

PAPER • OPEN ACCESS

A Reinforcement Learning framework for Wake Steering of Wind Turbines

To cite this article: Kjetil Olsen Lye *et al* 2023 *J. Phys.: Conf. Ser.* **2626** 012051

View the [article online](#) for updates and enhancements.

You may also like

- [Impact Analysis of Track Slip in Sediment on Submarine Tracked Vehicle Steering Performance](#)

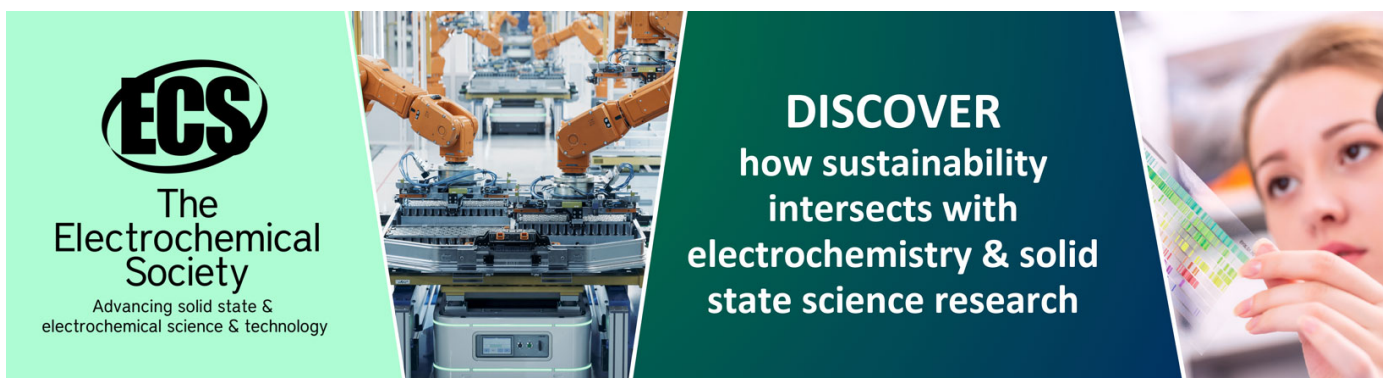
Zhang Hao, Duan Yubing, Shi Xin et al.

- [Predicting the benefit of wake steering on the annual energy production of a wind farm using large eddy simulations and Gaussian process regression](#)

Daan van der Hoek, Bart Doekemeijer, Leif Erik Andersson et al.

- [Field Validation of Wake Steering Control with Wind Direction Variability](#)

Eric Simley, Paul Fleming and Jennifer King



ECS
The
Electrochemical
Society
Advancing solid state &
electrochemical science & technology

DISCOVER
how sustainability
intersects with
electrochemistry & solid
state science research

A Reinforcement Learning framework for Wake Steering of Wind Turbines

Kjetil Olsen Lye^{1,*}, Mandar V Tabib¹, Kjetil André Johannessen¹

¹SINTEF Digital, Trondheim, Norway

*Corresponding author, kjetil.olsen.lye@sintef.no

Abstract. Ideally, optimum power for a single turbine is obtained when the wind-turbine is aligned with the wind direction. However in multi-turbine wind-farm set-up, wake effects lead to decreased power production from downstream turbine [1, 2, 3, 4, 5]. Hence, a control strategy based on wake steering involves misalignment of upstream turbines with the wind direction causing their wakes to deflect away from downstream wind turbines needs to be investigated. A great deal of work has been put into dynamically controlling the orientation of the individual wind turbines to maximize the power output of the farm [6, 7, 8, 9]. In the wake-steering based control, the misaligned wind turbines produce less power, while the performance of downstream turbines gets enhanced which increases overall net power gain for the wind power plant. Traditionally, the benefits of wake steering have been demonstrated assuming fixed wind directions (e.g., using high-fidelity modeling). Amongst the most recent techniques, particularly promising is the use of Reinforcement learning (RL), which is a branch of machine learning where models are trained to make decisions based on observations of their environment. It is a flexible framework for devising strategies for solving optimal control problems in a broad range of applications across the sciences. Early attempts at using Reinforcement learning for wake steering have been carried out [7, 8, 9], and show promising results. In practice, however, wake-steering controllers must operate in dynamic wind environments in which the wind conditions are estimated from imperfect measurements. Hence, a reinforcement learning framework is developed in this work for dynamic wind conditions. The results show that the framework is promising, and we compare the deep reinforcement learning approach against a considerably more expensive traditional optimization approach which serves as a good baseline. Future work could include looking at more realistic wake models, steering in the presence of noisy observations, and incorporating weather predictions.

1. Introduction

In this paper we develop a novel deep-reinforcement learning algorithm for optimal yaw-misalignment of wind turbines to maximize total power output of a wind turbine farm.

Traditionally it has been common to align the wind turbines with the wind direction, as depicted in Figure 1. This will create a *local* optimum for each turbine. However, aligning each turbine with the wind direction will create a significant drop in wind velocity in the downwind direction of the turbine, known as *wake effects* [1, 2, 3, 4, 5]. Due to these wake effects, aligning each turbine with the wind direction may yield sub-optimal power generation for the wind farm as a whole.

Recent works have therefore focused on misaligning the turbines with the wind direction, also known as yaw misalignment, to steer the wakes in such a way that a maximum overall farm power production is reached [10, 7, 8, 9].



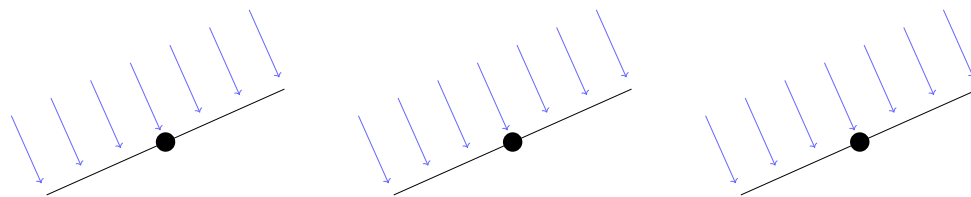


Figure 1: Yaw alignment of wind turbines with the wind direction.

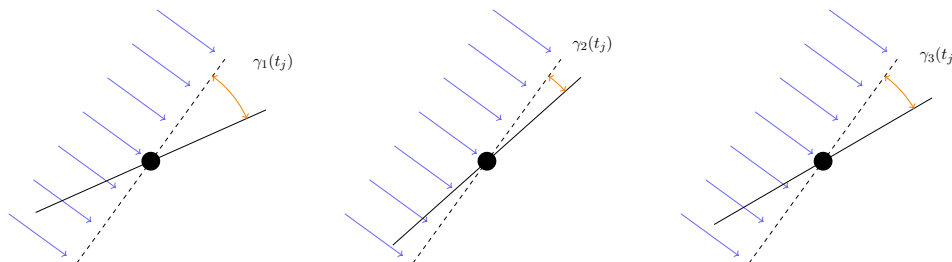


Figure 2: Yaw misalignment angles.

Early attempts at wake steering involved using a simplified, analytic wake model and finding critical points in this wake model to obtain optimal yaw misalignment angles [10]. In theory such an approach can be extended to more complex wake models or even direct numerical simulations, but in practice the run-time of even a single realization of one set of yaw misalignment angles can be prohibitively large in an operational setting.

Deep reinforcement learning (DRL) [11, 12, 13] has shown great promise in multiple fields, most notably in beating a series of Atari games [14]. While it is possible to run DRL on operational systems, due to its slow convergence, it usually performs best when a ample amount of simulation data is available. Due to this reason, one usually needs to train on very simplified simulation data in order to get reasonable training times. Preliminary studies of DRL for optimal wake steering of wind turbines have also shown great promise [7, 8, 9]. However, little is so far known of the optimality of the DRL results compared to a best case scenario optimization algorithm.

In this paper, we compare DRL against a traditional optimization algorithm knowing the full model. We introduce a new, randomized wind field through an Karuhnen-Loeve expansion to maximize coverage in the training phase. To achieve tractable run-times, we run a simplified engineering wake model and power output calculation. We rigorously test the proposed algorithm against two sound baseline models through alignment and traditional optimization.

2. Problem formulation

We use the PyWake python library [15] version 2.4.0 for understanding the wake from turbines and its influence on power production. We use a Gaussian based wind deficit model [16], specifically the `BastankhahGaussian` class in PyWake. The deflection model is based on large eddy simulations [17], and uses the `JimenezWakeDeflection` class in PyWake.

Concretely, we consider a wind farm with N turbines. For simplicity, we consider only two-dimensional placement in this paper and neglect any variation in the vertical direction. We assume background wind field $u : [0, \infty) \rightarrow \mathbb{R}^2$ varying in time but uniform in space, in both magnitude and direction. We let $\gamma_i(t)$ denote the yaw displacement angle for wind turbine i against the background wind field $u(t)$ at time t , see Figure 2. We collect the yaw displacement

angles into a vector $\vec{\gamma}(t) := (\gamma_1(t), \dots, \gamma_N(t))$. For each turbine $i = 1, \dots, N$, we let $P_i(t, \vec{\gamma}(t))$ be the power generated by wind-turbine i at time t given the background wind field $u(t)$, and the collective yaw displacement vector $\vec{\gamma}_i(t)$. We furthermore let $P(t, \vec{\gamma})$ denote the total power output of the wind farm, that is

$$P(t, \vec{\gamma}) := \sum_i P_i(t, \vec{\gamma}) \quad (1)$$

In this paper, we try to find optimal yaw displacement angles $\gamma_1, \dots, \gamma_N$ as a function of time for a given time-varying background wind field. To make the problem numerically tractable, we choose $M \in \mathbb{N}$ control times t_1, \dots, t_M for which we optimize the power output. We estimate the power of the wind turbine to be according to a steady-state power production estimate proportional to the local wind speed cubed multiplied by the cosine of the yaw misalignment angle. In the pure optimization setting, we seek to solve

$$\vec{\gamma}(t_j) = \arg \max_{\vec{\gamma}} \{P(t_j, \vec{\gamma})\}. \quad (2)$$

3. Deep reinforcement learning

In reinforcement learning, one considers a so-called Markov decision process (MDP) consisting of state space \mathcal{S} , an action space \mathcal{A} , paired with a transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ giving the conditional probabilities of transitioning from one state to another given an action, and the reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, R_{\max}]$, where $R(s_1, a, s_2)$ denotes the reward for going from state s_1 to s_2 through the action a .

In a reinforcement learning setting we typically seek an optimal policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ which selects an action given a state. $\pi(s, a)$ gives the probability of selecting an action a given an observed state s .

We will let the state space be $\mathcal{S} := [0, 1]^{4N+2}$ where we remind the reader that N is the number of turbines present. A state $s \in \mathcal{S}$ is defined to be a tuple consisting of normalized values of:

- (i) Wind direction (1 value).
- (ii) Wind speed (1 value).
- (iii) Power production per turbine (N values).
- (iv) Placement of turbines ($2N$ values),
- (v) Current yaw of turbines (N values),

We note that the placement of turbines is relevant for transfer learning, where one could envision training the policy for one placement of turbines and using it on another placement.

We define our action space be $\mathcal{A} := [-10, 10]^N$, where an action $(a_1, \dots, a_N) = a \in \mathcal{A}$ denotes the change in angle for each wind turbine. The evaluation of the reward function is given by (1), which again is computed by the simplified wake model described in the introduction.

We use the proximal policy optimization (PPO) algorithm to find the optimal policy [18]. We use the implementation found in the Stable Baseline python library [19].

4. Experiments and Results

4.1. Model setup

We show two experiments comparing three strategies for finding the optimal yaw misalignment angle as a function of time:

- (i) DRL : The DRL framework described in the previous section,
- (ii) PSO : A particle swarm optimization (PSO) procedure using PySwarm [20] to solve (2),
- (iii) Align : Aligning the turbine with the wind direction (corresponds to setting $\vec{\gamma} = 0$ in (1)).

We test with a time varying wind field given as

$$\vec{u}(t) = \begin{pmatrix} r(t) \cos(\theta(t)) \\ r(t) \sin(\theta(t)) \end{pmatrix} \quad (3)$$

where we in the test setup set

$$r(t) = 5 \cos\left(\frac{t}{60 \cdot 60} + 0.2\right) + 7,$$

and

$$\theta(t) = 80 \sin\left(\frac{t}{60 \cdot 60}\right) + 80.$$

In the case of the DRL algorithm, we first we first train the DRL algorithm with 1 000 000 timesteps of length 1 minute. We draw random wind velocities \vec{u} by (3) with r and θ given through a Karhunen–Loève expansion given as

$$r(\omega; t) := 5 \sum_{m=1}^{100} m^{-2} \sin\left(m\pi \frac{t}{60 \times 60}\right) (2X_m(\omega) - 1) + 7 \quad t \in [0, 1], \omega \in \Omega$$

and

$$\theta(\omega; t) := 80 \sum_{m=1}^{100} m^{-2} \sin\left(m\pi \frac{t}{60 \times 60}\right) (2X_m(\omega) - 1) + 80 \quad t \in [0, 1], \omega \in \Omega$$

where X_1^i, \dots, X_{100}^i for $i = 1, 2$ are uniformly independently distributed random variables over the probability space Ω . At an interval of 24 hours (1440 steps) we redraw X_1, \dots, X_{100} . We note that the training data from the Karhunen–Loève data is not meant to match the test setup, and does for instance not carry with it a phase shift. We stress that this is a strength rather than a weakness of the test: the training data is explicitly different from the test data, which is what one would expect in a real-world scenario.

We note that the runtime in both experiments for the PSO optimization uses approximately 100 seconds per iteration on a single CPU core on a AMD Ryzen 9 5950X, while the DRL (once trained) uses roughly 0.02 seconds on the same CPU core. The results from two experiments for which the above-listed three strategies are compared are as below :

4.2. Experiment one: Closely spaced turbines for strong wake effects

In order to stress test our DRL approach, we place five wind turbines on a line with 100 meters distance apart (0.77 turbine diameter), see Figure 3. Wake effects are dominating in this example, and we expect the DRL approach to clearly beat the simple alignment procedure. We measure the accumulated energy production in Figure 4, and note that the DRL approach beat both the alignment procedure but also the particle swarm optimization. We believe that the reason for the PSO being beat by the DRL is two-fold in this case: 1) The PSO could be given more resources (particles) to obtain a better optimum (at the expense of a longer runtime), and more importantly 2) while the PSO optimizes (2), the DRL algorithm also takes expected future gains into account in the sense that it actually optimizes a reward function of the form

$$\sum_j q_j \tilde{P}(t_j, \vec{\gamma}),$$

where q_j are discounting factors (so that one can weigh short-term, certain reward higher than longer term rewards) and \tilde{P} is the approximate (learned) power output of the wind farm. While the first point is possible to remedy for the PSO algorithm, we do note that giving the PSO algorithm more particles would render it useless in an operational setting even with this very simple wake model.

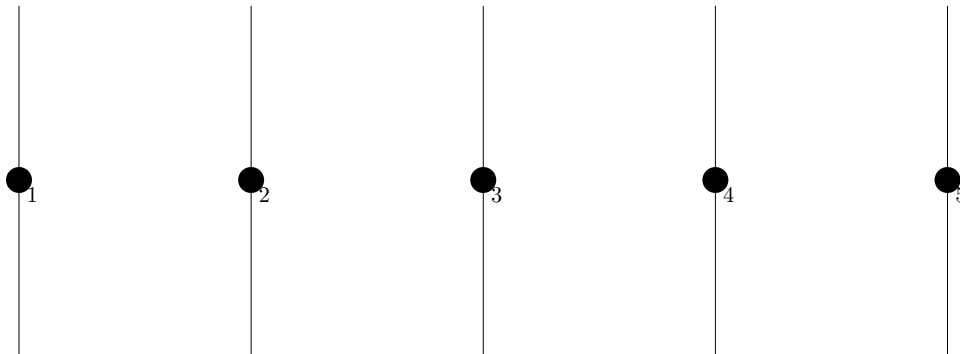


Figure 3: Closely spaced turbines with a pairwise distance of 100 meters.

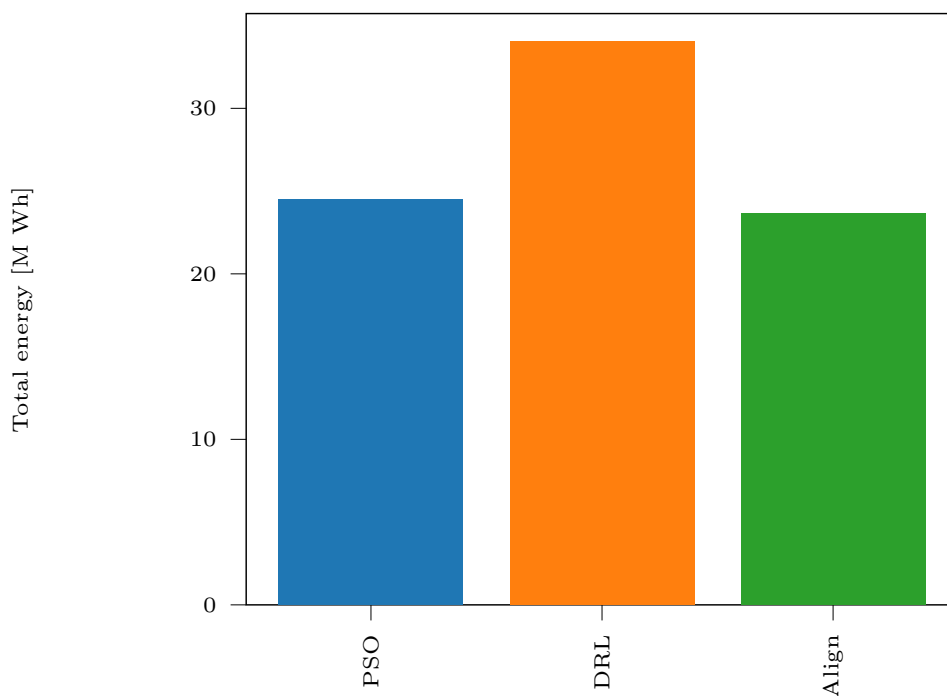


Figure 4: Overall power output for the closely spaced turbines with a pairwise distance of 100 meters.

4.3. Experiment two : Turbines spatially distributed in x and y for milder wake effects

In this experiment we distribute 16 turbines uniformly with an average minimum spacing of 500 meters (5 turbine diameters), see Figure 5 for an illustration. The results are plotted in Figure 6. We expect wake effects to have a lesser impact on this setup, simply due to the spacing of the turbines, but nevertheless verify that the DRL approach is able to match the results of both the PSO algorithm and the alignment procedure. Since the PSO and alignment procedure is so close in performance, we suspect the alignment procedure is already close to optimal for this configuration, and one can not expect any better results from the DRL approach.

5. Conclusion

In this work we developed a Deep reinforcement learning algorithm for optimal yaw angle misalignment for increasing the power production of wind farms. We compared the DRL

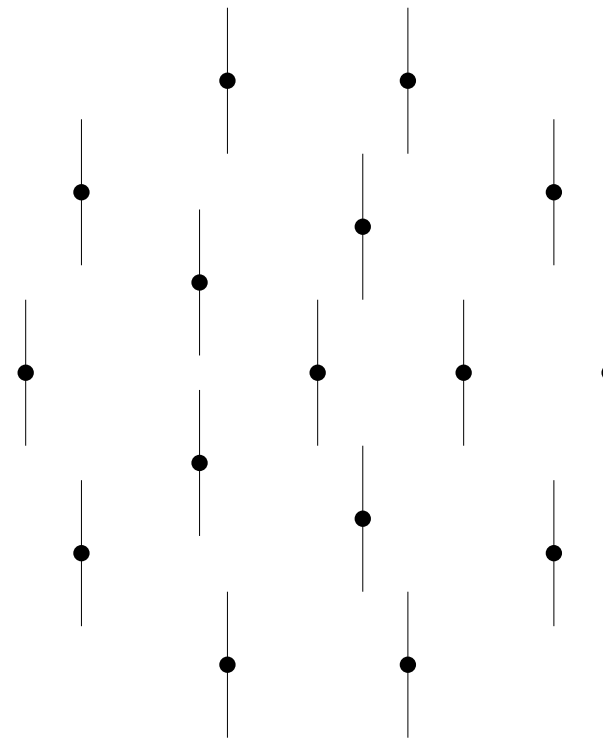


Figure 5: Uniformly spaced turbines with an average pairwise distance of 500 meter.

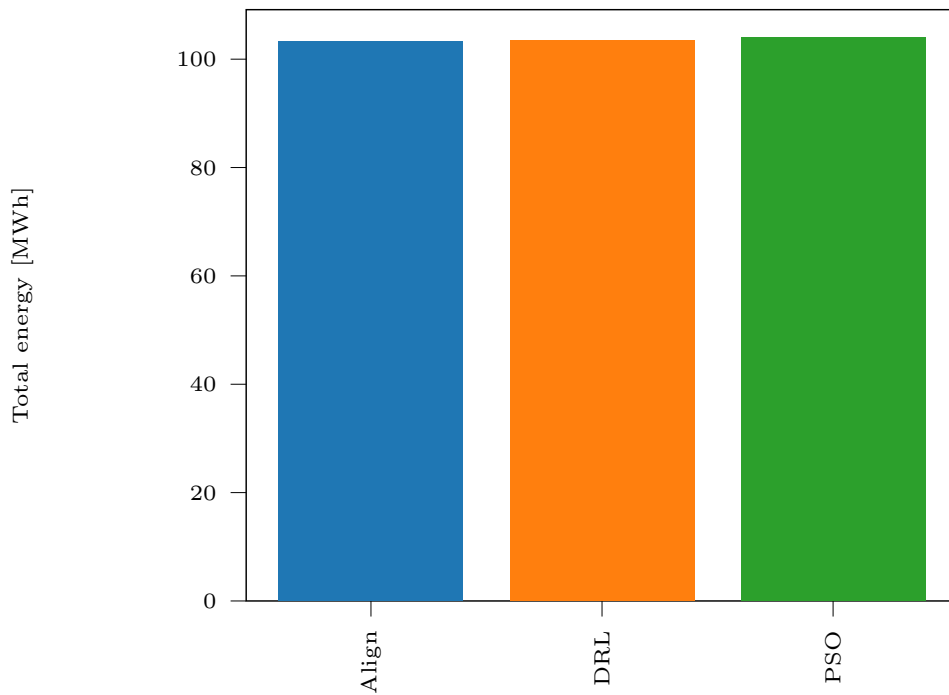


Figure 6: Overall power output for the uniformly spaced turbines with an average pairwise distance of 500 meters.

algorithm against a simple alignment algorithm and a traditional optimization algorithm. We saw that in extreme wake cases, the DRL algorithm easily beats the other approaches, and in less wake-dominated cases the DRL algorithm is able to match the other approaches. We conclude that the DRL shows great promise being able to match and beat state of the art algorithms for the wake steering problem. This DRL methodology can now be further developed by testing it with realistic wind farms and with complex wake models, and examining wake steering in the presence of noisy observations.

6. Acknowledgment

The authors acknowledge the financial support from the SEP funding and the funding from Research Council of Norway for *NORTHWIND:Norwegian Research Centre on Wind Energy* (project no. 321954).

7. References

- [1] Tabib M V, Rasheed A and Kvamsdal T 2015 *Journal of Physics: Conference Series* **625** 012032 URL <https://doi.org/10.1088/1742-6596/625/1/012032>
- [2] Siddiqui M S, Rasheed A, Tabib M and Kvamsdal T 2016 *Journal of Physics: Conference Series* **753** 032059 URL <https://doi.org/10.1088/1742-6596/753/3/032059>
- [3] Tabib M, Rasheed A and Kvamsdal T 2015 *Energy Procedia* **80** 302–311 URL <https://doi.org/10.1016/j.egypro.2015.11.434>
- [4] Fuchs F G, Rasheed A, Tabib M and Fonn E 2016 *Journal of Physics: Conference Series* **753** 082031 URL <https://doi.org/10.1088/1742-6596/753/8/082031>
- [5] Tabib M, Rasheed A and Fuchs F 2016 *Journal of Physics: Conference Series* **753** 032063 URL <https://doi.org/10.1088/1742-6596/753/3/032063>
- [6] Howland M F, Lele S K and Dabiri J O 2019 *Proceedings of the National Academy of Sciences* **116** 14495–14500 URL <https://doi.org/10.1073/pnas.1903680116>
- [7] Stanfel P, Johnson K, Bay C J and King J 2020 A distributed reinforcement learning yaw control approach for wind farm energy capture maximization *2020 American Control Conference (ACC)* (IEEE) URL <https://doi.org/10.23919/acc45564.2020.9147946>
- [8] Stanfel P, Johnson K, Bay C J and King J 2021 *Journal of Renewable and Sustainable Energy* **13** 043305 URL <https://doi.org/10.1063/5.0043091>
- [9] Dong H, Zhang J and Zhao X 2021 *Applied Energy* **292** 116928 URL <https://doi.org/10.1016/j.apenergy.2021.116928>
- [10] Howland M and Dabiri J 2019 *Energies* **12** cited By 14 URL <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85069609908&doi=10.3390%2fen12142717&partnerID=40&md5=bf288457695ffa1caeb1a2b32f5af672>
- [11] Li Y 2018 Deep Reinforcement Learning: An Overview arXiv:1701.07274 [cs] URL <http://arxiv.org/abs/1701.07274>
- [12] Arulkumaran K, Deisenroth M P, Brundage M and Bharath A A 2017 *IEEE Signal Processing Magazine* **34** 26–38 ISSN 1053-5888 arXiv:1708.05866 [cs, stat] URL <http://arxiv.org/abs/1708.05866>
- [13] Francois-Lavet V, Henderson P, Islam R, Bellemare M G and Pineau J 2018 *Foundations and Trends® in Machine Learning* **11** 219–354 ISSN 1935-8237, 1935-8245 arXiv:1811.12560 [cs, stat] URL <http://arxiv.org/abs/1811.12560>
- [14] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D and Riedmiller M 2013 Playing atari with deep reinforcement learning URL <https://arxiv.org/abs/1312.5602>
- [15] Pedersen M M, van der Laan P, Friis-Møller M, Rinker J and Réthoré P E 2019
- [16] Bastankhah M and Porté-Agel F 2014 *Renewable Energy* **70** 116–123 ISSN 0960-1481 special issue on aerodynamics of offshore wind energy systems and wakes URL <https://www.sciencedirect.com/science/article/pii/S0960148114000317>
- [17] Jiménez A, Crespo A and Migoya E 2010 *Wind Energy* **13** 559–572 (Preprint <https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.380>) URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/we.380>
- [18] Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O 2017 Proximal Policy Optimization Algorithms arXiv:1707.06347 [cs] URL <http://arxiv.org/abs/1707.06347>
- [19] Hill A, Raffin A, Ernestus M, Gleave A, Kanervisto A, Traore R, Dhariwal P, Hesse C, Klimov O,

Nichol A, Plappert M, Radford A, Schulman J, Sidor S and Wu Y 2018 Stable baselines <https://github.com/hill-a/stable-baselines>

[20] Miranda L J 2018 *Journal of Open Source Software* **3** 433 URL <https://doi.org/10.21105/joss.00433>