Deep Reinforcement Learning for Active Wake Control

Neustroev, G.; Andringa, Sytze P.E.; Verzijlbergh, Remco A.; de Weerdt, Mathijs M.

Publication date
2022

Document Version
Final published version

Published in
International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022

Deep Reinforcement Learning for Active Wake Control

Grigory Neustroev
Delft University of Technology
Delft, the Netherlands
g.neustroev@tudelft.nl

Sytze P. E. Andringa
Delft University of Technology
Delft, the Netherlands
s.p.e.andringa@tudelft.nl

Remco A. Verzijlbergh
Delft University of Technology & Whiffle
Delft, the Netherlands
r.a.verzijlbergh@tudelft.nl

Mathijs M. de Weerdt
Delft University of Technology
Delft, the Netherlands
m.m.deweerdt@tudelft.nl

ABSTRACT

Wind farms suffer from so-called wake effects: when turbines are located in the wind shadows of other turbines, their power output is substantially reduced. These losses can be partially mitigated via actively changing the yaw from the individually optimal direction. Most existing wake control techniques have two major limitations: they use simplified wake models to optimize the control strategy, and they assume that the atmospheric conditions remain stable. In this paper, we address these limitations by applying reinforcement learning (RL). RL forgoes the wake model entirely and learns an optimal control strategy based on the observed atmospheric conditions and a reward signal, in this case the power output of the farm. It also accounts for random transitions in the observations, such as turbulent fluctuations in the wind. To evaluate RL for active wake control, we provide a simulator based on the state-of-the-art FLORIS model in the OpenAI gym format. Next, we propose three different state-action representations of the active wake control problem and investigate their effect on the performance of RL-based wake control. Finally, we compare RL to a state-of-the-art wake control strategy based on FLORIS and show that RL is less sensitive to changes in unobservable data.

CCS CONCEPTS

• **Applied computing** → **Environmental sciences**; • **Computing methodologies** → **Sequential decision making**.

KEYWORDS

Deep Reinforcement Learning; Wind Energy; Active Wake Control

ACM Reference Format:
Grigory Neustroev, Sytze P. E. Andringa, Remco A. Verzijlbergh, and Mathijs M. de Weerdt. 2022. Deep Reinforcement Learning for Active Wake Control. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 10 pages.

1 INTRODUCTION

Deep reinforcement learning (RL) has been very successful in playing games, ranging from video games [22] to board games such as chess and Go [49]. While game-playing agents remain an interesting scientific challenge, we should remember to consider problems driven by real-life applications. These problems, however, bring both the challenges of the state space size and observability to a new level: whereas in games the state space can already be astronomical [54], in real-world control problems we must completely forget about using perfect state representation based on physical models. Historically, in the field of control, great progress has been made with models based on domain knowledge. These models provide a proper balance between expressiveness and computational efficiency. Open questions in research on RL are whether it can compete with the state of the art in such domains, under which conditions this is possible, and what remaining challenges for RL need to be addressed before it can surpass non-learning models.

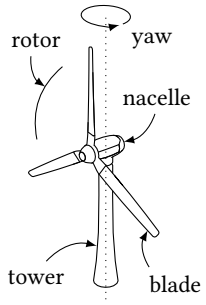

Figure 1: Turbine nomenclature.

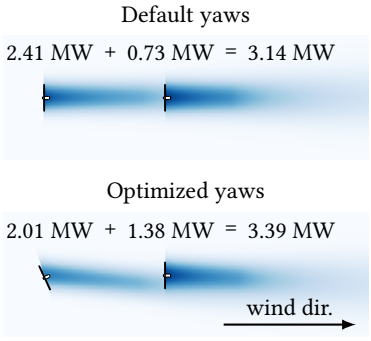

Figure 2: Overhead view of two wind turbines with yaw-based wake control. Darker areas have slower wind.

An example of a real control problem with an enormous state space is that of active wake control in a wind farm. When a wind turbine extracts energy from the wind, it creates a wake area behind its rotor [56]. The wind in this area has reduced velocity and increased turbulence. If another turbine is positioned in the wake, these factors have a negative impact on its power output. In large wind farms, these wake-induced losses can be substantial. For example, a study of an off-shore wind farm in Denmark shows a 12% energy loss due to wake effects [5], another in Canada reports 7%–13% [24]. As the number of wind farms around the world and their average size continue to grow [26], so do their wake-induced losses. Consequently, interest in active wake control is growing.

Early studies of wake effects mitigation focused on per-turbine control of either pitch [33, 47, 52] or generator torque [29]. Later, joint farm-level control of turbines has been demonstrated to be

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

Table 1: Comparison of the existing studies of reinforcement learning for active wake control

		Verstraeten et al. [59]	Stanfel et al. [51]	Dong et al. [12]	this paper
simulator	software	FLORIS [42]	FLORIS	SOWFA [14]	FLORIS
	OpenAI *Gym* API	yes	no	no	yes
	included elements	wind, turbines	wind, turbines	wind, turbines	wind, turbines, met masts
state	space	continuous	discrete	continuous	continuous
	per-turbine observ.	yaw, power	yaw	yaw, wind speed	yaw, lidar measurements
	per-met mast observ.	—	—	—	atm. measurements
	farm-wide observations	—	—	—	atm. measurements
	atm. measurements	wind speed and dir.	wind speed and dir.	—	multiple, see Table 2
	partial observability	no	yes	yes	yes
	noisy observations	no	yes	no	yes
action	space	$\{-1, 0, 1\}^n$, discrete	$\{-1, 1\}^n$, discrete	$[-1, 1]^n$, continuous	$[-1, 1]^n$, continuous
	representation	yaw-based	yaw-based	yaw-based	yaw-based + 2 more
reward	based on	power deficit	power increase	time-averaged power	total power
transition	time-varying data	—	TI, wind speed	internal simul. data	multiple, see Table 2
	stochastic model	—	Gaussian noise	—	multiple
learning	algorithm	GPRL [45]	Q-learning [61]	DDPG [31]	TD3 [16], SAC [20]
	is deep?	no	no	yes	yes

an efficient wake control strategy [18, 24]. This is done via active control of the turbine yaws (i.e., horizontal rotations, see Figure 1). When a turbine is yawed relative to the incoming wind, it has lower power output but the wake center shifts [60]. This wake deflection can be used to improve the power output of down-wind turbines, increasing the total power production, as shown in Figure 2.

A popular approach to active wake control is to use a simulator to estimate wake effects for different combinations of turbine yaws. These simulators use simplified mathematical models of wakes [10, 27, 28], including e.g. multi-zone [18] models. Optimization is then done numerically, for example, using gradient ascent.

Some numerical optimization approaches are so simplified that results are far from optimal, but more detailed models suffer from being computationally intensive, and therefore not feasible for real-time use. Machine learning addresses this issue by optimizing the wind farm control in a model-free way. Machine learning techniques used for wake optimization include game-theoretic control [18, 34], multi-agent Thompson sampling [57, 58], and—most importantly—reinforcement learning (RL) [12, 51, 59]. RL is especially suited for active wake control, as it accounts for the dynamic nature of the problem by including stochastic transitions between states. Nevertheless, studies of RL for active wake control remain limited.

In this paper, we investigate the benefits of RL for active wake control in dynamic atmospheric conditions. To do so, we implement a dynamic wind farm simulator. Our simulator uses FLORIS for each stable state, and supports arbitrary transition models between such states, defined by the user. Its design is driven by real-life wind farm operation: it supports varying angular velocities of the turbines and imitates installations not seen in other studies, such as meteorological masts and nacelle-mounted lidar systems. A detailed list of differences with the existing work on RL for active wake control is given in Table 1. We provide this simulator in the OpenAI *Gym* format [8], a standard for representing RL problems, to facilitate future research in active wake control from the RL community.

Additionally, we discuss two alternative representations of the control actions in this problem, not used in other studies. We compare the performances of state-of-the-art RL algorithms for each representation. Our experiments show that action encoding has a significant impact on the performance of RL methods.

Finally, we demonstrate the benefits of RL compared to model-based optimization. Having no explicit model, it is more robust to changes in unobserved data and to observation noise, which is especially important for real-life applications.

2 PRELIMINARIES

We begin with providing background information both on state-of-the-art model-based active wake control—primarily for the RL community—and on the principles of RL itself—aimed at wind energy researchers interested in this topic.

2.1 FLORIS Simulator and Optimizer

FLORIS (*Flow Redirection and Induction in Steady-State*) [42] is a wake modeling framework which includes many of the state-of-the-art steady-state models, and tools for analysis and optimization of wind farm layout and operation. It is open source, computationally cheap, and implemented in Python; for details please see the work by Annoni et al. [3].

Various studies highlight the accuracy of FLORIS simulations. For example, Gebraad et al. [18] apply a FLORIS-based control strategy in a high-fidelity computational fluid dynamics simulator. Schreiber et al. [48] perform wind tunnel tests. Fleming et al. [13] present a field trial on a commercial offshore wind farm. These and other applications allow us to consider FLORIS as state-of-the-art in both wake modeling and model-based active wake control.

Table 2: Atmospheric conditions in FLORIS

measurement	default val.	description
wind speed	8 m/s	
dir.	270 °	from north clockwise
shear	0.12 s^{-1}	change of speed with height
veer	0 °/m	change of direction with height
turb. intens.	0.06	coeff. of variation of wind speed
air density	1.225 kg/m^3	at 101325 Pa and 15 °C

FLORIS offers various analytical models to compute the wakes, but it does not explicitly model rapidly changing conditions due to turbulence and other small-scale atmospheric phenomena. Higher fidelity tools based on computational fluid dynamics such as large eddy simulation (LES) can be used for this. Examples of LES include *Simulator for On/Off-Shore Wind Farm Applications* (SOWFA) [14], *Dutch Atmospheric Large-Eddy Simulation* (DALES) [23], and *GPU-Resident Atmospheric Simulation Platform* (GRASP) [19].

Unfortunately, LES require substantial computational power. To apply their learning method, Dong et al. [12] performed 90 simulations, each of which took approximately 44 hours on 256 CPU cores. Each simulation consisted of just 1000 seconds of simulated time. This computational power is far beyond what is practical for RL, and beyond the reach of an average researcher. Moreover, not all of the LES models are open source, further limiting their applicability. As a result, steady-state computationally efficient simulators like FLORIS are more commonly used.

It is important to distinguish FLORIS the simulator and FLORIS the controller. A simulation in FLORIS is based on turbine specifications, such as the amount of power they produce at different wind speeds, turbine locations in the wind farm, and a set of atmospheric conditions presented in Table 2. Figure 2 shows a simulation in FLORIS with the default parameters and two turbines positioned at a distance of six rotor diameters ($6 \cdot D$).

These atmospheric conditions are used together with one of the wake models to predict steady-state wake locations and the wind flow throughout the farm. Based on this information, the total power output of the farm is represented as a function of the yaws [46]. This function can then be maximized by an optimizer. FLORIS includes such an optimizer.

FLORIS considers atmospheric conditions as steady, therefore they are represented by a vector of numbers, like the second column of Table 2. Since atmospheric conditions change over time, one of the possible ways to use FLORIS for control in a dynamic system is to use long-time averages. Another approach is to reinitialize it each time new conditions are observed, using either historical data or a simulated multivariate stochastic process.

2.2 Reinforcement Learning

Reinforcement learning is a machine learning technique for mapping observations to actions so as to maximize a numerical reward signal. It focuses on learning from interaction with an environment, like a simulator. RL agents have explicit goals, can sense aspects of their environment and influence it via their actions. RL agents are not told what actions to take but must discover actions that yield the most reward through trial and error [53]. A model-free RL system learns an optimal policy by estimating an action value function which predicts how good each action is in any given state.

Reinforcement learning uses the formal framework of Markov decision processes. In a Markov decision process, the agent and environment interact at discrete time steps. At each time step, the agent receives an observation of the environment's state and selects an action. One step later, the agent receives a reward and observes a new state [53]. These interactions continue, and the agent learns by evaluating which action led to which state and reward.

Several of the more recent successes of RL, such as Rainbow [22] and AlphaGo [49], are based on Deep Reinforcement Learning (DRL). DRL integrates deep learning into RL by representing the policy or other learned functions by a neural network. DRL is most effective in problems with a high-dimensional state-space [15], for example, learning from visual perceptual inputs made up of thousands of pixels [37]. State-of-the-art deep RL methods include among others *Deep Deterministic Policy Gradient* (DDPG) [31], *Twin Delayed Deep Deterministic Policy Gradient* (TD3) [16], and *Soft Actor-Critic* (SAC) [20]. These algorithms are included in the major RL libraries [1, 25, 38, 44] and therefore are easier to use for researchers with less knowledge about deep RL.

All of these are so-called actor-critic methods. They use deep neural networks to concurrently learn a policy that prescribes actions to take in each state (the actor), and state-action values that tell how good the actions chosen by the policy are (the critic).

DDPG updates both the actor and the critic using gradient descent. Its successor TD3 (Twin Delayed DDPG) adds a few tricks to stabilize the learning process. Namely, it uses two critics (hence twin learning) and updates the policy less frequently than DDPG (delayed). Additionally it slightly perturbs the actions to avoid a phenomenon known as catastrophic forgetting which may happen in deep neural networks when they stop receiving novel inputs.

SAC uses similar tricks, but has a non-deterministic policy with entropy regularization. The entropy coefficient controls how much exploration the policy does, and is usually automatically tuned, making SAC more adaptive. Since its conception, this algorithm has been one of the best performing deep RL methods.

2.2.1 RL for active wake control. Table 1 provides an overview of RL methods applied to active wake control problems.

The works of Verstraeten et al. [59] and Stanfel et al. [51] both use discrete actions with −1 and 1 standing for clockwise and counterclockwise rotations at a fixed angular velocity. Instead of directly using the power output of the wind farm as the reward signal, both use some form of reward shaping to construct a different reward signal. Both of these methods use non-deep RL. Both methods use steady-state simulations, with learning done separately per wind speed and direction. The optimal action is chosen based on current yaws. Therefore, in both cases transitions between different atmospheric conditions are not modelled, but transitions between yaws are taken into account.

Instead of neural networks, Verstraeten et al. [59] use Gaussian Processes RL (GPRL) for state-action value function approximation. This is paired with knowledge transfer between similarly positioned turbines to learn the optimal control strategy. This is the only article that uses multi-agent RL, showing its high efficiency. Stanfel et al.

[51] use simple Q-learning, but combine it with domain knowledge. For example, they apply Gaussian blur to the state-action value function, so that similar states do not have vastly different values.

Research on *deep* RL for active wake control in *dynamically changing* environments remains limited. To the extent of our knowledge, the only such application of deep RL was by Dong et al. [12]. They use an offline version of DDPG to learn from examples generated in a high-fidelity (LES) simulator, and then use the simulator to evaluate the resulting policy. Even though the wind speed is steady, LES accounts for fluctuations in the atmosphere caused by turbulence, creating stochastic transitions. As mentioned above, this required substantial computational power, but the results are sufficiently promising to further explore the use of deep RL for wake control. To do so, in this paper we analyze alternative action representations, two different DRL algorithms, and the performance with respect to changes in unobserved data and to observation noise.

3 ACTIVE WAKE CONTROL AS A REINFORCEMENT LEARNING PROBLEM

To be able to apply RL algorithms to the active wake control problem, we need to define it in terms of time steps, states, actions, rewards and one-step transitions. While this has been done in previous studies, the resulting formulations are usually highly abstract and do not reflect the realities of wind farm operation. For example, atmospheric measurements are captured directly at the turbine locations, or are assumed to be uniform across the wind farm. In practice, various measurement tools positioned throughout the farm can be used to provide atmospheric information, such as free-standing meteorological masts or lidar systems. We aim for a more realistic problem formulation that reflects this.

As mentioned earlier, we treat each time step as having a steady-state atmospheric conditions. At the end of a time step, the atmospheric conditions change and the control chosen by the agent is executed, causing a transition to a new state, which is again assumed to be steady. This process is repeated for a predefined number of steps $T_{\max}$. In our definition of the problem, we allow arbitrary chosen (but equal) time intervals Δt between observations and control events, typically a few seconds.

3.1 State Space

In RL, states describe the current environment as observed by the agent and contain all the information used by the agent to choose an action. At any single point of time, the wind farm can be assumed steady and thus can be represented by a FLORIS simulation. Nevertheless, not all of the simulation data is observable by the agent. It is thus important to consider what kind of information is available to the wind farm controller in practice and include only this information in the state description.

First, we assume that the current yaws γ_i of all of the turbines are known, otherwise controlling them may prove difficult. Additionally, a FLORIS simulation allows to measure atmospheric conditions presented in Table 2, and the control strategy may depend on these.

In the current implementation of FLORIS, wind speed, direction, and turbulence intensity (TI) vary across the wind farm and therefore should be measured at specific points in space. In a real-world wind farm such measurements come from meteorological masts or from sensors on the turbines. For example, these can be nacelle-mounted lidar (light detection and ranging) systems. They are installed behind the turbine rotor and can measure the wind in front of it at a distance of 10–300 meters [7, 50].

In contrast, wind shear, veer, and atmospheric density remain constant across the wind farm and can be defined for the wind farm as a whole. In real-life systems, this type of measurement exists as well. For example, some data may come from an external source, such as a meteorological forecast.

When creating a simulation, the user can specify: (a) the positions of meteorological masts and the measurements collected there; (b) which of the turbines are equipped with lidars and what is measured by them; (c) a list of per-farm measurements coming from an external data source. At runtime, the simulator registers the data according to this specification, arranges them into a numeric state vector $s \in \mathbb{R}^k$ and returns this vector to the user. For example, if a simulation includes three turbines that register their yaws γ_i, $i \in \{0, 1, 2\}$ and two masts that register the wind speed M and direction ϕ at their locations, the state is $s = [\gamma_0, \gamma_1, \gamma_2, M_0, \phi_0, M_1, \phi_1]^\top \in \mathbb{R}^7$.

In RL, it is common to normalize states. We define ranges of possible values for each measurement and include an option to rescale each observation to an interval between 0 and 1.

Finally, to account for imperfections in the measuring equipment (including yaw measurements), we allow state vector perturbations by a zero-mean Gaussian noise. This noise is independently drawn at each time step with a scale parameter defined by the user for each of the observed variables from a given list. The normalized observations are then clamped between 0 and 1. If the observations are not normalized, the noise is rescaled accordingly for each observed measurement.

3.2 Action Space

Each action $a = [a_0, a_1, \ldots, a_{n-1}]^\top \in [-1, 1]^n$ is a vector of length n, where n is the number of turbines. Each coordinate a_i encodes a yaw change of the i-th turbine. In FLORIS, when a turbine is rotated counterclockwise relative to the incoming wind, its yaw is positive, otherwise it is negative. We use the same convention.

The way that the yaw of the i-th turbine changes based on the coordinate a_i can be different. We consider three possible interpretations of actions, visualized in Figure 3.

3.2.1 Yaw-based action representation. The action tells how much the turbine yaw should change with respect to the current position. Zero action means that the turbine should remain still, and ±1 correspond to maximum possible rotations, that is $\pm\omega_{\max}$ degrees from the current position, where $\omega_{\max}$ is the maximum angular velocity of the turbine in degrees per time step. This is the representation used in the previous research on RL for active wake control. In this representation, if the current yaw angle of the i-th turbine is γ_i, the new yaw γ_i' will be $\gamma_i' = \gamma_i + a_i \cdot \omega_{\max}$.

3.2.2 Absolute angle representation. The action tells what the optimal yaw should be relative to some static direction. For example, the most prevalent wind direction can be used. In Figure 3 it is west. In this case, 0.5 corresponds to south, −1 and +1 to east, and −0.5 to north. If this desired new yaw is outside of the operational zone of the turbine, it will turn as far towards it as it can, either clockwise

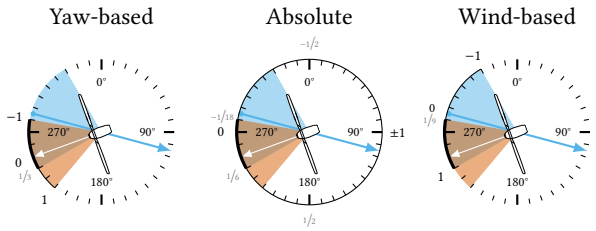

Figure 3: Action representations. The blue arrow shows the wind direction (coming from 285°). The white arrow indicates the turbine orientation (250°). The blue sector (top) shows the desired yaw range (±45° from the wind), and the orange sector (bottom) shows the reachable yaws for an angular velocity of 30°/step. The overlap (brown) shows the reachable yaws, which are the same in all cases.

or counterclockwise, depending on which direction is closer. For example, if the static direction α is 270° as in Figure 3, the next step yaw will be $\gamma_i' = \alpha - a_i \cdot 180°$.

3.2.3 Wind-based action representation. The action is represented as the optimal yaw relative to the current wind direction ϕ measured at the turbine's location. The actions of ±1 correspond to the maximum (desired) yaw relative to the wind. The new yaw is computed as $\gamma_i' = \phi + \frac{1}{2}(a_i + 1) \cdot (\gamma_{\max} - \gamma_{\min}) + \gamma_{\min}$.

After the new yaw angles γ_i' are calculated, they are adjusted to satisfy two constraints.

First, turbines cannot rotate faster than their maximum angular velocity $\omega_{\max}$. This constraint is based on physical limitations and should always be satisfied. In Figure 3, the possible yaws in the next time step are shown in orange. If the agent selects the new yaw γ_i' to be outside of the interval $[\gamma_i - \omega_{\max}, \gamma_i + \omega_{\max}]$, it is clipped to fit inside this interval. For example, if in the absolute representation the agent chooses any action a_i smaller than $-1/18$, it will result in the same new turbine orientation of 280°.

Second, the turbine's yaw relative to the wind should not be too large. This is because its power output is proportional to the cosine of the yaw, and drops fast as it turns away from the wind. To ensure a reasonable operational range, we define minimum $\gamma_{\min}$ and and maximum $\gamma_{\max}$ yaws. This constraint is shown in blue in Figure 3. If the turbine is within the desired yaw limits and attempts to leave them, it will stop. The new yaw is clipped to satisfy this constraint. In rare cases the turbine may end up outside of the desired yaw range, for example due to a sudden change in the wind direction. In the notation of Figure 3, this will result in blue and orange sectors not overlapping. In this case, the turbine should attempt to return as fast as possible to the operational yaw range (the blue sector), but may stay outside of it temporarily. No matter which action is chosen by the agent, the turbine will perform the same rotation—the one that minimizes the angle with the wind direction.

After these constraints are applied, the range of the possible next-step yaws is the same between the three representations. For example, in Figure 3, the new yaw can only be between 240° and 280°. The only thing that differs between the three representations is how the new yaws are computed based on the action vector.

3.3 Rewards

At each time step, FLORIS simulator calculates the total power output P of the wind farm in watts, which is the sum $P = \sum_{i=1}^{n} P_i$ of power outputs P_i of the individual turbines,

$$P_i = \tfrac{1}{2}\rho \cdot A_i \cdot M_i^3 \cdot 4ax_i(M_i) \cdot \left(1 - ax_i(M_i)\right)^2 \cdot \eta \cdot \cos^{p_P} \gamma_i.$$

Here ρ is the air density and M_i is the wind speed at the turbine, both of which are atmospheric conditions that may be included into the state vector and γ_i is the yaw of the i-th turbine, which depends on the i-th coordinate of the action vector. For a more detailed description of the remaining parameters and the function $ax_i(M_i)$, called the axial induction factor, we refer the reader to the paper by Gebraad et al. [18]. This equation shows that the reward is dependent both on the state and the action in a non-linear manner.

3.4 Transitions

When the environment transitions to a new steady state, two things change in the FLORIS simulator. First, the yaws are adjusted according to the action chosen by the agent.

Next, the atmospheric conditions change, resulting in changes in both the wind flow in the simulation, and in the atmospheric measurements registered at the next time step. The most obvious approach is to find a dataset of atmospheric conditions at the desired granularity and use it to generate transitions.

To create a simple yet realistic wind simulation, we looked at a publicly available dataset from *Hollandse Kust Noord (site B)* (HKNB) wind farm zone in the Netherlands [39]. This dataset was chosen because it includes all atmospheric parameters used by FLORIS. Furthermore, it is a practically relevant case, as active wake control will be investigated for the wind farm at this location [11].

The data is measured at ten-minute intervals, which is typical for such datasets. Unfortunately, this means that it cannot be used directly in turbine control experiments, as control typically happens more frequently. To address this issue, we fit a continuous-time stochastic process to the data. This allows us to use one time step for estimation and a different one for simulation.

We use a multivariate Ornstein–Uhlenbeck (MVOU) process [55]. It is a mean-reverting process, meaning that its parameters tend to return to long-term average values, for example, single prevalent direction or mean wind speed. Moreover, many commonly used stochastic processes can be seen as particular cases of MVOU process [36]. For these reasons, Ornstein–Uhlenbeck processes are used in wind modeling [4, 43]. Additionally, by increasing the mean-reversion coefficient of wind direction, we can force it to stay stable, emulating a popular experimental setup with a wind tunnel.

Formally, MVOU process is defined by the following stochastic differential equation $d\mathbf{x} = \Theta(\mathbf{m} - \mathbf{x})\, dt + \mathbf{S}\, d\mathbf{W}_t$. In simpler terms, this process can be described as follows. $\mathbf{m}$ is a vector of mean values to which the process tends to revert. Θ is the drift matrix. It determines the speed of reversion to the means $\mathbf{m}$. $\mathbf{W}_t$ is a multivariate Wiener process that adds random noise. $\mathbf{S}$ is the diffusion matrix that determines the noise covariance matrix $\Sigma = \mathbf{S}\mathbf{S}^\top$. A procedure described by Meucci [35] can be used to estimate the parameters of this process. When these parameters are known, a simulation procedure for arbitrary chosen time steps is provided by Vatiwutipong and Phewchean [55].

Table 3: Estimated parameters of the wind process. Empty cells correspond to zeros.

		drift			diffusion		
parameter	mean	$\log \tau$	$\log M$	ϕ_r	$\log \tau$	$\log M$	ϕ_r
log turbulence intensity	-2.2×10^{0}	2.5×10^{-3}	5.5×10^{-4}	-2.3×10^{-6}	1.3×10^{-2}	-2.1×10^{-4}	-4.4×10^{-4}
log wind speed	2.3×10^{0}	-2.1×10^{-5}	4.8×10^{-5}	5.3×10^{-7}		2.2×10^{-3}	2.5×10^{-4}
wind direction		3.1×10^{-3}	-3.6×10^{-3}	-8.3×10^{-7}			1.6×10^{-1}

We then estimated a process for three atmospheric measurements: turbulence intensity τ, wind speed M, and wind direction ϕ. Because turbulence intensity and wind speed cannot be negative, we applied a logarithmic transformation. For the wind direction, we applied a rotation so that the mean m_{ϕ_r} of the rotated process ϕ_r is equal to zero. This transformation means that the wind direction is measured relative to some prevalent direction, which becomes easier to set in the simulation. Figure 4 shows the wind data used in estimation and three simulated paths.

After the data transformation, we fitted a MVOU process for $\mathbf{x} = [\log \tau, \log M, \phi_r]^\top$. The estimation procedure requires data points at equal time intervals. To achieve this, we cropped the HKNB dataset to the first missing entry. The resulting wind parameters are presented in Table 3. For wind shear and veer, we used mean values in the dataset, $0.0094\,\mathrm{s}^{-1}$ and $-0.025\,^\circ/\mathrm{m}$ respectively.

3.5 OpenAI *Gym* Implementation

OpenAI *Gym* [8] is an open-source Python library of benchmark problems for RL. Each problem in *Gym* is represented by an environment which provides a unified API for RL algorithms to communicate with, making it the the field standard for RL problems. For this reason, we implement our simulator as a *Gym* environment to make it easier for other RL researchers to use [40]. This environment supports all of the elements of state and action representation mentioned in this section, as well as an arbitrary transition function. We provide four basic variants of the transition model, but users can define their own, more sophisticated transition models, for example, using time-varying MVOU processes, or entirely different stochastic models of the wind. If the wind process is not specified, the environment uses steady wind from FLORIS.

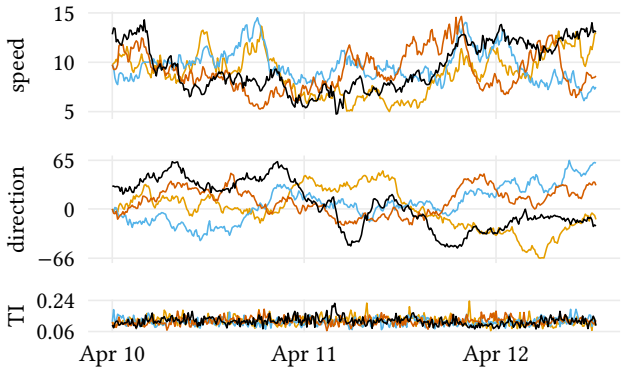

Figure 4: Sample paths of the simulated atmospheric conditions. Black line shows historical data used in estimation.

For the MVOU process, the user can provide a list of n measurement names, whether the logarithmic transformation needs to be taken for each of the measurements, the mean vector of length n, and two $n \times n$ matrices of drift and diffusion. For the wind direction, we additionally use the principal wind direction relative to which it has been measured. After the direction data is generated, it is rotated by that angle. This direction is 270° by default, meaning that the wind comes primarily from the west. This is a common practice in wake control experiments.

4 EXPERIMENTS

Using our *Gym* environment, we performed two experiments where we compare RL to two control strategies. The baseline strategy is to ignore the wake effects, turning the turbines to face the incoming wind. The second strategy is given by the FLORIS optimizer. It optimizes the yaws numerically based on the wind flow model in the simulation. In contrast, RL needs no such model.

4.1 Action Representation Benchmark

In this experiment, we test the effect of action representation on the performance of two state-of-the-art RL algorithms: TD3 and SAC. We omit DDPG even though it is used by [12] because TD3 is its direct successor. The hyperparameters used for each method are available at https://doi.org/10.4121/19107257. We use a setup where one meteorological mast and three turbines are positioned in a line. This single-line layout is commonly used in evaluation of wake control strategies in a wind tunnel, as it represents the worst possible scenario because of the many wake interactions.

We use a MVOU process to simulate the wind as described in Section 3.4, but with a single adjustment: we increase the mean-reversion rate of wind direction by changing the drift coefficient of the wind θ_{ϕ_r,ϕ_r} from -8.3×10^{-7} to 10^{-2}. This forces wind direction to stay within $270^\circ \pm 5^\circ$ but still change with time. Other parameters remained as listed in Table 3. The dependencies of turbulence intensity and wind speed on the wind direction are unchanged.

The state space is a vector of length five, which includes the yaw angles and two measurements provided by the meteorological mast: wind speed and direction. While turbulence intensity changes over time, it is not observed by the wind farm operator. For FLORIS, we used turbulence intensity of 0.12 and wind veer and shear presented in Table 3. We allowed the turbines to turn at the maximum angular velocity of 1°/sec. Further, we used the parameters from the NREL 5 MW reference turbines. These and other FLORIS parameters are taken from the default multi-zone wake model.

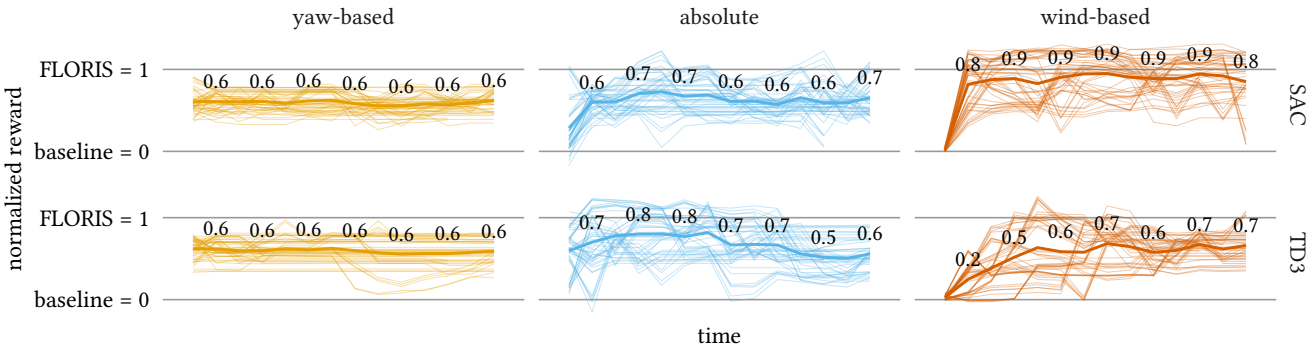

Figure 5: FLORIS-normalized reward of RL agents for different action representations over one month of simulated time for SAC (top) and TD3 (bottom). Thin and thick lines represent individual evaluations and means respectively.

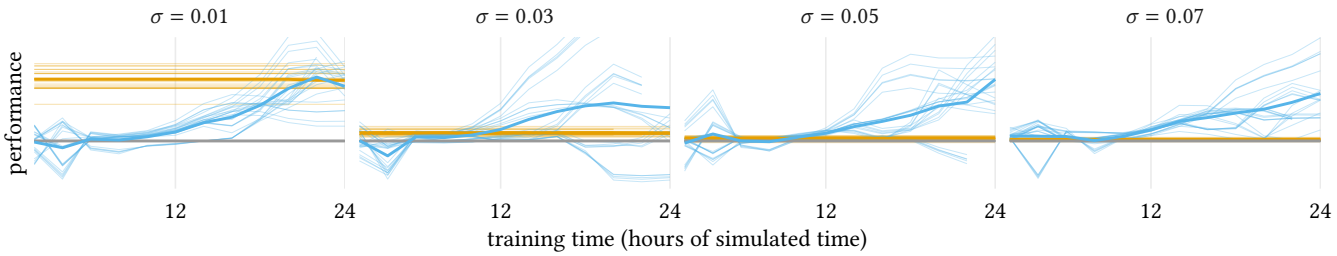

Figure 6: Rewards in the noisy observations benchmark. The gray line (bottom) is the baseline method, the orange horizontal line is FLORIS, and the blue lines show the learning progress of SAC.

For each RL method, we trained 10 agents on different random seeds. To evaluate the performance of the learned policies, we separated training from evaluation as follows. For training, we simulated a week of wind farm operation with time intervals of 10 sec. The evaluation of the momentarily learned policy of each agent is done every twelve hours of simulated time (i.e., 14 times) in five randomly generated environments. Each such evaluation lasts for eight hours of simulated time (2880 steps), during which the total reward is compared against two benchmark strategies: a *baseline* in which each turbine faces the incoming wind, and a model-based control strategy offered by *FLORIS*.

Because different evaluation environments contain different atmospheric conditions, the total power output of these benchmark strategies changes across environments. To compare, we normalize the results so that in each evaluation the total reward of the baseline policy is equal to zero, and of FLORIS to one. In this experiment, FLORIS has access to the exact simulation model sans turbulence intensity, justifying how we use it to indicate a 100% performance.

The rescaled results are presented in Figure 5. While the yaw-based representation may seem to be the most intuitive one, it performs poorly. The reason behind this is that it often fluctuates between the extreme yaws, because either positive or negative actions are chosen too often, leading to a drift in the turbine yaw.

To better understand this effect, consider a situation where the wind is steady. In the other two representations, the optimal action is the same for any current yaw. In the yaw-based representation, however, this is not the case, and if the same action is performed at all time steps, the turbine keeps turning either clockwise or counterclockwise until it reaches the end of the desired yaw sector. Therefore different actions need to be learned for different states. Assuming that learning constant values is easier for a deep neural network, other representations will lead to better performance.

The wind-based representation is the best performing one. To understand why, consider the baseline strategy of always facing the wind. For any down-wind turbine this is the optimal strategy. In the wind-based representation, this strategy is yaw-independent, making it easy to learn. In other representations, the optimal action depends on the incoming wind direction. These results show how the performance of RL methods depends on action representation in the active wake control problem.

Of the two RL agents, SAC performs better than TD3, and learns almost a perfect strategy in the given timeframe. Interestingly, SAC sometimes outperforms FLORIS. This is possible because of the interactions between wind speed, direction and turbulence intensity. While the latter is not observed, its changes can be derived (up to a noise parameter) from other wind data. We speculate that in some of the experiments SAC performed so well because it was able to find a better turbulence representation than the average turbulence intensity known to FLORIS.

4.2 Noisy Observations Benchmark

Of the two benchmarks in the previous experiment, SAC outperformed TD3, but FLORIS offered a better control strategy most of the time. This is because it has a perfect model of the environment, which is not true in practical applications. In this experiment, we compare RL to FLORIS in the presence of imperfect observations.

To illustrate the capabilities of our simulation environment, we slightly adjust the experimental setup of the previous section. First, we remove the mast. Instead, we use per-turbine measurements of wind speed and direction, and a farm-wide measurement of turbulence intensity for both FLORIS-based controller and SAC. Next, we move the second and third turbines by $1/4 \cdot D$ south (down) and north respectively. This makes the problem harder, as it no longer has two symmetric solutions. Finally, in this experiment the time step is 1 second instead of 10 for a more realistic control.

To generate faulty observations, we use four different levels of noise: $\sigma \in \{0.01, 0.03, 0.05, 0.07\}$, that is, after the observations are normalized between 0 and 1, we perturb them with a Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma)$. Only the wind measurements (speed, direction, turbulence intensity) are perturbed, and the yaws are unchanged. We train 5 RL agents for 1 day of simulated time (86400 steps). The evaluations are performed every 2 hours of simulated time (7200 steps) and last for 30 minutes of simulated time (1800 steps). Each evaluation uses five different environments.

The results of this experiment are presented in Figure 6 and Table 4. FLORIS-based optimization struggles to outperform the baseline strategy as the noise scale grows, dropping from 9.5% improvement over the baseline to just 0.2%. While SAC also suffers from the noise in the observations, its performance improvement is between 8.5% and 7.4%, giving a statistically significant improvement over FLORIS-based control in noisy environments.

5 CONCLUSION

Active wake control is a promising real-life application of RL. On the one hand, this problem can be very difficult to solve. Its states are only partially observable, the observations are noisy, and the state-action space can be extremely large for large wind farms. On the other hand, emerging research in this domain indicates that the RL community is well equipped to solve this problem, potentially saving millions of dollars in energy losses due to wake effects.

To facilitate future research in this direction, we have presented a new simulator for this problem. It is based on the state-of-the-art steady-state atmospheric simulator called FLORIS. Our simulator includes many aspects of the problem not seen in the RL research of active wake control before, such as decoupling of measurement devices from turbines and changes in wind conditions. Our simulator is implemented as an OpenAI *Gym* environment, is easy to use off the shelf, and is completely open source.

Table 4: Performance improvement in percent over the baseline in the noisy observations benchmark. For SAC, the final learned strategy is used.

	FLORIS		SAC	
noise, σ	mean	95% conf. int.	mean	95% conf. int.
0.01	**9.54**	9.01 — 10.11	8.46	7.04 — 9.65
0.03	1.24	1.04 — 1.42	**5.15**	0.96 — 10.04
0.05	0.42	0.32 — 0.50	**9.53**	8.07 — 11.02
0.07	0.23	0.18 — 0.29	**7.35**	5.85 — 9.01

While previous RL approaches for this wake control all use the same action encoding, we identified two possible alternatives. We then experimentally showed that the choice of such an encoding has a great impact on the performance of learning methods. Interestingly, the most common one—yaw-based—performed the worst in our experiments. Soft actor-critic, while a golden standard in RL research, has never been applied to active wake control before, and we demonstrated that it shows better performance than TD3.

Finally, we showed that in the presence of imperfect observations, a deep RL agent is capable of learning a better strategy than the state-of-the-art model-based one.

Deep RL for wake control holds great promise for further refinement: first, FLORIS is steady state, which means it optimizes yaws only for the particular time the state was measured. RL methods have techniques to predict the next state and would therefore pick an action that is best suited for the duration until the next course of action can be taken. Second, where FLORIS has a fixed set of parameters, RL techniques can easily be augmented with other potentially relevant data picked up by sensors. Especially deep RL techniques deem to be promising when data gets highly-dimensional. Third, we expect deep RL techniques to outperform model-based optimal control such as FLORIS in terms of computational efficiency, which is especially relevant for big windfarms.

Besides further exploring potential benefits of RL, also some more technical questions remain. Is there an even better action encoding system? Or a different state representation? Are there alternative reward shaping methods? While we investigated some state-of-the-art deep RL methods, sophisticated alternatives exist. RAINBOW [22] combines aspects of many existing RL algorithms. Distributional RL [6] provides an alternative learning paradigm by using distributions instead of deterministic state-action values.

Practical implementation of active wake control methods comes with challenges as well. The wind farm operator needs to maximize power production, but also to minimize structural loads on the turbines. This can be done via safe RL [17] or multi-objective RL [32]. Another problem is scalability; perhaps multi-agent RL [21] can learn to perform active wake control in large-scale wind farms. Finally, RL requires exploration, which will inevitably cost money to the wind farm owner. This can be addressed by using offline RL [2, 30] and learning from the past data, or by using more sample-efficient methods, such as optimistic RL [9, 41]. Finally, the evaluation of the performance of RL vs. model-based wind farm control in more realistic atmospheric environments as present in field tests and atmospheric LES models remains an open topic.

We hope that this work sparks interest of the RL community in this problem, and that our results will make it easier for other researchers to develop new methods for active wake control.

ACKNOWLEDGMENTS

This research received funding from the Netherlands Organization for Scientific Research (NWO).

REFERENCES

[1] Joshua Achiam. 2018. Spinning Up in Deep Reinforcement Learning. https://github.com/openai/spinningup.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An Optimistic Perspective on Offline Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, Hal Daumé and Aarti Singh (Eds.), Vol. 119. PMLR, Vienna, Austria, 104–114.

[3] Jennifer Annoni, Paul Fleming, Andrew Scholbrock, Jason Roadman, Scott Dana, Christiane Adcock, Fernando Porte-Agel, Steffen Raach, Florian Haizmann, and David Schlipf. 2018. Analysis of Control-Oriented Wake Modeling Tools Using Lidar Field Results. *Wind Energy Science* 3, 2 (2018), 819–831.

[4] Jonathan Pablo Arenas-López and Mohamed Badaoui. 2020. The Ornstein–Uhlenbeck Process for Estimating Wind Power under a Memoryless Transformation. *Energy* 213, 118842 (2020), 15.

[5] Rebecca Jane Barthelmie, S. Frandsen, K. Hansen, J. Schepers, K. Rados, W. Schlez, A. Neubert, L. Jensen, and S. Neckelmann. 2009. Modelling the Impact of Wakes on Power Output at Nysted and Horns Rev. In *European Wind Energy Conference*, Vol. 2. WindEurope, Marseille, France, 1351–1373.

[6] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). JMLR.org, Sydney, Australia, 449–458.

[7] Edwin T. G. Bot. 2016. *Flow Analysis with Nacelle-Mounted LiDAR*. Technical Report. Energieonderzoek Centrum Nederland.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. https://gym.openai.com. arXiv:arXiv:1606.01540

[9] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. 2019. Better Exploration with Optimistic Actor-Critic. In *Advances in Neural Information Processing Systems*, Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Edward Fox, and Roman Garnett (Eds.), Vol. 32. Curran Associates, Inc., Vancouver, Canada, 1785–1796.

[10] Antonio Crespo and Julio Hernández. 1996. Turbulence Characteristics in Wind-Turbine Wakes. *Journal of Wind Engineering and Industrial Aerodynamics* 61, 1 (1996), 71–85.

[11] CrossWind. 2021. Innovations. Retrieved October 25, 2021 from https://www.crosswindhkn.nl/innovations.

[12] Hongyang Dong, Jincheng Zhang, and Xiaowei Zhao. 2021. Intelligent Wind Farm Control via Deep Reinforcement Learning and High-Fidelity Simulations. *Applied Energy* 292 (2021), 116928.

[13] Paul Fleming, Jenniger Annoni, Jigar J. Shah, Linpeng Wang, Shreyas Ananthan, Zhijun Zhang, Kyle Hutchings, Peng Wang, Weiguo Chen, and Lin Chen. 2017. Field Test of Wake Steering at an Offshore Wind Farm. *Wind Energy Science* 2, 1 (2017), 229–239.

[14] Paul Fleming, Pieter Gebraad, Jan-Willem van Wingerden, Sang Lee, Matt Churchfield, Andrew Scholbrock, John Michalakes, Kathryn Johnson, and Pat Moriarty. 2013. SOWFA Super-Controller: A High-Fidelity Tool for Evaluating Wind Plant Control Approaches. In *Proceedings of European Wind Energy Association*. European Wind Energy Association, Vienna, Austria, 10.

[15] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. 2018. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning* 11, 3-4 (2018), 219–354.

[16] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholm, Sweden, 1587–1596.

[17] Javier Garcıa and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[18] Pieter Gebraad, F. Teeuwisse, J. W. Wingerden, Paul Fleming, S. Ruben, Jason Marden, and Lucy Pao. 2016. Wind Plant Power Optimization through Yaw Control Using a Parametric Model for Wake Effects—A CFD Simulation Study. *Wind Energy* 19 (2016), 95–114.

[19] Ciaran Gilbert, Jakob Messner, Pierre Pinson, Pierre-Julien Trombe, Remco Verzijlbergh, Pim Dorp, and Harmen Jonker. 2020. Statistical Post-Processing of Turbulence-Resolving Weather Forecasts for Offshore Wind Power Forecasting. *Wind Energy* 23, 4 (2020), 884–897.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholm, Sweden, 1861–1870.

[21] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2019. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (2019), 750–797.

[22] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* 32, 1 (2018), 3215–3222.

[23] Thijs Heus, C. C. van Heerwaarden, H. J. J. Jonker, A. Pier Siebesma, S. Axelsen, K. van den Dries, O. Geoffroy, A. F. Moene, D. Pino, S. R. de Roode, and J. Vilà-Guerau de Arellano. 2010. Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and Overview of Its Applications. *Geoscientific Model Development* 3, 2 (2010), 415–444.

[24] Michael F. Howland, Sanjiva K. Lele, and John O. Dabiri. 2019. Wind Farm Power Optimization through Wake Steering. *Proceedings of the National Academy of Sciences* 116, 29 (2019), 14495–14500.

[25] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. 2021. CleanRL: High-Quality Single-File Implementations of Deep Reinforcement Learning Algorithms. https://github.com/vwxyzjn/cleanrl/.

[26] Mark Z. Jacobson and Mark A. Delucchi. 2009. A Path to Sustainable Energy by 2030. *Scientific American* 301, 5 (2009), 58–65.

[27] Niels Otto Jensen. 1983. *A Note on Wind Generator Interaction.* Technical Report. Risø National Laboratory. 16 pages.

[28] Ángel Jiménez, Antonio Crespo, and Emilio Migoya. 2010. Application of a LES Technique to Characterize the Wake Deflection of a Wind Turbine in Yaw. *Wind Energy* 13, 6 (2010), 559–572.

[29] Kathryn E. Johnson. 2004. *Adaptive Torque Control of Variable Speed Wind Turbines.* Technical Report. National Renewable Energy Laboratory, Golden, Colorado USA. 107 pages.

[30] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *ArXiv* abs/2005.01643 (2020), 43.

[31] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous Control with Deep Reinforcement Learning. In *4th International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.). ICLR, San Juan, Puerto Rico, 10.

[32] Chunming Liu, Xin Xu, and Dewen Hu. 2015. Multiobjective Reinforcement Learning: A Comprehensive Overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 3 (2015), 385–398.

[33] Daria Madjidian and Anders Rantzer. 2011. A Stationary Turbine Interaction Model for Control of Wind Farms. *18th IFAC World Congress Proceedings Volumes* 44, 1 (2011), 4921–4926. 18th IFAC World Congress.

[34] Jason R. Marden, Shalom D. Ruben, and Lucy Y. Pao. 2013. A Model-Free Approach to Wind Farm Control Using Game Theoretic Methods. *IEEE Transactions on Control Systems Technology* 21, 4 (2013), 1207–1214.

[35] Attilio Meucci. 2005. *Risk and Asset Allocation* (first ed.). Springer, New York.

[36] Attilio Meucci. 2009. Review of Statistical Arbitrage, Cointegration, and Multivariate Ornstein–Uhlenbeck. https://ssrn.com/abstract=1404905. , 20 pages.

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518 (02 2015), 529–33.

[38] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A Distributed Framework for Emerging AI Applications. In *13th USENIX Symposium on Operating Systems Design and Implementation*. USENIX association, Carlsbad, California United States, 561–577.

[39] Ministry of Economic Affairs Netherlands Enterprise Agency and Climate Policy. 2019. Hollandse Kust Noord (Site B) Dataset. Retrieved September 30 from https://offshorewind.rvo.nl/file/view/55040229/Processed+data+HKNB.

[40] Grigory Neustroev, Sytze P.E. Andringa, Remco A. Verzijlbergh, and Mathijs M. de Weerdt. 2022. The Wind Farm Gym. https://github.com/AlgTUDelft/wind-farm-env. https://doi.org/10.4121/19107257

[41] Grigory Neustroev and Mathijs Michiel de Weerdt. 2020. Generalized Optimistic Q-Learning with Provable Efficiency. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, and G. Sukthankar (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, Auckland, New Zealand, 913–921.

[42] NREL. 2021. FLORIS. Version 2.4. https://github.com/NREL/floris

[43] Sergey Obukhov, Emad M. Ahmed, Denis Y. Davydov, Talal Alharbi, Ahmed Ibrahim, and Ziad M. Ali. 2021. Modeling Wind Speed Based on Fractional Ornstein–Uhlenbeck Process. *Energies* 14, 17 (2021), 5561.

[44] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8.

[45] Carl Edward Rasmussen and Malte Kuss. 2003. Gaussian Processes in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf (Eds.), Vol. 16. MIT Press, Vancouver and Whistler, British Columbia, Canada, 751–758.

[46] Andreas Rott, Bart Doekemeijer, Janna Kristina Seifert, Jan-Willem van Wingerden, and Martin Kühn. 2018. Robust Active Wake Control in Consideration of Wind Direction Variability and Uncertainty. *Wind Energy Science* 3, 2 (2018), 869–882.

[47] Gerard Schepers and S. P. van der Pijl. 2007. Improved Modelling of Wake Aerodynamics and Assessment of New Farm Control Strategies. *Journal of Physics: Conference Series* 75, 012039 (2007), 8.

[48] Johannes Schreiber, E. M. Nanos, Filippo Campagnolo, and Carlo L. Bottasso. 2017. Verification and Calibration of a Reduced Order Wind Farm Model by Wind Tunnel Experiments. *Journal of Physics: Conference Series* 854, 012041 (2017), 11.

[49] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529 (2016), 484–489.

[50] Matt Smith, Michael Harris, John Medley, and Chris Slinger. 2014. Necessity is the Mother of Invention: Nacelle-Mounted Lidar for Measurement of Turbine Performance. *Energy Procedia* 53 (2014), 13–22. EERA DeepWind' 2014, 11th Deep Sea Offshore Wind R&D Conference.

[51] Paul Stanfel, Kathryn Johnson, Christopher J. Bay, and Jennifer King. 2021. Proof-of-Concept of a Reinforcement Learning Framework for Wind Farm Energy Capture Maximization in Time-Varying Wind. *Journal of Renewable and Sustainable Energy* 13, 4 (2021), 14.

[52] Maarten Steinbuch, W. W. de Boer, O. H. Bosgra, S. A. W. M. Peters, and J. Ploeg. 1988. Optimal Control of Wind Power Plants. *Journal of Wind Engineering and Industrial Aerodynamics* 27, 1 (1988), 237–246.

[53] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). MIT Press, Cambridge, UK.

[54] John Tromp and Gunnar Farnebäck. 2006. Combinatorics of Go. In *5th International Conference on Computers and Games*, David et al Hutchison (Ed.). Springer, Turin, Italy, 84–99.

[55] Pat Vatiwutipong and Nattakorn Phewchean. 2019. Alternative Way to Derive the Distribution of the Multivariate Ornstein–Uhlenbeck Process. *Advances in Difference Equations* 2019, 276 (2019), 7.

[56] L. Nord-Jan Vermeer, Jens Sørensen, and Antonop Crespo. 2003. Wind turbine wake aerodynamics. *Progress in Aerospace Sciences* 39, 6 (2003), 467–510.

[57] Timothy Verstraeten, Eugenio Bargiacchi, Pieter J. K. Libin, Jan Helsen, Diederik M. Roijers, and Ann Nowé. 2020. Multi-Agent Thompson Sampling for Bandit Applications with Sparse Neighbourhood Structures. *Scientific Reports* 10, 6728 (2020), 13.

[58] Timothy Verstraeten, Pieter-Jan Daems, Eugenio Bargiacchi, Diederik M. Roijers, Pieter J.K. Libin, and Helsen Jan. 2021. Scalable Optimization for Wind Farm Control Using Coordination Graphs. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. IFAAMAS, online, 1362–1370.

[59] Timothy Verstraeten, Pieter Libin, and Ann Nowé. 2020. Fleet Control Using Coregionalized Gaussian Process Policy Iteration. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, Giuseppe De Giacomo, Alejandro Catala, Bistra Dilkina, Michela Milano, Senen Barro, Alberto Bugarin, and Jerome Lang (Eds.), Vol. 325. IOS Press, Santiago de Compostela, Spain, 1571–1578.

[60] Jan Willem Wagenaar, L. A. H. Machielse, and J. G. Schepers. 2012. Controlling Wind in ECN's Scaled Wind Farm. In *Proceedings of Europe Premier Wind Energy Event (EWEA 2012)*, Vol. 1. EWEA, Copenhagen, Denmark, 161–168.

[61] Christopher John Cornish Hellaby Watkins. 1989. *Learning from Delayed Rewards*. PhD Thesis. King's College, Cambridge, UK.