the simulation can be ch can be passed as absolute or as relative paths.

`void initializeSimulation()`

This function initializes the simulation, e.g. the simulation is reset and structural ramp-up is carried out.

`void runFullSimulation()`

This function runs all timesteps for all turbines of the simulation as defined in the simulation object. This is equivalent to pressing the *Start Simulation* l function it is not possible to *interact* with the simulation before it is finished. To interact with the simulation you need to create your own simulation lo timestep.

`void advanceController_at_num(double *vars, int num = 0)`

This function advances the controller shared library that is assigned to the selected turbine (argument *num*). When calling this function the controller c `setControlVars_at_num(double *vars, int num = 0)`). The controller outputs are also returned in the *vars* array, and can be processed further:

- vars[0] = generator torque [Nm]
- vars[1] = yaw angle [deg]
- vars[2] = pitch blade 1 [deg]
- vars[3] = pitch blade 2 [deg]
- vars[4] = pitch blade 3 [deg]

`void advanceTurbineSimulation()`

This function advances the turbine simulation for all turbines and finishes the timestep.

`void storeProject(char *str)`

This functions stores a project file. The file location has to be passed as a *char pointer*. File names can be passed as absolute or as relative paths.

`void exportResults(int type, char *filepath, char* filename, char *filter)`

This function saves the results of the current simulation to a file in the chosen format

- type: specifies the export format: 0: QBlade ASCII; 1: HAWC2 ASCII; 2: HAWC2 BINARY; 3: OpenFAST BINARY
- filepath: sets the folder in which the export file(s) will be stored
- filename: sets the name of the export file, do not add an extension. The extension will be added automatically, based on the chosen expo at
- filter: allows to point to a global result filter file (see Global Export Filter), specify full fill path and extension

`void setLogFile(char *str)`

This functions sets the path to a log file that will be created to store the dobug output. This is helpful when accessing the SIL interface from a tool that

```
void closeInstance()
```

This function closes the instance of QBlade and frees the memory.

```
void loadTurbulentWindBinary(char *str)
```

This function allows to load a turbulent windfield that is stored in binary format. The file location has to be passed as a *char pointer*. File names can be

```
void addTurbulentWind(double windspeed, double refheight, double hubheight, double dimensions, int gridPoints,double length, double dT, char *turbul
```

This function allows to define and add a turbulent windfield (using TurbSim) to the simulation. If a turbulent windfield is used the function `setPowerLaw`

- windspeed: the mean windspeed at the reference height [m/s]
- refheight: the reference height [m]
- hubheight: the hubheight, more specifically the height of the windfield center [m]
- dimensions: the y- and z- dimensions of the windfield in meters [m]
- length: the simulated length of the windfield in seconds [s]
- dT: the temporal resolution of the windfield [s]
- turbulenceClass: the turbulence class, can be “A”, “B” or “C”
- turbulenceType: the turbulence type, can be “NTM”, “ETM”, “xEWM1” or “xEWM50” - where x is the turbine class (1,2 or 3)
- seed: the random seed for the turbulent windfield
- vertInf: vertical inflow angle in degrees [deg]
- horInf: horizontal inflow angle in degrees [deg]

```
void setPowerLawWind(double windspeed, double horAngle, double vertAngle, double shearExponent, double referenceHeight)
```

This function can be called before or at any time after the simulation has been initialized with `initializeSimulation()` to statically or dynamically char (https://en.wikipedia.org/wiki/Wind_profile_power_law) and its inflow direction. The arguments for this function are:

- windspeed: constant windspeed in m/s [m/s]
- horAngle: the horizontal inflow angle in degrees [deg]
- vertAngle: the vertical inflow angle in degrees [deg]
- shearExponent: this is the exponent for the power law boundary layer profile, if this is set to 0 the windspeed is constant with height [-]
- referenceHeight: this is the height at which the velocity in the boundary layer is the defined windspeed, usually set to the hubheight [m]
- exemplary call: addTurbulentWind(12,115,115,220,20,60,0.1,”A”,”NTM”,1000000,0,0);

```
void setDebugInfo(bool isDebug)
```

This function enables the debug output if set to true.

```
void setGranularDebug(bool dStr, bool dSim, bool dTurb, bool dCont, bool dSer)
```

This function enables a granular debug output.

- dStr: enable structural model debug info
- dSim: enable simulation debug info
- dTurb: enable turbine debug info
- dCont: enable controller debug info
- dSer: enable serializer debug info

```
void setTimestepSize(double timestep)
```

This function can be used to set the timestep size (in [s]) if the user wants to change this value from the project or simulation definition file. It needs to

```
void setRPMPrescribeType_at_num(int type, int num = 0)
```

This function can be used to change the rpm prescribe type. It needs to be called before `initializeSimulation()`.

- 0 - RPM prescribed during ramp-up only
- 1 - RPM prescribed for the whole simulation
- 3 - no prescribed RPM

```
void setRampupTime(double time)
```

This function can be used to change the ramp-up time from the value specified in the project or simulation file, call before `initializeSimulation()`.

```
void setInitialConditions_at_num(double yaw, double pitch, double azimuth, double rpm, int num = 0)
```

This function may be used to set the turbine initial yaw [deg], collective pitch [deg], azimuthal angle [deg] and initial rotSpeed [rpm] to a value differen `initializeSimulation()`.

```
void setTurbinePosition_at_num(double x, double y, double z, double rotx, double roty, double rotz, int num = 0)
```

This function sets the turbine tower bottom x, y and z position [m], and xrot, yrot zrot rotation [deg]. It can be called before `initializeSimul` if for example during cosimulation with a hydrodynamics software that models the floater.

`void setControlVars_at_num(double *vars, int num = 0)`

This function applies the control actions to the selected turbine (argument *num*) for torque, pitch and yaw angle. If it is called after the function `advanc` be modified). The following data needs to be passed in the array *vars*.

- vars[0] = generator torque [Nm];
- vars[1] = yaw angle [deg];
- vars[2] = pitch blade 1 [deg];
- vars[3] = pitch blade 2 [deg];
- vars[4] = pitch blade 3 [deg];

`void setExternalAction(char *action, char *id, double val, double pos, char *dir, bool isLocal, int num)`

This is a general purpose function that can be used to apply an external action to the simulated turbine.

The action can be of different types, defined by the parameter **action**. All of these actions are not accumulated and are reset at every timestep, or in o function at every timestep or it will automatically be reset to zero. The different types are:

- ADDMASS: adds mass to a location, in [kg]
- ADDFORCE: adds a force to a location, in [N]
- ADDTORQUE: adds a torque to a location, in [Nm]
- SETLENGTH: sets the delta Length of a cable, in [m]
- SETAFC: sets the state of an AFC element [-]
- SETTORQUE: sets the generator torque, in [Nm]
- SETYAW: sets the yaw angle, in [deg]
- SETPITCH: sets the pitch angle for BLD_X, in [deg]
- SETBRAKE: sets the brake modulation [0-1]

Some actions are applied to a certain location ID, indicated by the parameter **id**, the different locations are:

- CAB_<X>: applies the action to the guycable with ID <X>. Actions on cables are: SETLENGTH, ADDMASS, ADDFORCE
- MOO_<X>: applies the action to the mooring line with ID <X>. Actions on moorings are: SETLENGTH, ADDMASS, ADDFORCE
- SMOO_<X>: applies the action to the shared mooring line with ID <X>. Actions on moorings are: SETLENGTH, ADDMASS, ADDFORCE
- TRQ: applies the action to the torquetube. Actions on the torquetube are: ADDFORCE, ADDTORQUE, ADDMASS
- BLD_<X>: applies the action to blade <X>. Actions on the blades are: ADDFORCE, ADDTORQUE, ADDMASS
- STR_<X>_<Y>: applies the action to strut <X> of blade <Y>. Actions on the struts are: ADDFORCE, ADDTORQUE, ADDMASS
- AFC_<X>_<Y>: applies the action to AFC <X> of blade <Y>. Actions on the AFC elements are: SETAFC
- SUB_<X>: applies the action to the substructure element with ID <X>. Actions on the substructure elements are: ADDFORCE, ADDTORQUE, ADI
- JNT_<X>: applies the action to the substructure joint with ID <X>. Actions on the substructure joints are: ADDFORCE, ADDTORQUE, ADDMASS
- HUB: applies the action to the free LSS hub node. Actions on the hub node are: ADDFORCE, ADDTORQUE, ADDMASS
- HUBFIXED: applies the action to the fixed non-rotating hub node. Actions on the hub node are: DDFORCE, ADDTORQUE, ADDMASS

The remaining parameters are used to further define the action that is applied, their coordinate systems, etc.

- The parameter **val** specifies the mass [kg], torque [Nm], force [N], delta length [m] or AFC state [-].
- The parameter **pos** sets the normalized position [0-1] at which the mass, force or torque is applied. Only has an effect on elements, not on nodes.
- The parameter **dir** specifies the direction along which the force or torque is applied, options are “X”, “Y”, “Z”.
- The parameter **isLocal** specifies sets whether the direction is defined in global or local (element or node) coordinates.
- The parameter **num** specifies the turbine instance to which the action is applied.

`void getWindspeed(double x, double y, double z, double *velocity)`

This function can be called to get the current windspeed at the chosen position (x,y,z), returns the windspeed vector in the *double pointer* velocity.

- velocity[0] = x-component [m/s];
- velocity[1] = y-component [m/s];
- velocity[2] = z-component [m/s];

`void getWindspeedArray(double *posx, double *posy, double *posz, double *velx, double *vely, double *velz, int arraySize)`

This function can be called to get the current windspeed for an array of positions

- posx = double array of position x-components;
- posy = double array of position y-components;
- posz = double array of position z-components;
- velx = double array of velocity x-components evaluated at the pos array;
- vely = double array of velocity y-components evaluated at the pos array;
- velz = double array of velocity z-components evaluated at the pos array;
- arraySize = the size of the pos and velocity arrays;

`void getTowerBottomLoads_at_num(double *loads, int num)`

This function can be used to obtain the loads at the bottom of the tower. The main purpose of this is to be used in conjunction with the `setTurbinePos` floater.

```
void getTurbineOperation_at_num(double *vars, int num = 0)
```

This function returns useful turbine operational parameters from the selected turbine (argument *num*). Typically, this data is used to feed the logic of a

- vars[0] = rotational speed [rad/s]
- vars[1] = power [W]
- vars[2] = Abs HH wind velocity [m/s]
- vars[3] = yaw angle [deg]
- vars[4] = pitch blade 1 [deg]
- vars[5] = pitch blade 2 [deg]
- vars[6] = pitch blade 3 [deg]
- vars[7] = oop blade root bending moment blade 1 [Nm]
- vars[8] = oop blade root bending moment blade 2 [Nm]
- vars[9] = oop blade root bending moment blade 3 [Nm]
- vars[10] = ip blade root bending moment blade 1 [Nm]
- vars[11] = ip blade root bending moment blade 2 [Nm]
- vars[12] = ip blade root bending moment blade 3 [Nm]
- vars[13] = tor blade root bending moment blade 1 [Nm]
- vars[14] = tor blade root bending moment blade 2 [Nm]
- vars[15] = tor blade root bending moment blade 3 [Nm]
- vars[16] = oop tip deflection blade 1 [m]
- vars[17] = oop tip deflection blade 2 [m]
- vars[18] = oop tip deflection blade 3 [m]
- vars[19] = ip tip deflection blade 1 [m]
- vars[20] = ip tip deflection blade 2 [m]
- vars[21] = ip tip deflection blade 3 [m]
- vars[22] = tower top acceleration in global X [m/s^2]
- vars[23] = tower top acceleration in global Y [m/s^2]
- vars[24] = tower top acceleration in global Z [m/s^2]
- vars[25] = tower top fore aft acceleration [m/s^2]
- vars[26] = tower top side side acceleration [m/s^2]
- vars[27] = tower top X position [m]
- vars[28] = tower top Y position [m]
- vars[29] = tower bottom force along global X [Nm]
- vars[30] = tower bottom force along global Y [Nm]
- vars[31] = tower bottom force along global Z [Nm]
- vars[32] = tower bottom bending moment along global X [Nm]
- vars[33] = tower bottom bending moment along global Y [Nm]
- vars[34] = tower bottom bending moment along global Z [Nm]
- vars[35] = current time [s]
- vars[36] = azimuthal position of the LSS [deg]
- vars[37] = azimuthal position of the HSS [deg]
- vars[38] = HSS torque [Nm]
- vars[39] = wind speed at hub height [m/s]
- vars[40] = HH wind velocity x [m/s]
- vars[41] = HH wind velocity y [m/s]
- vars[42] = HH wind velocity z [m/s]

```
double getCustomData_at_num(char *str, double pos = 0, int num = 0)
```

This function can be used to access the current value from an arbitrary turbine simulation variable in QBlade. Specify the data name as is would appea ‘Angle of Attack at 0.25c (at section) Blade 1 [deg]’ you can specify the normalized position along the blade length using the ‘pos’ variable. As an exam `getCustomData_at_num("Angle of Attack at 0.25c (at section) Blade 1 [deg]", 0.85,0)`.

```
double getCustomSimulationData(char *str)
```

This function can be used to access the current value from an arbitrary *simulation time graph* variable in QBlade.

Python Example: Running the QBlade Library

The following code example (*sampleScript.py*) is an example for a light weight Python script that utilizes the QBlade SIL interface. There are many ways to sophisticated algorithms could be implemented instead of using a standard controller. This exemplary script only uses a small amount of the fun ity

In this Python example script the directory *dll_directory* is searched for shared library files. If a shared library file is found, the library is loaded by creating a
been created any function of the QBlade library can be accessed by calling `QBLIB.function_XYZ()`. All lines of code that are needed to load the QBlade lib

After the QBlade library has been loaded a simulation object is imported and a simulation is started over 500 timesteps. During the simulation loop differ
advanced and its signals are passed to the turbine actuators. After the simulation loop has finished the simulation is stored into a project file, for later insp

Listing 113 : sampleScript.py

```
import os
import sys
from ctypes import *
from QBladeLibrary import QBladeLibrary

# Define the directory where the QBlade library is located
dll_directory = "../"

# Search for library files matching the pattern QBlade*.dll or QBlade*.so
dll_files = [f for f in os.listdir(dll_directory) if 'QBlade' in f and ('.dll' in f or '.so' in f)]

# Check if any matching files are found
if not dll_files:
        print('No matching QBlade*.dll or QBlade*.so files found in the specified directory:',os.path.abspath(dll_directory))
        sys.exit(1)  # Exit the script with a non-zero status to indicate an error

# Use the first matching file
dll_file_path = os.path.join(dll_directory, dll_files[0])

# Display the selected shared library file
print(f'Using shared library file: {dll_file_path}')

#loading the QBlade library from the folder below the location of sampleScript.py, if calling this script not from the script folder directly
QBLIB = QBladeLibrary(dll_file_path)

#creation of a QBlade instance from the library
QBLIB.createInstance(1,32)

#loading a project or sim-file, in this case the DTU_10MW_Demo project or simulation definition file
#QBLIB.loadSimDefinition(b"./DTU_10MW_Demo.sim") #uncomment this line to load a simulation definition file
QBLIB.loadProject(b"./NREL_5MW_Sample.qpr")

#initializing the sim and ramp-up phase, call before starting the simulation loop
QBLIB.initializeSimulation()

#we will run the simulation for 500 steps before storing the results
number_of_timesteps = 500

#start of the simulation loop
for i in range(number_of_timesteps):

        #advance the simulation
        QBLIB.advanceTurbineSimulation()

        #assign the c-type double array 'loads' with length [6], initialized with zeros
        loads = (c_double * 6)(0,0,0,0,0,0)
        #retrieve the tower loads and store the in the array 'loads' by calling the function getTowerBottomLoads_at_num()
        QBLIB.getTowerBottomLoads_at_num(loads,0)

        #uncomment the next line to try changing the position of the turbine dynamically
        #QBLIB.setTurbinePosition_at_num(-0.2*i,0,0,0,i*0.1,i*0.1,0)

        #example how to extract a variable by name from the simulation, call as often as needed with different variable names, extracting rpm
        rpm = QBLIB.getCustomData_at_num(b"Rotational Speed [rpm]",0,0)
        time = QBLIB.getCustomData_at_num(b"Time [s]",0,0) #example how to extract the variable 'Time' by name from the simulation
        AoA = QBLIB.getCustomData_at_num(b"Angle of Attack at 0.25c (at section) Blade 1 [deg]",0.85,0) #example how to extract the variable

        #example how to extract a 3 length double array with the x,y,z windspeed components at a global position of x=-50,Y=0,Z=100m from the
        windspeed = (c_double * 3)(0,0,0)
        QBLIB.getWindspeed(-50,0,100,windspeed)

        #assign the c-type double array 'ctr_vars' with length [5], initialized with zeros
        ctr_vars = (c_double * 5)(0);
        #advance the turbine controller and store the controller signals in the array 'ctr_vars'
        QBLIB.advanceController_at_num(ctr_vars,0)

        #pass the controller signals in 'ctr_vars' to the turbine by calling setControlVars_at_num(ctr_vars,0)
        QBLIB.setControlVars_at_num(ctr_vars,0)

        #print out a few of the recorded data, in this case torque, tower bottom force along z (weight force) and rpm
        print("Time:","{:3.2f}".format(time),"   Windspeed:","{:2.2f}".format(windspeed[0]),"  Torque:","{:1.4e}".format(ctr_vars[0]),"    RP

#the simulation loop ends here after all 'number_of_timesteps have been evaluated

#storing the finished simulation in a project as NREL_5MW_Sample_completed, you can open this file to view the results of the simula                nsi
QBLIB.storeProject(b"./NREL_5MW_Sample_completed.qpr")

#storing the simulation results in QBlade ASCII format in the file NREL_5MW_Sample_results.txt
```

```
QBLIB.exportResults(0,b"./",b"NREL_5MW_Sample_results",b"")

#closing the QBlade instance to free memory
QBLIB.closeInstance()

#unloading the QBlade library
del QBLIB.lib
```

Python Example: Definition of the QBladeLibrary Class

The script *QBladeLibrary.py* defines the class *QBladeLibrary* and loads the shared object. This script is just a suggestion on how to interface with the QBlad

Listing 114 : QBladeLibrary.py

```
from ctypes import *
from sys import platform

class QBladeLibrary:

        def __init__(self, shared_lib_path):

                try:
                        self.lib = CDLL(shared_lib_path)
                        print("Successfully loaded ", shared_lib_path)
                except Exception as e:
                        print("Could not load the file ", shared_lib_path)
                        print(e)
                        return

                #setting the library Path, so that the Library knows about its location!
                self.lib.setLibraryPath(shared_lib_path.encode('utf-8')) #setting the library Path, so that the DLL knows about its location

                #here the imported functions are defined

                self.loadProject = self.lib.loadProject
                self.loadProject.argtype = c_char_p
                self.loadProject.restype = c_void_p

                self.loadSimDefinition = self.lib.loadSimDefinition
                self.loadSimDefinition.argtype = c_char_p
                self.loadSimDefinition.restype = c_void_p

                self.getCustomData_at_num = self.lib.getCustomData_at_num
                self.getCustomData_at_num.argtypes = [c_char_p, c_double, c_int]
                self.getCustomData_at_num.restype = c_double

                self.getCustomSimulationData = self.lib.getCustomSimulationData
                self.getCustomSimulationData.argtype = c_char_p
                self.getCustomSimulationData.restype = c_double

                self.getWindspeed = self.lib.getWindspeed
                self.getWindspeed.argtypes = [c_double, c_double, c_double, c_double * 3]
                self.getWindspeed.restype = c_void_p

                self.getWindspeedArray = self.lib.getWindspeedArray
                self.getWindspeedArray.argtypes = [POINTER(c_double), POINTER(c_double), POINTER(c_double), POINTER(c_double), POINTER(c_dou
                self.getWindspeedArray.restype = c_void_p

                self.storeProject = self.lib.storeProject
                self.storeProject.argtype = c_char_p
                self.storeProject.restype = c_void_p

                self.exportResults = self.lib.exportResults
                self.exportResults.argtypes = [c_int, c_char_p, c_char_p, c_char_p]
                self.exportResults.restype = c_void_p

                self.setLibraryPath = self.lib.setLibraryPath
                self.setLibraryPath.argtype = c_char_p
                self.setLibraryPath.restype = c_void_p

                self.setLogFile = self.lib.setLogFile
                self.setLogFile.argtype = c_char_p
                self.setLogFile.restype = c_void_p

                self.createInstance = self.lib.createInstance
                self.createInstance.argtypes = [c_int, c_int]
                self.createInstance.restype = c_void_p

                self.closeInstance = self.lib.closeInstance
                self.closeInstance.restype = c_void_p

                self.addTurbulentWind = self.lib.addTurbulentWind
                self.addTurbulentWind.argtypes = [c_double, c_double, c_double, c_double, c_int, c_double, c_double, c_char_p, c_ch
                self.addTurbulentWind.restype = c_void_p

                self.setExternalAction = self.lib.setExternalAction
```

```
                self.setExternalAction.argtypes = [c_char_p, c_char_p, c_double, c_double, c_char_p, c_bool, c_int]
                self.setExternalAction.restype = c_void_p

                self.loadTurbulentWindBinary = self.lib.loadTurbulentWindBinary
                self.loadTurbulentWindBinary.argtype = c_char_p
                self.loadTurbulentWindBinary.restype = c_void_p

                self.setTimestepSize = self.lib.setTimestepSize
                self.setTimestepSize.argtype = c_double
                self.setTimestepSize.restype = c_void_p

                self.setInitialConditions_at_num = self.lib.setInitialConditions_at_num
                self.setInitialConditions_at_num.argtypes = [c_double, c_double, c_double, c_double, c_int]
                self.setInitialConditions_at_num.restype = c_void_p

                self.setRPMPrescribeType_at_num = self.lib.setRPMPrescribeType_at_num
                self.setRPMPrescribeType_at_num.argtypes = [c_int, c_int]
                self.setRPMPrescribeType_at_num.restype = c_void_p

                self.setRampupTime = self.lib.setRampupTime
                self.setRampupTime.argtype = c_double
                self.setRampupTime.restype = c_void_p

                self.setTurbinePosition_at_num = self.lib.setTurbinePosition_at_num
                self.setTurbinePosition_at_num.argtypes = [c_double, c_double, c_double, c_double, c_double, c_double, c_int]
                self.setTurbinePosition_at_num.restype = c_void_p

                self.getTowerBottomLoads_at_num = self.lib.getTowerBottomLoads_at_num
                self.getTowerBottomLoads_at_num.argtypes = [c_double * 6, c_int]
                self.getTowerBottomLoads_at_num.restype = c_void_p

                self.initializeSimulation = self.lib.initializeSimulation
                self.initializeSimulation.restype = c_void_p

                self.advanceTurbineSimulation = self.lib.advanceTurbineSimulation
                self.advanceTurbineSimulation.restype = c_void_p

                self.advanceController_at_num = self.lib.advanceController_at_num
                self.advanceController_at_num.argtypes = [c_double * 5, c_int]
                self.advanceController_at_num.restype = c_void_p

                self.setDebugInfo = self.lib.setDebugInfo
                self.setDebugInfo.argtype = c_bool
                self.setDebugInfo.restype = c_void_p

                self.setUseOpenCl = self.lib.setUseOpenCl
                self.setUseOpenCl.argtype = c_bool
                self.setUseOpenCl.restype = c_void_p

                self.setGranularDebug = self.lib.setGranularDebug
                self.setGranularDebug.argtypes = [c_bool, c_bool, c_bool, c_bool, c_bool]
                self.setGranularDebug.restype = c_void_p

                self.setControlVars_at_num = self.lib.setControlVars_at_num
                self.setControlVars_at_num.argtypes = [c_double * 5, c_int]
                self.setControlVars_at_num.restype = c_void_p

                self.getTurbineOperation_at_num = self.lib.getTurbineOperation_at_num
                self.getTurbineOperation_at_num.argtypes = [c_double * 41, c_int]
                self.getTurbineOperation_at_num.restype = c_void_p

                self.setPowerLawWind = self.lib.setPowerLawWind
                self.setPowerLawWind.argtypes = [c_double, c_double, c_double, c_double, c_double]
                self.setPowerLawWind.restype = c_void_p

                self.runFullSimulation = self.lib.runFullSimulation
                self.runFullSimulation.restype = c_void_p
```

Matlab Example: Running the QBlade Library

This is an example for using the QBlade library within Matlab. It reproduces the Python example above. An object of the class QBladeLibrary, that contain

Listing 115 : sampleScript.m

```
%%
clear all
close all
clc

% Find all files containing 'QBlade' and either '.dll' or '.so' in their names
libSearchDirectory = '../';
sharedLibFiles = dir(fullfile(libSearchDirectory, '*QBlade*'));
sharedLibFiles = sharedLibFiles(contains({sharedLibFiles.name}, {'.dll', '.so'}));

if isempty(sharedLibFiles)
        fprintf('No matching QBlade*.dll files or QBlade*.so found in the specified directory.');
```

```
        return;
end

% Use the first matching dll file and print out its name
sharedLibFilePath = fullfile(libSearchDirectory, sharedLibFiles(1).name);
fprintf('Using DLL file: %s\n', sharedLibFilePath);

% create an object of the class 'QBladeLibrary' that contains all interface
% functions
QBLIB = QBladeLibrary(sharedLibFilePath);

QBLIB.createInstance(1,32);

% since matlab is unable to display the console output from the library, we
% store the output in a log file
QBLIB.setLogFile('./LogFile.txt')

QBLIB.loadProject('NREL_5MW_Sample.qpr')

QBLIB.initializeSimulation()

number_of_timesteps = 500;

f = waitbar(0,'Initializing Simulation') ;

for i = 1:1:number_of_timesteps

        %advance the simulation
        QBLIB.advanceTurbineSimulation()

        %assign the c-type double array 'loads' with length [6], initialized with zeros
        loads = libpointer('doublePtr',zeros(6,1));
        %retrieve the tower loads and store the in the array 'loads' by calling the function getTowerBottomLoads_at_num()
        QBLIB.getTowerBottomLoads_at_num(loads,0);
        %dereferencing the 'loads' pointer and accessing its first value
        loads.Value(1);

        %uncomment the next line to try changing the position of the turbine dynamically
        %QBLIB.setTurbinePosition_at_num(-0.2*i,0,0,0,i*0.1,i*0.1,0)

        %example how to extract a variable by name from the simulation, call as often as needed with different variable names, extracting rpm
        rpm = QBLIB.getCustomData_at_num('Rotational Speed [rpm]',0,0);
        t = QBLIB.getCustomData_at_num('Time [s]',0,0);  %example how to extract the variable 'Time' by name from the simulation
        AoA = QBLIB.getCustomData_at_num('Angle of Attack at 0.25c (at section) Blade 1 [deg]',0.85,0); %example how to extract the variable

        %example how to extract a 3 length double array with the x,y,z windspeed components at a global position of x=-50,Y=0,Z=100m from the
        windspeed = libpointer('doublePtr',zeros(3,1));
        QBLIB.getWindspeed(-50,0,100,windspeed);

        %assign the c-type double array 'ctr_vars' with length [5], initialized with zeros
        ctr_vars = libpointer('doublePtr',zeros(5,1));
        %advance the turbine controller and store the controller signals in the array 'ctr_vars'
        QBLIB.advanceController_at_num(ctr_vars,0)

        %pass the controller signals in 'ctr_vars' to the turbine by calling setControlVars_at_num(ctr_vars,0)
        QBLIB.setControlVars_at_num(ctr_vars,0)

        fprintf('Time: %3.2f    Windspeed: %2.2f    Torque: %1.4e       RPM: %2.2f      Pitch: %2.2f    AoA at 85%%: %2.2f\n',t,windspeed.Val

        waitbar(i/number_of_timesteps,f,'QBlade Simulation Running')

end

close(f)

%storing the finished simulation in a project as NREL_5MW_Sample_completed, you can open this file to view the results of the simulation insi
QBLIB.storeProject('./NREL_5MW_Sample_completed.qpr')

%storing the simulation results in QBlade ASCII format in the file NREL_5MW_Sample_results.txt
QBLIB.exportResults(0,'./','NREL_5MW_Sample_Results','')

%closing the instance of the shared library, if this fail it can lead to unexpected behavior
QBLIB.closeInstance()

%unloading the shared library
QBLIB.unload()
```

Matlab Example: Definition of the QBladeLibrary Class

This code shows how the class *QBladeLibrary* is defined in the Matlab environment. To load the library, a header file *QBladeLibInclude.h* is required that co

Listing 116 : QBladeLibrary.m

```
classdef QBladeLibrary
        properties
                lib % DLL handle
```

```
	end

	methods
		% Constructor
		function obj = QBladeLibrary(dllPath)
			% Load DLL
			obj.lib = loadlibrary(dllPath,'QBladeLibInclude.h','alias','QBLIB');
			calllib('QBLIB','setLibraryPath',dllPath)
		end

		% Destructor
		function unload(obj)
			% Unload Library
			if libisloaded('QBLIB')
				unloadlibrary 'QBLIB'
			end;
		end

		% Function to call library function
		function createInstance(obj,clDevice,groupSize)
			calllib('QBLIB', 'createInstance', clDevice, groupSize);
		end

		function loadProject(obj,str)
			calllib('QBLIB', 'loadProject', str);
		end

		function loadSimDefinition(obj,str)
			calllib('QBLIB', 'loadSimDefinition', str);
		end

		function initializeSimulation(obj)
			calllib('QBLIB', 'initializeSimulation');
		end

		function runFullSimulation(obj)
			calllib('QBLIB', 'runFullSimulation');
		end

		function advanceController_at_num(obj,vars,num)
			calllib('QBLIB', 'advanceController_at_num', vars, num);
		end

		function advanceTurbineSimulation(obj)
			calllib('QBLIB', 'advanceTurbineSimulation');
		end

		function storeProject(obj,str)
			calllib('QBLIB', 'storeProject',str);
		end

		function exportResults(obj, type, filepath, filename, filter)
			calllib('QBLIB', 'exportResults',type, filepath, filename, filter);
		end

		function closeInstance(obj)
			calllib('QBLIB', 'closeInstance');
		end

		function setLogFile(obj,str)
			calllib('QBLIB', 'setLogFile',str);
		end

		function loadTurbulentWindBinary(obj,str)
			calllib('QBLIB', 'loadTurbulentWindBinary', str);
		end

		function addTurbulentWind(obj,windspeed, refheight, hubheight, dimensions, gridPoints, length, dT, turbulenceClass, turbulen
			calllib('QBLIB', 'addTurbulentWind', windspeed, refheight, hubheight, dimensions, gridPoints, length, dT, turbulence
		end

		function setPowerLawWind(obj,windspeed,horAngle,vertAngle,shearExponent,referenceHeight)
			calllib('QBLIB', 'setPowerLawWind',windspeed,horAngle,vertAngle,shearExponent,referenceHeight);
		end

		function setDebugInfo(obj,isDebug)
			calllib('QBLIB', 'setDebugInfo', isDebug);
		end

		function setUseOpenCl(obj,isOpenCl)
			calllib('QBLIB', 'setUseOpenCl', isOpenCl);
		end

		function setGranularDebug(obj,dStr,dSim,dTurb,dCont,dSer)
			calllib('QBLIB', 'setGranularDebug',dStr,dSim,dTurb,dCont,dSer);
		end

		function setTimestepSize(obj,timestep)
			calllib('QBLIB', 'setTimestepSize', timestep);
		end
```

```
94
95                  function setRPMPrescribeType_at_num(obj,type,num)
96                          calllib('QBLIB', 'setRPMPrescribeType_at_num',type,num);
97                  end
98
99                  function setRampupTime(obj,time)
100                         calllib('QBLIB', 'setRampupTime',time);
101                 end
102
103                 function setInitialConditions_at_num(obj,yaw,pitch,azimuth,rpm,num)
104                         calllib('QBLIB', 'setInitialConditions_at_num',yaw,pitch,azimuth,rpm,num);
105                 end
106
107                 function setTurbinePosition_at_num(obj,x,y,z,xrot,yrot,zrot,num)
108                         calllib('QBLIB', 'setTurbinePosition_at_num',x,y,z,xrot,yrot,zrot,num);
109                 end
110
111                 function setControlVars_at_num(obj,vars,num)
112                         calllib('QBLIB', 'setControlVars_at_num',vars,num);
113                 end
114
115                 function setExternalAction(obj,action,id,val,pos,dir,isLocal,num)
116                         calllib('QBLIB', 'setExternalAction',action,id,val,pos,dir,isLocal,num);
117                 end
118
119                 function getWindspeed(obj,x,y,z,velocity)
120                         calllib('QBLIB', 'getWindspeed',x,y,z,velocity);
121                 end
122
123                 function getWindspeedArray(obj,posx,posy,posz,velx,vely,velz,arraySize)
124                         calllib('QBLIB', 'getWindspeedArray',posx,posy,posz,velx,vely,velz,arraySize);
125                 end
126
127                 function getTowerBottomLoads_at_num(obj,loads,num)
128                         calllib('QBLIB', 'getTowerBottomLoads_at_num',loads,num);
129                 end
130
131                 function getTurbineOperation_at_num(obj,vars,num)
132                         calllib('QBLIB', 'getTurbineOperation_at_num',vars,num);
133                 end
134
135                 function output = getCustomData_at_num(obj,var,i,j)
136                         output = calllib('QBLIB','getCustomData_at_num',var,i,j);
137                 end
138
139                 function output = getCustomSimulationData(obj,var)
140                         output = calllib('QBLIB','getCustomSimulationData',var);
141                 end
142
143         end
144 end
```

QBlade 2.0.7 beta

Structural Dynamics

- Added Timoshenko Beams
- Added Timoshenko FPM 6x6 Beams
- Added Anisotropic damping model
- Added nonlinear stiffness model for mooring lines (to model polyester and nylon ropes)
- Added calculation of blade mass inertia matrix to *Struct Model info* * Added strain to structural sensor outputs
- Added strain to structural sensor outputs
- Added function to automatically export 6x6 blade structural property tables from windIO yaml files
- Added NLDATA<X> tables to define nonlinear distributions (stress/strain, displacement/force, etc...)
- Removed first force value for y=0 from nonlinear spring/damper tables
- Distributed blade added masses now included in blade COG and inertia calculations in *Struct Model info*
- Structural blade and strut beams are now always placed at the user defined elastic center
- The reference position of the local section coordinate system is now the pitch axis
- Constraints are now defined in the coordinate system of Joint1ID, except for constraints to the fixed ground
- Improved modal analysis and Campbell diagram functionality (QB-EE only)
- Can now visualize real, imag, mag, and phase values after a modal analysis (QB-EE only)
- Fixed evaluation of global rotation displacement for reversed rotors
- Changed: now NACCM, NACINER and HUBINER can be directly specified within one line (6 entries for inertia, 3 for CM position)
- Old keywords NACCMX, NACCMZ, NACYINER are still accepted, but should be replaced eventually
- Fixed issue where external loading data was not assigned for substructure joints
- Fixed issue where external loading was not correctly assigned when a substructure only was simulated

Aerodynamics

- Added Dynamic Wake Meandering (DWM) model (mash up between Fast.Farm and HAWC2Farm models)
- Added axial, tangential, and radial aerodynamic force to *Structural Time Graphs*
- Added adaptive relaxation to speed up gamma bound fixed point iteration

- Added *Aerodynamic Strut Graph* type to visualize distribution of aerodynamic variables over struts (VAWT only)
- Finished first working version of WindIO YAML import for airfoil, polars, and blades
- Fixed vortex line core growth model, now correctly working with time offset
- Fixed UBEM to work properly for reversed rotors
- Fixed issue with the reprojection of the moment coefficient before applying NC, attached flow, or DS corrections
- Fixed behavior of the tower shadow model in certain conditions

Hydrodynamics

- Added functionality for one-sided Morison drag for improved low-frequency excitation predictions
- Added functionality to assign frequency-filtered axial drag coefficients for improved low-frequency excitation predictions
- Added NBODY>1 feature of WAMIT to substructure definitions for multiple interacting hydro bodies
- Added tangential cable drag coefficient, as optional column in HYDROMEMBERCOEFF tables
- Added options to deactivate individual QTF DOF's (DIFF_INACTIVE_DOF, SUM_INACTIVE_DOF)
- Changed the sign for the definition of the misalignement between wind and waves, when specified via a DLC table
- Fixed issue in the mooring system where buoyancy wasn't accounted for
- Fixed issue with turbine global position and water depth for offshore bottom-fixed turbines
- Fixed issue with read-in functions for RAD and EXC potflow files (incompatible with Windcrete dataset)

Wave Generation

- Added nonlinear streamfunction waves and constrained streamfunction waves
- Added functionality to paste streamfunction waves over constrained newWaes
- Added display variable at instantaneous sea level (ISL)
- Changed regular linear waves to now start with a phase shift of 90°
- Changed random number generation to be consistent across different platforms, e.g. Unix/Windows, using Mersenne Twister engine

Wind Generation

- Added Mann turbulence generator
- Added Mann generator option to create an "added turbulence" windfield for DWM m
- Added option to create and export windfields from simulations
- Improved behavior of WindFieldGenerator to better conform to IEC standards

- Changed random number generation to be consistent across different platforms, e.g. Unix/Windows, using Mersenne Twister engine

Control Systems

- Updated DTU controller to employ original source code and location of the wpData file(s)
- Changed CpCtCq file of ROSCO controller to be defined relative to the controller library
- Fixed issue regarding an incompatibility with older ROSCO controller versions

General Improvements

- Improved computational speed for QBlade-CE (~2-3x)
- Improved stability and numerous bug fixes
- Finished import/export functionality for wind farm layouts (QB-EE only)
- In DLC definition files a cell (or entry) beginning with a hashtag (#) is ignored now (QB-EE only)
- Added new interpolation functions HERMITE, C2 splines and Akima (for structural properties and blade geometry)
- Added functionality to specify the constrained DoF's between struts and blades and struts and the torquetube (VAWT)
- Added visualization of IEA reference coordinate systems to QSimulation module
- Polar Cl and Cn slopes are now obtained via linear regression
- Added libxlnt to read data directly from excel documents
- Added functionality so that "submerged" rotors in an "offshore" simulation get the water velocity, density, and viscosity as inputs for "aerodynamic" load calculations, this enables combined wind-hydro turbine simulations
- Fixed issue in the DLL interface where advanceStructure() and AdvanceAero() were called in the wrong order
- Fixed issue in cut-plane import/export related to timesteps when QSim had "store from time" enabled
- Fixed issue where STORE_SIM was not recognized during ASCII simulation import

QBlade 2.0.6.3 beta

- fixed issue where for certains node/member arrangements a substructure could be over-constrained
- fixed issue where for a substructure only turbine definition NaN inertia values were shown in model overview
- fixed automatic scene scaling for displaced substructure-only configurations
- orientations of subjoints and the transition piece can now also be defined by means of consecutive Euler rotations
- optimizations of OpenCl particle kernels (QBlade-EE)
- fixed crash of modal analysis of substructure-only configurations (QBlade-EE)

- fixed issue where the multi-turbine (.mta) ASCII import could crash if some files were missing (QBlade-EE)

QBlade 2.0.6.2 beta

- added total Morison force to output for substructure members in hydrodynamic time graphs
- fixed some small issues with the openCl device selection when not in the GUI mode (CLI or SIL)
- fixed normal calculation for .stl blade export at blade hub and tip end faces
- added an optional export filter to rearrange/filter the export files that are generated
- added a getCustomSimulationData function to the SIL interface
- substructure constraint loads can now be added to the graphs by specifying the constraint with the identifier “CST”
- added export filter feature to CLI
- fixed issue where CLI was searching for a “wrong” binary file name

QBlade 2.0.6.1 beta

- fixed typos
- in BEM/DMS analysis power is now always displayed as kW
- fixed issue with turbine sorting for multi turbine simulations

QBlade 2.0.6 beta

Substructure

- added MOORLOADS to add buoyancy forces and loads to mooring lines
- ADDMASS functionality was extended so that also diagonal inertia 3xIXX & offdiagonal 3xIXY (in local body coords) can be added to a structure
- added RECTANGULAR members to the substructure definition, these members can use directional hydro coefficients for the local x and y directions
- added a new and more consistent table format for MOORELEMENTS and CABLELEMENTS that uses only 6 instead of 7 entries
- added substructure elements, constraints and spings/dampers to global mooring systems

Control / SIL

- added the functionality to “wire” custom external library and controller data channels from the swap arrays to turbine actions
- added interface for external controller libraries

- added function SetExternalAction() to DLL interface to enable highly customized simulations / controllers
- added visualization of hydrodynamic Morison forces
- when simulating with the ROSCO controller the “PerfFileName” is now also serialized in the same way as the WPData file for the DTU controller

Wave Generation

- added wave probes to the wave module
- added constrained embedded wave feature to linear waves
- added option to merge linear waves

Misc

- turbines (and floater, mooring systems) can now be globally rotated
- implemented the IAG dynamic stall model
- added Cn and Ct graphs to polars and 360 polars
- added feature to assign distributed damping properties for the blade, towers and struts
- added feature to account for nacelle drag

QBlade-EE

- added aerodynamic damping calculation and damped frequencies to modal analysis (QB-EE only)
- implemented linearization of buoyancy for modal analysis (QB-EE only)
- OpenCl: improved kernel performance by ~20%

Furthermore, a lot of features and fixes based on community feedback have been implemented in various modules and the overall stability and robustness has been improved.

QBlade 2.0.5.2 alpha

- public release of QBlade’s Software in Loop (SIL) interface
- project files now 100% compatible between EE and CE versions (dedicated EE format is omitted)
- catching a possible crash that could occur when pressing the “Mode Animation” botton before the simulation has finished
- fixed issue where an impulsive aerodynamic load was present at the first timestep due to an error in relative velocity initialization
- fixed an issue in shared mooring systems
- in substructure files TP_INTERFACE_POS_1 can now also be used instead of TP_INTERFACE_POS (which was omitting the _1 part)

QBlade 2.0.5.1 alpha

- LLFVW: bound circulation core size is now defined based on aerodynamic panel width instead of chord
- fixed a small issue in wake coarsening where the redistribution of shed vorticity could lead to a small induction “jump”
- default blade discretization type is now COSINE
- improved evaluation of 360° polar parameters, such as slope, aoa_0, Cl_0 etc...
- OYE and ATEFlap DS models are now limited to the range of +-50° AoA
- fixed major issue in implementation of nonlinear p-y curves
- from now on the version stringis part of the binary files to better distinguish them
- improved custom data aquisition for python and matlab (SIL) interfaces

QBlade 2.0.5 alpha

- added blade force in X_c and X_b coordinate systems to the standard outputs
- added functionality to the assignement of nonlinear springs and dampers for substructures
- overhauled read-in functionality for WAMIT diffraction and excitation files
- directionality for 2nd order difference loads now taken into account
- arbitrary orientations can now be assigned to substructure nodes, substructure node coordinate systems can now be displayed
- added option to read-in WAMIT .8 files
- added correct Reynolds number to steady BEM outputs
- fixed issue where TSR string was set to zero after creating a bladeLoading definition
- fixed bug that corrupted project files after a polar was edited with “Edit Current Polar Points” and then discarded
- added optional generator efficiency
- fixed initial camera view angles for QTurbine and QSimulation modules
- renamed StrModel variables for aerodynamic and generator power and torque
- improved import/export functionality of velocity cut-plane definitions
- fixed broken link to forum
- added controller SWAP array to getCustomData() function of the DLL interface

QBlade 2.0.4.9 alpha

- added CPmin variable to results of the XFoil polar analysis, corrected evaluation of friction drag coefficient from XFoil
- bugfix: overhauled interface with Xfoil binary which is now working with absolute ins relative path names
- bugfix: fixed a crash that occurred when a TDMS object was deleted in the GUI

- added blade root forces to default sensors
- added FAST binary format to the avaliable export formats for simulation timeseries
- fixed issue where when using hubheight inflow files the horizontal inflow angle was not read in properly
- changed the sign in the definition of the horizontal inflow angle to be in line with the most common convention
- bugfix: prevent UBEM crashes that occurred at inflow velocity of zero
- tower bodies, torquetube bodies & strut bodies can now have buoyancy & addedmass & dynamic pressure coefficients assigned to model hydrokinetic turbines. model hydrokinetic turbines as onshore turbines with changed air density

QBlade 2.0.4.8 alpha

- chord can now be optimized idependent of twist
- optimize PROP dialog now hidden during HAWT blade design
- displaced water volume added to hydrodynamic variables
- when a simulation is diverging the last 3 timesteps are removed from the data to prevent NaN in data
- added yaw event to turbine events

QBlade 2.0.4.7 alpha

- default sensors added for tower top and nacelle (velocity, acceleration, deflection)
- fixed issue in DS models that could occur when “bad quality” polar data (such as with negative slope) was used
- removed structural time integrator selection from SimulationCreatorDialog, HHT is now default
- fixed issue where the tower drag coefficient was not read from the structural data table
- fixed issue with the tower shadow model, the position of tower shadow is now the instantaneous position of floating turbine
- added info for RNA and Tower COG to turbine design module, inertia info displayed now around the global COG
- when importing TurbSim .inp files the TurbSim console output is now displayed
- added delete by selection for turbine objects
- graph data can now be directly copied to clipboard
- several small gui improvements

QBlade 2.0.4.6 alpha

- fixed error where the current yaw angle read from the structural model had the wrong sign

- fixed error when during import of linear waves from a time series the mean heading angle was read in radians and not degrees
- added yaw angle to structural outputs
- added ROSCO 2.4.1 controller library

QBlade 2.0.4.5 alpha

- Implemented various checks to prevent users from defining overconstrained nodes during substructure generation that could cause divergence in the structural solver; checking SUBELEMENTSRIGID and SUBCONSTRAINT data tables

QBlade 2.0.4.4 alpha

- Bugfix in steady state BEM for HAWT's

QBlade 2.0.4.3 alpha

- Fixed an issue in the classical steady state BEM iteration that appeared at large (above optimum) TSR's.

QBlade 2.0.4.2 alpha

- There were issues with the OpenCL.dll under Windows, this dll has been replaced with a more compatible version that should detect OpenCL for most users

QBlade 2.0.4.1 alpha

- Fixed issue with virtual camber transformation, where values were not read from dialog
- Improved behavior of FoilTable when Foil selection is changed

QBlade 2.0.4 alpha

- This is the first public release of QBlade CE. Be aware that this is an alpha release which will be revisioned after the first user feedback arrives and incompatibilities and errors are fixed.

Validation of the ATEFlap Dynamic Stall Model

Two exemplary validation graphs for the ATEFlap Model from the publication by Wendler et al. [1] are shown in Fig. 155, where the general sensitivity of the dynamic stall hysteresis loop to the reduced frequency and amplitude is well reproduced.

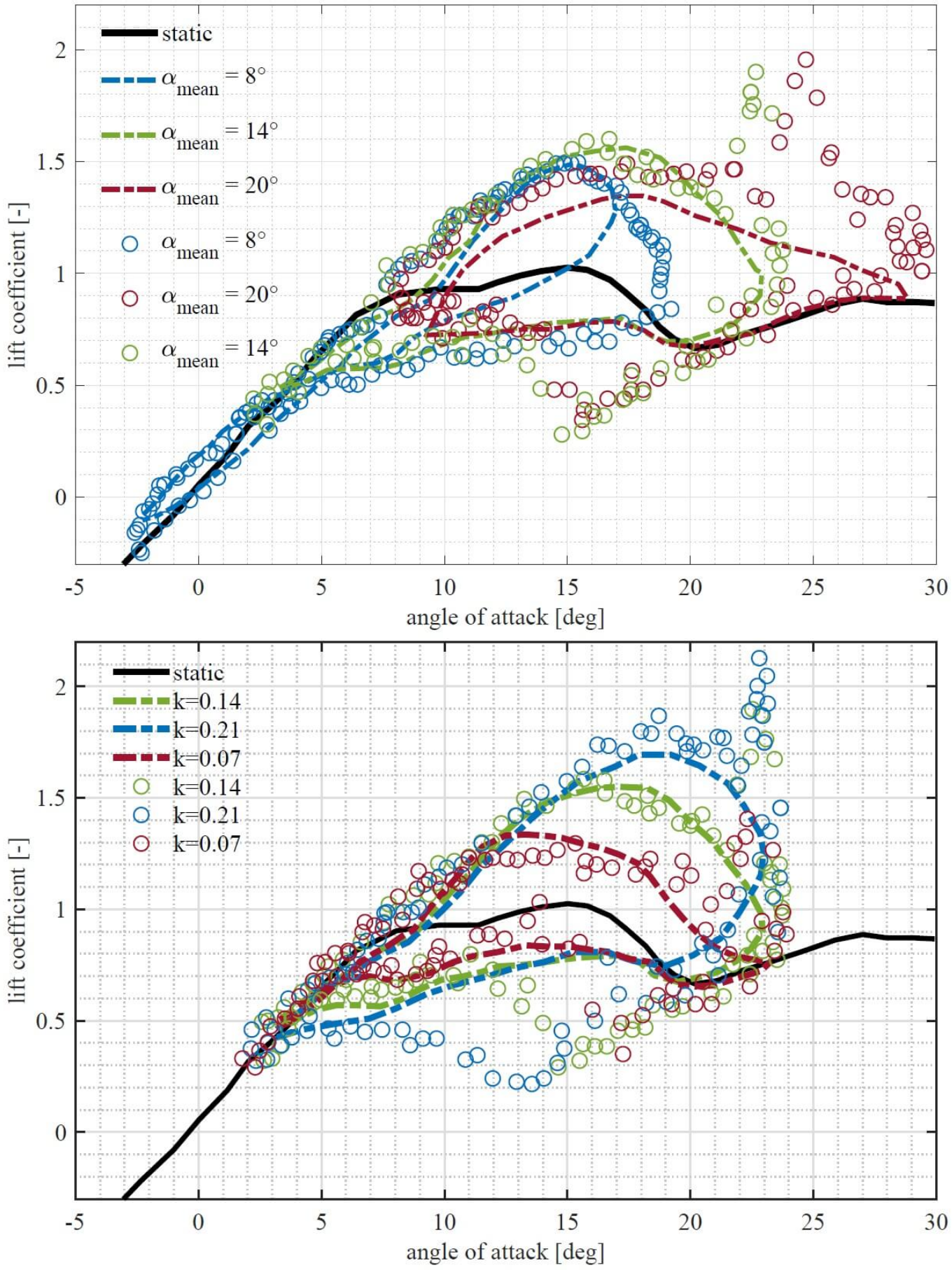


Fig. 155 *Validation of the unsteady aerodynamics model with OSU test data [2] (in circles) of th airfoil; top: varying mean AoA; bottom: varying dimensionless frequency.*

[1] Juliane Wendler, David Marten, George Pechlivanoglou, Christian Oliver Paschereit, and Christian Navid Nayeri. An Unsteady Aerodynamics Model for Lifting Line Free Vortex Wake Simulations of HAWT and VAWT in QBlade. In *Proceedings of ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition*. 2016.

[2] NREL. OSU Wind-Tunnel Test Data S809. 1999. URL: https://wind.nrel.gov/airfoils/OSU_data/data/ (visited on 2019-07-12).

Validation Tests for Potential Flow Models with Morison Drag (LPMD)

The linear potential flow model in QBlade (QB) (see Linear Potential Flow Theory) was validated in a series of test load cases using two different turbine models: a spar buoy and a semisubmersible. The linear potential flow model was enhanced with quadratic drag forces via the Morison equation (see Morison Equation) and is termed LPMD in this document. The load cases were chosen with increasing complexity to make sure the individual modules were working correctly.

The results were validated against the open-source aero-hydro-elastic code OpenFAST (OF) [1] (version 2.5.0). The same turbine models and test cases were setup and used in QB and OF. Not all of the hydroelastic modelling capabilities of QB are present in OF. When validating certain hydrodynamic models, the hydrodynamic capabilities of QB were adapted to reflect those of OF to have a better comparison. These modifications are mentioned in the appropriate load cases below.

In the following sections, the results for each model are presented.

OC3 Spar Buoy LPMD Model

The first model, named OC3 model in this document, is the floating 5 MW wind turbine mounted on a spar buoy from the OC3 Report [2]. This model was considered so that we have a simple geometry to test the different hydrodynamic models. Fig. 156 shows the OC3 LMPD model within the QB GUI. The substructure was considered rigid in this model to evaluate only the hydrodynamic modules. For this model, the mooring system was modeled explicitly and the localized buoyancy model was used.

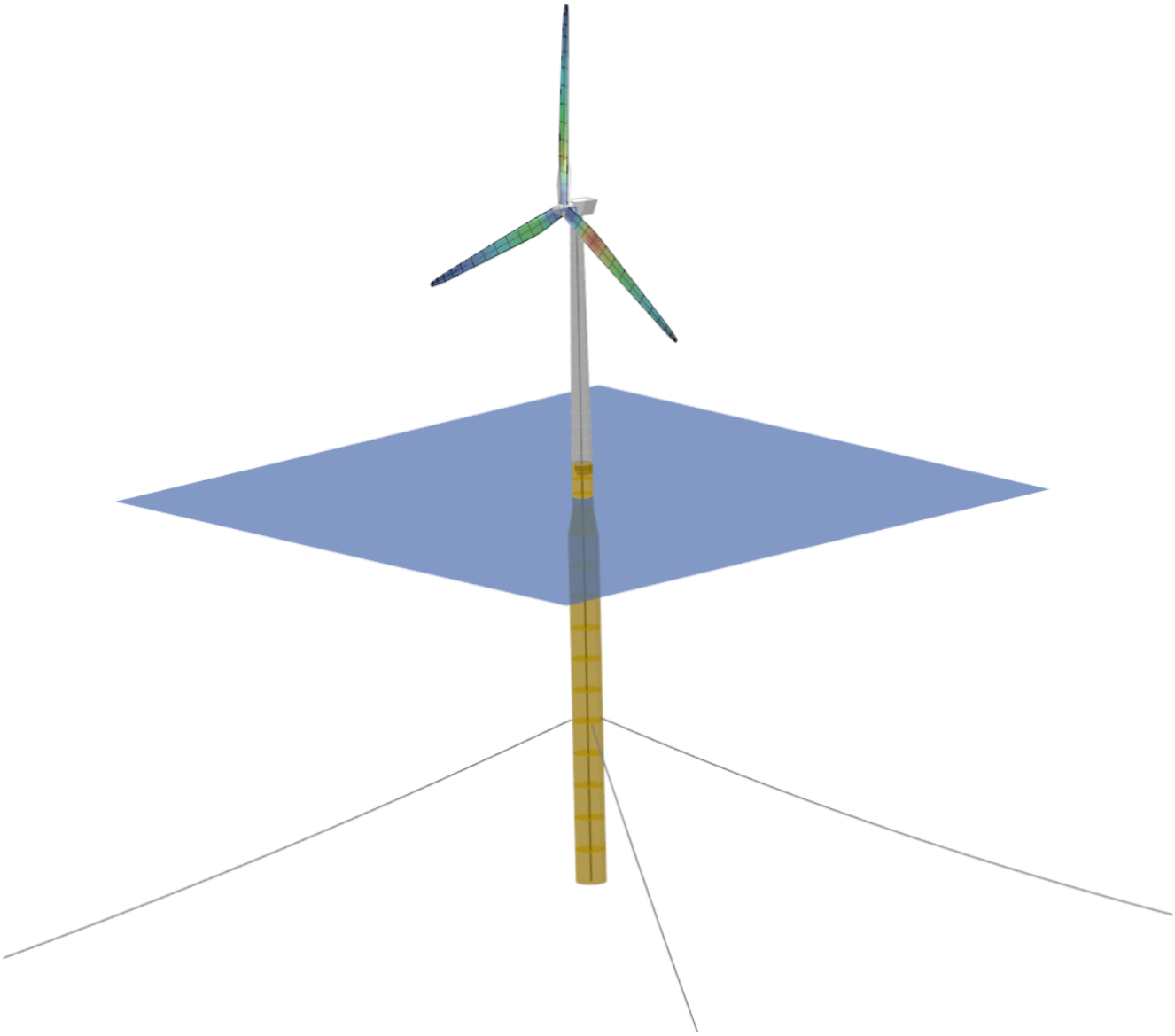


Fig. 156 *The OC3 model displayed in the QB GUI featuring a spar buoy substructure.*

OC3 LPMD No Wave Tests

The no wave tests are split into two parts, the first and main part consists in free decay tests in still water. The second part considers the turbine reaction to no wave and constant current conditions. We will describe first the results from the decay tests and later present the results for the constant current tests.

OC3 LPMD Free Decay Tests

The free decay tests simulate the turbine model in free decay from an initial position in still water condition. No aerodynamic loads were considered in the decay tests. All the 6 degrees of freedom (DOFs) are considered for in the decay tests. The evaluation of the results is done 1) visually, by inspecting the time series of all isochronic DOFs and 2) quantitatively, by analyzing the eigenfrequency and damping of the disturbed DOF. The latter analysis was done according to the procedure presented in [3].

Fig. 157 to Fig. 159 show the time series for the decay tests of the surge, heave and pitch DOFs. From these figures we can see that the predictions from QB and OF are practically identical. This is not only the case for the initially disturbed DOF but for the other DOFs as well. In the surge and pitch decay tests, there is a small offset between the heave time series. This offset comes from the different way the buoyancy is calculated between OF and QB. In OF, the buoyancy is accounted for by linearized stiffness matrices. In QB, the buoyancy forces is taken are calculated explicitly by taking into account the locally displaced water volume.

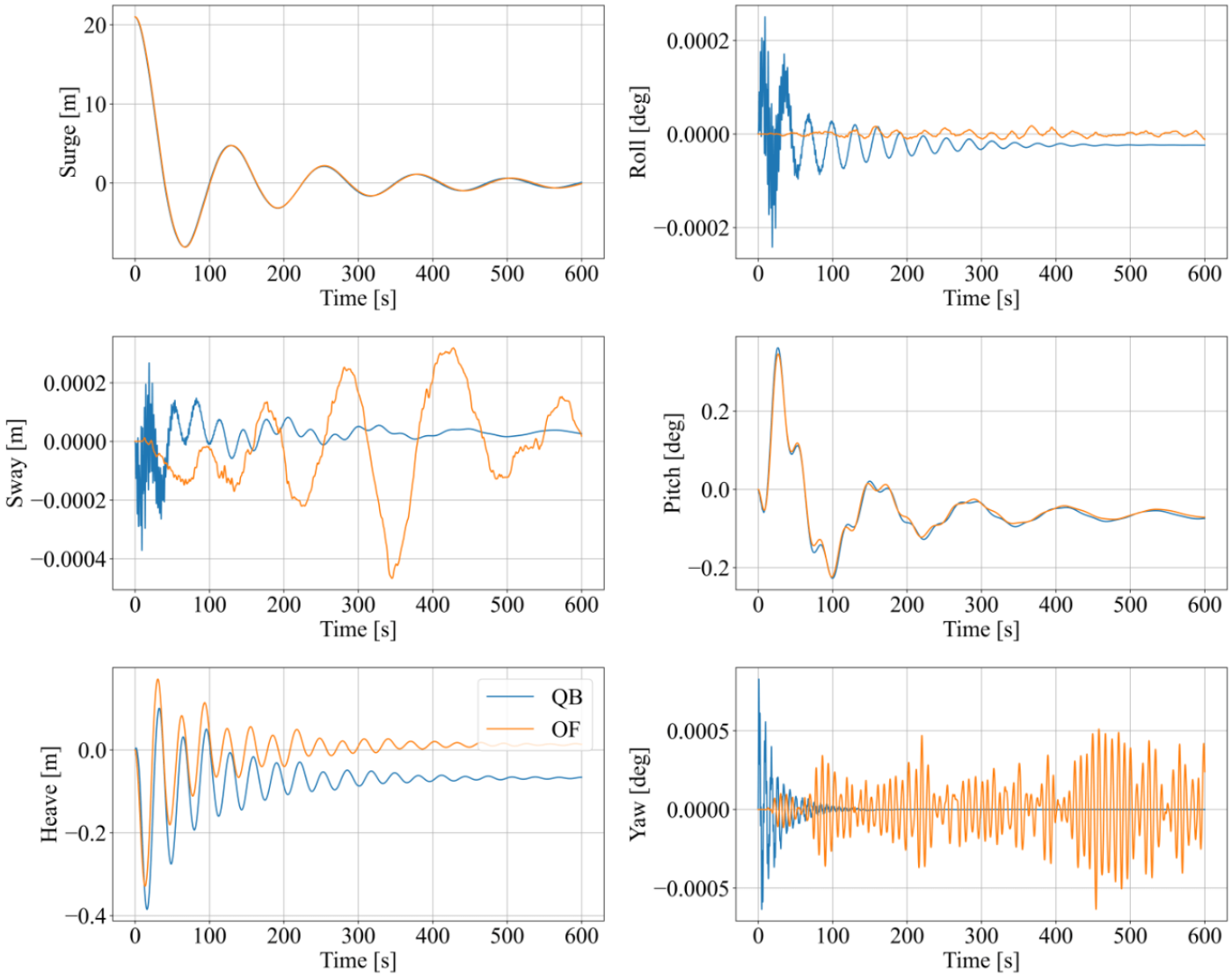


Fig. 157 *Time series of the OC3 surge decay test.*

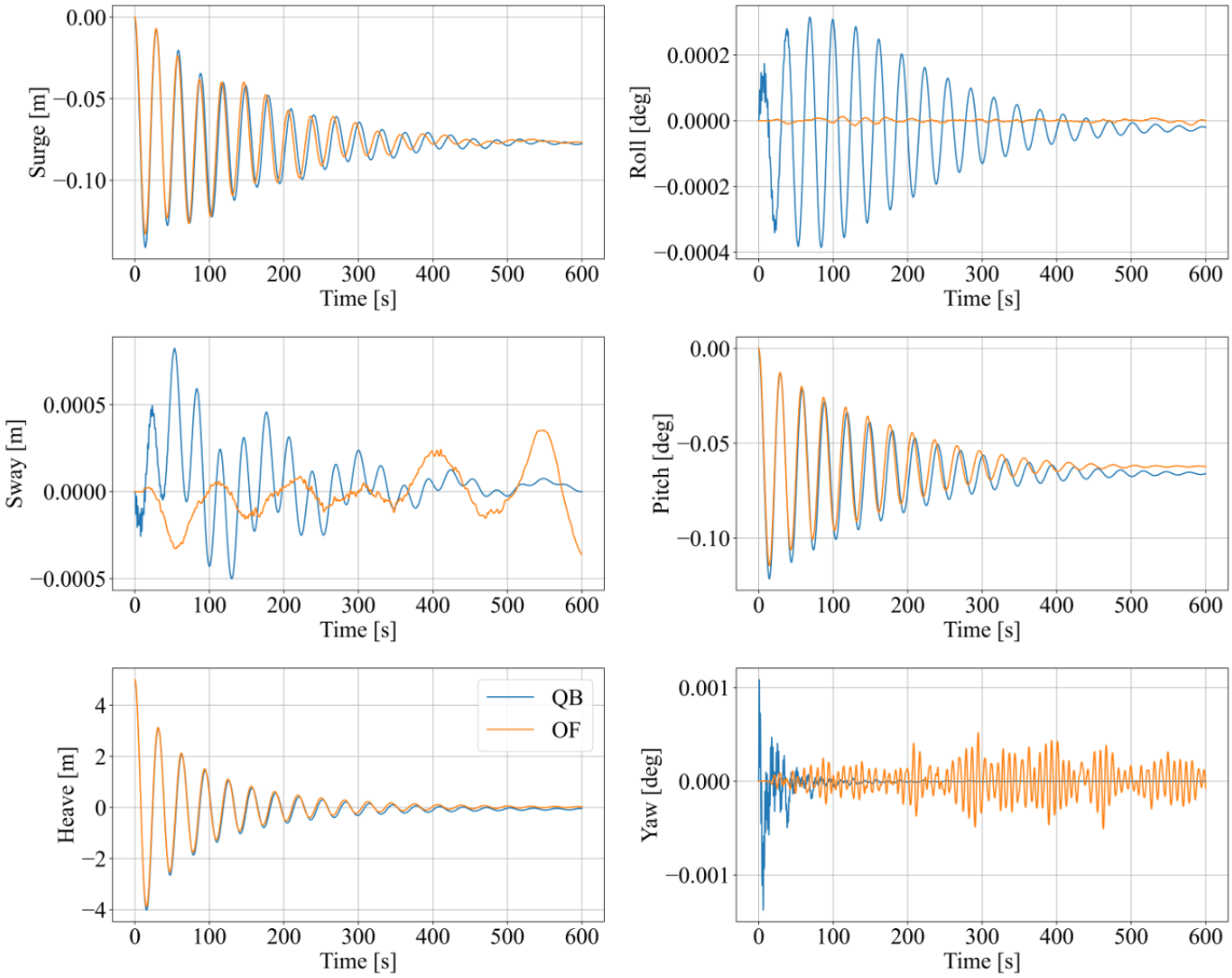


Fig. 158 *Time series of the OC3 heave decay test.*

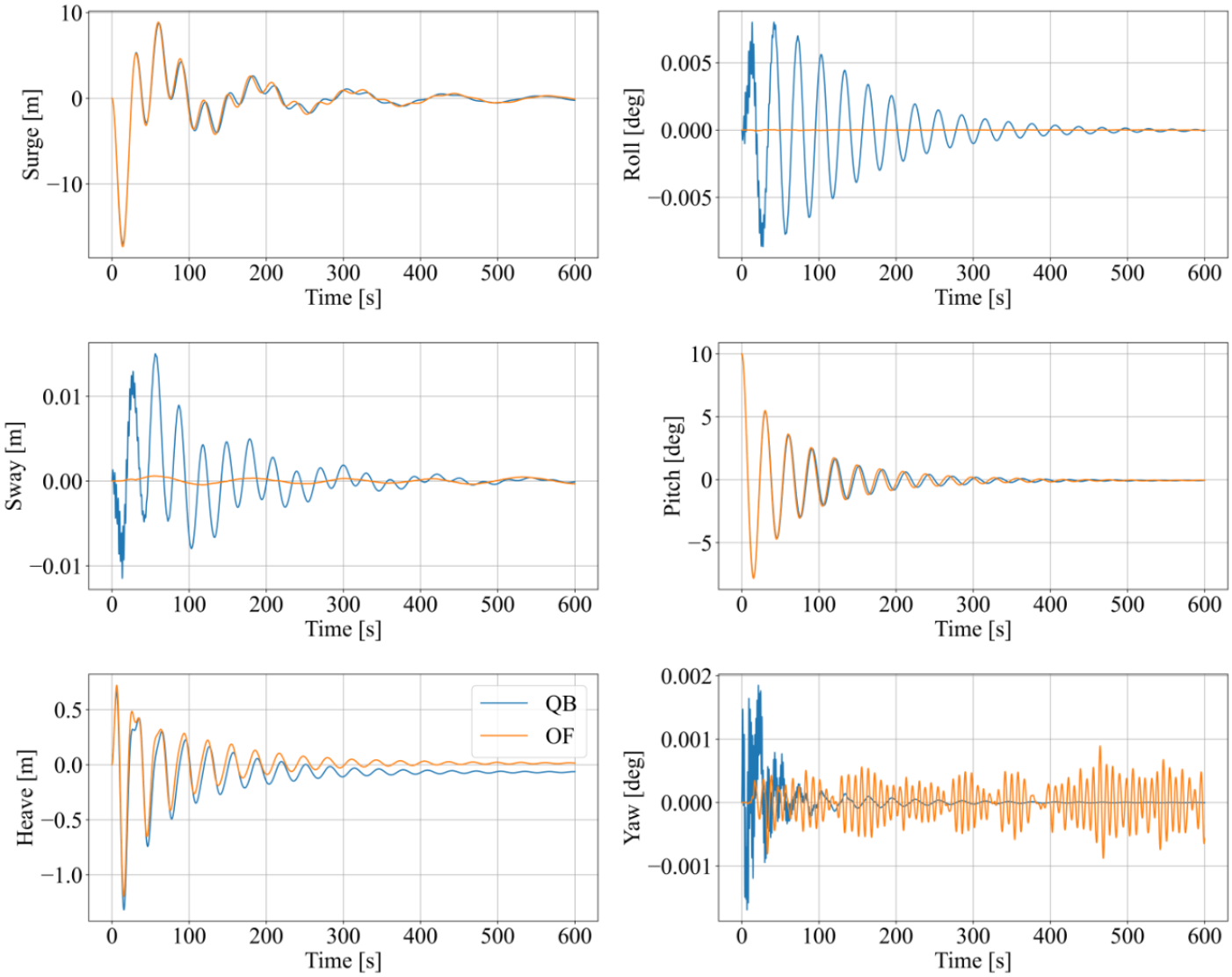


Fig. 159 *Time series of the OC3 pitch decay test.*

The decay tests for the other three DOFs (sway, roll, yaw) were also simulated. The time series are not included in this document since the findings are very similar.

Fig. 160 to Fig. 162 show the corresponding tensions at the fairlead and anchor locations of the mooring systems for the surge, heave and pitch decay tests. We can see good agreement between both codes. Again, for the heave decay tests, there is a small offset in the tensions between the QB and OF simulations. It can be explained by the small offset in the heave neutral position between both codes and the different approaches used in both codes to model the mooring system.

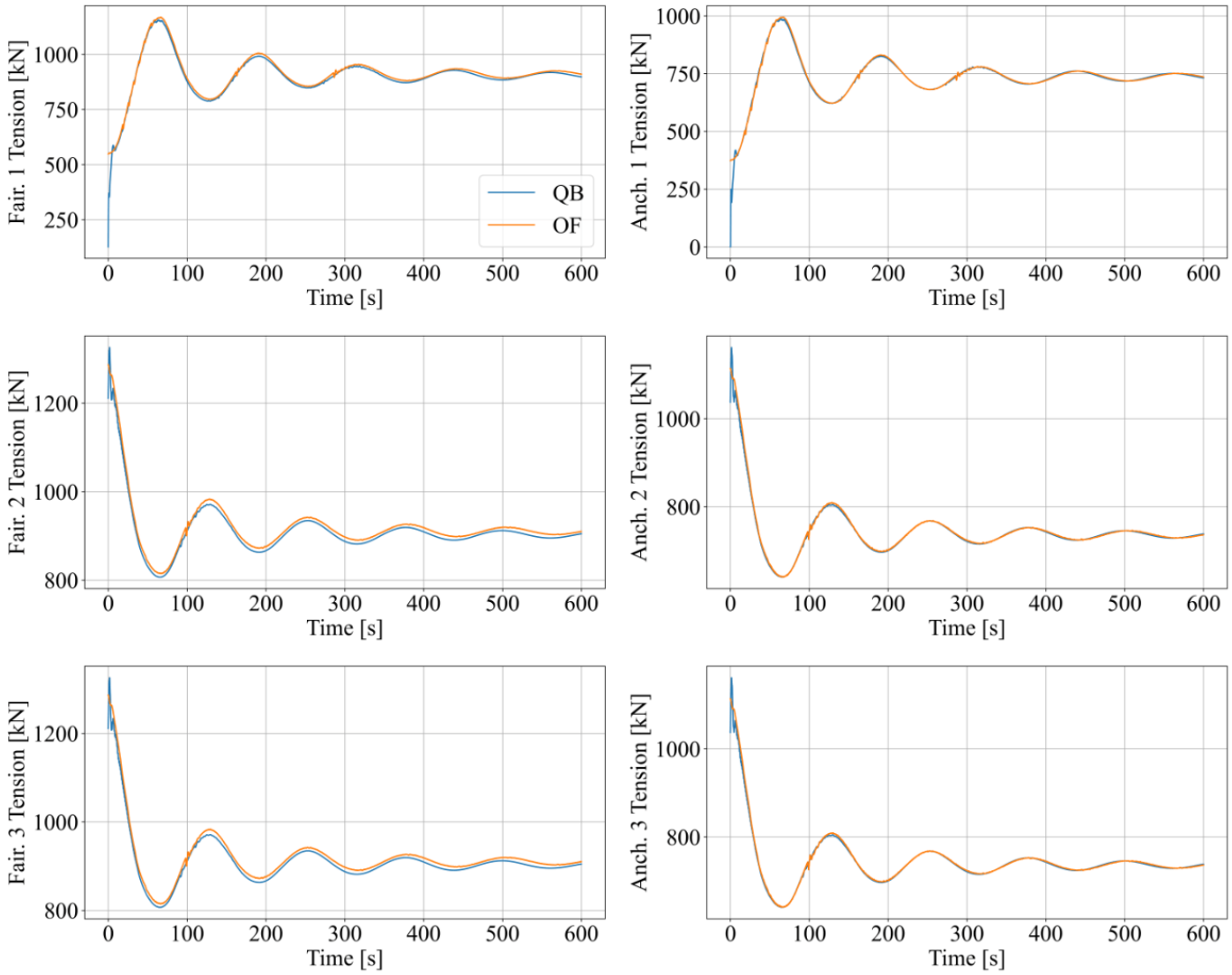


Fig. 160 *Mooring line tensions for the OC3 surge decay test.*

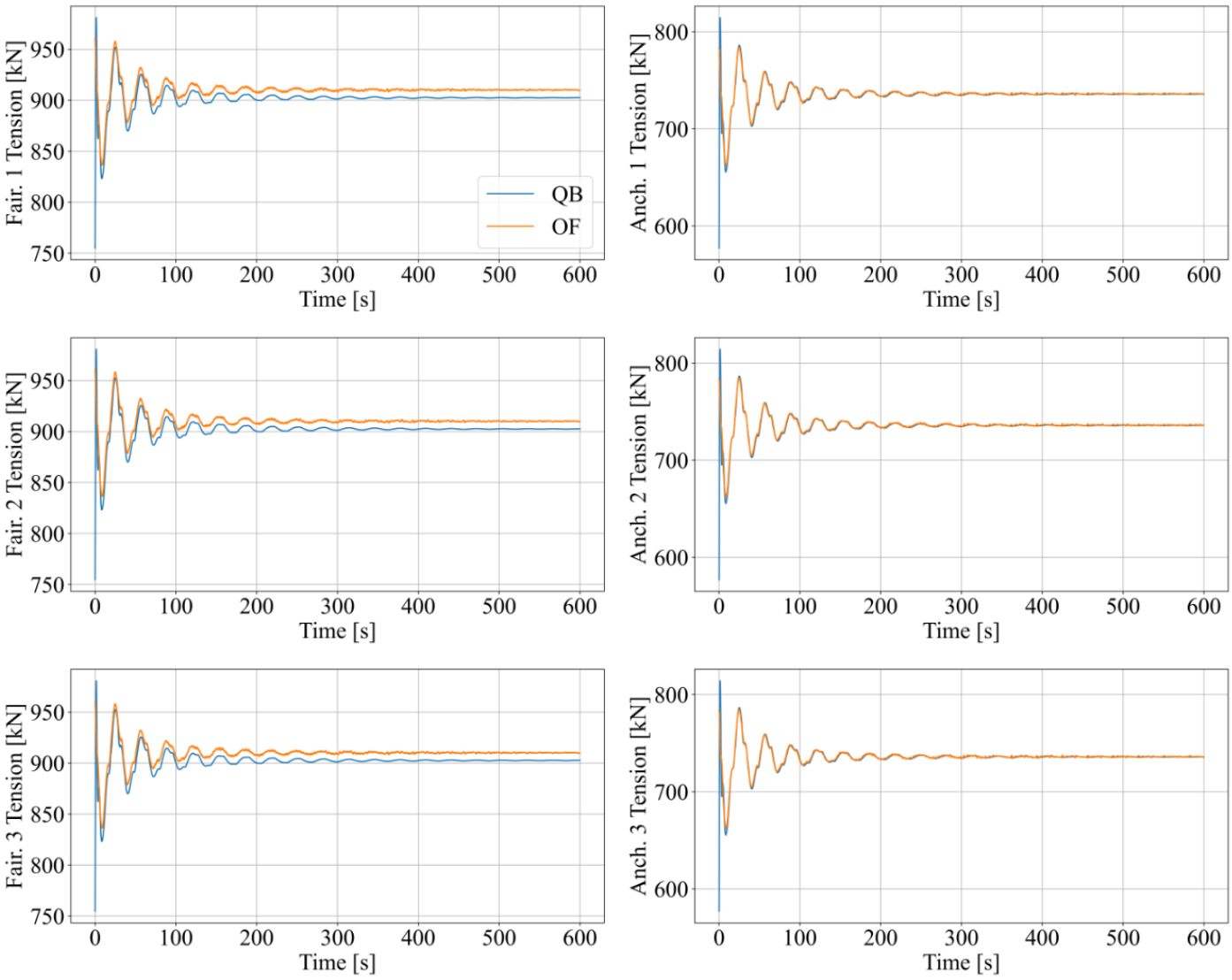


Fig. 161 *Mooring line tensions for the OC3 heave decay test.*

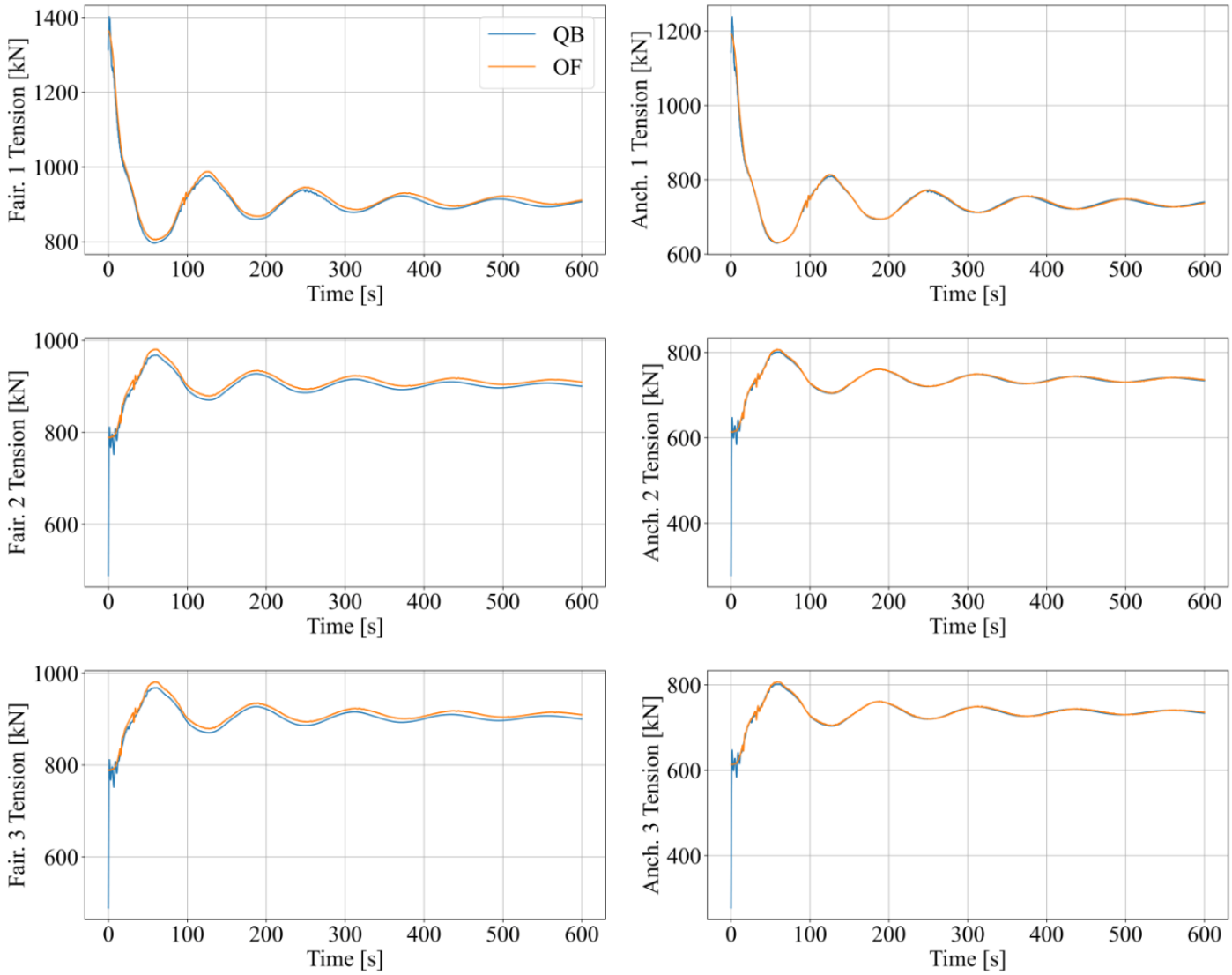


Fig. 162 *Mooring line tensions for the OC3 pitch decay test.*

The decay tests were also analyzed quantitatively by comparing the eigenfrequencies and damping characteristics of the floater for the initially displaced DOFs. Fig. 163 shows the results for the six performed decay tests. We can see in this figure that the normalized eigenfrequencies agree very well between both codes. There are significant differences in the damping characteristics, especially for the surge, sway, heave and yaw DOF.

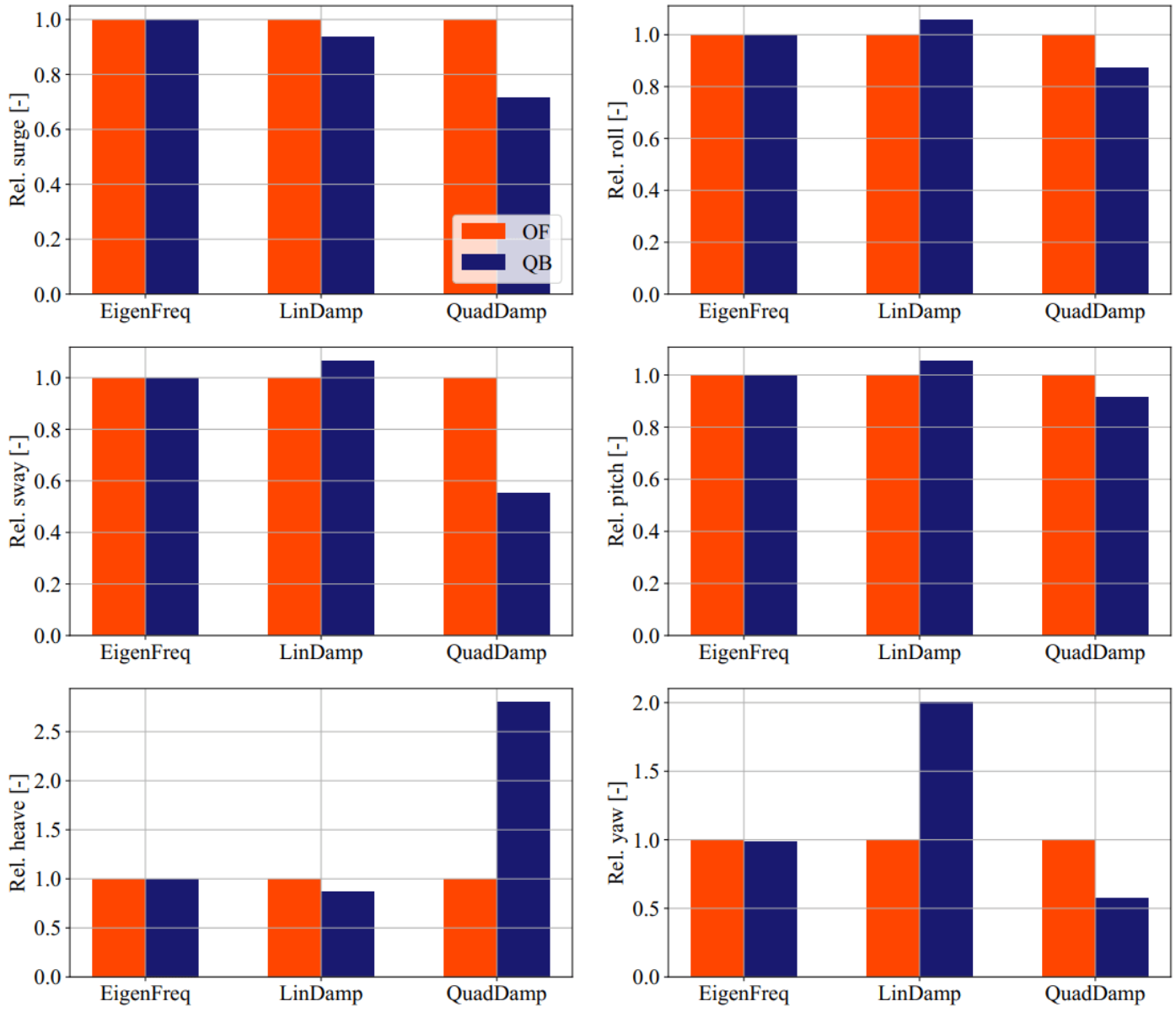


Fig. 163 *Normalized eigenfrequencies and damping behaviour of the OC3 model for the considered decay tests.*

The differences are assumed to come from the different mooring system models. OF uses MoorDyn, which uses a lumped-mass formulation for modelling axial elasticity [4]. QB uses a cable structural formulation (see Multi Body Beam Formulation) coupled with the Morison equation (see Morison Equation) to account for the hydrodynamic forces on the mooring system. In order to test if the mooring system formulation causes the different damping behavior, we replaced the mooring system models with linear stiffness matrices in both codes. Since there is a significant linear damping term in the hydrodynamic matrices in the surge, sway, heave and yaw DOFs, a linear damping relation was assumed for the evaluation of the time series.

Fig. 164 shows the normalized eigenfrequencies and linear damping characteristics for the models with linearized mooring systems. We can see now that the values for all DOFs except the ya align. For this turbine model the yaw DOF does not couple with the other DOFs and is almost exclusively determined by the stiffness, damping and inertia matrices, since the quadratic damping

term of the Morison equation is not applied to cylinder rotations. So the decay behavior can be calculated analytically. Using the stiffness, inertia and damping values and adding the rotational inertia of the turbine, an analytic eigenfrequency of 0.118 Hz was determined. This value is only 3% off from the values gotten in the QB and OF calculations. The analytical damping ratio is 4.4%. This value differs from the one obtained in QB simulations by 0.5% and from the one by OF simulations by 17%. It is therefore assumed that the value obtained by QB simulations is the correct one.

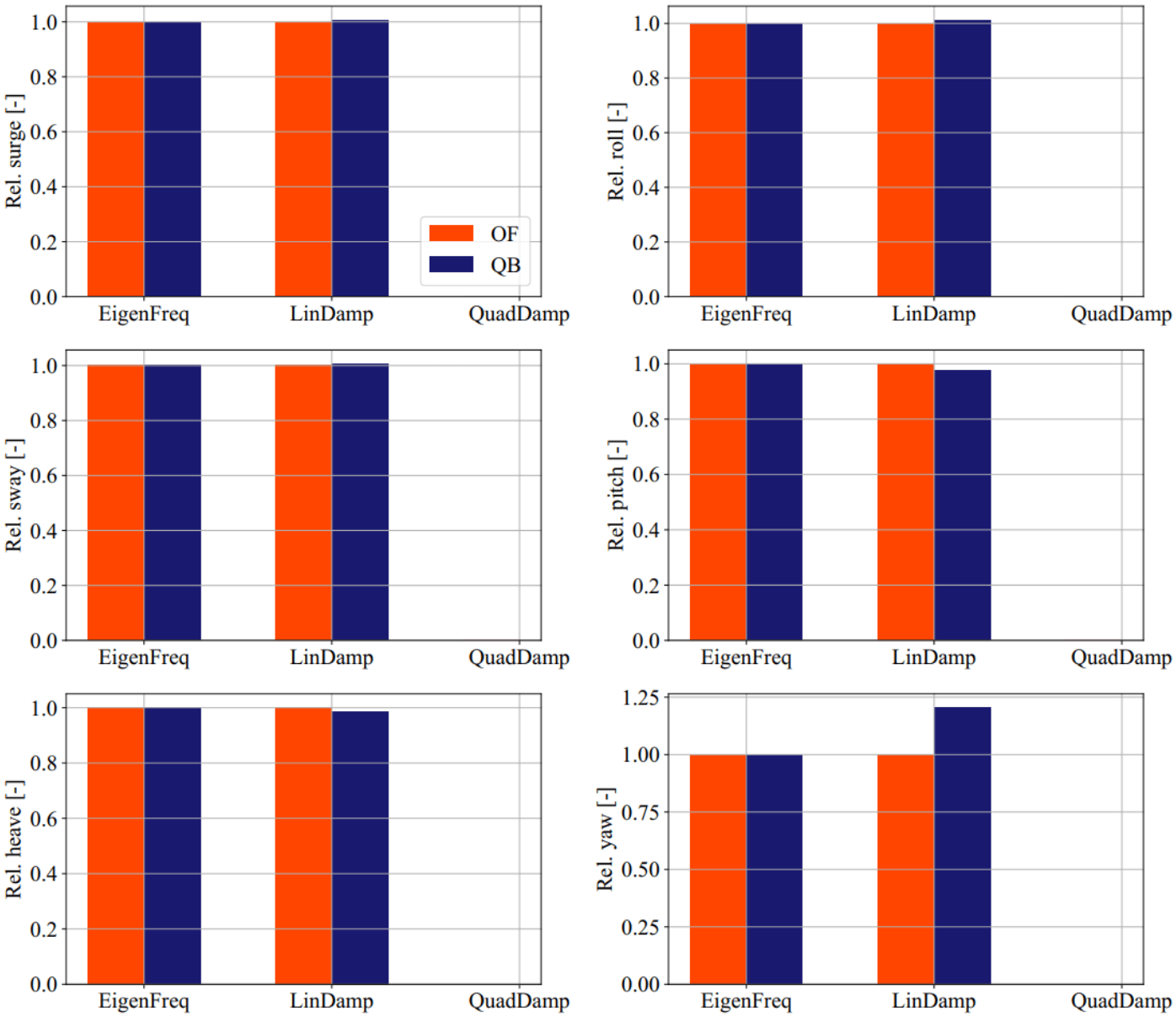


Fig. 164 *Normalized eigenfrequencies and damping behaviour of the OC3 model with linear mooring for the considered decay tests.*

OC3 LPMD Current-Only Tests

The second part of the no wave tests comprised constant current tests. For these tests, two current profiles were selected according to [5]: a power-law profile – representing a tidal current velocity – and a linear profile – representing a wind-generated current (see Currents). The current direction is aligned with the positive surge direction. The turbine was initially in its original undisplaced position

and the simulations included the transient response to the current profiles. No aerodynamic loads were applied in this case.

Fig. 165 shows the time series of the wind-generated current test with a surface current magnitude of 3 m/s and decrease rate of 0.2 1/s. We can observe in this figure that the turbine behaves almost identically for the affected DOFs (surge and pitch) when simulated with QB and OF. The different behavior in the heave DOF can be attributed to the different ways the buoyancy is modelled in QB and OF. The right column plots of Figure 25 show the water particle velocities at three different locations along the turbine substructure: at 0 m, at -10 m and at -20 m. We can see that the water velocities do not fully match. The reason for this is twofold. Firstly, the water particle velocities from QB are shown in the moving coordinate system of the substructure while in OF a fixed set of positions is used for the water particle velocity output. Secondly, OF interpolates the water particle velocities to the output nodes while QB shows the velocities for the center of the member that is closest to the selected output location. In the case of Fig. 165, the center member location and the velocity output location was slightly different.

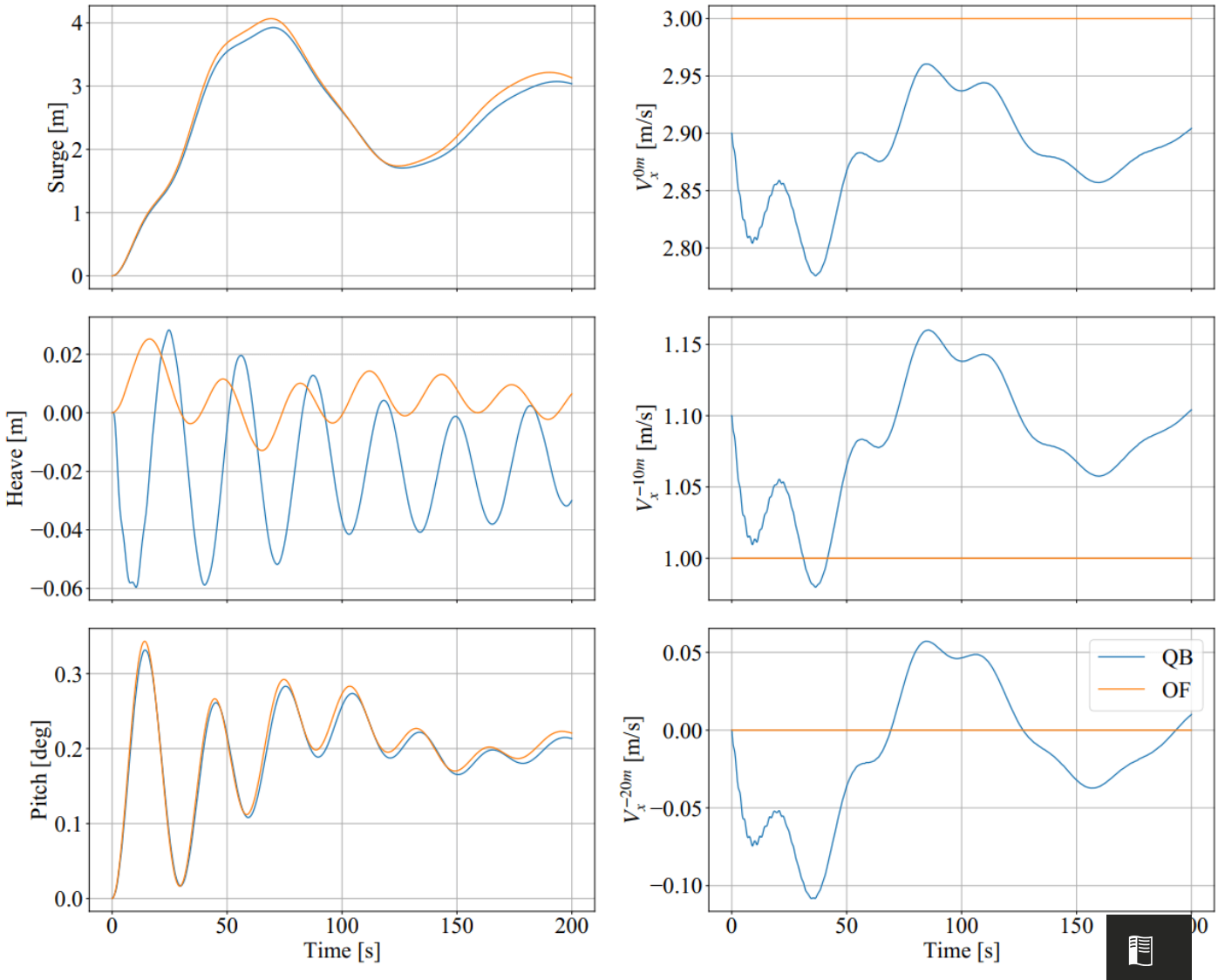


Fig. 165 *Displacements and water velocities for the OC3 model in a wind-generated current test case.*

OC3 LPMD Regular Wave Cases

The regular wave cases aim to validate the implementation of the first-order wave excitation forces in QB. This done in load cases where a single wave train with varying amplitude, period and direction is used to model the sea state. In order single out the effect of the excitation force implementation, the respective models in both simulation tools are set up to be as similar as possible. Therefore, the mooring system and the buoyancy are modeled with a linearized stiffness matrix. Both tools make use of the same excitation force impulse response function (IRF) computed in WAMIT [6]. No wave stretching model was used in QB so that the modelling considerations between QB and OF were as close as possible (see Kinematic Stretching). OF does currently not allow wave stretching models to be implemented in HydroDyn [7]. Similar to the free decay tests, no aerodynamic loads were considered. The validation is done by analyzing the time series and frequency spectra of the excited OC3 model.

Fig. 166 and Fig. 167 show the time series and corresponding spectra of the three DOFs (surge, heave, pitch) excited by an incoming wave from 0° direction for two regular wave cases. The first case has a wave height of 6 m and a period of 10 s and the second case has a wave height of 8 m and a period of 12 s. In both cases, an initial transient is present and completely dies out after approximately 300s. This explains the additional peaks at lower frequencies than the wave frequency in the respective spectral plots. Afterwards, a constant frequency excitation by the linear wave is present in all DOFs. It can be noted that the translations and rotations of the floater are predicted in an identical manner between both tools. Hence, a correct implementation of the first-order wave excitation loads may be concluded. It can be further noted that changing the wave height and period does not induce any differences between both tools.

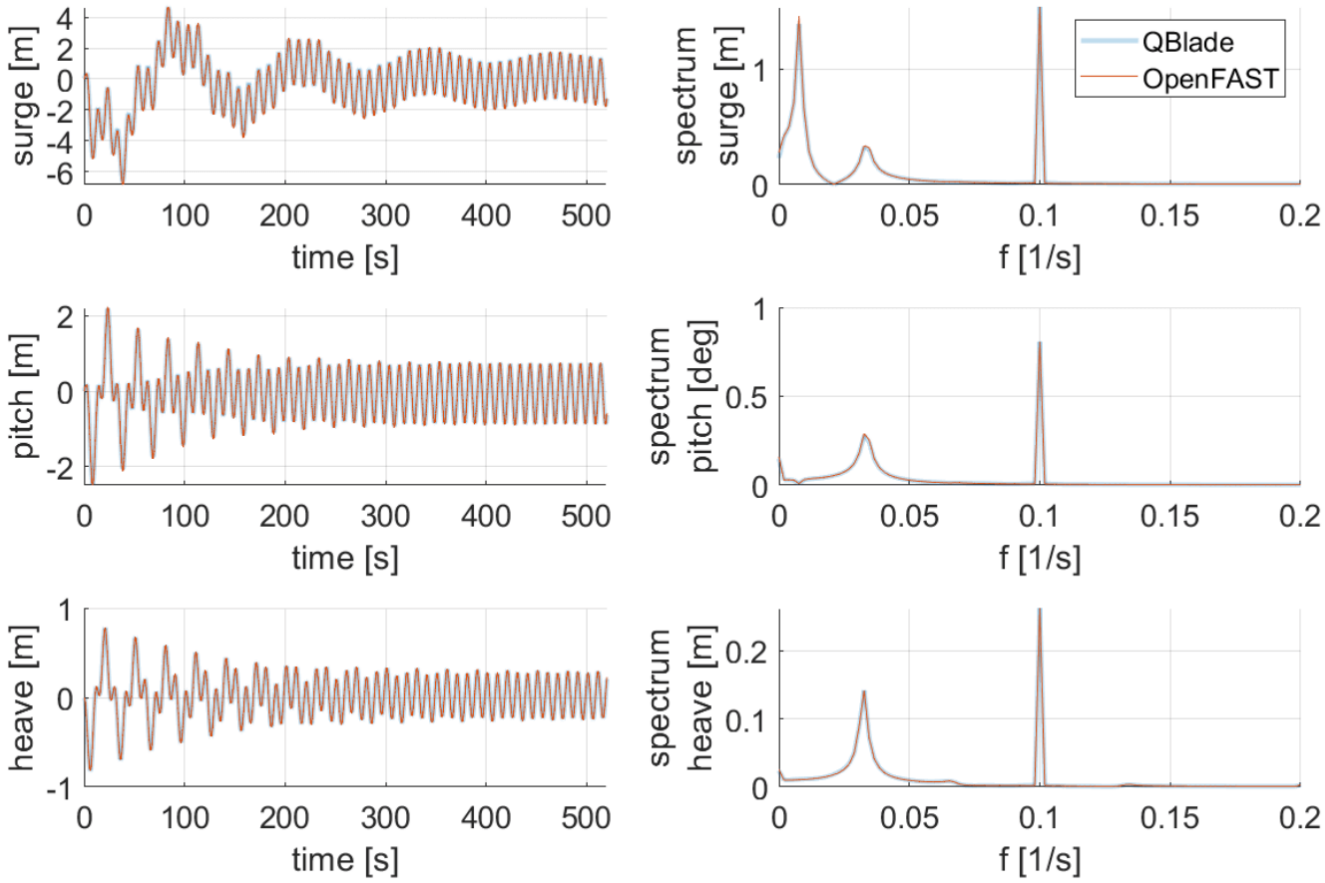


Fig. 166 *Time series (left column) and corresponding spectra (right) of relevant DOFs for regular waves with a wave height of 6 m and a period of 10 s (0° incoming angle).*

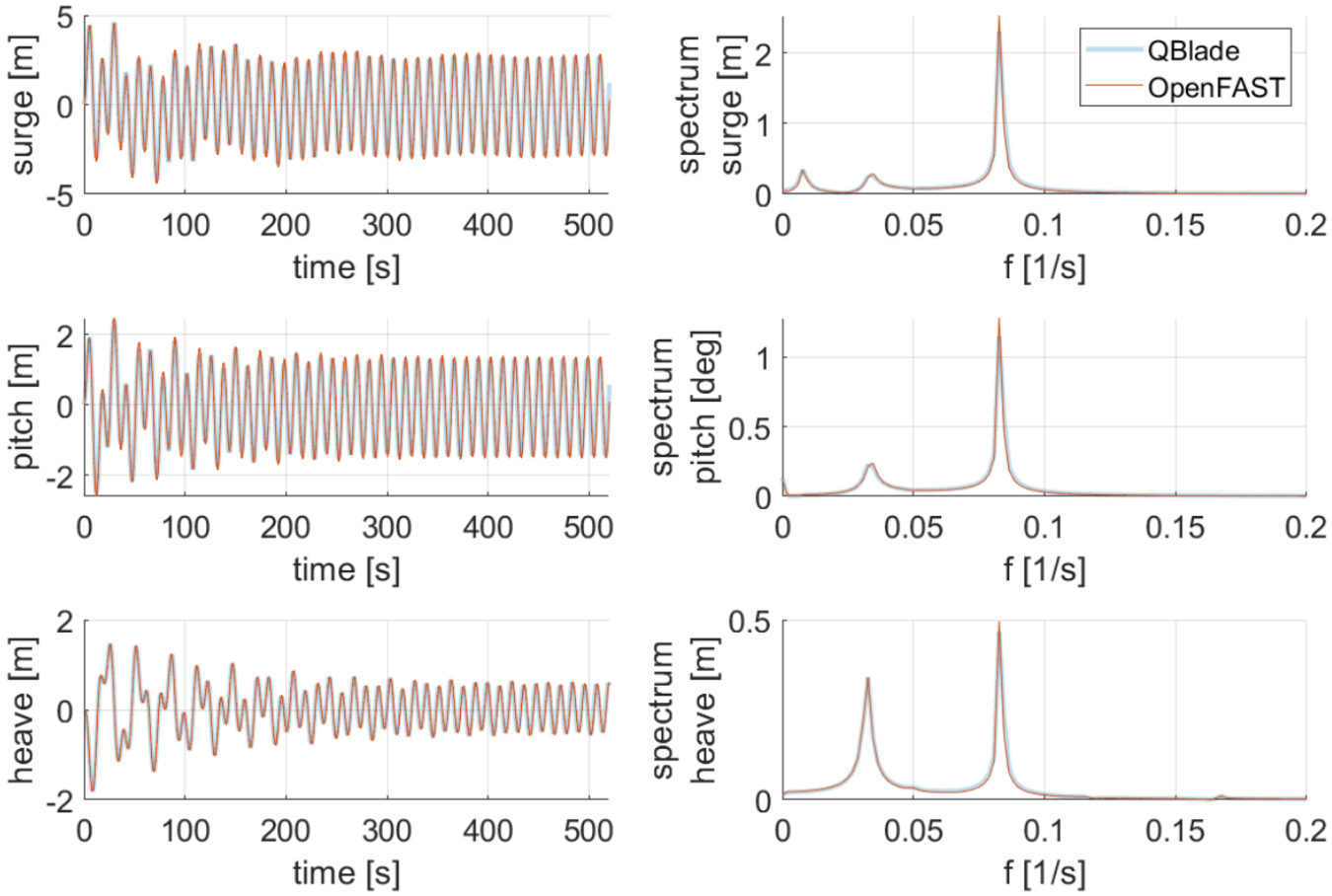


Fig. 167 *Time series (left column) and corresponding spectra (right) of relevant DOFs for regular waves with a wave height of 8 m and a period of 12 s (0° incoming angle).*

The next feature requiring validation is the influence of waves that approach the floater with an oblique angle. The excitation input file provides IRFs with a degree spacing of 10 degrees (-180°:10°:180°). Thus, a wave that heads in from an intermediate angle requires interpolation of the IRF. In order to validate this, Fig. 168 shows the time series of all DOFs for a regular wave with a wave height of 6 m, a period of 10 s and a wave angle of 45 degrees. Once again both tools show very good agreement in the floater response. Very slight differences in the yaw DOF might be attributed to differences in the interpolation algorithms.

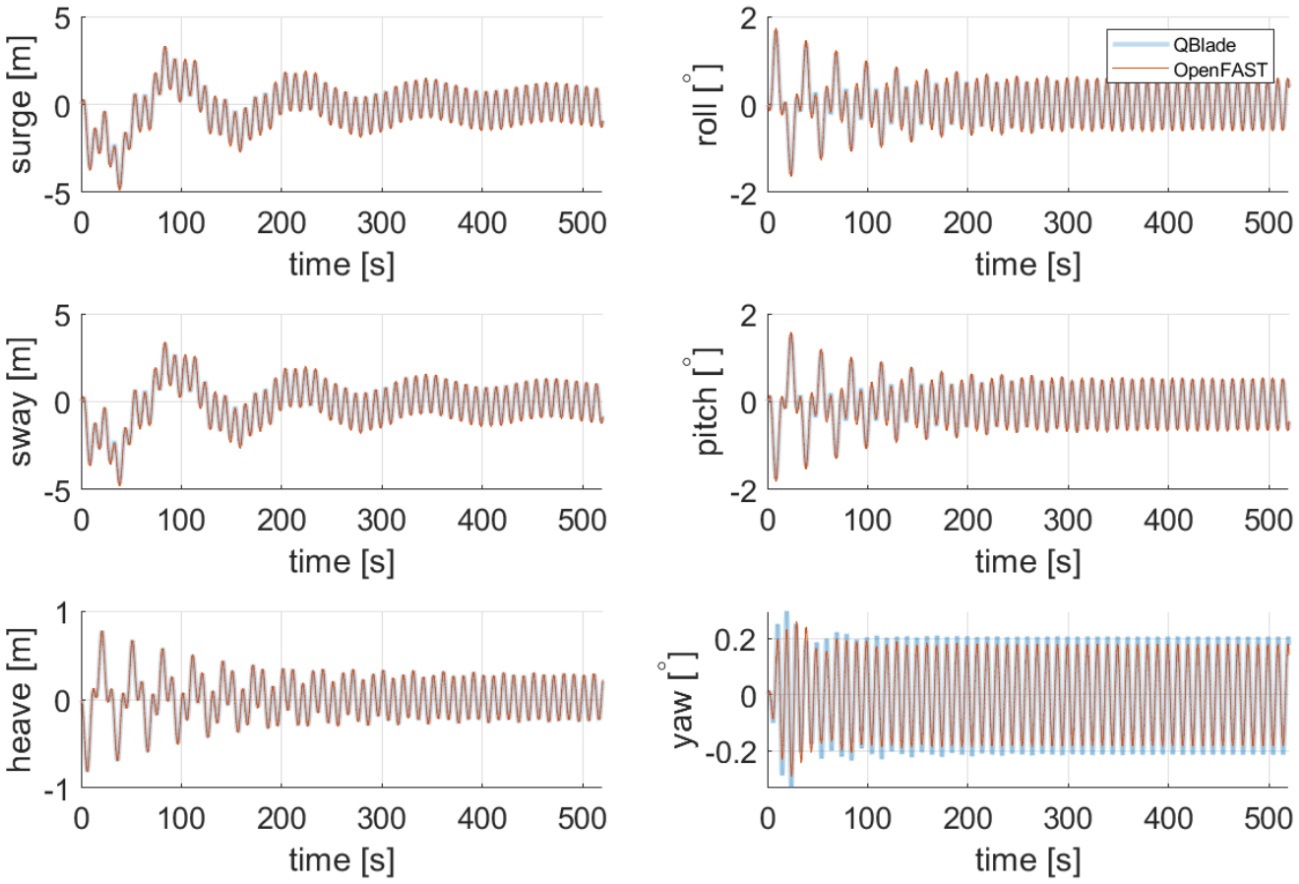


Fig. 168 *Time series of all DOFs for regular waves with a wave height of 6 m and a period of 10 s (45° incoming angle).*

OC3 LPMD Irregular Wave Cases

The validation of the first-order excitation loads in irregular waves was performed in a similar fashion to the regular wave cases. Accordingly, the mooring system and buoyancy are modelled via linearized matrices. The IRFs were precomputed in WAMIT and are identical to the ones used in OF. At this point it is worth mentioning that the algorithmic approach doesn't change within the hydrodynamic model of QB for an irregular wave field compared to a wave field consisting of a single wave train. The reason is, that the excitation loads are computed for every single wave train and superposed linearly (see Linear Wave Theory). Two separated cases are analyzed. Firstly, an irregular wave field based on a JONSWAP spectrum with uni-directional waves is considered. Secondly, a directional spread of the wave trains is added in order to further validate the direction-dependent calculation of the wave loads. To increase the statistical validity of the results, six runs were carried out for each simulated case. The significant wave height amounts to H_s = 6 m and the peak spectral period to T_p = 10 s. A peak enhancement factor of γ = 3.3 was chosen. The floater response in all 6 DOFs will serve as the validation parameter. Finally, statistical parameters of the floater response and the tower base loads will be presented.

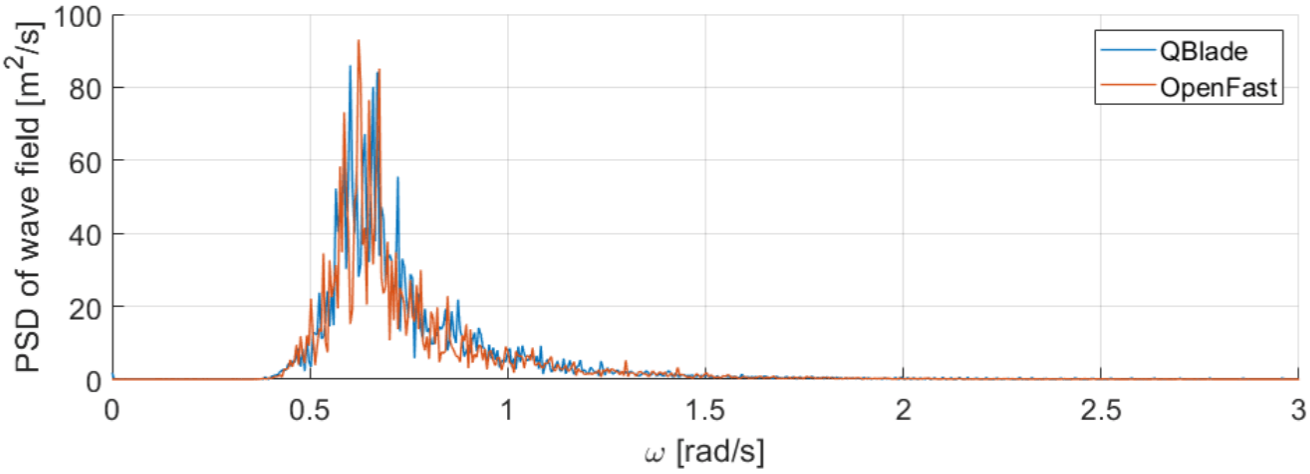


Fig. 169 *Averaged JONSWAP spectrum with* H_s *= 6 m,* T_p *= 10 s and* γ *= 3.3.*

Fig. 169 shows the averaged wave fields that serve as an input to the calculation of the wave forces. In Fig. 170 the floater response for an irregular wave field with uni-directional waves is presented. The three excited DOFs (surge, heave, pitch) show good accordance between QB and OF. The results are averaged over the last 250 s of the aforementioned six runs, each with 800 s total duration. The peak at the floater eigenfrequency arises due to not completely damped out initial transients.

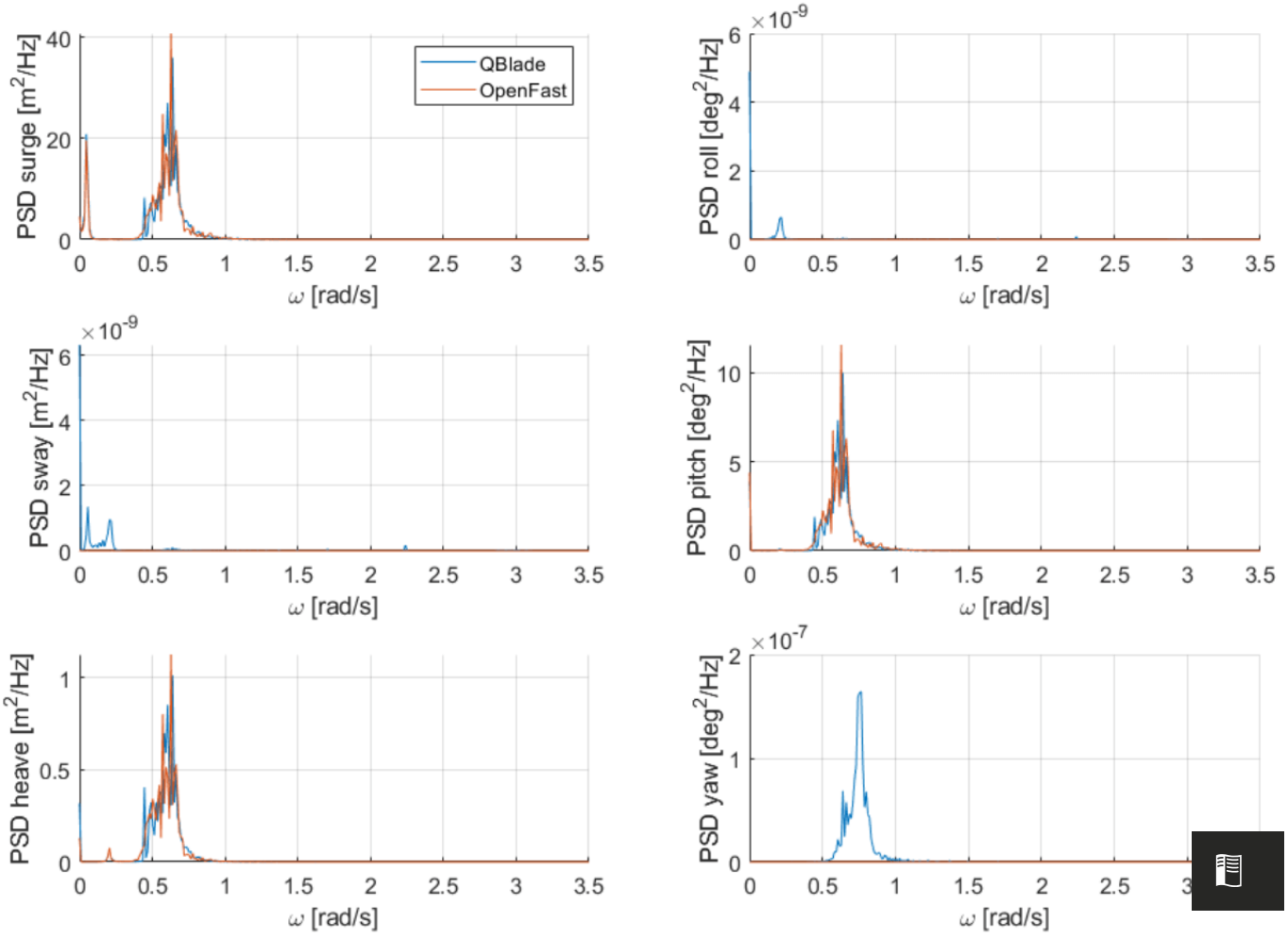


Fig. 170 *Averaged PSDs of the floater translations and rotations in uni-directional, irregular waves.*

Fig. 171 shows the PSDs of the floater response for multi-directional waves in all 6 DOF s. To reduce the influence of the initial transient, the total simulation length was set to 1200s. Again. The PSDs are averaged over the last 300s of six different simulations.

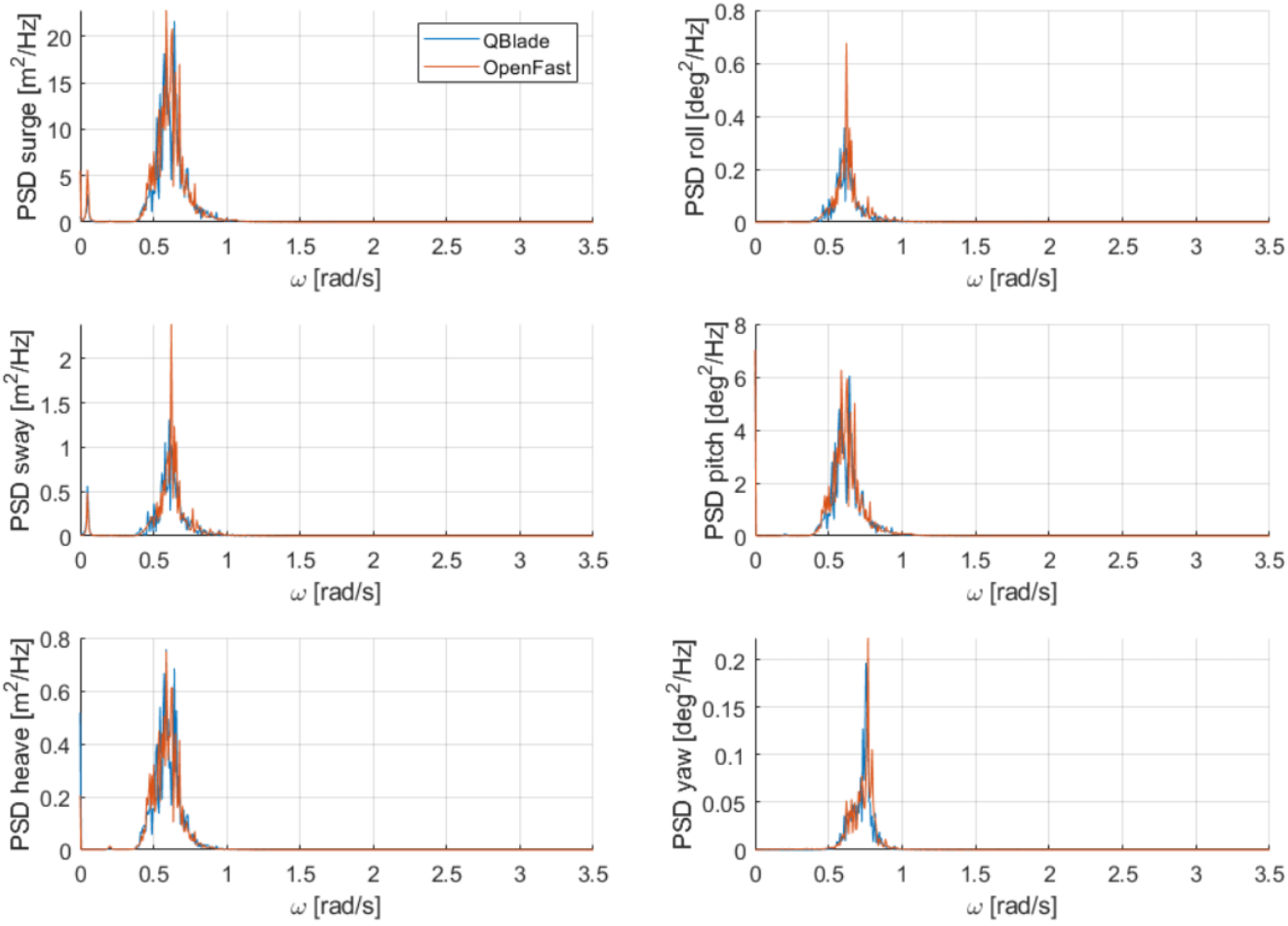


Fig. 171 *Averaged PSDs of the floater translations and rotations in multi-directional, irregular waves.*

Fig. 172 shows the time average, the standard deviation as well as minima and maxima of the floater response in the 6 DOFs of both tools. Small deviations between the codes are visible but in general a similar behavior is visible once again. An exact matching between these statistical parameters may not be expected as they depend on the occurrence of severe wave groups. A longer simulation time would presumably increase the agreement between the compared codes.

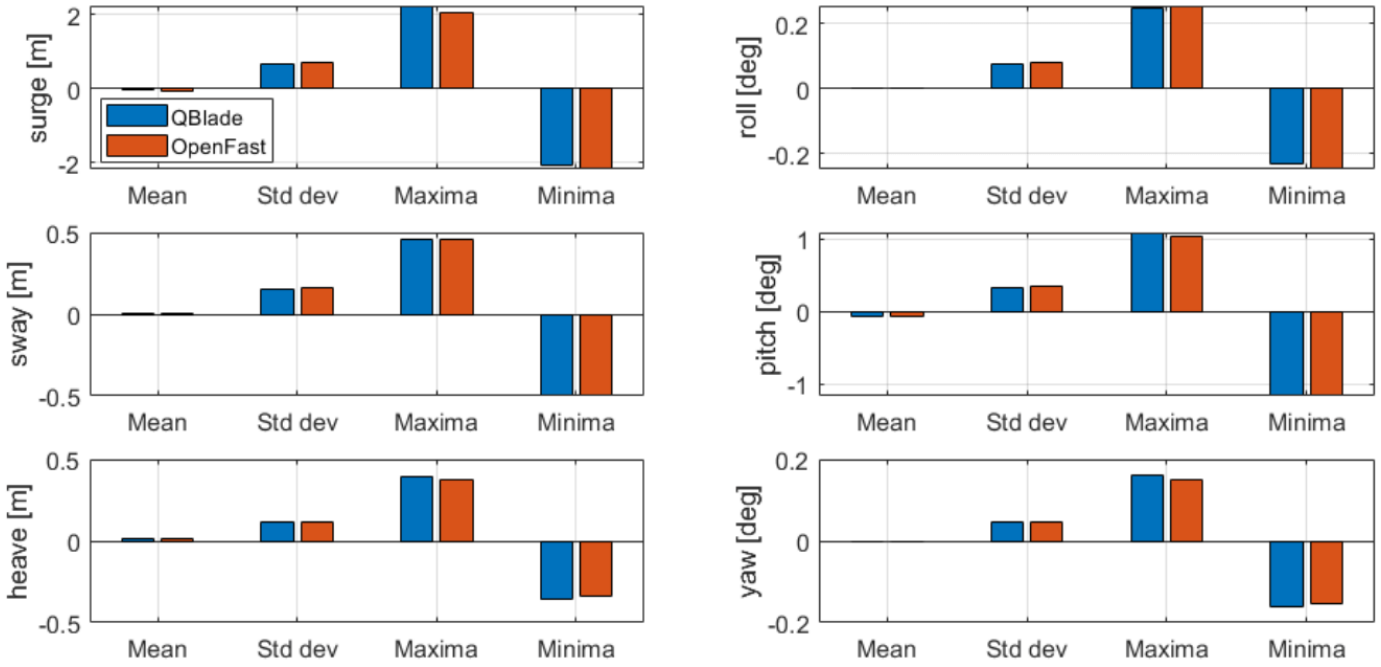


Fig. 172 *Comparison of mean, standard deviation and extreme values of all 6 DOFs for multi-direction irregular waves*

OC3 LPMD Irregular Waves with Current

For the irregular wave test cases, a combination of wave and currents was also considered. For this case, a JONSWAP spectrum with a significant wave height of H_s = 6 m, a peak spectral period of T_p = 10 s and a value of γ = 3.3 was again chosen. This wave spectrum was combined with a constant tide-induced current with a power law of 1/7th and a surface current value of 0.5 m/s. The wave and current directions aligned with the positive surge direction. Six repetitions were considered to account for the statistical variance of irregular waves. In this test case, the same wave elevation input was used for OF and QB simulations. Again, no aerodynamic loads were applied on the turbine. The mooring systems were modelled explicitly.

We can see in Fig. 173 the averaged PSDs of the six repetitions for all DOFs for both simulations. The figure shows that the substructure dynamics for the relevant DOFs under irregular sea states and constant currents is virtually identical if simulated with OF or QB.

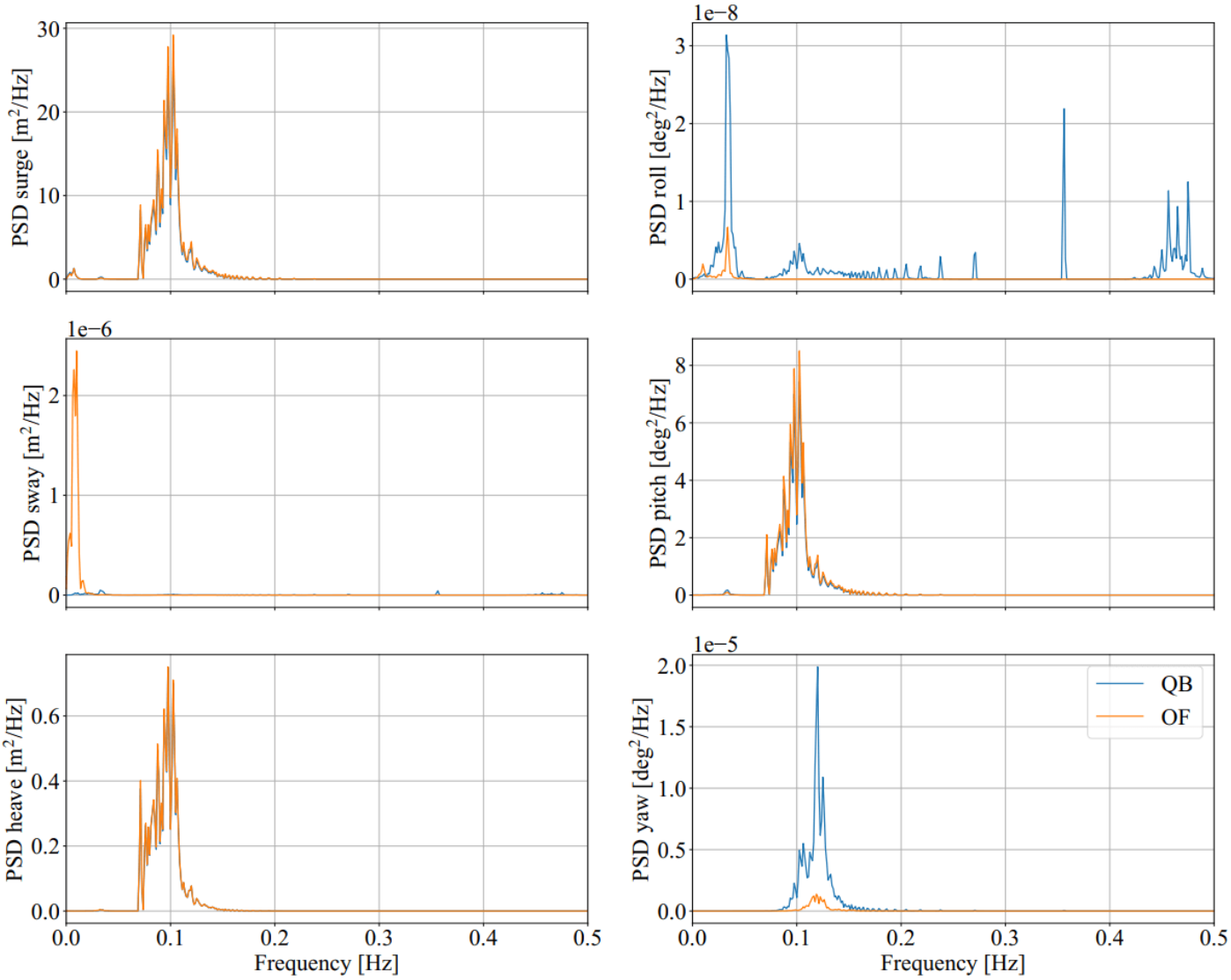


Fig. 173 *Averaged PSDs of all DOFs afor irregular wave and current simulations*

Fig. 174 shows the averaged PSD of the corresponding tensions at the fairlead and anchor positions for the irregular wave plus current cases. We can see in this figure that there is a larger variation of the fairlead and anchor tensions for the downwind mooring lines (lines 2 and 3). This difference can be attributed to the different mooring system modelling that is present in QB and OF.

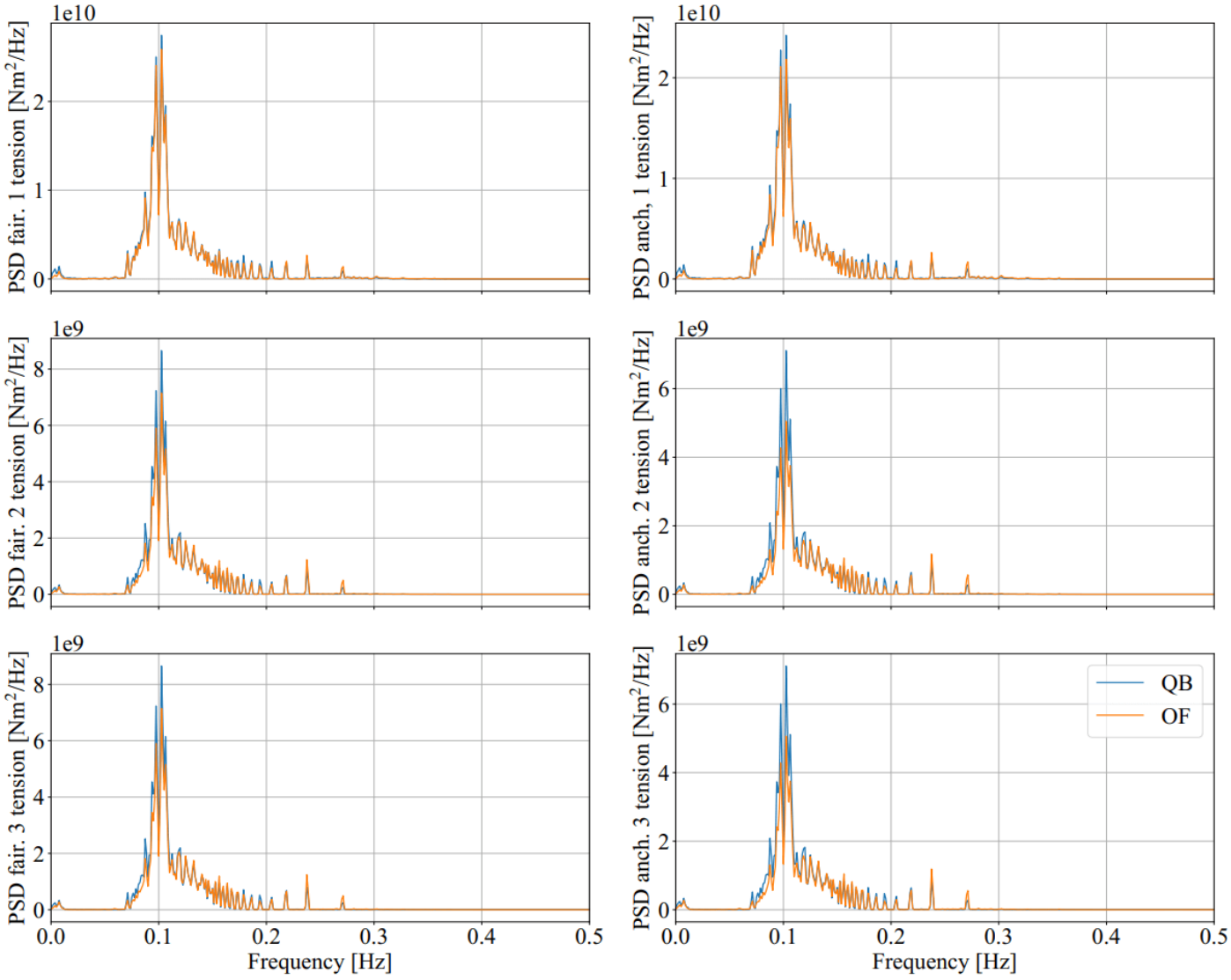


Fig. 174 *Averaged PSDs of the fairlead and anchor tension for the irregular wave tests with constant current*

OC4 Semisubmersible LPMD Model

The second model considered in this validation – named OC4 model in this document – is the floating 5 MW wind turbine mounted on a semisubmersible substructure from the OC4 Report [8]. The geometry of the OC4 model within the QB GUI is shown in Fig. 175. This figure shows clearly the more complicated geometry of the turbine model. Hence, the hydrodynamic behavior of this model is expected to be more complicated. The substructure was again considered rigid in this model to evaluate only the hydrodynamic modules in this more challenging geometry. Unless otherwise stated, the mooring system was modeled explicitly and the localized buoyancy model was used for this model.

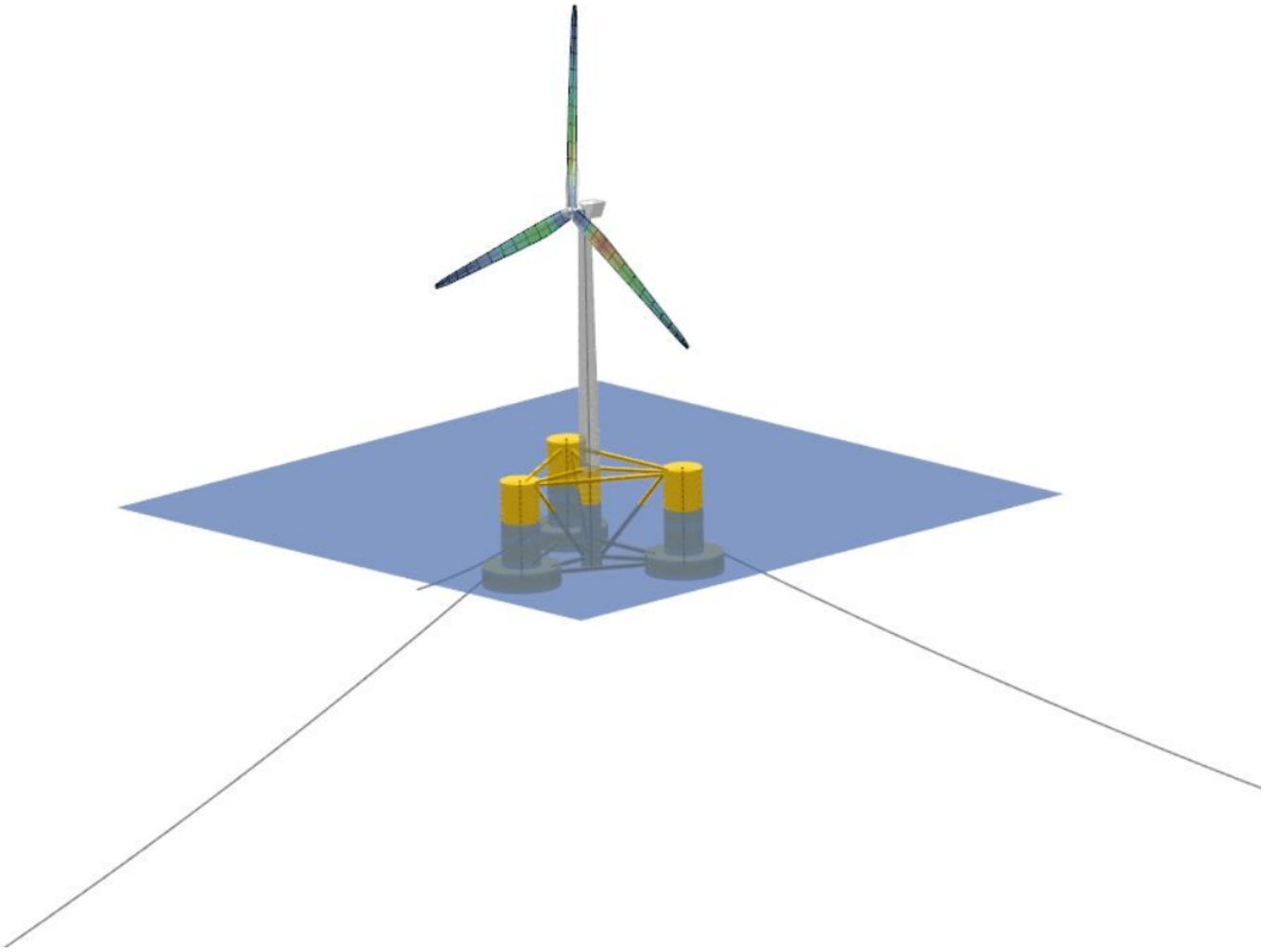


Fig. 175 *The OC4 model displayed in the QB GUI featuring a semisubmersible substructure.*

OC4 LPMD Free Decay Tests

For this model, decay tests were again performed in still water for four DOFs and compared to the same simulations performed with OF. Again, the main difference between both codes in these tests were the mooring system modelling and the way the buoyancy was calculated.

Fig. 176 to Fig. 178 show the free decay tests for the surge, pitch and yaw DOF. We can see again that the results for QB and OF are very similar. Especially for the disturbed DOFs and the DOF that are directly coupled to them, the differences between both codes are small. We note again a small difference in the mean of the heave position. This comes from the different buoyancy models used in QB and OF.

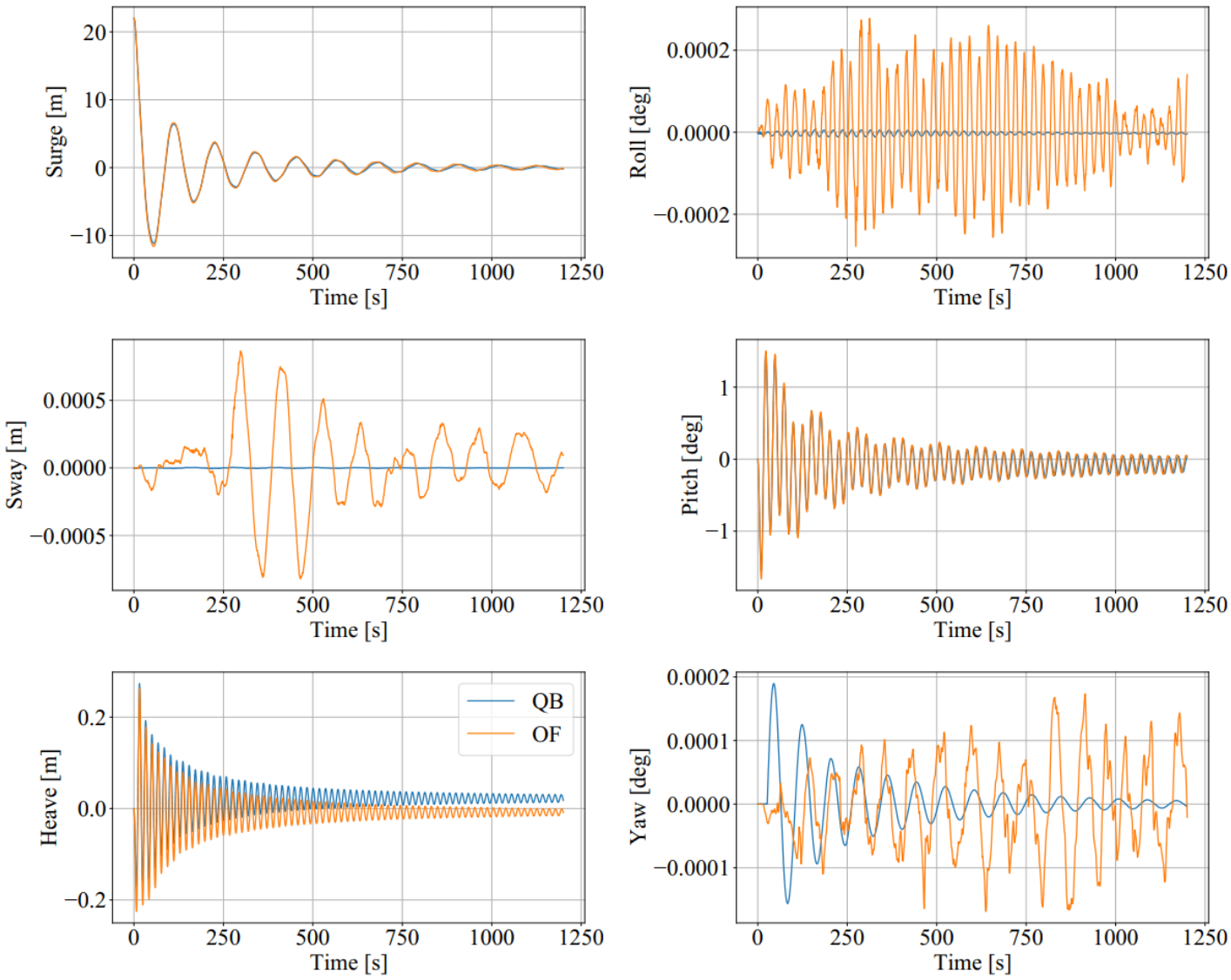


Fig. 176 *Time series of the OC4 model surge decay test.*

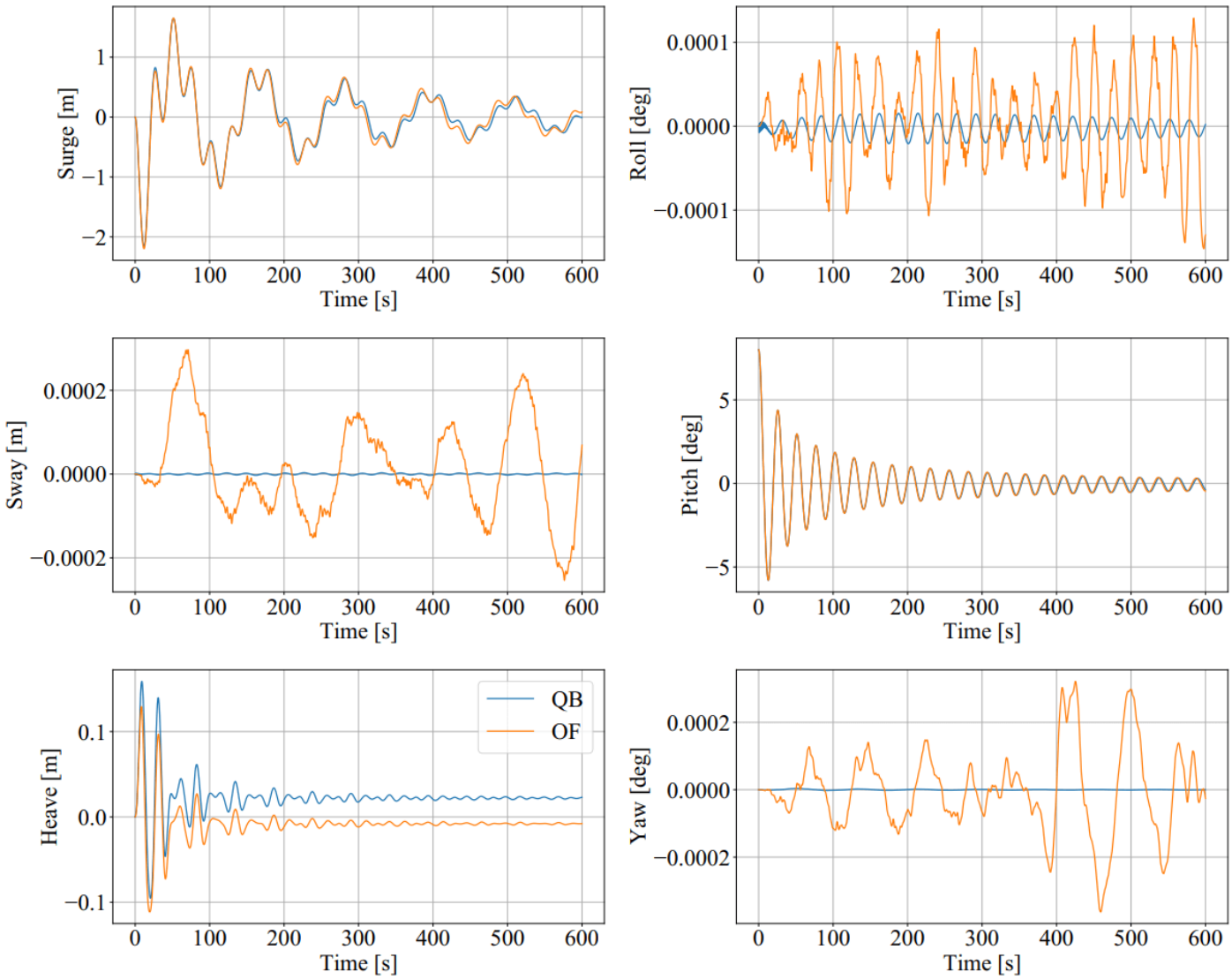


Fig. 177 *Time series of the OC4 model pitch decay test.*

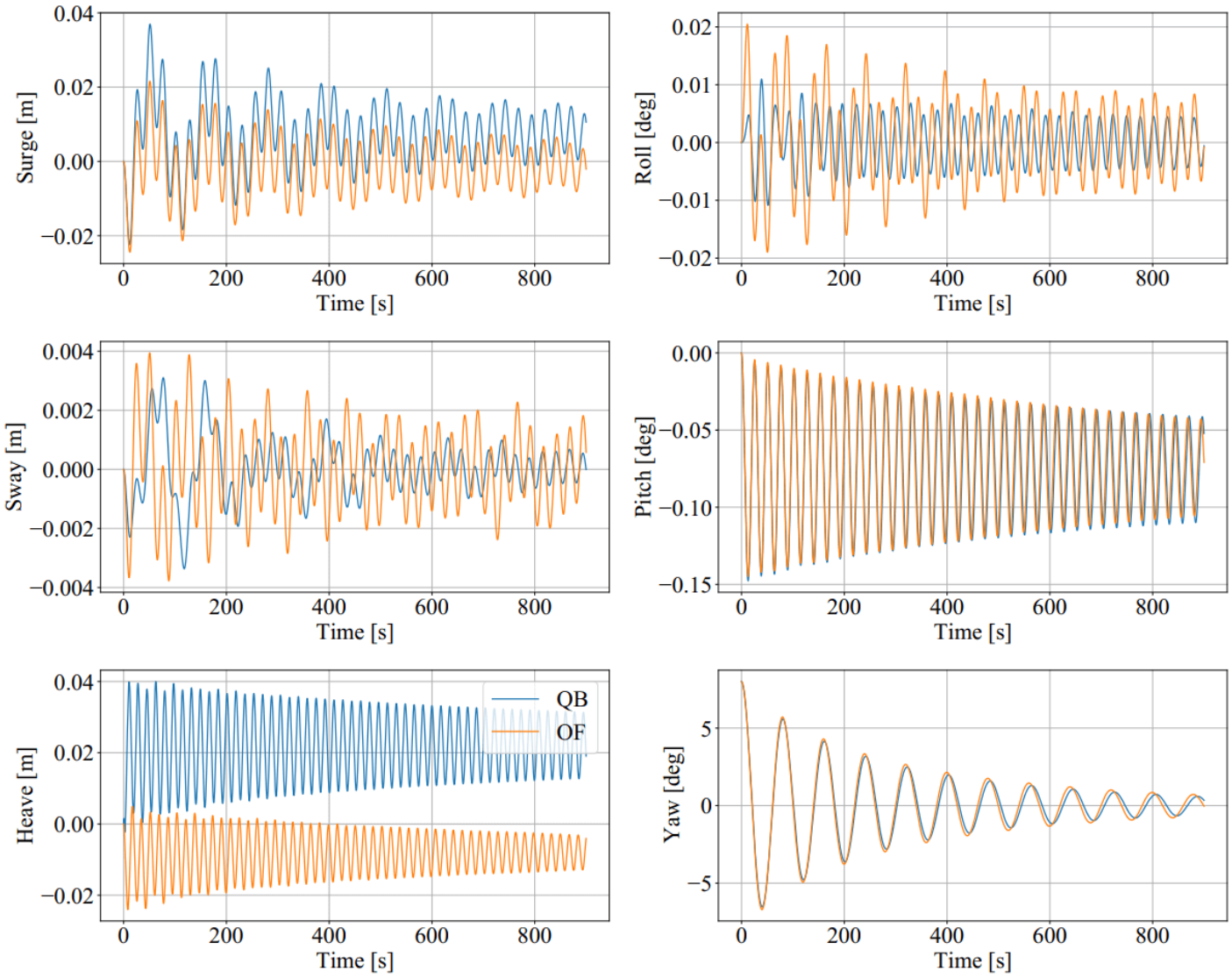


Fig. 178 *Time series of the OC4 model yaw decay test.*

Fig. 179 shows the corresponding tensions at the fairlead and anchor locations of the mooring systems for the surge decay tests. Again, we can see good agreement between both codes. There is an offset in the tensions between the OF and QB simulations. This can be explained by the small offset in the heave neutral position between both codes and also from the different modelling approaches used for the mooring system. The tensions for the other DOFs were also analyzed but not included here for brevity reasons. The findings of these other decay tests are equivalent to the ones shown in Fig. 179.

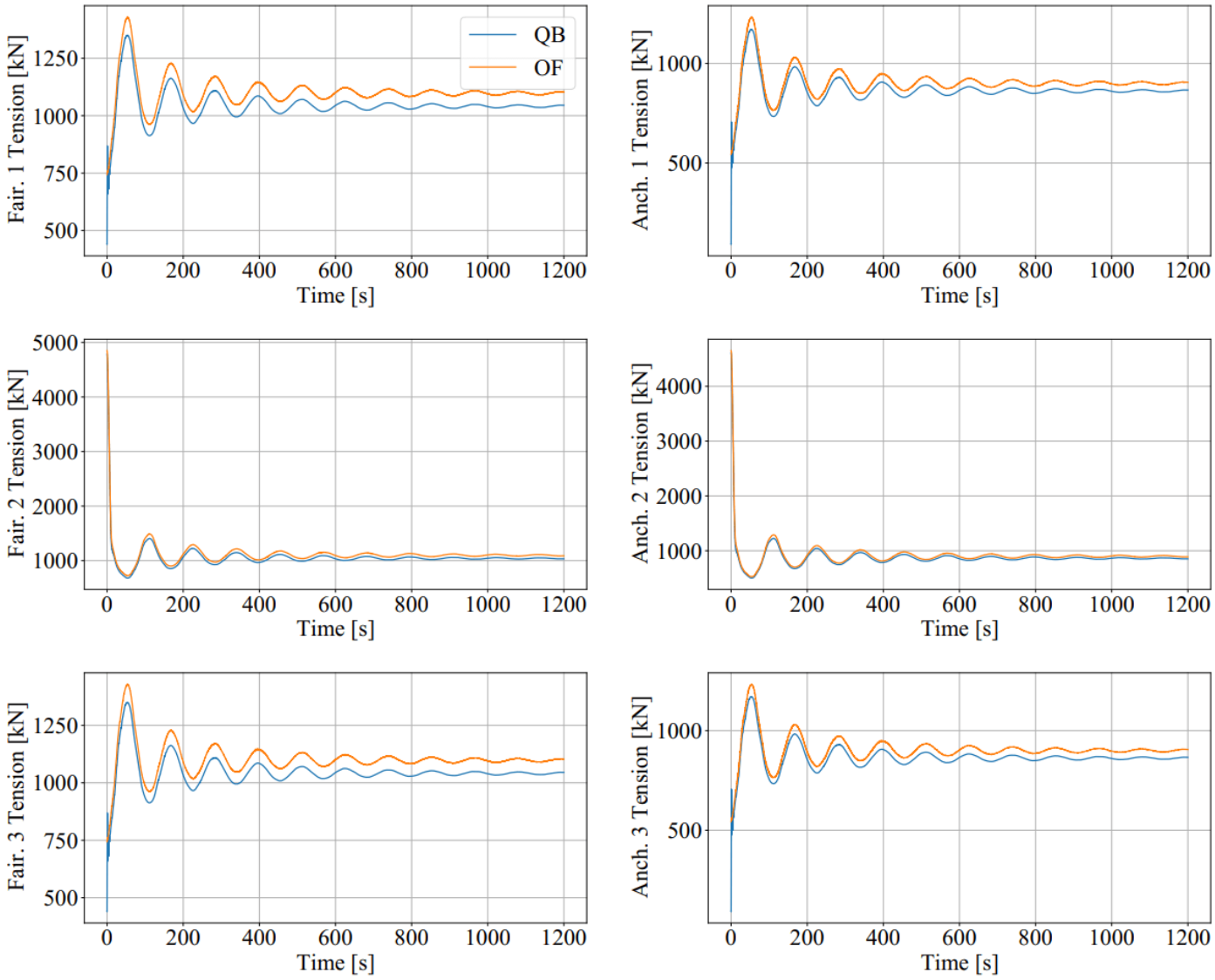


Fig. 179 *Mooring line tensions for the OC4 surge decay test.*

The numerical values for the frequencies and damping coefficients of the decay tests were also analyzed for these cases. The results are shown in Fig. 180. In this figure, we can see that the relative values of eigenfrequencies between both codes are close to 1. This agreement can also be seen visually in the figures above. The same can be said for the quadratic damping term. As for the linear damping term, we can see that there are some differences between the relative values of both codes. As with the OC3 model, the differences can be traced back to the different mooring system modeling used in both codes.

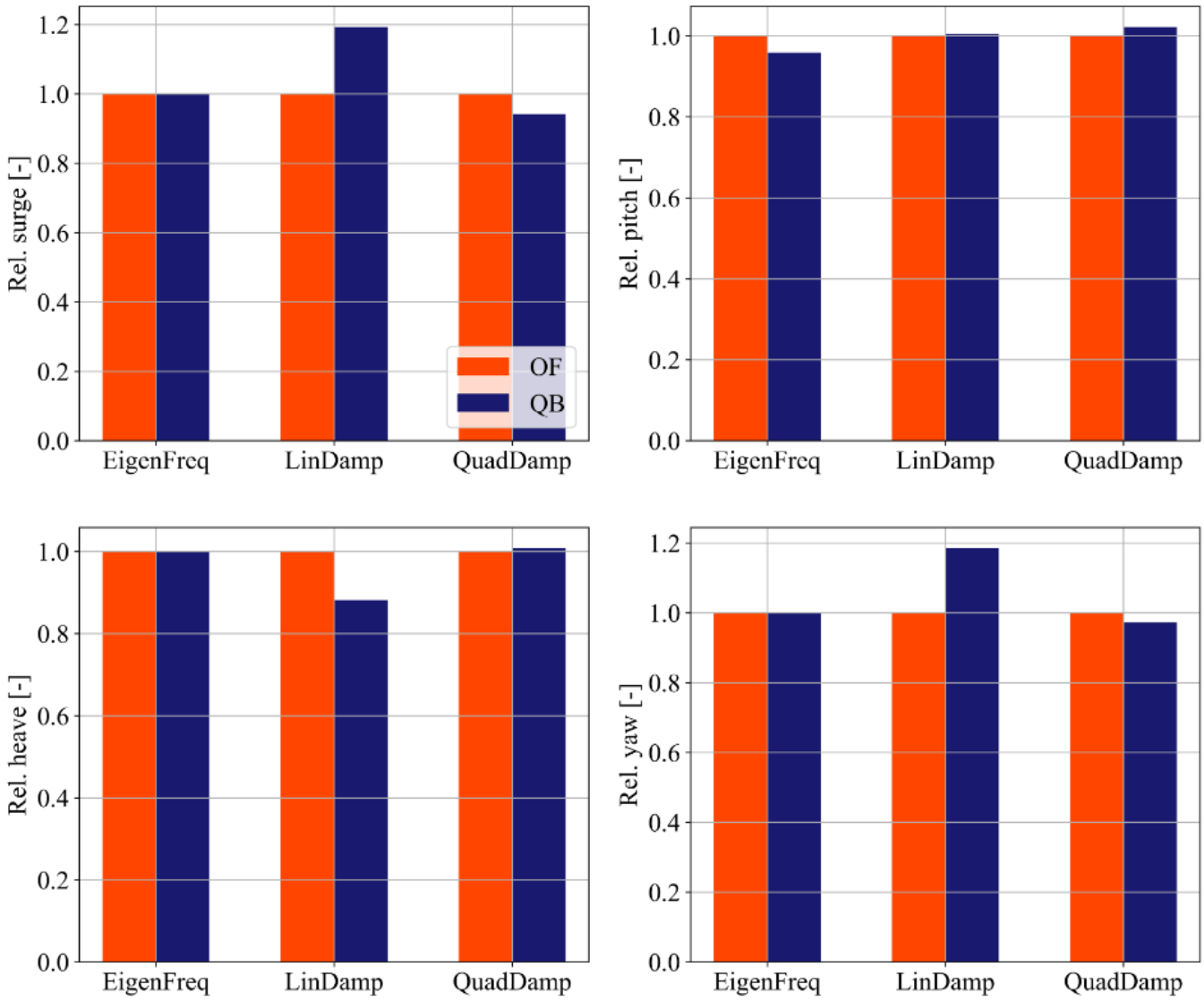


Fig. 180 *Normalized eigenfrequencies and damping behaviour of the OC4 model for the considered decay tests.*

OC4 LPMD Regular Wave Tests

The regular wave tests were performed with linear Airy waves for two selected cases. One case had a wave height of H = 6 m and a period of T = 10 s. The second case had a wave height of H = 8 m and a period of T = 12 s. For these cases, a simulation time of 1000 s was chosen and the first 300 s were discarded. This is because we were only interested in the response of the turbine once the initial transients were settled. No aerodynamic loads were considered and the wave direction was chose to coincide with the direction of the positive surge DOF. As for the wave tests for the OC3 model, no wave stretching model was applied in QB so that the modelling considerations between QB and OF were as close as possible.

It should be noted here that for the more complex geometry of the OC4 model, the buoyanc el will affect the response of the substructure to the incoming waves. This comes from the fact he bodies that provide buoyancy for the OC4 model are spatially distributed. As the wave passes the

substructure, the local buoyancy forces induce additional forces and moments that affect the principal DOFs of the substructure. These forces and moments are not accounted for if only a linear constant force and restoring force matrix is used to account for the buoyancy.

To verify the hydrodynamic models for the radiation forces, the wave excitation forces and the diffraction forces as well as the quadratic drag forces from the Morison equation, a modified OC4 LPMD model was built that includes a linear buoyancy model in a similar fashion as OF. This model was termed “QB Lin” in this section. Additionally, the complete OC4 LPMD model with the distributed buoyancy forces was also modelled. This model was used to analyze the effect of the buoyancy model on the substructure response and on the tensions of the mooring system.

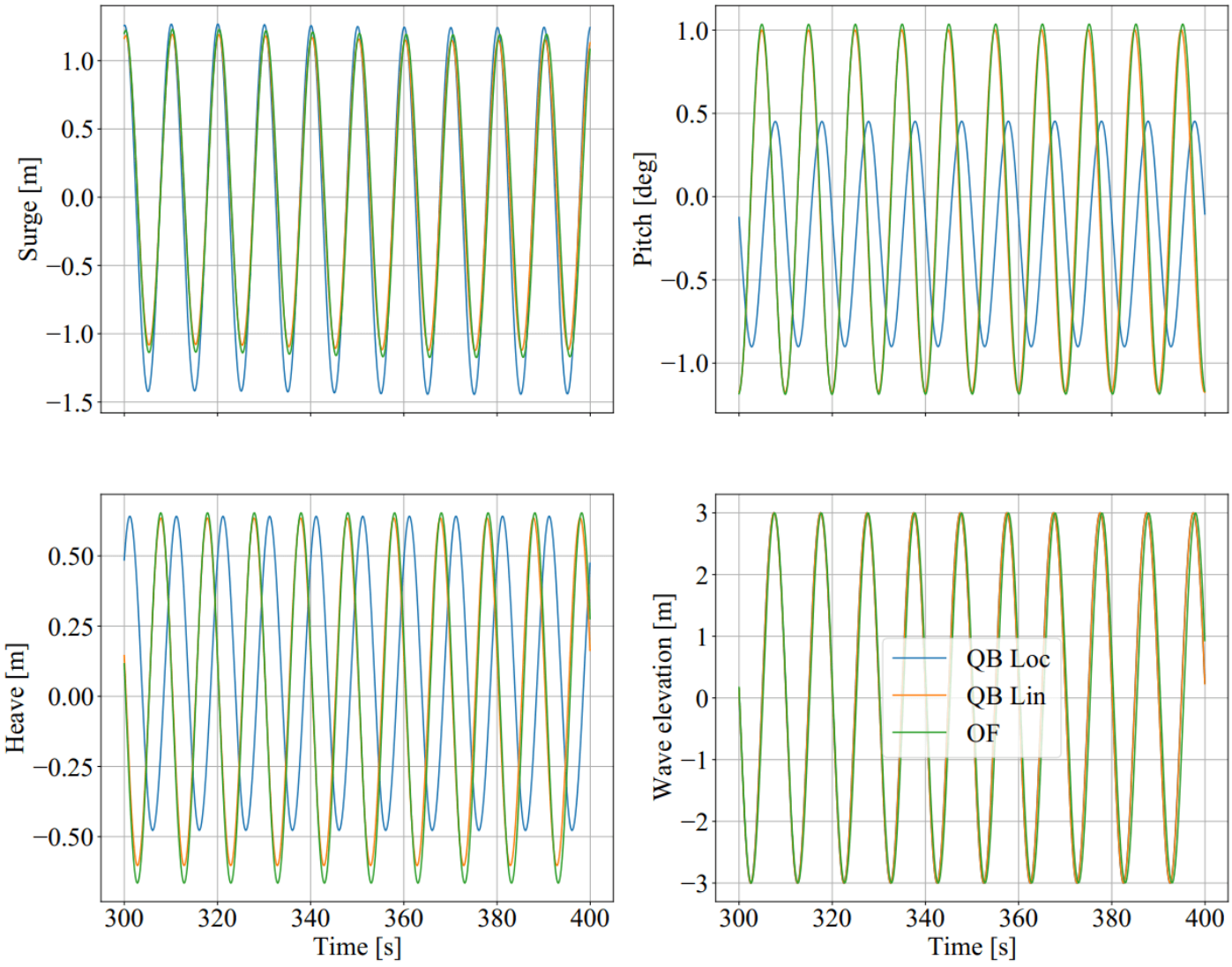


Fig. 181 *Relevant DOFs and wave elevation for regular sea state with H = 6 m and T = 10 s. QB Loc = QB local buoyancy model, QB Lin = QB linear buoyancy model.*

Fig. 181 shows the response of the surge, heave and pitch DOF as well as the wave elevation for the test case with H = 6 m and T = 10 s. We can see in this figure that the response of the line buoyancy model in QB (QB Lin) is practically identical to the response in OF. This validates t

hydrodynamic modules for radiation, diffraction and wave excitation forces as well as the quadratic drag forces in the more complicated geometry of the OC4.

If we use the local buoyancy model in QB (QB Loc), we can see in Fig. 181 that all the relevant DOFs are affected. We can see that the surge DOF oscillates with a larger amplitude and reaches more negative values compared to the linear buoyancy model. In addition, the amplitude and phase of the pitch and heave DOFs change if we use the local buoyancy model. In particular, the amplitude of both DOFs is smaller and the phase shifts with a positive magnitude. This behavior was not seen in the regular wave calculations of the OC3 model because all the buoyancy forces were concentrated on the spar axis and were not spatially distributed.

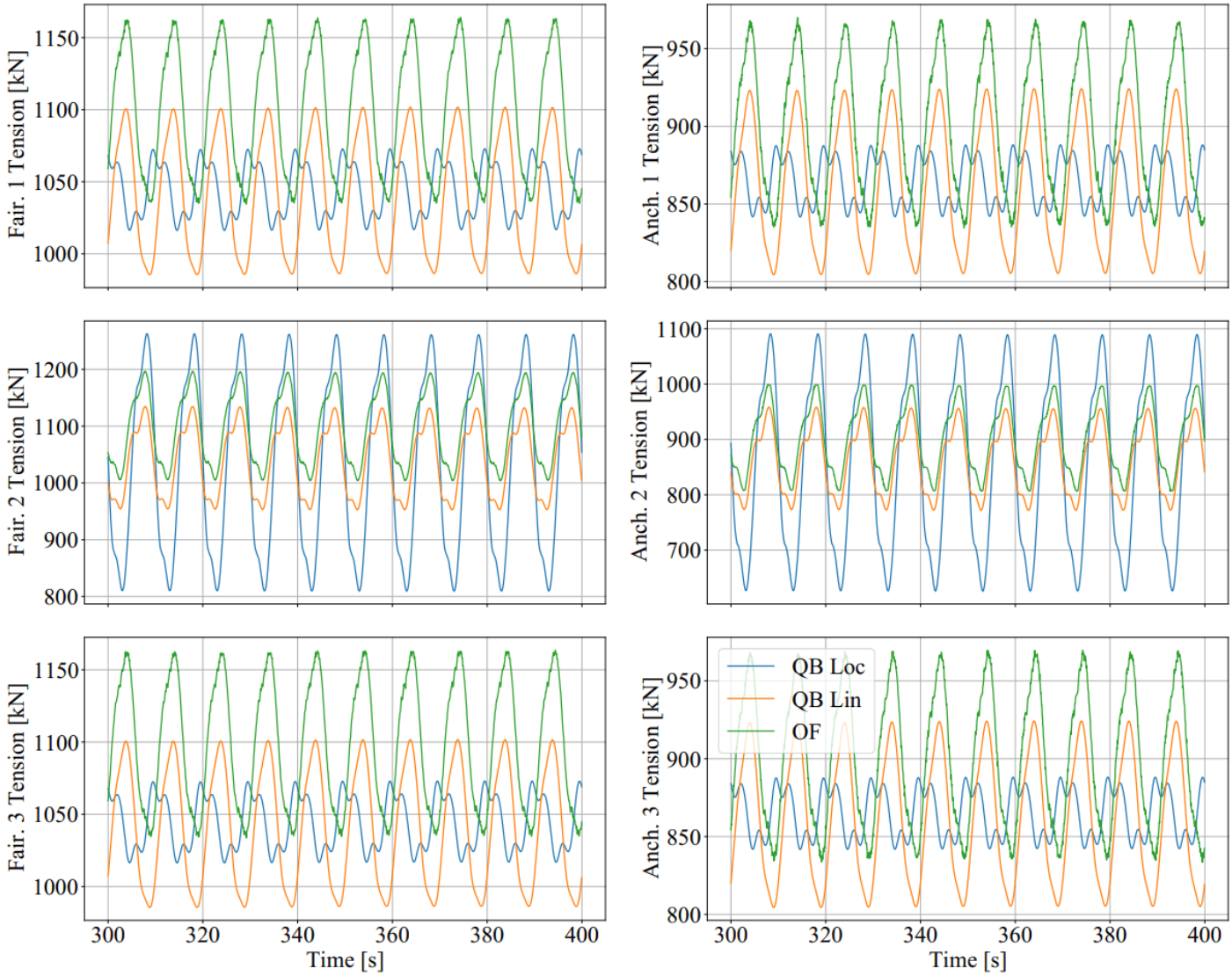


Fig. 182 *Mooring line tensions for regular sea state with H = 6 m and T = 10 s. QB Loc = QB local buoyancy model, QB Lin = QB linear buoyancy model.*

This different behavior due to the local buoyancy model also has an effect on the tensions of the mooring system. In Fig. 182 we can see the mooring line tensions of the regular sea state sh Fig. 181. We can see that there is the slightly higher average value of the mooring tension in the OF calculations compared to the QB Lin calculations. This offset was also identified in the decay tests

and is attributed to the different mooring system modelling in both codes. Important here is that the phase and magnitude of the tensions is comparable in with the linear buoyancy model.

This behavior changes if we include the local buoyancy model. Here, the amplitude of the tension from mooring line 2 increases significantly compared to the QB Lin simulations. For the mooring lines 1 and 3, the effective amplitude of the tension oscillations gets reduced due to an additional oscillation that is shifted in phase. These differences can be attributed to the different response pattern in heave and pitch of the OC4 model when the local buoyancy is used.

Fig. 183 and Fig. 184 show the relevant DOFs and mooring line tensions for the regular sea state with H = 8 m and T = 12 s. We can again see that the response of QB Lin is comparable to OF in both the substructure dynamics and the tensions of the mooring lines.

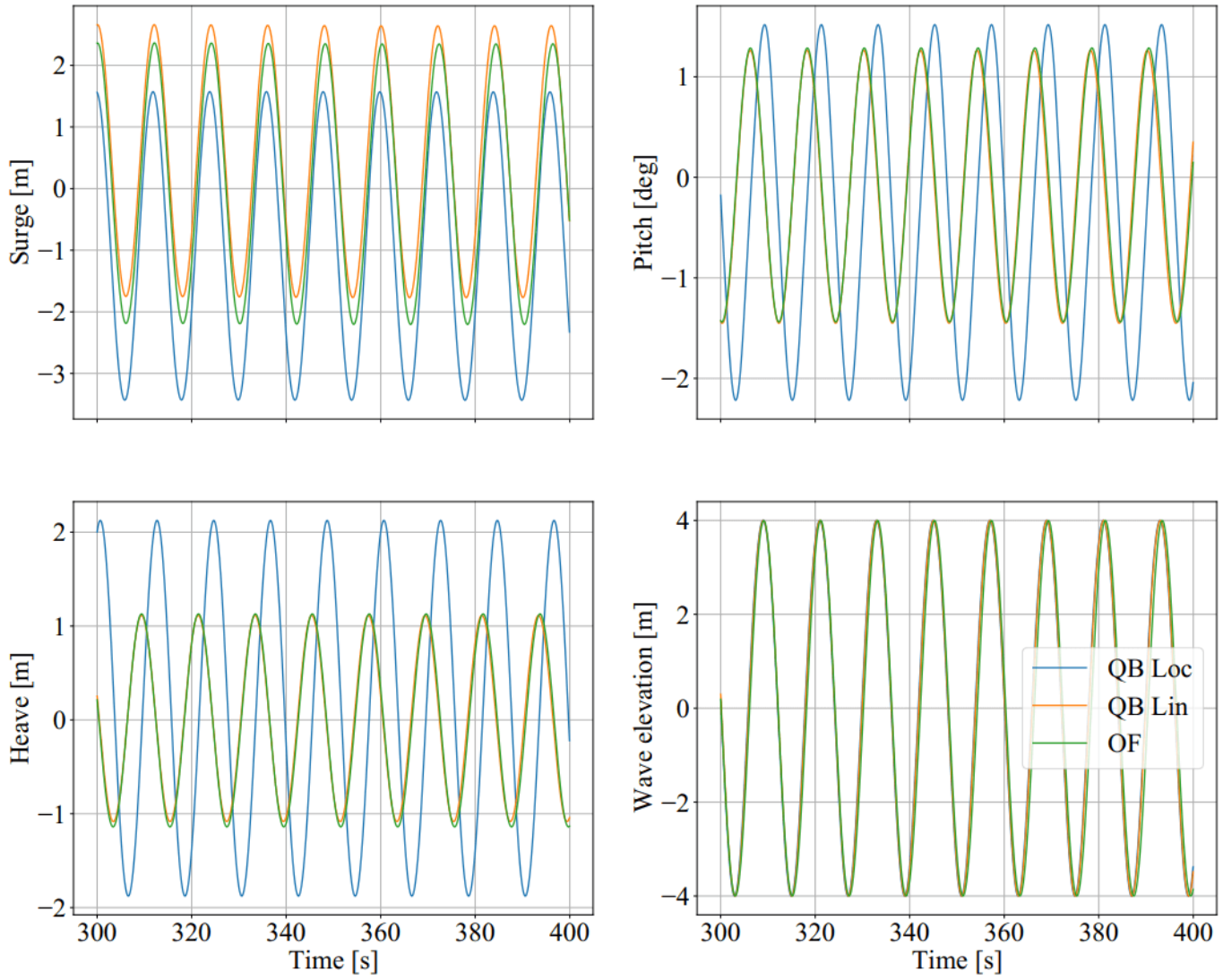


Fig. 183 *Relevant DOFs and wave elevation for regular sea state with H = 8 m and T = 12 s. QB Loc = QB local buoyancy model, QB Lin = QB linear buoyancy model.*

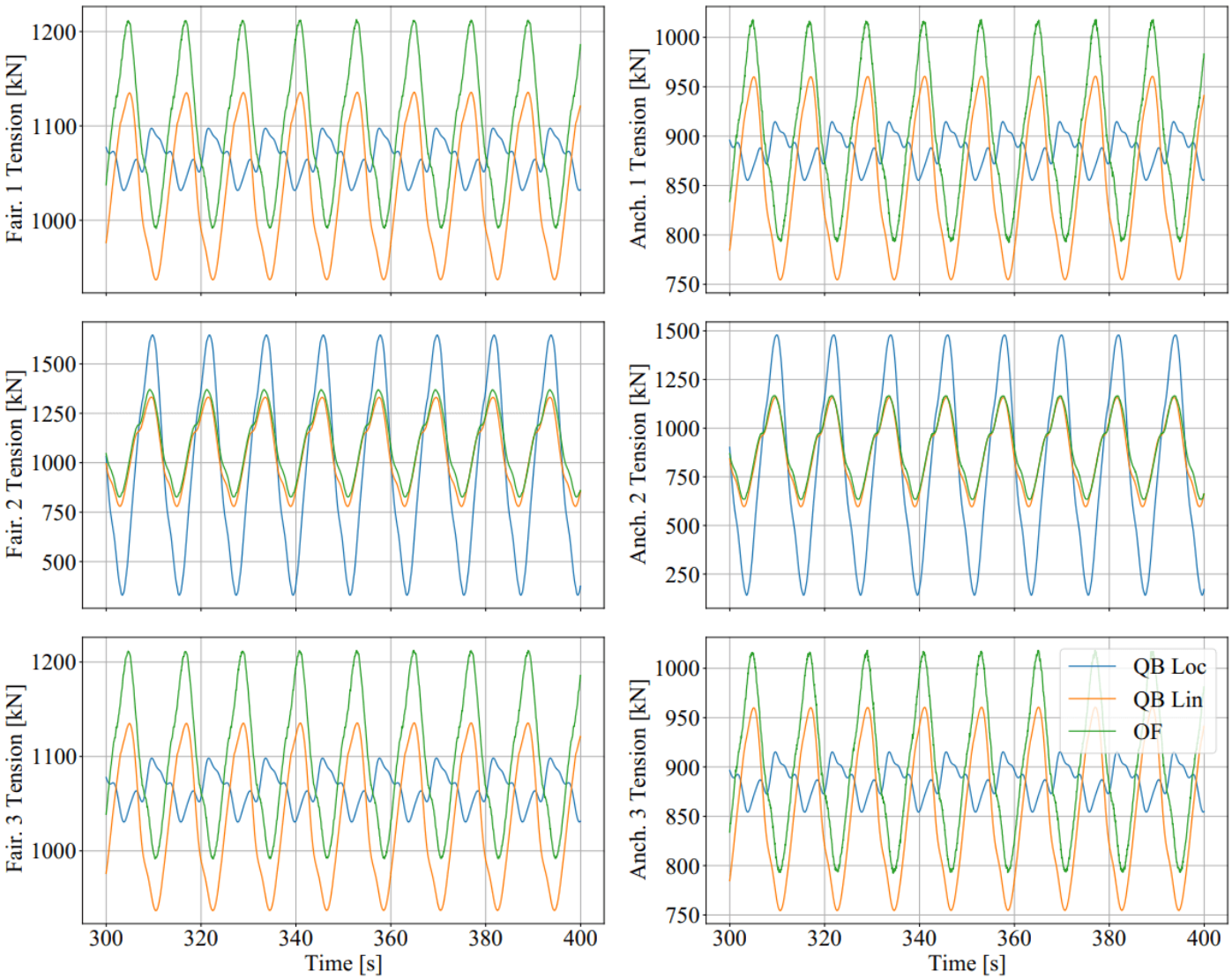


Fig. 184 *Mooring line tensions for regular sea state with H = 8 m and T = 12 s. QB Loc = QB local buoyancy model, QB Lin = QB linear buoyancy model.*

The response of the substructure if the local buoyancy model is used changes even more drastic for this test case. In Fig. 183 we see that the mean surge displacement of the QB Loc simulations is now negative compared to the positive mean surge displacement of the QB Lin simulations. Now, the heave and pitch DOF have a larger amplitude compared to the QB Lin simulations and there is again a positive phase shift.

The effects on the mooring line tensions can be seen in Fig. 184. In this figure, it can be clearly seen that the tension of the mooring line 2 is significantly affected for this test case. The amplitude of the oscillation is more than doubled in the QB Loc simulations compared to the QB Lin simulations.

OC4 LPMD Irregular Wave Tests

The OC4 LPMD model was also validated for irregular wave sea states. Six random sea states with a JONSWAP spectrum with H_s = 6 m, T_p = 10 s and γ = 3.3 were used. The simulation length was 1200 s and the first 400 s were not considered in the analysis to discard initial transient effects. Again, no aerodynamic loads were considered in these cases and the wave propagation direction was

chosen to be aligned with the positive surge direction. Additionally, no wave stretching model and no second order wave forces were included in the simulations. The response of the turbine model was done in a statistical manner by comparing the averaged PSDs of the DOFs.

As we could see in the regular wave validation tests, the buoyancy model has an important effect on the substructure response and the mooring line tensions. For these calculations, the linear buoyancy model was used in QB. Although the local buoyancy calculation is deemed more accurate, we chose the linear model to allow for a better comparison between OF and QB. In addition, the mooring system was also simulated using linear matrices in both codes. We chose to do this to again align the modeling approaches as much as possible. The mooring system is considered validated based on the validation tests done beforehand.

Fig. 185 shows the averaged PSDs of all six DOFs for the irregular sea states. In this figure, the results from three different simulation setups are shown. The original OC4 LPMD model in QB and OF is labeled accordingly in this figure. Additionally, the results QB MSL are also presented in this figure. In the latter simulation setup, the quadratic forces of the Morison equation are calculated considering that the Morison element are wetted up to the mean sea level. This effectively neglects the local wave elevation when calculating the wetted surface of the substructure that will be considered for the Morison drag calculations. This approach is the one implemented in OF. In contrast, QB considers the local wave elevation (incl. wave stretching) to determine the wetted surface of the elements and apply the Morison forces. See modelin considerations in Morison Equation for more details. The setup QB MSL was thus chosen to have a simulation setup that matches the OF modelling.

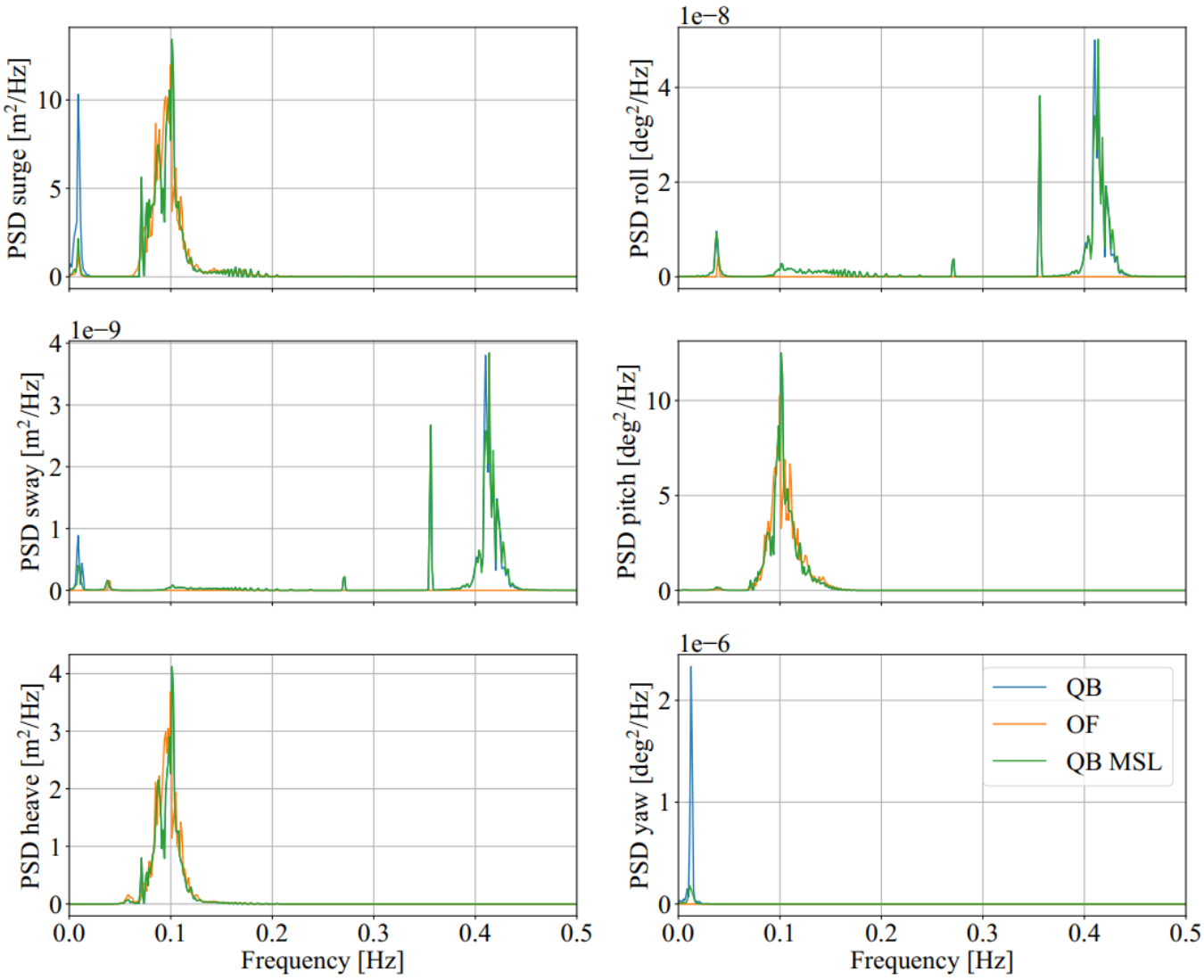


Fig. 185 *Averaged PSDs of all DOFs of the OC4 LPMD model for the irregular sea state with* H_s *= 6 m,* T_p *= 10 s and* γ *= 3.3.*

We can see in Fig. 185 that the averaged PSD of all DOFs are very similar between the OF and QB simulations. There is a distinct difference at the low frequency regime of the surge DOF. The peak corresponding to the surge eigenfrequency is significantly larger in the QB simulations compared to the OF simulations. Notably, if we change the QB simulations to consider the wetted surface up to the mean sea level (QB MSL), the difference in the peak at the surge eigenfrequency vanishes.

This phenomenon can be explained as follows. By considering the Morison drag calculations up to the local wave elevation only, there will be a net positive drag force in the surge direction. This is because as the wave particles retract during the trough of the wave, less surface will be wetted and less drag force will act on the substructure compared to the case where the substructure sees a wave crest. In the latter case, the wetted surface will encompass up to the mean sea level (no wave stretching). The irregular sea state will include time periods where the wave heights are large and time periods where the wave heights are small. The former scenario leads to high average m rge forces while the latter scenario to low mean surge forces. The restoring forces from the linear mooring system will therefore let the OC4 model oscillate at its surge eigenfrequency when the

average surge force is temporarily small. This non-linear phenomenon cannot happen it the wetted surface is considered constant at all times.

OC4 LPMD Second-Order Wave Excitation Forces

Second-order hydrodynamic loads play an increasing role for semi-submersible offshore structures such as the OC4 model. Duarte et al. [9] even state that the floater response is dominated or at least in impacted in the same order of magnitude by the second order hydrodynamic loads as by their first order relatives. Bearing this in mind, a first evaluation of the implementation of the second-order wave load module, described in Linear Potential Flow Theory seems reasonable on the OC4 platform. In the following, the force spectra of the second order wave loads acting in all 6 DOFs on the OC4 model in irregular waves are presented. The previously used JONSWAP spectrum with H_s = 6 m, T_p = 10 s and γ = 3.3 once again was used to create the uni-directional (from 0 deg) wave field. The used QTFs were computed in WAMIT and are identical between QB and OF.

Fig. 186 shows the similarity in the results between QB and OF. Both tools agree very well in the excited frequencies and also in the respective amplitudes. We can further see, that the sum- and difference frequency loads clearly excite the floater outside of the present frequencies of the JONSWAP spectrum, especially in the pitch and heave DOF.

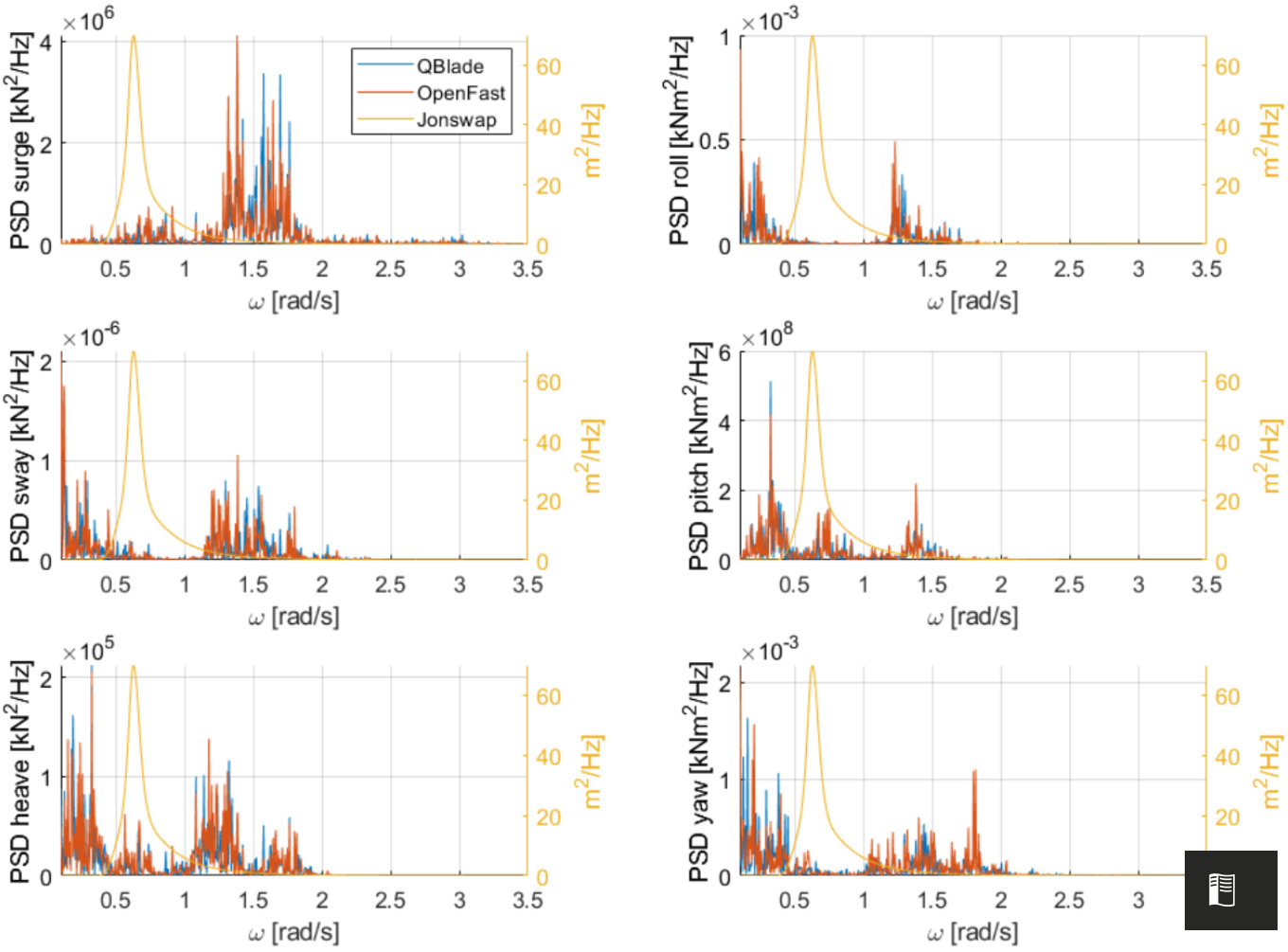


Fig. 186 *Second-order hydrodynamic (sum- and difference-frequency) loads in irregular waves, full QTFs.*

Fig. 187 shows the result of the approximated second-order difference-frequency loads in all 6 DOFs using Newman's approximation. Again, strong similarities between the used simulation tools become evident. It also can be noted that the difference loads contain energy at frequencies lower than those present in the JONSWAP spectrum.

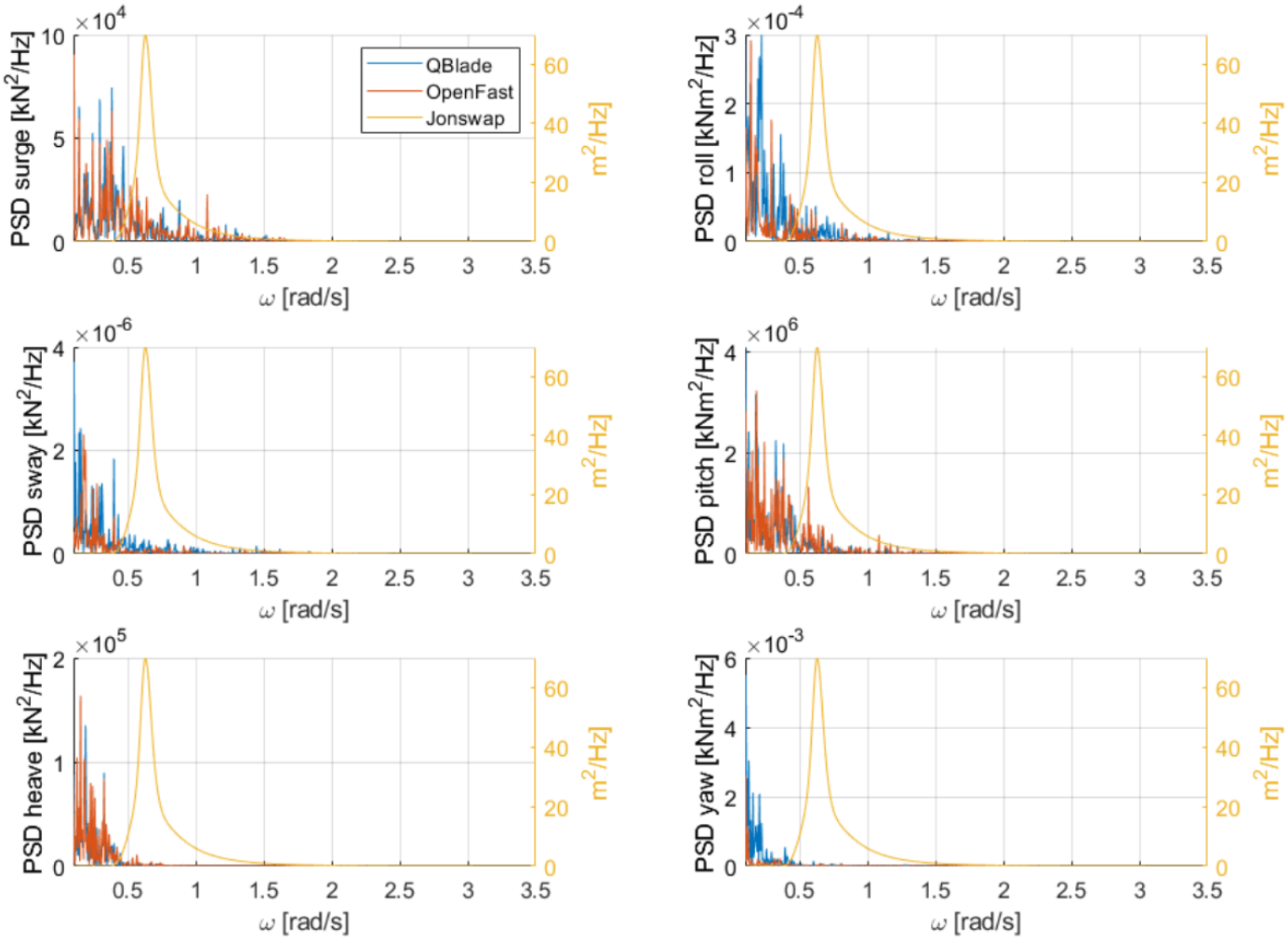


Fig. 187 *Second-order hydrodynamic difference-frequency loads in irregular waves, Newman approximation.*

[1] OpenFAST. OpenFAST. https://github.com/OpenFAST/openfast, 2021. [Online; accessed 2021-12-07].

[2] J. Jonkman. Definition of the Floating System for Phase IV of OC3. Technical Report TP-500-47535, NREL, Golden, Colorado, USA, 2010.

[3] A. Robertson, J. Jonkman, F. Wendt, A. Goupee, and H Dagher. Definition of the OC5 DeepCwind Semisubmersible Floating System. https://a2e.energy.gov/data/oc5/oc5.phase2/attach/oc5.phase2.model.definitionsemisubmersible-floating-system-phase2-oc5-ver15.pdf, 2021. [Online; accessed 2021-11-11].

[4] M. Hall and A. Goupee. Validation of a lumped-mass mooring line model with deepcwind semisu ble model test data. *Ocean Engineering*, 104(1):590–603, 2015. URL:

https://www.sciencedirect.com/science/article/abs/pii/S0029801815002279, doi:10.1016/j.oceaneng.2015.05.035.

[5] DNV. Recommended Practice DNV-RP-C205. Technical Report DNV-RP=C205, DNV, Baerum, Norway, 2014.

[6] WAMIT Inc. WAMIT. https://www.wamit.com/index.htm, 2021. [Online; accessed 2021-12-08].

[7] NREL. HydroDyn User Guide and Theory Manual. https://openfast-wave-stretching.readthedocs.io/en/f-wave_stretching/source/user/hydrodyn/#, 2021. [Online; accessed 2021-11-16].

[8] A. Robertson, J. Jonkman, M. Masciola, H. Song, A. Goupee, A. Coulling, and C. Luan. Definition of the Semisubmersible Floating System for Phase II of OC4. Technical Report TP-5000-60601, NREL, Golden, Colorado, USA, 2014.

[9] T. M. Duarte, A. JNA Sarmento, and J. Jonkman. Effects of second-order hydrodynamic forces on floating offshore wind turbines. In *32nd ASME Wind Energy Symposium*. 01 2014. doi:10.2514/6.2014-0361.

Validation Tests for Morison Equation (ME)

The full Morison Equation (ME) model in QBlade (QB) (see Morison Equation) was validated in a series of test load cases using a semisubmersible substructure model. The load cases were chosen with increasing complexity to make sure the individual modules were working correctly. The test cases include decay tests, regular wave tests and irregular wave tests. Again, the OC4 ME model was considered rigid for these test cases in order to validate the hydrodynamic models.

The results were validated against the open-source aero-hydro-elastic code OpenFAST (OF) [1] (version 2.5.0). The same turbine models and test cases were setup and used in QB and OF. Not all of the hydroelastic modelling capabilities of QB are present in OF. When validating certain hydrodynamic models, the hydrodynamic capabilities of QB were adapted to reflect those of OF to have a better comparison. These modifications are mentioned in the appropriate load cases below.

OC4 Semisubmersible ME Model

The OC4 model was also modelled using the Morison equation only. The geometry of the model is shown in Fig. 188. The Morison equation modelling has the advantage of allowing distributed loading during the simulations and hence enable hydroelastic simulations (see Validation and Examples of Hydroelasticity). As a drawback, the hydrodynamic coefficients for the Morison equation are obtained empirically and are constant for all frequencies. This can lead to modelling inaccuracies compared to the LPMD approach, where the radiation, diffraction and excitation matrices are frequency dependent (see Linear Potential Flow Theory). The hydrodynamic coefficients for the OC4 ME model were taken from the OC4 Report [2] with additional coefficients taken from [3].

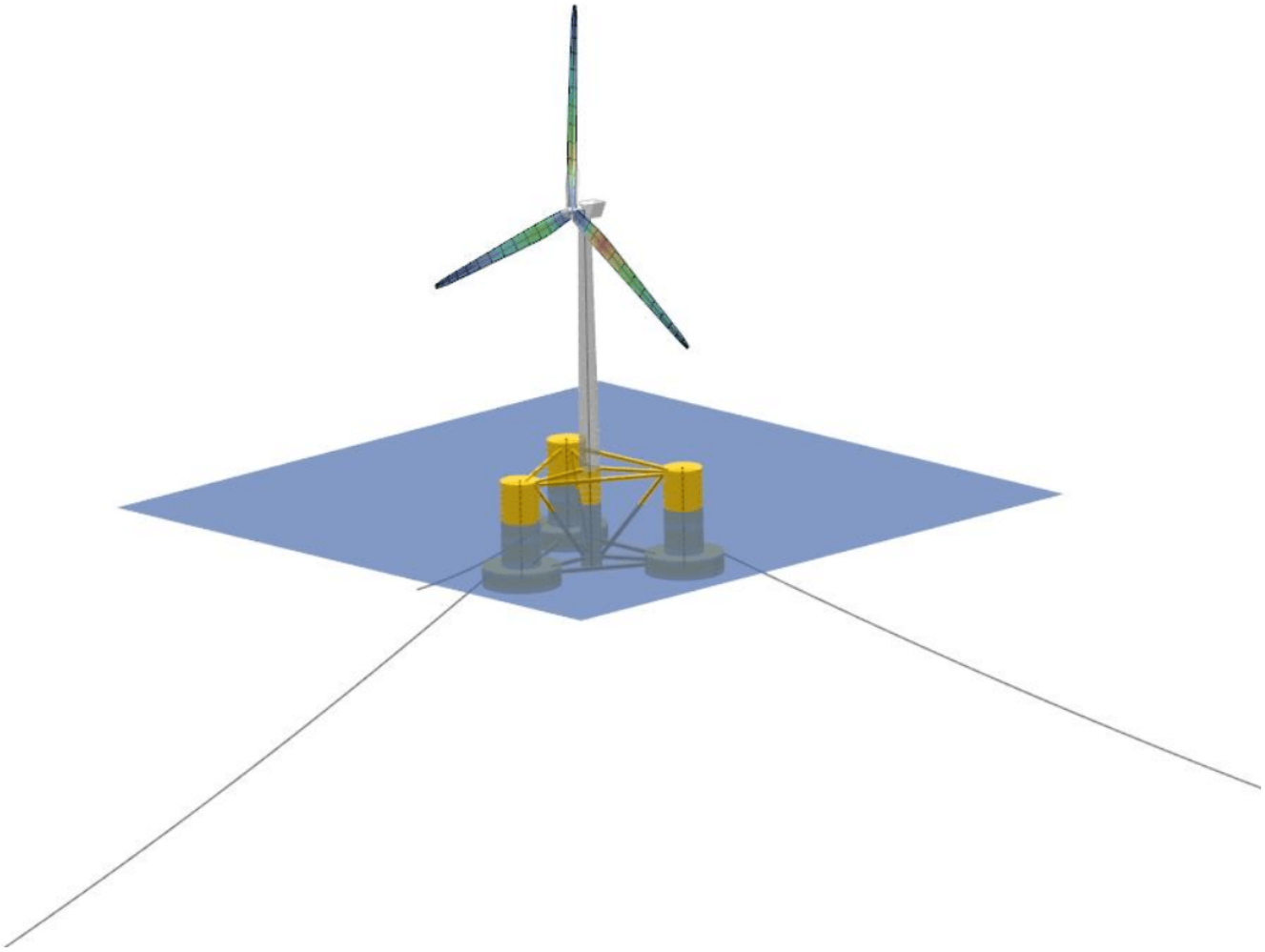


Fig. 188 *The OC4 model displayed in the QB GUI featuring a semisubmersible substructure.*

OC4 ME Free Decay Tests

The first test cases were free decay tests with still water. To validate the full Morison equation present in the OC4 ME model, an equivalent ME model was setup in OF. According to the reference [3], the ME model in OF is not able to account for distributed buoyancy. Hence, a linearized buoyancy was used in the OF calculations. This can have a significant effect in the behavior of the substructure when waves are present (see OC4 LPMD Regular Wave Tests).

Fig. 189 to Fig. 191 show the time series of the surge, heave and pitch decay tests. We can see that the models in both codes show a very similar decay behavior. There is a small offset in the heave position due to the different buoyancy models. Also, if we compare with the free decay test of the OC4 LMPD model (see OC4 LPMD Free Decay Tests), we can see that the oscillations damp out more slowly in the decay tests with the OC4 ME model. This is because the OC4 ME model does not have radiation damping forces which act as an additional linear damping term.

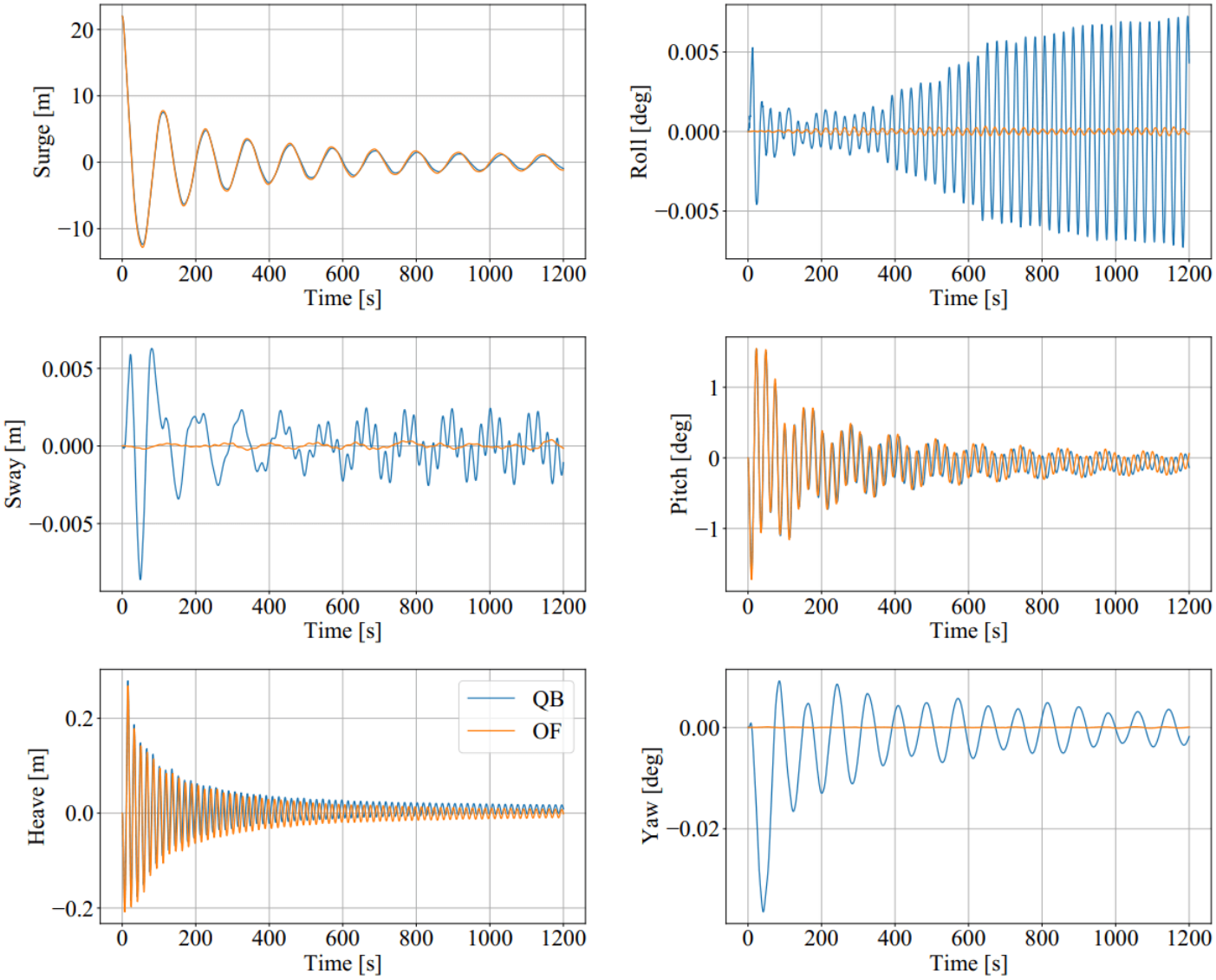


Fig. 189 *Time series of the OC4 model surge decay test.*

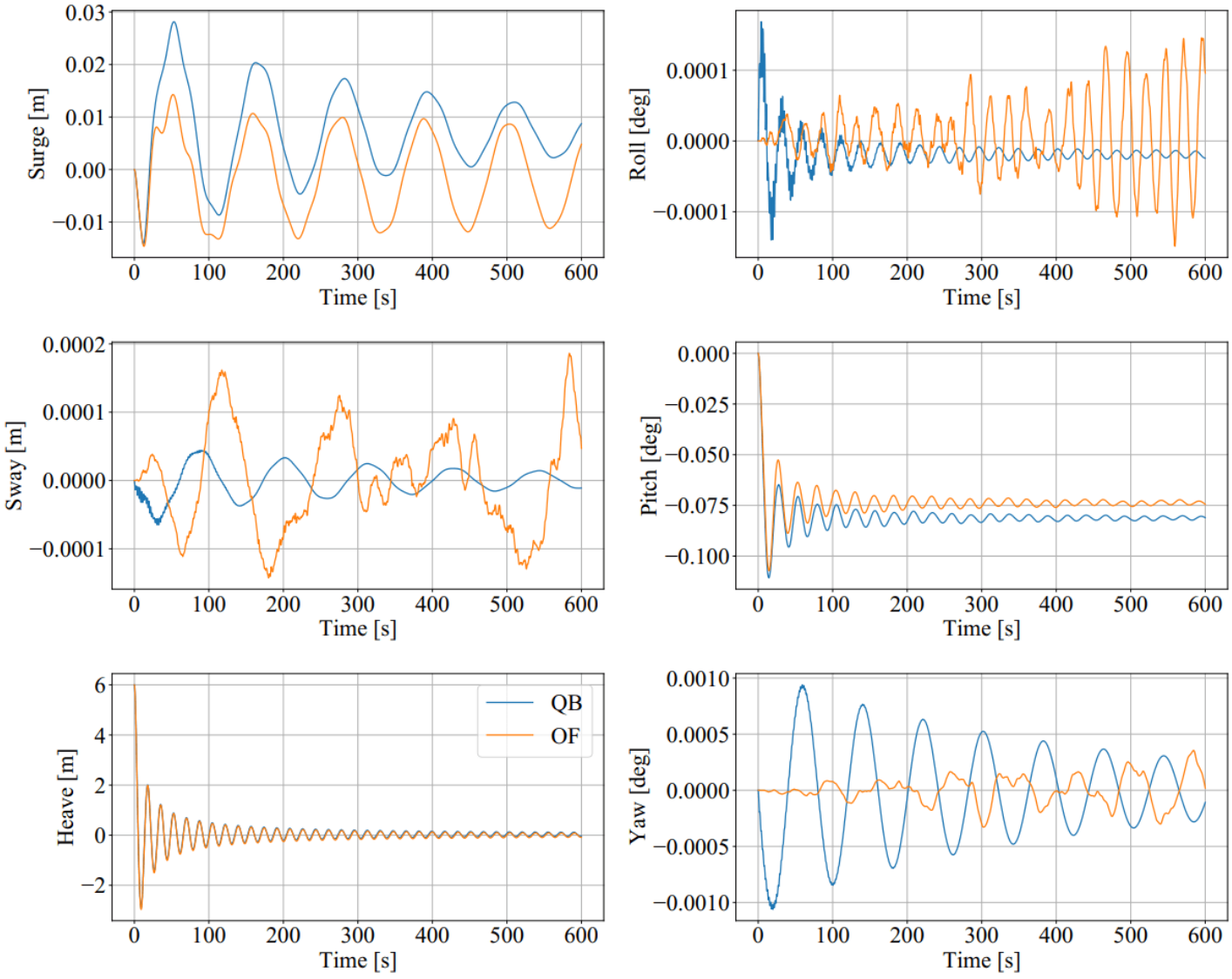


Fig. 190 *Time series of the OC4 model heave decay test.*

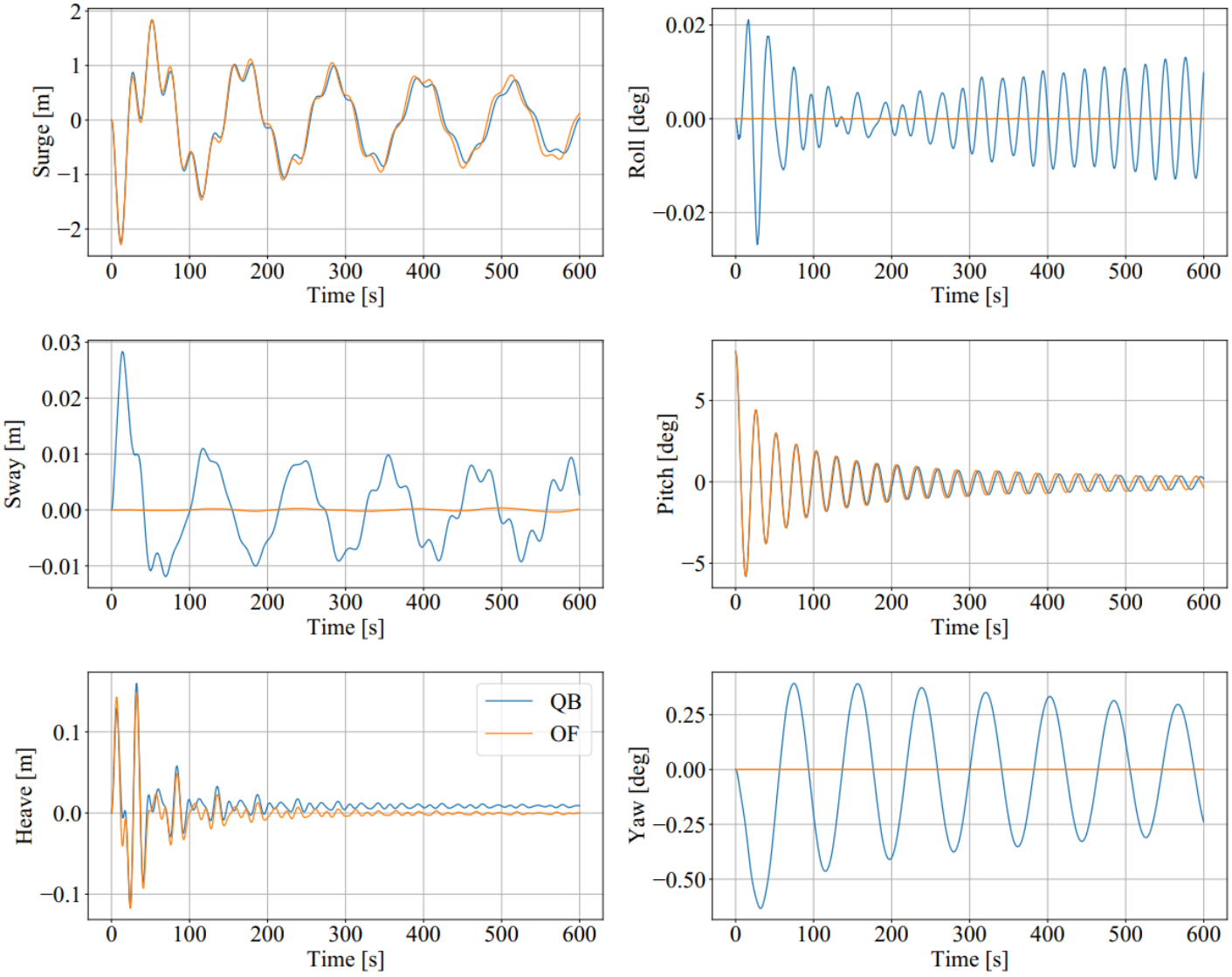


Fig. 191 *Time series of the OC4 model pitch decay test.*

We note that the tensions in the mooring system was validated for the OC3 and OC4 models in Validation Tests for Potential Flow Models with Morison Drag (LPMD), so it won't be repeated here.

Fig. 192 shows the numerical relative values of the eigenfrequencies and damping coefficients of the decay tests for the surge, heave, pitch and yaw degrees of freedom (DOFs). The eigenfrequencies and dampings were obtained according to the procedure presented in [4]. The linear damping term was ommited since there is no linear damping present in this model. We can see in this figure that the values for the frequencies and damping coefficients are very similar in both codes. There seems to be a discrepancy in the eigenfrequency of the surge DOF. This difference comes from the method we used to determine the eigenfrequency. For the surge DOF, the numerical value of the eigenfrequency is low and it is therefore close to the frequency resolution we used to determine it. In OF and QB, the peaks in the frequency transform of the signal were shifted in the frequency range by one resolution point. This already accounted for the difference seen in Fig. 192. Visual inspection of Fig. 18[illegible]dy gives an empirical proof that the frequencies of the surge decay test are very similar.

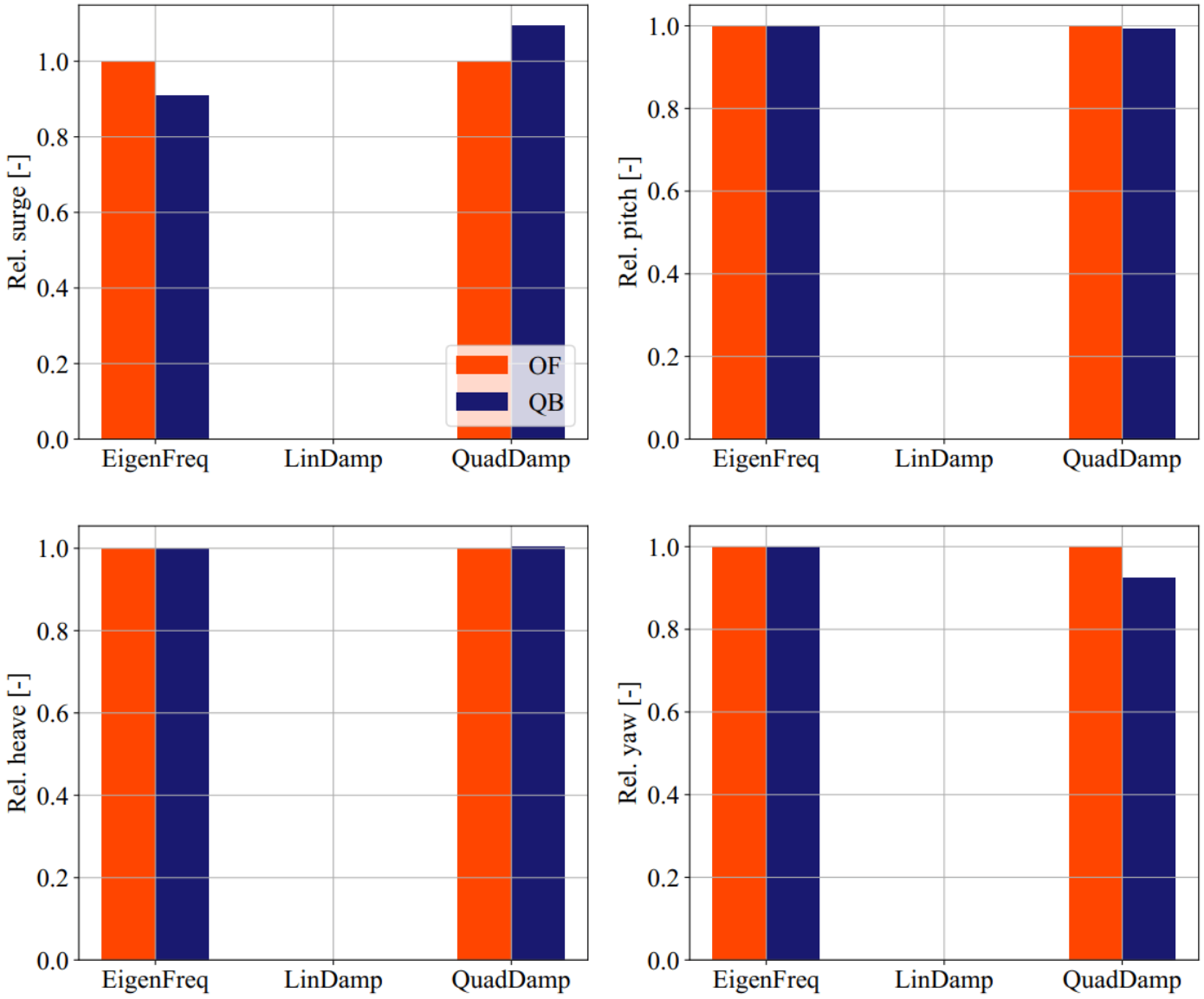


Fig. 192 *Normalized eigenfrequencies and damping behaviour of the OC4 model for the considered decay tests.*

OC4 ME Regular Wave Tests

The regular wave tests were performed with linear waves for two selected cases. One case had a wave height of H = 6 m and a period of T = 10 s. The second case had a wave height of H = 8 m and a period of T = 12 s.

For these cases, the OC4 ME was adapted to have a linearized buoyancy model and a linearized mooring system model. Additionally, the wetted surface was considered to go until the mean sea level instead of the local wave elevation (see Modeling Considerations for Morison Elements). This was done to better compare the QB model with the one present in the OF calculations. From test cases presented in Validation Tests for Potential Flow Models with Morison Drag (LPMD), we can consider the buoyancy and mooring models validated. By aligning the modelling considerations between QB and OF, we can better validate the full Morison model developed in QB.

Diffraction forces will play a role for Morison elements that have a diameter larger than a fifth of the wavelength of the incoming wave [5]. For the OC4 ME model, this would be relevant for the large base and upper columns if the turbine operates at low sea states [2]. In QB, the full Morison model can be extended with the MacCamy-Fuchs correction (MCFC) to take into account the diffraction effects [6].

The regular wave test cases considered three sea states: the first one characterized by H = 0.67 m and T = 4.8 s, the second by H = 6 m and T = 10 s and the third by H = 8 m and T = 12 s. The wave direction was aligned with the positive surge direction. According to [2], the diffraction forces will be relevant for the first sea state.

Fig. 193 to Fig. 195 show the surge, pitch and heave DOFs and the wave elevation for the three regular sea states. We can see in these figures that the results form OF and QB align fairly well in all three sea states. There are some small differences in the heave response in all three cases. When we enable the MacCamy-Fuchs correction in QB, we can see that especially the surge DOF is affected in Fig. 193 and Fig. 194. For the sea state with the smallest wave height, we see the largest differences between the models with and without the MCFC. For larger wave heights (Fig. 194), there are still some differences between the calculations with and without MCFC. These differences practically vanish for the largest wave height case (Fig. 195). This qualitative behavior corresponds to the expected behavior that the MCFC mostly affects sea states where the diameter of the Morison element is comparable to the wave length of the incoming wave.

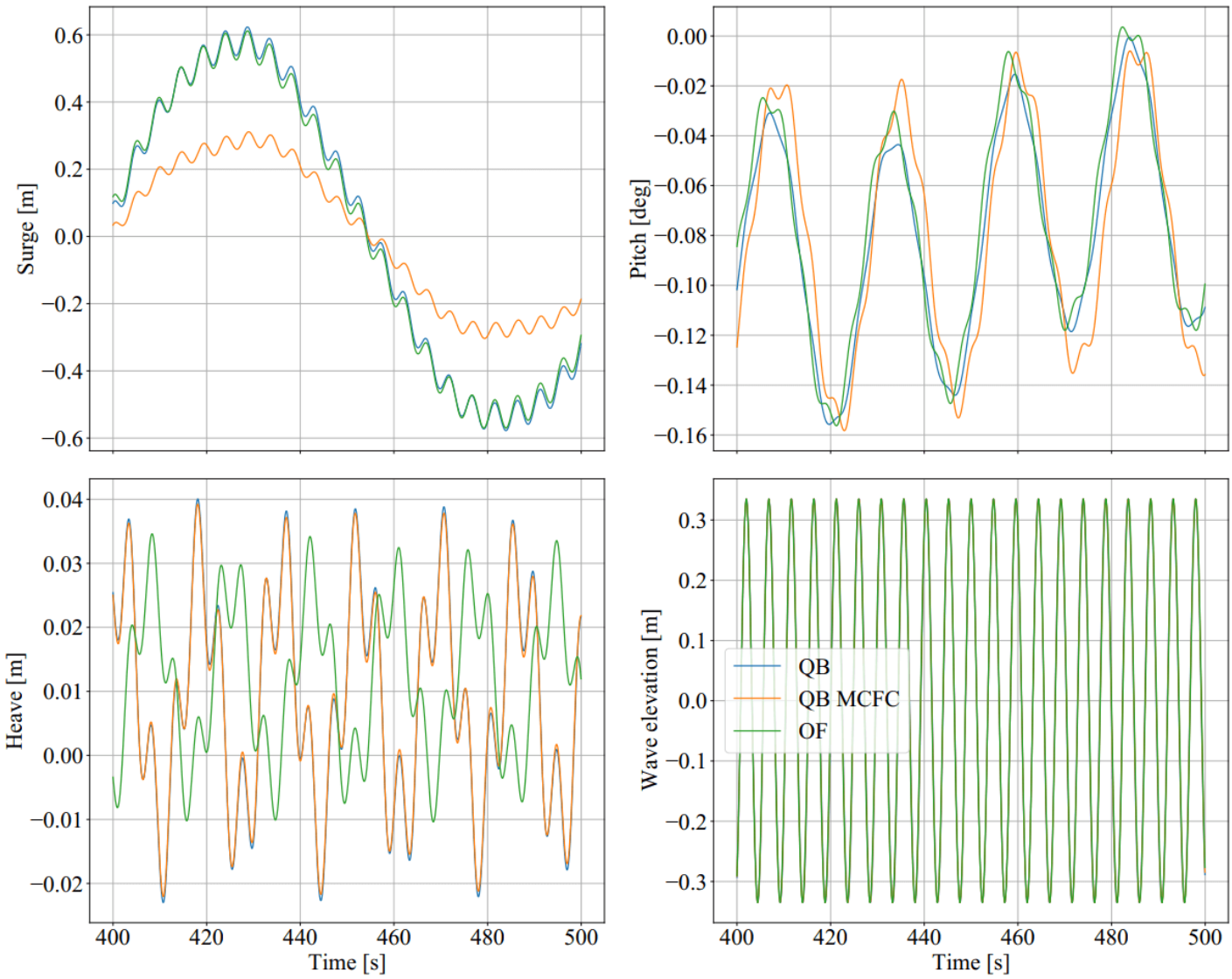


Fig. 193 *Relevant DOFs and wave elevation for regular sea state with H = 0.67 m and T = 4.8 s. QB MCFC = QB with MacCamy-Fuchs correction.*

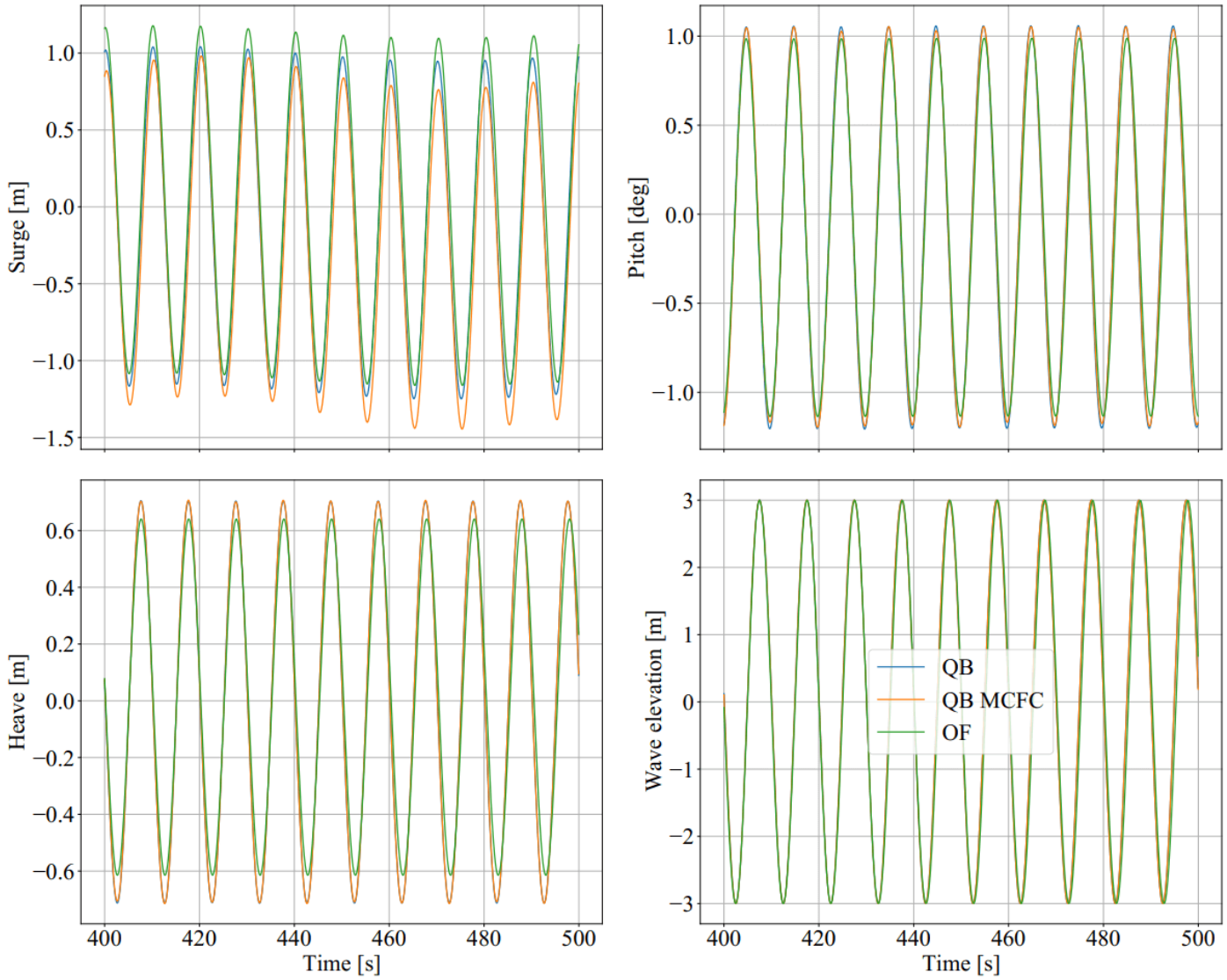


Fig. 194 *Relevant DOFs and wave elevation for regular sea state with H = 6 m and T = 10 s. QB MCFC = QB with MacCamy-Fuchs correction.*

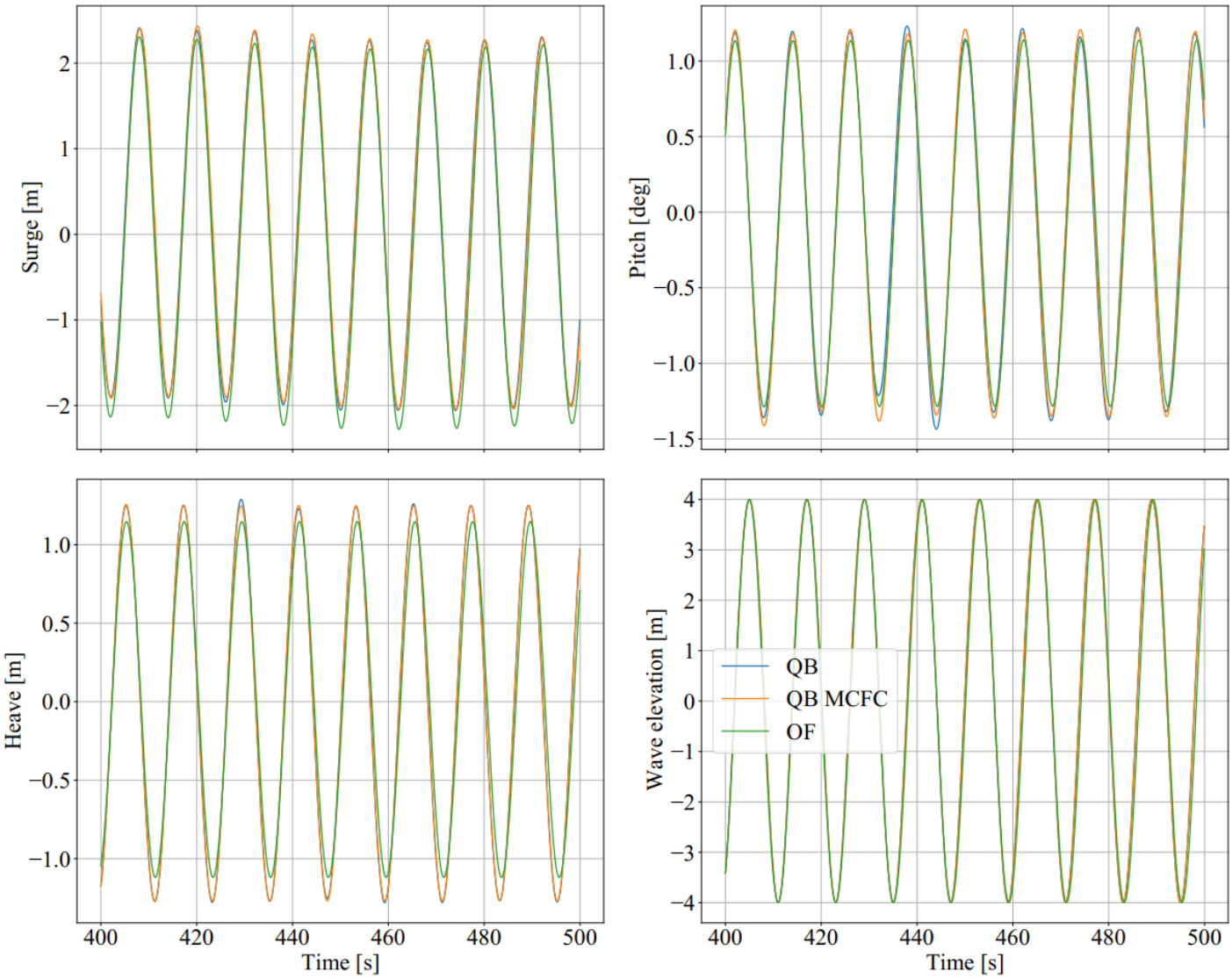


Fig. 195 *Relevant DOFs and wave elevation for regular sea state with H = 8 m and T = 12 s. QB MCFC = QB with MacCamy-Fuchs correction.*

OC4 ME Irregular Wave Tests

The OC4 ME was also tested in sea states with irregular waves and compared to the results from OF simulations. We used six stochastic sea states with a JONSWAP spectrum (H_s = 6, T_p = 10 s, γ = 3.3) and compared the averaged PSD of all DOFs. To have a good alignment of the modeling assumptions between QB and OF, we again used a linear buoyancy and a linear mooring model. Also, the wetted surface was assumed to go until the mean sea level and no MCFC was used in the QB simulations. For the irregular wave tests, the wave direction was aligned with the positive surge direction and no aerodynamic loads were considered.

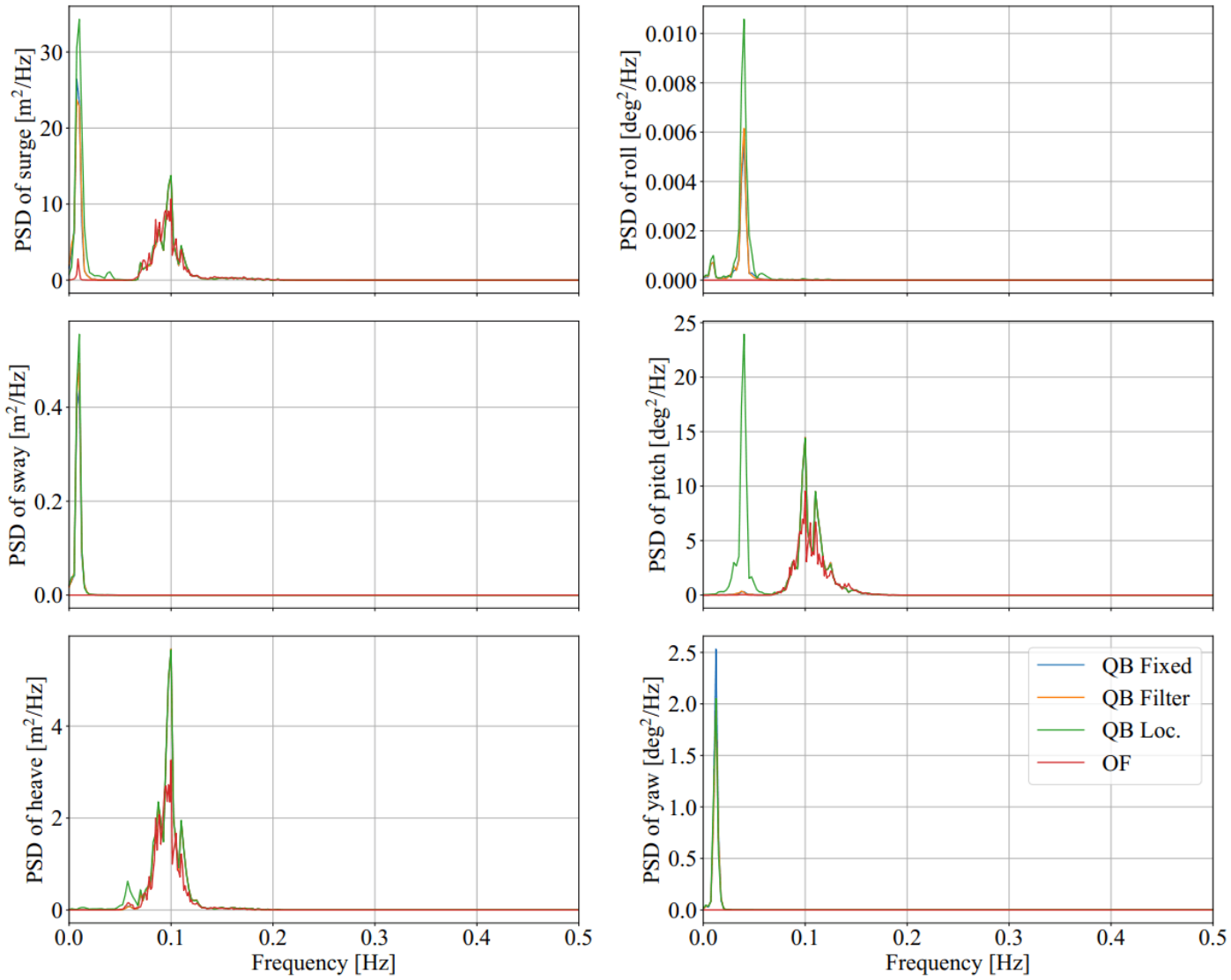


Fig. 196 *Averaged PSDs of all DOFs of the OC4 ME model for the irregular sea state with* H_s *= 6 m,* T_p *= 10 s and* γ *= 3.3.*

The comparison was done in a statistical manner by comparing the six-simulation-averaged PSD for the six DOFs. Fig. 196 shows the results of the irregular sea state test cases. We can see that the responses of the OF and QB simulations generally agree well. For the QB simulations, we considered simulations with the three ME implementation options (see Modeling Considerations for Morison Elements). The first option, QB Loc., considered the instantaneous local position of the Morison elements to calculate the water particle kinematics. The second option, QB Filter, considered the low-pass filtered position of the Morison elements to determine the water particle kinematics. The third option, QB Fixed, considered the fixed initial position of the Morison elements for the kinematic calculations. The last option is also implemented in OF [7].

We can see in Fig. 196 that all simulations have a comparable PSD behavior for the wave excitation frequencies (around 0.1 Hz). The higher peaks in the heave and pitch DOFs come from the higher response of the OC4 ME model to wave excitation forces around these frequencies (see e.g. Fig. 194). The strongest differences are seen for the QB Loc. and OF calculations in the low frequency range. The QB Loc. calculations show a peak in the eigenfrequencies of the pitch and surge DOFs

while the OF calculations do not. This nonlinear response in pitch disappears for the QB Filter and QB Fixed simulations. It can therefore be attributed to the local instantaneous approach when calculating the water kinematics. The surge DOF still shows a peak in the surge eigenfrequency even when filtered or fixed approach is used for calculating the water kinematics. Further investigation is required to fully understand this phenomenon.

[1] OpenFAST. OpenFAST. https://github.com/OpenFAST/openfast, 2021. [Online; accessed 2021-12-07].

[2] (1, 2, 3) A. Robertson, J. Jonkman, M. Masciola, H. Song, A. Goupee, A. Coulling, and C. Luan. Definition of the Semisubmersible Floating System for Phase II of OC4. Technical Report TP-5000-60601, NREL, Golden, Colorado, USA, 2014.

[3] (1, 2) F. Wendt, A. Robertson, J. Jonkman, and G. Hayman. Verification of new floating capabilities in fast v8. In *33rd ASME Wind Energy Symposium*. 01 2015. doi:10.2514/6.2015-1204.

[4] A. Robertson, J. Jonkman, F. Wendt, A. Goupee, and H Dagher. Definition of the OC5 DeepCwind Semisubmersible Floating System. https://a2e.energy.gov/data/oc5/oc5.phase2/attach/oc5.phase2.model.definitionsemisubmersible-floating-system-phase2-oc5-ver15.pdf, 2021. [Online; accessed 2021-11-11].

[5] O. M. Faltinsen. *Sea Loads on Ships and Offshore Structures*. Cambridge University Press, 1993.

[6] IEC61400-3-1 Standard. IEC 61400-3-1: Wind Energy Generation Systems - Part 3-1: Design Requirements for fixed offshore wind turbines. Standard, International Electrotechnical Commission, Geneva, Switzerland, 2019.

[7] NREL. HydroDyn User Guide and Theory Manual. https://openfast-wave-stretching.readthedocs.io/en/f-wave_stretching/source/user/hydrodyn/#, 2021. [Online; accessed 2021-11-16].

Validation and Examples of Hydroelasticity

In this page, examples of hydroelastic simulations and validations with multiple substructure geometries are presented.

OC4 Semisubmersible ME Hydroelastic Model

The OC4 Semisubmersible ME Model validated in Validation Tests for Morison Equation (ME) can also be modeled with fully elastic members. The elastic members, together with the full Morison equation (see Morison Equation) at each element allow us to have fully hydroelastic simulations within QB. The structural properties of the individual elements were taken from [1].

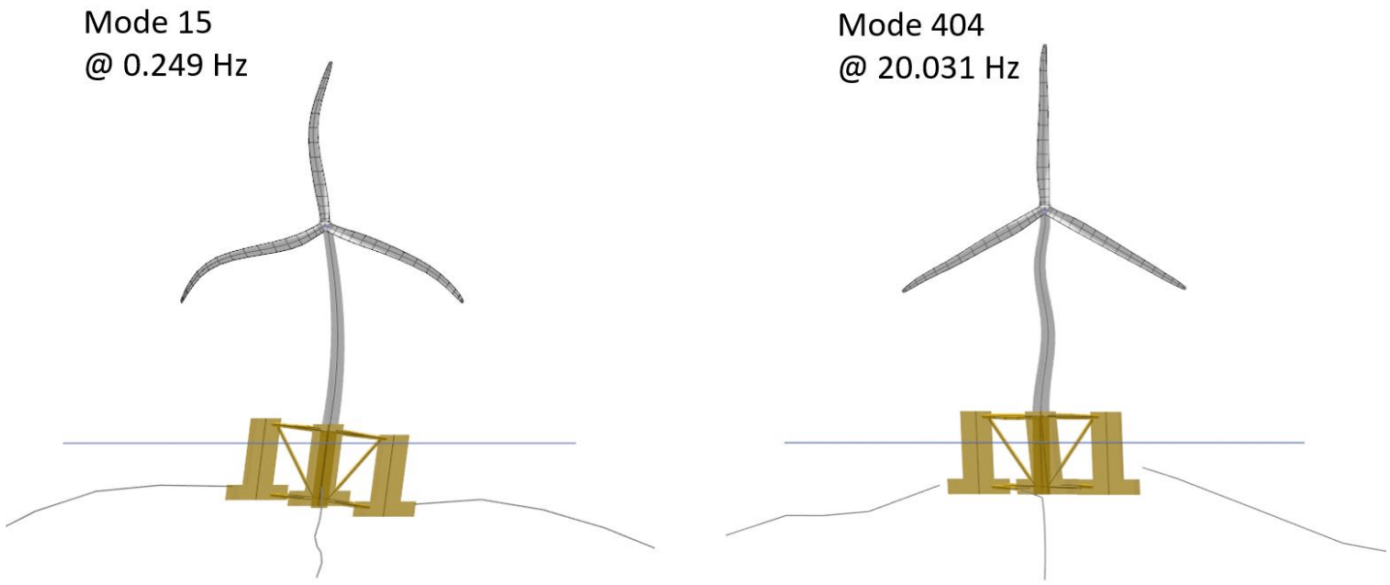


Fig. 197 *Two exemplary modes of the flexible OC4 ME model.*

Fig. 197 show two exemplary eigenmodes of the OC4 ME flexible model. Mode 15 on the left side has frequency 0.249 Hz and shows a combination of blade, tower and substructure deflections. For the substructure, it is mainly the cross-braces that are deflecting. Mode 404 on the right side of Fig. 197 has a frequency of 20.031 Hz. This mode shows a deflection of the three base and upper columns relative to each other. We note that there is also a tower deflection taking place in this mode. The high frequency of Mode 404 is also an argument for simulating the OC4 substructure as a rigid body if the local forces on the substructure are not required. The excitation frequencies from wind and waves are significantly lower than the eigenfrequencies of these modes. We also note that both modes shown in Fig. 197 also include the deflection of the mooring cables. All flexible [illegible]ers are taken into account when performing the eigenmode analysis within QB.

The flexible OC4 ME model was simulated in a regular sea state with H = 6 m and T = 10 s. No aerodynamic loads were applied on the turbine and the wave direction was chosen to coincide with the positive surge direction. The local loads for two locations were recorded: the lowest position of the main column and the connection point of the cross brace 1 with the upper column 1 (see Fig. 198).

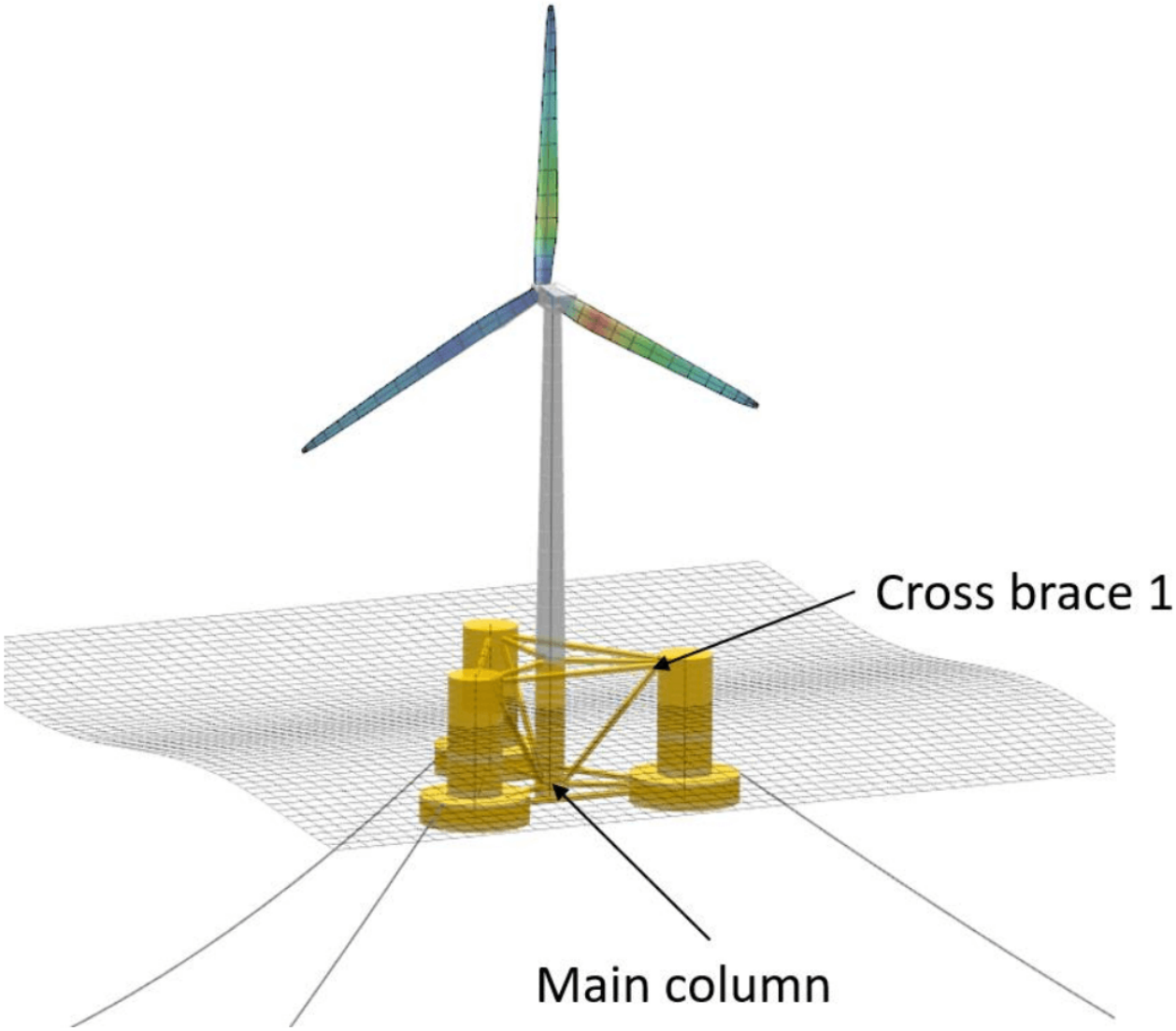


Fig. 198 *Load sensor locations for flexible OC4 ME calculations.*

Fig. 199 shows the local forces at the main column and the cross brace for the initial 400 s of the regular sea state simulation. We can see that the forces behave as expected. The source of the local forces at the main column bottom location arise mainly from the hydrodynamic and gravitational loads. The local force in the x-direction (surge) has an oscillatory behavior around 0 kNm, which can be explained by the oscillatory nature of the hydrodynamic wave loads in surge direction. This also explains the small forces in the y-direction (sway) since no wave loads are acting in this direction. Finally, in z-direction (heave), the oscillatory wave loads have a non-zero mean value. This co om the gravitational loads acting in the z-direction. These localized loads can be used to design the individual components of a floating wind turbine substructure.

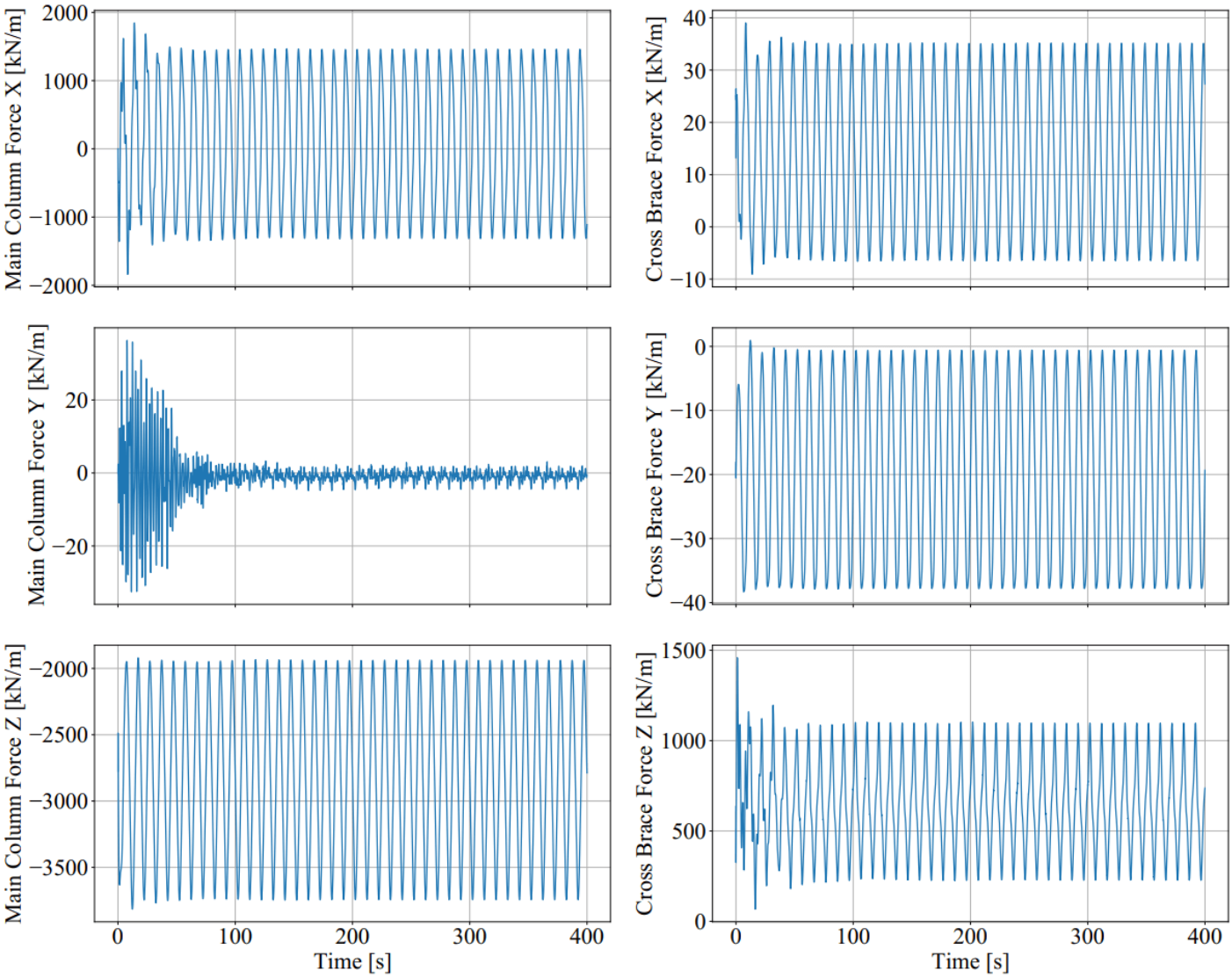


Fig. 199 *Local forces at two locations of the OC4 ME flexible substructure.*

Fig. 200 shows the corresponding local deflections of the main column and cross brace. We can see in this figure that the substructure deflections at these positions are very small, not even reaching 1 mm. This comes from the large stiff structures that make up the OC4 substructure. Fig. 200 shows that the rigid substructure assumption made for this model is a valid.

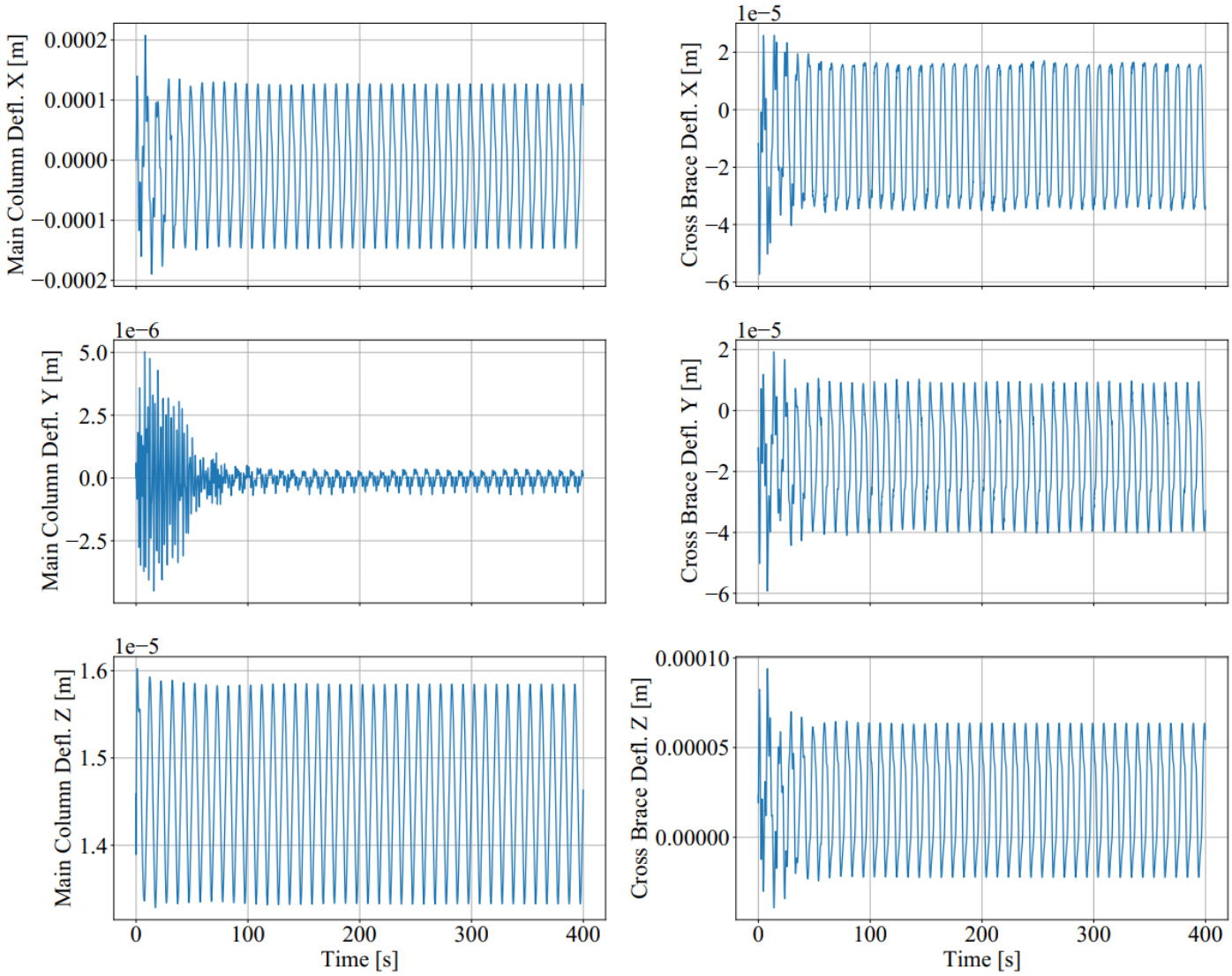


Fig. 200 *Local deflections at two locations of the OC4 ME flexible substructure.*

10 MW TetraSpar Hydroelastic Model

In Fig. 201, we show a hydroelastic model of a 10 MW turbine mounted on an up-scaled TetraSpar substructure [2]. The modelling of this substructure is particularly challenging because it features a flexible suspension between the upper floater structure and the lower keel. Within QB, this suspension system can be modelled as a cable element similar to the mooring system (Section XXX). The TetraSpar model shown in Fig. 201 is composed of flexible Morison elements and we are therefore capable of accurately calculating the local loading on each element even when the substructure model is highly deflected. An example of this is shown on the right hand side of Fig. 201. This extreme roll position is unlikely to happen in reality, but being able to simulate such conditions in QB is a good example of the modelling capabilities of the tool.

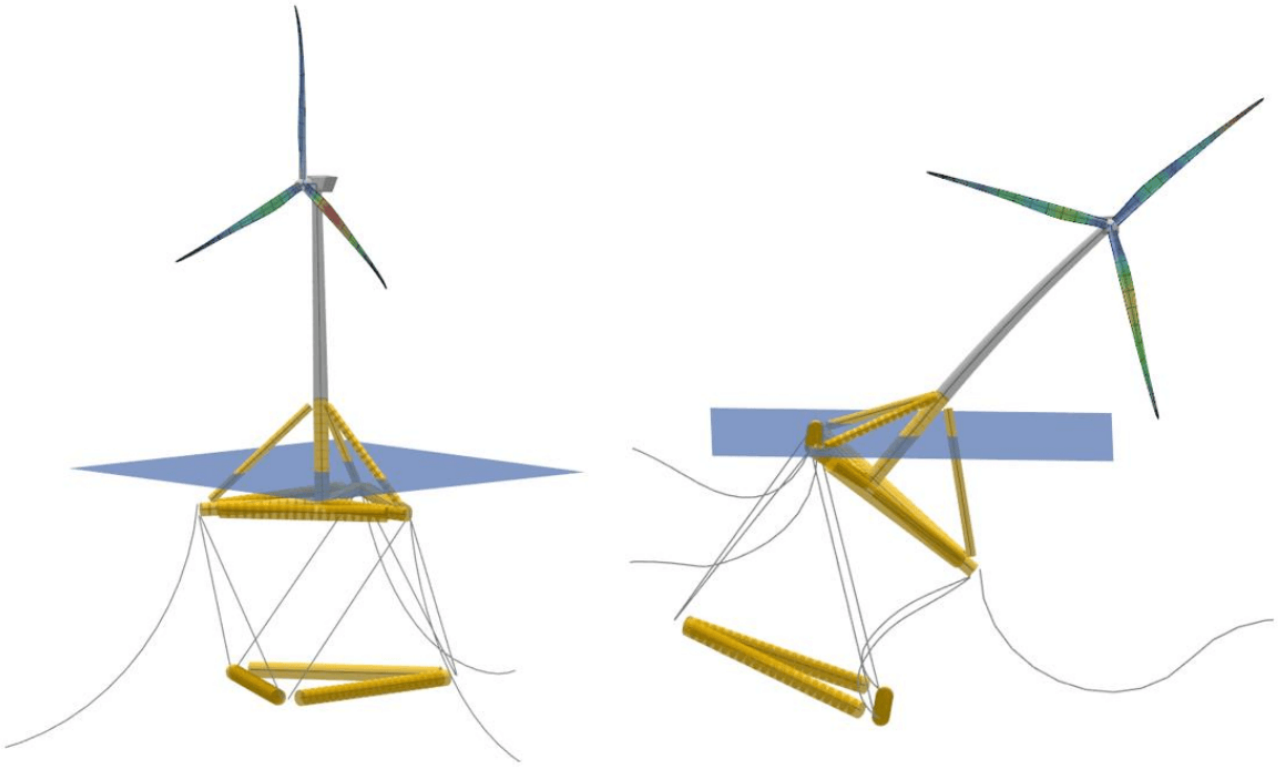


Fig. 201 *Hydroelastic model of a 10MW turbine on an up-scaled TetraSpar substructure.*

[1] DTU. HAWC2 OC4 Semisubmersible model. https://www.hawc2.dk/models, 2021. [Online; accessed 2023-05-19].

[2] Michael Borg, Morten Walkusch Jensen, Scott Urquhart, Morten Thøtt Andersen, Jonas Bjerg Thomsen, and Henrik Stiesdal. Technical definition of the tetraspar demonstrator floating wind turbine foundation. *Energies*, 2020. URL: https://www.mdpi.com/1996-1073/13/18/4911, doi:10.3390/en13184911.

QBlade Docs License

CC BY-NC-ND 4.0

All content of this documentation (docs.qblade.org) is licensed under the Creative Commons License CC BY-NC-ND 4.0. this license allows you to share (copy and redistribute) the material of the documentation in any medium or format under the following terms:

- **Attribution**: You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **Non Commercial**: You may not use the material for commercial purposes.
- **No Derivatives**: If you remix, transform, or build upon the material, you may not distribute the modified material.

For more details on this license see the license text below:

Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights. Effective Technological Measures mean those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996,

and/or similar international agreements. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license. Licensor means the individual(s) or entity(ies) granting rights under this Public License. NonCommercial means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning. Section 2 – Scope.

License grant. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to: reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and produce and reproduce, but not Share, Adapted Material for NonCommercial purposes only. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions. Term. The term of this Public License is specified in Section 6(a). Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material. Downstream recipients. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise Licensed Rights by any recipient of the Licensed Material. No endorsement. Nothing in this P License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by,

the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i). Other rights.

Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise. Patent and trademark rights are not licensed under this Public License. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes. Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

Attribution.

If You Share the Licensed Material, You must:

retain the following if it is supplied by the Licensor with the Licensed Material: identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated); a copyright notice; a notice that refers to this Public License; a notice that refers to the disclaimer of warranties; a URI or hyperlink to the Licensed Material to the extent reasonably practicable; indicate if You modified the Licensed Material and retain an indication of any previous modifications; and indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License. For the avoidance of doubt, You do not have permission under this Public License to Share Adapted Material. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable. Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only provided You do not Share Adapted Material; if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database. For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights. Section 5 – Disclaimer of Warranties and Limitation of Liability.

Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability. Section 6 – Term and Termination.

This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or upon express reinstatement by the Licensor. For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License. Sections 1, 5, 6, 7, and 8 survive termination of this Public License. Section 7 – Other Terms and Conditions.

The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License. Section 8 – Interpretation.

For the avoidance of doubt, this Public License does not, and shall not be interpreted to, red it, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License. To the extent possible, if any provision of this Public

License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

QBlade-CE License

Academic Public License

Academic Public license

(QBlade CE License v1.1)

This license outlines the terms and conditions for the non-commercial use of QBlade Community Edition (QBlade-CE), specifically tailored for academic institutions and individuals engaged in teaching, research, and personal or educational purposes. Users of QBlade-CE under this license are granted rights similar to those in the GNU General Public License (GPL) for non-commercial purposes. This license allows for the use of QBlade-CE at no cost in the specified non-commercial settings. Commercial usage of QBlade requires a separate commercial license. This includes, but is not limited to, usage in for-profit environments where research is conducted to develop or enhance products, commercial service offerings, and use by or in collaboration with commercial entities in government-funded, EU-funded, military, or similar research projects. Joint research projects with commercial entities are also prohibited under this license and require transitioning to the Enterprise Edition of QBlade (QBlade-EE). For information on obtaining a commercial license, or if you are uncertain about whether your intended use falls under this license's scope, please contact the QBlade authors at info@qblade.org or visit our website at https://qblade.org for further guidance.

In addition to the terms and conditions specified above, QBlade Community Edition (QBlade-CE) also functions as an evaluation version for QBlade Enterprise Edition (QBlade-EE). Commercial users are hereby granted permission to use QBlade-CE for a reasonably limited period of time solely for the purpose of evaluating its suitability for their needs. It is important to note that during this evaluation period, QBlade-CE may not be directly applied or used in any commercial task or project.

What are the rights given to noncommercial users? Similarly to GPL, you have the right to use the software, to distribute copies, to receive source code, to change the software and distribute your modifications of the modified software. Also similarly to the GPL, if you distribute verbatim or modified copies of this software, they must be distributed under this license.

By modeling the GPL, this license guarantees that you're safe when using QBlade-CE in your work, for teaching or research. This license guarantees that QBlade-CE will remain available free of charge for nonprofit use. You can modify QBlade-CE to your purposes, and you can also share your modifications. Even in the unlikely case of the authors abandoning QBlade-CE entirely, this l permits anyone to continue developing it from the last release, and to create further release this license.

We believe that the combination of noncommercial open-source and commercial licensing will be beneficial for the whole user community, because income from commercial licenses will enable faster development and a higher level of software quality, while further enjoying the informal, open communication and collaboration channels of open source development.

The precise terms and conditions for using, copying, distribution and modification follow.

ACADEMIC PUBLIC LICENSE

TERMS AND CONDITIONS FOR USE, COPYING, DISTRIBUTION AND MODIFICATION

0. Definitions

“Program” means a copy of QBlade-CE, which is said to be distributed under this Academic Public License.

“Work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Using the Program” means any act of creating executables that contain or directly use libraries that are part of the Program, running any of the tools that are part of the Program, or creating works based on the Program.

Each licensee is addressed as “you”.

1. Permission is hereby granted to use the Program free of charge for noncommercial purposes, including teaching and academic research at universities, colleges and other educational institutions and personal non-profit purposes. For using the Program for commercial purposes, including but not limited to consulting activities, design of commercial hardware or software networking products, or any form of commercial collaboration, including joint research with a commercial entity, government-funded, EU-funded, military or similar research projects, explicit commercial licensing is required. You have to contact the Author or visit https://qblade.org for an appropriate license. Permission is also granted to use the Program for a reasonably limited period of time for the purpose of evaluating its usefulness for a particular purpose. However, during this limited evaluation period permission is not granted to apply or use QBlade-CE in the context of a commercial task or project.

2. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and

to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 2 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files, the nature of the changes and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose regulations for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. (If the same, independent sections are distributed as part of a package that is otherwise reliant on, or is based on the Program, then the distribution of the whole package, including but not restricted to the independent section, must be on the unmodified terms of this License, regardless of who the author of the included sections was.)

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based or reliant on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 3) in object code or executable form under the terms of Sections 2 and 3 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 2 and 3 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a fee for a fee no greater than your incurred costs of physically performing the distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 2 and 3 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b) above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. Nothing else grants you permission to modify or distribute the Program or its derivative works; law prohibits these actions if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License and all its terms and conditions for copying, distributing or modifying the Program or works based on it, to do so.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by th
parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then you must not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

9. If the distribution and/or use of the Program are restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded by the original copyright holder. In such case, this License incorporates the limitation as if written in the body of this License.

NO WARRANTY

10. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

11. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED ON IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

The Academic Public License is initially written by Andras Varga (in public domain) and has been adapted by David Marten for the distribution of QBlade-CE. This adaption of the license text is also licensed under the CC0 Public domain license.

QBlade-EE License

The Enterprise Edition of QBlade (**QBlade-EE**), is distributed under a commercial license, that explicitly allows the commercial application of the software. The license text of this commercial QBlade Enterprise Edition Software License Agreement is available upon request from license@qblade.org.

Floating and Node-Locked License Files

Commercial licenses of **QBlade Enterprise Edition** are implemented through license files. There are two types of license files: floating and node-locked.

Floating License Files

Floating license files necessitate an active internet connection to validate the license. This validation occurs via an HTTPS request to a licensing service provider. Each floating license is allocated a certain number of seats. Within these constraints, any number of QBlade-EE instances can run concurrently on a single machine for each available seat. However, once all allocated seats are in use, no additional QBlade instances can be initiated on new machines until a seat becomes available by terminating all QBlade instances on a machine previously using a seat.

For continuous operation of QBlade-EE, periodic license validation checks require an uninterrupted internet connection.

Debugging Floating License Activation Issues

Instances where QBlade-EE, once activated, crashes leading to a non-release of a license seat, thereby occupying a license unjustly, can arise. In such a case the Command Line Interface (CLI) can be used to manually free the license seat or quire additional information. The following functionality exists:

`QBladeEE -cli GET_MACHINES`

Through the **GET_MACHINES** argument, the application can list all currently activated machines for this floating license.

`QBladeEE -cli DEACTIVATE:MACHINE_ID`

Supports deactivation of the machine that is identified by **MACHINE_ID**, with a special case for deactivating all machines activated through this license (**DEACTIVATE:ALL**). **Note:** Deactivation results in a failed license validation upon the next check for the deactivated machine, potentially disrupting ongoing simulations.

`QBladeEE -cli LICENSE_DEBUG`

Activates a debug mode providing detailed insights into the license activation and troubleshooting process.

Resolving OpenSSL Issues on Windows

On certain Windows machines, the SIL (dll) version of QBlade-EE may encounter issues with initializing the OpenSSL libraries. These problems typically arise when the system cannot locate the necessary OpenSSL libraries in the QBlade directory. To fix this, you can add the QBlade directory to the system's PATH variable with the following steps:

1. **Open the Start Search Bar**: Type "System Environment Variables" and select "Edit the system environment variables" from the search results.
2. **System Properties Window**: In the System Properties window, click the "Environment Variables" button near the bottom of the Advanced tab.
3. **Edit the PATH Variable**: In the Environment Variables window, scroll to find the 'Path' variable under the 'System variables' section and select it. Then click 'Edit...'.
4. **Add QBlade Directory**: In the Edit Environment Variable window, click 'New' and paste the full path to the QBlade directory where the OpenSSL libraries (libssl-1_1-x64.dll and libcrypto-1_1-x64.dll) are located.
5. **Save and Exit**: Click 'OK' to close each window, ensuring your changes are saved.

After completing these steps, restart Windows to allow the changes to take effect. This should resolve the issue with initializing the OpenSSL libraries in QBlade's SIL interface.

Node-Locked License Files

Unlike floating licenses, node-locked licenses are tied to a specific hardware ID, eliminating the need for an internet connection. This license is uniquely issued for and restricted to the designated machine. Although node-locked licenses support offline operation, they lack the flexibility of floating licenses and are unsuitable for cloud computing or transferring between machines.

Node-locked licenses are ideally suited for standalone installations in secure or isolated environments where internet connectivity is a concern, or for users who prefer a simple, one-time licensing process without the need for ongoing management. They are also beneficial for organizations that prioritize software security and license control, as they provide a straightforward approach to licensing without the complexities of network-based checks or validations.

By choosing a node-locked license, organizations can ensure a robust and secure licensing mechanism that aligns with their specific operational needs and constraints, albeit with the understanding that such licenses offer less operational flexibility compared to their floating counterparts.