Delft University of Technology

Alternating Maximisation for Active Wake Control

Enhancing static yaw optimisation and reducing noise in multi-agent deep reinforcement learning for dynamic yaw control

Onno Verberne

TU Delft

Alternating Maximisation for Active Wake Control

Enhancing static yaw optimisation and reducing noise in multi-agent deep reinforcement learning for dynamic yaw control

by

Onno Verberne

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday July 15, 2024 at 15:00

Student number:	5883407	
Project duration:	November 27, 2023 – July 15, 2024	
Thesis committee:	Prof. dr. ir. F. A. Oliehoek,	TU Delft, chair, responsible supervisor
	Prof. dr. ir. M. M. de Weerdt,	TU Delft,
	Dr. G. Neustroev,	TU Delft, daily supervisor

Cover: Argentinian wind turbines by Bernadette Verberne

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

The journey of this thesis began with the Intelligent Decision Making Project from last year, where Greg introduced me to the fascinating problem of multi-agent reinforcement learning for active wake control. What initially drew me to this project was not only the challenge inherent in multi-agent reinforcement learning but also the potential to contribute something meaningful to society by working on wind energy. This intersection of complex problem-solving and societal impact ignited my passion for the topic.

Working alongside Marcus on this project was an incredibly rewarding experience. Our collaborative efforts and shared enthusiasm for the subject matter solidified our desire to pursue a thesis in this area. The process was both challenging and enlightening.

My focus throughout the thesis was on conducting research that yielded concrete results. I was driven by the desire to contribute something meaningful and substantial. However, this drive led to a unique challenge: I often found myself with solutions and results, but without concrete questions to address. This paradoxical situation became a humorous yet significant part of my research journey, reflecting the dynamic and iterative nature of scientific inquiry.

The support from Greg was invaluable throughout my thesis. His supervision and dedication were instrumental in guiding me through the complexities of my research. Greg's commitment to my success went above and beyond what I could have expected, and for that, I am deeply grateful. Additionally, the feedback and insights from my professors Frans and Mathijs were crucial in refining my work and helping me arrive at a thesis that I am truly proud of.

This thesis represents a significant milestone in my academic journey. While I am proud of what I have achieved, I acknowledge that I will likely never be fully satisfied with my accomplishments. There is always more to learn, more to improve, and more to discover.

With heartfelt thanks to everyone who supported and guided me, I present this thesis as a testament to our collective efforts and as a foundation for future exploration and innovation.

Onno Verberne
Delft, July 2024

Summary

This thesis investigates the application of alternating maximisation for active wake control in wind farms, focusing on both numerical static yaw optimisation and multi-agent deep reinforcement learning for dynamic yaw control. As the size and number of offshore wind farms continue to grow, effectively mitigating wake effects — where the airflow behind a turbine is disturbed, reducing the efficiency of downstream turbines — becomes increasingly important to maximise power output and ensure economic profitability.

Current numerical strategies are either sub-optimal, as they do not leverage the shape of the optimisation surface well, or expensive to optimise due to the computational demands of high-fidelity simulations. Similarly, current reinforcement learning approaches are too sample inefficient on contemporary wind farms of 40+ turbines to learn high-quality policies and likely suffer from credit assignment problems, other agents performing exploration, and relative over-generalisation. Alternating maximisation emerges as a promising candidate to address these challenges by providing an efficient and potentially more effective optimisation method.

In static yaw optimisation, the research analysed the effects of wind direction, wind speed, turbine layout, and yaw misalignment on the optimisation surface. Key findings indicate that Nash equilibria are transient and significantly influenced by these factors.

The sampling-based approach to alternating maximisation outperformed other methods, proving robust against variations in initial yaw configurations and optimisation orders. This approach required fewer samples and appears suitable for practical applications.

For dynamic yaw control, alternating maximisation with policy gradients demonstrated greater sample efficiency compared to distributed policy gradient methods. However, convergence to sub-optimal policies was common due to limited exploration capabilities. An oscillating exploration strategy enabled effective exploration of yaw angles, leading to high-quality policies.

The study primarily focused on farm power output, neglecting structural loading and leading-edge erosion. Future research should integrate these factors and explore dynamic yaw control under time-varying wind conditions.

Overall, alternating maximisation shows potential in enhancing wind farm performance through efficient optimisation strategies. Future work should refine these methods and address identified limitations to maximise practical applicability.

Contents

List of Figures

List of Tables

1

Introduction

Recent decades have seen an ever increasing amount of off-shore wind farms, furthering the contribution of wind to sustainable energy. The global average for wind farm sizes now approaches 40 turbines, with trends showing around a 22% increase in number of installed turbines per farm for pre/under-construction projects [23]. Naturally, the primary goals of operating wind farms are harvesting green energy and ensuring economic profitability, which are achieved by maximising farm power production and turbine longevity.

At present, wind farms are not being operated at optimal efficiency due to aerodynamic losses stemming from sub-optimal turbine control. As turbines extract energy from wind, they leave a wake of slower moving and more turbulent air, potentially hindering downwind turbines through so called wake effects [106]. Using technical terminology, an upwind turbine is said to be "waking" a downwind turbine. In practice, many turbine farms suffer wake-induced losses as their constituent turbines are each operated in isolation using a face-the-wind strategy [117]. However, as wind turbines are often operated in a farm setting, this face-the-wind strategy is sub-optimal with respect to the combined farm output, as it neglects the negative effect of wakes on downwind turbines.

Wake effects caused by upstream turbines can account for energy losses on the order of 10%–20% according to studies from the Danish Technical University [3] and the American National Renewable Energy Laboratory (NREL) [41]. It must be noted, however, that losses induced by wake effects are hard to separate from another source of aerodynamic losses, called blockage effects, due to their tight coupling [8]. Blockage effects are caused by in-flowing wind slowing down as it collides with lower velocity in-farm wind. Despite their tight coupling with wake effects, blockage effects are estimated to only account for up to a 5% loss in power production [64], making wake effect mitigation a more viable research direction.

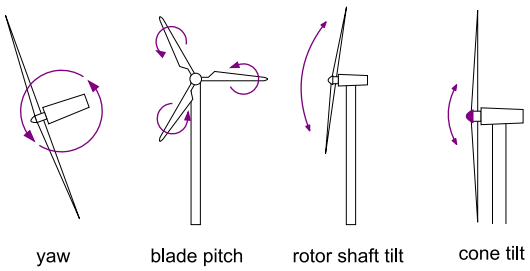

Figure 1.1: Active wake control methods.

As wind farms continue to grow in size [23], so will the number of turbine wakes and subsequent aerodynamic losses. Traditionally, these wake interactions have been mitigated through numerical modelling and simulation [29, 30, 41], which, while effective, demand significant computational resources (e.g., a cluster) if high model-accuracy is desired [31]. The approach by Fleming et al. [31] required 34.4 real-life hours on using the NREL Red Mesa supercomputer to *evaluate* a single setting of their wake control strategies under one wind condition. Training effective strategies requires comparatively more time than that of a single evaluation. For this reason, we require scalable optimisation methods for the wind farms of the future.

In the following sections, we will describe methods that balance computation time with the reduction of wake effects, and discuss the remaining challenges with these methods.

1.1. Wake Mitigation Strategies

In order to address the challenges posed by wake effects, various strategies known as wake control have been developed. Wake control can be separated into two broad categories:

- *passive*, which includes layout optimisation [103], hub-height optimisation [18, 51], and installation of structures such as wake-deflection barriers between turbines [62]; and

- *active*, which involves actively controlling turbine orientations (see Figure 1.1) such as yaw [37, 31, 111], blade pitch [52], rotor-shaft tilt angle [31, 111], or cone angle [111] to redirect the wake in response to wind conditions.

In case of existing wind farms, passive wake control strategies are likely to be expensive to implement as they require additional construction. On the contrary, active strategies generally utilise existing turbine-control methods, offering a more flexible and potentially cost-effective solution. An additional argument can be made for retrofitting solutions as the composite materials in turbine blades are hard to recycle [1, 81] and increasing the energy capture capacity of existing farms reduces the need for new construction.

Based on papers outlining the various wake control strategies by Fleming et al. [31] and Nash, Nouri, and Vasel-Be-Hagh [62], yaw control appears to be a promising strategy, yielding good power output whilst keeping additional structural loads minimal. By comparison, pitch control potentially yields more power at the cost of significant structural loading, tilt control is not well supported in existing turbines, and cone control seems to have a negative effect on power [62]. For these reasons, the primary focus of this thesis will be on active wake control via turbine yawing.

1.2. State of the Art in Yaw Control

Research into dynamic yaw control has been ongoing for at least four decades [19]. However, the current standard in existing wind farms still relies on individual turbine yaw-control [117], where the yaw-misalignment angle with the wind is minimised (i.e., turbines face the wind) to maximise a turbine's energy capture, as opposed to farm-wide yaw-control, which accounts for wake effects.

Yaw control, as a sub-field of active wake control, encompasses both dynamic yaw control and static yaw optimisation. Dynamic yaw control concerns finding strategies for yaw control in response to varying wind conditions. In contrast, static yaw optimisation pertains to optimising turbine yaw angles for a specific steady-state wind condition, and must therefore be done in simulation. Nevertheless, static yaw optimisation can be a part of dynamic yaw control strategies, such as lookup-table based yaw-control methods [29, 30, 89, 90, 80].

The upcoming sub-sections will explore conventional and contemporary optimization approaches for active wake control.

1.2.1. Numerical Optimisation

Regarding farm-wide control, lookup-table based methods [29, 30, 89, 90, 80] are the current field-tested approach. In this method, optimal yaw-angles are pre-computed for specific steady-state wind conditions. Pre-computation is often preferred over online computation, as accurate wake modelling is a non-trivial task. Optimal yaw angles for a selected set of wind conditions can be found via static yaw optimisation by, for example, employing gradient-based [37] or game-theoretic [35] optimisation in steady-state simulators like FLOw Redirection and Induction in Steady-state (FLORIS) [65]. Then, during execution, interpolation (e.g., nearest-neighbour through binning of wind parameters [30, 80]) is used to generalise the model to novel wind conditions.

Understandably, numerical methods relying on pre-computation require modelling of wake aerodynamics. This can be done with different levels of detail, creating a large trade-off between computational complexity and model accuracy. This trade-off can be observed through the various low- [66, 76], mid- [9, 54], and high-fidelity [40, 17, 57] simulators currently in use.

Model accuracy is paramount, as energy capture, even for the face-the-wind strategy of individual turbine control, can be sensitive to modelling errors [28]. Fortunately, model-free approaches [37] may sidestep the potential modelling errors by learning directly from data. However, a recent paper by Neustroev et al. [63] has demonstrated that noisy observations can also severely degrade the performance of numerical methods, which much therefore be taken into account in their application.

With the trend toward larger wind farms [23], numerical approaches using static yaw optimisation will become intractable to optimise naively, as the problem's search space grows exponentially with the number of turbines. Consequently, we require more sophisticated methods to explore the search space efficiently. To this end, Fleming et al. [32] propose optimising turbines sequentially instead of jointly using a method they named Serial-Refine. By breaking down the high-dimensional optimisation problem into a sequence of lower-dimensional optimisation tasks, the computation may become more tractable, although, potentially at the cost of optimality. Despite this, Serial-Refine [32] achieves performance on par with conventional gradient-based approaches in approximately a tenth of the time.

In contrast to numerical methods, machine-learning approaches do not suffer as much from

noise, nor are they impacted by the same modelling errors as they do not require an explicit model of the environment. In addition, machine learning methods are often cheap to execute due to allocating the vast majority of computation to the training phase, a period where model parameters are estimated from data, analogous to the precomputation step for lookup tables. Recently, reinforcement learning — a sub-field of machine learning — sees interest from researchers for dynamic yaw control for its capacity to deal with dynamic and stochastic environments [83, 94, 63, 46].

1.2.2. Reinforcement Learning

Reinforcement learning is a branch of machine learning where agents learn to make decisions by interacting with the environment and receiving feedback in the form of a reward signal [100]. The goal of reinforcement learning is to develop strategies, commonly called policies, that maximise the cumulative rewards over time. Unlike conventional machine learning, which regresses between input and output variables, reinforcement learning is characterised by a trial-and-error approach, where agents explore actions and learn from the consequences [100]. This paradigm is particularly suited for tasks requiring sequential decision-making and sees use in various domains such as robotics and autonomous systems.

In recent years, the field of reinforcement learning has demonstrated its efficacy in dynamic yaw control through both single-agent approaches [121, 24, 25, 116, 118, 55], where a single agent controls all turbines simultaneously, and more commonly multi-agent approaches [107, 95, 94, 71, 26, 2, 22, 46], where each agent controls either a single turbine or group of turbines.

Many approaches [107, 95, 121, 94, 24, 26, 25, 116, 118, 2, 55] use a fixed freestream wind direction to reduce the complexity of the problem. While a fixed wind direction is a good first step in solving dynamic yaw control through reinforcement learning, it is not representative of reality. Besides, optimal yaw-misalignment angles are highly sensitive to the wind direction [80], furthering the need for methods suitable for time-varying wind directions.

To this end, Kadoche et al. [46] have applied a multi-agent learning algorithm, independent proximal policy optimisation [114], to dynamic yaw control in variable wind conditions. In their experiments, the authors used real-life wind data from the North Sea. Moreover, their approach achieved success for farm sizes up to 151 turbines, where prior research is limited to at most 21 turbines.

In a recent paper by Neustroev et al. [63], the authors have shown that the way control actions are encoded has a large effect on the performance of deep reinforcement learning methods. Their results have shown that a wind-based action representation, where control actions are given in desired yaw-misalignment angles, currently performs best. The authors reasoned that, because the optimal control actions for steady wind conditions are identical, irrespective of the current yaw, the control actions are easier to learn and allow for more fine-tuned control as the turbine reaches the desired yaw-misalignment.

Lastly, there might still be performance to be gained over current solutions, as the approach by Kadoche et al. [46] on larger farms is still eclipsed by the numerical Serial-Refine method [32]. Highlighted by a relative power gain of 7.29% and 8.48% respectively. Furthermore, research on smaller farms has has been able to extract around 15% over baseline, which is more in line with the estimated 10–20% [3, 41] in power losses through wake effects.

In short, reinforcement learning is increasingly being explored for dynamic yaw control due to its ability to handle complex and challenging environments. It is inherently able to handle scenarios noisy observations, where numerical methods would fail. Furthermore, reinforcement learning can generalise to novel wind conditions, contrary to numerical methods, which operate based on a limited set of wind conditions due to computational costs. However, reinforcement learning remains challenged by the large search spaces induced by variable wind and large farm sizes. Consequently, reinforcement learning can fall short in power gain compared to numerical methods.

1.3. Challenges in Reinforcement Learning for Yaw Control

As the state space of the dynamic yaw control problem scales exponentially with the number of turbines, common single-agent methods require an exponential increase in the number of training samples to adequately explore the search space. Current single-agent approaches [121, 24, 25, 116, 118, 55] have not exceeded fifteen turbines, highlighting their practical limitations.

Appropriately, multi-agent reinforcement learning can be employed to alleviate this sampling problem by factoring the joint search-space into local sub-problems of smaller dimensionality. Choosing

the correct factorisation enables exploitation of the decentralised structure of the problem through parallel computation, and a local search space that does not scale exponentially with respect to the number of turbines. Despite these advantages, scaling up still proves difficult. To date, only one approach, by Kadoche et al. [46], has been tested on contemporary farm sizes of 40+ turbines.

However, reformulation as a multi-agent problem potentially introduces uncertainty caused by the actions of other agents into the environment, which can complicate learning if not handled appropriately [68]. Similarly, agents must behave in a cooperative manner to solve the dynamic yaw control problem, as acting greedily devolves into employing the face-the-wind strategy, which is sub-optimal in the presence of wake effects when turbines are positioned downwind from each other.

Credit Assignment

In multi-agent reinforcement learning problems where agents work towards a common goal, such as yaw control, it is typical to enforce cooperation through the reward signal. Current research is split between using common reward [26, 71, 46] and individual rewards [95, 94, 22].

Common rewards, such as the power output of a wind farm, are a straightforward way to enforce cooperation, as optimising for the common goal implicitly demands cooperation. Through yaw control, agents must sacrifice the output of their turbine such that one or more downwind turbines may produce more power, resulting in a overall power gain. The straightforward implementation of common reward does come with a caveat, agents must now overcome a credit assignment problem [14] arising from the inseparability of the reward signal due to wake interactions. If we are unable to reason about each agents' contribution to the common reward, we cannot give them correct feedback, through gradient updates or other means.

Using individual turbine power outputs to solve the credit assignment problem within active wake control is not an option. Individual power outputs do not take into account the effect of waking downwind turbines, and would result in use of the face-the-wind strategy. Current research seeks to address this short-coming through reward shaping. For example, by adding the power output of downwind turbines to individual power outputs [95, 94] or giving more reward for larger yaw angles to upwind turbines [22]. However, both are surrogates for the total farm output and neither guarantees optimality for the original problem.

Other Agents Performing Exploration

The problems arising from other agents performing exploration within multi-agent dynamic yaw control are twofold.

Firstly, under the common assumption that agents are rewarded based on the farm's total power production, other agents performing exploration complicates credit assignment by injecting noise into the reward signal. As agents explore different yaws, their sub-optimal actions can negate the power gains made by any other agent, and vice versa, which could make learning a good policy difficult. Moreover, if the exploration is uncoordinated, exploitative (greedy) agents may receive misleadingly poor rewards [110]. As the number of agents grows, it is reasonable to expect these problems will only become more challenging.

Secondly, other agents performing exploration injects noise into the local observations of downwind agents. Determining from local observations whether a decrease in wind speed comes from the environment or from the actions of another agent is non-trivial. For this reason, it may take more samples to learn effective policies.

Non-Stationarity of Policies

Non-stationarity of policies is a common learning problem in multi-agent reinforcement learning, and occurs due to the all agents learning simultaneously. Because agents update their policy, the optimal policy of other agents changes, resulting in a moving-target learning problem [11]. Moreover, from the perspective of an agent, one that does not directly observe the actions or internal state of other agents, the environment appears non-stationary due to other agents' policy updates affecting the observation probabilities, transition function, or rewards. As a result, the convergence guarantees of single-agent reinforcement learning no longer hold [68]. Despite this, independent learning, in which each agent considers the other agents as part of the environment, achieves success in multi-agent reinforcement learning [21, 110, 114], and more importantly, in dynamic yaw control [46]. However, the performance of independent learning in dynamic yaw control remains sub-optimal and can be further improved.

Relative over-generalisation

A problem known as relative over-generalisation [110] occurs when good but sub-optimal rewards lead agents to converge to a *sub-optimal Nash equilibrium*, that is, a local optimum where no agent can unilaterally improve the joint reward. Preventing convergence to or escaping

from Nash equilibria stemming from relative over-generalisation requires stronger exploration strategies than widely employed epsilon-greedy exploration [112], a common strategy where noise is injected into agents' actions, resulting in exploration of actions that deviate slightly from the policy learned by the agent.

In the context of dynamic yaw control, both negative and positive yaw-misalignment angles deflect wind around downwind turbines and can result in relatively good power gains over baseline. Figure 1.2 displays this fact using a power heat-map for the yaw angles of the leading two turbines in the single-line layout scenario presented in Figure 1.3. Because turbines rotate with a limited angular velocity, future yaw angles are dependent on current yaw angles, hence, uniformly random exploration is expected to explore around the current yaw angle. Consequently, relative over-generalisation could occur when the incorrect yaw-misalignment direction (clockwise or counter-clockwise) is chosen through random exploration. Without more sophisticated exploration strategies, agents may be unable to collectively explore the correct yaw-misalignment direction to find the global optimum.

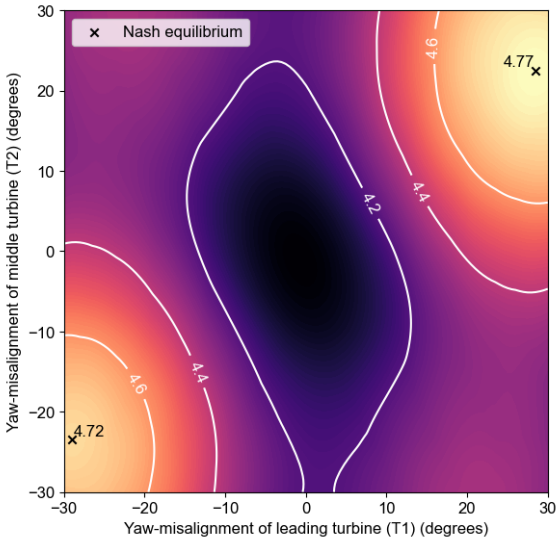

Figure 1.2: Heat-map of farm power (MW) for different yaws of the leading and middle turbines in the single-line layout scenario from Figure 1.3. The axes represent the yaw angles. Darker colours represent lower power output and lighter colours represent higher power ouput.

Summary

From the previously outlined challenges in multi-agent reinforcement learning for dynamic yaw control, we might expect that as farm sizes increase, the challenges posed by credit assignment, other agents performing exploration, and relative over-generalisation will increase in severity. For this reason, we conjecture that multi-agent methods for dynamic yaw control will gain the most success in their application to large wind farms by addressing these challenges.

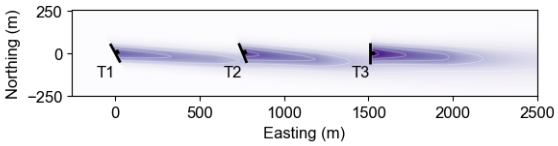

Figure 1.3: Wind map for a single-line layout with three turbines and wind coming from the west. Using the yaw-configuration from the Nash equilibrium in the top-right of Figure 1.2

1.4. Addressing the Challenges

We suggest alternating maximisation as a promising method for addressing multiple challenges in multi-agent reinforcement learning for active wake control. Although it has not been previously applied to these specific problems, it eliminates the issues stemming from credit assignment and other agents performing exploration, and potentially enables simple solutions to address relative over-generalisation. Moreover, it offers a straightforward and adaptable approach that can be integrated with virtually any reinforcement learning method. In the following section, we will describe the procedure in more detail and highlight its potential in active wake control.

1.4.1. Alternating Maximisation

Alternating maximisation is a common algorithm for optimising multivariate functions jointly over all variables and forms the basis for other widely used algorithms like expectation maximisation [6]. The core idea behind alternating maximisation is to split up a difficult joint optimisation problem into a sequence of easier optimisations on grouped subsets of variables and then to solve these subproblems iteratively [6]. The idea of alternating maximisation has been extended by Nair et al. [61] to solve decentralised partially observable Markov decision processes, a formal framework for cooperative multi-agent decision making used in reinforcement learning, yielding significant speed ups at the time.

In the context of multi-agent decision making, alternating maximisation is an iterative procedure that optimises the agents' policies one by one. Each agent individually optimises a best-response policy that maximises the joint reward, while the policies of other agents remain fixed. This process is

repeated until the joint policy converges to a Nash equilibrium [61], that is, a local optimum where no single agent can unilaterally improve the joint reward. Unfortunately, Nash equilibria found by this hill-climbing process can be arbitrarily bad [68]. Despite this, alternating maximisation does see practical use in reinforcement learning, where it is used for its computational efficiency [27, 109] and stability [102]. Moreover, Teh et al. [102] liken the alternating maximisation procedure to trust-region methods such as trust-region policy optimisation [74, 87], which is a state-of-the-art approach in reinforcement learning, further hinting at the potential benefits of alternating maximisation.

When applying alternating maximisation to dynamic yaw control, the procedure makes only one agent explore while keeping the policies of other agents fixed; this eliminates the noise from other agents performing exploration. By the same token, the credit assignment problems created by concurrent exploration are quashed. Furthermore, the best-response problems are drastically smaller than the joint optimisation problem as they concern only one turbine, which simplifies the exploration process. For that reason, we may be able to prevent relative over-generalisation within the best-response problems. However, as stated before, alternating maximisation only guarantees convergence to a Nash equilibrium, which is not necessarily a more optimal solution than can be found through joint optimisation.

We posit that the shape of the yaw optimisation surface may be conducive to hill-climbing via best-response sub-problems, making the optimisation of yaw-control more manageable. This is primarily because there are only two significant ways to adjust the yaw of the turbine to deflect wakes around downwind turbines: rotating clockwise or counter-clockwise. This binary choice simplifies the decision-making process.

For example, in a common single-line layout with axis aligned wind (see Figure 1.3), which represents the worst possible scenario because of the many wake interactions [63], there is a preferred direction: counter-clockwise rotation (i.e., a positive yaw-angle). This corresponds to the optimum in the top-right of Figure 1.2. More importantly, assuming turbines start by facing the wind, the best-response of the leading turbine, as shown in Figure 1.4, is found also found in the counter-clockwise direction. Consequently, hill-climbing via subsequent best-response leads to the global optimum. The corresponding convergence path is overlaid on the heat-map in Figure 1.5.

Moreover, in scenarios where no turbines are positioned directly downwind, the optimal yaw angle is straightforward to determine: turbines should face the wind to maximise their power output. Any deviation from this orientation results in decreased efficiency, particularly if it causes additional wake effects on downwind turbines. Thus, it becomes evident that starting with a base face-the-wind orientation and applying hill-climbing will naturally lead to the optimal solution in such cases.

For these reasons, the optimisation process for each turbine largely revolves around selecting the correct yaw direction. Once the correct direction is chosen, climbing to the optimum of the chosen mode (hill) should be trivial. The local optimisation for each turbine, when iteratively combined, leads to an effective global optimisation of the entire single-line layout. Since most offshore wind farms make use of grid-like layouts [88], like the the single-line layout, we expect this favourable structure to exist in real-life wind farms.

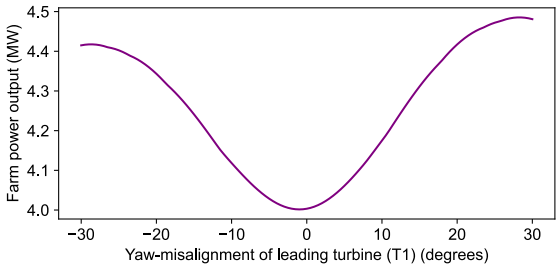

Figure 1.4: Best-response sub-problem for the leading turbine in the single-line layout of Figure 1.3.

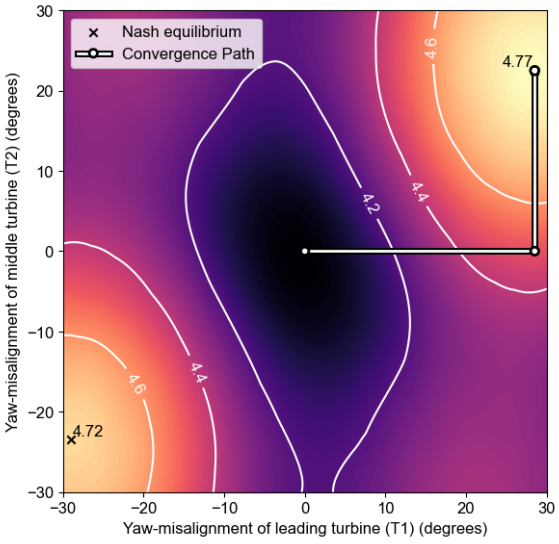

Figure 1.5: Heat-map of farm power (MW) for the leading and middle turbines in Figure 1.3 with the convergence path resulting from solving best-response problems.

1.5. Research Objective

The goal of this thesis is to investigate the benefits and challenges of applying alternating maximisation to active wake control, specifically focusing on numerical optimisation of static yaw control, and multi-agent deep reinforcement learning for dynamic yaw control.

To address this goal, we propose answering the following research questions:

Q1 In static yaw optimisation, which atmospheric conditions and wind farm layouts are favourable to alternating maximisation and which are hindering it?

To thoroughly investigate the factors affecting structure of the optimisation surface in static yaw optimisation, we focus on the following four sub-research questions:

Q1.1 How does wind direction affect the shape of the optimisation surface in static yaw optimisation?

Q1.2 How does wind speed affect the shape of the optimisation surface in static yaw optimisation?

Q1.3 How does the wind farm layout affect the shape of the optimisation surface in static yaw optimisation?

Q1.4 How do turbine yaw-misalignment angles affect best-responses in static yaw optimisation?

Q2 How can alternating maximisation method be effectively applied to static yaw optimisation, and what factors influence its performance and optimality of the resulting Nash equilibria?

Q3 In dynamic yaw control via reinforcement learning, does alternating maximisation impact the convergence to global optima?

Q4 Can domain knowledge be used to prevent convergence to local optima in dynamic yaw control for reinforcement learning agents trained using alternating maximisation?

1.6. Thesis Structure

The main goal of this thesis is to answer the aforementioned research questions. The remaining text is organised into a preliminaries chapter followed by four content chapters, each dedicated to addressing on of the primary research questions.

Chapter 2 covers the preliminaries, exploring the fundamental concepts and theories necessary for understanding the content chapters. This includes an overview of the active wake control problem, reinforcement learning, as well as a multi-agent formulation of the dynamic yaw control problem.

Chapter 3, the first content chapter, addresses RQ1, exploring the impact of wind direction, wind speed, different turbine layouts, and varying yaw-misalignment on the structure of yaw optimisation, and serves to build our domain knowledge on active wake control.

Chapter 4 focuses on RQ2, examining the conditions under which alternating maximisation leads to sub-optimal outcomes, exploring factors such as gradient-based versus sample-based methods, initial yaw configurations, and optimisation order, furthering domain knowledge applicable to reinforcement learning for dynamic yaw control.

Chapter 5 addresses RQ3, comparing the performance and convergence behaviour of agents trained using different optimisation strategies, and providing insights into their effectiveness and limitations for dynamic yaw control.

Chapter 6 explores RQ4, investigating the potential of incorporating domain knowledge into exploration strategies to enhance the performance of reinforcement learning agents trained through alternating maximisation.

Lastly, in Chapter 7. we summarise our key findings, discuss their implications, and suggest directions for future work.

2 Preliminaries

We begin by providing the necessary background information on the topics of active wake control and wind simulation, followed by the core tenets of sequential decision-making and reinforcement learning, also covering work related to the challenges outlined in Chapter 1. Thereafter, we will introduce active wake control as a multi-agent reinforcement learning problem. We aim to provide a common language for researchers from both fields, refraining the use of any domain specific acronyms as to maintain readability for researchers from both backgrounds.

2.1. Active Wake Control

Active Wake Control is a strategy for operating wind farms that seeks to maximise the total power output through managing wakes produced by wind turbines, specifically by minimising wake-induced losses. Wind turbines create a wake area behind them as the wind drives the turbine blades [106], these wakes are characterised by decreased wind speeds and increased turbulence. Both effects lead to decreased power output for any turbine located within the waked area.

It is possible to mitigate adverse wake effects by controlling the strength and direction of the wake, enhancing the farms total energy output [62, 63]. The primary means of manipulating turbine wakes involve adjusting the nacelle yaw and tilt angles to redirect the wake away from downwind turbines in the horizontal and vertical plane, respectively [31, 62]. Additionally, modifications to the blade pitch and torque controls can accelerate wake recovery [62], the process of flow regressing to an undisturbed state via turbulent mixing [104].

Campagnolo et al. [12] have shown that a yaw angle of 16° can increase the farms power output by 15%, using a wind tunnel setup with three inline turbines. Simulation-based studies have yielded similar results [63, 24, 26, 71, 22, 55, 46, 36, 85]. Furthermore, yaw control appears to keep structural loads relatively low, compared to pitch control, which potentially yields more power at the cost of higher structural loads [31, 62]. Similarly, tilt control does not apply much extra loading to the turbine [62], and has demonstrated power increases up to 12% [111]. However, considerably less research into tilt control has been conducted despite the promising results, likely due to few existing turbines supporting the control mechanism [111]. In short, yaw control appears to be a fruitful approach, yielding good gains in total power output while keeping additional structural loads minimal, in addition to hosting a comparatively large body of research.

2.1.1. Static Yaw-Optimisation

Static yaw-optimisation concerns optimising the yaw of a turbine for a specific wind condition. It is a non-linear multivariate optimisation problem, with the number of dimensions generally equal to the number of turbines. There are often multiple modes in the optimisation surface, as there are often two ways (i.e., clockwise and counter-clockwise rotation) to deflect wind around downwind turbines, which makes this a non-trivial optimisation problem. If one were to simply follow the direction of the steepest gradient, one would eventually end up at the peak of one of the modes – a local optimum – in the optimisation surface, with no way to escape that mode via gradient ascent and find the global optimum.

Commonly, static yaw optimisation is used as a pre-computation step for lookup-table based yaw-control [29, 30, 89, 90, 80], a numerical yaw control method characterised by responding to current wind conditions, as opposed to planning for future wind conditions. Optimisation of the problem can be achieved through various means, such as gradient-based [37], game-theoretic [35], or sampling-based [32] optimisation. All of which are usually done in a simulator, like FLOw Redirection and Induction in Steady-state (FLORIS) [65], although model-free methods [37] could also be applied in real-life wind farms.

2.1.2. Dynamic Yaw Control

Dynamic yaw control is an extension of static yaw-optimisation. By introducing time-varying wind or limiting the angular velocity with which turbines can rotate, the space of solutions grows much larger than that of static yaw-optimisation. For example, by introducing time varying wind directions, the solution space grows from $|Y|^N$ to $(|Y| \cdot |D|)^N$, where N is the number of turbines, $|Y|$ is the number of yaws, and $|D|$ is the number of wind directions. Even with a coarse sampling resolution of 1°, sampling wind directions leads to a significant increase in size of the search-space.

Moreover, the introduction of a limit on the rotational angular velocity introduces a temporal component into the problem. There is no longer a single best response to every wind condition, as the response now depends on the current yaw-angle. Furthermore, a limited yaw speed restricts exploration to be localised around the current yaw-angle, as it now takes time to move the turbine from one yaw setting to another, which previously was instant in the static yaw-optimisation problem. As a result, moving the turbine through the full range of yaw angles requires a concerted effort.

2.1.3. Simulation

The dynamics of wake propagation and interaction can be studied through experimentation or through simulation models. Simulation models are the most cost-effective and accessible way to run wind experiments, and allow for the most customisation such as varying turbine layouts, turbine types, and wind conditions. This freedom to configure makes simulation the preferred approach for prototyping active wake control strategies.

Naturally, when it comes to simulation there exists a trade-off between model accuracy and speed. Simulators are usually categorised based on their model-fidelity, for which the following three categories are common: low, mid, and high. The choice of simulator depends on balancing the need for speed and precision at different stages of research and development.

Low-fidelity simulation generally makes use of hand-crafted wake interaction models, such as the model by Jensen [42] and others [20, 34, 44, 56, 92]. These models usually operate in a steady-state environment, where wakes are calculated for a specific wind condition and propagate instantly, forsaking any dynamic flow interactions between turbines [54]. Their simplicity is their greatest strength, enabling fast prototyping of active wake control strategies by leveraging cheap computations. The two primary simulators in this category are FLORIS [65] by the American National Renewable Energy Laboratory, and PyWake [76] by the Technical University of Denmark Department of Wind and Energy Systems.

On the other end of the spectrum are high-fidelity simulators, which use computational fluid dynamics to evolve the flow field over time, yielding much higher model accuracy and capturing dynamic flow interactions between turbines. Large-eddy simulation is the mathematical model of choice for simulating turbulent flow within computation fluid dynamics. Simulators using this approach include DALES [40, 84], SOWFA [17], and ASPIRE [57] (formerly GRASP, derived from DALES.) Unfortunately, high-fidelity simulators are very computationally expensive, often require a cluster to operate [49, 31, 24, 26].

Mid-fidelity simulators form a middle ground between low and high-fidelity simulation, trading modelling detail for computational efficiency. WFSim is a relatively popular mid-fidelity simulator [121, 25, 26, 116] by Boersma et al. [9], which uses 2D Navier-Stokes equations to trade model accuracy for computational efficiency. More recently, HAWC2Farm, by Liew et al. [54], uses aeroelastic wind turbine models in a simplified turbulent flow field, compared to computational fluid dynamics.

As stated before, there is a difference in dynamics between steady-state and the two other modelling paradigms due to the absence of dynamic flow interactions within steady-state simulations. Dynamic flow interactions allow for the propagation of wakes over time, which makes learning control strategies harder by obscuring the action effect through a delayed reward signal. To this end, Gebraad, Dam, and Wingerden [37] propose incorporating a delay structure to model the wake travelling from one turbine to the nearest downwind turbine in model-free methods. Unfortunately, their method must reconverge every time wind conditions change as this updates which downwind turbine is nearest. Stanfel et al. [94] improve on this method by computing the wake delay for a turbine after taking an action and locking the control until the wake has propagated. The average power output is then measured over a time window to get an improved estimate for the reward. In their work, Stanfel et al. [94] make use of a modified FLORIS environment called Dynamic FLORIS introduced by [108], which makes use of the Taylor frozen wake assumption[101] by only updating each grid point in the flow field after a set amount of time based on the distance to the location of change, thereby simulating a propagating wake.

2.2. Sequential Decision-Making

Sequential decision-making involves making a series of decisions over time, where each decision influences future options and outcomes. The actor within a sequential decision-making process is often called an agent. An agent observes the environment and perform an action, thereby causing the environment to transition into a new state and deliver the agent a reward [4]. This process is repeated ad infinitum or until a terminal state is reached. The goal of solving sequential decision making processes is often to maximise the cumulative reward.

In the context of sequential decision-making, a key assumption is that the environment remains stationary [70]. This implies that the transition function and reward function do not change over time. When the environment is stationary, agents can reliably learn from past experiences, as the outcomes of their actions remain predictable, even if the environment is stochastic.

2.2.1. Single-Agent Decision-Making

A multi-armed bandit problem is a classical model in decision-making. The name is derived from a gambler (an agent) facing several slot machines (referred to as "one-armed bandits"), each with an unknown reward function [100]. The key property of a multi-armed bandit is that the agent only observes the rewards from their actions. The agent must repeatedly play the same game to find the optimal strategy which maximises the reward. Formally, a multi-armed bandit problem consists of a set of actions (arms), each providing a reward drawn from an unknown distribution.

A Markov Decision Process (MDP) [5] is a mathematical framework used to model sequential decision-making problems, where outcomes are stochastic and partially under the control of a decision maker, the agent. What sets it apart from a multi-armed bandit is an evolving state, which is observed by the agent. Within this framework an agent takes an action a based on the fully observable current state s which transitions the environment to the next state s' and yields the agent a reward r. An MDP is formally defined as a 5-tuple (S, A, T, R, γ), where:

- S is a finite set of states.
- A is a finite set of actions.
- $T : S \times A \times S \to [0, 1]$ is the transition probability function, where $T(s'|s, a)$ is the conditional probability of transitioning from state s to state s' by taking action a.
- $R : S \times A \to \mathbb{R}$ is the reward function, where $R(s, a)$ is the immediate reward received after taking action a in state s.
  - Optionally: $R : S \to \mathbb{R}$, where the immediate reward is given by $R(s')$ for reaching state s'.
- $\gamma \in [0, 1)$ is the discount factor, representing the importance of future rewards compared to immediate rewards.

Markov Decision Processes can be used to model various kinds of decision-making problems, most of which can be categorised as either finite-horizon or infinite-horizon problems. Finite-horizon MDPs often give a positive reward for reaching a certain goal state at which the episode ends, and optionally a negative reward for every other state. In infinite-horizon MDPs the agent is provided with a reward signal – which can be positive or negative – for taking actions, here the desired behaviour is to maximise the cumulative reward. Solving an MDP can be done through planning, where the objective is to find a policy $\pi : S \to A$, which maximises the expected sum of discounted rewards over time, starting from some initial state. The desired behaviour of the optimal policy is therefore largely shaped by the reward function.

Partial Observability

While Markov Decision Processes provide a robust framework for modelling sequential decision-making problems, they assume that the agent has perfect knowledge of the current state. This assumption is often unrealistic in real-world applications where noise exists or full observability of the environment is not feasible. To address these limitations, Partially Observable Markov Decision Processes (POMDPs) extend the MDP framework to handle uncertainty in the state space [93][47]. Since the state in a POMDP is uncertain, an optimal decision-maker can no longer base its policy directly on what it observes. An agent within a POMDP must therefore maintain a belief over which state it currently resides in, either through maintaining a history or through a probability distributions over possible states $s \in S$.

Similarly to MDPs, POMDPs can be categorised as finite-horizon and infinite-horizon. Unlike MDPs, it is common for a POMDP to have a continuous state- and action space. As a consequence, solving POMDPs exactly is often intractable. To this end, many approximate methods have been proposed, one of which will be discussed in the upcoming section on reinforcement learning.

2.2.2. Multi-Agent Decision-Making

In addition to applications in single-agent decision making, Markov decision processes and partially observable Markov decision processes extend to cooperative multi-agent decision-making through the use of a centralised agent which controls all agents simultaneously and receives their joint observations, formally these extensions are known as multi-agent Markov decision processes and multi-agent partially observable Markov decision processes [68]. However, this joint control strategy encounters significant computational challenges when scaling the number of agents due to the curse of dimensionality. This challenge arises from a combinatorial explosion in the number of possible states and actions, which are exponential in the number of agents.

To address these limitations, decentralised partially observable Markov decision processes have been introduced [68]. They form a natural extension to the previous centralised example by opting for a decentralised control strategy, where each agent is modelled individually, resulting in a state-action space complexity that is sub-exponential in the number of agents. Agents in a Dec-POMDP receive their individual observations and have their own set of actions. Formally, a Dec-POMDP can be defined as an 8-tuple $(D, S, A, T, \Omega, O, R, \gamma)$:

- $D = \{1, \dots, n\}$ is the set of n agents.
- S is a set of states.
- $A = \bigtimes_{i \in D} A_i$ is a set of joint actions.
- $T : S \times A \times S \to [0, 1]$ is the transition probability function, where $T(s'|s, a)$ is the conditional probability of transitioning from state s to state s' by taking action a.
- $\Omega = \bigtimes_{i \in D} \Omega_i$ is a set of joint observations.
- $O : S \times A \times O \to [0, 1]$ is the observation probability function, where $O(o|a, s')$ is the conditional probability of seeing observation o after taking action a and ending up in s'.
  - Optionally: $O : S \to [0, 1]$, where the observation probability solely depends on the state s'.
- $R : S \times A \to \mathbb{R}$ is the reward function, where $R(s, a)$ is the immediate reward received after taking joint action a in state s.
  - Optionally: $R : S \to \mathbb{R}$, where the immediate reward is given by $R(s')$ for reaching state s'.
- $\gamma \in [0, 1)$ is the discount factor, representing the importance of future rewards compared to immediate rewards.

In an ideal case, a decentralised partially observable Markov decision process can factor the state and action spaces, reducing the overall complexity from exponential to polynomial in the number of agents and yielding a more compact representation. The resulting framework has been formalised as a factored decentralised partially observable Markov decision process (Factored Dec-POMDP), where the core idea is to exploit the conditional independence of the problem variables [67]. In a similar fashion, Factored Dec-POMDPs compactly represent the reward function under the assumption that it is additively separable by decomposing the reward into n local rewards, one for each agent. This differs from partially observable stochastic games (POSGs) where each agent has its own individual reward function [38].

Nash Equilibria

Nash equilibria are a game-theoretic concept, representing a situation where no player has an action that they prefer over the equilibrium action [69]. A common example of a Nash equilibrium can be found in the prisoners dilemma, a non-cooperative game. When one player confesses, it is best for the other player to also confess. In this local optimum, neither player can improve their own outcome, they have to work together to improve.

In the context of cooperative multi-agent decision-making, a Nash equilibrium represents a local optimum where no agent can unilaterally improve the joint outcome by adapting their own policy. In different terms, a Nash equilibrium is a set of policies where each agent's policy is a best response for the policies employed by other agents [68].

Independent Learning

In the field of multi-agent decision-making, independent learning refers to the scenario where a multi-agent learning problem is decentralised into single-agent problems [110, 114]. Multiple agents learn simultaneously without coordinating their learning processes or directly sharing information. Each agent operates based on its own observations, actions, and received rewards, while treating other agents as part of the environment.

Within this learning paradigm and from the perspective of an agent, the environment becomes non-stationary as a result of other agents' policy updates [68]. Accordingly, the convergence guarantees of single-agent learning are forgone, which are predicated on the assumption of a stationary environment [70]. Despite this, independent learning does achieve competitive performance [21, 110, 114, 46].

Alternating Maximisation

Alternating maximisation is a common algorithm for optimising multivariate functions jointly over all variables and forms the basis for other widely used algorithms like expectation maximisation [6]. The core idea behind alternating maximisation is to split up a difficult joint optimisation problem into a sequence of easier optimisations on grouped subsets of variables and then to solve these subproblems iteratively [6]. This idea can be applied to solve decentralised partially observable Markov decision processes, where it is an iterative procedure that optimises the agents' policies one by one [61]. Each agent individually optimises a best-response policy that maximises the joint reward, while the policies of other agents remain fixed. This process is repeated until the joint policy converges to a Nash equilibrium [61], which is not necessarily an equally or more optimal solution than found through joint optimisation [68]. Random restarts may be used to explore novel regions of the search space and better local optima [61].

2.3. Related Work Addressing Multi-Agent Learning Pathologies

In multi-agent systems, various pathologies can arise that complicate the learning process. These pathologies include non-stationarity [72], the credit assignment problem [14], other agents performing exploration [110], and relative over-generalisation [112]. Understanding and addressing these challenges is crucial for learning effective strategies in multi-agent systems. Various approaches have been developed to address the challenges of multi-agent learning pathologies. In the following subsections, we describe a few notable methods, in addition to our novel approach for addressing these challenges within active wake control.

2.3.1. Difference Rewards

Reward differencing is a method for tackling the credit assignment problem. As described by Wolpert and Tumer [115], the aim of this method is to fix the poor "signal-to-noise" ratio present in large multi-agent systems when discerning an individual action's contribution to the joint reward. To determine an agent's effect on the joint payoff, their contribution is marginalised out of the reward by computing the expected reward for the joint action given its current policy. By taking the difference between this expected reward and the actual reward, we arrive at a reward signal that better reflects an action's contribution to that reward.

A state-of-the-art reinforcement learning method, counterfactual multi-agent policy gradients [33], has applied the idea of differencing to deep reinforcement learning via approximating Q-value functions, which represent the expected return of an action in a given state. However, as outlined by Castellini et al. [13], learning the Q-value function in multi-agent settings can be difficult due to compounding factors of bootstrapping, moving targets, and Q-values' dependence on joint actions. These can make counterfactual multi-agent policy gradients difficult to scale up, which is required dynamic yaw control. To this end, Castellini et al. [13] show that function approximation of the reward function can yield better results in larger multi-agent settings through their algorithm Dr.ReinforceR, a difference rewards policy gradient algorithm derived from REINFORCE [113].

2.3.2. Value Decomposition

In fully cooperative settings, value decomposition [98, 82, 122, 43] is a method for implicitly solving the credit assignment problem [122, 43]. At its core, value decomposition addresses credit assignment by learning by learning to map individual Q-value functions to the joint Q-value function. However, as previously addressed by Castellini et al. [13], learning these Q-value functions in multi-agent settings can be hard. We argue that this is compounded within dynamic yaw control due to the stochastic nature of the wind. Agents have no control over the in-flowing wind, and by extension their observations and transitions. Consequently, bootstrapping Q-values from wind may require many samples, where other reinforcement-learning methods fare better.

2.3.3. Group-based Learning

One of the challenges of other agents performing exploration is the lack of coordination [110]. Group-based learning has emerged as a potential solution, as it promotes effective coordination and information sharing among agents. This approach, as explored by Zhang, Abdallah, and Lesser [119] and Zhang and Lesser [120], involves grouping agents and coordinating by exploiting the locality of interactions, which has resulted in accelerated learning and improved performance. A similar idea has been applied by Dong and Zhao [26] to dynamic yaw control to improve the sample efficiency of deep reinforcement learning methods.

However, within dynamic yaw control, the locality of interactions is very dependent on wind direction, as it determines which turbines wake each other. Consequently, the use of fixed groups is probably

sub-optimal. Learning with naive dynamic group sizes would likely be non-trivial, as the reward function would change over time, making the environment non-stationary. One could attempt to address this through normalisation of group rewards, if it accounts for the highly non-linear wake interactions unique affecting each turbine's power output. Such reward shaping is currently an unsolved problem within reinforcement learning for active wake control [95, 94, 22].

2.3.4. Lenient Learning

Lenient learning, as introduced by Wiegand et al. [112], is a method for addressing the pathology of relative over-generalisation. Recently, the method has also been applied to address miscoordination within exploration [110], as well as non-stationarity in multi-agent systems [73]. The core idea behind lenient learning is that agents make bad decisions early on in their training, which we should apply leniency to. In practise, this leniency is applied to Q-values, the expected return of a state-action pair. State-action pairs start out with a lenient or optimistic value, which decays to the learned value as the pair is the visited. As a result, agents will initially explore the lenient actions, after which they will start to exploit the optimal actions given the learned Q-values.

2.3.5. Concluding Remarks

While the aforementioned methods provide significant advancements in addressing multi-agent learning pathologies, they each have their limitations and specific areas of applicability. A promising candidate for addressing these issues, particularly in the context of active wake control, is alternating maximisation. This method, which as not been extensively explored in this domain, offers a potentially robust solution for the credit assignment problem, other agents performing exploration, and relative over-generalisation. By exploring with one agent at a time, challenges imposed by credit assignment and other agents performing exploration are quashed, while relative over-generalisation may be simpler to address, as described in Chapter 1

2.4. Reinforcement Learning

Reinforcement learning is a branch of machine learning where agents learn to make decisions by interacting with the environment and receiving feedback in the form of a (delayed) reward signal [100]. The goal of reinforcement learning is to develop strategies, commonly called policies, that maximise the cumulative rewards over time. This paradigm is particularly suited for tasks requiring sequential decision-making, where agents must also navigate complex environments and maximising cumulative rewards over time.

One of the challenges within reinforcement learning is the trade-off between exploration and exploitation [100]. Agents exploit known good actions to maximise rewards, but to find such actions agents must explore. This problem can be especially hard in stochastic tasks, where outcomes are uncertain, each section must be tried repeatedly to learn an effective policy. Within active wake control, such uncertainty can be caused by the partial observability of the wind, and noise induced by imperfect sensors. Multi-agent settings exacerbate this, as there is no centralised agent that controls all turbines, meaning other agents' policies and their effects must be learned. However, these policies can change over time. This form of uncertainty is called non-stationarity, also known as the moving target problem, and is one of the multi-agent learning pathologies.

2.4.1. Policy Gradient Methods

Policy gradient methods are a family of reinforcement learning algorithms used to optimise a policy directly. Policy gradient methods use a policy that is differentiable with respect to its parameters (e.g., a neural network) and then optimise these parameters using gradient ascent. The core idea is to adjust the policy parameters in the direction that maximises the expected cumulative reward [99].

Formally, let $\pi_\theta(a|s) = \Pr(a \mid s, \theta)$ denote the policy, parameterised by θ, which gives the probability of taking action a in state s. The objective is to maximise the expected return $J(\theta) = \mathbb{E}_{\pi_\theta}[G]$, where $G = \sum_{t=0}^{\infty} \gamma^t r_t$ represents the discounted cumulative reward, also known as the return. The policy gradient theorem [99] states that the gradient of $J(\theta)$ with respect to θ can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]$$

where $Q^\pi(s,a) = \mathbb{E}[G \mid s_t = s, a_t = a, \pi]$ is the action-value function, which represents the expected sum of cumulative rewards for taking action a in state s. This expression forms the basis of policy gradient methods, where the policy parameters are iteratively updated using stochastic gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

with α being the learning rate.

REINFORCE [113] is one of the foundational policy gradient methods and is directly derived from the policy gradient theorem. As such, REINFORCE is is an on-policy algorithms, meaning it explore and learn by sampling actions according to their current stochastic policy. Moreover, the algorithm is a Monte-Carlo method, meaning it executes a its current policy for T time steps in an episode. The policy update can then be done via:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T-1} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

where G_t represents the return at time step t.

REINFORCE has theoretical convergence guarantees [113], under which it will converge to a local optimum. However, as a Monte-Carlo method, it may suffer from high variance between episodes, resulting in slow convergence [100].

In cooperative multi-agent settings, a distributed policy gradient [77] variant can be used to optimise each agent's policy. Distributed policy gradient methods are amongst the few methods with convergence guarantees in multi-agent settings [77]. The policy updates for each agent i are as follows:

$$\theta_i \leftarrow \theta_i + \alpha \sum_{t=0}^{T-1} \gamma^t G_t \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i|s_t)$$

2.4.2. Stochastic Policy Gradients

In stochastic continuous control problems, such as dynamic yaw control, it is common to represent the policy distribution using a Normal distribution [113]. A function approximator, including neural networks, is used to compute the action distribution parameters (i.e., mean and variance) given the current state. However, many such control problems, including dynamic yaw control, have bounded action spaces. In these scenarios, using a distribution with infinite support, for instance, a Gaussian distribution, introduces a bias in the distribution due to boundary effects [16]. Actions outside of the bounds are usually clipped, meaning that if there are sufficiently good rewards near the action boundaries, there will be an over-representation of "good" actions near the boundaries. As a result, gradient ascent will bias the policy distribution towards the action bounds.

To this end, Chou, Maturana, and Scherer [16] suggest using a Beta distribution, which is a continuous probability distribution defined on the interval $[0, 1]$. It is parameterised by two shape parameters, α and β, which control the shape of the distribution, allowing it to take on uniform, U-shaped, and bell-shaped distributions. This flexibility allows the Beta distribution to model a wide range of action distributions. Moreover, the shape parameters provide a natural mechanism to control the exploration-exploitation trade-off. By tuning α and β, the policy can be made more deterministic (peaked) or more exploratory (flatter), allowing adaptive exploration strategies during learning. Petrazzini and Antonelo [78] later applied the work by Chou, Maturana, and Scherer [16] to improve the performance of Proximal Policy Optimisation [86], a state-of-the-art family of policy gradient methods, on continuous control tasks in bounded action spaces.

During evaluation, a stochastic policy can be made deterministic, which may be preferred for some problems where precise control is required or where noisy actions are undesired, such as in dynamic yaw control [30, 48, 89, 90]. However, in a stochastic environment, a stochastic policy can perform better than a deterministic policy [91].

2.5. Dynamic Yaw Control as a Multi-Agent Reinforcement Learning Problem

In this work, we will be applying multi-agent reinforcement learning algorithms to the dynamic yaw control problem. In order to communicate through a common language grounded in theory, we formulate the problem as a decentralised partially observable Markov decision process. We use the recent theoretical framework for active wake control as a reinforcement learning problem by Neustroev et al. [63] as a basis for our description. Similarly, we adapt the FLORIS V2.4 [65] driven OpenAI Gym [10] environment by Neustroev et al. [63] for our simulation experiments.

The following sections define the states, observations, actions, rewards, and transition function.

2.5.1. States and Observations

A Markov decision process is characterised by its Markovian state, of which the future state depends only upon the present state. Since the flow-field of a wind farm evolves via physical processes, we can postulate this assumption holds, supporting the use of a Markovian state for modelling wind. However, letting agents observe the complete flow-field is both computationally impractical as well as physically infeasible. Thus agents observe only specific data-points, from which they must extrapolate the state of the environment.

Using wind parameters from the current flow-field state s to generate observations, at each time step t the observation o_i for turbine i comprises: wind speed at the turbine M, wind direction at the turbine ϕ, the freestream turbulence intensity I, as well as the turbine's current yaw angle y, resulting in the vector $[M, \phi, I, y] \in \mathbb{R}^4$. The yaw angle is included because the next yaw angle depends on it, omitting the measurement would make the environment non-stationary. Each measurement is normalised to an interval between 0 and 1 according to the range of possible values, with the exception of the yaw, which is normalised to the interval [-1, 1], making the yaw observation-space identical to the action-space described in the following section. Additionally, to simulate sensor noise, at each time step each individual normalised measurement is perturbed by independently sampled zero-mean Gaussian noise with a variance of 0.1.

2.5.2. Actions

Each joint action $a = \{a_i \mid \forall i \in D\}$ comprises D individual actions $a_i \in [-1, 1]$ corresponding to a set of *desired* yaw-misalignment angles $Y = \{y_i \mid \forall i \in D, \}$ where y_i is within the interval $[\text{min yaw}, \text{max yaw}]$. Recent reinforcement learning approaches for active wake control [71, 63, 55, 22, 46] use similar yaw-misalignment angle based action-representations, which have been shown to be the current best action-representation [63].

In this action representation, $a = +1$ corresponds to the maximum positive yaw angle relative to the wind, a counter-clockwise rotation. The new desired yaw is calculated as:

$$y' = \phi + \frac{1}{2}(a + 1) \cdot (y_{\text{max}} - y_{\text{min}}) + y_{\text{min}}$$

where y_{max} and y_{min} represent the yaw boundaries [63]. The final yaw angle is is computed by first clipping y' between the yaw boundaries, after which it clipped the interval $[y - \omega_{\text{max}}, y + \omega_{\text{max}}]$ given by the maximum angular velocity ω_{max}.

2.5.3. Rewards

With the goal of maximising power production, the agents receive a common reward based on the instantaneous power output of the entire farm. In essence, the reward function judges the quality of the chosen yaw configuration. We use a common reward to enforce cooperation between agents, as maximising collective power production does require cooperation through sacrificing upwind turbines' individual output. Alternatively, individual rewards would collapse the agents' optimal policies to the face-the-wind policy as that maximises an individual turbine's power production.

As is standard practise within reinforcement learning, the reward is normalised to the interval $[0, 1]$. Within this normalisation, 0 and 1 represent the minimum and maximum possible power outputs of the farm given the power curves of the turbines, which do not take into account any wake effects.

2.5.4. Transition function

The transition function is identical to the one described by Neustroev et al. [63]. In this model, each transition to a new steady state in the FLORIS simulator involves two main steps. First, the yaw angles of the turbines are adjusted according to the actions chosen by the agent. Next, the atmospheric conditions change, leading to variations in wind flow and atmospheric measurements at the subsequent time step.

To realistically simulate these transitions, Neustroev et al. [63] employed a continuous-time stochastic process, specifically a multivariate Ornstein-Uhlenbeck (MVOU) process [105]. The MVOU process is a mean-reverting stochastic model that tends to return to long-term average values, such as a prevalent wind direction or mean wind speed. This property makes it well-suited for modelling wind dynamics [63].

Parameters for the MVOU process were fitted on the Hollandse Kust Noord (site B) dataset [60] by Neustroev et al. [63] using the method described by Meucci [59].

3

Structures in the Optimisation Surface of Static Yaw Optimisation

In this chapter, we will describe the behaviour of modes and Nash equilibria in the optimisation surface of yaw configurations through visual analysis. Each section focuses on a specific variable in the yaw optimisation problem. By identifying the layouts and wind conditions under which Nash equilibria form, we may be able reveal favourable conditions for alternating maximisation which we can exploit. Similarly, finding adversarial conditions enables us to put bounds on the applicability of alternating maximisation within yaw control.

3.1. Simulation Setup

In our analyses, the wind is assume to be coming from the west, corresponding to 270°, originating from the left side of the plots. The wind speed is set to 9 m/s. Turbines are yawed to face the wind, unless specified otherwise. The simulations are conducted using FLORIS V2.4 [65] with the default wake models and atmospheric conditions, and with the NREL 5 MW turbine [45].

3.1.1. Layout

The wind farm layouts are specified by the number of rows and columns. For instance, a single-line layout consisting of 1 row and 3 columns will be specified as a "1 × 3" layout, see Figure 3.1 for an example. The turbines are placed on a Cartesian grid and are spaced according to two main parameters: downwind spacing and crosswind spacing, both measured in rotor diameters. The downwind spacing, which is the distance between turbines along the wind direction (columns) is set to six rotor diameters. The crosswind spacing, the distance perpendicular to the wind direction (rows), is set to four rotor diameters. This configuration reflects a realistic but dense wind farm layout, according to the guidelines specified by Masters [58].

The single-line layout will be the most commonly used layout for our analyses, as it represents the worst case for wake interactions. The 1 × 3 single-line layout is the minimal size for which creating optimisation surface plots is justifiable, as the rear turbine is not optimised due to the face-the-wind policy being always optimal. This leaves us with an interpretable 2D plot of the leading two turbines, where the joint variation of turbine yaws is clear. Moreover, unless the wind direction changes from 270°, there will be no added value from adding more rows, as the wakes will interact very little across rows.

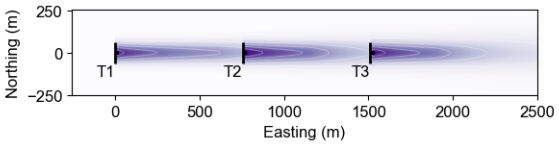

Figure 3.1: Flow field of the horizontal wind-layer at hub-height (90m) for the 1 × 3 layout under 9 m/s wind speed and 270° wind direction

3.2. Wind Direction

Wind direction determines which turbines are positioned downwind of one another. If a given turbine has a downwind neighbour, it can deflect its wake in two ways, clockwise and counter-clockwise rotation, resulting in at least two modes in the optimisation surface. Alternatively, if there is no downwind turbine, it is optimal to have no yaw-misalignment, resulting in a single mode. Consequently, wind direction significantly influences the shape of the optimisation surface, and hence impacts optimal yaw-misalignment angles, as indicated by Qian and Ishihara [80].

A clear illustration of this influence on the shape of the optimisation surface is depicted in Figure 3.2, where the sub-optimal Nash equilibrium in the lower left of the 270° plot disappears due to a slight change in wind direction. As we further increase

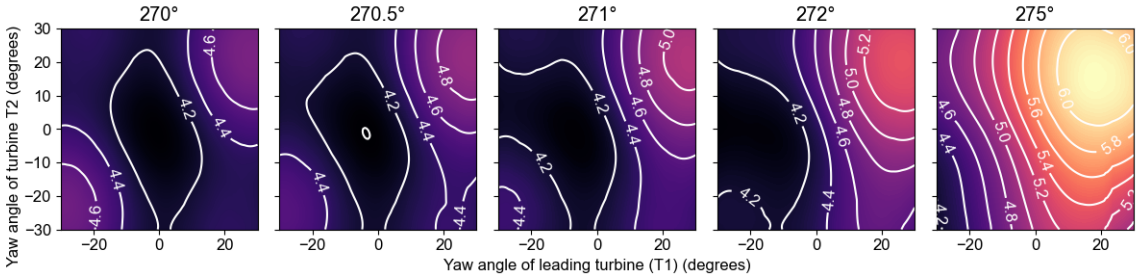

Figure 3.2: Power heat-maps (MW) for the leading two turbines of the 1 × 3 single-line layout under 9 m/s wind with varying wind directions.

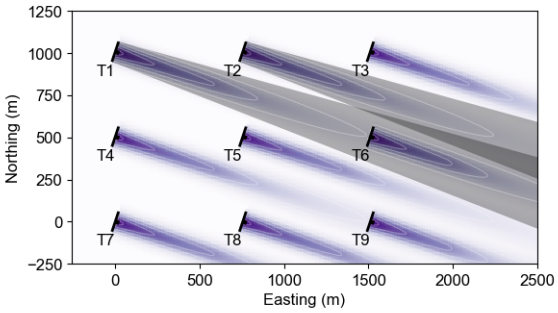

Figure 3.3: Flow field of the horizontal wind-layer at hub-height (90m) for the 3 × 3 layout under 9 m/s wind speed and 288.435° wind direction. The shaded areas denote waked areas when varying the wind direction by ±1.5°

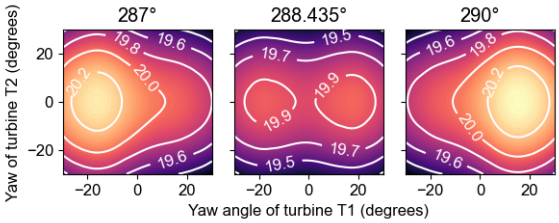

Figure 3.4: Power heat-maps (MW) for T1 and T2 in the 3 × 3 layout under 9 m/s wind speed with varying wind directions.

the change in wind direction, the local optimum diminishes and the optimal mode begins dominating the solution space, in addition to migrating towards the face-the-wind policy.

Extending our view to the 3 × 3 layout, displayed in Figure 3.3, we increase the rotation of the wind, such that wakes interact across rows. The wind direction at which cross-row waking first occurs, when turbine T1 wakes T6, can be calculated by taking the inverse tangent of the ratio between their vertical and horizontal distance, expressed in rotor diameters: $270° + \arctan \frac{4D}{12D} \approx 288.435°$. The shaded areas denote the possible waked areas when varying the wind direction 288.43±1.5°.

Figure 3.4 displays how a second local optimum emerges as the incoming wind direction approaches 288.435°. Similar to the Nash equilibria present in the 270° plot in Figure 3.4, the local optima are short-lived, only existing for a narrow band of wind directions. Interestingly, we can observe a preference towards the positive yaw angles for turbine T1 in the 288.435° plot, corresponding to a counter-clockwise rotation which deflects the wake in the clockwise direction below T6. This preference could stem from adverse affects found by mixing the wakes of T1 and T2 in the overlapping shaded areas shown in Figure 3.3.

To solidify the short-lived nature of local optima, we examine the next occurrence of cross-row waking, where turbines wake their closest diagonal neighbour. Specifically, we look at turbine T4 and T5 in the middle row, as the pair has incoming and outgoing wakes. The wind direction can be computed similarly as before: $270° + \arctan \frac{4D}{6D} \approx 303.69°$. Further rotation of the wind is not necessary due to the symmetric nature of the layout.

From Figure 3.5 we can observe how three new local optima form, as we approach the wind angle at which turbine T4 wakes it diagonal neighbour T8. The best optimum shifts from negative to positive yaw-misalignment angles, after which the remaining local optima quickly diminish again. Furthermore, we observe that there are no sub-optimal Nash equilibria formation. Specifically, each sub-optimal mode on the optimisation surface aligns with a more optimal mode. To elaborate, the worst local optimum, found in the negative yaw angles, performs worse than the mixed yaw-angle optima. These mixed optima, in turn, are inferior to the optimum located in the positive yaw angles. Each of these optima is aligned with a superior one along one dimension, which is advantageous for the alternating maximisation process.

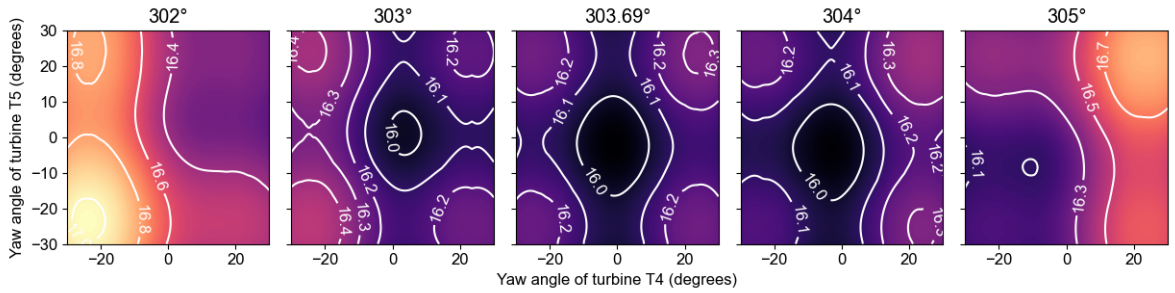

Figure 3.5: Power heat-maps (MW) for T4 and T5 in the 3 × 3 layout under 9 m/s wind speed with varying wind directions.

Based these findings, we can reasonably conclude that many Nash equilibria in yaw configurations are short-lived, existing only for narrow bands of wind directions where turbines are aligned along the freestream wind vector. Moreover, Nash equilibria formed by aligned turbines seem to predominantly exist in local optima where turbines are all positively yawed or all negatively yawed, not in mixed-yaw cases.

3.3. Wind speed

Wind speed a large effect on optimal yaw-misalignment angles [80]. The amount of power produced by a wind turbine and the amount of wake deflection for a given yaw-misalignment angle are functions of wind speed. Consequently, wind speed, similar to wind direction, may strongly influence the shape of the optimisation surface.

Figure 3.6 displays how the optimisation surface changes as wind speeds vary. At wind speeds up to 12 m/s, two Nash equilibria form where the two leading turbines are both positively or both negatively yawed. As wind speeds increase beyond 12 m/s, two local optima for mixed (positive and negative) yaw angles increase in prominence, but remain inferior to the non-mixed yaw Nash equilibria. When wind speeds reach approximately 16 m/s, the optimality gap between the optimal policy and any other policy shrinks to less than 0.04 MW. Ultimately, facing the wind becomes optimal, as wind speed in the wake is adequate for maximum power generation at downwind turbines.

3.4. Layouts

Clearly, The spatial arrangement of turbines influences how wakes interact and determines which turbines are downwind of each other. As local optima seem to predominantly form when turbines are aligned, it may be interesting to determine how layout depth (i.e., the number of aligned turbines) affects the optimisation surface.

Figure 3.7 depicts power heat-maps for single-line layouts of increasing depth. From these plots, we may infer that the optimality gap between the two Nash equilibria increases proportionately to the number of downwind turbines. Moreover, as depth increases, the modes in the surface appear to shrink and increase in prominence.

Adversarial Layouts

We define adversarial turbine layouts as layouts where the sequence of best responses will guide alternating maximisation towards a sub-optimal Nash-equilibrium. Under the assumption that optimisation starts from a face-the-wind policy, all yaw-misalignment angles at 0°, an example of such a layout can be constructed from the single-line layout by translating the second turbine down by $\frac{1}{20}$ rotor diameters, corresponding to 6.3 m. Such a translation could occur naturally in an offshore wind farm through drifting of floating wind turbines.

The implications of this modification on the alternating maximisation process are profound. While the global optimum remains in the positive yaw angles (see Figure 3.8, top-left plot), the leading turbine's initial best-response changes to a negative yaw angle, as shown the top plot of in Figure 3.9. This shift leads the middle turbine to also adopt a negative yaw angle as its best response. Consequently, the procedure converges to the sub-optimal Nash equilibrium.

Interestingly, the observed effect is sensitive to the magnitude of the translation. Increasing the translation to $\frac{3}{20}$ turbine diameters shifts the global optimum to the negative yaw angles, as depicted in the top-right plot of Figure 3.8. As a result, the new convergence path becomes optimal. Furthermore, the previous global optimum becomes a local optimum, but not a Nash equilibrium, indicating that their existence is sensitive to such translations.

Expanding our analysis, we investigate translations on a 1 × 4 layout. The bottom row of Figure 3.8 depicts the resulting heat-maps. The bottom

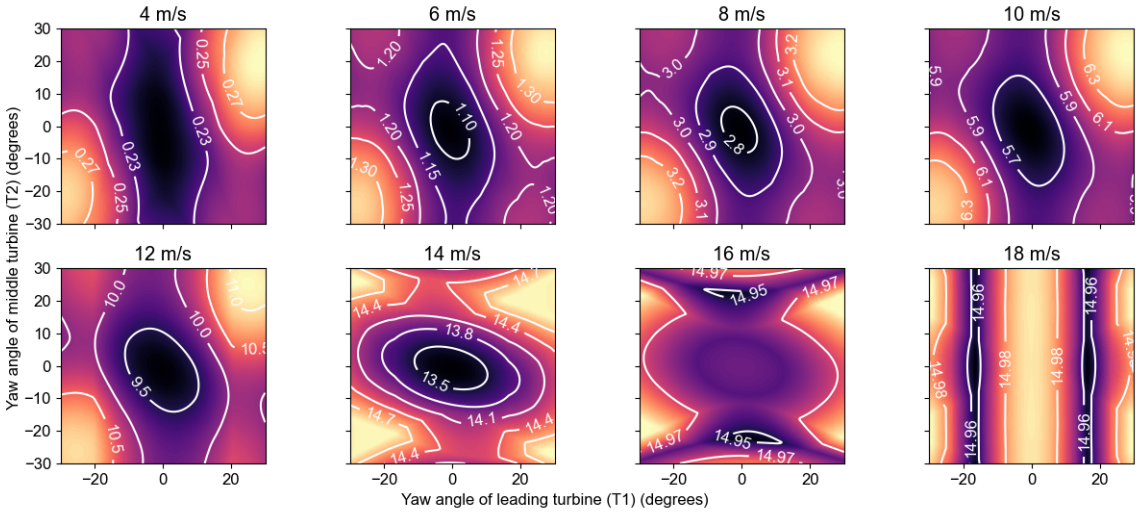

Figure 3.6: Power heat-maps (MW) for the leading two turbines of the 1 × 3 single-line layout under 270° wind direction with varying wind speeds. **Note:** the colour map range differs between plots.

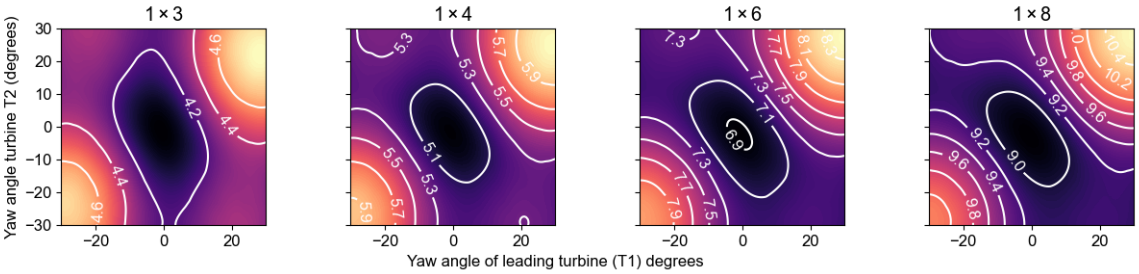

Figure 3.7: Power heat-maps (MW) for the leading two turbines of various single-line layouts under 9 m/s wind speed and 270° wind direction. **Note:** the colour map range differs between plots.

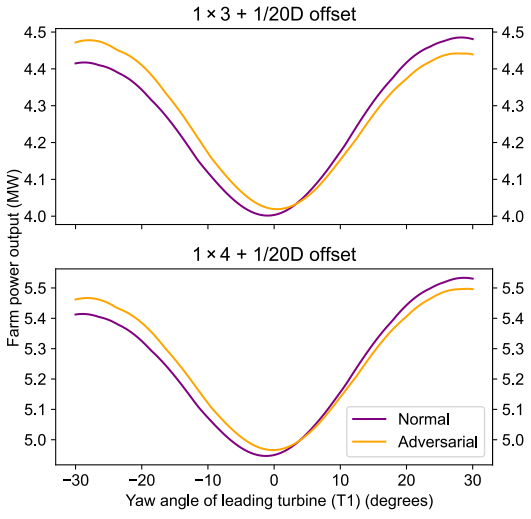

Figure 3.9: Best response of the leading turbine in the 1 × 3 and 1 × 4 layouts with $\frac{1}{20}$D adversarial variants, under 9 m/s wind speed, 270° wind direction, and all other turbines facing the wind.

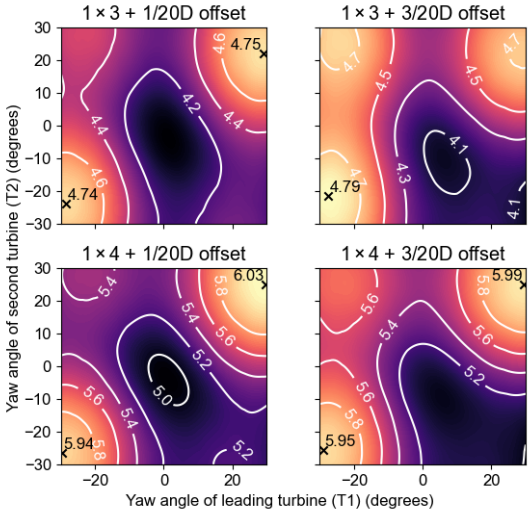

Figure 3.8: Power heat-maps (MW) including Nash equilibria for the leading two turbines of various adversarial layouts under 9 m/s wind speed and 270° wind direction. **Note:** the colour map range differs between the 1 × 3 and 1 × 4 layouts.

plot in Figure 3.9 displays how the addition of a turbine mitigates the adverse effects of translation on the best-response. Only when the translation is increased to $\frac{3}{20}$ diameters does the adverse effect occur, which is evident in the bottom-right heat-map of Figure 3.8. This suggests that the shape of the optimisation surface of deeper layouts is less sensitive to adverse effects from translations.

To sum up, adversarial structures can occur in grid layouts through small translations of turbines, although the effects transient since the Nash equilibria in static yaw optimisation exist only for narrow bands of wind directions. Furthermore, the small optimality gap between Nash equilibria in the investigated layouts indicates that alternating maximisation may be effectively applied in practice.

3.5. Yaw misalignment

In this section we will investigate how the yaws of upwind and downwind turbines influence the best-responses of other turbines. As is evident from the shapes of the overall heat-map plots, the optimal yaw-angles of turbines are correlated. For example, when turbines are aligned along the wind vector, such as in Figure 3.7, and the leading turbine holds a positive yaw angle, the best-response of the middle turbine is also a positive yaw angle. Similarly, a negative yaw angle held by the leading turbine results in a negative yaw-angle as best-response. For this reason, we investigate the behaviour of best-responses when varying yaw angles.

To this end, we use a 1x8 single-line layout with all turbines initialised to face the wind. Turbines are numbered from upwind to downwind, that is, the lead turbine is T1 and the rear turbine is T8. However, the rear turbine is excluded from the analysis as its best response is always to face the wind.

Figure 3.10 illustrates the optimal yaw responses of turbines T2–T7 when varying the yaw angle of turbine T1. The data indicates that best responses generally align in the same rotational direction — either clockwise or counter-clockwise — as the yawing turbine T1. This alignment effect is more pronounced in turbines closer to T1. Conversely, turbines located further away show a stronger preference for positive yaw angles (counter-clockwise rotation) in their optimal responses.

Interestingly, the magnitude of the optimal responses of some turbines inversely correlates with the yaw angle of T1. This may be attributed to the increased wind speed resulting from T1's significant deflection of the wind. A higher wind speed necessitates a smaller yaw angle to achieve the same deflection compared to lower wind speeds.

Figure 3.11 presents the results of a similar experiment, but with the downwind turbine T7 being yawed. The observations reveal that the alignment effect remains, although it is weaker compared to when the upwind turbine T1 is yawed. Notably, turbines T1–T4 do not exhibit the same following behaviour as the other turbines; they maintain a positive yaw angle as their best-response. This likely occurs because the leading turbines can deflect more wind with a positive yaw angle. For T1–T4, there appears to be more benefit in maintaining a positive yaw-angle than in aligning with their the downwind neighbour to join their wakes.

In summary, the experiments demonstrate that the yaw responses of turbines T2 through T7 generally align with the yaw direction of the leading turbine, with this effect being more pronounced in turbines closer to the yawing turbine. However, turbines further away show a stronger preference for positive yaw angles. When the downwind turbine T7 is yawed, the following effect is weaker, and the leading turbines T1–T4 maintain a positive yaw angle, indicating greater benefits from this strategy

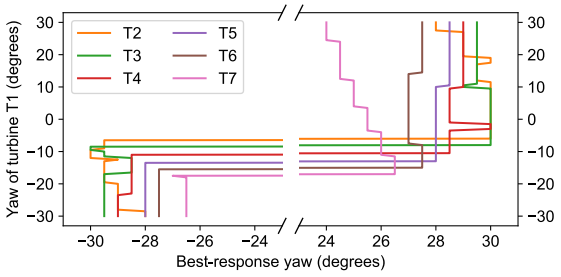

Figure 3.10: Best responses of turbines T2–T7 in response to yawing of turbine T1 in a 1 × 8 layout under 9 m/s wind speed and 270° wind direction.

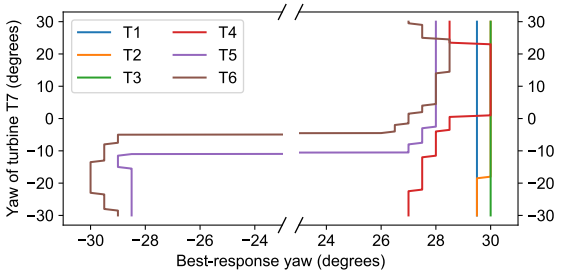

Figure 3.11: Best responses of turbines T1–T6 in response to yawing of turbine T7 in a 1 × 8 layout under 9 m/s wind speed and 270° wind direction.

3.6. Conclusion

In conclusion, the findings of this chapter illustrate the intricate effects of wind direction, wind speed, turbine layouts, and yaw misalignment on the shape of the optimisation surface in static yaw optimisation.

Nash equilibria in yaw configurations are shown to be transient, typically existing for only narrow bands of wind directions where turbines align along the freestream wind vector. These equilibria tend to form local optima where all turbines are either positively or negatively yawed, rather than in mixed-yaw configurations. These findings indicate that alternating maximisation may be a fruitful approach for many wind directions. In addition, in cases with sub-optimal Nash-equilibria, there are only two candidate regions to explore: large positive or negative yaw angles.

Wind speed significantly affects optimal yaw-misalignment angles. At lower wind speeds (up to 12 m/s), Nash equilibria emerge where turbines are all yawed either positively or negatively. As wind speeds increase beyond 12 m/s, mixed yaw angles gain prominence but remain sub-optimal compared to non-mixed configurations. At approximately 16 m/s the optimality gap between the different policies narrows, making all strategies nearly equivalent in effectiveness.

Naturally, wind speed also affects the power output of the farm as a whole. For that reason, wind speed should be taken into account when exploring in time-varying wind conditions. Alternatively, too few samples may result in a misleading reward signal with a false maximum during the high wind-speed samples.

The layout depth also plays a role in the shape of the optimisation surface. As the number of aligned turbines increases, the optimality gap between Nash equilibria grows, and the modes in the optimisation surface decrease in width but grow in prominence. However, the mode of the global optimum remains comparatively wider. Consequently, the direction in which to optimise becomes clearer with layout depth.

Adversarial structures, where best-response problems mislead the convergence path to sub-optimal Nash equilibria, can occur in grid layouts through small translations of turbines. However, because the Nash equilibria are dependent on the wind directions, they remain transient. Furthermore, the small optimality gap between Nash equilibria in the investigated layouts indicates that alternating maximisation may be effectively applied in practice.

Lastly, the experiments on yaw misalignment reveal that turbines generally align their yaw responses with the leading turbine's yaw direction. This effect diminishes with distance, as turbines further away prefer positive yaw angles. Yawing the downwind turbine results in a weaker alignment effect, with leading turbines maintaining a positive yaw angle to maximise wind deflection and power generation benefits.

Under the assumption that optimal yaw configurations contain yaw-misalignment angles pointing in roughly the same direction, the aligning behaviour of turbines with their neighbours puts an emphasis on the initial set of optimisations. If the correct direction is identified early on, the likelihood of converging to an optimal policy may be significantly improved.

Overall, these findings show that the shape of the optimisation surface for static yaw optimisation is generally favourable towards the application alternating maximisation.

4

Static Yaw Optimisation via Alternating Maximisation

Static yaw optimisation aims to find the best turbine yaw-misalignments per specific wind condition, and forms an integral part of certain dynamic yaw control strategies. This chapter explores how alternating maximisation can be effectively applied to static yaw optimisation, and investigates its relevance both as a promising strategy in itself and for insights it may provide into the application of alternating maximisation to dynamic yaw control.

Our findings from the previous chapter suggest that alternating maximisation could be a productive approach for static yaw optimisation in many wind conditions. The transient nature of Nash equilibria in yaw configurations, the limited amount of candidate regions to explore, and the aligning behaviour of turbines support its applicability.

Understanding the efficacy of alternating maximisation in static yaw optimisation also has implications for dynamic yaw control. By evaluating it in static scenarios, we can identify specific challenges when applying this method to dynamic conditions, such as the impact of a limited rotation speed on the efficacy of alternating maximisation.

To provide a comprehensive understanding of how to effectively apply alternating maximisation to static yaw optimisation, we refine the primary research question (Q2) of this chapter *"How can alternating maximisation method be effectively applied to static yaw optimisation, and what factors influence its performance and optimality of the resulting Nash equilibria?"* into the following sub-research questions:

Q2.1 What approach for solving the best-response problem converges to better optima in static yaw optimisation?

Q2.2 How does the initial yaw configuration affect the Nash equilibria found through alternating maximisation?

Q2.3 How does the optimisation order affect the Nash equilibria found through alternating maximisation?

This chapter begins by outlining the basic implementation of alternating maximisation for static yaw optimisation. It then delves into optimally solving the best-response sub-problems. Next, the assumptions underpinning the standard implementation will be questioned. Specifically, the initial yaw configuration and the order of optimisation. Both may have a profound effect on the convergence path of alternating maximisation, as we have demonstrated in the previous chapter that the best-responses of turbines are dependent on the yaws of their neighbours.

4.1. Problem Statement

Static yaw optimisation, being a single-state problem with only one wind condition, can be effectively modelled as a combinatorial multi-armed bandit [15] problem with continuous actions. In this formulation, the agent must simultaneously choose N continuous actions, one for each turbine yaw-misalignment angle. After selecting the yaw angles, the agent receives a reward based on the total power output of the wind farm. The setup for the combinatorial multi-armed bandit problem can be formalised as follows:

- Each turbine $i \in \{1, 2 \dots, N\}$ represents a component of the joint arm in the combinatorial multi-armed bandit problem.
- $A : [-30, 30]^N$ is the action space, where $a = (a_1, a_2, \dots, a_N) \in A$ represent the chosen yaw angles.
- $R : A \to \mathbb{R}$ is the reward function, where $R(a)$ is the steady-state farm output for the yaw angles represented by the joint action a.

4.2. Alternating Maximisation

We begin our exploration of static yaw optimisation with a naive implementation of alternating maximisation, which we name Alternating Yaw Optimisation. This initial algorithm is inspired by the Boolean yaw angle method by Stanley et al. [96] and the Serial-Refine method by Fleming et al. [32], both of which are state-of-the-art iterative techniques designed for static yaw optimisation.

Following the methodology of Stanley et al. [96] and Fleming et al. [32], the procedure starts by sorting the turbines in upwind to downwind order, and initialising all turbines to face the wind. The core of procedure involves sequentially solving best-response problems, where the yaw angle of a turbine is optimised, in the established order. This process constitutes one iteration alternating maximisation, and is repeated until convergence.

To solve the best-response problems in each step of the iterative process, we employ the sequential least squared programming (SLSQP) [50] method, a gradient-based algorithm and the default for FLORIS V2.4 [65]. The accompanying pseudo code for alternating yaw optimisation can be found below in Algorithm 1.

Algorithm 1: Alternating yaw optimisation

Input: Turbines $D = \{1, 2, ..., N\}$ sorted by optimisation order
Input: Initial yaws $Yaws = \{yaw_i \mid \forall i \in D\}$
Input: optimise$(Yaws, turbine_id) \rightarrow ([\text{min yaw}, \text{max yaw}]^n, \mathbb{Z}^+)$ turbine optimisation function
Output: Optimised yaws *Yaws*

```
function AlternatingYawOptimisation(atol)
    obj ← 0
    prev ← 0
    while obj − prev > atol do
        prev ← obj
        forall i ∈ D do
            obj, yaw ← optimise(Yaws, i)
            set Yaws_i ← yaw
    return Yaws
```

4.2.1. Experiment

In this experiment, we compare the performance and efficiency of alternating yaw optimisation against two state-of-the-art optimisation methods.

Experimental Setup

For our experimental setup, we use the NREL 5MW wind turbine [45], simulated using FLORIS V2.4 [65]. All turbines are homogeneous and have their yaw boundaries set at $[-30°, 30°]$. The wind speed for all experiments is set at 9 m/s, which should allow wakes to propagate sufficiently far, while being moderate enough to not make the face-the-wind policy optimal when turbines are aligned with the wind. Additionally, we vary the wind direction ±33° around the prevailing wind direction 270° (west) with 3° steps.

We conduct experiments on various spatial configurations, starting with a basic 3×3 layout. Then, we introduce more turbines to assess the impact on convergence properties by expanding to 5×5 and 7×7 layouts. To isolate the effects of simply adding more turbines from the more complex wake interactions that occur in deeper layouts, we also test alternating yaw optimisation on a wide 10×2 layout and a deep 2×10 layout.

To evaluate the effectiveness of alternating yaw optimisation, we will compare it against two state-of-the-art methods. Our primary comparison will be with the sequential least squares programming (SLSQP) method used by FLORIS [65], as it employs the same gradient-based optimiser as our approach. However, like alternating maximisation, FLORIS is known to get stuck in local optima due to its gradient-based nature.

To address this limitation, we will also compare our method against differential evolution [97], an optimisation technique that is expected to perform better due to its ability to explore the search space more thoroughly. Differential evolution does not rely on gradient information, which ideally allows it to avoid getting trapped in local optima and potentially find more global solutions. Moreover, differential evolution is a strong global optimisation algorithm that sees wide use across multiple scientific domains [7], and it is well suited for high-dimensional and highly multi-modal problems [79].

Exhaustively searching for the optimal solution is not feasible because the search space is too large and the function evaluations are too computationally expensive. For example, using a relatively small farm layout with 9 turbines, a search resolution of 1°, and yaw boundaries of $[-30°, 30°]$, exhaustive search already requires $61^9 \approx 10^{16}$ function evaluations. Therefore, differential evolution serves as a practical alternative that balances exploration and exploitation of the search space.

By comparing alternating maximisation with sequential least squares programming (FLORIS) and differential evolution, we aim to understand whether our method offers any advantages over traditional gradient-based approaches and to see

how it measures up against a more exploratory optimisation technique. The performance of each method will be measured against the baseline face-the-wind strategy. Hyperparameters for each method can be found in Appendix A.

Results

The results of the experiment are presented in Table 4.1 and Table 4.2. Surprisingly, alternating yaw optimisation consistently outperforms the other methods across multiple layouts, with the exception of the 3 × 3 layout, and multiple metrics, including mean, median, 5th percentile, and 1st percentile scores. This indicates that alternating yaw optimisation is both effective and robust. Nevertheless, we would have expected differential evolution to perform better due to its strong search capabilities. However, the performance of all methods is remarkably close, with differences never exceeding 1 percentage point. These close scores may imply that the potential for further improvement is limited.

Interestingly, differential evolution also has the worst outliers, as shown by the 1st percentile scores, which could be attributed to is stochastic nature. Specifically, it appears that it has gotten trapped in a local optimum which is worse than the face-the-wind strategy, meaning its population initialisation did not cover the face-the-wind strategy and was unable to find it through exploration.

Furthermore, differential evolution exhibited the lowest standard error across all layouts. However, the differences in standard error between the methods were small and appeared more related to the layout configuration than the optimisation method used. This is logical, as the performance gap for the face-the-wind strategy is expected to scale with layout depth. Specifically, in wind conditions where few turbines are aligned along the wind vector, the face-the-wind strategy is more likely to be optimal, whereas in conditions where turbines are positioned downwind of each other, the strategy may be less effective.

While sequential least squares programming performed the worst across the board in terms of relative performance, it did so using the least amount of function evaluations. Conversely, alternating maximisation with sequential least squares programming used by far the most function evaluations, requiring an order more than other methods. This finding is unexpected, as we anticipated that the gradient-based method would need fewer samples due to the one-dimensional nature of the alternating maximisation sub-problems. This discrepancy suggests that sequential least-squares programming struggles to identify maxima in the best-response problems of alternating maximisation.

Layout	Mean	SE	Median	P5	P1
	Alternating yaw optimisation (SLSQP)				
3 × 3	105.15	**1.28**	102.09	100.02	100.00
5 × 5	**108.49**	2.15	102.73	**100.19**	**100.13**
7 × 7	**109.95**	**2.58**	**103.24**	**100.25**	**100.15**
wide	**104.02**	1.00	101.20	**100.00**	**100.00**
deep	**108.69**	**2.76**	102.89	**100.10**	**100.01**
	Sequential least squares programming				
3 × 3	105.55	1.43	102.06	100.00	100.00
5 × 5	107.99	**2.13**	102.69	100.06	100.00
7 × 7	109.80	**2.58**	103.13	100.06	100.02
wide	103.93	0.99	101.20	**100.00**	**100.00**
deep	108.51	2.77	102.65	100.00	100.00
	Differential evolution				
3 × 3	**105.78**	1.44	**102.16**	**100.06**	**100.04**
5 × 5	108.22	2.14	**102.78**	100.07	99.95
7 × 7	109.93	2.60	**103.24**	100.16	97.34
wide	103.93	**0.98**	**101.30**	**100.00**	**100.00**
deep	108.65	2.77	**102.97**	100.07	98.67

Table 4.1: Relative performance in percentage to the baseline (facing the wind) in the alternating yaw optimisation experiment.

Layout	Mean	SE	Median	P95	P99
	Alternating yaw optimisation (SLSQP)				
3 × 3	14013	567	15257	18143	18234
5 × 5	41163	953	41561	43554	54973
7 × 7	87844	4242	82482	126417	164833
wide	28917	1194	30741	34727	34731
deep	34816	1862	34602	50475	67311
	Sequential least squares programming				
3 × 3	**726**	83	**964**	**981**	**990**
5 × 5	**1313**	**141**	**1730**	**1770**	**1770**
7 × 7	**2436**	**183**	**2916**	**2961**	**2970**
wide	**1273**	**114**	**1520**	**1540**	**1540**
deep	**1107**	**130**	**1488**	**1522**	**1540**
	Differential evolution				
3 × 3	1576	**79**	1510	2260	2420
5 × 5	5364	820	4708	13812	16812
7 × 7	14240	3666	6810	45730	68560
wide	2843	237	2526	4968	5768
deep	3668	446	2947	7810	9810

Table 4.2: Number of function evaluations rounded to nearest integer in the alternating yaw optimisation experiment.

In conclusion, alternating yaw optimisation has proven to be an effective method for static yaw optimisation, consistently delivering superior performance across multiple metrics and layouts. The method's robustness is evident, as neither the addition of more turbines, nor the increased complexity of wake interactions in deeper layouts adversely impacts the convergence properties. In contrast to the comparatively good gain in power output of alternating yaw optimisation with sequential least squares programming, its inefficiency highlights a critical area for improvement. Exploring alternative gradient-based methods or different paradigms may yield better results in terms of both performance and computational efficiency. However, it is important to note that our conclusions are limited by the inability to perform an exhaustive search for the optimal solution due to computational constraints. Despite the lack of a conclusive global optimum, the superior performance of alternating yaw optimisation over other methods is promising, suggesting that the shape of the optimisation surface is lends itself to optimisation via alternating maximisation.

4.3. Solving the Best-Response Problem

The results for alternating yaw optimisation using sequential least squares programming demonstrate its effectiveness in static yaw optimisation. However, because the method relies on gradient-based solutions for best-response problems, it becomes trapped in local optima, resulting in suboptimal outcomes or an extended convergence process which requires an exorbitant amount of function evaluations.

To mitigate these limitations, we apply a sampling-based method, which has the potential to explore a broader range of options and avoid local optima. Since function evaluations are expensive, both in simulation and real-life wind farms, it is crucial to adopt an efficient sampling strategy.

To this end, we employ a coarse-to-fine search, which is a sampling-based technique commonly applied to problems with large search spaces [75, 53]. While the scale of best-response problems is not large by itself, expensive function evaluations increase its effective size. The Coarse-to-Fine approach aims to exclude impractical areas of the initial search space and focus on exploiting the promising regions [75]. By first sampling coarsely, promising regions can be identified and their neighbourhoods explored through fine sampling.

In our implementation of coarse-to-fine sampling for static yaw optimisation, we sample a set of z equidistant points as our coarse yaws. Next, we select the modes from this noisy signal as our candidate solutions. We opt for modes and not solely the maximum, as the best-response problems can be multi-modal and we are not guaranteed to sample the maximum of each mode. Figure 4.1 displays such a scenario where a false maximum may be selected. Additionally, we add the current yaw angle to our candidate solutions to prevent regression due to sampling errors. Next, we explore the candidate solution neighbourhoods using search radius r at a resolution of Δy and select the best yaw. The pseudo code for the coarse-to-find sampling method is displayed below in Algorithm 2.

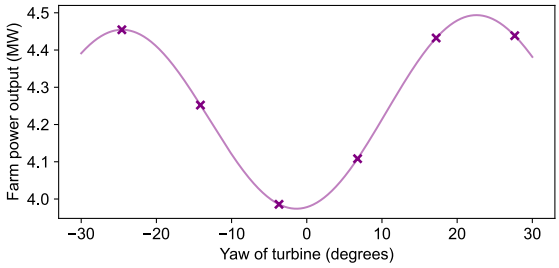

Figure 4.1: A false maximum due to coarse sampling.

Algorithm 2: Coarse-to-fine search

Input: $yaws_{\text{coarse}} \in [y_{\min},\ y_{\max}]^z$
z coarse equidistantly sampled yaws
Input: $r \in \mathbb{R}$ candidate search radius
Input: $\Delta y \in \mathbb{R}$ fine sampling resolution
Input: $\text{eval}(Yaws) \to \mathbb{R}$
objective evaluation function
Output: $(obj,\ yaw_{\text{best}}) \in \mathbb{R} \times [y_{\min},\ y_{\max}]$
function CoarseToFineSearch($Yaws, i$)
  $yaw_{\text{prev}} \leftarrow Yaws_i$
  $obj_{\text{coarse}} = [\,]$
  forall $y \in yaws_{\text{coarse}}$ **do**
    set $Yaws_i \leftarrow y$
    push $(obj_{\text{coarse}}, \text{eval}(Yaws))$
  $M \leftarrow \text{modes}(obj_{\text{coarse}}) \cup \{yaw_{\text{prev}}\}$
  $yaw_{\text{candidates}} \leftarrow$
  $\bigcup_{y \in M} \left\{ y - r + \Delta y \cdot k \mid k \in \mathbb{Z}, 0 \le k \le \frac{2r}{\Delta y} \right\}$
  forall $y \in yaw_{\text{candidates}}$ **do**
    set $Yaws_i \leftarrow y$
    if $\text{eval}(Yaws) > obj$ **then**
      $yaw_{\text{best}} \leftarrow y$
      $obj \leftarrow \text{eval}(Yaws)$
  set $Yaws_i \leftarrow yaw_{\text{best}}$
  return obj, yaw_{best}

Layout	Mean	SE	Median	P95	P99
	Alternating yaw optimisation (SLSQP)				
3 × 3	14013	567	15257	18143	18234
5 × 5	41163	953	41561	43554	54973
7 × 7	87844	4242	82482	126417	164833
wide	28917	1194	30741	34727	34731
deep	34816	1862	34602	50475	67311
	Alternating yaw optimisation (coarse-to-fine)				
3 × 3	**975**	**15**	**941**	**1126**	**1130**
5 × 5	**3404**	**134**	**3912**	**3929**	**4256**
7 × 7	**7424**	**372**	**7692**	**10252**	**10261**
wide	**2163**	**42**	**2089**	**2647**	**2710**
deep	**2436**	**96**	**2095**	**3148**	**3151**

Table 4.4: Number of function evaluations rounded to nearest integer in the best-response solving experiment.

Layout	Mean	SE	Median	P5	P1
	Alternating yaw optimisation (SLSQP)				
3 × 3	105.15	**1.28**	102.09	100.02	100.00
5 × 5	108.49	**2.15**	102.73	100.19	100.13
7 × 7	109.95	**2.58**	103.24	100.25	100.15
wide	104.02	**1.00**	101.20	**100.00**	**100.00**
deep	108.69	**2.76**	102.89	100.10	100.01
	Alternating yaw optimisation (coarse-to-fine)				
3 × 3	**105.79**	1.44	**102.16**	**100.06**	**100.05**
5 × 5	**108.61**	**2.15**	**102.78**	**100.21**	**100.19**
7 × 7	**110.30**	2.62	**103.27**	**100.33**	**100.21**
wide	**104.05**	**1.00**	**101.30**	**100.00**	**100.00**
deep	**108.80**	**2.76**	**102.97**	**100.11**	**100.07**

Table 4.3: Relative performance in percentage to the baseline (facing the wind) in the best-response solving experiment.

4.3.1. Experiment

In this experiment, we compare the effectiveness of gradient-based and sampling-based approaches in solving the best-response problem within the alternating maximisation framework for static yaw optimisation. We aim to assess their overall performance using the same experimental setup as previously described. For the coarse-to-fine sampling-based method, we use the following parameters: a coarse sampling resolution of 3°, which results in $z = (y_{\max} - y_{\min})/3° + 1 = 21$, a candidate search radius r of 1.5°, and a fine sampling resolution Δy of 0.1°.

From Table 4.3 it can be seen that the sampling-based approach achieves superior performance across all layouts. More importantly, the performance gap between alternating yaw optimisation and the methods from the previous experiment is closed. This suggests that gradient-based methods do get stuck in the local optima of best-response problems, though, the performance difference between the two methods is slim.

Additionally, Table 4.4 displays the vastly reduced number of function evaluations required when using coarse-to-fine sampling. The efficiency of the method bring the number of function evaluations down an order of magnitude, sitting between firmly between differential evolution and sequential least squares programming.

Based on these findings, we may conclude that a sampling-based approach is most suited, both in terms of performance and efficiency, for solving best-response problems in static yaw optimisation.

4.4. Yaw-Angle Initialisation

Current iterative methods for static yaw optimisation, such as those discussed in Stanley et al. [96] and Fleming et al. [32], initialise turbine yaw angles to face the wind. However, neither method provides a detailed rationale for this specific design choice. Given the influence of turbine yaw angles on the best responses of neighbouring turbines, it is important to explore alternative initialisation strategies that could potentially affect optimisation performance.

Our investigation considers both positive and negative initialisation of turbine yaw angles. This exploration is motivated the findings in Chapter 3, which indicate that positive and negative yaw angles are generally good candidate solutions for yaw optimisation. The rationale behind this approach is rooted in the possible influence of initialisation on convergence paths.

When turbines are initialised with positive yaw angles, the configuration is theoretically closer to a global optimum, assuming that such an optimum exists in the positive yaw region. Conversely, negative initialisation is expected to be nearer to a sub-optimal Nash equilibrium. Due to the influence of initial yaw configurations on subsequent best responses, these initial states are likely to attract the solutions towards their respective Nash equilibria.

4.4.1. Experiment

For each yaw-angle initialisation strategy, we use the same 3 × 3 turbine layout with the previously described experimental setup, and apply both alternating yaw optimisation methods for solving the best-response problems. We measure the farm power output and sample efficiency to compare the effectiveness of each strategy.

Yaw init.	Mean	SE	Median	P5	P1
Alternating yaw optimisation (SLSQP)					
Standard	105.15	**1.28**	102.09	**100.02**	**100.00**
Positive	**105.60**	1.43	**102.14**	**100.02**	99.95
Negative	**105.60**	1.41	102.06	99.96	99.93
Alternating yaw optimisation (coarse-to-fine)					
Standard	**105.79**	1.44	**102.16**	**100.06**	**100.05**
Positive	**105.79**	1.44	**102.16**	**100.06**	**100.05**
Negative	105.71	**1.40**	**102.16**	**100.06**	**100.05**

Table 4.5: Relative performance in percentage to the baseline (facing the wind) in the yaw-angle initialisation experiment. **Note:** the best performing initialisation-strategy is highlighted separately for each method.

Layout	Mean	SE	Median	P95	P99
Alternating yaw optimisation (SLSQP)					
Standard	14013	567	15257	18143	18234
Positive	14014	**267**	14086	**15178**	**15180**
Negative	**10676**	644	**10846**	15207	17068
Alternating yaw optimisation (coarse-to-fine)					
Standard	**975**	**15**	**941**	**1126**	**1130**
Positive	1391	64	1416	1727	2290
Negative	1344	54	1415	1695	1704

Table 4.6: Number of function evaluations rounded to nearest integer in the yaw-angle initialisation experiment. **Note:** the best performing initialisation-strategy is highlighted separately for each method.

The results are presented in Table 4.5 and Table 4.6 and reveal distinct differences in performance based on the chosen initialisation strategy, with notable variations between the sampling-based and gradient-based approaches.

For the sampling-based approach, both standard face-the-wind and positive initialisation result in identical performance. This suggests that both initialisations are drawn to the same optima. Conversely, negative initialisation leads to worse performance than either approach, indicating that starting near a sub-optimal Nash equilibrium negatively impacts the overall optimisation outcome.

The gradient-based approach exhibits a different behaviour. Both positive and negative initialisations perform better on average than the standard face-the-wind initialisation, yet, both are worse than any initialisation for the sampling-based approach. However, the data also reveal that these initialisations have outliers that perform worse than the face-the-wind strategy, as indicated by the 1st percentile. We speculate these outliers arise from scenarios where the face-the-wind strategy is optimal. In such cases, starting with a positive or negative yaw angle places the turbines far from the optimal face-the-wind configuration. As a result, the gradient-based method, which is sensitive to initial conditions and local optima, may get stuck in a local optimum before reaching the optimal face-the-wind strategy, thus resulting in worse performance than the face-the-wind strategy.

This discrepancy between the sampling and gradient-based approaches extends to the efficiency of the initialisation strategies. For the sampling-based method, the face-the-wind initialisation is most efficient, with both positive and negative initialisations seeing a significant increase in the number of function evaluations. On the contrary, for the gradient-based method, the face-the-wind initialisation is least efficient. Unexpectedly, the negative initialisation is on average more sample efficient than the positive initialisation, while having worse outliers.

In conclusion, the results indicate that a face-the-wind initialisation strategy may be preferred, given that the best-response problems are solved optimally. The positive initialisation strategy yields identical performance, but is less sample efficient. Since wind farms are currently operated with a face-the-wind strategy, this approach may be more conducive to actual farm operations as the effects of large positive yaw-misalignments are less studied.

4.5. Optimisation Order

Next, we tackle the order in which turbines are optimised. In previous approaches [96, 32], turbines are optimised in upwind-to-downwind order without providing a rationale for the design choice. Moreover, the order in which turbines are optimised may influence the performance of alternating maximisation for static yaw optimisation. Given the interdependence of turbine yaw angles, the optimisation order may affect both the convergence speed and the quality of the final solution.

In our investigation, we explore different optimisation orders to determine their impact on the optimisation process. We start with the standard upwind-to-downwind order, where turbines are optimised sequentially from the front of the wind farm to the back. This approach is based on the logical assumption that upwind turbines, which experience less wake effect, should be optimised first to minimise the downstream impact of wakes.

Layout	Mean	SE	Median	P95	P99
Alternating yaw optimisation (SLSQP)					
Standard	14013	**567**	15257	18143	**18234**
Reverse	**11915**	1073	**10014**	**15252**	32282
Cross	12111	910	**10014**	21286	24066
Alternating yaw optimisation (coarse-to-fine)					
Standard	**975**	**15**	**941**	**1126**	**1130**
Reverse	1182	63	944	1719	1719
Cross	1068	47	942	1506	1719

Table 4.8: Number of function evaluations rounded to nearest integer in the optimisation order experiment. **Note:** the best performing order is highlighted separately for each method.

Order	Mean	SE	Median	P5	P1
Alternating yaw optimisation (SLSQP)					
Standard	**105.15**	1.28	**102.09**	**100.02**	**100.00**
Reverse	104.98	**1.24**	102.05	**100.02**	**100.00**
Cross	104.99	**1.24**	102.07	**100.02**	**100.00**
Alternating yaw optimisation (coarse-to-fine)					
Standard	**105.79**	**1.44**	**102.16**	**100.06**	**100.05**
Reverse	105.78	**1.44**	**102.16**	**100.06**	**100.05**
Cross	**105.79**	**1.44**	**102.16**	**100.06**	**100.05**

Table 4.7: Relative performance in percentage to the baseline (facing the wind) in the optimisation order experiment. **Note:** the best performing order is highlighted separately for each method.

We also examine the reverse order, from downwind to upwind, to assess whether starting with turbines that are more affected by wake interactions can provide any advantages. Additionally, we test a cross-wind order, optimising turbines column by column, perpendicular to the wind direction, to evaluate how lateral wake interactions might influence the results.

4.5.1. Experiment

For each optimisation order, we use the same 3 × 3 turbine layout, use the same experimental setup as in previous experiments, and apply both alternating yaw optimisation approaches for solving the best-response problems.

The results are presented in Table 4.7 and Table 4.8. The gradient-based approach exhibited variability in performance depending on the optimisation order. Specifically, the downwind-to-upwind (reverse) order and the cross-wind order resulted in worse outcomes. These orderings likely lead to premature convergence to local optima, as the gradient-based method relies on local information and can be influenced by the sequence of optimisation. Conversely, for the sampling-based approach, we found no difference in optimality between the different optimisation orders. This suggests that the sampling-based method's ability to explore the search space broadly compensates for any potential drawbacks associated with the sequence in which turbines are optimised.

Regarding efficiency, we can observe that the standard upwind-to-downwind ordering performs best when using the coarse-to-fine optimisation method, yielding about a 10% lead in mean performance over the other orderings. By contrast, using a gradient-based approach, the results are flipped, with both reverse and cross orderings being most efficient.

These findings suggest that the specific optimisation order is not of importance for the optimality of alternating yaw optimisation, as long as the best-response problems are solved optimally. However, the order does matter for efficiency. For these reasons, we suggest using the upwind-to-downwind order, as it is most optimal and sample efficient, and minimises the downstream impact of wakes early on in the optimisation process leading to extra energy capture during optimisation.

4.6. Conclusion

This chapter has explored alternating yaw optimisation for static yaw control in wind farms, examining its performance through a series of experiments involving different initialisation strategies, best-response problem-solving methods, and optimisation orders.

Alternating yaw optimisation consistently outperformed other methods across various layouts and metrics, demonstrating its robustness and effectiveness. The sampling-based approach for solving best-response problems proved to be the most optimal, robust, and sample efficient, closing the remaining performance gaps with differential evolution. By contrast, the gradient-based approach suffered from convergence to local optima in the best-response problems and required an order of magnitude more samples in every experiment, leading to worse performance.

Moreover, the face-the-wind initialisation strategy is preferred, given that the best-response problems are solved optimally. The positive initialisation strategy yields identical performance, but is less sample efficient. Since wind farms are currently operated with a face-the-wind strategy, this approach may be more conducive to actual farm

operations as the effects of large positive yaw-misalignments are less studied.

Furthermore, the results from our experiments suggest that the specific optimisation order is not of importance for the optimality of alternating yaw optimisation, as long as the best-response problems are solved optimally. However, the upwind-to-downwind order was more efficient by a margin of approximately 10%. For these reasons, we suggest using the upwind-to-downwind order, in addition to minimising the downstream impact of wakes early on in the optimisation process leading to extra energy capture during optimisation.

In summary, alternating yaw optimisation with a sampling-based approach is the current most effective method for static yaw optimisation via alternating maximisation. Moreover, alternating maximisation with sampling is robust to design choices concerning the yaw-initialisation strategy, and optimisation order. Positive initialisation offers marginal gains, but the standard face-the-wind strategy is nearly as effective and aligns with current practices. Optimisation order has no impact on optimality given optimal best-response solvers, though the upwind-to-downwind order is recommended to for sample efficiency and to minimise downwind wake impacts early on. These results highlight the potential for applying alternating maximisation to dynamic yaw control. Moreover, these promising outcomes suggest that any sub-optimal performance in dynamic yaw control can likely be attributed to differences between the two problems: the impact of maximum angular velocity on exploration functions, and the enlarged search-space induced by time-varying wind conditions.

5

Alternating Maximisation with Multi-Agent Deep Reinforcement Learning for Dynamic Yaw Control

In the previous chapter, we investigated numerical methods for static yaw optimisation, and have shown the efficacy of alternating maximisation in such a regime. Moreover, it appears the Nash equilibria in the yaw optimisation surface that are found through alternating maximisation can be quite good, resulting in robust performance.

This chapter aims to explore the application of alternating maximisation to multi-agent deep reinforcement learning for dynamic yaw control, specifically focusing on policy gradients. We will outline why deep reinforcement learning is a logical choice for such a problem, and how optimising from a noisy reward signal as well as exploring with a limited angular velocity complicates learning.

Deep reinforcement learning is well suited for dynamic yaw control for several reasons. First, real-life systems are often noisy due to imperfect sensors or partial observability of the wind. Reinforcement learning agents can learn policies that account for this noise, as demonstrated by Neustroev et al. [63] in their noisy observations benchmark. By contrast, yaw control strategies based on static yaw optimisation, such as FLORIS, do not account for this noise; they optimise for a specific wind condition under the assumption of perfect measurement, not considering that such measurements could be noisy. Second, optimisers like FLORIS can get stuck in local optima, as demonstrated in the previous chapter, because they do not explore. While reinforcement learning algorithms such as REINFORCE also have convergence guarantees for local optima, they can escape such local optima through exploration. However, the exploration-exploitation trade-off remains a key challenge in reinforcement learning [100]. Reinforcement learning agents may require many samples to converge to a good policy. For this purpose, we propose to use alternating maximisation for deep reinforcement learning, as it previously used to improve sample efficiency [109].

As described in Chapter 1, we hypothesise that issues stemming from multi-agent credit assignment, other agents performing exploration, and relative over-generalisation are the primary challenges for multi-agent dynamic yaw control. Alternating maximisation eliminates the first two problems by letting only one agent perform exploration and policy updates at a time. However, the possibility of relative over-generalisation still persists. Moreover, the search space for dynamic yaw control is larger than that of static yaw optimisation, which can be a challenge for reinforcement learning agents if data is limited relative to the size of the search space. As exemplified by farm sizes within single-agent reinforcement learning approaches to dynamic yaw control [121, 24, 25, 116, 118, 55] not exceeding fifteen turbines.

Compounding this challenge, is the fact that turbines have maximum rotation speed with which they can yaw, which introduces a temporal component to the problem. Future yaw angles now depend on past yaw angles. Agents needs to make a concerted effort over multiple time steps to move the turbine from one end of the possible yaws to the other end, which diminishes an agent capability to explore. Moreover, the dependence on past yaws limits stochastic exploration to a random walk. As a result, relative over-generalisation could occur due to the limited exploration capabilities. This is especially the case for reinforcement learning, as it often relies on random exploration. Consequently, agents must be capable of learning effective policies, often with limited data, while learning to plan for and explore with the finite angular velocity they are given.

5.1. Alternating Maximisation with Policy Gradient

To apply alternating maximisation to dynamic yaw control, we need to factor the multi-agent reinforcement learning problem into N sub-problems. We opt for an independent learning paradigm, as this simplifies the learning process, and has been shown to work for dynamic yaw control [46]. Each agent operates in a partially observable environment, where the other agents are assumed to be part of the environmental dynamics. Consequently, the policy probabilities $\pi_{\theta_i}(\boldsymbol{a}|\boldsymbol{s})$ are solely calculated with respect to that agent's actions, the policy has no notion of a "joint action".

We presume that an on-policy approach, like policy gradient, may be well suited for alternating maximisation, as old experiences may be misleading. Learning from past experiences is generally beneficial. However, in our case old experiences are not representative of the current world, since the policies of other agents have changed, which we have shown to greatly impact the shape of the reward surface in Chapter 3. As a result, the non-stationarity of policies induced by the learning of other agents could make previously high-reward state-action pairs bad. Such a misleading reward signal may destabilise learning if not handled appropriately. For this reason, we use an on-policy method in our research.

For the purpose of optimisation, we use the single-agent REINFORCE algorithm, described in Chapter 2, as it a straightforward policy gradient method. Ideally, through this choice of method, it becomes clear what the effect of alternating maximisation is on learning multi-agent dynamic yaw control.

Based on the findings on optimisation order and turbine initialisation from the previous chapter, we opt to use the upwind-to-downwind optimisation order, as it is most optimal and sample efficient. By the same token, we use the face-the-wind yaw-angle initialisation strategy for its optimality and sample efficiency. Additionally, these choices may be preferred as they are most congruent with current farm operation.

Every agent initially uses a face-the-wind policy, serving a deterministic starting point. During an agent's exploration face, this policy is replaced with a stochastic policy derived from policy gradient methods. This stochastic policy allows for exploration of the action space, improving the chances of discovering optimal actions.

Once the exploration phase concludes, the agent switches to a deterministic policy, utilising the mean of the stochastic policy distribution. This deterministic approach is chosen to prevent high frequency yaw adjustments, which are undesirable for turbine operation [30, 48, 89, 90]. Consistent with the needs of dynamic yaw control, which involves bounded action spaces, we select the Beta distribution for our stochastic policy. The Beta distribution is particularly well-suited for this task, as is bounded and its shape is flexible, allowing for a wide range of policies.

Summarising our methodology, we present the pseudo-code outlining the steps of our alternating maximisation process with policy gradients below in Algorithm 3. From this point forward, we will refer to this method as Alternating Policy Gradient.

Algorithm 3: Alternating Maximisation with Policy Gradients for Dynamic Yaw Control

Input: Set of D agents, sorted in upwind-to-downwind order
function AlternatingPolicyGradients(N_{iter}, N_τ)
  Initialise agents with face-the-wind policies.
  Initialise Beta policy networks π_{θ_i}.
  forall $j \in 1, 2, \ldots,\ N_{\text{iter}}$ **do**
    forall agents $i \in D$ **do**
      Enable stochastic policy π_{θ_i}
      Roll out trajectories $\tau = (\tau_1, \tau_2, \ldots, \tau_{N_\tau})$ with π_{θ_i}
      Compute policy gradient from $\tau_1, \tau_2, \ldots, \tau_{N_\tau}$ using *REINFORCE*.
      Update π_{θ_i} with using stochastic gradient ascent.
      Limit π_{θ_i} to distribution mean.

5.2. Experiment

In this experiment, we evaluate the effect of alternating maximisation on the performance of policy gradients. For this purpose, we use the distributed policy gradient [77], described in Chapter 2. Both methods are configured with identical hyperparameters, which are documented in Appendix A.

The experimental setup features a 3 × 3 layout, as described in Chapter 3, with each turbine using the NREL 5 MW reference turbine model [45], with a maximum angular velocity of 1° per second and a [-30°, 30°] yaw boundary. The wind conditions are constant, with wind coming from the west (270°), which presents a the most adversarial scenario for power production for this layout. Agents' policies must produce significant yaw-misalignment angles to deflect wakes around downwind turbines.

The wind speed is maintained at 9 m/s. Additional wind parameters are derived from the action representation benchmark by Neustroev et al. [63], based on the FLORIS model calibrated to the Hollandse Kust Noord (Site B) Dataset [60]. These parameters include a turbulence intensity of 0.12, wind shear of 9.3×10^{-3}, and wind veer of -2.5×10^{-2}.

The observations of each agent i at each time step t, as described in Chapter 2, comprise: wind speed at the turbine M_i^t, wind direction at the turbine ϕ_i^t, the freestream turbulence intensity I_i^t, as well as the turbine's current yaw angle y_i^t, resulting in the vector $[M_i^t, \phi_i^t, I_i^t, y_i^t] \in \mathbb{R}^4$. With each observation being normalised to [0,1], except for the yaw, which is normalised to [-1,1] corresponding to the yaw boundaries. Similarly, the action space for each agent is given by the interval [-1,1] and corresponds to the yaw boundaries [-30°, 30°].

For the alternating policy gradient method, there are 12 iterations of alternating maximisation. In each iteration, each agent sequentially explores for 1 episode, containing 1 trajectory of 900 time steps. For the 3×3 layout with $N = 9$ agents, this results in $9 \cdot 900 = 8100$ time steps per iteration. To ensure a fair comparison, the policy gradient method is allocated 12 episodes, corresponding to the 12 iterations of alternating maximisation. Each episode comprises 1 trajectory of 8100 time steps. This results in both methods utilising the equivalent total number of environment steps $8100 \cdot 12 = 97200$, and an equal amount of policy updates for each agent.

Each method is trained on 10 unique random seeds, and evaluated after every alternating maximisation iteration (or equivalently, after every distributed policy gradient episode) on one evaluation episode with one unique random seed and a length of 3600 time steps. More random seeds for the evaluation would serve no further benefit as the wind conditions are constant. Performance evaluation is conducted by measuring the relative performance in cumulative power production against a baseline face-the-wind policy.

The results of this experiment are displayed in Figure 5.1 and Table 5.1. Alternating policy gradient performs on average slightly worse than distributed policy gradient, yielding about a percentage point difference in mean improvement over baseline. This difference of means likely stems from a single outlier run, where distributed policy gradient was able to find a higher-quality policy. However, the alternating maximisation based method appears to have a more stable learning curve, with a tighter 95% confidence interval, which corroborates the claims by Wan, Xu, and Li [109] about alternating maximisation's stability. We speculate the difference in optimality may stem from the alternating policy gradient agents each having $1/N$ the amount of exploration time compared to their distributed policy gradient counter parts. As a result, they are more susceptible to approximation errors and their stochastic exploration policy taking a bad random walk, which does not give a sufficiently representative picture of the optimisation surface, leading the agents to converge to a false optimum.

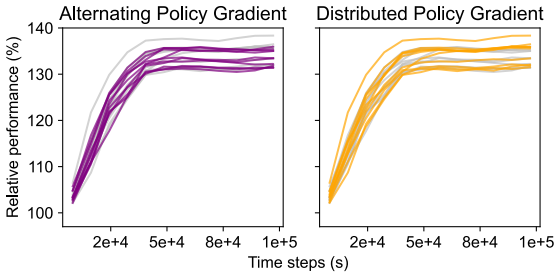

Figure 5.1: Relative performance in percentage to baseline (facing the wind) during training in the alternating maximisation with policy gradients experiment.

Method	Mean	95% conf. int.
Alternating PG	133.53	132.28 — 134.77
Distributed PG	**134.68**	133.02 — 136.33

Table 5.1: Relative performance in percentage to baseline (facing the wind) of the final policies in the alternating maximisation with policy gradients experiment.

5.3. Analysis

To confirm our assumption that the difference in optimality stems from insufficient exploration, we select a sample of the first episode in the training of turbine T1 (see Figure 3.3 in Chapter 3). In this seed, the alternating policy gradient agent converged to the sub-optimal policy, in the negative yaw angles, and distributed policy gradient agent converged to the higher-quality policy, in the positive yaw angles. Figure 5.2 displays the training yaw angles of turbine T1 during this first episode. From these yaws, we can infer that the alternating maximisation explored a smaller range of yaw angles compared to its distributed policy gradient counterpart, which could be a factor in its sub-optimal performance. We speculate that this smaller exploration range in combination with a reduced amount of data, compared to the distributed policy gradient agents, has lead to more approximation errors and subsequently a worse policy.

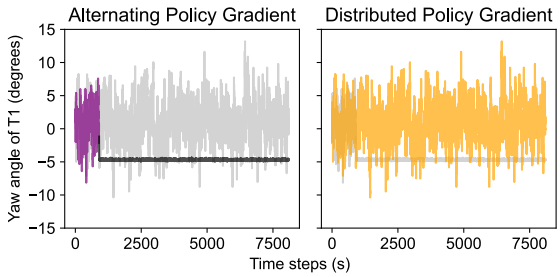

Figure 5.2: Yaw of turbine T1 during the first training episode of the alternating maximisation with policy gradients experiment. The determinised portion of the training episode for Alternating Policy Gradient has been coloured black.

Besides the difference between the two methods, we can observe that neither method was able to explore the yaw range well, with both methods staying within a range of $\pm 14°$. We posit this is a direct effect of the limited angular velocity of the turbines in conjunction with the stochastic exploration policies. Specifically, as the yaw strays further from zero, there is an over-representation of actions which drive the yaw back to zero. For instance, if a turbine is yawed at 10°, then all control actions corresponding to yaws $-30 \leq \text{yaw} < 10$ drive the yaw closer to zero, meaning two thirds of the actions yaw the turbine closer to facing the wind. Consequently, the exploration is biased around the face-the-wind policy.

For these reasons, we require stronger methods to sufficiently explore the state-action space, such that agents can gather enough information to steer clear of false optima.

5.4. Conclusion

Policies trained using our regime for alternating maximisation with policy gradients appear to be more susceptible to converge to false optima on average, though they have less variance between the learned policies. We theorise that the difference in optimality stems from a reduced amount of exploration that alternating policy gradient agents can perform, compared to distributed policy gradient agents within the same time frame.

However, as the average differences in optimality are slight, we contend that alternating maximisation agents are able to learn good policies from this reduced amount of data, likely due to not suffering from noise induced by other agents performing exploration. This reduced amount of exploration may be beneficial to real-life farm operation, where unnecessary yawing of turbines is undesirable.

Furthermore, as alternating policy gradient agents were able to learn from comparatively little data, perhaps there exists a middle ground between factored and fully joint exploration, where noise in the reward signal is low and policies can be learned quickly for multiple agents concurrently.

Lastly, we have observed that stochastic exploration policies cannot adequately explore the state-action space in the dynamic yaw control problem. We theorise problem arises due to the limited angular velocity of turbines, which causes an over-representation of actions driving the yaw back to zero in the wind-based action representation. Consequently, we require stronger exploration strategies to cover more of the yaw-space.

6

Guiding Exploration in Alternating Maximisation for Dynamic Yaw Control

In the previous chapter, we explored the application of alternating maximisation to multi-agent deep reinforcement learning for dynamic yaw control using policy gradients. Our results highlighted a limitation: alternating policy gradient agents can explore less compared to distributed policy gradients in the same time frame, leading to worse performance. However, the performance gap was slight, suggesting alternating policy gradient provides a clear reward from which a good but sub-optimal policy can be learned.

Moreover, we found that in both methods, combining a limited angular velocity with stochastic exploration results in poor coverage of the state-action space. The random-walk nature of this exploration regime, in conjunction with an over-representation of actions moving the yaw angle to zero, limited the explored yaw range to ±14°. By contrast, the numerical methods from Chapter 4 were not constrained by a limited angular velocity, allowing for sufficient exploration of the state-action space to identify optimal yaw-configurations.

Furthermore, in Chapter 3, we have shown that optimal yaws can span the entire range of possible yaws. More specifically, in the scenario where turbines are aligned along the wind vector, the optimal yaws are often found near the yaw boundaries. Alternatively, when not aligned, the optimal yaw-angles are closer to zero.

In light of these findings, this chapter focuses on developing exploration policies tailored specifically to exploring the entire range of yaws. Our goal is to create exploration strategies for alternating maximisation that can effectively navigate the entire range of yaw angles, enabling agents to learn the true optimal configurations rather than being misled by false optima

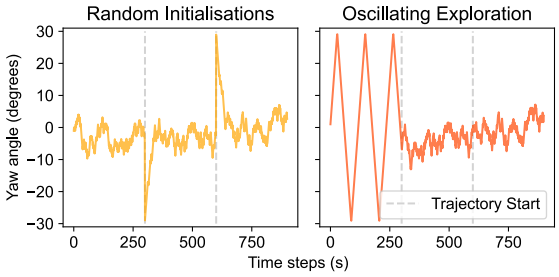

Figure 6.1: Yaw angles for the exploration strategies in an example episode of 3 trajectories, each of length 300, using hyperparameters from the upcoming exploration experiment.

6.1. Random Initialisation

Our first proposed exploration method uses random initialisation to explore the full breadth of yaws. This method lends from the random restarts done in joint equilibrium search for policies [61], a dynamic programming method for solving decentralised Markov decision processes which uses alternating maximisation.

In our approach, turbines are initialised with a random yaw angle, selected from a predefined set Y_0, at the start of each trajectory in their exploration episode. Let Y_0 be a set of size k with all members within the interval $[y_{\min}, y_{\max}]$: $Y_0 = \{y_{0,i} \mid y_{0,i} \in [y_{\min}, y_{\max}], i = 1, 2, \ldots, k\}$. This initial yaw angle allows the turbines to explore novel state-action pairs, which counterbalances the exploitation nature of reinforcement learning policies. Specifically, we set $k = 3$ with $Y_0 = \{-30°, 0°, 30°\}$ These angles are chosen because they are likely candidates for optimal yaw configurations, as shown in Chapter 3, and their inclusion aims to facilitate a balanced exploration strategy. Furthermore, the selection of 30° (as opposed

to e.g., 25°) is based on the over-representation of actions driving the yaw to zero, which we described in Chapter 5. In this way, exploration is certain to include the yaw boundaries, which potentially hold quality solutions. The resulting yaw angles are depicted in Figure 6.1.

If the number of initial yaws k exceeds the number of trajectories per episode, the initial yaws should be sampled without replacement. Alternatively, if the number of trajectories exceeds the number of yaw angles in the set Y_0, the set should be repeated until the size matches or exceeds the number of trajectories, after which yaws should again be sampled without replacement. This approach ensures unbiased selection of initial yaw angles.

We hypothesise that random initialisation may not extend well to the joint exploration case. If one would do concurrent random initialisation, there are $|Y|^N$ possible initial configurations, where $|Y|$ denotes the number of possible initial yaw angles, and N represents the number of turbines. For a set of three initial yaw angles and a group of 25 turbines, this would result in $3^{25} \approx 8.5 \times 10^{11}$ unique combinations of initial yaws. The vast majority of which will lead to sub-optimal power output, and as a result bias the training data with many negative samples. For these reasons, this approach is likely a poor candidate for joint exploration.

Alternatively, one could decide to initialise all turbines with identical yaws. This approach may be better, as we expect optimal yaw-angles to point in roughly the same direction. However, the explored region of the search space would be drastically reduced. Investigating whether this specific reduction is beneficial, we leave to future research.

6.2. Oscillating Exploration

For our second method, we propose to forcefully explore the entire range. In this approach, we replace some of the trajectories with forced exploration trajectories, where agents oscillate between the ends of the yaw boundaries. By adding these exploration episodes, we ensure agents explore the full range of possible yaws while still allowing for exploitation, ideally, allowing them to learn where the true optimum in the reward signal is.

We speculate oscillating exploration may be the best approach for alternating maximisation, given that it ensures agents explore the full breadth of the possible yaws in each episode, similar to the numerical methods described in Chapter 4. However, it must be noted that this type of exploration depends on the type of simulator used. In our case, we use a steady state simulator, like FLORIS, in which propagates wakes instantly, and as a result the oscillation rate can be arbitrary as long as it enables full exploration of yaws within the episode. By contrast, in large-eddy simulations, the wakes need time to propagate, thus a much slower oscillation rate may be required to fully capture the effect of the given yaw angle.

To facilitate oscillation, we keep track of the current yaw angle y_t and direction of oscillation $d_t \in \{-1, 1\}$. To select the next control action a_t, we add the maximum delta yaw ω_{max} to the current yaw, in the direction of oscillation. When either of the yaw boundaries, y_{min} or y_{max}, are reached, we clip the action to the valid range and flip the direction of oscillation. Computation of the exploration action is summarised in the following formulas:

$$a_t = \frac{\text{clip}(y_t + d_t \cdot \omega_{\text{max}}, y_{\text{min}}, y_{\text{max}})}{y_{\text{max}} - y_{\text{min}}} \tag{6.1}$$

$$d_{t+1} = \begin{cases} -d_t & \text{if } |a_t| = 1, \\ d_t & \text{otherwise.} \end{cases} \tag{6.2}$$

As a byproduct of our use of the Beta policy, selecting either $-30°$ or $30°$ as a control action would make optimisation of this policy impossible. The equivalent actions supplied to the Beta distribution, 0 and 1, are undefined, and would result in infinities when attempting to compute the policy gradient. For this reason, we reduce the yaw boundaries for oscillation by 1°. Figure 6.1 displays the resulting yaw angles of this strategy.

Similar to the previous exploration strategy, we contend that this approach is best suited to alternating exploration. Attempting to explore the joint search space using concurrent oscillating exploration would require the oscillation periods to be co-prime, lest they explore a repeated sequence which does not cover the entire search space. What is more, these oscillation periods would grow very large as the number of turbines increases, as the numbers in sets of co-prime integers grows fast. This problem could be alleviated to a certain degree by applying co-prime periods only among turbines which influence each other. One could then solve a graph colouring problem to assign the oscillation periods to turbines.

Furthermore, if one decides to let turbines oscillate with the same periods, akin to the approach described in the final paragraph of Section 6.1, then the explored search space would be further reduced. Because the oscillation is deterministic and noiseless, all agents would explore identical yaw-angles at the same time.

6.3. Experiments

In this experiment, we evaluate the effects of guided exploration on policies learned with alternating policy gradient. We use the same experiment setup as in Chapter 5.

For both exploration methods, we keep the number of time steps per episode the same, but reduce the trajectory length down from 900 to 300 time steps, and increase the number of trajectories per episode from 1 to 3. Then, for the oscillating exploration strategy, we enable the guided exploration only for the first trajectory of each episode.

The results are presented in Figure 6.2 and Table 6.1. Random initialisation performs slightly better than the standard alternating policy gradient, but the differences fall within margin of error. Additionally, agents using random initialisation tend to converge slower than their stochastic counterparts.

In contrast, the oscillating exploration strategy significantly improves performance, achieving nearly 4%-higher performance than the other strategies. Furthermore, the confidence interval of the mean for the final policy is notably narrower, at less than half a percentage point wide, and non-overlapping with any of the methods from this chapter and the previous chapter. This indicates that forced exploration through oscillation enables agents to consistently converge to a high-quality policy.

Interestingly, agents trained with oscillating exploration show worse initial scores, slower convergence, and more varied policies during the learning process compared to other methods. However, the long-term benefits of this strategy potentially outweigh these initial drawbacks, resulting in a more reliable and effective final policy.

We speculate that the performance discrepancy between the two methods can be attributed to the nature of the control actions selected by the oscillating exploration strategy. Oscillating exploration selects control actions that are close to the current yaw angle, thereby associating those control actions with the rewards for that specific yaw angle.

Exploration Method	Mean	95% conf. int.
Stochastic (Default)	133.53	132.28 — 134.77
Random Initialisation	133.68	132.44 — 134.91
Oscillating Exploration	**137.28**	137.06 — 137.50

Table 6.1: Relative performance in percentage to baseline (facing the wind) of the final policies in the alternating policy gradient exploration experiment.

6.4. Analysis

To confirm whether oscillating exploration is performing optimally, we examine the final policies by looking at the yaw angles produced for each turbine and random seed. Figure 6.3 displays the yaw angles, binned per 5°. The plots are supplemented by the solution found by coarse-to-fine alternating yaw optimisation, which did not have a limited angular velocity.

The oscillating exploration strategy nearly consistently converges to the target solution given by alternating yaw optimisation. Conversely, both the standard alternating policy gradient and random initialisation methods converge to local optima with mixed yaw angles. We speculate that the stochastic exploration in conjunction with random restarts is still inadequate for exploring the state-action space well, as the control actions do not line up with the current yaw angles, making it unclear which control actions are optimal in the given time frame. This could be alleviated by collecting more trajectories, however, given the success of the oscillation strategy, we suggest not pursuing our random initialisations implementation further.

Additionally, for the rear turbines (T3, T6, and T9), the solution quality is higher for the oscillating exploration method, compared to the other two methods, with the variance in yaw angles between policies being much lower. This reduced variance indicates a higher consistency in the policies generated by oscillating exploration, further underscoring its ability to guide agents towards more reliable and effective final configurations.

6.5. Conclusion

This chapter has explored the development and evaluation of two domain-knowledge driven exploration strategies for alternating maximisation in multi-agent deep reinforcement learning for dynamic wake control. Building on the limitations identified in Chapter 5, we aimed to created exploration policies capable of navigating the full range of yaw angles.

Our experiments revealed that initialising turbines with random yaws at the start of their exploration trajectories performs slightly better than standard alternating policy gradient. However, the differences were within margin of error. More importantly, the oscillating exploration strategy emerged as significantly superior, consistently achieving nearly 4%-higher performance than other strategies. This method also produced a notably narrow and non-overlapping confidence interval for the mean performance of the final policy.

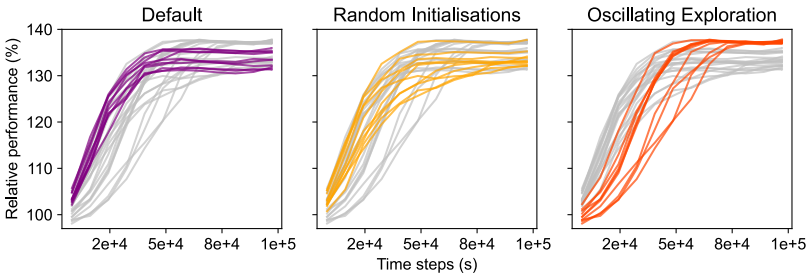

Figure 6.2: Relative performance in percentage to baseline (facing the wind) during training in the alternating policy gradient exploration experiment.

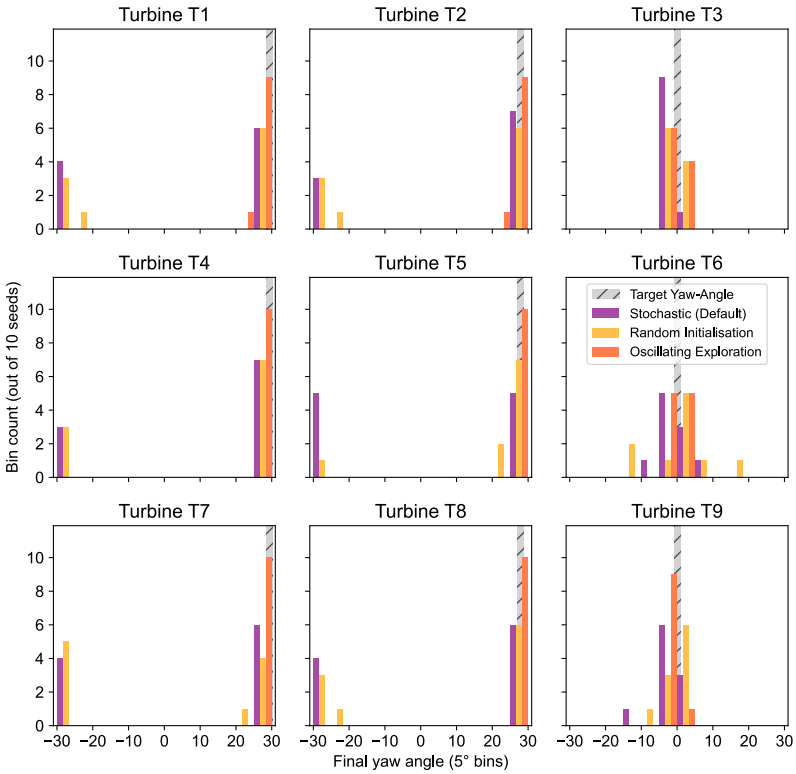

Figure 6.3: Final policy yaw-angles, binned per 5°, for each of the 10 random seeds in the alternating policy gradient exploration experiment. The target solution is given by alternating yaw optimisation with coarse-to-fine sampling.

Analysis of the final policy confirmed that oscillating exploration was virtually consistent in finding the target solution given by coarse-to-fine alternating yaw optimisation. Conversely, the default and random initialisation strategies tended to find local optima with mixed yaw angles. We speculate that oscillating exploration performs well because its actions are close to the current yaw angle, which associates those actions to their steady-state yaw angles. This clear association allows agents to exploit the wind-based action-representation.

In conclusion, the oscillating exploration strategy provides an effective approach for alternating maximisation with multi-agent deep reinforcement learning in dynamic yaw control. By enabling comprehensive exploration of the yaw space, this strategy overcomes the limitations of traditional methods when faced with turbines with a limited rotation speed. Furthermore, these results are consistent with the findings from Chapter 4, which showed alternating maximisation can lead to high-quality solutions, further validating the effectiveness of this approach in different yaw control scenarios.

Future work could explore the application of these findings to dynamic yaw control with time-varying wind conditions, but a more detailed discussion of these implications will be reserved for the conclusion chapter of the thesis.

7

Conclusion

This thesis has investigated the application of alternating maximisation to active wake control, focusing on both static yaw optimisation and dynamic yaw control. Our research aimed to address key challenges and evaluate the benefits of this approach through a series of targeted research questions. Specifically, we examined the effects of atmospheric conditions and wind farm layouts on static yaw optimisation (Q1) and its sub-questions (Q1.1–Q1.4), assessed the effectiveness and influencing factors of the alternating maximisation method in static yaw optimisation (Q2) and its sub-questions (Q2.1–Q2.3), explored the impact of alternating maximisation on convergence in dynamic yaw control via reinforcement learning (Q3), and investigated the role of domain knowledge in preventing local optima convergence for reinforcement learning agents (Q4).

This concluding chapter is organised as follows: first, we summarise the findings for static yaw optimisation, detailing how wind direction, speed, farm layout, and yaw-misalignment angles influence the optimisation surface. Next, we discuss the application of alternating maximisation, highlighting the best-response problem approaches, the effect of initial yaw configurations, and optimisation order on the resulting Nash equilibria. Following this, we present our results on dynamic yaw control, focusing on the convergence impacts and strategies to avoid local optima. We then address the limitations of our study, providing an evaluation of the methodologies and assumptions used. Finally, we reflect on the broader implications of our findings for future research and practical implementations in active wake control.

7.1. Conclusions

In static yaw optimisation, our findings highlighted the intricate effects of wind direction, wind speed, turbine layouts on the optimisation surface. Nash equilibria were shown to be transient and influenced by these factors, indicating to use that alternating maximisation is a promising method for many wind conditions. Wind speed affects the prominence of local optima, with mixed yaw angles becoming more significant as speeds increase, though still generally sub-optimal. Layout depth also influences the optimisation surface; as the number of aligned turbines increases, the optimality gap between Nash equilibria grows, and the modes in the surface become narrower but more prominent, clarifying the optimisation direction. Adversarial structures in grid layouts caused by turbine translations can mislead convergence to sub-optimal Nash equilibria, but these remain transient due to wind direction dependency. The small optimality gap in the investigated layouts suggests practical applicability of alternating maximisation. Yaw misalignment experiments show that turbines tend to align their yaw responses with the leading turbine's direction, diminishing with distance. This aligning behaviour underscores the importance of initial optimisations, as early identification of the correct direction could improve convergence speed and optimality.

We discovered that alternating maximisation, particularly with a sampling-based approach, consistently outperformed other methods, differential evolution and sequential least squares programming, across various metrics and layouts. The robustness of this method was evident, as it was less sensitive to initial yaw configurations and optimisation orders than the gradient-based alternative, making it suitable for practical applications in wind farms. Evidently, a gradient-based approach is less suitable for practical applications, as it was more sensitive to initial conditions, was not able to optimally solve the best-response problems in static yaw optimisation consistently, and required many more samples to solve the best-response problems. Moreover, we found that using a face-the-wind yaw initialisation, which is congruent with current farm operation, was the most optimal and sample-efficient initialisation strategy. By contrast, a negative yaw-angle initialisation appeared to draw convergence to sub-optimal Nash

equilibria. Furthermore, of the optimisation orders tested, an upwind-to-downwind optimisation order was found to most optimal while requiring the least amount of function evaluations.

For dynamic yaw control, alternating maximisation with policy gradient demonstrated greater sample efficiency compared to distributed policy gradient methods. Both methods tended to converge to sub-optimal policies due to limited exploration capabilities. This limitation stems from combining the wind-based action-representation a turbine's limited rotation speed, resulting in an over-representation of actions that drive the yaw back to zero, restricting the range of explored yaw angles.

Nevertheless, given the limited scale of the best-response problems in dynamic yaw control, we were able to develop an adequate exploration strategy which covers the entire range of yaws. Using an oscillating exploration strategy lead to significantly enhanced performance. With this exploration strategy, agents were able to consistently converge to high-quality policies. In contrast, exploration by initialising the yaws at the start of each trajectory, while still exploring the full range of yaws, did not achieve the desired results. This discrepancy between the two methods indicates that exploring the full range of yaws is insufficient for a high-quality policy. We speculate that oscillating exploration performs well because its actions are close to the current yaw angle, which associates those actions to their steady-state yaw angles. This clear association allows agents to exploit the wind-based action-representation.

Overall, our findings indicate that alternating maximisation is a promising method for numerical static yaw optimisation across various wind conditions and turbine layouts, and in multi-agent deep reinforcement learning for dynamic yaw control in constant wind conditions.

7.2. Limitations

Our work has primarily focused on farm power output as the optimal performance metric, neglecting to account for structural loading or leading-edge erosion in our evaluations. While maximising power output is critical, considering potential adverse effects and their impact on turbine longevity and maintenance costs is also essential.

Additionally, the accuracy of our results is inherently limited by the fidelity of the simulation models used. Real-world conditions may introduce complexities not captured in the simulations, such as turbulent flow interactions and environmental changes such as time-varying wind. The dynamic nature of wind speeds and directions expands the search space and complicates the exploration of the state-action space, as agents cannot control atmospheric conditions. Consequently, learning high-quality policies under these varying conditions may require a significantly more samples.

Future studies should integrate these factors to provide a more holistic assessment of alternating-maximisation-based optimisation strategies.

7.3. Implications and Future Work

Our findings demonstrate the potential of alternating maximisation to improve wind farm performance through both static yaw optimisation and dynamic yaw control. The success of sampling-based numerical optimisation and oscillation exploration for dynamic yaw control highlights the importance of comprehensive exploration of best-response problems in overcoming local optima and achieving optimal yaw configurations. While our research has laid a foundation for alternating maximisation in active wake control, future work should focus on applying these strategies to dynamic yaw control under time-varying conditions, exploring the balance between exploration and exploitation in more complex scenarios.

For time-varying wind conditions, the scale of rewards is significantly influenced by factors such as wind direction and wind speed, which agents cannot control. With limited samples, high rewards from episodes of high wind speeds can mislead agents into favouring actions associated with these high rewards, even if they are not optimal in other conditions. To address this issue, we recommend implementing reward normalisation based on wind speed and direction.

Furthermore, we posit that there may be more efficient ways to solve the best-response problems in static yaw optimisations. Our coarse-to-fine sampling strategy works well for solving the best-response problems, however, it wastes many samples in the fine-sampling step. A more efficient strategy may initially sample coarsely and attempt further optimisation of the best-response problem through Lipschitz optimisation [39]. By leveraging the property of Lipschitz continuity, which puts bounds on how fast a function can grow, the position of the optimum could be quickly constrained. To facilitate such optimisation, the required Lipschitz coefficient could be roughly approximated from the power curves of turbines, which assume no wake interactions.

References

[1] Per Dannemand Andersen et al. “Recycling of wind turbines”. English. In: *DTU International Energy Report 2014*. Technical University of Denmark, 2014, pp. 91–97. ISBN: 978-87-550-3969-8.

[2] Eugenio Bargiacchi et al. “Multi-agent RMax for Multi-Agent Multi-Armed Bandits”. en. In: *Proceedings of Adaptive and Learning Agents Workshop 2022* (2022).

[3] Rebecca Jane 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. ISSN: 1095-4244. DOI: 10.1002/we.348.

[4] A. G. Barto, R. S. Sutton, and C. J.C.H. Watkins. *Learning and Sequential Decision Making*. Technical Report. USA: University of Massachusetts, Aug. 1989.

[5] Richard Bellman. “A Markovian Decision Process”. In: *Journal of Mathematics and Mechanics* 6.5 (1957). Publisher: Indiana University Mathematics Department, pp. 679–684. ISSN: 0095-9057. URL: https://www.jstor.org/stable/24900506 (visited on 02/28/2024).

[6] James C. Bezdek and Richard J. Hathaway. “Some Notes on Alternating Optimization”. en. In: *Advances in Soft Computing — AFSS 2002*. Ed. by Nikhil R. Pal and Michio Sugeno. Berlin, Heidelberg: Springer, 2002, pp. 288–300. ISBN: 978-3-540-45631-5. DOI: 10.1007/3-540-45631-7_39.

[7] Bilal et al. “Differential Evolution: A review of more than two decades of research”. In: *Engineering Applications of Artificial Intelligence* 90 (Apr. 2020), p. 103479. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2020.103479. URL: https://www.sciencedirect.com/science/article/pii/S095219762030004X (visited on 05/07/2024).

[8] James Bleeg et al. “Wind Farm Blockage and the Consequences of Neglecting Its Impact on Energy Production”. en. In: *Energies* 11.6 (June 2018), p. 1609. ISSN: 1996-1073. DOI: 10.3390/en11061609. URL: http://www.mdpi.com/1996-1073/11/6/1609 (visited on 04/17/2024).

[9] Sjoerd Boersma et al. *A control-oriented dynamic wind farm model: WFSim*. en. Oct. 2017. DOI: 10.5194/wes-2017-44. URL: https://wes.copernicus.org/preprints/wes-2017-44/wes-2017-44.pdf (visited on 07/05/2024).

[10] Greg Brockman et al. *OpenAI Gym*. en. arXiv:1606.01540 [cs]. June 2016. URL: http://arxiv.org/abs/1606.01540 (visited on 05/11/2024).

[11] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. “Multi-agent Reinforcement Learning: An Overview”. en. In: *Innovations in Multi-Agent Systems and Applications - 1*. Vol. 310. Series Title: Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–221. ISBN: 978-3-642-14434-9 978-3-642-14435-6. DOI: 10.1007/978-3-642-14435-6_7. URL: http://link.springer.com/10.1007/978-3-642-14435-6_7 (visited on 06/25/2024).

[12] Filippo Campagnolo et al. “Wind tunnel testing of a closed-loop wake deflection controller for wind farm power maximization”. en. In: *Journal of Physics: Conference Series* 753 (Sept. 2016), p. 032006. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/753/3/032006. URL: https://iopscience.iop.org/article/10.1088/1742-6596/753/3/032006 (visited on 03/02/2024).

[13] Jacopo Castellini et al. “Difference Rewards Policy Gradients”. en. In: *Neural Computing and Applications* (Nov. 2022). arXiv:2012.11258 [cs]. ISSN: 0941-0643, 1433-3058. DOI: 10.1007/s00521-022-07960-5. URL: http://arxiv.org/abs/2012.11258 (visited on 05/16/2024).

[14] Yu-Han Chang, Tracey Ho, and Leslie Pack Kaelbling. “All learning is local: multi-agent learning in global reward games”. In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS’03. Cambridge, MA, USA: MIT Press, Dec. 2003, pp. 807–814. (Visited on 05/15/2024).

[15] Wei Chen, Yajun Wang, and Yang Yuan. “Combinatorial Multi-Armed Bandit: General Framework and Applications”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research. Issue: 1. Atlanta, Georgia, USA: PMLR, June 2013, pp. 151–159. URL: https://proceedings.mlr.press/v28/chen13a.html.

[16] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. “Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution”. en. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Sydney, Australia, 2017.

[17] Matt Churchfield and Sang Lee. *NWTC Design Codes (SOWFA)*. en. 2012. URL: https://wind.nrel.gov/designcodes/simulators/SOWFA/ (visited on 03/02/2024).

[18] Andreas Wolf Ciavarra et al. “Wind farm optimization with multiple hub heights using gradient-based methods”. en. In: *Journal of Physics: Conference Series* 2265.2 (May 2022). Publisher: IOP Publishing, p. 022012. ISSN: 1742-6596. DOI: 10.1088/1742-6596/2265/2/022012. URL: https://dx.doi.org/10.1088/1742-6596/2265/2/022012 (visited on 02/28/2024).

[19] R Clayton and P Filby. “Measured effects of oblique flows and change in blade pitch angle on performance and wake development of model wind turbines”. In: *Proceedings of the fourth BWEA Wind Energy Conference*. Cranfield, UK, 1982, pp. 214–224.

[20] A. Crespo and J. Herna´ndez. “Turbulence characteristics in wind-turbine wakes”. en. In: *Journal of Wind Engineering and Industrial Aerodynamics* 61.1 (June 1996), pp. 71–85. ISSN: 01676105. DOI: 10.1016/0167-6105(95)00033-X. URL: https://linkinghub.elsevier.com/retrieve/pii/016761059500033X (visited on 03/03/2024).

[21] Robert H. Crites and Andrew G. Barto. “Elevator Group Control Using Multiple Reinforcement Learning Agents”. en. In: *Machine Learning* 33.2 (Nov. 1998), pp. 235–262. ISSN: 1573-0565. DOI: 10.1023/A:1007518724497. URL: https://doi.org/10.1023/A:1007518724497 (visited on 06/25/2024).

[22] Zhiwen Deng et al. “Decentralized yaw optimization for maximizing wind farm production based on deep reinforcement learning”. In: *Energy Conversion and Management* 286 (June 2023), p. 117031. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2023.117031. URL: https://www.sciencedirect.com/science/article/pii/S0196890423003771 (visited on 03/03/2024).

[23] H. Díaz and C. Guedes Soares. “Review of the current status, technology and future trends of offshore wind farms”. In: *Ocean Engineering* 209 (Aug. 2020), p. 107381. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2020.107381. URL: https://www.sciencedirect.com/science/article/pii/S002980182030411X (visited on 05/13/2024).

[24] Hongyang Dong, Jincheng Zhang, and Xiaowei Zhao. “Intelligent wind farm control via deep reinforcement learning and high-fidelity simulations”. In: *Applied Energy* 292 (June 2021), p. 116928. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2021.116928. URL: https://www.sciencedirect.com/science/article/pii/S0306261921004086 (visited on 03/02/2024).

[25] 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 (May 2022). Conference Name: IEEE Transactions on Control Systems Technology, pp. 1281–1295. ISSN: 1558-0865. DOI: 10.1109/TCST.2021.3102476. URL: https://ieeexplore.ieee.org/abstract/document/9512381 (visited on 07/05/2024).

[26] Hongyang Dong and Xiaowei Zhao. “Intelligent Wind Farm Control via Grouping-Based Reinforcement Learning”. In: *2022 European Control Conference (ECC)*. July 2022, pp. 993–998. DOI: 10.23919/ECC55457.2022.9838151. URL: https://ieeexplore.ieee.org/document/9838151 (visited on 04/13/2024).

[27] R. Emery-Montemerlo et al. “Approximate solutions for partially observable stochastic games with common payoffs”. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.* July 2004, pp. 136–143. URL: https://ieeexplore.ieee.org/document/1373472 (visited on 04/19/2024).

[28] L. Fingersh and P. Carlin. “Results from the NREL Variable-Speed Test bed”. In: *1998 ASME Wind Energy Symposium*. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1998-50. American Institute of Aeronautics and Astronautics, 1998. DOI: 10.2514/6.1998-50. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1998-50 (visited on 04/12/2024).

[29] Paul Fleming et al. “Field test of wake steering at an offshore wind farm”. en. In: *Wind Energy Science* 2.1 (May 2017), pp. 229–239. ISSN: 2366-7451. DOI: 10.5194/wes-2-229-2017. URL: https://wes.copernicus.org/articles/2/229/2017/ (visited on 04/18/2024).

[30] Paul Fleming et al. “Initial results from a field campaign of wake steering applied at a commercial wind farm – Part 1”. English. In: *Wind Energy Science* 4.2 (May 2019). Publisher: Copernicus GmbH, pp. 273–285. ISSN: 2366-7443. DOI: 10.5194/wes-4-273-2019. URL: https://wes.copernicus.org/articles/4/273/2019/ (visited on 04/18/2024).

[31] Paul A. Fleming et al. “Evaluating techniques for redirecting turbine wakes using SOWFA”. In: *Renewable Energy*. Special issue on aerodynamics of offshore wind energy systems and wakes 70 (Oct. 2014), pp. 211–218. ISSN: 0960-1481. DOI: 10.1016/j.renene.2014.02.015. URL: https://www.sciencedirect.com/science/article/pii/S0960148114000950 (visited on 04/13/2024).

[32] Paul A. Fleming et al. “Serial-Refine Method for Fast Wake-Steering Yaw Optimization”. en. In: *Journal of Physics: Conference Series* 2265.3 (May 2022), p. 032109. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/2265/3/032109. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2265/3/032109 (visited on 04/18/2024).

[33] Jakob Foerster et al. *Counterfactual Multi-Agent Policy Gradients*. en. arXiv:1705.08926 [cs]. Dec. 2017. URL: http://arxiv.org/abs/1705.08926 (visited on 05/16/2024).

[34] Sten Frandsen et al. “Analytical modelling of wind speed deficit in large offshore wind farms”. en. In: *Wind Energy* 9.1-2 (Jan. 2006), pp. 39–53. ISSN: 1095-4244, 1099-1824. DOI: 10.1002/we.189. URL: https://onlinelibrary.wiley.com/doi/10.1002/we.189 (visited on 03/03/2024).

[35] P. M. O. Gebraad et al. “Wind plant power optimization through yaw control using a parametric model for wake effects-a CFD simulation study: Wind plant optimization by yaw control using a parametric wake model”. en. In: *Wind Energy* 19.1 (Jan. 2016), pp. 95–114. ISSN: 10954244. DOI: 10.1002/we.1822. URL: https://onlinelibrary.wiley.com/doi/10.1002/we.1822 (visited on 03/02/2024).

[36] P.M.O. Gebraad, P.A. Fleming, and J.W. van Wingerden. “Comparison of actuation methods for wake control in wind plants”. In: *2015 American Control Conference (ACC)*. ISSN: 2378-5861. July 2015, pp. 1695–1701. DOI: 10.1109/ACC.2015.7170977. URL: https://ieeexplore.ieee.org/document/7170977 (visited on 03/02/2024).

[37] Pieter Gebraad, Filip Dam, and J. W. Wingerden. “A model-free distributed approach for wind plant control”. In: June 2013, pp. 628–633. ISBN: 978-1-4799-0177-7. DOI: 10.1109/ACC.2013.6579907.

[38] E. Hansen, D. Bernstein, and S. Zilberstein. “Dynamic Programming for Partially Observable Stochastic Games”. In: July 2004. URL: https://www.semanticscholar.org/paper/Dynamic-Programming-for-Partially-Observable-Games-Hansen-Bernstein/b9764ed9cf14b439235987dfe65d35bb6ce406ef (visited on 02/29/2024).

[39] Pierre Hansen and Brigitte Jaumard. “Lipschitz Optimization”. en. In: *Handbook of Global Optimization*. Ed. by Reiner Horst and Panos M. Pardalos. Boston, MA: Springer US, 1995, pp. 407–493. ISBN: 978-1-4615-2025-2. DOI: 10.1007/978-1-4615-2025-2_9. URL: https://doi.org/10.1007/978-1-4615-2025-2_9 (visited on 07/03/2024).

[40] T. Heus et al. “Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications”. en. In: *Geoscientific Model Development* 3.2 (Sept. 2010), pp. 415–444. ISSN: 1991-9603. DOI: 10.5194/gmd-3-415-2010. URL: https://gmd.copernicus.org/articles/3/415/2010/ (visited on 03/02/2024).

[41] Michael F. Howland, Sanjiva K. Lele, and John O. Dabiri. “Wind farm power optimization through wake steering”. en. In: *Proceedings of the National Academy of Sciences* 116.29 (July 2019), pp. 14495–14500. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1903680116. URL: https://pnas.org/doi/full/10.1073/pnas.1903680116 (visited on 03/02/2024).

[42] N. Jensen. “A note on wind generator interaction”. In: 1983. URL: https://www.semanticscholar.org/paper/A-note-on-wind-generator-interaction-Jensen/385f829b931fa647df18d8f2226dd577b63f8660 (visited on 03/03/2024).

[43] Kun Jiang et al. “Credit assignment in heterogeneous multi-agent reinforcement learning for fully cooperative tasks”. en. In: *Applied Intelligence* 53.23 (Dec. 2023), pp. 29205–29222. ISSN: 1573-7497. DOI: 10.1007/s10489-023-04866-0. URL: https://doi.org/10.1007/s10489-023-04866-0 (visited on 05/16/2024).

[44] Ángel Jiménez, Antonio Crespo, and Emilio Migoya. “Application of a LES technique to characterize the wake deflection of a wind turbine in yaw”. en. In: *Wind Energy* 13.6 (Sept. 2010), pp. 559–572. DOI: 10.1002/we.380. URL: https://onlinelibrary.wiley.com/doi/10.1002/we.380 (visited on 03/03/2024).

[45] J. Jonkman et al. *Definition of a 5-MW Reference Wind Turbine for Offshore System Development*. English. Tech. rep. NREL/TP-500-38060. National Renewable Energy Lab. (NREL), Golden, CO (United States), Feb. 2009. DOI: 10.2172/947422. URL: https://www.osti.gov/biblio/947422 (visited on 05/08/2024).

[46] Elie Kadoche et al. “MARLYC: Multi-Agent Reinforcement Learning Yaw Control”. In: *Renewable Energy* 217 (Nov. 2023), p. 119129. ISSN: 0960-1481. DOI: 10.1016/j.renene.2023.119129. URL: https://www.sciencedirect.com/science/article/pii/S0960148123010431 (visited on 02/28/2024).

[47] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1 (May 1998), pp. 99–134. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(98)00023-X. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X (visited on 02/28/2024).

[48] Stoyan Kanev. “Dynamic wake steering and its impact on wind farm power production and yaw actuator duty”. In: *Renewable Energy* 146 (Feb. 2020), pp. 9–15. ISSN: 0960-1481. DOI: 10.1016/j.renene.2019.06.122. URL: https://www.sciencedirect.com/science/article/pii/S0960148119309565 (visited on 05/03/2024).

[49] Jahnavi Kantharaju et al. “Wind resource modelling of entire sites using Large Eddy Simulation”. en. In: *Journal of Physics: Conference Series* 2507.1 (May 2023), p. 012015. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/2507/1/012015. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2507/1/012015 (visited on 03/02/2024).

[50] Dieter Kraft. *A software package for sequential quadratic programming*. Place: Köln Series: Forschungsbericht. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, DFVLR, 88-28 Volume: 88-28. 1988. URL: https://www.tib.eu/de/suchen/id/TIBKAT%3A016896521.

[51] Gökay Kütükçü and Oğuz Uzol. “Monte-Carlo simulations based hub height optimization using FLORIS for two interacting onshore wind farms”. In: *Journal of Renewable and Sustainable Energy* 14.6 (Nov. 2022), p. 063304. ISSN: 1941-7012. DOI: 10.1063/5.0107244. URL: https://doi.org/10.1063/5.0107244 (visited on 02/28/2024).

[52] Jaejoon Lee et al. “Blade pitch angle control for aerodynamic performance optimization of a wind farm”. In: *Renewable Energy*. AFORE 2011(Asia-Pacific Forum of Renewable Energy 2011) 54 (June 2013), pp. 124–130. ISSN: 0960-1481. DOI: 10.1016/j.renene.2012.08.048. URL: https://www.sciencedirect.com/science/article/pii/S0960148112005186 (visited on 04/13/2024).

[53] Jun Li, Kai Zou, and Lining Xing. “Coarse-to-fine evolutionary search for large-scale multi-objective optimization: An application to ratio error estimation of voltage transformers”. English. In: *Frontiers in Energy Research* 10 (Sept. 2022). Publisher: Frontiers. ISSN: 2296-598X. DOI: 10.3389/fenrg.2022.988772. URL: https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2022.988772/full (visited on 06/29/2024).

[54] Jaime Liew et al. “Extending the dynamic wake meandering model in HAWC2Farm: a comparison with field measurements at the Lillgrund wind farm”. English. In: *Wind Energy Science* 8.9 (Sept. 2023). Publisher: Copernicus GmbH, pp. 1387–1402. ISSN: 2366-7443. DOI: 10.5194/wes-8-1387-2023. URL: https://wes.copernicus.org/articles/8/1387/2023/ (visited on 05/15/2024).

[55] Jaime Liew et al. “Model□free closed□loop wind farm control using reinforcement learning with recursive least squares”. en. In: *Wind Energy* (July 2023), we.2852. ISSN: 1095-4244, 1099-1824. DOI: 10.1002/we.2852. URL: https://onlinelibrary.wiley.com/doi/10.1002/we.2852 (visited on 03/03/2024).

[56] Daria Madjidian and Anders Rantzer. “A Stationary Turbine Interaction Model for Control of Wind Farms”. In: *IFAC Proceedings Volumes*. 18th IFAC World Congress 44.1 (Jan. 2011), pp. 4921–4926. ISSN: 1474-6670. DOI: 10.3182/20110828-6-IT-1002.00267. URL: https://www.sciencedirect.com/science/article/pii/S1474667016443867 (visited on 03/03/2024).

[57] S Mancini and M Caboni. “Towards the use of large eddy simulations for the generation of the atmospheric boundary layer inflow for wind turbine load calculations”. en. In: (2022).

[58] Gilbert M. Masters. *Renewable and Efficient Electric Power Systems*. en. 1st ed. Wiley, July 2004. ISBN: 978-0-471-28060-6 978-0-471-66882-4. DOI: 10.1002/0471668826. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/0471668826 (visited on 05/24/2024).

[59] Attilio Meucci. *Risk and Asset Allocation*. Ed. by M. Avellaneda et al. Springer Finance. Berlin, Heidelberg: Springer, 2005. ISBN: 978-3-540-22213-2 978-3-540-27904-4. DOI: 10.1007/978-3-540-27904-4. URL: http://link.springer.com/10.1007/978-3-540-27904-4 (visited on 06/27/2024).

[60] Ministry of Economic Affairs Netherlands Enterprise Agency and Climate Policy (last). *Hollandse Kust Noord (Site B) Dataset*. en. 2019. URL: https://offshorewind.rvo.nl/files/view/00419133-21ab-470b-bfe6-c010dfc76c66/1567080325hkn_20190815_fugro_hknb_processed%20data.zip (visited on 06/11/2024).

[61] Ranjit R. Nair et al. “Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings”. In: Aug. 2003. URL: https://www.semanticscholar.org/paper/Taming-Decentralized-POMDPs%3A-Towards-Efficient-for-Nair-Tambe/ad684c16d6dd740f3967690c3caa404377c46365 (visited on 04/13/2024).

[62] Ryan Nash, Reza Nouri, and Ahmad Vasel-Be-Hagh. “Wind turbine wake control strategies: A review and concept proposal”. In: *Energy Conversion and Management* 245 (Oct. 2021), p. 114581. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2021.114581. URL: https://www.sciencedirect.com/science/article/pii/S0196890421007573 (visited on 03/02/2024).

[63] 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. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2022, pp. 944–953. ISBN: 978-1-4503-9213-6. (Visited on 02/28/2024).

[64] Takafumi Nishino and Scott Draper. “Local blockage effect for wind turbines”. en. In: *Journal of Physics: Conference Series* 625 (June 2015), p. 012010. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/625/1/012010. URL: https://iopscience.iop.org/article/10.1088/1742-6596/625/1/012010 (visited on 04/17/2024).

[65] NREL. *FLORIS. Version 2.4*. Publication Title: GitHub repository. 2021. URL: https://github.com/NREL/floris/tree/v2.4.

[66] NREL. *FLORIS. Version 3.0*. Publication Title: GitHub repository. 2022. URL: https://github.com/NREL/floris/tree/v3.0.

[67] Frans A Oliehoek, Matthijs T J Spaan, and Shimon Whiteson. "Exploiting Locality of Interaction in Factored Dec-POMDPs". en. In: *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*. May 2008, pp. 517–524.

[68] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-28927-4 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8. URL: http://link.springer.com/10.1007/978-3-319-28929-8 (visited on 02/28/2024).

[69] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. en. 12. print. Cambridge, Mass.: MIT Press, 2006. ISBN: 978-0-262-65040-3.

[70] Sindhu Padakandla, Prabuchandran K. J, and Shalabh Bhatnagar. "Reinforcement Learning in Non-Stationary Environments". en. In: *Applied Intelligence* 50.11 (Nov. 2020). arXiv:1905.03970 [cs, stat], pp. 3590–3606. ISSN: 0924-669X, 1573-7497. DOI: 10.1007/s10489-020-01758-5. URL: http://arxiv.org/abs/1905.03970 (visited on 03/09/2024).

[71] Venkata Ramakrishna Padullaparthi et al. "FALCON- FArm Level CONtrol for wind turbines using multi-agent deep reinforcement learning". In: *Renewable Energy* 181 (Jan. 2022), pp. 445–456. ISSN: 0960-1481. DOI: 10.1016/j.renene.2021.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0960148121013227 (visited on 04/13/2024).

[72] Gregory Palmer. "Independent Learning Approaches: Overcoming Multi-Agent Learning Pathologies In Team-Games". In: University of Liverpool Repository, 2020. DOI: 10.17638/03077940. URL: https://livrepository.liverpool.ac.uk/id/eprint/3077940 (visited on 06/15/2024).

[73] Gregory Palmer et al. *Lenient Multi-Agent Deep Reinforcement Learning*. en. arXiv:1707.04402 [cs]. Feb. 2018. URL: http://arxiv.org/abs/1707.04402 (visited on 05/18/2024).

[74] Razvan Pascanu and Yoshua Bengio. *Revisiting Natural Gradient for Deep Networks*. en. arXiv:1301.3584 [cs]. Feb. 2014. URL: http://arxiv.org/abs/1301.3584 (visited on 04/19/2024).

[75] Samin Payrosangari et al. "Meta-hyperband: Hyperparameter Optimization with Meta-learning and Coarse-to-Fine". en. In: *Intelligent Data Engineering and Automated Learning – IDEAL 2020*. Ed. by Cesar Analide et al. Cham: Springer International Publishing, 2020, pp. 335–347. ISBN: 978-3-030-62365-4. DOI: 10.1007/978-3-030-62365-4_32.

[76] Mats Pedersen et al. "PyWake 2.5.0: An open-source wind farm simulation tool". In: (Feb. 2023). Publisher: DTU Wind, Technical University of Denmark. URL: https://gitlab.windenergy.dtu.dk/TOPFARM/PyWake.

[77] Leonid Peshkin et al. *Learning to Cooperate via Policy Search*. en. arXiv:cs/0105032. May 2001. URL: http://arxiv.org/abs/cs/0105032 (visited on 06/09/2024).

[78] Irving G. B. Petrazzini and Eric A. Antonelo. *Proximal Policy Optimization with Continuous Bounded Action Space via the Beta Distribution*. en. arXiv:2111.02202 [cs]. Nov. 2021. URL: http://arxiv.org/abs/2111.02202 (visited on 06/15/2024).

[79] Kenneth Price, Rainer Storn, and Jouni Lampinen. *Differential Evolution*. en. Natural Computing Series. Berlin/Heidelberg: Springer-Verlag, 2005. ISBN: 978-3-540-20950-8. DOI: 10.1007/3-540-31306-0. URL: http://link.springer.com/10.1007/3-540-31306-0 (visited on 05/07/2024).

[80] Guo-Wei Qian and Takeshi Ishihara. "Wind farm power maximization through wake steering with a new multiple wake model for prediction of turbulence intensity". In: *Energy* 220 (Apr. 2021), p. 119680. ISSN: 0360-5442. DOI: 10.1016/j.energy.2020.119680. URL: https://www.sciencedirect.com/science/article/pii/S0360544220327870 (visited on 04/18/2024).

[81] Manjeet Rani et al. “A review on recycling and reuse methods for carbon fiber/glass fiber composites waste from wind turbine blades”. In: *Composites Part B: Engineering* 215 (June 2021), p. 108768. ISSN: 1359-8368. DOI: 10.1016/j.compositesb.2021.108768. URL: https://www.sciencedirect.com/science/article/pii/S1359836821001608 (visited on 05/01/2024).

[82] Tabish Rashid et al. *QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning*. en. arXiv:1803.11485 [cs, stat]. June 2018. URL: http://arxiv.org/abs/1803.11485 (visited on 05/16/2024).

[83] Aitor Saenz-Aguirre et al. “Artificial Neural Network Based Reinforcement Learning for Wind Turbine Yaw Control”. en. In: *Energies* 12.3 (Jan. 2019). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 436. ISSN: 1996-1073. DOI: 10.3390/en12030436. URL: https://www.mdpi.com/1996-1073/12/3/436 (visited on 07/05/2024).

[84] Jerôme Schalkwijk et al. “A Year-Long Large-Eddy Simulation of the Weather over Cabauw: An Overview”. en. In: *Monthly Weather Review* 143.3 (Mar. 2015), pp. 828–844. ISSN: 0027-0644, 1520-0493. DOI: 10.1175/MWR-D-14-00293.1. URL: http://journals.ametsoc.org/doi/10.1175/MWR-D-14-00293.1 (visited on 03/02/2024).

[85] J. Schreiber, B. Salbert, and C. L. Bottasso. “Study of wind farm control potential based on SCADA data”. en. In: *Journal of Physics: Conference Series* 1037.3 (June 2018). Publisher: IOP Publishing, p. 032012. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1037/3/032012. URL: https://dx.doi.org/10.1088/1742-6596/1037/3/032012 (visited on 03/02/2024).

[86] John Schulman et al. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347 [cs]. Aug. 2017. DOI: 10.48550/arXiv.1707.06347. URL: http://arxiv.org/abs/1707.06347 (visited on 06/16/2024).

[87] John Schulman et al. *Trust Region Policy Optimization*. en. arXiv:1502.05477 [cs]. Apr. 2017. URL: http://arxiv.org/abs/1502.05477 (visited on 04/19/2024).

[88] Javier Serrano González et al. “Optimal wind-turbine micro-siting of offshore wind farms: A grid-like layout approach”. In: *Applied Energy* 200 (Aug. 2017), pp. 28–38. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.05.071. URL: https://www.sciencedirect.com/science/article/pii/S0306261917305676 (visited on 06/26/2024).

[89] Eric Simley, Paul Fleming, and Jennifer King. “Design and analysis of a wake steering controller with wind direction variability”. English. In: *Wind Energy Science* 5.2 (Apr. 2020). Publisher: Copernicus GmbH, pp. 451–468. ISSN: 2366-7443. DOI: 10.5194/wes-5-451-2020. URL: https://wes.copernicus.org/articles/5/451/2020/ (visited on 04/18/2024).

[90] Eric Simley et al. “Wake Steering Wind Farm Control With Preview Wind Direction Information”. In: *2021 American Control Conference (ACC)*. ISSN: 2378-5861. May 2021, pp. 1783–1789. DOI: 10.23919/ACC50511.2021.9483008. URL: https://ieeexplore.ieee.org/abstract/document/9483008 (visited on 04/18/2024).

[91] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. “Learning Without State-Estimation in Partially Observable Markovian Decision Processes”. en. In: Book Title: Machine Learning Proceedings 1994. Elsevier, 1994, pp. 284–292. ISBN: 978-1-55860-335-6. DOI: 10.1016/B978-1-55860-335-6.50042-8. URL: https://linkinghub.elsevier.com/retrieve/pii/B9781558603356500428 (visited on 06/10/2024).

[92] Sara Siniscalchi-Minna et al. “A non-centralized predictive control strategy for wind farm active power control: A wake-based partitioning approach”. In: *Renewable Energy* 150 (May 2020), pp. 656–669. ISSN: 0960-1481. DOI: 10.1016/j.renene.2019.12.139. URL: https://www.sciencedirect.com/science/article/pii/S0960148119320129 (visited on 03/03/2024).

[93] Richard D. Smallwood and Edward J. Sondik. “The Optimal Control of Partially Observable Markov Processes over a Finite Horizon”. In: *Operations Research* 21.5 (Oct. 1973). Publisher: INFORMS, pp. 1071–1088. ISSN: 0030-364X. DOI: 10.1287/opre.21.5.1071. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.21.5.1071 (visited on 02/28/2024).

[94] P. Stanfel et al. “Proof-of-concept of a reinforcement learning framework for wind farm energy capture maximization in time-varying wind”. en. In: *Journal of Renewable and Sustainable Energy* 13.4 (July 2021), p. 043305. ISSN: 1941-7012. DOI: 10.1063/5.0043091. URL: https://pubs.aip.org/jrse/article/13/4/043305/364108/Proof-of-concept-of-a-reinforcement-learning (visited on 03/03/2024).

[95] Paul Stanfel et al. “A Distributed Reinforcement Learning Yaw Control Approach for Wind Farm Energy Capture Maximization”. In: *2020 American Control Conference (ACC)*. ISSN: 2378-5861. July 2020, pp. 4065–4070. DOI: 10.23919/ACC45564.2020.9147946. URL: https://ieeexplore.ieee.org/document/9147946 (visited on 04/13/2024).

[96] Andrew P. J. Stanley et al. “Fast yaw optimization for wind plant wake steering using Boolean yaw angles”. English. In: *Wind Energy Science* 7.2 (Mar. 2022). Publisher: Copernicus GmbH, pp. 741–757. ISSN: 2366-7443. DOI: 10.5194/wes-7-741-2022. URL: https://wes.copernicus.org/articles/7/741/2022/ (visited on 05/05/2024).

[97] Rainer Storn and Kenneth Price. “Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces”. In: *Journal of Global Optimization* 23 (Jan. 1995).

[98] Peter Sunehag et al. *Value-Decomposition Networks For Cooperative Multi-Agent Learning*. en. arXiv:1706.05296 [cs]. June 2017. URL: http://arxiv.org/abs/1706.05296 (visited on 05/16/2024).

[99] Richard S Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999. URL: https://papers.nips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html (visited on 03/01/2024).

[100] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. en. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, 2020. ISBN: 978-0-262-03924-6.

[101] G. I. Taylor. “The Spectrum of Turbulence”. In: *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* 164.919 (Jan. 1997). Publisher: Royal Society, pp. 476–490. DOI: 10.1098/rspa.1938.0032. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.1938.0032 (visited on 03/03/2024).

[102] Yee Whye Teh et al. *Distral: Robust Multitask Reinforcement Learning*. en. arXiv:1707.04175 [cs, stat]. July 2017. URL: http://arxiv.org/abs/1707.04175 (visited on 04/19/2024).

[103] Jared J. Thomas et al. “A comparison of eight optimization methods applied to a wind farm layout optimization problem”. English. In: *Wind Energy Science* 8.5 (June 2023). Publisher: Copernicus GmbH, pp. 865–891. ISSN: 2366-7443. DOI: 10.5194/wes-8-865-2023. URL: https://wes.copernicus.org/articles/8/865/2023/ (visited on 04/13/2024).

[104] Maarten Paul Van Der Laan, Mads Baungaard, and Mark Kelly. *Brief communication: A clarification of wake recovery mechanisms*. en. July 2022. DOI: 10.5194/wes-2022-56. URL: https://wes.copernicus.org/preprints/wes-2022-56/wes-2022-56.pdf (visited on 06/15/2024).

[105] P. Vatiwutipong and N. Phewchean. “Alternative way to derive the distribution of the multivariate Ornstein–Uhlenbeck process”. en. In: *Advances in Difference Equations* 2019.1 (Dec. 2019), p. 276. ISSN: 1687-1847. DOI: 10.1186/s13662-019-2214-1. URL: https://advancesindifferenceequations.springeropen.com/articles/10.1186/s13662-019-2214-1 (visited on 06/11/2024).

[106] L. J. Vermeer, J. N. Sørensen, and A. Crespo. “Wind turbine wake aerodynamics”. In: *Progress in Aerospace Sciences* 39.6 (Aug. 2003), pp. 467–510. ISSN: 0376-0421. DOI: 10.1016/S0376-0421(03)00078-2. URL: https://www.sciencedirect.com/science/article/pii/S0376042103000782 (visited on 03/02/2024).

[107] Timothy Verstraeten, Pieter JK Libin, and Ann Nowé. *Fleet Control using Coregionalized Gaussian Process Policy Iteration*. en. arXiv:1911.10121 [cs, eess, stat]. Nov. 2019. URL: http://arxiv.org/abs/1911.10121 (visited on 03/03/2024).

[108] Sanjana Vijayshankar et al. “Deep Reinforcement Learning for Automatic Generation Control of Wind Farms”. In: *2021 American Control Conference (ACC)*. New Orleans, LA, USA: IEEE, May 2021, pp. 1796–1802. ISBN: 978-1-66544-197-1. DOI: 10.23919/ACC50511.2021.9483277. URL: https://ieeexplore.ieee.org/document/9483277/ (visited on 03/03/2024).

[109] Kejia Wan, Xinhai Xu, and Yuan Li. “Learning Distinct Strategies for Heterogeneous Cooperative Multi-agent Reinforcement Learning”. en. In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Ed. by Igor Farkaš et al. Cham: Springer International Publishing, 2021, pp. 544–555. ISBN: 978-3-030-86380-7. DOI: 10.1007/978-3-030-86380-7_44.

[110] Ermo Wei and Sean Luke. “Lenient learning in independent-learner stochastic cooperative games”. In: *The Journal of Machine Learning Research* 17.1 (Jan. 2016), pp. 2914–2955. ISSN: 1532-4435.

[111] Miao Weipao et al. “Numerical Investigation of Wake Control Strategies for Maximizing the Power Generation of Wind Farm”. In: *Journal of Solar Energy Engineering* 138.034501 (Apr. 2016). ISSN: 0199-6231. DOI: 10.1115/1.4033110. URL: https://doi.org/10.1115/1.4033110 (visited on 03/02/2024).

[112] R. P. Wiegand et al. “An Analysis of Cooperative Coevolutionary Algorithms A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University”. In: 2003. URL: https://www.semanticscholar.org/paper/An-Analysis-of-Cooperative-Coevolutionary-A-in-of-Wiegand-Jong/f2905fba1c104c59ee1007de9be013618dda7ae2 (visited on 05/15/2024).

[113] Ronald J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. en. In: *Machine Learning* 8.3 (May 1992), pp. 229–256. ISSN: 1573-0565. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696 (visited on 03/01/2024).

[114] Christian Schroeder de Witt et al. *Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?* arXiv:2011.09533 [cs]. Nov. 2020. DOI: 10.48550/arXiv.2011.09533. URL: http://arxiv.org/abs/2011.09533 (visited on 03/03/2024).

[115] D. H. Wolpert and K. Tumer. “Optimal Payoff Functions for Members of Collectives”. In: *Advances in Complex Systems* 4.2/3 (2001), pp. 265–279.

[116] Jingjie Xie et al. “Wind Farm Power Generation Control Via Double-Network-Based Deep Reinforcement Learning”. In: *IEEE Transactions on Industrial Informatics* 18.4 (Apr. 2022). Conference Name: IEEE Transactions on Industrial Informatics, pp. 2321–2330. ISSN: 1941-0050. DOI: 10.1109/TII.2021.3095563. URL: https://ieeexplore.ieee.org/abstract/document/9478204 (visited on 07/05/2024).

[117] Jian Yang et al. “Review of control strategy of large horizontal-axis wind turbines yaw system”. en. In: *Wind Energy* 24.2 (2021), pp. 97–115. ISSN: 1099-1824. DOI: 10.1002/we.2564. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2564 (visited on 04/11/2024).

[118] Shanghui Yang et al. “Cooperative yaw control of wind farm using a double-layer machine learning framework”. In: *Renewable Energy* 193 (June 2022), pp. 519–537. ISSN: 0960-1481. DOI: 10.1016/j.renene.2022.04.104. URL: https://www.sciencedirect.com/science/article/pii/S0960148122005729 (visited on 07/05/2024).

[119] Chongjie Zhang, Sherief Abdallah, and V. Lesser. “Integrating organizational control into multi-agent learning”. en. In: *Adaptive Agents and Multi-Agent Systems* (2009). URL: https://www.semanticscholar.org/paper/Integrating-organizational-control-into-multi-agent-Zhang-Abdallah/d5de2965923bc35e03bbf3503008d3f08508804c (visited on 05/16/2024).

[120] Chongjie Zhang and Victor Lesser. “Coordinated Multi-Agent Reinforcement Learning in Networked Distributed POMDPs”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 25. ISSN: 2374-3468, 2159-5399 Issue: 1 Journal Abbreviation: AAAI. Aug. 2011, pp. 764–770. DOI: 10.1609/aaai.v25i1.7886. URL: https://ojs.aaai.org/index.php/AAAI/article/view/7886 (visited on 05/16/2024).

[121] Huan Zhao et al. “Cooperative Wind Farm Control With Deep Reinforcement Learning and Knowledge-Assisted Learning”. In: *IEEE Transactions on Industrial Informatics* 16.11 (Nov. 2020). Conference Name: IEEE Transactions on Industrial Informatics, pp. 6912–6921. ISSN: 1941-0050. DOI: 10.1109/TII.2020.2974037. URL: https://ieeexplore.ieee.org/abstract/document/8999726 (visited on 07/05/2024).

[122] Meng Zhou et al. “Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11853–11864. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/8977ecbb8cb82d77fb091c7a7f186163-Paper.pdf.

A

Hyperparameters

A.1. Static Yaw Optimisation

This section contains the hyperparameters used for numeric static yaw optimisation in Chapter 4

A.1.1. Sequential Least Squares Programming

For sequential least squares programming, we use the FLORIS V2.4 [65] default parameters: 50 iterations, a tolerance of 10^{-12}, and a step size of 0.1 for approximation of the Jacobian. As well as normalising the yaw angles to [0, 1] and dividing the power output (objective) by the initial power output.

A.1.2. Differential Evolution

To set the hyperparameters for differential evolution we use the rules of thumb given by Price, Storn, and Lampinen [79]. As yaw optimisation is a multi-modal problem with dependencies between variables, we set the mutation rate F to dither between 0.5 and 1, the recombination constant CR to 0.9, and the population size to $20N$ where N is the dimensionality of the problem. For convergence testing, an absolute tolerance of 0.01 MW is used.

A.1.3. Alternating Yaw Optimisation

For the gradient-based alternating yaw optimisation method we employed the sequential least squares programming optimiser using the previously described FLORIS default parameters. For the coarse-to-fine sampling-based method, we used a coarse sampling resolution of 3°, which results in $z = (y_{\max} - y_{\min})/3° + 1 = 21$, a candidate search radius r of 1.5°, and a fine sampling resolution Δy of 0.1°.

A.2. Dynamic Yaw Control

Both policy gradient methods were allocated a total of 97200 time steps (seconds) to train 9 agents. For distributed policy gradient we used 12 episodes each containing 1 trajectory, resulting in a trajectory length of $97200/12 = 8100$ time steps. For alternating policy gradient, we used 12 iterations of alternating maximisation, each iteration was split up into 9 episodes (one for each agent) with 1 trajectory per episode, resulting in a trajectory length of $97200/12/9 = 900$ time steps. The remaining hyperparameters are summarised in the following table:

Parameter	Value
γ (Discount Factor)	0.3
Policy Network Layers	[64, 64]
Policy Gradient Method	REINFORCE [113]
Activation Function	Tanh
Policy Distribution	Beta
Optimiser	Adam
Learning Rate	0.01
β_1, β_2 (Betas)	[0.9, 0.999]
ϵ (Epsilon)	10^{-8}
λ (Weight Decay)	0

Table A.1: General hyperparameters for the dynamic yaw control experiments.

A.2.1. Random Initialisation

Each alternating policy gradient episode of 900 time steps is split up into 3 trajectories of 300 time steps. Each trajectory starts from a random initial yaw chosen from the set $\{-30°, 0°, 30°\}$.

A.2.2. Oscillating Exploration

Each alternating policy gradient episode of 900 time steps is split up into 3 trajectories of 300 time steps. The first trajectory of each episode utilises the oscillating exploration strategy using the turbine's maximum angular velocity $\omega_{\max}$ of $1\,°\,s^{-1}$.