Incorporation of floater rotation and displacement in a static wind farm simulator

Riva, Riccardo; Pedersen, Mads M.; Pirrung, Georg; Bredmose, Henrik; Feng, Ju

Published in:
The Science of Making Torque from Wind (TORQUE 2024): Offshore wind

Link to article, DOI:
10.1088/1742-6596/2767/6/062019

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

Incorporation of floater rotation and displacement in a static wind farm simulator

Journal of Physics: Conference Series **2767** (2024) 062019 doi:10.1088/1742-6596/2767/6/062019

Incorporation of floater rotation and displacement in a static wind farm simulator

Riccardo Riva, Mads M. Pedersen, Georg Pirrung, Henrik Bredmose and Ju Feng

Department of Wind and Energy Systems, Technical University of Denmark, Frederiksborgvej 399, 4000 Roskilde, Denmark

E-mail: {ricriv, mmpe, gepir, hbre, jufen}@dtu.dk

Abstract. Thanks to the development of several wake models, recent years have seen a huge development of static wind farm simulators. As new wind farms are planned for deep waters, there is a need to extend these tools to account for the floater rotation and displacement. In this work, we have addressed this task by selecting a reduced set of environmental conditions and generating a database from medium-fidelity simulations, whose computational cost has been reduced by developing a new static solver. Then, we trained neural networks and embedded them in `PyWake` via a model-agnostic interface. Finally, the wake deflection due to rotor tilt has been modeled by extending a state-of-the-art model. The results indicate that the rotor can move downwind by up to 10% of the diameter and rotate in tilt by 3° at rated wind speed.

1 Introduction

In the past few years, static wind farm simulators like `FLORIS` [1] and `PyWake` [2] have flourished, and allowed to estimate wind farms' annual energy production fast and accurately. This is due to the development of several wake and blockage models, tuned against countless experiments and simulations [3, 4]. The transition from bottom-fixed to floating offshore wind farms yields a need for simulation tools suitable for wind farm layout and control optimization that take into account the motion of the wind turbine floater due to wind, waves, and current [5]. Static wind farm simulators employ remarkably simple turbine models, where the performance is described by power and thrust coefficient as a function of the wind speed. On the other hand, the motion of the floater is determined by the balance between hydrodynamic, inertial, and aerodynamic loads, which depends on the turbine and floater design. For this reason, it is important to achieve a balance between computational complexity and simplicity of representation. The static displacement of the floater means that the wind farm layout varies with the inflow and sea conditions, while the static floater pitch results in skewed inflow to the wind turbine and vertical wake deflection, similarly to the effects of rotor yaw. Furthermore, active yaw control may lead to more individual platform displacement and horizontal wake deflection. These effects modify the wind farm power and loads. In this study, however, only power production is considered.

Previous works on this topic were based on engineering consideration, which limits their applicability to the considered type of floater and does not leverage the general topology allowed by some multi-body simulators [6]. The concept of slightly changing the wind farm layout depending on the wind speed and direction has been previously explored in [7].

In this work, we will include the floater motion in `PyWake`, by employing an artificial neural network trained using medium-fidelity simulations, comprised of a multi-body model with BEM aerodynamics.

The benefit of this approach is that it can accommodate floaters of arbitrary topology, and leverage the physics of the dynamic simulator. Furthermore, neural networks are fast to evaluate and when they use smooth activation functions they enable gradient-based optimization. The neural networks are evaluated through the model-agnostic `surrogates_interface` package [8]. The computational cost of the medium-fidelity simulations is lowered by using a newly developed static solver in the `HAWC2` multi-body software. Finally, the floater tilt is modeled by extending a state-of-the-art engineering wake deflection model.
This manuscript is organized as follows. Sec. 2 illustrates the turbine model, the `HAWC2` static solver algorithm, the procedure for training the neural networks, the `surrogates_interface` package, the extension of the wake deflection model to include rotor tilt and finally how the floating wind farm power is computed. Sec. 3 shows the results, which comprise the effect of floater pitch on the power, how much the turbine moves, and a wind farm AEP calculation. Finally, in Sec. 4 we will draw the conclusions.

2 Methodology

2.1 Turbine modeling

For this study, we have selected the IEA 15 MW Reference Wind Turbine, with the WindCrete spar-buoy floating platform [9, 10]. The turbine properties are summarized in Tab. 1. The sea has a standard depth of 200 m, and the floater is anchored to the sea bed with 3 mooring lines, each having two anchor points at the fairlead. The turbine is shown in Fig. 1, and is modeled in the `HAWC2` multi-body software [11]. The blades and tower are modeled via Timoshenko beam elements, while nonlinear deformations are handled via the floating frame of reference formulation. Structural couplings in the blades are captured through the fully populated stiffness matrix. The mooring lines are modeled using cable elements. The aerodynamics is rendered through the Blade Element Momentum theory, including modifications for non-straight blades [12, 13]. The aerodynamic model includes a skewed inflow model that accounts for both the influence of the total angle between the inflow and the rotor disk, see Section 2.5, on the mean induction and of the yaw and tilt angles individually on the azimuthal variation of induced velocities. The hydrodynamics is included through the Morison equation. The model is completed by the DTU Wind Energy Controller [14], where the well-known floater-pitch instability [15] has been fixed by reducing the frequency of the regulator mode in the full load region, similarly to [10].

Table 1: Properties of the IEA 15 MW RWT.

Parameter	Value
Rated power	15 MW
Rated wind speed	10.59 m/s
Rotor diameter, D	240 m
Design tip-speed ratio	9.0

2.2 `HAWC2` static solver

It is common practice to initialize a simulation from an initial condition where the structural members are undeflected. As the external loads are applied, the system receives a step-like forcing that excites the turbine modes. Since several modes are low frequency and lightly damped, it requires at least 2000 seconds to reach an equilibrium condition for steady wind and sea. For this work, we have used the newly implemented static solver of `HAWC2` which computes the nonlinear static solution and starts the simulation from that, this way avoiding most of the initial transients. The static solver is described in Algorithm 1. The factorization and solution of the sparse linear system are done through the Intel `oneMKL PARDISO` library. At the time of writing, the static solver does not yet support external controllers, therefore the simulation is performed with fixed pitch and rotor speed. The pitch and rotor speed look-up table is obtained by performing dynamic simulations from the cut-in to the cut-out wind speeds with no current.
The simulation is then carried out with steady external conditions using a sparse Newmark solver [16]. The BEM method is implemented on a polar grid [12], but the static solution is currently obtained for the fixed initial blade location, with azimuthal interpolation of elastic blade deflection from blade locations to the polar grid points. For uniform inflow without tilt, yaw, or shear, the static solution will

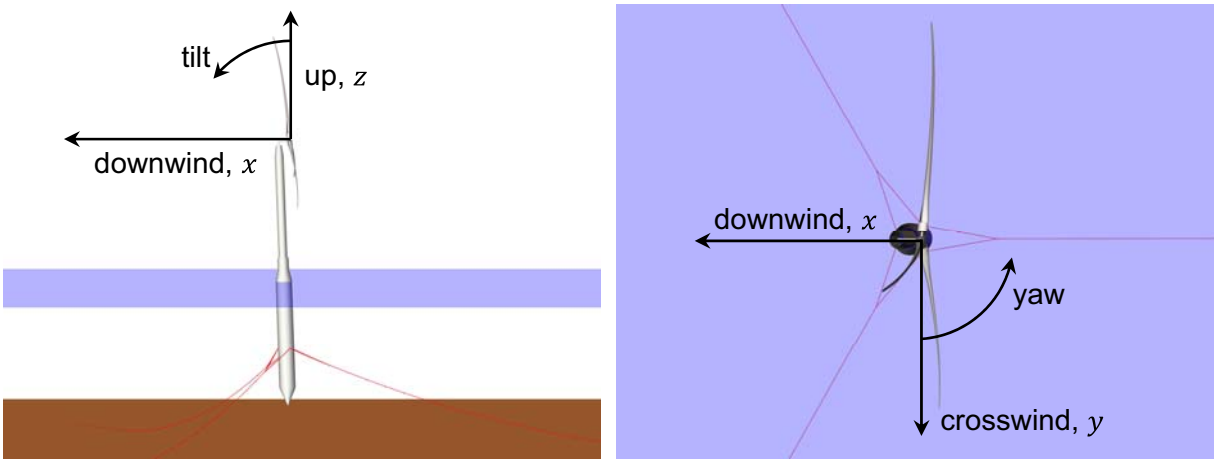

Figure 1: Visualization of the IEA 15 MW RWT, with `PyWake` coordinate frame.

Algorithm 1 `HAWC2` static solver.

Initialize system.
Compute hydrodynamic added stiffness matrix.
Allocate and compute sparse stiffness matrix $\boldsymbol{K}$.
while not converged and $n_{\text{iterations}} < n_{\max}$ **do**
  Converge Blade Element Momentum theory.
  Update stiffness matrix.
  Compute residual $\boldsymbol{r}$, which includes all external loads.
  Solve linear system $\boldsymbol{K}\boldsymbol{\Delta x} = \boldsymbol{r}$.
  Update nodes using the increment $\boldsymbol{\Delta x}$.
end while

match the result obtained by a dynamic simulation. For non-uniform inflow, the static solution is not perfect, but close enough to the equilibrium that 200 seconds are sufficient for the minor initial transients to dampen out. The computational time for the initial static solution is negligible compared to the dynamic simulation, so this implies an order of magnitude reduction in the computational cost.

2.3 Surrogate model training

Setting the turbulence intensity (TI) and waves to 0 allowed us to quickly reach equilibrium conditions using `HAWC2`, and obtain seed-independent results. The slight increase in power due to turbulence is neglected in this study. The vertical wind shear exponent is set to 0.1, to model a generic offshore site. The `HAWC2` and surrogate model inputs are: wind speed V, wind direction θ, current speed W and current-wind misalignment ϕ. The yaw bearing is clamped to achieve a nominal yaw angle of 0. Therefore, the measured yaw is due to the flexibility of the structure. Since the wind is not uniform, nor parallel to the shaft, and gravity is present, the turbine is anisotropic. It follows that the equilibrium is not constant, but to a more or lesser extent periodic, which required to average the response over the final periods of the simulations. The surrogate model output is composed of the average: downwind, crosswind, and upward displacements of the rotor center, the rotor tilt, yaw, electrical power, and thrust coefficient. The reference frame for the output is the one used by `PyWake`, shown in Fig. 1. Its origin is in the rotor center for no wind and current.

A database of input conditions has been generated by following the *wind to loads* procedure [17]. We have assumed that each input variable follows a uniform distribution. The wind speed bounds are [3, 25] m/s, while the current speed bounds are [0, 2] m/s. Both the wind direction and current-wind misalignment bounds are [0, 360] deg. The input domain is evenly covered by sampling nearly $6 \cdot 10^4$ points from the distribution using a Halton sequence with scrambling, as implemented in `OpenTURNS` [18, 19]. The `HAWC2` static solver failed to converge in about 6% of the cases. Improving its robustness will be an object of future research.

In the following, we will indicate an input point with $\boldsymbol{x} = (x_j) \in \mathbb{R}^{n_x}$. All points will be collected in the input array $\boldsymbol{X} = (x_{i,j}) \in \mathbb{R}^{n_p \times n_x}$, where index i denotes the point number and j the feature. For consistency, we will indicate a measured output point with $\boldsymbol{y} = (y_j) \in \mathbb{R}^{n_y}$. The output array is then $\boldsymbol{Y} = (y_{i,j}) \in \mathbb{R}^{n_p \times n_y}$. A column of the input array is indicated as $\boldsymbol{X}_j$, while one of the output array as $\boldsymbol{Y}_j$. The input and output databases are then split into 70% for training, 15% for validation, and the remaining 15% for testing.
Since two input features are angles, it is convenient to transform a point from $\boldsymbol{x} = (V, \theta, W, \phi)$ to $\tilde{\boldsymbol{x}}$ as

$$\begin{aligned} \tilde{x}_1 &= V\cos(\theta) & \tilde{x}_3 &= W\cos(\phi) \\ \tilde{x}_2 &= V\sin(\theta) & \tilde{x}_4 &= W\sin(\phi) \end{aligned} \tag{1}$$

In this way, 0° and 360° become coincident, rather than be as far as possible from each other.
Before training the surrogates, the input and output datasets are transformed with a gain-offset scaler. By introducing a gain array $\boldsymbol{\gamma}$ and an offset array $\boldsymbol{o}$, an input point is transformed as

$$\bar{\boldsymbol{x}} = \boldsymbol{\gamma} \odot \tilde{\boldsymbol{x}} + \boldsymbol{o} \tag{2}$$

where $\odot$ is the element-wise product. Output points are transformed analogously. A min-max scaler is then obtained by setting the gain for feature j as $\gamma_j = (x_{\max} - x_{\min})/(\max(\boldsymbol{X}_j) - \min(\boldsymbol{X}_j))$, and the offset as $o_j = x_{\min} - \min(\boldsymbol{X}_j)\gamma_j$. A standard scaler is instead obtained by computing the mean m_j and standard deviation σ_j for each feature. The gain is then set to $\gamma_j = 1/\sigma_j$ and the offset to $o_j = -m_j/\sigma_j$. To avoid information leak, all scalers are fitted on the training set, using `scikit-learn` [20].
As a surrogate model type, we have chosen a feed-forward neural network with fully-connected layers. The ANN is implemented in `TensorFlow` [21] and trained using the Adam optimizer.
The input dataset $\boldsymbol{X}$ is first transformed using Eq. (1), and then transformed with a min-max scaler with bounds $[-1, +1]$. All displacements and rotations are transformed using a standard scaler. The electrical power presents unique challenges and has been the subject of several articles. Without ad hoc techniques, it is in fact common to obtain oscillations in the full load region. This problem was addressed in [22] with the polynomial chaos expansion by introducing the logistic transformation. However, for neural networks, we developed a more practical solution. First, the electrical power is transformed with a min-max scaler having $\min(\boldsymbol{Y}) = 0$ and bounds $[-1, +1)$. Then, the tanh activation function is applied to the output layer. This way, the predicted electrical power will be between 0 and the rated value for any input. Similarly, the thrust coefficient is transformed with a min-max scaler having $\min(\boldsymbol{Y}) = 0$, $\max(\boldsymbol{Y}) = 1$ and bounds $[-1, +1]$. By applying again the tanh activation function to the output layer, it is ensured that $0 < C_T < 1$.
We have optimized the number of hidden layers and neurons using `Optuna` [23] and a grid search optimizer. The search space is composed of 1 to 4 hidden layers and 5 to 30 neurons. As objective functions, we have considered the Mean-Squared Error (MSE) on the training and validation set and the ratio of model parameters to points in the training set. The latter expresses in quantitative form Occam's razor principle, and thanks to the large number of simulations it always remains small. Training the ANN for 10^5 epochs provides a low MSE, while over-fitting is prevented by stopping the training if the MSE on the validation set does not decrease anymore (which in our case never occurred). The smoothness of the MSE is guaranteed by a default learning rate of 10^{-3} and a batch size as large as the training set. The accuracy of the ANN is evaluated using the MSE and R^2 metrics computed on the test set and shows very good accuracy for all outputs.

2.4 The `surrogates_interface` package

Several surrogate model types have been invented over the years, such as Polynomial Chaos Expansion, Kriging, and Artificial Neural Networks. Each one has been implemented in various libraries, and although these libraries offer similar functionalities, they all provide a slightly different interface. We have developed the open source `surrogates_interface` [8] package to provide a common interface to several libraries, so that `PyWake` users could use their favorite one. At the time of writing, this package supports `TensorFlow` and `SMT` [24].
The surrogate model output is computed as $\boldsymbol{y} = \boldsymbol{\psi}_y^{-1}(\boldsymbol{f}(\boldsymbol{\psi}_x(\tilde{\boldsymbol{x}}))$, where $\boldsymbol{\psi}_x$ is the input transformation, $\boldsymbol{f}$ the surrogate model and $\boldsymbol{\psi}_y^{-1}$ the inverse output transformation. This operation, and its analytical gradient, are implemented in `surrogates_interface`.

2.5 Extension of the wake deflection model to account for floater pitch

The wake deflection model by Jiménez [25] has been extended to also model vertical wake deflection due to rotor tilt and floater pitch, such that the wake behind a floating wind turbine will deflect upwards if the turbine tilts backward due to for example the rotor thrust.
The horizontal and vertical rotations of the rotor are combined into a single angle, $\theta_{total,angle}$, with direction $\theta_{total,direction}$, see Fig. 2, right.

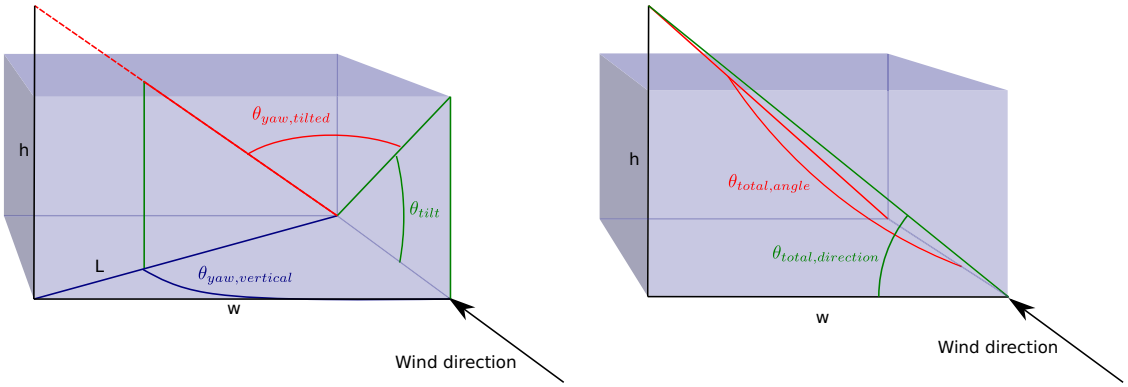

Figure 2: Definition of rotor yaw and tilt.

For an onshore turbine, the orientation of the rotor is defined in terms of tilt and yaw, where tilt is the angle between the shaft and the horizontal plane and yaw is the angle between the inflow wind direction and the shaft around the vertical (tower) axis. For floating turbines, it may be convenient to define the tilt as the platform pitch plus rotor tilt and yaw as the angle between the inflow wind direction in the tilted frame of reference. The yaw angle around the vertical axis, $\theta_{yaw,vertical}$ can be obtained from the yaw angle in the tilted frame of reference, $\theta_{yaw,tilted}$ by

$$\theta_{yaw,vertical} = \arctan\left(\frac{\tan(\theta_{yaw,tilted})}{\cos(\theta_{tilt})}\right) \tag{3}$$

We can now calculate w, L and h (normalized by the depth of the box) in Fig. 2 left,

$$w = \tan\left(\theta_{yaw,vertical}\right) \tag{4a}$$
$$L = \sqrt{w^2 + 1} \tag{4b}$$
$$h = L\tan(\theta_{tilt}) \tag{4c}$$

Finally, the angle and the direction of the combined angle are obtained by

$$\theta_{total,angle} = \arctan\left(\sqrt{h^2 + w^2}\right) \tag{5a}$$
$$\theta_{total,direction} = \arctan2\left(h, w\right) \tag{5b}$$

From $\theta_{total,angle}$, the amount of wake deflection, $\Delta total$ is calculated by integrating the rate of deflection, as defined by Jiménez [25], from the rotor to the point of interest.n The deflection rate depends on yaw angle, thrust coefficient, normalized downstream distance, and a tuning parameter. Finally, the lateral and vertical projection of the deflection is found by

$$\Delta y = \cos\left(\theta_{total,direction}\right) \cdot \Delta total \tag{6a}$$
$$\Delta z = \sin\left(\theta_{total,direction}\right) \cdot \Delta total \tag{6b}$$

In `PyWake`, the deflection is incorporated by adding the deflection of the wake from turbine, i, to turbine, j, $\Delta S_{ij} = (0, \Delta y, \Delta z)$ to the "real" turbine interdistance, $S_{ij} = (x_j - x_i, y_j - y_i, z_j - z_i)$ before passing it to the wake deficit model.

2.6 Incorporation of the surrogate model into `PyWake`

To utilize the platform displacement and floater rotation obtained from the surrogate model, the iterative `All2AllIterative` wind farm model of `PyWake`, see Algorithm 2, must be used. The reason is that the problem contains a cyclic variable dependency: The wake, ΔU_{ij} from an upstream turbine, i, to a downstream turbine, j, depends on the distance between the turbines including displacement of the floaters, $S_{ij} + \Delta S_{ij}$, and the floater displacement of turbine j depends on the wind speed including wakes, $U_{w,j} = U_\infty - \Delta U_{ij}$. Algorithm 2 shows the updated simulation procedure after incorporation of the surrogate model to account for mean static displacement and rotation of the floating platforms. We indicate with u_∞ the free-stream wind speed. The wind speed at each wind turbine, after including wake effects is indicated with $\boldsymbol{u_w}$. The pitch and yaw angles of each floater are $\boldsymbol{\theta_{\text{pitch}}}$ and $\boldsymbol{\theta_{\text{yaw,tilted}}}$. The thrust coefficient and power of each turbine are $\boldsymbol{C_T}$ and $\boldsymbol{P}$. $\boldsymbol{S_{xij}}$ is the distance in downwind, crosswind, and vertical direction between turbine i and j, while $\boldsymbol{\Delta S_{xij}}$ is the floater displacement in downwind, crosswind, and vertical direction between turbine i and j. Finally, $\boldsymbol{\tilde{S}_{xij}}$ is the distance between turbine i and j in the deflected wake frame of reference.

Algorithm 2 `All2AllIterative` wind farm model including floater motion.

$\boldsymbol{u_w} \leftarrow u_\infty$
$\boldsymbol{S_{xij}} \leftarrow \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}^T & \boldsymbol{y} - \boldsymbol{y}^T & \boldsymbol{z} - \boldsymbol{z}^T \end{bmatrix}$
$\boldsymbol{\theta_{pitch}} \leftarrow 0$
while not converged **do**
  $\boldsymbol{C_T} \leftarrow \mathit{WindTurbine.C_T}(\boldsymbol{u_w}, \boldsymbol{\theta_{\text{pitch}}}, \boldsymbol{\theta_{\text{yaw}}})$
  $\boldsymbol{\Delta S_{xij}}, \boldsymbol{\theta_{\text{pitch}}}, \boldsymbol{\theta_{\text{yaw}}} \leftarrow \mathit{Surrogate}(\boldsymbol{u_w}, \theta, \phi, W)$
  $\boldsymbol{\tilde{S}_{xij}} \leftarrow \mathit{DeflectionModel}(\boldsymbol{u_w}, \boldsymbol{C_T}, \boldsymbol{S_{xij}} + \boldsymbol{\Delta S_{xij}}, \boldsymbol{\theta_{\text{pitch}}}, \boldsymbol{\theta_{\text{yaw}}})$
  $\boldsymbol{\Delta U_{ij}} \leftarrow \mathit{DeficitModel}(\boldsymbol{u_w}, \boldsymbol{ct}, \boldsymbol{\tilde{S}_{xij}})$
  $\boldsymbol{u_w} \leftarrow U_\infty - \sum_i \boldsymbol{\Delta U_{ij}}$
end while
$\boldsymbol{P} \leftarrow \mathit{WindTurbine.power}(\boldsymbol{u_w}, \boldsymbol{\theta_{\text{pitch}}}, \boldsymbol{\theta_{\text{yaw}}})$

3 Results

To measure the dependency of the electrical power from the floater pitch, we have set up a series of `HAWC2` simulations with fixed bottom. Fig. 3 (left) shows the electrical power as a function of the floater pitch for a few wind speeds below rated. Naturally, the maximum power is obtained when the shaft is aligned with the wind. This is obtained at about $-6°$, because of the design rotor tilt angle and wind shear. The exact floater pitch varies with the thrust because of the flexibility of the support structure. The dependency of the power from the floater pitch is more pronounced at higher wind speeds (below rated), which indicates that the impact on the AEP depends on the Weibull distribution. Fig. 3 (right) shows the power curve with the floater pitch set to 0°, in blue, and the one obtained by evaluating the neural network in orange. As expected, the difference between the two (green) shows the largest differences in correspondence of the maximum thrust at the rated wind speed.

Analyzing the database has revealed the most dominant functional dependencies, some of which will be shown hereafter, by evaluating the neural networks.

Fig. 4, shows the downwind displacement and rotor tilt as a function of the wind speed for aligned current. We can clearly observe that their trend is similar to the one of the thrust and that the displacement grows with the current speed. For this specific turbine model, the rotor tilt instead does not depend on the current speed and is shown for no current. The maximum downwind displacement is obtained at the rated wind speed, where it reaches $10\%D$. As mentioned earlier, the shaft tilt implies that the tilt starts from about 6° at the cut-in wind speed, and reaches a maximum of 9.5° at rated.

Fig. 5 shows the crosswind displacement and yaw angle with respect to the current speed for some current-wind misalignment. Their amount is governed by the angle between the thrust, the floater drag and the reaction force provided by the mooring lines. The maximum crosswind displacement is about $\pm 3\%D$, while the maximum yaw $\pm 0.5°$. The upward displacement (not shown for brevity) reaches 0.5 m and is thus negligible.

As a wind farm model, we have selected the one described in [26], which allows the conservation of mass and momentum and a Gaussian shape for the wake profile. The wake expansion is a function of the local turbulence intensity. A visualization of the flow map is shown in Fig. 6. As expected, the wake

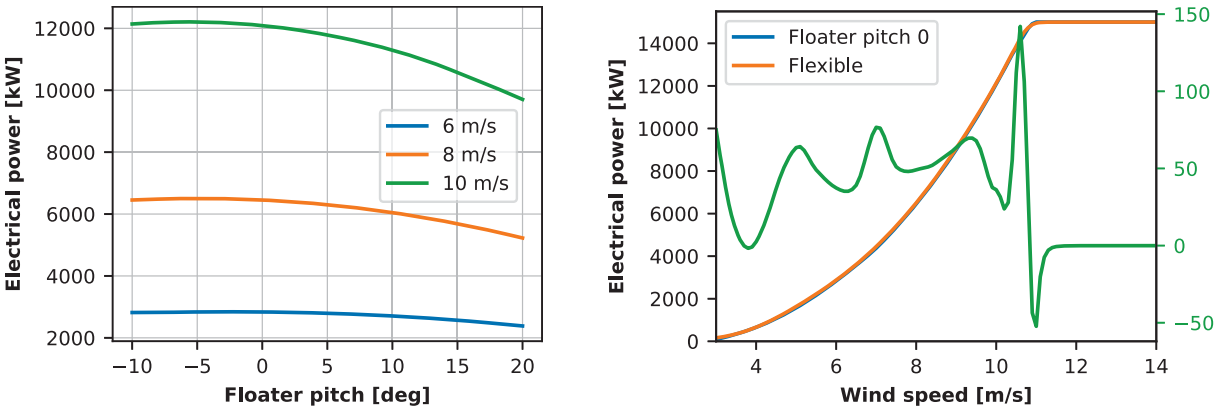

Figure 3: Effect of the floater pitch on the electrical power for different wind speeds (left). Comparison between the electrical power obtained for floater pitch fixed at 0° and the flexible one (right).

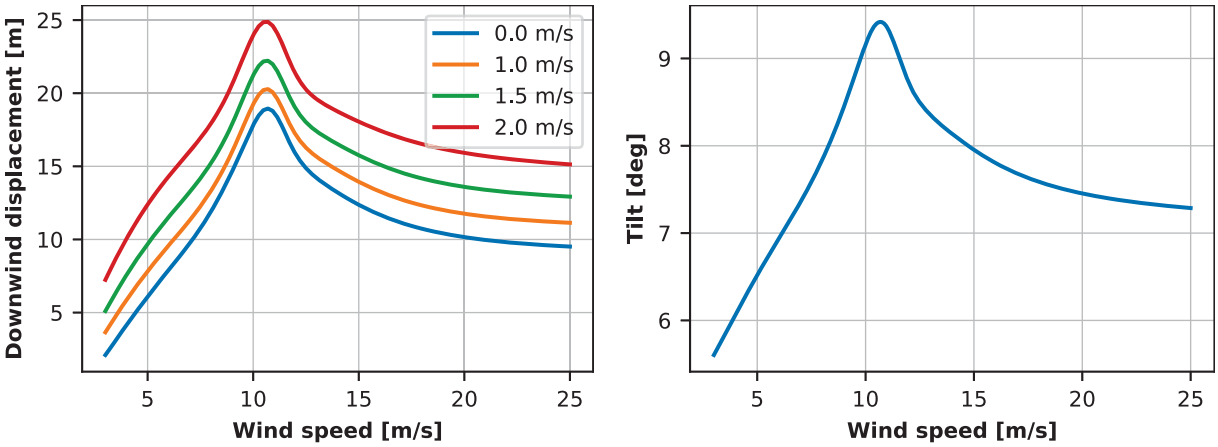

Figure 4: Downwind displacement and rotor tilt as a function of wind speed for $\theta = \phi = 0°$ and some current speeds.

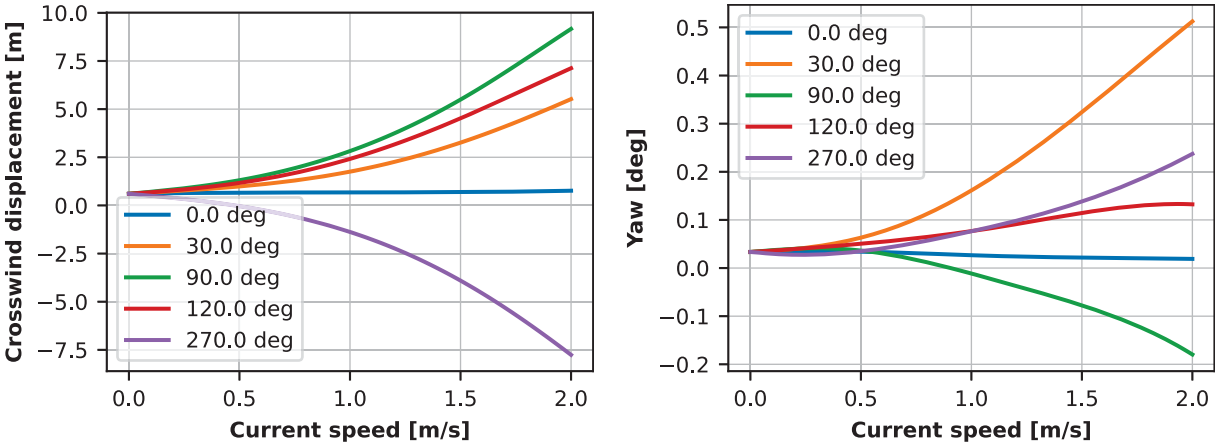

Figure 5: Crosswind displacement and rotor yaw as a function of current speed for $V = 10$ m/s, $\theta = 0°$ and some current-wind misalignment.

deflects upward for positive rotor tilt and downward for negative tilt. The flow field is far from uniform and suggests the usage of a rotor average model when evaluating wind farm performance, such as a circular Gauss integration.

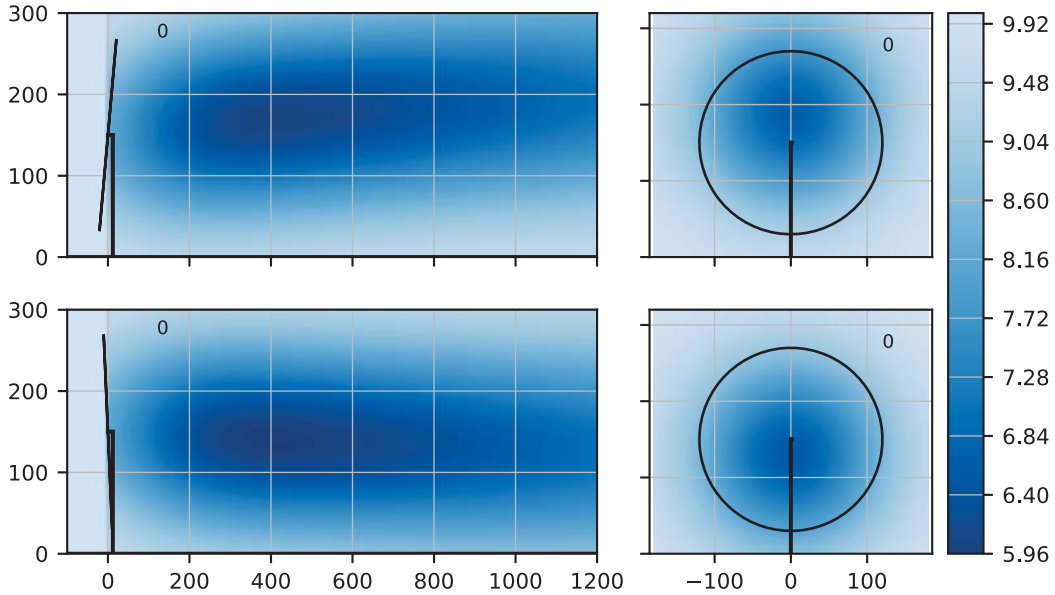

Figure 6: Wind speed with rotor tilt set to 10° (top), and −5° (bottom). For both cases, the free stream wind speed is 10 m/s and the TI 5%. The wind speed in the front view (right) is measured $4D$ downstream. Wind shear is not included.

To observe the impact of the floater motion on the Annual Energy Production (AEP), we have set up a simple wind farm. 4 IEA 15 MW are placed at the vertices of a rectangle, with sides $4D$ and $5D$. The Weibull distribution is taken from the Horns Rev 1 site, while the wind rose is uniform and the TI set to 5%. For the bottom-fixed case, with the floater pitch set to 0°, we obtain an AEP of 302.76 GWh. Enabling the floater motion modifies the AEP, depending on the current speed and direction. We have considered 60 different combinations of current speed in [0, 2] m/s and current direction in [0°, 360°), and observed that the AEP increased between 0.07% and 0.08%. This result depends however on the farm layout and turbulence intensity. A single AEP evaluation for the bottom-fixed case takes about 9.5 seconds on a standard laptop, while introducing the floater motion causes the computational time to increase by a factor of 2.5. The reader can consult [27] for a floating wind farm optimization, and additional considerations on the computational cost. The increased computational cost appears to be due to the slower convergence of the `All2AllIterative` algorithm. Improving its speed will be the object of future research.
In conclusion, the rotor displacement and orientation modify the wake and in turn the AEP. Since the impact is relatively small, and the increase in computational time large, it is recommended to include this feature only in the final phase of a wind farm layout optimization. The AEP change might increase when wind farm control is introduced [28]. For example, the yaw error of upstream turbines will change not only the wake deflection but also the turbine position.

4 Conclusions

In this work, we have extended `PyWake` to simulate floating wind farms. This has been accomplished by training neural networks that provide the rotor displacement and orientation, the electrical power, and the thrust coefficient as a function of the environmental conditions. The networks have been trained using `HAWC2` simulations, whose computational cost has been dramatically reduced by developing a new static solver. The spurious oscillations of the electrical power, common to many surrogate models, have been eliminated by combining a min-max scaler with the tanh activation function applied to the output layer. The networks are evaluated using a new model-agnostic library. Finally, the wake deflection due to floater tilt has been modeled by extending Jiménez's model.

The methodology has been applied to the IEA 15 MW reference wind turbine. It was found that the current speed and direction have a minor impact on the turbine position and power, but that is nonetheless significant in the context of wind farm layout and control optimization.
Including intentional yaw angle errors, for example to achieve wake deflection away from a downstream turbine, will induce a crosswind displacement of the upstream turbine that will cancel out a part of the desired wake deflection. Thus, modeling floater displacement could be essential for the development of advanced controllers for floating wind farms.
The atmospheric turbulence and wave motion cause the induction to change dynamically, which in turn affects the wake expansion. This aspect, as well as the current implementation, will be verified in the future using CFD simulations.

Acknowledgements
This research was supported by: the IDEA project funded by the Danish Energy Technology Development and Demonstration Program (EUDP) under project number 134-21029, the COREWIND project funded by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 815083, and IEA Wind.
The authors gratefully acknowledge the computational and data resources provided by the Sophia HPC Cluster at the Technical University of Denmark, DOI: 10.57940/FAFC-6M81.
The authors would like to acknowledge Anders Melchior Hansen for developing the `HAWC2` static solver.

References

[1] Fleming P, King J, Bay CJ, Simley E, Mudafort R, Hamilton N, et al. Overview of FLORIS updates. Journal of Physics: Conference Series. 2020 9;1618(2):022028.

[2] Pedersen MM, van der Laan P, Friis-Møller M, Forsting AM, Riva R, Romàn LAA, et al.. PyWake. Zenodo; 2023.

[3] Archer CL, Vasel-Be-Hagh A, Yan C, Wu S, Pan Y, Brodie JF, et al. Review and evaluation of wake loss models for wind energy applications. Applied Energy. 2018 9;226:1187-207.

[4] Göçmen T, Laan Pvd, Réthoré PE, Diaz AP, Larsen GC, Ott S. Wind turbine wake models developed at the technical university of Denmark: A review. Renewable and Sustainable Energy Reviews. 2016 7;60:752-69.

[5] Jacobsen A, Godvik M. Influence of wakes and atmospheric stability on the floater responses of the Hywind Scotland wind turbines. Wind Energy. 2021 2;24(2):149-61.

[6] Kheirabadi AC, Nagamune R. Real-time relocation of floating offshore wind turbine platforms for wind farm efficiency maximization: An assessment of feasibility and steady-state potential. Ocean Engineering. 2020 7;208:107445.

[7] Mahfouz MY, Cheng P. A passively self-adjusting floating wind farm layout to increase the annual energy production. Wind Energy. 2023 3;26(3):251-65.

[8] Riva R, Pedersen MM. Surrogate Models Interface. Zenodo; 2023. Available from: `https://gitlab.windenergy.dtu.dk/surrogate-models/surrogate-models-interface`.

[9] Gaertner E, Rinker J, Sethuraman L, Zahle F, Anderson B, Barter G, et al. Definition of the IEA 15-Megawatt Offshore Reference Wind Turbine. International Energy Agency; 2020.

[10] Ramos-García N, Kontos S, Pegalajar-Jurado A, González Horcas S, Bredmose H. Investigation of the floating IEA Wind 15 MW RWT using vortex methods Part I: Flow regimes and wake recovery. Wind Energy. 2022 3;25(3):468-504.

[11] Larsen TJ, Hansen AM. How 2 HAWC2, the user's manual. Roskilde: Technical University of Denmark; 2021.

[12] Madsen HA, Larsen TJ, Pirrung GR, Li A, Zahle F. Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact. Wind Energy Science. 2020 1;5(1):1-27.

[13] Li A, Gaunaa M, Pirrung GR, Meyer Forsting A, Horcas SG. How should the lift and drag forces be calculated from 2-D airfoil data for dihedral or coned wind turbine blades? Wind Energy Science. 2022 7;7(4):1341-65.

[14] Meng F, Lio WH, Barlas T. DTUWEC: an open-source DTU Wind Energy Controller with advanced industrial features. Journal of Physics: Conference Series. 2020 9;1618(2):022009.

[15] Larsen TJ, Hanson TD. A method to avoid negative damped low frequent tower vibrations for a floating, pitch controlled wind turbine. Journal of Physics: Conference Series. 2007 7;75:012073.

[16] Gözcü O, Verelst DR. The effects of blade structural model fidelity on wind turbine load analysis and computation time. Wind Energy Science. 2020 4;5(2):503-17.

[17] Dimitrov N, Kelly MC, Vignaroli A, Berg J. From wind to loads: wind turbine site-specific load estimation with surrogate models trained on high-fidelity load databases. Wind Energy Science. 2018 10;3(2):767-90.

[18] Baudin M, Dutfoy A, Iooss B, Popelin AL. In: OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation. Springer International Publishing; 2015. p. 1–38.

[19] Ökten G, Shah M, Goncharov Y. Random and Deterministic Digit Permutations of the Halton Sequence; 2012. p. 609-22.

[20] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

[21] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. Available from: `https://www.tensorflow.org/`.

[22] Murcia JP, Réthoré PE, Dimitrov N, Natarajan A, Sørensen JD, Graf P, et al. Uncertainty propagation through an aeroelastic wind turbine model using polynomial surrogates. Renewable Energy. 2018 4;119:910-22.

[23] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2019. .

[24] Saves P, Lafage R, Bartoli N, Diouane Y, Bussemaker J, Lefebvre T, et al. SMT 2.0: A Surrogate Modeling Toolbox with a focus on hierarchical and mixed variables Gaussian processes. Advances in Engineering Software. 2024 2;188:103571.

[25] Jiménez Á, Crespo A, Migoya E. Application of a LES technique to characterize the wake deflection of a wind turbine in yaw. Wind Energy. 2009 12;13(6):559-72.

[26] Zong H, Porté-Agel F. A momentum-conserving wake superposition method for wind farm power prediction. Journal of Fluid Mechanics. 2020 4;889:A8.

[27] Feng J, Pedersen MM, Riva R, Bredmose H, Santos P. Design optimization of floating offshore wind farms using a steady state motion and flow model. Journal of Physics: Conference Series. 2024.

[28] Nanos EM, Bottasso CL, Tamaro S, Manolas DI, Riziotis VA. Vertical wake deflection for floating wind turbines by differential ballast control. Wind Energy Science. 2022 8;7(4):1641-60.