










WindGym: A Reinforcement Learning Environment for Wind Farm Control

Marcus Binder Nilsen ¹, Julian Quick ¹, Teodor Olof Benedict Åstrand ¹, Ernestas Simutis¹, and Pierre-Elouan Mikael Réthoré ¹

¹ Department of Wind and Energy Systems, Technical University of Denmark, Roskilde, Denmark 
Corresponding author

DOI: [10.21105/joss.09946](https://doi.org/10.21105/joss.09946)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Sébastien Boisgérault](#)  

Reviewers:

- [@lorenzoschena](#)
- [@Manishms18](#)

Submitted: 25 November 2025

Published: 28 April 2026

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

WindGym is an open-source Python package for reinforcement-learning (RL) based control of wind farms. It provides both single-agent and multi-agent environments, following the Gymnasium API for centralized controllers and the PettingZoo API for multi-agent settings, enabling drop-in use with mainstream RL frameworks ([Terry et al., 2021](#); [Towers et al., 2024](#)). WindGym is built on top of DYNAMIKS, a multi-fidelity flow simulation framework, which allows users to seamlessly adjust between computational speed and physical fidelity within a single interface ([Pedersen et al., 2026](#)).

The goal of WindGym is to lower the barrier to reproducible research and benchmarking within the field of RL for wind farm control by standardizing interfaces and providing built-in examples, reward utilities, and tests. The package is MIT-licensed and comes with documentation, continuous integration, and ready-to-run training pipelines, making it straightforward for researchers to prototype, compare, and share RL-based wind farm control strategies.

Statement of need

Wind energy is projected to play an increasingly important role in global energy production if the transition towards climate neutrality is to be realized ([IEA, 2021](#); [International Renewable Energy Agency \(IRENA\), 2022](#)). Today, most wind turbines are placed closely together in wind farms to leverage shared infrastructure and reduce land use ([Vondelen, 2024](#)). However, this introduces the wake effect, where an upstream turbine impedes the incoming flow, resulting in decreased wind speed and increased turbulence for downstream turbines. This can lead to decreased power output and increased structural loads ([Howland & Dabiri, 2020](#)). One way to mitigate this phenomenon is wake steering, where turbines are intentionally misaligned with the wind to help steer the wake away from downstream turbines ([Annoni et al., 2018](#)).

Developing control algorithms for wind farms is not a trivial task. One area that has been gaining increased interest is using RL to learn control strategies based on simulated wind farm environments ([Abkar et al., 2023](#); [Göçmen et al., 2025](#)). However, even though interest in this field is increasing, much of the work remains fragmented, with many researchers using custom simulators or failing to publish their code bases. WindGym addresses this gap by providing an RL-first framework that follows the de facto RL APIs, abstracts different wind-farm simulation back-ends within a unified interface, and includes examples and tests to support reproducibility. By lowering the barrier to entry, WindGym enables systematic comparisons across algorithms, reward definitions, and simulator fidelity levels.

State of the Field

When we began developing WindGym, no existing package combined dynamic wind farm simulation with standard RL interfaces. The package *wind-farm-env* (Neustroev et al., 2022) existed as the only existing open source option, but it is built on Floris (NREL, 2025), with no obvious way of implementing transient wake behaviour. Since then, *WFCRL* (Monroc et al., 2025) has emerged, providing RL environments built on Fastfarm (Jonkman et al., 2017) and Floris. We believe that WindGym offers a distinct advantage. Because it is built on DYNAMIKS (Pedersen et al., 2026), a multi-fidelity framework that allows users to interchange fidelity levels within a single codebase, researchers can train agents using fast, low-fidelity simulations and validate them with higher-fidelity models without changing their RL setup. Additionally, WindGym provides both single-agent and multi-agent environments through Gymnasium and PettingZoo APIs, whereas *WFCRL* currently focuses on multi-agent scenarios.

Software Design

WindGym's architecture prioritizes simplicity and modularity. The core design centres on a single main environment file (`WindFarmEnv`) that encapsulates all essential logic for state management, action processing, and reward computation. The multi-agent variant (`MultiAgentWindFarmEnv`) is implemented as a thin wrapper around this core, mapping the centralized interface to per-turbine observations and actions. This approach minimizes code duplication and ensures consistent behaviour across control paradigms.

We deliberately adopted the Gymnasium and PettingZoo APIs as they represent the de facto standards in RL research. This decision lowers the barrier to entry for researchers already familiar with these interfaces and enables seamless integration with popular training libraries such as *Stable-Baselines3* (Raffin et al., 2021) and *CleanRL* (Huang et al., 2022).

The simulation back-end is abstracted behind a clean interface, allowing users to swap between DYNAMIKS for dynamic simulations and *PyWake* for steady-state analysis without modifying their RL code. This modularity supports diverse research directions, whether investigating large-scale RL training, robust control under uncertainty, or algorithm comparisons across fidelity levels.

Flexibility is maintained throughout: reward functions, observation spaces, and termination conditions are all configurable, enabling researchers to adapt the environment to their specific research questions rather than being constrained by rigid defaults.

Functionality

WindGym supports both centralized and decentralized control formulations. In the single-agent variant, a single controller issues actions for the entire farm following the Gymnasium API. In the multi-agent variant following the PettingZoo API, each turbine maps to its own agent with separate observation and action spaces, allowing researchers to switch between paradigms with minimal code changes.

The package provides interchangeable physics back-ends: DYNAMIKS for dynamic, higher-fidelity transient simulations, and *PyWake* for fast, analytical wake models. These can be swapped without altering the RL setup, enabling researchers to trade off speed and fidelity as needed.

Reward specification is a central feature. WindGym includes utilities for common formulations such as raw power, baseline-normalized power, and delta-power rewards, as well as optional penalty terms. Users can also implement custom reward functions.

Finally, reproducibility is a core concern. The environment is tested for consistency of

observation and action spaces, correct termination behavior, and deterministic toggles. Continuous integration and curated examples help ensure that results can be reproduced across setups.

WindGym includes predefined example environment presets that bundle turbine layouts, physics back-end, and configuration into a single function call. The following minimal example creates a three-turbine row, runs one episode with random actions, and prints the cumulative reward:

```
from WindGym.presets import three_turbine_row
import numpy as np

env = three_turbine_row()           # V80 turbines
obs, info = env.reset()
total_reward = 0.0
for _ in range(50):
    action = env.action_space.sample()
    obs, reward, terminated, truncated, info = env.step(action)
    total_reward += reward
    if terminated or truncated:
        break
env.close()
```

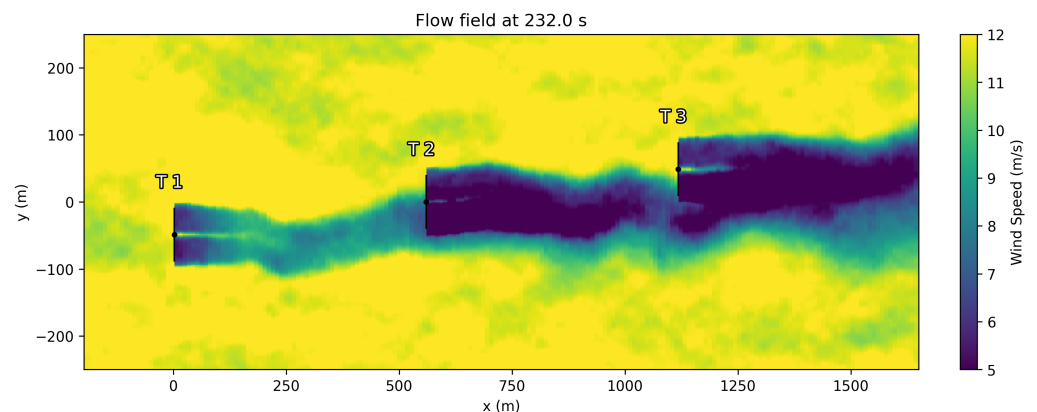


Figure 1: Simulated wind speed flow field from a three-turbine row environment, showing wake interactions between V80 turbines at hub height. Wind flows from left to right; the velocity deficit behind each turbine illustrates the cascading wake effect that WindGym environments capture.

Additional presets (`two_by_two_grid`, `nine_turbine_grid`) and a companion Quick Start notebook are included in the repository.

The full documentation of the library is available at <https://sys.pages.windenergy.dtu.dk/windgym/>

Research Impact Statement

WindGym is designed to serve as a shared experimental platform for the wind energy RL community. To our knowledge, four research papers currently utilize WindGym as their experimental environment, demonstrating its adoption for novel contributions in RL-based wind farm control (Åstrand et al., 2026; Nilsen, Quick, et al., 2026; Nilsen, Åstrand, et al., 2026; Quick et al., 2026). The package integrates with the widely adopted Gymnasium and PettingZoo APIs, enabling direct compatibility with mainstream RL training libraries such as

Stable-Baselines3 (Raffin et al., 2021) and CleanRL (Huang et al., 2022), which lowers the barrier for researchers to adopt WindGym in their existing workflows.

The package is designed for community readiness: comprehensive documentation explains core concepts and usage patterns, worked examples demonstrate training and evaluation workflows, and an extensive test suite ensures reliability across updates. WindGym is developed and maintained at the Technical University of Denmark (DTU) and is openly available through both GitHub and GitLab, where we actively encourage external contributions.

AI Usage Disclosure

The WindGym codebase was initiated before the widespread adoption of large language models and coding assistants, with the foundational architecture developed without AI assistance. As these tools matured, they were incorporated into the development workflow in the following ways: refactoring existing code for improved consistency and maintainability, generating documentation content, and developing a substantial portion of the unit test suite. All AI-generated code was reviewed and validated by human developers before integration.

For this paper, AI tools were used to provide feedback on clarity and wording during the drafting process. Grammarly was used for grammar and style checking. No content was generated wholesale by AI without human review and revision.

References

- Abkar, M., Zehtabiyani-Rezaie, N., & Iosifidis, A. (2023). Reinforcement learning for wind-farm flow control: Current state and future actions. *Theoretical and Applied Mechanics Letters*, 13(6), 100475. <https://doi.org/10.1016/j.taml.2023.100475>
- Annoni, J., Fleming, P., Scholbrock, A., Roadman, J., Dana, S., Adcock, C., Porté-Agel, F., Raach, S., Haizmann, F., & Schlipf, D. (2018). Analysis of control-oriented wake modeling tools using lidar field results. *Wind Energy Science*. <https://doi.org/10.5194/wes-3-819-2018>
- Åstrand, T., Nilsen, M. B., Tsaklis, I., Göçmen, T., Réthoré, P.-E., & Dimitrov, N. (2026). *Load constrained wind farm flow control through multi-objective multi-agent reinforcement learning*. <https://arxiv.org/abs/2604.22795>
- Göçmen, T., Liew, J., Kadoche, E., Dimitrov, N., Riva, R., Andersen, S. J., Lio, A. W., Quick, J., Réthoré, P.-E., & Dykes, K. (2025). Data-driven wind farm flow control and challenges towards field implementation: A review. *Renewable and Sustainable Energy Reviews*, 216, 115605. <https://doi.org/10.1016/j.rser.2025.115605>
- Howland, M. F., & Dabiri, J. O. (2020). Influence of wake model superposition and secondary steering on model-based wake steering control with SCADA data assimilation. *Energies*. <https://doi.org/10.3390/en14010052>
- Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., & Araújo, J. G. M. (2022). CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274), 1–18. <http://jmlr.org/papers/v23/21-1342.html>
- IEA. (2021). *Net zero by 2050*. International Energy Agency. <https://www.iea.org/reports/net-zero-by-2050>
- International Renewable Energy Agency (IRENA). (2022). *World energy transitions outlook 2022: 1.5°C pathway*. IRENA. ISBN: 978-92-9260-429-5
- Jonkman, J. M., Annoni, J., Hayman, G., Jonkman, B., & Purkayastha, A. (2017).

- Development of fast. Farm: A new multi-physics engineering tool for wind-farm design and analysis. *35th Wind Energy Symposium*, 0454. <https://doi.org/10.2514/6.2017-0454>
- Monroc, C. B., Bušić, A., Dubuc, D., & Zhu, J. (2025). *WFCRL: A multi-agent reinforcement learning benchmark for wind farm control*. <https://doi.org/10.52202/079017-4235>
- Neustroev, G., Andringa, S. P. E., Verzijlbergh, R. A., & Weerdt, M. M. de. (2022, May). Deep reinforcement learning for active wake control. *International Conference on Autonomous Agents and Multi-Agent Systems*.
- Nilsen, M. B., Åstrand, T. O. B., Göçmen, T., & Réthoré, P.-E. (2026). *Hierarchical RL-MPC control for dynamic wake steering in wind farms*. <https://arxiv.org/abs/2604.22797>
- Nilsen, M. B., Quick, J., Göçmen, T., Dimitrov, N., & Réthoré, P.-E. (2026). *Accelerating reinforcement learning for wind farm control via expert demonstrations*. <https://arxiv.org/abs/2604.22794>
- NREL. (2025). *FLORIS*.
- Pedersen, M. M., Steiner, J., Nilsen, M. B., Lohmann, J., Hodgson, E. L., Riva, R., Troldborg, N., Andersen, S. J., Larsen, G., Verelst, D. R., & Réthoré, P.-E. (2026). *Dynamiks 0.0.4: An open-source dynamic wind system simulator*. <https://gitlab.windenergy.dtu.dk/DYNAMIKS/dynamiks>
- Quick, J., Nilsen, M. B., Bechmann, A., Le, T. N., & Rethore, P.-E. M. (2026). *Adversarial sensor errors for safe and robust wind turbine fleet control*. <https://arxiv.org/abs/2604.08750>
- Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1–8. <http://jmlr.org/papers/v22/20-1364.html>
- Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L. S., Dieffendahl, C., Horsch, C., Perez-Vicente, R., & others. (2021). Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 15032–15043.
- Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., & others. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv Preprint arXiv:2407.17032*.
- Vondelen, A. V. (2024). Synchronized dynamic induction control: An experimental investigation. *Journal of Physics Conference Series*. <https://doi.org/10.1088/1742-6596/2767/3/032027>