Universiteit Leiden

Master Computer Science

A Quasi-Dynamic Wind Farm Reinforcement Learning Environment with Wake Transport and Partial Observability

Name:	Alessandro Marincioni
Student ID:	s3442209
Date:	01/04/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Álvaro Serra-Gómez
2nd supervisor:	Thomas Moerland

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

1 Introduction **3**

2 Related Work **5**
- 2.1 Wind farm simulators . . . 5
- 2.2 Wind farm optimal control approaches . . . 7

3 Methods **8**
- 3.1 Wind farm Gym environments . . . 9
  - 3.1.1 Constrained optimization problem formulation . . . 9
  - 3.1.2 Fixed wind environment . . . 13
  - 3.1.3 Changing wind environment . . . 14
  - 3.1.4 Quasi-dynamic environment . . . 16
- 3.2 Algorithms . . . 22
  - 3.2.1 Proximal Policy Optimization . . . 22
  - 3.2.2 Floris Serial-Refine . . . 23
- 3.3 Environment layouts . . . 24
- 3.4 Experimental design . . . 26
  - 3.4.1 Floris Serial-Refine Analysis . . . 26
  - 3.4.2 Fixed and changing wind environment . . . 27
  - 3.4.3 Quasi-dynamic environment . . . 28
  - 3.4.4 Computational efficiency comparison . . . 29

4 Results **29**
- 4.1 Floris Serial-Refine . . . 30
- 4.2 Fixed wind environment . . . 31
- 4.3 Changing wind environment . . . 33
- 4.4 Quasi-dynamic environment . . . 34
- 4.5 Computational efficiency comparison . . . 36

5 Discussion **37**
- 5.1 Contributions of the study to the literature . . . 37
- 5.2 Limitations . . . 38
- 5.3 Future work . . . 38

6 Conclusion **39**

A Hyperparameter search and training details **44**

B Convergence of PPO training **44**

Abstract

Wind farms are an important source of renewable energy, but their efficiency is affected by wake interactions between turbines. Reinforcement learning (RL) has emerged as a promising approach to optimize wind farm control and mitigate power losses due to wake effects. However, RL development is constrained by the choice of training environments, which are either too simplified and may not realistically depict the problem or too computationally expensive to train RL agents over many trajectories.

This work introduces an environment for quasi-dynamic wind farm control, balancing computational efficiency, that enables RL training while maintaining physical realism. The proposed environment incorporates wake transfer effects and partial observability, both important aspects of real-world wind farm control which controllers must consider to optimize power generation.

Through experiments, we evaluate the performance of various controllers on steady-state and quasi-dynamic environments. Our results indicate that baselines and learned policies developed for steady-state environments have limited transferability to quasi-dynamic environments, with a decrease in generated power of 0.14% in our larger experiments when compared to the power generated by aligning the turbines directly with the upstream wind. In contrast, even relatively simple RL agents can effectively learn control strategies in the more complex and high-dimensional quasi-dynamic settings, increasing the power output by 0.33% in the same settings. These findings highlight the importance of environment design in wind farm control and the potential of RL agents. Through the development of the fast, quasi-dynamic environment, this study facilitates research on such agents, helping in the development of more efficient wind farm control strategies.

1 Introduction

Wind energy is a renewable, abundant and secure power source. As such, it plays an important role in meeting the European Union's decarbonization objectives while delivering clean and affordable electricity to households and industry. The EU's renewable energy target of at least 42.5% by 2030 will require wind energy capacity to increase, among other renewable sources [1]. This expansion plan shows the importance of maximizing the efficiency of both existing and future wind farm installations. One aspect of wind power production is optimal control, which aims to reduce the impact of wake effects and maximise power production. When wind passes through a turbine, it creates a region of reduced wind speed and increased turbulence downstream, known as a wake. These wakes can reduce the power output of downstream turbines significantly, with studies showing intra-farm efficiency losses in the 10%-20% range [2] [3]. Optimizing the control strategy of individual turbines can mitigate these wake effects and increase the total power output of the wind farm.

Different approaches exist to tackle this optimal control problem, from static yaw offset strategies to learned control policies [4]. Methods for optimizing wind farm control in reinforcement learning (RL) often rely on simulators to train or pretrain the agents. Deploying untrained agents directly in a real wind farm is not feasible because it would cause loss of generated energy and potential damage to structures. In wind farm optimal control, RL agents use wind farm simulators to receive rewards based on how much power their control policy is capable of generating. The observations provided to these agents are very limited, typically including the yaw of the turbines, some local wind information at the turbine, and the general wind speed

and direction. Occasionally, extra wind data from wind masts is provided, but this information remains limited and localized. Various simulation environments have been developed to model the complex aerodynamic interactions between turbines and their wakes. These simulators serve as development and validation tools for wind farm control algorithms. However, existing control-oriented wind farm simulators typically fall into two categories: they are either computationally expensive to run with high accuracy or fast with reduced fidelity.
High-fidelity simulators usually run computational fluid dynamics (CFD) calculations to generate realistic flow fields and which are used to compute power production by turbines [5, 6]. SOWFA [7] is an example of a high fidelity simulator. It computes Large Eddy Simulations (LED) to simulate wind in the wind farm. Medium-fidelity simulators sacrifice some of the simulation quality in favor of speed. An example of such an environment is WFSim [8], which models wind dynamics through two-dimensional Navier-Stokes equations. Although WFSim simulations feature more realistic dynamics than low-fidelity simulators, they are unable to model changing wind directions as boundary conditions, limiting their applicability. In contrast, FLORIS [9] is a fast, low-fidelity simulator which runs steady-state computations of the flow field in wind farms. It is commonly used in wind farm optimization due to its speed and ease of use, as Gym versions of the environment exist. While this environment allows for fast simulation speed, all transient dynamics of the system, such as wake transport effects, are lost due to the fact that the computed flow fields are steady-state. Moreover, the simplicity of the simulation makes the environment fully observable through the measurements of sensors that are commonly available in wind farms. This gap between the properties of the simulated environment and real wind farms is problematic, as it may hinder the transfer of models and results derived from FLORIS to real-life scenarios.
To address this limitation, multiple methods have been developed to run quasi-dynamic wind farm simulations starting from low-fidelity steady-state simulations, like FLORIDyn [10] and FOWFSimDyn [11]. While implmentations of these wind farm models exist, they aim to simulate the wind farm system and are not suitable for optimal control tasks featuring agent-system interactions like the training of reinforcement learning agents. This hinders the applicability of these simulators and makes it difficult for practitioners to use them to develop new algorithms. This impact is observed in the literature, as most studies train RL agents on FLORIS and WFSim, where it is not possible to effectively simulate changing wind directions.

This work aims to address the gap between existing wind farm simulators and reinforcement learning needs by implementing a simplified version of the FLORIDyn dynamic wake modeling in a standardized Gym environment. The FLORIDyn approach is based on the idea of sampling steady-state information at each timestep through sets of points and translating them according to wind conditions. The accumulation of these points over multiple timesteps is then used to generate a flow field which features the movement of wake effects, despite being derived from steady-state simulations. This is implemented within a control-oriented framework using the FLORIS Gym environment as the foundation.

Experiments on simulation speed show that the proposed dynamic environment runs significantly faster than existing alternatives while maintaining only a limited performance loss compared to the FLORIS environment. The results of the experiments on agent performance over different environments indicate that while strong baselines for FLORIS, such as Floris Serial-Refine, outperform other methods in the FLORIS environment, they fail to transfer effectively to the new dynamic environment. In contrast, even simple reinforcement learning (RL) agents

demonstrate the ability to be trained effectively in the dynamic environment, suggesting that RL approaches may be better suited for handling high-dimensional, partially observable wind farm control tasks.

The scope and contributions of the study are as follows. This work aims to address the gap between existing RL wind farm simulators and real-life wind farms. As training RL agents can require running a significant amount of simulations, this study aims to create simulators with low computational cost, while bridging the reality gap of current simulators by generating quasi-dynamic flow fields from arbitrary freestream wind conditions. The contributions of the study are the following:

- We provide a fast, quasi-dynamic wind farm simulator that enables the study of optimal control in wind farms with partially observable conditions. The environment is shown to be capable of computing simulations faster than real-time, achieving speeds over 65,000$\times$ faster than SOWFA, making it an efficient alternative for generating quasi-dynamic flow fields. It also features a Gym interface for easy integration and use for agent based modelling.
- We conduct experiments comparing existing baselines to simple reinforcement learning (RL) agents, both in steady-state and quasi-dynamic environments, to evaluate how the different environments affect the performance and applicability of existing methods.
- We find that RL control policies can outperform the baselines in quasi-dynamic environments.

This document is organized as follows. Section 2 discusses related work, providing an overview of existing simulation environments and optimal control methods for wind farm control. Section 3 details the methodology, including the problem formulation, the description of the dynamics of the different environments that are used in the study, the wind turbine layouts, the controllers evaluated in the study and the experimental setup. The results, presented in Section 4, include experiments on the Serial-Refine baseline, comparisons of different controllers across environments, and a brief computational efficiency analysis. Section 5 provides a discussion of the findings, followed by the conclusions in Section 6. Finally, additional details on the hyperparameter search for PPO agents and the convergence of training are in the Appendix A and B respectively.

2 Related Work

This section reviews previous research in wind farm control, covering both simulation environments and control strategies. Section 2.1 summarizes existing wind farm simulators, focusing on their fidelity and applicability for control tasks. Next, Section 2.2 describes different techniques for wind farm control, including both model-based and model-free methods.

2.1 Wind farm simulators

Different control oriented simulators have been used in the literature to study the control of wind farms. These simulators have different levels of complexity to balance the trade-off between simulation fidelity and computational costs. High-fidelity simulators aim to generate

realistic simulations which represent the underlying physical dynamics of the wind accurately, whereas low-fidelity simulators implement simplified wind dynamics which can be computed more quickly. Three simulators that are commonly used in wind farm control research are FLORIS [9], WFSim [8] and SOWFA [7].

- SOWFA is a high-fidelity simulator that models the wind dynamics in a wind farm through a three-dimensional Large Eddy Simulation (LES) flow field model. This simulator is capable of representing flow field dynamics with high accuracy, but this higher quality of simulation comes at a significant computational cost. This makes the simulator a poor choice when a high number of trajectories need to be computed.

- WFSim is a medium-fidelity simulator. At each timestep, it solves a two-dimensional form of the unsteady turbulent Navier-Stokes equations along a horizontal plane located at the hub height of the wind turbines within a wind farm. The two-dimensional approximation of the system allows for faster computation, but makes the simulation less realistic, neglecting three-dimensional wind dynamics effects. Nonetheless, additional measures are taken to limit the negative impact of this approximation. Although this method can simulate transient wake effects, its formulation assumes that the upstream boundary wind conditions are fixed in direction. This limits the applicability to control tasks, as changing wind conditions cannot be modeled with this simulator.

- FLORIS is a low-fidelity steady-state control focused wind farm simulator. It is a common choice of environment for RL research for wind farm optimal control due to its relatively low computational cost and ease of use, compared to the other available simulators. It is implemented in Python and a Gym [12] wrapper is available to allow its usage in conjunction with other RL frameworks. FLORIS simulations model the wind dynamics as steady-state computations. To do so the simulator uses the current wind speed, wind direction and turbine yaws to compute the state of the wind throughout the wind farm. This kind of simulation ignores any transient dynamics of the wind throughout the farm, such as wake meandering. Because of this, wake effects caused by turbines propagate instantly through the wind farm at each timestep.

Although the three methods offer different speed-accuracy trade-offs, they all have critical shortcomings. SOWFA is computationally prohibitive for experiments that require many trajectories. WFSim and FLORIS, while more efficient, introduce approximations that compromise accuracy or generalizability, limiting their applicability in real-world scenarios. WFSim is limited to simulations with fixed freestream wind directions, and FLORIS computes steady-state simulations.
Quasi-dynamic low-fidelity simulators were developed to overcome the limitations of the aforementioned approaches. These simulators aim to model wake transport or wake meandering while keeping computational costs reasonable. Two examples of such approaches are FloriDyn [13, 10] and FOWFSimDyn [11]. FloriDyn uses steady-state simulations created with FLORIS and simulates wind dynamics by propagating the wake computed by FLORIS through the wind farm over time. To introduce wind dynamics, the FloriDyn model introduces observation points (OPs). These OPs are used to describe the local wake characteristics of FLORIS at their specific location and are generated behind wind turbines at each timestep. As time progresses, the OPs move downstream according to freestream wind conditions, representing the mass of air traveling through the wind farm. This approach allows the wake effects to travel through the

wind farm and be transported by the wind, generating effects of wake transport. This makes the environment simulations more complex than the ones from FLORIS, where wake transport phenomena are absent.
FOWFSim-Dyn is a dynamic simulator designed to model floating offshore wind farms, capturing the coupled effects of wake propagation and platform motion under time-varying wind conditions. It simulates wake dynamics by modeling the wake centerline propagation and momentum recovery. It also simulates the dynamics of offshore platforms.
As this study focuses on simulating quasi-dynamic wind conditions without platform dynamics, the FloriDyn formulation is chosen as a reference for the development of the quasi-dynamic environment. This allows the proposed environment to generate simulated trajectories quickly while still featuring complex wind dynamics. Contrary to WFSim and FLORIS, this environment allows for both changing freestream wind conditions and complex wind dynamics, and can be used to find control policies in more challenging settings.

2.2 Wind farm optimal control approaches

Wind farm optimal control methods are usually separated into two types: model-based and model-free approaches.

Model-based approaches aim to model the wind optimal control task as a constrained optimization problem. These methods often feature an analytic representation of the relationship between yaw settings of turbines and generated power or wake effects, and they use various approaches to find yaw settings that optimize power. Due to the complexity of wind dynamics, wind simulators cannot be used directly, and surrogate models which model wind dynamics in tractable analytic forms are used instead. These surrogate models allow the problem optimization task to be tractable and to obtain optimal yaw settings for different wind conditions. Studies have proposed to model the wind farm system through simplified parametric wake effect models [14, 15, 16], and used these surrogate models to develop control strategies. While the refinement of surrogate models has allowed to improve control strategies, these methods have been shown to be sensitive to unmodeled dynamics and uncertainties in the system. Moreover, the actual performance of these model-based controllers can differ significantly from the analytical results.

Model-free approaches have been explored by other studies to address these limitations of model-based methods. These methods optimize control policies without relying on a predefined mathematical model of the system. Instead of analytically solving for optimal values, they learn from observed interactions between control variables (e.g., turbine yaw angles), system parameters (e.g., wind speed, direction, and atmospheric conditions), and target outputs (e.g., power generation). These methods allow to decouple the optimal control and wind farm simulation tasks, and thus avoid the problem of constructing an accurate analytic wind farm power function. In these approaches, data of the system dynamics are either provided by real-life measurements or generated synthetically by simulators, and different optimization methods are applied to the data, agnostic of the underlying system dynamics formulation. Multiple data-driven approaches have been proposed, including a game-theoretic search algorithm [17], a gradient ascent algorithm [18], maximum power point tracking methods [19, 20], a Bayesean optimization approach [21] and more.

Reinforcement learning methods are data-driven approaches of particular interest, as deep reinforcement learning methods can address the high system complexity of wind farms, while also being robust to uncertainties and noise. These methods learn a control policy which optimizes a reward signal by interacting with an environment. Various RL approaches have been proposed to optimize wind farm control in recent years [22]. Most studies developed agents using FLORIS and WFSim environments, with SOWFA being an uncommon choice due to its low speed. These studies show the capability of RL agents for wind farm control, and indicates a growing interest in developing such methods.

Different approaches have been used to train agents in the FLORIS environment. Multi-agent approaches have been developed to distribute the wind farm control task at the turbine level [23, 24]. One study has trained TD3 and SAC agents, providing a variety of sensor data to the agents and evaluating the impact of action representation on model performance [25]. The transfer of FLORIS-trained agents to dynamic environments has also been considered, to address the high computational cost of training in dynamic environments [26].

Multiple studies have tested variations of DDPG agents and have applied them on the WFSim environment, augmenting the DDPG architecture with double networks [27], a composite experience replay [28] or knowledge assisted methods to take into account the wear and tear of the infrastructure [29].

Finally, one study has trained RL models using high-fidelity simulation data from SOWFA [30]. It introduces a reward regularization module to address the non-markovian aspect and stochasticity of SOWFA simulations, and splits training in offline and online RL phases.

Most works presented in this section focus on low-fidelity and medium-fidelity environments to avoid the computational cost of training RL agents in dynamic simulators. In studies that use low-fidelity FLORIS simulations, the quality of simulation is significantly lower, which limits the transferability of results to real-life control tasks. Studies that use medium-fidelity WFSim simulations can only simulate fixed wind directions, limiting the scope of the study to simple wind conditions. A specific training process is devised to limit the number of computations of SOWFA simulations for the work that uses the high-fidelity dynamic SOWFA simulator. First, a set of trajectories is generated to train a DDPG model offline. After that, the model is trained online: the learned policy is used to generate 700s of SOWFA trajectories, which are then used to update the model. This online training process is repeated twice. While this process allows to limit the generated trajectories to 1400s of SOWFA simulations for online training, trajectories are created using only two versions of the control policy, which limits the exploration of policies significantly.

This work aims to address the issues of the different simulators used in the training of control policies for wind farm control and make the use of quasi-dynamic environments more accessible. This is done by providing a fast quasi-dynamic environment, facilitating the development of better data-driven methods for wind farm control.

3 Methods

This section outlines the methodologies used in the study, focusing on wind farm control environments and optimization algorithms. Section 3.1 introduces the simulation environments used in the study, including a formulation of wind farm control as a constrained optimization problem and MDP formulations. Section 3.2 details the control algorithms: Proximal Policy Optimization (PPO) for reinforcement learning and Floris Serial-Refine as a baseline. Section

3.3 describes the environment layouts, while Section 3.4 presents the experimental design, analyzing the Floris Serial-Refine baseline, algorithm performance across the different environments, and computational efficiency of the proposed environment.

3.1 Wind farm Gym environments

This section describes the environments that are used for the experiments of this study. Three environments are considered. The first two are based on the FLORIS [9, 31] simulator, and use it to compute steady-state simulations for the flow field. The third environment is the one introduced in this study and it computes the flow field through a quasi-dynamic formulation inspired by FloriDyn. The first steady-state environment is a fixed wind environment, where freestream wind conditions are constant throughout each episode. The second steady-state environment features changing freestream conditions within episodes. The quasi-dynamic environment is the most complex environment of the three, featuring changing freestream wind conditions, a much larger state space, wake transport effects and partial observability.
The optimal control problem of maximising power generation in a wind farm is formulated as a constrained optimization problem in Section 3.1.1. The implementation details, properties and MDP formulations of each environment are described in the following sections, with Section 3.1.2 describing the fixed wind environment, Section 3.1.3 describing the changing wind environment and 3.1.4 describing the quasi-dynamic environment. For each type of environment, multiple environments are created with different wind turbine layouts and wind farm sizes to run experiments at different scales. The layouts used in the experiments are presented in Section 3.3.

3.1.1 Constrained optimization problem formulation

The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be

formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim$

$\mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to actions a_t. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field is represented as F_t. In steady-state

environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. The problem of finding an optimal control policy for a wind farm to maximize power generation can be formulated as a constrained optimization problem. We consider trajectories of length T with timesteps $t \in \mathcal{T} = \{1, 2, \ldots, T\}$. The number of wind turbines in the wind farm is defined as N and turbines and their parameters are assigned indices $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. The freestream wind speed and direction are defined as v_t and ϕ_t. The freestream wind direction is initialized as a random direction $\phi_0 \sim \mathcal{U}(0, 2\pi)$ and the freestream wind speed is initialized within a range $v_0 \sim \mathcal{U}(v_{\min}, v_{\max})$. The yaw offsets of each turbine i with respect to the freestream wind direction are represented as $\psi_{i,t}$ and are initialized within a range $\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max})$. The vector of yaw offsets of all turbines at a given timestep is defined as $\boldsymbol{\psi}_t = [\psi_1, \ldots, \psi_N]$. The vector representing the information provided to the control policy at each timestep is defined as o_t and it includes information about the freestream wind conditions and the yaw offsets of all turbines. The turbine yaws are controlled according to the actions a_t of a control policy $\pi(o_t)$. The actuation function $f(\boldsymbol{\psi}_t, a_t)$ rotates the wind turbines according to the control policy and actuation constraints of speed and maximum angle, simulating the turbine dynamics. The freestream wind parameters evolve according to a function $g(\phi_t, v_t)$. The flow field of the wind farm is the speed and direction of the wind throughout the wind farm. F_t represents the flow field at a given timestep. In steady-state environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$ which takes the current freestream parameters and turbine yaw offsets as inputs. In quasi-dynamic environments the flow field is computed by a function $\mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ which also takes the representation of the flow field computed at the previous timestep as input. The initial conditions for the flow field in the quasi-dynamic case F_0 represent the steady-state freestream flow. At each timestep, each turbine generates power according to a function $\mathcal{P}_{i,t}(F_t, \psi_{i,t}^{abs})$ which depends on the flow field and the absolute orientation of the turbine $\psi_{i,t}^{abs} = \phi_t + \psi_{i,t}$.

$$\max_{\pi} \quad \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}} \mathcal{P}_{i,t}(F_t, \psi_{i,t}^{abs}) \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}$$

s.t.

$$\phi_0 \sim \mathcal{U}(0, 2\pi) \qquad \text{(Initial wind direction)}$$

$$v_0 \sim \mathcal{U}(v_{\min}, v_{\max}) \qquad \text{(Initial wind speed)}$$

$$\psi_{i,0} \sim \mathcal{U}(-\psi_{\max}, \psi_{\max}) \qquad \forall i \in \mathcal{N} \text{(Initial turbine yaws)}$$

$$o_t = [\phi_t, v_t, \boldsymbol{\psi}_t] \qquad \forall t \in \mathcal{T} \text{(State observation)}$$

$$a_t = \pi(o_t) \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \text{(Control policy)}$$

$$\boldsymbol{\psi}_{t+1} = f(\boldsymbol{\psi}_t, a_t) \qquad \forall t \in \mathcal{T} \text{(Actuation function)}$$

$$\boldsymbol{\psi}_t^{abs} = \boldsymbol{\psi}_t + \phi_t \qquad \forall t \in \mathcal{T} \text{(Absolute turbine orientation)}$$

$$\phi_{t+1}, v_{t+1} = g(\phi_t, v_t) \qquad \forall t \in \mathcal{T} \setminus \{T\} \text{(Freestream wind dynamics)}$$

$$F_t = \begin{cases} \mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t) & \text{if steady-state or t=0} \\ \mathcal{M}_{QD}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1}) & \text{else} \end{cases} \qquad \forall t \in \mathcal{T} \text{(Flow field model)}$$

3.1.2 Fixed wind environment

The fixed wind environment is the simplest environment used for this study. For each episode, the initial wind direction ϕ_0 and wind speed v_0 are chosen at random and kept constant throughout the episode: $g(\phi_t, v_t) = \phi_0, v_0$.
As the flow field computation is done under a steady-state assumption and the freestream wind conditions are constant, the only variable that affects the change of the flow field across timesteps of an episode is the set of turbine yaw offsets $\boldsymbol{\psi}_t$. Thus, to optimize the power production in this environment, controllers need to find the optimal yaw configuration given the freestream wind conditions and find efficient trajectories to achieve them. The Floris simulator computes the steady-state flow field of the wind throughout the wind farm given the freestream wind conditions and yaws of the turbines $F_t = \mathcal{M}_{SS}(\phi_0, v_0, \boldsymbol{\psi}_t)$. In principle, the computation of the state of a flow field would require complete knowledge of the state of the flow field at the previous timestep. In the case of steady-state computations information about the flow field at the previous timestep is not required, as the steady-state of the flow field is not affected by transient dynamics. Figure 1 shows a rendering of the fixed wind environment at two timesteps.

MDP formulation

The wind farm optimal control problem is formulated as a MDP which is defined by the tuple (S, A, P, R, γ). As Floris computes the steady-state of the flow field, it only requires information about the freestream wind and turbine yaws to compute a simulation step. Consequently, the state of the environment $s_t \in S$ can be defined as the list $s_t = [\phi_t, v_t, \boldsymbol{\psi}_t] = [\phi_0, v_0, \boldsymbol{\psi}_t]$. The action $a_t \in A$ represents the list target yaws that each turbine aims to achieve at the next timestep. Each yaw can be encoded as an absolute target angle or relative target angle with respect to the current freestream wind direction. The choice of representation depends on the controller policy. Actions are provided by a policy $a_t = \pi(o_t)$ which takes sensor data observations o_t as input. In this environment the observation vector o_t matches the state vector s_t, i.e. the environment is fully observable, thus $a_t = \pi(o_t) = \pi(s_t)$. P represents the transition probability distribution $P(s_{t+1}|s_t, a_t)$. In the fixed wind environment the freestream wind conditions are constant, and the yaws of the turbines rotate according to the target defined by a_t. The rotation is constrained by a maximum absolute angle offset with respect to the freestream wind, and a rotation speed limit. The reward function $R(s_t, a_t)$ is a scaling of the total power generated in the wind farm. The power is computed as $\mathcal{P}_{i,t}(F_t, \psi_{i,t}^{abs})$, and depends on the flow field and the current yaws of the turbines, which are computed from the current state. Finally, γ is the discount factor.

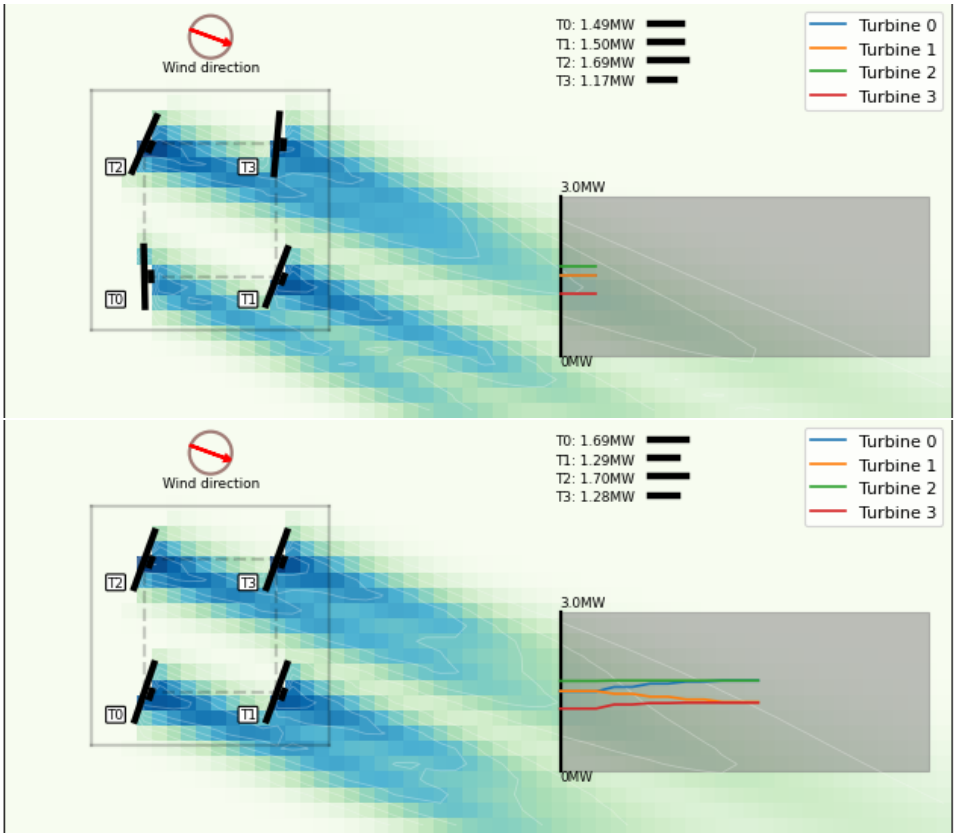

Figure 1: Rendering of the fixed wind environment at two different timesteps. Top picture shows the rendering at the first timestep after initialization and bottom picture shows the environment at timestep 5. The colored background indicates the intensity of the wake effect caused by the four turbines, shown here as black lines. Each turbine is assigned a number and information about power generation is shown on the right side of the pictures. Note that the wind conditions are fixed throughout the episode, as shown by the wind direction indicator at the top of each figure not changing. Because of this, changes in the wake effects are entirely driven by the orientations of the turbines.

3.1.3 Changing wind environment

The second type of environment that is used in this study is a Floris environment with changing freestream wind conditions. In this environment it is possible to define a wind process that dictates the changes in freestream wind conditions at each timestep. For this study, the wind process used to model changing freestream wind conditions implements a random walk of wind direction and speed. In practice $g(\phi_t, v_t)$ samples new values for wind direction and speed uniformly within a range: $\phi_{t+1} \sim \mathcal{U}(\phi_t - \epsilon_{\phi_-}, \phi_t + \epsilon_{\phi_+})$, $v_{t+1} \sim \mathcal{U}(v_t - \epsilon_{v_-}, v_t + \epsilon_{v_+})$. Freestream velocity is also clipped to be within a set range. The maximum direction change of the wind is 3 degrees per timestep and the maximum wind speed change is 1m/s per timestep. The Floris simulator computes the steady-state flow field of the wind throughout the wind farm given the freestream wind conditions and yaws of the turbines $F_t = \mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$. Note that contrary to the fixed wind environment, the freestream wind conditions change at each timestep in this environment. This means that given starting freestream wind conditions, the flow field is no longer fully controllable by actuating the wind turbine, and depends on factors that the controllers have no impact over. Thus, the optimal yaw configuration for the windfarm turbines changes at every timestep. The flow field is modelled as steady-state and computed

through Floris. Note that as opposed to the fixed wind environment, this computation now depends on the changing freestream wind conditions: $F_t = \mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$.
This change in the environment ensures that, given an appropriate choice of wind process, the environment will be closer to real wind farms, where the wind conditions change over time, and controllers have to take these changes into account to maximise power. Figure 2 shows a rendering of the changing wind environment at three timesteps.

MDP formulation

The wind farm optimal control problem is formulated as a MDP which is defined by the tuple (S, A, P, R, γ). The MDP can be formulated similarly to the case of the fixed wind environment. The actions $a_t \in A$ are defined in the same way, as the environment is still fully observable: $a_t = \pi(o_t) = \pi(s_t)$ and $o_t = s_t$. The reward function $R(s_t, a_t)$ and discount factor γ are also defined in the same way as in the fixed wind environment. The state of the environment s_t now features changing wind directions and is defined as $s_t = [\phi_t, v_t, \boldsymbol{\psi}_t]$. The transition probability function $P(s_{t+1}|s_t, a_t)$ implements the turbine actuation in the same way as the fixed wind environment, and the random walk wind process as described above.

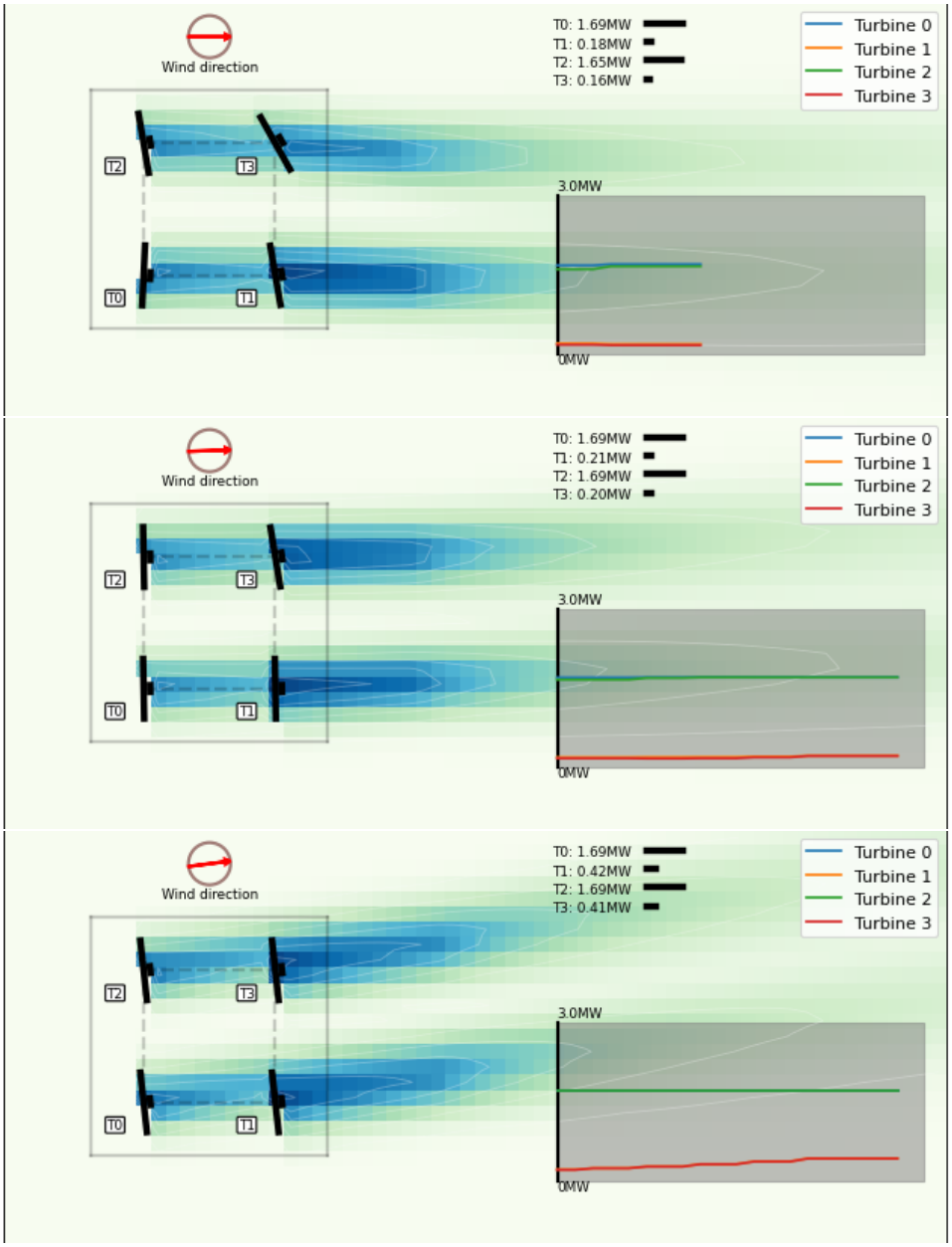

Figure 2: Rendering of the changing wind environment at three different timesteps. Pictures from top to bottom show the rendering at timesteps 1, 5 and 10. The colored background indicates the intensity of the wake effect caused by the four turbines, shown here as black lines. Information on the turbines, wind and power generation is shown in the same way as in the fixed wind environment. Note that in this environment the wind conditions change over the episode, as shown by the wind direction indicator changing over time.

3.1.4 Quasi-dynamic environment

This study introduces a novel type of environment for RL based on the FloriDyn approach. This environment, which is referred to as quasi-dynamic in this work, is the third type of environment used in this study. It aims to bridge the reality gap of existing RL wind farm environments by implementing system dynamics where the evolution of the flow field of the wind in the wind farm is no longer steady-state, but depends on the previous state of the

wind instead. The evolution of the flow field over time is modeled through a set of points, called observation points, which travel through the wind farm and carry information about steady-state wake effects. The flow field computation for this environment is expressed as $F_t = \mathcal{M}_{QD}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$ to reflect the dependence on previous states of the flow field. In this formulation, flow field representation F_t includes both information about the wind speed and direction of the wind throughout the wind farm, and information about the current set of observation points, which are used to update the flow field at the next timestep. The freeflow wind conditions change in the same way as in the changing wind environment, following a random walk of the ϕ_t and v_t parameters.

The increased complexity of this environment makes it so the observation vector o_t is no longer capable of fully capturing the true state of the environment. Because of this, the Sequential Decision Problem (SDP) of this environment is expressed as a Partially Observable Markov Decision Process (POMDP) to account for the partial observability of the environment state.

Quasi-dynamic flow field computation

The environment dynamics are a simplified version of the ones described in the FloriDyn paper. They are based on the idea of transporting information of steady-state wake effects throughout the wind farm through so-called observation points. The generation and evolution of observation points over time and flow field computation are described in Algorithm 1. The computation of generated power and rewards from the current flow field is described in Algorithm 2.

The step-by-step process of the algorithm is described in detail as follows. At every timestep, the freestream wind conditions ϕ_t and v_t are updated according to the wind process. By default this is the same random walk wind process used for the Floris environments. The updated freestream wind conditions are used to compute the steady-state wake of the wind farm through a Floris simulation step. Then a set of observation points $\mathbf{OP}_t$ are created for each wind turbine. These observation points carry information about their location in the wind farm, the timestep of their creation and the wake effect they represent. Each observation point stores part of the current steady-state wake effect as a list of steady-state wake values at increasing distances from the turbine, in the direction of the freestream wind. These values are stored so that over time the observation point can represent farther and farther sections of the stored wake. Figure 3 displays the process of generating new observation points, highlighting their location and the information they store and represent.

Algorithm 1 Observation points and flow field computation

Input: Timestep t, freestream wind direction ϕ_t and speed v_t, turbine yaw angles $\boldsymbol{\psi}_t = [\psi_{1,t}, \ldots, \psi_{N,t}]$, maximum observation point age T_{OP}, previous flow field representation $F_{t-1} = [v_{t-1}(\mathbf{p}), \phi_{t-1}(\mathbf{p}), \mathbf{OP}_{t-1}]$, with observation points $\mathbf{OP}_{t-1} = \{\text{OP}_1, \ldots, \text{OP}_{M_{t-1}}\}$, two dimensional Gaussian G and simulation parameters (e.g., simulation Δt, wind farm bounds, etc.)

Output: Updated flow field $F_t = [v_t(\mathbf{p}), \phi_t(\mathbf{p}), \mathbf{OP}_t]$

$F_{ss,t} = \mathcal{M}_{SS}(\phi_t, v_t, \boldsymbol{\psi}_t)$
$\Delta\mathbf{p} = [v_t \cos\phi_t, v_t \sin\phi_t] \cdot \Delta t$
$\mathbf{OP}_t \leftarrow \{\}$
for all $\text{OP}_i \in \mathbf{OP}_{t-1}$ **do**
  $\mathbf{p}_i, F_{\text{Wake}}, t_0 \leftarrow \text{OP}_i$
  if $t - t_0 < T_{\text{OP}}$ **then**
    $\mathbf{OP}_t \leftarrow \mathbf{OP}_t \cup \{[\mathbf{p}_i + \Delta\mathbf{p}, F_{\text{Wake}}, t_0]\}$
  end if
end for
for all wind turbines WT **do**
  $\mathbf{p}_1, \ldots, \mathbf{p}_K \leftarrow$ pointsBehind(WT) //equally spaced behind the turbine
  for k=1,...,K **do**
    $F_{\text{Wake}} \leftarrow [F_{ss,t}(\mathbf{p}_k + \Delta\mathbf{p} \cdot j)]_{j=0}^{T_{\text{OP}}}$
    $\mathbf{OP}_t \leftarrow \mathbf{OP}_t \cup \{[\mathbf{p}_k, F_{\text{Wake}}, t]\}$
  end for
end for
$\Delta v_t(\mathbf{p}), \Delta\phi_t(\mathbf{p}) = \sum_{i=1}^{M_t} G(\mathbf{p} - \mathbf{p}_{\text{OP}_i}) \cdot F_{\text{Wake,OP}_i}[t - t_{\text{OP}_i}]$
$v_t(\mathbf{p}) = v_t - \Delta v_t(\mathbf{p})$
$\phi_t(\mathbf{p}) = \phi_t - \Delta\phi_t(\mathbf{p})$
return $F_t = [v_t(\mathbf{p}), \phi_t(\mathbf{p}), \mathbf{OP}_t]$

Algorithm 2 Wind farm generated power and rewards from observation points

Input: Flow field $F_t = [v_t(\mathbf{p}), \phi_t(\mathbf{p}), \mathbf{OP}_t]$, freestream wind direction ϕ_t and wind speed v_t, turbine yaws with respect to the north $\boldsymbol{\psi}_t^{abs} = [\psi_{1,t}^{abs}, \ldots, \psi_{N,t}^{abs}]$, timestep t, reward scaling K_R.

Output: Generated power $\mathcal{P}_t$ and reward R_t.

for all wind turbines WT_i **do**
  $\mathbf{p}_1, \ldots, \mathbf{p}_K \leftarrow$ pointsAhead(WT_i)
  $\overline{v}_{i,t} = \frac{1}{K} \sum_{k=1}^{K} v_t(\mathbf{p}_k)$
  $\overline{\phi}_{i,t} = \frac{1}{K} \sum_{k=1}^{K} \phi_t(\mathbf{p}_k)$
  $\mathcal{P}_{i,t} = \mathcal{P}_{Floris}(\overline{v}_{i,t}, \overline{\phi}_{i,t}, \psi_{i,t}^{abs})$
end for
$\mathcal{P}_t = \sum_{i=1}^{N} \mathcal{P}_{i,t}$
$R_t = K_R \cdot \mathcal{P}_t$
return $\mathcal{P}_t, R_t$

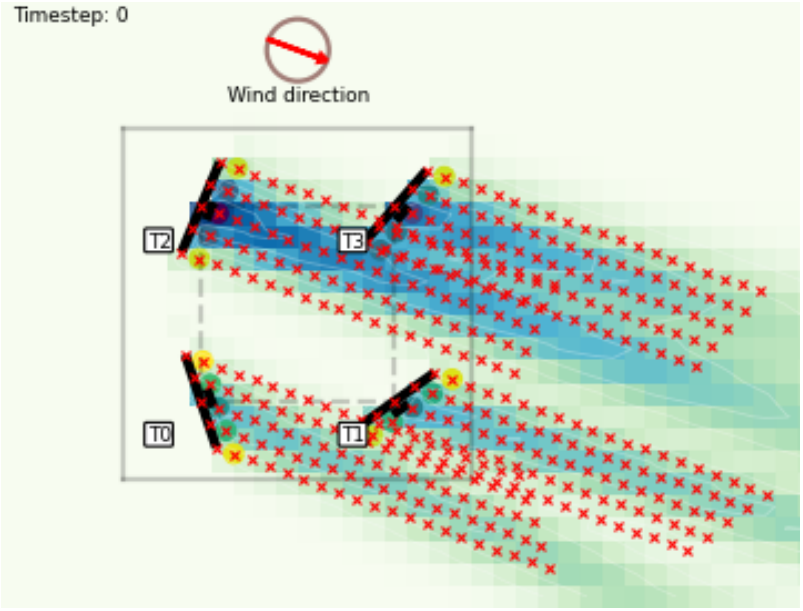

Figure 3: Rendering of the environment highlighting the observation point generation process. The red crosses indicate the points at which the steady-state wake effects are sampled and stored. Upon generation, observation points represent the steady-state wake effect behind the turbines. Over time they represent wake effects farther from their turbine of origin. In the figure this progression is visible as lines of red crosses starting from the back of each turbine. The colored circles next to the wind turbines represent the observation points. Their color indicates the magnitude of the wake effect that they represent. The wake effects of the turbines in the background are the steady-state wake effects computed with Floris.

After the generation of new observation points, all observation points are updated as follows. The position of each observation point is updated with velocity and direction corresponding to the freestream wind conditions. Note that while it would be possible to move each point with the local wind speed and direction it represents, this approach is not used in the more recent formulations FloriDyn [10]. Figure 4 shows how the observation points change over time, both in position and value of represented wake.

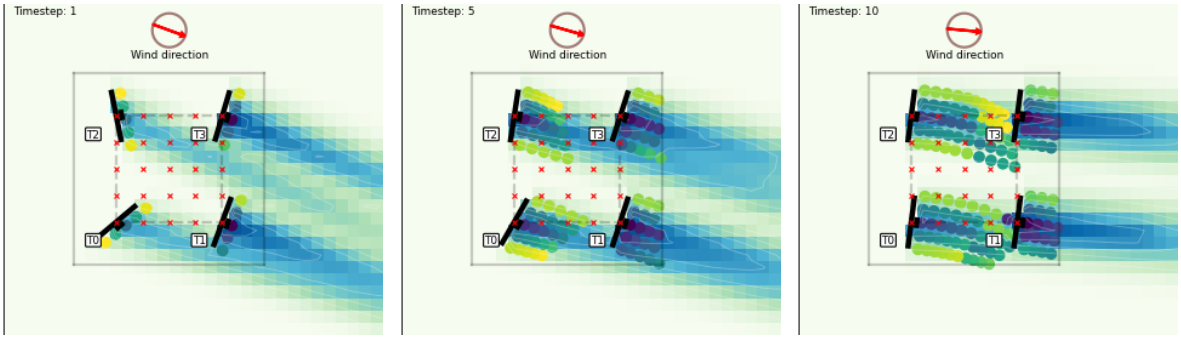

Figure 4: Rendering of the environment at three timesteps highlighting the movement of observation points over time. Left to right, pictures represent the environment at timesteps 1, 5 and 10. The observation points are updated over time both in their location and in the magnitude of the wake that they represent, shown here as the color of each observation point. The movement of the observation points over time allows steady-state information to propagate through the wind farm over time. The grey square sorrounding the farm represents the boundaries of the wind farm. Observation points crossing them are discarded.

A maximum observation point age T_{OP} is also defined as a parameter of the environment. Points that have existed for more than the age maximum are deleted before the update step. While this step hinders realism, as it limits the maximum number of timesteps during which wind information can travel, this mechanism limits the computational cost of the simulation. As observation points can only exist for a fixed number of timesteps, the maximum number of existing points is always bounded by the product of number of turbines, number of observation points generated per turbine, and maximum age of observation points. This ensures that the computational cost of the simulation is bounded.

Finally, the set of observation points is used to compute a flow field. To compute the flow field, the local information of the wake effects carried by the observation points is diffused through the flow field using a Gaussian centered at the location of the observation points. Then, the wind speed and direction offsets of the wake effects are added to the freeflow wind represented by the global wind speed and direction. This process allows to compute the flow field at arbitrary locations in the wind farm. An example of a dynamic flow field computed in this way at different timesteps is shown in Figure 5. The power generated by each wind turbine is computed through Floris by providing an average estimate of the wind speed perpendicular to the wind turbine. In practice, to reduce computational costs, the flow field is computed and stored as values at grid locations through a matrix representation. The spatial resolution of the matrix can be tweaked to create more coarse representations that are faster to compute. This coarser resolution decreases computational requirements for larger wind farms, allowing the model to scale more efficiently with increasing farm size.

MDP formulation

The wind farm optimal control problem is formulated as a POMDP which is defined by the tuple $(S, A, P, R, \Omega, O, \gamma)$. The action $a_t \in A$ and discount factor γ are defined in the same way as the previous environments. The flow field computation for this environment is different from the previous environments. In this case the flow field is generated through a smoothing of the information carried by the observation points, which are transported according to the freestream wind. The flow field for this environment is represented as $F_t = [v_t(\cdot), \phi_t(\cdot), \mathbf{OP}_t]$, and its evolution is governed by $F_t = \mathcal{M}_{QD}(\phi_t, v_t, \boldsymbol{\psi}_t, F_{t-1})$. The operations represented by this function are described in Algorithm 1.

Given the current values of observation points $\mathbf{OP}_t = \{\mathrm{OP}_1, \ldots, \mathrm{OP}_{M_t}\}$, freestream wind ϕ_t, v_t

and turbine yaws $\boldsymbol{\psi}_t$, the environment is fully defined. Thus, the state of the environment $s_t \in S$ can be expressed as $s_t = [\phi_t, v_t, \boldsymbol{\psi}_t, \mathbf{OP}_t]$. While this state fully represents the environment at each timestep, its information is not available through sensors which are usually available in real-life wind farms. Because of this, the observations of the environment o_t, belonging to the set of all possible freestream wind conditions and yaw configurations $o_t \in \Omega$, are modeled through a conditional observation probability function $O(o_t|s_t, a_t)$. This function is deterministic in the case of this environment and maps each state to the observation vector $o_t = [\phi_t, v_t, \boldsymbol{\psi}_t]$. Given the aforementioned definitions of state and action, the transition probability function $P(s_{t+1}|s_t, a_t)$ represents the evolution of freeflow wind conditions and observation points. The turbine actuation and random walk wind process is defined in the same way as in the previous environments. The evolution of the set of observation points $\mathbf{OP}_t$, which generates new points and updates old ones, is described in Algorithm 1. The reward function $R(s_t, a_t)$ is computed from the flow field F_t as described in Algorithm 2 and represents a scaling of the total power generated in the wind farm at each timestep.
This work also explores the impact of privileged information on agent performance in this environment. Thus, a privileged set of observations $\tilde{o}_t \in \tilde{\Omega}$ is defined. It contains the information of observations o_t and expands it with information about the flow field at a set of locations. Privileged information is thus described as $\tilde{o}_t = [\phi_t, v_t, \boldsymbol{\psi}_t, v_t(\tilde{\mathbf{p}}_1), \phi(\tilde{\mathbf{p}}_1), \ldots, v_t(\tilde{\mathbf{p}}_P), \phi(\tilde{\mathbf{p}}_P)]$ with $\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_P$ being the points at which flow field information is observed. The conditional privileged observation probability function $\tilde{O}(\tilde{o}_t|s_t, a_t)$ maps each state s_t to privileged observations $\tilde{o}_t$. Information about the flow field at points $\mathbf{p}$ is obtained from the flow field representation F_t.

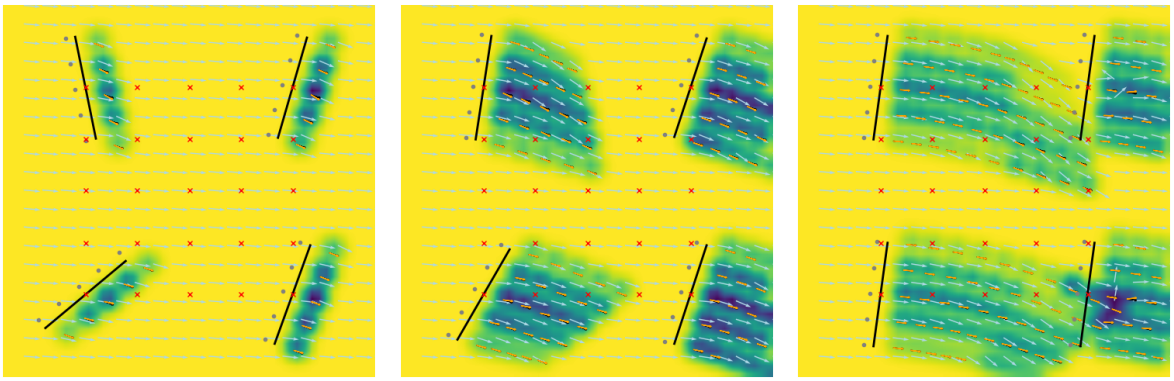

Figure 5: Three representations of the dynamic flow field computed from observation points. From left to right the flow field is computed at timesteps 1, 5 and 10. The colored background shows the wind speed at every point in the wind farm, with darker colours rerpesenting slower wind speeds. Orange arrows represent the wind direction offset stored at each observation point, and the blue arrows represent the total wind direction computed at each position. Red crosses are examples of locations at which the flow field can be sampled to be provided as input to control policies and agents.

The dynamic environment offers two key advantages over steady-state simulators:

- **Wake effects are affected by wake transport**: The wake generated by upstream turbines affects downstream turbines with a delay in time. This is implemented by the movement of observation points throughout the wind farm.

- **Agents are unable to observe the full state of the system**: The state of the flow field at a given timestep depends on the set of observation points at the previous timestep. As these observation points move throughout the wind farm, and real wind farm usually feature sensors which are sparse and usually positioned at the wind turbines, realistic agents are unable to observe the full state of the system.

Both of these properties of the simulation are present in real-life wind farms, where the impact of wake effects on downstream turbines is delayed, and the information available to the wind turbine controller is limited. This allows the simulator to be more similar to real wind farms, possibly aiding in the transfer from simulated controllers to real ones.

3.2 Algorithms

This section presents the two algorithms used for wind farm control in this study. Section 3.2.1 introduces Proximal Policy Optimization, a reinforcement learning method, and outlines the training process used in this study. Section 3.2.2 introduces Floris Serial-Refine, a strong baseline for wind farm control in steady-state conditions.

3.2.1 Proximal Policy Optimization

The reinforcement learning algorithm used for the experiments is Proximal Policy Optimization (PPO). PPO is an Actor-Critic algorithm, meaning that during training it uses two networks to train a policy and a value function. The policy function is a parametric function $\pi_\theta(a|s)$ and is used by the agent to choose what action to do in the environment. The value function is a parametric function $V_\omega(s)$ and is used to estimate the value of observed states. PPO optimizes a policy by minimizing a clipped surrogate objective, which prevents excessively large policy updates to achieve more stable learning.

The objective function for the policy is:

$$L^{\mathsf{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \mathsf{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathsf{old}}}(a_t|s_t)}$ is the probability ratio of the current policy π_θ to the old policy $\pi_{\theta_{\mathsf{old}}}$.
- $\hat{A}_t$ is the advantage estimate at time t, calculated using the Generalized Advantage Estimation (GAE) method [32].
- ϵ is a hyperparameter that controls the clipping range to limit the magnitude of policy updates.

The value function $V_\omega(s)$ is optimized to minimize the mean squared error (MSE) between its predictions and a target:

$$L^{\mathsf{VF}}(\omega) = \mathbb{E}_t \left[\left(V_\omega(s_t) - V_{\mathsf{target}}(s_t) \right)^2 \right]$$

where $V_{\mathsf{target}}(s_t)$ is computed using GAE and the old value function $V_{\mathsf{target}}(s_t) = \hat{A}_t + V_{\omega_{\mathsf{old}}}(s_t)$.

The total loss function combines the clipped surrogate loss, value loss, and an entropy bonus to encourage exploration:

$$L(\theta, \phi) = L^{\mathsf{CLIP}}(\theta) - c_1 L^{\mathsf{VF}}(\omega) + c_2 H[\pi_\theta],$$

where c_1 and c_2 are coefficients, and $H[\pi_\theta]$ is the entropy of the policy.

The PPO implementation that is used is provided by the Stable-Baselines3 library [33]. The training process for PPO agents is as follows: each agent is trained on an environment for 0.1M timesteps. During training, each agent is periodically evaluated environments on a set of fixed seeds. Then, the model weights that achieve the best validation performance during training are evaluated again, on a set of test seeds for the environment. The average total reward on the test seeds is the performance score of the agent. This metric is used to compare the model to other methods.
In practice, to ensure that all models are evaluated fairly and to minimize the effects of stochasticity, two sets of 10 wind processes are created at random and saved. One set corresponds to the set of validation seeds and the other corresponds to the set of testing seeds. During training, each agent is evaluated periodically on the validation wind processes, and their average total reward is recorded. The model weights that achieve the highest average total reward on the validation wind processes are saved and tested on the test wind processes. The average total reward and average total power of these model weights are then saved and used to compare the performance to the model to other baselines. Note that all performance metrics that are used for comparison are run on the preset evaluation wind processes, to ensure that the comparisons are not affected by the stochasticity of the wind processes. PPO agents are evaluated using their policy in deterministic mode.
The hyperparameters used for each PPO model are found through random search. Details on the hyperparameter search and training process are provided in Appendix A.

3.2.2 Floris Serial-Refine

Serial-Refine is a method to optimize the yaw angles of turbines for wake steering in a steady-state context. Given a fixed wind farm layout, this method learns a map from freestream wind direction to turbine yaw offsests which maximise steady-state power generation:

$$\mathbf{\Psi}_{SR}(\phi) = \arg\max_{\boldsymbol{\psi}} \sum_{i=0}^{N} P_i(F_{ss}(\phi, v, \boldsymbol{\psi}), \psi_i)$$

where F_{ss} represents the steady-state flow computed through Floris $F_{ss} = \mathcal{M}_{SS}(\phi, v, \boldsymbol{\psi})$, and wind speed v is kept constant. Note that there is no dependence to time t, as this method works under steady-state assumptions, and there is no wind process involved, as freestream wind conditions are sampled by the algorithm.
To find the optimal set of turbine yaws, the algorithm works in two distinct phases: the Serial pass and the Refine pass. The turbines are first ordered from upstream to downstream, and then the algorithm proceeds as follows:

- **Serial pass**: In the Serial pass, the algorithm iterates through the wind farm turbines from upstream to downstream, and for each one it evaluates a fixed number of different yaw angles. For each turbine the algorithm selects the yaw angle that maximizes the total wind farm power production and stores it for subsequent power evaluations of downstream turbines.

- **Refine pass**: After the initial Serial pass, the algorithm moves to the Refine pass. This second pass refines the yaw angles by evaluating different angles, but now the angles are evaluated in a smaller offset range centered around the previously chosen yaw angle from the Serial pass.

By performing these two passes, the Serial-Refine method finds the optimal yaw angles efficiently while avoiding the exhaustive search over all possible yaw angles that would be required in more computationally expensive methods. The result is an approach that balances computational efficiency with the ability to optimize wind farm power output. The algorithm is outlined in Algorithm 3.
In our study, we use Serial-Refine as a benchmark to compare the effectiveness of other methods, particularly focusing on the quality of solutions it produces, which have been shown to be comparable in performance to more expensive and exhaustive numerical optimization methods. As this method yields a set of optimal yaws, and not a control policy, the yaws computed by the method have been used in conjunction with a proportional controller. This has multiple advantages. Firstly, using a proportional controller allows this method to be compared to other control policies. The set of yaws found by Serial-Refine optimizes the total power generation of the wind farm. In the case of wind farms simulated with Floris, this generated power corresponds to the power generated at the steady-state. To compare policies, such as those implemented by RL agents, to this metric, these agents would have to be ran for multiple timesteps, until the control policy converges to a set of turbine yaws, and then the power generated at the final step would constitute a metric of the power generated at the steady-state. While this metric can be used to compare a RL agent to Serial-Refine, it has significant drawbacks. This metric is not affected by the trajectory taken to achieve the final state and effectively ignores the actual control for all timesteps except the last. Because of this, policies that traverse suboptimal intermediate states in these trajectories can yield the same performance metric as policies that implement the optimal trajectory and yield the maximum total generated power.
Moreover, while the optimal yaws can be used to be obtain performance metrics for the case of fixed wind speed and directions, they are unable to be used in the case of environments where the wind conditions change over time. Setting the turbine yaws to the optimal ones at each timestep would be possible, but it ignores the control constraints present in real-life scenarios, where the actuation of the rotation of the wind turbines to achieve a given yaw setting is limited in speed and range. A proportional controller based on the optimal Serial-Refine yaws is able to control the yaws of the turbines of the wind farm in the case of changing wind conditions, allowing the method to be evaluated in a broader set of environments.

3.3 Environment layouts

Three wind farm layouts are defined for this study. The first layout, referred to as the 4 Symmetric layout, consists of four turbines positioned at the vertices of a 250x250m square. The second and third layouts, named the 8 LHS layout and 16 LHS layout, are generated using the Latin Hypercube Sampling (LHS) method to place wind turbines apart from one another. In the 8 Turbines LHS layout, the turbines are distributed within a 750x750 meter area, while the 16 Turbines LHS layout covers a larger 1500x1500 meter area. The LHS method ensures that turbines are distributed randomly along the x and y axes while maintaining adequate spacing between them. This approach achieves a balance between randomness and realism,

Algorithm 3 Floris Serial-Refine Algorithm for Wind Farm Yaw Optimization

Input: Turbine layout, set of Q freestream wind direction discretizations $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_Q]$, wind speed v, coarse yaw offsets δ_{coarse} (default {-30°, -15°, 0°, 15°, 30°}), fine yaw offsets δ_{fine} (default {-7.5°, -3.75°, 0°, 3.75°, 7.5°}).

Output: Approximate optimal yaw offsets for each wind direction discretization: $\boldsymbol{\Psi}_{SR} = [\boldsymbol{\psi}_{\phi_1}, \ldots, \boldsymbol{\psi}_{\phi_Q}]$

$\boldsymbol{\Psi}_{SR} \leftarrow [\mathbf{0}, \ldots, \mathbf{0}]$
for each wind direction discretization $\phi_i \in \boldsymbol{\phi}$ **do**
  $\boldsymbol{\psi}_{opt} \leftarrow \mathbf{0}$
  $P_{opt} = P(\phi, v, \boldsymbol{\psi})$
  {Serial Pass - Coarse Search}
  for each turbine j, sorted upstream to downstream **do**
    for $\delta \in \delta_{coarse}$ **do**
      $\boldsymbol{\psi}_{test} \leftarrow \boldsymbol{\psi}_{opt}$
      $\psi_{test,j} \leftarrow \delta$
      $P_{test} \leftarrow P(\phi, v, \boldsymbol{\psi}_{test})$
      if $P_{test} > P_{opt}$ **then**
        $P_{opt} \leftarrow P_{test}$
        $\boldsymbol{\psi}_{opt} \leftarrow \boldsymbol{\psi}_{test}$
      end if
    end for
  end for
  {Refine Pass - Fine Search}
  for each turbine i, sorted upstream to downstream **do**
    for $\delta \in \delta_{fine}$ **do**
      $\boldsymbol{\psi}_{test} \leftarrow \boldsymbol{\psi}_{opt}$
      $\psi_{test,i} \leftarrow \delta$
      $P_{test} \leftarrow P(\phi, v, \boldsymbol{\psi}_{test})$
      if $P_{test} > P_{opt}$ **then**
        $P_{opt} \leftarrow P_{test}$
        $\boldsymbol{\psi}_{opt} \leftarrow \boldsymbol{\psi}_{test}$
      end if
    end for
  end for
  $\boldsymbol{\Psi}_{SR}[i] = \boldsymbol{\psi}_{\phi_i} \leftarrow \boldsymbol{\psi}_{opt}$
end for

return $\boldsymbol{\Psi}_{SR}$

reflecting the design of real wind farms where turbines are spaced to optimize wind flow and minimize the impact of wake effects. Renderings of the three layouts are shown in Figure 6.

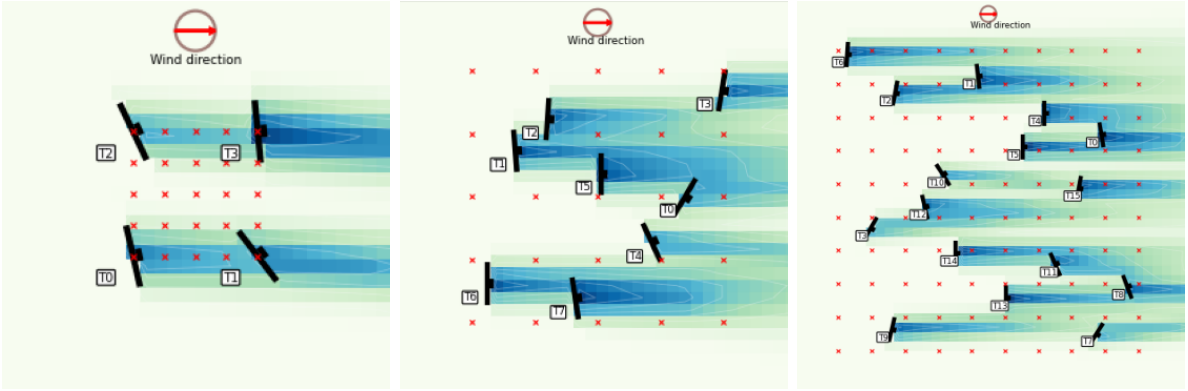

Figure 6: Rendering of the three environment layouts with randomized initial turbine yaws. From left to right, 4 Symmetric, 8 LHS and 16 LHS wind turbine layouts. The red crosses represent possible grids for the localization of privileged information sensing.

3.4 Experimental design

This section describes the experiments that are run in this study. For the first set of experiments, the Floris Serial-Refine algorithm is run on all environment layouts to replicate the experiments from the original paper and evaluate its performance and compare it to other solvers. Next, we conduct experiments to compare the performance of PPO with the Floris Serial-Refine proportional controller across the three environment types: fixed wind, changing wind, and quasi-dynamic wind conditions. These experiments are aimed at showing how the environment properties affect the applicability of each algorithm.

3.4.1 Floris Serial-Refine Analysis

The first experiments consist of running the optimization algorithm of Floris Serial-Refine and comparing it to other solvers. The methods being compared find optimal yaw configurations for different freestream wind conditions in a steady-state setting and are not control policies. Experiments are run on all environment layouts and a different set of optimal wind turbine yaws is found for each. The performance of Floris Serial-Refine is compared to other numerical optimization methods, to confirm its validity as a powerful baseline on these environments. The optimization methods that Serial-Refine is compared to are the Sequential Least Squares Programming (SLSQP) optimizer from the Scipy library [34] and a geometric method [35] which finds approximately optimal yaw angles based on the wind farm geometry. Both the SLSQP and geometric optimizers are implemented as part of the Floris simulator, with SLSQP being the default optimizer used in Floris prior to the development of Floris Serial-Refine. The evaluation metric used to compare the different optimizers is normalized power gain. This metric is computed as the ratio between the total power generated by a given method divided by the power generated by the SLSQP optimization method. The total power is computed by summing the generated power over all wind directions. In these experiments, the 360 degree range is divided into 120 intervals, each separated by a 3 degree step.

The optimal yaws computed through the Floris Serial-Refine algorithm are then used to qualitatively evaluate their adaptability to changing wind conditions. This is done by computing the difference in adjacent optimal yaws for each turbine. This metric shows the optimal one-step action that a control policy would have to implement to be able to always traverse optimal states, for each 3-degree change in wind direction. This is of interest because if it were possible to implement such a policy, then there would be no need to optimize over trajectories, as once an optimal state is reached, single-step control policies would already achieve optimal control in the fixed wind and changing wind environments.

3.4.2 Fixed and changing wind environment

Then experiments are run on the fixed and changing wind environments which are based on the FLORIS steady-state simulations. In these experiments different control policies are compared to evaluate their effectiveness in these two environments. The wind farm layouts that are used for these experiments are the 4 Symmetric layout and the 8 LHS layout, to limit computational costs. The control policies that are compared in these experiments are the following:

- Random policy: This policy randomly samples an action from the action space at every step.
$$a_t = [\hat{\psi}_1, \dots, \hat{\psi}_N] \quad \hat{\psi}_i \sim \mathcal{U}(-\psi_{max}, \psi_{max}) \quad \forall i \in \mathcal{N}$$

- Naïve policy: This policy always acts to point the wind turbines directly to the wind. This corresponds to always aiming to have zero offset with respect to the freestream wind direction.
$$a_t = [\hat{\psi}_1, \dots, \hat{\psi}_N] = [0, \dots, 0]$$

- Proportional Serial-Refine: This policy implements a proportional controller to move the turbine yaws towards the optimal steady-state configuration computed through the Floris Serial-Refine algorithm. While this policy takes steady-state wake effects into account, as the yaws are optimized to minimize its effect on power generation, the trajectories taken by this control policy are not optimized for it. They simply traverse intermediate states in the segment between the current yaw configuration and the optimal yaws, regardless of whether they may be suboptimal for power generation or not.
$$a_t = [\hat{\psi}_1, \dots, \hat{\psi}_N] = \mathbf{\Psi}_{SR}(\phi_t)$$

- PPO: This is the policy obtained by training PPO on the given environment. This policy is run deterministically for evaluation purposes. Contrary to the Proportional Serial-Refine policy, this policy is trained to maximise the cumulative reward corresponding to total power generation. This means that it will try to optimize control considering the whole trajectory of an episode, including the traversal of suboptimal yaw configurations.
$$a_t = [\hat{\psi}_1, \dots, \hat{\psi}_N] = \pi_\theta(o_t)$$

  In the case of privileged observations, the observation taken as input for the policy is changed accordingly:
$$a_t = [\hat{\psi}_1, \dots, \hat{\psi}_N] = \pi_\theta(\tilde{o}_t)$$

For these experiments the evaluation metric is the average total power generated in an episode, and all the control policies are evaluated on the same set of randomized evaluation wind processes. For the fixed wind environment the randomized wind processes implement fixed wind conditions for 10 randomly chosen wind directions and speeds, and for the changing wind environment they consist in 10 randomly sampled wind processes of changing wind direction. For the fixed wind environment, a curve of total generated power over all possible wind directions is computed, to evaluate how the model performs across different wind conditions.

3.4.3 Quasi-dynamic environment

Finally, experiments are run on the quasi-dynamic environment. They are run 4 Symmetric layout and the 8 LHS layout. This experiment compares the average total generated power of the different control policies used in the changing wind environment. The same set of randomized wind process trajectories used in the changing wind environment is used for evaluation. As this environment is characterized by partial observability, some additional PPO control policies featuring privileged observations are evaluated. These observations include measurements of wind direction and speed at different locations in the wind farm. The measurement locations are distributed across the wind farm, arranged in a grid pattern. Coarser and finer spacings of the grid pattern are evaluated for each environment layout. For the 4 symmetric layout spacings of 25, 75 and 125 meters are evaluated, whereas for the 8LHS layout the spacings are of 100, 150 and 200 meters, due to the differences in size between them.
As the dynamic environment features a different observation space and system dynamics than the steady-state environments, the Serial-Refine optimal yaw computations should in principle be recomputed for this. Unfortunately this is not feasible due to the increased dimensionality of the problem. The Serial-Refine method is designed for the simpler steady-state formulation of the wind farm dynamics problem where the wake effects are primarily determined by the current wind direction. This allows to consider all the possible discretized states of the wind direction, and search the optimal yaw angles for each of them. The method then prdouces a lookup table with these results. However, this approach is not applicable to real-world wind farms, where the wind state depends on a significantly larger number of variables. In the steady-state case, the Serial-Refine algorithm must perform a search for each quantization of the global wind direction. In scenarios featuring wind dynamics that are not steady-state, assuming that only realistic sensors are available, the algorithm would need to run a search for optimal yaws for each combination of possible values of global wind direction, turbine yaws and initial wind conditions throughout the entire trajectory. This is because without having perfect information of the state of the flow field of the wind farm at each step, the history of all these variables is required to derive the current state of the wind farm. This is different in the steady-state case, where the current global wind conditions and turbine yaws entirely define the system state. Defining Q_{wind} and Q_{yaws} as the quantization of the representation of the global wind direction and turbine yaws, N the number of turbines and H as trajectory length, the number of searches required by the Serial-Refine method for the two type of environments can be expressed as follows:

$$\text{Steady searches} = Q_{wind}$$

$$\text{Dynamic searches} = (Q_{wind} \cdot Q_{yaws}^{N})^{H}$$

when assuming fixed starting flow field conditions. These formulas show that the number of searches and size of the lookup table generated by the Serial-Refine method scales exponentially both with the number of turbines in the wind farm and with the length of the trajectory horizon. In the case of the 8LHS layout, assuming a quantization angular metrics into 120 values and considering a short trajectory horizon of 5 timesteps, this results in a lookup table of $(120 * 120^8)^5 \approx 3.66 \times 10^{93}$ entries, each corresponding to an optimal yaw search problem, making the overall computation intractable. Because of this, experiments on the quasi-dynamic environment have to use the steady-state Serial-Refine controller as proxy, as applying the algorithm to the quasi-dynamic case is impossible without significant changes.

3.4.4 Computational efficiency comparison

A computational efficiency study is run on the quasi-dynamic environment to measure its speed and compare it to other environments. The experiment closely resembles the one ran in another work [36]. The average time to compute simulation steps is compared between SOWFA, FLORIS, and the proposed quasi-dynamic environment. Measurements of simulation speed for SOWFA that are considered are the ones reported in the earlier study [36]. Simulation speeds of FLORIS and the quasi-dynamic environment are computed and averaged over one episode for two different settings. The first features and environment layout with two turbines aligned in the direction of incoming wind, $\phi_t = 270$. The wind is fixed in direction and speed, with a constant speed of $v_t = 8m/s$ for 1000s. The second setting features 9 turbines placed in a 3x3 grid in a wind farm of 3x3km. The wind speed has constant value of $v_t = 8m/s$, and its direction follows:

$$\phi_t = \begin{cases} 255°, & 0 \leq t < 300 \text{ s} \\ 255° + \frac{(195°-255°)}{300}(t-300), & 300 \leq t < 600 \text{ s} \\ 195°, & 600 \leq t \leq 900 \text{ s} \end{cases}$$

For the quasi-dynamic environment, for both layouts the flow field wind direction and speed are computed at locations in a 150x150 grid, and the simulation timestep is $\Delta t = 5s$. The time limit T_{OP} for observation points is set to unbounded for this experiment.

4 Results

This section presents the results of the experiments of this study. Section 4.1 shows the results of the Floris Serial-Refine baseline validation, reproducing results from the original paper and analyzing the optimal yaw angles it finds on the various environment layouts. Section 4.2 compares PPO to the baseline controllers in fixed wind steady-state conditions, while Section 4.3 extends this comparison to changing wind steady-state scenarios. Section 4.4 evaluates the performance of the PPO and privileged PPO control policies in the quasi-dynamic environment, comparing them to the baselines. Finally, Section 4.5 presents the results of the computational efficiency study, comparing the proposed quasi-dynamic environment to the SOWFA and FLORIS simulators using data from a previous study.

4.1 Floris Serial-Refine

The Floris Serial-Refine algorithm and two other optimization algorithms were applied to all three environment layouts to find the set of angles that optimize the power generation in a fixed wind steady-state scenario. The results normalized with respect to the most computationally expensive method, SLSQP, are shown in Figure 7. In all cases, Floris-Serial refine is able to achieve the best performance. In the simpler environment with the 4 Symmetric layout both methods display the same performance. In the more complex environments with 8 and 16 turbines, the Serial-Refine algorithm outperforms SLSQP despite its lower computational cost. The geometric algorithm performs significantly worse than the rest in all cases. These results show that Serial-Refine is a suitable baseline to evaluate against in fixed wind, steady-state conditions. The results also match with the findings of the original paper, where layouts with more turbines showcase a higher power gain for Serial-Refine against SLSQP.

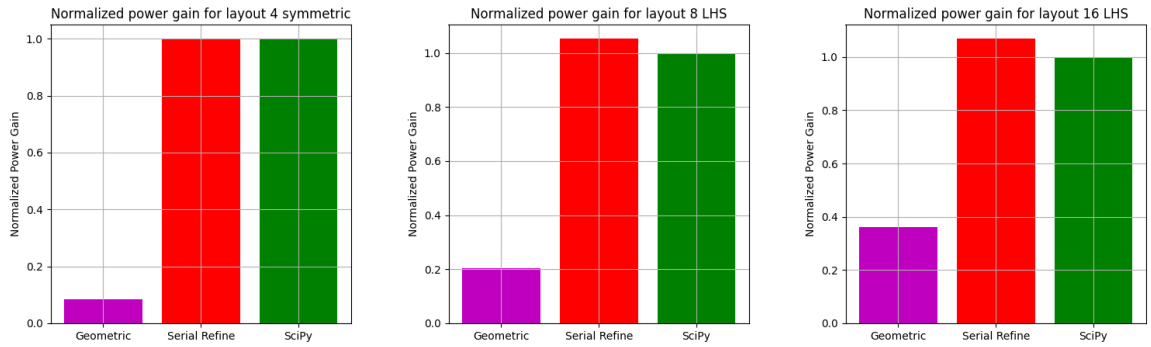

Figure 7: Normalized power gain plots for Floris Serial-Refine, SLSQP and the geometric algorithm. The normalized power gain is the ratio of the power gain of a method and the SLSQP method with respect to the naïve policy.

Figure 8 shows the results regarding how the optimal angles of the turbine yaws change with respect to the global wind direction. The first two plots show that when transitioning between adjacent global wind directions the optimal turbine yaws can change significantly. For the 4 Symmetric and 8 LHS environment layouts, the optimal yaw change for some turbines can reach as much as 17.19 and 25 degrees. This means that in the case that the global wind direction changes by 3 degrees, the angle granularity of the plots, the wind turbines may have to rotate by those angles in a single timestep in the worst case.
This shows that an optimal controller may not always traverse optimal states in a changing wind scenario where steady-state simulations compute the wind flow if the turbines are unable to rotate sufficiently quickly. The plot on the right in Figure 9 shows how the percentage of global wind directions where it is possible to avoid suboptimal yaw configurations changes depending on the turning speed constraints of the wind turbines. It shows that in environments with more turbines it is more common to traverse suboptimal states, but the effect is mitigated in turbines with faster actuation.

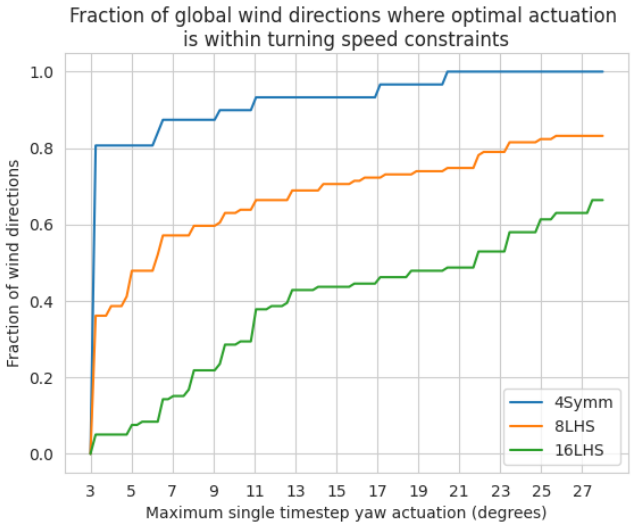

Figure 9: Fraction of wind directions where optimal actuation is possible as a function of single timestep turning constraints of the turbines. All turbine layouts are considered.

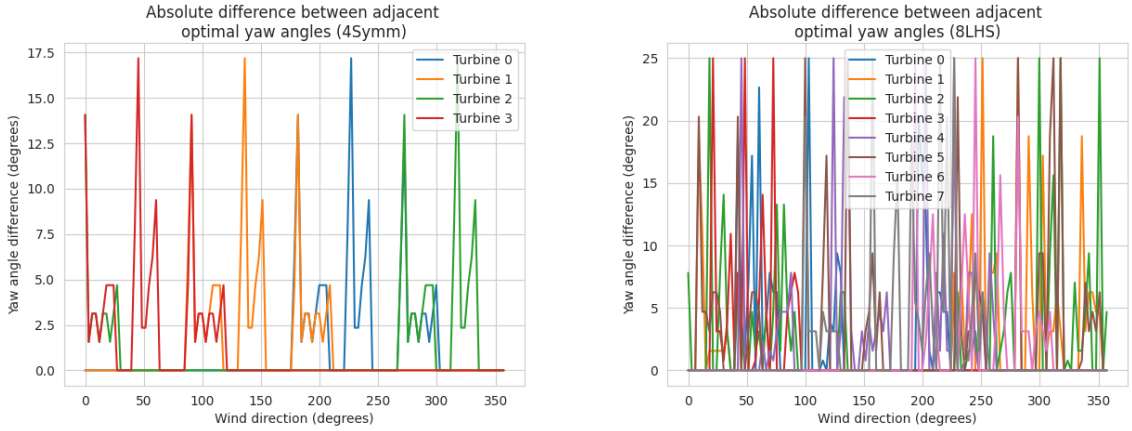

Figure 8: Absolute angle difference between optimal yaw configurations corresponding to adjacent global wind directions for the 4 symmetric and 8 LHS layouts as found by the Serial-Refine algorithm. Adjacent global angles are 3 degrees apart from one another.

4.2 Fixed wind environment

The results of the experiments of the fixed wind environment on all wind turbine layouts are shown in Figure 10. Results show that in all environments the best performing method to maximise power generation in this context is the Proportional Serial-Refine controller. This result is expected as the method extensively searches good wind turbine configurations and leverages them to implement a control policy. While potentially limited in identifying optimal controls from suboptimal initial yaw positions, the method's performance remains robust when considering the full trajectories of episodes. The PPO models perform similarly to the naïve policy, producing slightly more power. The random policy performs the worst out of all the approaches considered, as expected. These results are consistent across all environment layouts. Over a one-year period, the Proportional Serial-Refine and PPO controllers would generate an increase in energy production of up to 4,072 MWh (+1.47%) and 6.58 MWh (+0.002%),

respectively, in the 16-turbine wind farm layout, based on the results. While both control policies yield an increased power output, these gains remain minor relative to the baseline output.

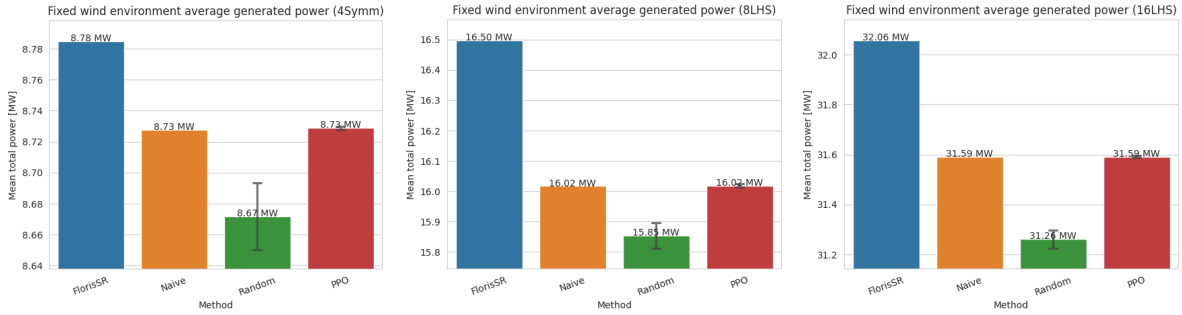

Figure 10: Performance of the Proportional Serial-Refine policy, random policy, and PPO policy on the fixed wind environment. Plots left to right concern the 4 Symmetric, 8 LHS and 16 LHS environment layouts. Results are the average power increase over the naïve policy of 10 evaluation runs with the same seed. Standard deviation is plotted for the random and PPO policies to show the variability of performance, as the random policy is stochastic, and PPO agents vary in performance across different training runs.

Figure 11 shows the power generation for different wind directions. It shows that the Proportional Serial-Refine policy is able to consistently outperform all others. This effect is especially clear in the LHS environments with more wind turbines. These results also show that while the naïve and PPO policies perform similarly on average, the power they generate for different wind directions is not always the same. These effects could be attributed to the training process of PPO, in which the wind conditions are randomized for each episode. The change in performance across different angles with respect to the naïve policy can be caused by random fluctuations in the wind directions that are encountered during training.

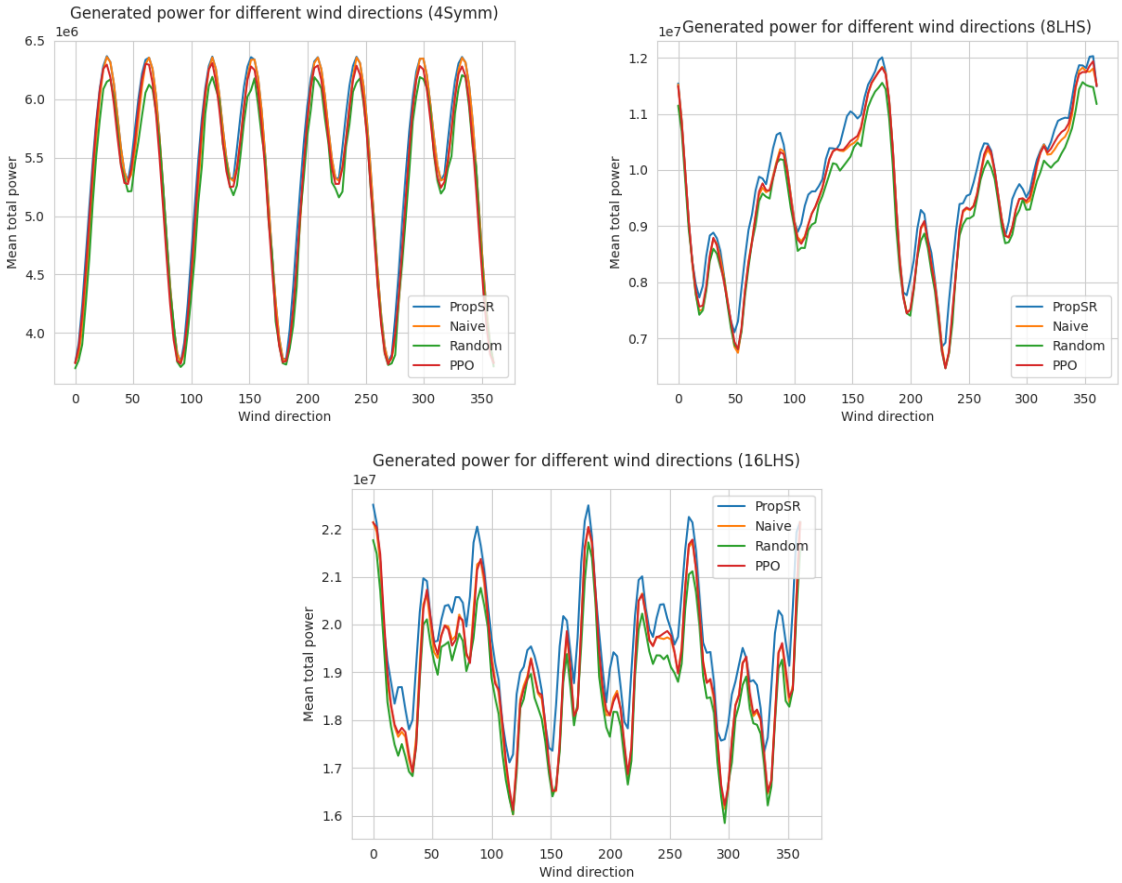

Figure 11: Power generation curve of the different control policies over different global wind directions. Plots left to right, top to bottom correspond to the 4 Symmetric, 8 LHS and 16 LHS environment layouts respectively.

4.3 Changing wind environment

The results of the experiments on the changing wind environment on all wind turbine layouts are shown in Figure 12. Similarly to the fixed wind environment results, the performance of the PPO policies and the naïve policy are similar. Moreover, the proportional controller based on the optimal yaws found through Serial-Refine is still the best performing agent. The random policy performs the worst in all cases. Over a one-year period, the Proportional Serial-Refine and PPO controllers would generate an increase in energy production of up to 5,575 MWh (+2.02%) and 46 MWh (+0.017%), respectively, in the 16-turbine wind farm layout, based on the results. As in the fixed wind environment, the gains remain minor relative to the baseline output. The similarity in the results with the previous experiment suggests that changing wind conditions do not affect the complexity of the environment in the studied cases.

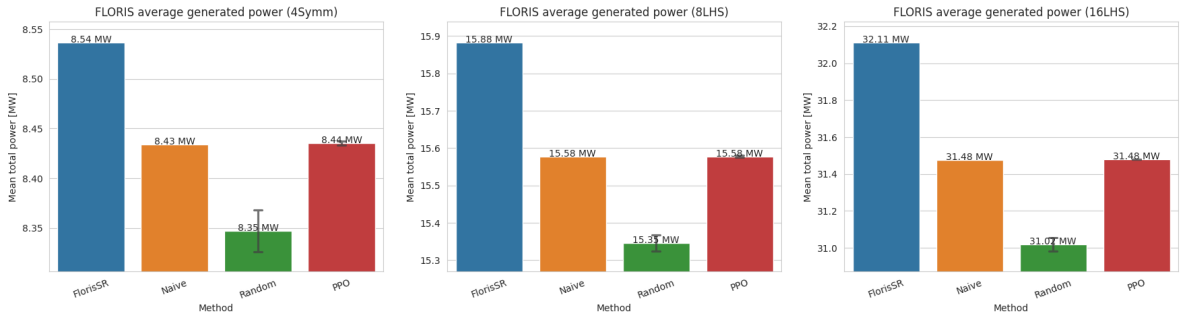

Figure 12: Performance of the Proportional Serial-Refine policy, random policy, and PPO policy on the changing wind environment. Plots left to right correspond to the 4 Symmetric, 8 LHS and 16 LHS environment layouts. Results are the average power generated by each method. Standard deviation is plotted for the random and PPO policies to show the variability of performance, as the random policy is stochastic, and PPO agents vary in performance across different training runs.

4.4 Quasi-dynamic environment

The results of the experiments on the quasi-dynamic environment on the 4 Symmetric and 8 LHS wind turbine layouts are shown in Figure 13. In this environment type, PPO is the method that achieved the best performance, closely followed by PPO with privileged information. Contrary to the steady-state environments, the performance of the PPO policies showcase a higher variability across training runs. The naïve baseline performed worse than the PPO methods, followed by the Proportional Serial-Refine and random policies. These results are in contrast with the ones of the fixed and changing wind environments, where Proportional Serial-Refine was the best approach in all cases. This is especially evident in the case of the 4 Symmetric layout, where this method performs only slightly better than the random policy. The impact on energy generation over a one-year period in the larger 8LHS wind farm layout with respect to the naïve policy is the following. The Proportional Serial-Refine controller generates less energy with a 155 MWh (-0.14%) decrease, whereas the PPO and privileged PPO policies generate 356 MWh (+0.33%) and 213 MWh (+0.20%) more energy. Similarly to the other environments, while these results demonstrate a positive impact on energy production, the effect remains modest.

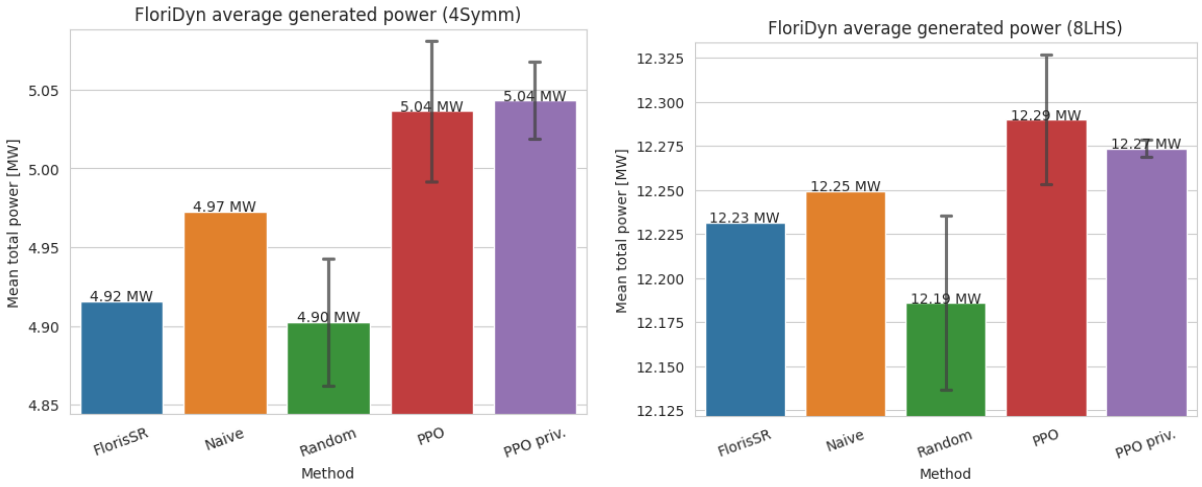

Figure 13: Performance of the Proportional Serial-Refine policy, random policy, and PPO policies on the changing wind environment. Plots left to right concern the 4 Symmetric, 8 LHS and 16 LHS environment layouts. Results are the average power increase over the random policy of 10 evaluation runs with the same seed. Standard deviation is plotted for the random, PPO and privileged PPO policies to show the variability of performance, as the random policy is stochastic, and PPO agents vary in performance across different training runs.

The performance gap between the PPO agents and the other approaches suggests that a learned control policy may be better suited to the quasi-dynamic environment. Contrary to the fixed wind and changing wind environments, the wind dynamics of the quasi-dynamic environment feature wake transport. This means that the wake effects of upstream turbines requires multiple timesteps to propagate to the downstream turbines. This difference makes approaches such as the Proportional Serial-Refine controller unsuitable for this task. This is because this controller leverages the optimal steady-state turbine yaws configuration to implement a control policy. While this works well in steady-state environments, the optimal yaw configuration of the steady-state cannot be used in a quasi-dynamic wind case, where the global wind conditions affect different parts of the wind farm with different delays.
It is also interesting to note that the privileged PPO model performs better than the PPO model in the 4 symmetric layout, but the opposite is observed in the 8LHS layout. In principle, as the environment features limited observability, providing the agent with more information should allow it to perform better, as it would be able to better infer the true state. The inferior performance in the 8LHS case could be due to multiple reasons. For example, an increased observation space can increase the sampling complexity of the problem. This may affect the convergence of the models. Appendix B shows learning curves of the PPO agents during hyperparameter search to illustrate this. Another possible reason for the performance degradation with respect to PPO is the choice of privileged information. While the points at which the privileged information is sampled were equally spaced to ensure a full coverage of the wind farm, these points do not depict the entire wind flow of the farm. The different layouts of the turbines may cause the points at which data is collected to be less valuable in the case of the 8LHS layout, as opposed to the 4 Symmetric one.

4.5 Computational efficiency comparison

Figures 14 and 15 shows the rendering of simulations created with the quasi-dynamic environment on the two considered environment layouts at different timesteps on the 2 turbines and 3x3 turbines layouts respectively. Table 1 reports the average computation time of each timestep for the different layouts and simulators. These results show that the quasi-dynamic computations slow down the simulation speed by one order of magnitude. Nevertheless, the proposed quasi-dynamic environment generates simulations more than 65,000 times faster than SOWFA. This shows that the quasi-dynamic environment is a fast alternative when a high number of non steady-state trajectories are needed.

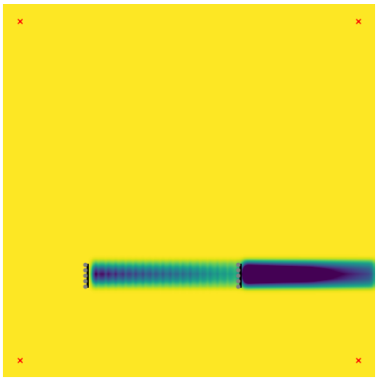

Figure 14: Flow field generated in the quasi-dynamic environment for the two turbines layout with fixed wind direction and speed.

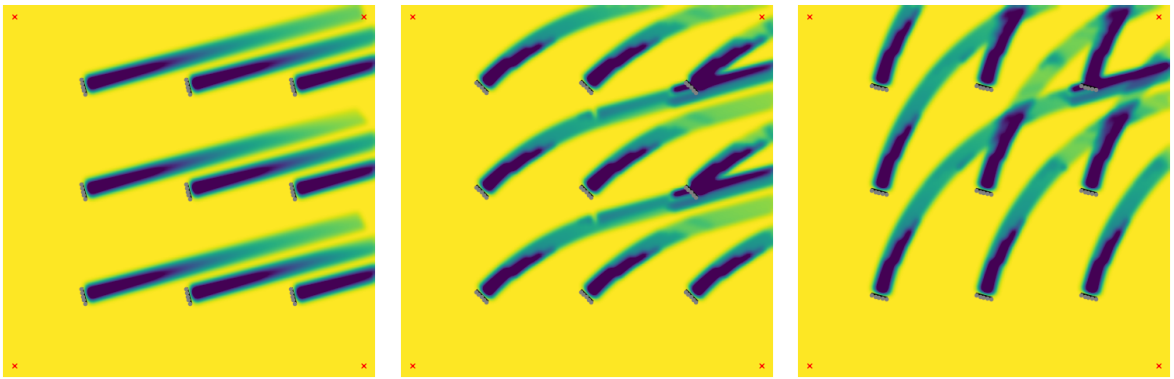

Figure 15: Flow field generated in the quasi-dynamic environment of the 3x3 wind turbine layout with changing wind direction. From left to right renders correspond to times 300s, 450s and 600s.

Wind	Turbines	Farm Size	SOWFA*	FLORIS*	FLORIS Gym	Quasi-Dynamic
Fixed	2	2×2km	$0.29 \cdot 10^4$	0.15	$0.34 \cdot 10^{-2}$	0.07
Changing	3×3	3×3km	$0.27 \cdot 10^5$	0.4	0.012	0.41

Table 1: Computation speed of each timestep for different wind conditions and farm layouts across simulators. Computation speed is represented as the average ratio of the time to compute a timestep and the simulation time that is computed in that interval. FLORIS and FLORIS Gym compute steady-state simulations while SOWFA computes dynamic simulations. The Quasi-Dynamic column refers to the environment introduced in this study. Columns indicated with an asterisk show the values reported a previous study [36].

5 Discussion

This study introduces a novel quasi-dynamic environment for wind farm simulation focused on the training of reinforcement learning agents. This environment solves some limitations of fast steady-state environments by implementing quasi-dynamic wind behaviour, featuring wake transport and partial observability, two important properties present in the real-life wind farm control. The environment dynamics are based on the ones described in the FloriDyn environment, which leverages fast-to-compute, steady-state simulations to implement a wind farm simulator with the aforementioned properties. This environment and steady-state Floris environments are then used in this study to compare the performance of PPO agents with other wind farm control baselines and evaluate their applicability to the different simulators.

5.1 Contributions of the study to the literature

This work extends previous efforts in developing fast quasi-dynamic wind farm simulators [10, 11]. It presents a novel quasi-dynamic wind farm environment based on the FloriDyn wind dynamics formulation, and designed to be compatible with RL frameworks by following the Gym paradigm. This aims to facilitate the use of the environment for reinforcement learning research, where previous environments were limited either by the simplicity of dynamics or by the implementation choices that made them difficult to integrate with modern machine learning and reinforcement learning frameworks. Moreover, the comparison with baselines and reinforcement learning agents highlights the need for different approaches in quasi-dynamic environments, where the assumptions of the simpler steady-state case no longer hold. Specifically, strong baselines for steady-state environments such as the Proportional Serial-Refine controller do not perform as well in the more complex quasi-dynamic environment, whereas the PPO agents are capable of learning policies which outperform all the baselines in these more complex environments.

This finding is in line with expectations, as the Serial-Refine method is specifically designed for smaller input spaces. In contrast, RL agents learn a fixed-size parametric function that maps the large input space to actions by identifying and leveraging patterns in the system dynamics. Moreover, RL models can take real-valued observations as input and thus can represent input data more efficiently than the Proportional Serial-Refine controller, which relies on the discretization of continuous values to build a lookup table. Because of these reasons, RL models are a more natural choice of agents for the higher-dimensional problem of

control in the case of the quasi-dynamic environment than the Serial-Refine method, which cannot be adapted to it. The results of this study align with these considerations, suggesting that RL may be a more suitable approach to solve these problems. Additionally, RL approaches can leverage existing research on POMDP problems and physics-informed neural networks to further enhance performance.
In summary, the findings of the study are the following. This study has shown that PPO can be an effective choice of agent for wind farm control in dynamic environments, where other approaches may fail. Moreover, privileged information has been leveraged to learn stronger policies in some cases. These results can help guide researchers working on realistic wind farm control problems. The findings of this study can help develop better control strategies for wind farms, leading to more green power generation in both new and existing farms. The novel quasi-dynamic environment will allow researchers and practitioners to develop control algorithms for complex wind farm environments more easily, as it features wake transport and partial observability, while leveraging the low computational cost of steady-state simulators, and can be integrated with existing RL frameworks.

5.2 Limitations

Despite its contributions, this study has several limitations. Firstly, the quasi-dynamic environment simulation is not based on a physically derived formulation of wind dynamics, unlike simulators which are based on computational fluid dynamics (CFD) [7, 8] or steady-state simulators [9]. This may affect the transferability of simulation-trained RL models to real wind farms. Instead, it is an implementation based on physical models that balances fidelity and computational efficiency while preserving important properties of real wind farm dynamics. A validation study would be useful to assess the extent of this trade-off, particularly in terms of its impact on the transfer of RL policies trained on the environment to real wind farms. Secondly, the study focuses on a limited set of simple RL agents. Expanding the investigation to include a broader range of models, including models tailored to POMDPs like recurrent PPO [37], could be beneficial.
Another limitation of this study concerns the assumptions regarding the wind farm control environment, which are based on the parameters defined in the FLORIS Gym environment. In this environment, wind turbines are actuated at each timestep and are only limited in the maximum rotation per timestep. However, real-world wind farms may have stricter control constraints, such as permitting yaw adjustments only at fixed intervals rather than continuously. This discrepancy could be significant, as in the case of infrequent actuation the ability to optimize for dynamic wind conditions may be limited. The transient wind dynamics may subside in the interval between actuations, reducing the effectiveness of control strategies.

5.3 Future work

This work also provides several possible avenues for future research. The simulator could be improved in multiple ways. Making the underlying code faster by porting it to JAX [38] for GPU support, would allow for more extensive training, although porting the FLORIS simulation to JAX may not be trivial. Another option is to develop a neural simulator that mimics the trajectories of high-fidelity simulators like SOWFA. While this would also enable GPU support, creating such a simulator would require significant computational resources.

This study has evaluated PPO policies and compared them to baselines. This has shown the potential of RL policies in this task, but the policies that were evaluated are simple and do not fully address the partial observability of the optimal control problem. Exploring the performance of models specific to POMDPs could allow to more accurately assess the potential of RL policies for wind farm control. A natural extension to this study would be to evaluate recurrent PPO[37]. Evaluating further POMDP models might be fruitful in finding effective controllers for this task.
The study on partial observability has mainly looked at how privileged information affects performance. However, since this information isn't available in real wind farms, it would be important to study how to transfer knowledge from policies trained with privileged information to policies that work with limited information. Several approaches could be suitable for this, such as Scaffolder [39] or Informed Dreamer [40]. Moreover, privileged data used in this study mainly focuses on wind flow information sampled in a grid pattern. Future works could explore different ways to provide partial observability data to agents, such as finding effective patterns to sample wind information at. Another option would be to leverage convolutional neural networks to processes the full environment state. This could help agents learn which areas of information are most important for control decisions.
Finally, physics-informed models could be effective in learning control policies [41]. These methods could help agents learn the underlying physics and then use this knowledge to develop effective policies that can be adapted to real-world scenarios with limited information. This direction is particularly important, since wind farms have very limited observability, and understanding the underlying physics of the system could help fill these information gaps.

6 Conclusion

This work addresses the limitations of traditional low-fidelity steady-state wind farm simulators for power optimization, such as FLORIS, which implement simplified wind farm dynamics in favour of computational speed. A novel quasi-dynamic wind farm simulator environment is introduced, based on the FloriDyn formulation and tailored to reinforcement learning. This environment leverages the low computational costs of computing steady-state simulations to produce wind farm environments that feature complex wind dynamics. The environment features wake transport dynamics and partial observability, both properties of real wind farms that must be taken into account in the design of a wind farm controller. It is compatible with popular reinforcement learning frameworks, making it a useful tool for researching RL-based wind farm control, contributing to both reinforcement learning research and to the production of sustainable wind energy.
Through experiments, this study shows that strong lookup table-based methods like Floris Serial-Refine are capable of effectively addressing the task of optimizing power generation in the simpler steady-state environments, but fail when addressing the novel quasi-dynamic environment. In contrast, results indicate that RL agents are capable of learning policies in the quasi-dynamic environment which outperform all considered baselines. These results suggest that reinforcement learning algorithms may be better suited for tackling the high-dimensional, partially observable problem of wind farm optimal control. Preliminary results also suggest that leveraging privileged information at training time to then be distilled into deployable models may be an effective approach to create strong control policies, but further work is required in this direction.

Despite these advancements, the study has limitations. The RL methods that are evaluated in the experiments are simple and not specific to POMDP problems. This suggests that while the results are promising, more improvements should be possible with approaches tailored to this problem. Additionally, while the quasi-dynamic environment introduces system dynamics which are present in real wind farms to simpler steady-state environments, validation is needed to assess the transferability of results to real wind farms.
Future work should focus on addressing these limitations and expanding the scope of the research. Moreover, RL methods tested in this study are not specific to POMDPs. Testing approaches specific to POMDPs could further improve the performance of RL controllers.
In conclusion, this study introduces a novel wind farm simulation environment that features important dynamic properties of real wind farms while maintaining computational efficiency, providing a valuable tool for reinforcement learning research in wind farm control. Results in the environments suggest that RL agents, in particular PPO, have the potential to outperform traditional baselines in quasi-dynamic settings, highlighting their suitability for complex control tasks. Although there are limitations, including the need for validation studies and broader agent exploration, the findings of this study can help guide future improvements in wind farm control and optimization. Future work can leverage partial observability and physics-informed models to enhance the performance of RL-based control strategies to real wind farms, leading to more efficient and sustainable energy production.

References

[1] Council of the European Union European Parliament. *amending Directive (EU) 2018/2001, Regulation (EU) 2018/1999 and Directive 98/70/EC as regards the promotion of energy from renewable sources, and repealing Council Directive (EU) 2015/652*. 2023. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32023L2413.

[2] R. J. Barthelmie et al. "Modelling and measuring flow and wind turbine wakes in large wind farms offshore". In: *Wind Energy* 12.5 (2009), pp. 431–444. DOI: https://doi.org/10.1002/we.348. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.348. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.348.

[3] Simon C. Warder and Matthew D. Piggott. "The future of offshore wind power production: Wake and climate impacts". In: *Applied Energy* 380 (2025), p. 124956. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2024.124956. URL: https://www.sciencedirect.com/science/article/pii/S0306261924023390.

[4] Ali C. Kheirabadi and Ryozo Nagamune. "A quantitative review of wind farm control with the objective of wind farm power maximization". In: *Journal of Wind Engineering and Industrial Aerodynamics* 192 (2019), pp. 45–73. ISSN: 0167-6105. DOI: https://doi.org/10.1016/j.jweia.2019.06.015. URL: https://www.sciencedirect.com/science/article/pii/S0167610519305240.

[5] Minjeong Kim, Hyeyeong Lim, and Sungsu Park. "Comparative Analysis of Wind Farm Simulators for Wind Farm Control". In: *Energies* 16.9 (2023). ISSN: 1996-1073. DOI: 10.3390/en16093676. URL: https://www.mdpi.com/1996-1073/16/9/3676.

[6] S.-P. Breton et al. "A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375.2091 (2017), p. 20160097. DOI: 10.1098/rsta.2016.0097. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2016.0097. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2016.0097.

[7] Paul Fleming et al. *SOWFA+ super controller user's manual*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2013.

[8] S. Boersma et al. "A control-oriented dynamic wind farm model: WFSim". In: *Wind Energy Science* 3.1 (2018), pp. 75–95. DOI: 10.5194/wes-3-75-2018. URL: https://wes.copernicus.org/articles/3/75/2018/.

[9] NREL. *FLORIS 3.4*. https://github.com/NREL/floris/tree/v3. 2022.

[10] M. Becker et al. "The revised FLORIDyn model: implementation of heterogeneous flow and the Gaussian wake". In: *Wind Energy Science* 7.6 (2022), pp. 2163–2179. DOI: 10.5194/wes-7-2163-2022. URL: https://wes.copernicus.org/articles/7/2163/2022/.

[11] Ali C. Kheirabadi and Ryozo Nagamune. "A low-fidelity dynamic wind farm model for simulating time-varying wind conditions and floating platform motion". In: *Ocean Engineering* 234 (2021), p. 109313. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109313. URL: https://www.sciencedirect.com/science/article/pii/S0029801821007290.

[12] Greg Brockman et al. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[13] Pieter M O Gebraad and J W van Wingerden. "A Control-Oriented Dynamic Model for Wakes in Wind Plants". In: *Journal of Physics: Conference Series* 524.1 (June 2014), p. 012186. DOI: 10.1088/1742-6596/524/1/012186. URL: https://dx.doi.org/10.1088/1742-6596/524/1/012186.

[14] P. M. O. Gebraad et al. "Wind plant power optimization through yaw control using a parametric model for wake effects—a CFD simulation study". In: *Wind Energy* 19.1 (2016), pp. 95–114. DOI: https://doi.org/10.1002/we.1822. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.1822. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1822.

[15] P. Hulsman, S. J. Andersen, and T. Göçmen. "Optimizing wind farm control through wake steering using surrogate models based on high-fidelity simulations". In: *Wind Energy Science* 5.1 (2020), pp. 309–329. DOI: 10.5194/wes-5-309-2020. URL: https://wes.copernicus.org/articles/5/309/2020/.

[16] Zamiyad Dar et al. "Windfarm Power Optimization Using Yaw Angle Control". In: *IEEE Transactions on Sustainable Energy* 8.1 (2017), pp. 104–116. DOI: 10.1109/TSTE.2016.2585883.

[17] Jason R. Marden, Shalom D. Ruben, and Lucy Y. Pao. "A Model-Free Approach to Wind Farm Control Using Game Theoretic Methods". In: *IEEE Transactions on Control Systems Technology* 21.4 (2013), pp. 1207–1214. DOI: 10.1109/TCST.2013.2257780.

[18] Pieter M.O. Gebraad, Filip C. van Dam, and Jan-Willem van Wingerden. "A model-free distributed approach for wind plant control". In: *2013 American Control Conference*. 2013, pp. 628–633. DOI: 10.1109/ACC.2013.6579907.

[19] P. M. O. Gebraad and J. W. van Wingerden. "Maximum power-point tracking control for wind farms". In: *Wind Energy* 18.3 (2015), pp. 429–447. DOI: https://doi.org/10.1002/we.1706. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.1706. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1706.

[20] Chunghun Kim et al. "A model-free method for wind power plant control with variable wind". In: *2014 IEEE PES General Meeting — Conference Exposition*. 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939478.

[21] Jinkyoo Park and Kincho H. Law. "A data-driven, cooperative wind farm control to maximize the total power production". In: *Applied Energy* 165 (2016), pp. 151–165. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2015.11.064. URL: https://www.sciencedirect.com/science/article/pii/S0306261915015147.

[22] Mahdi Abkar, Navid Zehtabiyan-Rezaie, and Alexandros Iosifidis. "Reinforcement learning for wind-farm flow control: Current state and future actions". In: *Theoretical and Applied Mechanics Letters* 13.6 (2023), p. 100475. ISSN: 2095-0349. DOI: https://doi.org/10.1016/j.taml.2023.100475. URL: https://www.sciencedirect.com/science/article/pii/S2095034923000466.

[23] Zhiwen Deng et al. "Decentralized yaw optimization for maximizing wind farm production based on deep reinforcement learning". In: *Energy Conversion and Management* 286 (2023), p. 117031. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2023.117031. URL: https://www.sciencedirect.com/science/article/pii/S0196890423003771.

[24] Paul Stanfel et al. "A Distributed Reinforcement Learning Yaw Control Approach for Wind Farm Energy Capture Maximization". In: *2020 American Control Conference (ACC)*. 2020, pp. 4065–4070. DOI: 10.23919/ACC45564.2020.9147946.

[25] Grigory Neustroev et al. "Deep Reinforcement Learning for Active Wake Control". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022, pp. 944–953. ISBN: 9781450392136.

[26] Claire Bizon Monroc et al. "WFCRL: A Multi-Agent Reinforcement Learning Benchmark for Wind Farm Control". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 133254–133281. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/f0a4a0ecdc29a0087c0848948e2fce81-Paper-Datasets_and_Benchmarks_Track.pdf.

[27] Jingjie Xie et al. "Wind Farm Power Generation Control Via Double-Network-Based Deep Reinforcement Learning". In: *IEEE Transactions on Industrial Informatics* 18.4 (2022), pp. 2321–2330. DOI: 10.1109/TII.2021.3095563.

[28] Hongyang Dong and Xiaowei Zhao. "Composite Experience Replay-Based Deep Reinforcement Learning With Application in Wind Farm Control". In: *IEEE Transactions on Control Systems Technology* 30.3 (2022), pp. 1281–1295. DOI: 10.1109/TCST.2021.3102476.

[29] Huan Zhao et al. "Cooperative Wind Farm Control With Deep Reinforcement Learning and Knowledge-Assisted Learning". In: *IEEE Transactions on Industrial Informatics* 16.11 (2020), pp. 6912–6921. DOI: 10.1109/TII.2020.2974037.

[30] Hongyang Dong, Jincheng Zhang, and Xiaowei Zhao. "Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations". In: *Applied Energy* 292 (2021), p. 116928. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2021.116928. URL: https://www.sciencedirect.com/science/article/pii/S0306261921004086.

[31] Grigory Neustroev et al. "Deep Reinforcement Learning for Active Wake Control". In: *International Conference on Autonomous Agents and Multi-Agent Systems*. Online: IFAAMAS, May 2022.

[32] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2018. arXiv: 1506.02438 [cs.LG]. URL: https://arxiv.org/abs/1506.02438.

[33] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[34] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[35] A. P. J. Stanley, C. J. Bay, and P. Fleming. "Enabling control co-design of the next generation of wind power plants". In: *Wind Energy Science* 8.8 (2023), pp. 1341–1350. DOI: 10.5194/wes-8-1341-2023. URL: https://wes.copernicus.org/articles/8/1341/2023/.

[36] Maarten J. van den Broek and Jan-Willem van Wingerden. "Dynamic Flow Modelling for Model-Predictive Wind Farm Control". In: *Journal of Physics: Conference Series* 1618.2 (Sept. 2020), p. 022023. DOI: 10.1088/1742-6596/1618/2/022023. URL: https://dx.doi.org/10.1088/1742-6596/1618/2/022023.

[37] Marco Pleines et al. *Generalization, Mayhems and Limits in Recurrent Proximal Policy Optimization*. 2022. arXiv: 2205.11104 [cs.LG]. URL: https://arxiv.org/abs/2205.11104.

[38] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

[39] Edward S. Hu et al. "Privileged Sensing Scaffolds Reinforcement Learning". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=EpVe8jAjdx.

[40] Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. *Informed POMDP: Leveraging Additional Information in Model-Based RL*. 2024. arXiv: 2306.11488 [cs.LG]. URL: https://arxiv.org/abs/2306.11488.

[41] Adithya Ramesh and Balaraman Ravindran. *Physics-Informed Model-Based Reinforcement Learning*. 2023. arXiv: 2212.02179 [cs.LG]. URL: https://arxiv.org/abs/2212.02179.

A Hyperparameter search and training details

The hyperparameters used for the PPO agents in the different experiments were found through random search. 10 different configurations of hyperparameters are evaluated for each experiment. Each configuration of hyperparameters is evaluated twice and its scores are averaged to reduce the impact of the stochasticity of the training process on the score metric. The ranges and different possible values for hyperparameters are defined in Table 2.

Table 2: Hyperparameter search ranges for PPO agent experiments

Hyperparameter	**Search Range/Values**
Discrete Values	
Batch Size	$\{128, 256, 512\}$
Network Layers	$\{2, 3, 4, 5\}$
Network Width	$\{128, 256, 512\}$
Number of PPO Epochs	$\{5, 10, 15\}$
Privileged obs spacing	$\{25, 75, 125\}$ or $\{100, 150, 200\}$
Continuous Ranges	
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-3}]$
Discount Factor (γ)	$[0.8, 0.99]$
GAE Lambda (λ)	$[0.9, 1.0]$
Clip Range	$[0.1, 0.3]$
Entropy Coefficient	$[0.0, 0.1]$
Value Function Coefficient	$[0.4, 0.6]$
Maximum Gradient Norm	$[0.25, 0.75]$

B Convergence of PPO training

All PPO models were trained for 100,000 steps during the hyperparameter search. While this is computationally expensive due to the complexity of the environment, it may be fruitful to train models for more steps. A higher step count may be particularly necessary for larger environments (e.g., LHS8 or LHS16) or privileged agents, which involve larger observation spaces and may require more exploration. This section presents learning curves to provide qualitative insights into the convergence behavior of different PPO models in the quasi-dynamic environment, comparing standard PPO and privileged PPO agents. The learning curves for the 4 Symmetric environment are shown in Figure 16 and the learning curves for the 8LHS environment are shown in Figure 17.

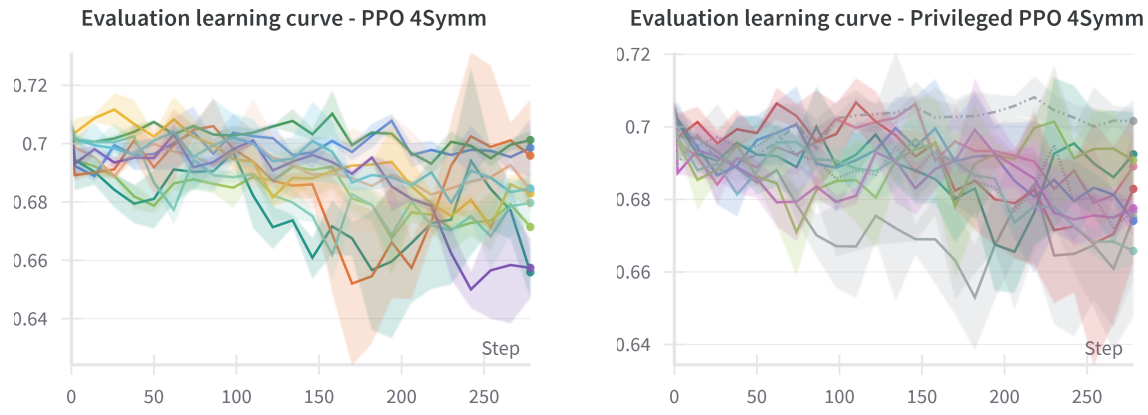

Figure 16: Learning curves of PPO agents in the 4 Symmetric quasi-dynamic environment. Plots show the average reward of the model during training for the evaluation seeds. Results are shown for 10 iterations of random search.

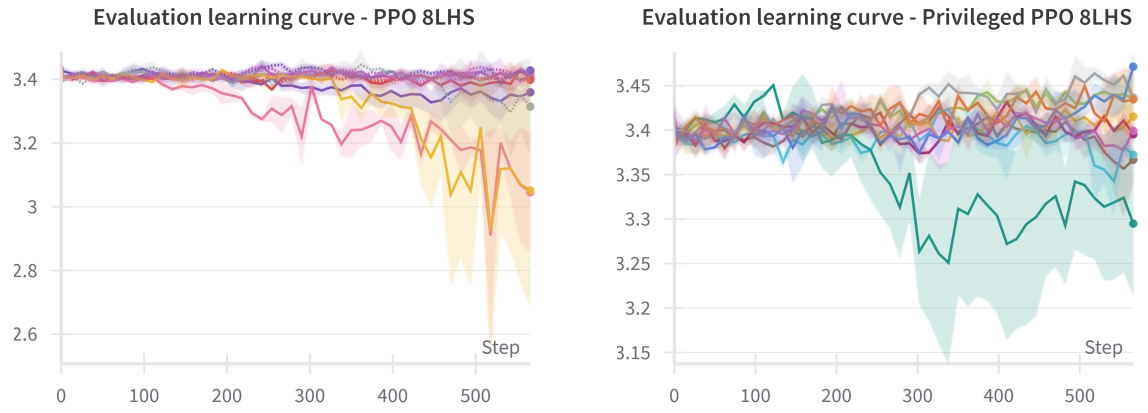

Figure 17: Learning curves of PPO agents in the 8LHS quasi-dynamic environment. Plots show the average reward of the model during training for the evaluation seeds. Results are shown for 10 iterations of random search.

The plots indicate that most seeds seem to have converged by the end of training for the 4 Symmetric and 8LHS layouts. Some hyperparameter settings show performance degradation over training in the 8LHS environment. Finally, some seeds do not seem to have converged by the end of training in the 8LHS environment. This may indicate that longer training runs could help improve the performance of privileged PPO in the 8LHS environment, but more experiments are needed to confirm this.