SISSA
Scuola
Internazionale
Superiore di
Studi Avanzati

Physics Area - PhD course in
Physics and Chemistry of Biological Systems

Learning to fly exploiting complex wind fields

Candidate:
Claudio Leone

Advisor:
Antonio Celani

Academic Year 2021-22

Contents

Introduction

The work of this thesis is focused on showing how Machine Learning proves effective in predicting the behaviour of complex wind velocity fields. On a large scale the motion of winds is the object of study of meteorology, as the precision of weather forecasting depends on the ability to determine the evolution of the air currents conditioned by the distribution of atmospheric pressure and temperature around the globe. If we move from a global to a local perspective, aerodynamics studies the impact of the air flow on structures like wings and sails and the forces that arise from this interaction. This leads to the derivation of empirical laws describing how such forces depend on the shape and velocity of the objects we are interested in. What is captivating for us is how to exploit the power that can be extracted from the wind in order to use it for different tasks that span from energy production to motion through the air velocity field.

As far as the former is concerned, the most common technology nowadays in use to convert eolic energy into electricity contemplates the employment of wind towers, conveniently placed in windswept regions in the land or off-shore. In fact, the first two chapters of this thesis work regard a recent alternative to wind towers which goes under the name of Airborne Wind Energy [1]: it consists in the extraction of usable power from the wind using a flying object - like a kite or a glider - connected with a tether to a station on the ground or on a vehicle.

Therefore, wind power is commonly used for many human chores that extend from sailing and paragliding, where the work of the aerodynamic force is the primary source of energy, to flying planes, where the lift acting on the wings is generated thanks to the engine propulsion on the aircraft. The usage of Airborne Wind Energy to travel from point A to point B in space pertains mainly to the first category, utilizing tethered kites to carry a vehicle on the ground or on the sea, resembling the popular activity of kite surfing (fig. 1a).

Moreover, many animal species are able to exploit the behaviour of air flows in order to move efficiently from one place to another: this is especially known for migrating birds, being capable of using thermals to reduce their energy expenditure during their intercontinental movements [2], but it is quite common among spiders too, which, even without any wing apparatus, take advantage of the drag force exerted on their silk lines in order to get dispersed from their birthplace. The last chapter of this thesis is deputed to investigate this phenomenon, called *spider ballooning* (fig. 1b).

In both cases the main question we pose is the following: which is the best strategy

Figure 1: a) An example of application of Airborne Wind Energy technology for ship transport by the French company SkySails. The kite uses the wind to reduce the fuel consumption by the diesel engine, contributing to pull the ship across the sea. b) A spider on the verge of taking off and perform ballooning: the air can drag the silk filament, which works as sail and makes the spider become airborne.

in order to maximize the power extraction from the wind velocity field? This translates into choosing the best time to take-off for the spider in order to fly as far as possible. Similarly, for a kite-vehicle set-up, it means looking for the set of actions which allows for the maximum displacement of the vehicle.

This question underlines the possibility to formulate both tasks as optimization processes: they consist in the individuation of the decisions that maximize a target function, such as harvested power.

The quest for optimality

Striving for optimality is the main drive in a large variety of contexts: it can be potentially characterized as an organizing principle in biology as well as a design principle for artificial forms of intelligent systems. Therefore, looking for optimality principles is particularly appealing from the point of view of evolution, as it can be argued that behaviour is shaped the way it is by natural selection.

Mathematical optimization has a long history that deeps its roots into the origins of differential calculus with the work of Fermat and Lagrange. This subject found astonishing development in the 20th century, stimulated by progress and increase in complexity especially in the field of logistics: the theory of *linear programming* - meant to maximize or minimize a linear function under a set of constraints - was independently formulated by Dantzig and Kantorovich to solve planning and cost problems in the U.S. and Soviet army respectively [3, 4]. However, numerous new branches of research proliferated as the functions to be optimized not necessarily respected linearity requirements or they contained randomness. More specifically, the work of pioneers Richard Bellman and Lev Pontryagin [5] among others shed light on the branch of mathematical optimization that deals with finding a control for a dynamical system over a period of time such that an

objective function is optimized, going under the name of Optimal Control Theory ever since.

Bellman specifically introduced *dynamic programming* as a method for simplifying complex decision problems by decomposing them into smaller pieces in a recursive fashion [6]. Dynamic programming is widely considered the only feasible way of solving general stochastic optimal control problems. Such brilliant piece of research constitutes the basis of other techniques to solve decision-making problems.

Among those, the late 1980s saw the birth of Reinforcement Learning. In the words of its creators Richard Sutton and Andrew Barto [7], Reinforcement Learning was meant to be the union of two threads: one descending directly from the lesson by Bellman and can be conceived as a generalization of dynamic programming theory; the other draws inspiration from behavioural psychology especially in the works of Skinner and Pavlov, adding to the first one the possibility to learn from experience, which was absent in Optimal Control Theory.

In fact Reinforcement Learning proves very useful in dealing with optimal control tasks with the great advantage of not needing the knowledge of the dependence of the objective function on the dynamical system in study, shifting instead its focus towards data elaboration and analysis.

Reinforcement Learning as an optimization technique

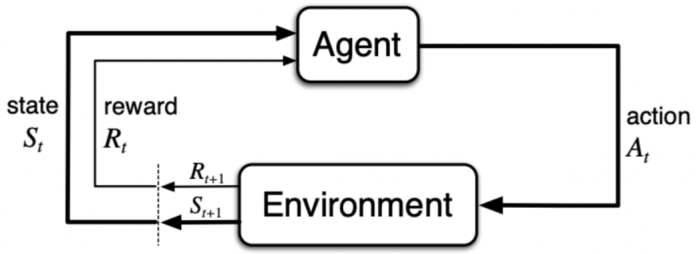

Figure 2: **Reinforcement Learning scheme**. The core of the Reinforcement Learning framework is reported in this scheme. The subject of the learning is called *agent* and it interacts with the external environment via actions, which provide feedback in the form of rewards and observation of the new state.

Reinforcement Learning is generally regarded as a branch of the wider field of Machine Learning. Its working cycle is very intuitive and it is reported in fig. 2. Differently from Supervised and Unsupervised learning, it is best suited for contexts in which it is necessary to interact with a dynamical environment which provides the algorithm with a stream of data to be analyzed *online*. The subject of the learning, called *agent*, interacts with the surrounding environment by performing actions which provide it with a feedback in the form of a reward signal and of information about its condition. As the terminology is evidently borrowed from neuro-psychology, the agent is pushed to learn

in the same flavour of its human/animal counterpart, understanding which actions will lead it to the highest future rewards, making this scheme particularly suitable to address optimization problems.

The scope of Reinforcement Learning virtually encompasses all possible decision-making problems, from those pertaining to the regime of the aforementioned optimal control – where the agent is assumed to know everything about the state of the world – to those in which partial information about the environment is accessible to the agent (fig. 3). Beyond providing solutions to decision-making problems, Reinforcement Learning – as it might be obvious from the name – offers the algorithms for the learning process. Not only, then, one might hope to draw conclusions about the optimality of the behaviours that one observes but also about the computations required to learn from experience.

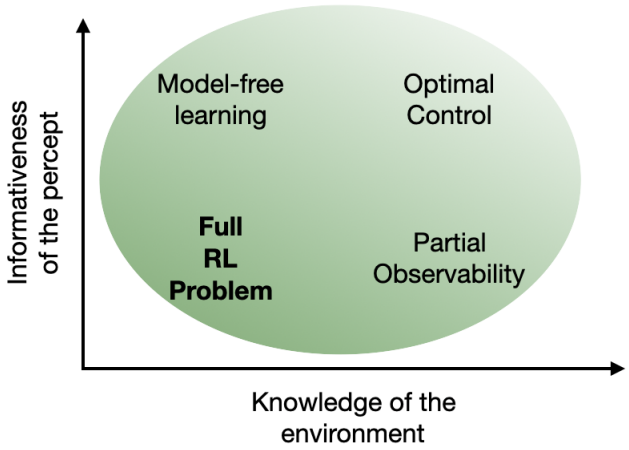

Figure 3: **Reinforcement Learning areas of application**. Reinforcement Learning encompasses a wide range of decision-making problem, which can be characterized on the basis of the knowledge of the laws governing the environment (which increases moving along the x-axis) and on the amount of information available to the agent about the state of the environment (y-axis). Optimal Control Theory corresponds to the special case in which a model of the dynamics is assumed to be known by the agent as well as the full configuration of the system. In this realm optimal solutions are calculated from the prediction of future outcomes according to a known model of the dynamics. Whenever the information coming from the environment is incomplete we move downwards on the y-axis towards partial observability. Going towards the origin we move away from model-based approaches - in that they require the complete knowledge of the laws of Nature - to model-free learning. The work in this thesis will in fact focus on problems in which there is no *a priori* knowledge of either the state of the environment nor the model of its evolution, pertaining to the region labeled by the full Reinforcement Learning problem.

The independence of Reinforcement Learning from *a priori* knowledge about the surrounding environment makes it particularly suitable for our purposes. The learning algorithm and the dynamics of the system can be developed separately as the goodness

of predictions made by the algorithm exclusively depends on data collected from the environment and not on its inner laws. This accounts for major generality leaving space for application to different architectures and in diverse conditions.

With the techniques derived from Reinforcement Learning and from Machine Learning widely speaking, we are given a powerful tool to exploit wind energy without the need of recurring to over-simplified models of the wind flow. Our approach is two-fold: if on one hand we naturally aim at performance, being capable of efficiently harvesting complex velocity fields, on the other we plan to maintain a physicist's perspective on Machine Learning, asking ourselves what are the relevant features that lead to success searching for motif behind good results.

Learning to fly using complex wind fields

As we anticipated at the beginning, the tasks that we want to tackle are related to the exploitation of wind energy in order to move efficiently in the velocity field avoiding crashing or delaying the landing as much as possible.

The control of Airborne Wind Energy systems is an open and challenging problem and state-of-the-art research on this topic addresses it with a technique called Non-Linear Model Predictive Control. This allows for optimizing the power functional within a certain time horizon and being able to integrate new information while operating. In light of what we argued before, we believe in the potential of a Reinforcement Learning-driven approach to this matter.

As we will see, its model-free approach allows us to deal with a higher degree of complexity in the environment without operating approximations, which are instead inevitable in the standard control frame. One driving example is the presence of turbulence, which is an essential and unavoidable feature of natural flow on one hand but on the other it is intractable for traditional control methods since it would require a huge number of degrees of freedom to model it, resulting in heavy computations with very little predictive power.

Moreover, it is straight-forward to frame the structure of the Airborne Wind Energy extraction problem in the scheme of fig. 2. The purpose is to keep the kite flying as information from its status is continuously flowing in providing useful data to the control algorithm. Reacting to these observations, the algorithm must take actions with the purpose of maximizing the harvested energy in the long run. This task obviously requires continuous interplay between the agent, which we can individuate with the software deputed to the decision-making, and the environment surrounding the kite.

Regarding spider ballooning instead, the goal is the same but the available actions are somewhat limited. In fact the spider has no way to control its flight after becoming airborne. It follows that the decision lies in the choice of take-off time that leads to the most profitable results in terms of traveled distance. Previous quantitative works on this phenomenon are limited to modeling, as in our knowledge there are no past attempts of understanding the optimization process underlined by the search for the best choice for the spider to become airborne.

This structure allows for a slightly simpler treatment, at least in terms of a preliminary analysis on which velocity time-series are more suited for a successful flight in the future. This is why we initially turn to a logistic regression model in order to recognize the features of profitable local wind histories starting from labeled data of past flights.

The thesis is divided into three chapters. In chapter 1 Airborne Wind Energy is introduced, focusing on its employment in towing a ship across the sea. We highlight the success of a simple Reinforcement Learning algorithm in navigating the ship-kite system efficiently in a turbulent wind flow with a low-dimensional input. All simulations are performed in a virtual environment and it is possible to come out with an effective set of actions sufficient to reproduce the best strategy and understand its main aspects. The relevant observables are selected resorting to expert domain knowledge on the system we analyze and this choice gives fruitful results in terms of performance, with a limited price in complexity and memory usage. The content of this chapter has been submitted to arxiv.org.

Chapter 2 analyzes more deeply the reasons for the success of this choice. We study a slightly more refined architecture for energy harvesting as a test bench for two different algorithms: the tabular one employed in chapter 1, underlining the generality proper of model-free learning among different structures, and a second one involving a simple feed-forward deep-network. Even if more articulate, the second program does not overpower the results of the first one, but provides a deeper insight on its strength, partially answering to the reasons of its success.

Finally, chapter 3 faces the phenomenon of ballooning spiders. We develop a new model for the dynamics of the airborne spider and we study the decision-making process at the basis of its take-off action. In this case the optimization process in nature does not happen on the time-scale of the individual, as the ability to respond to the local wind conditions appears plugged in the genetics of the spider species that perform ballooning. It is one example of how evolution can be framed in terms of optimization, developing one trait that favours fast dispersion of spiderlings, in order to avoid starvation and cannibalism.

Chapter 1

Reinforcement Learning Control for Ship-towing in a Turbulent Environment

Airborne Wind Energy is a lightweight technology that allows power extraction from the wind using kites. At variance with standard wind turbines, the airfoil orientation can be dynamically controlled in order to maximize performance. The dynamical complexity of kite aerodynamics in the turbulent atmosphere makes this problem unapproachable by conventional methods such as optimal control theory, which rely on an accurate and tractable analytical model of the dynamical system. Here we propose to attack this problem through Reinforcement Learning, a technique that – by repeated trial-and-error interactions with the environment – learns to associate observations with profitable actions without invoking prior knowledge of the system. We show that in a simulated fully-turbulent atmosphere Reinforcement Learning finds an efficient way to tow a vehicle for long distances by controlling a flying kite. The algorithm we use is based on a small set of intuitive observations and its physically transparent interpretation allows us to cast the approximately optimal strategy as a simple set of manoeuvring instructions.

1.1 Introduction

Airborne wind energy (AWE) is a technology aiming to obtain usable power by means of flying devices [1] which has the potential of replacing the traditional towered wind turbine architecture [8]. Airborne wind energy systems usually consist in an airfoil providing traction power - generally a kite or a glider - which is either connected by a tether to a ground station that converts power into electricity by a turbine, or used to tow a vehicle [9, 10].

There are many advantages of AWE over standard wind turbines: reduced costs for construction; lower environmental impact; saving in building materials; ease in displacing the device [11, 12]. Here we focus on one distinguishing feature of AWE that is the

possibility of adapting to the rapid and local changes in wind conditions by controlling the orientation of the kite. This allows to maintain the kite airborne and effective under varying wind and weather conditions [13].

The task of finding the best way of manoeuvring the kite in order to maximise power production has been previously addressed by means of optimal control theory [14, 15, 16], using methods from Non-linear Model Predictive Control [17]. These are essentially planning algorithms that crucially rely on a dynamical model in order to predict the future evolution of the system. For AWE this requirement translates in finding an accurate model of both the dynamics of the kite and of the turbulent wind. Such model would involve a huge number of dynamical degrees of freedom and the resulting optimization problem would then become computationally intractable. Moreover, even assuming that the previous difficulties could be overcome, turbulence is famously characterized by its unpredictability, undercutting the power of predictive control. To bypass these difficulties, the effect of wind turbulence is often ignored or too crudely approximated, for instance by adding small uncorrelated fluctuations [18, 11, 19]. The control strategies so obtained may however turn out to be severely suboptimal when put to the test in a realistic turbulent environment.

Here, we propose to use Reinforcement Learning (RL) to find effective control strategies for AWE in a realistic turbulent environment. The core idea of RL consists in learning how to control a system in order to achieve a long-term goal without relying on detailed *a priori* knowledge of its dynamics and therefore avoiding the shortcomings of predictive control [7]. By repeatedly interacting with the environment that surrounds it, the controller learns by trial and error to associate valuable actions to specific contexts [20]. RL has been shown to be competitive with Model Predictive Control even in situations where an accurate model of the system is available [21]. The ability of RL in finding effective control strategies in dynamic and unpredictable environment such as the turbulent atmosphere has been recently showcased in complex tasks such as thermal soaring and balloon navigation [22, 23].

In this chapter we provide a proof of concept that RL can be successfully used for AWE. We consider a simulated environment that simulates the dynamics of a ship towed by a kite that flies in the turbulent atmospheric boundary layer. A major difficulty of this task is to find appropriate controllers that sense and exploit the variable wind strength and at the same time avert disastrous crashes. We find that a compact RL algorithm based on a small set of physically intuitive observations and controls is able to find effective strategies to tow the vehicle for long distances, efficiently converting the energy of the turbulent wind into directed motion.

1.2 The virtual environment

In this section we describe the virtual environment that simulates the kite dynamics in a turbulent flow.

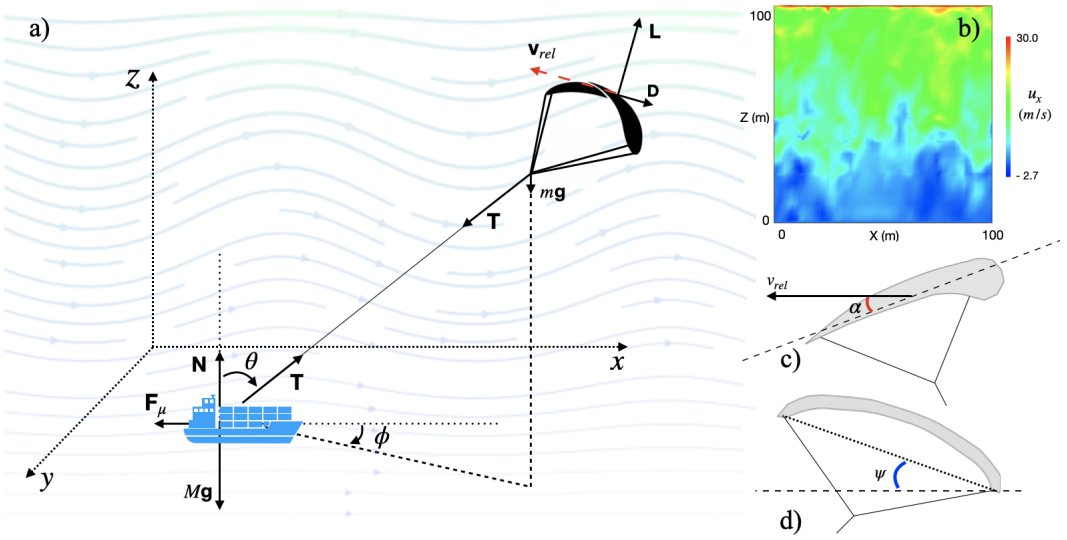

Figure 1.1: a) Sketch of the kite-ship system. b) Snapshot of the horizontal wind velocity in the turbulent flow on the x-z plane. c) The attack angle α is the angle between the longitudinal axis of the kite and the relative velocity; its control allows the kite to sink and rise. d) The bank angle ψ changes the direction of the lift force and its control makes the airfoil turn left and right.

1.2.1 Kite and vehicle dynamics

We consider the system composed by a kite connected to a vehicle (e.g. a ship) through an inextensible cable. The vehicle can move on the surface, while the kite is able to travel on the whole space above it. Sagging of the rope is neglected as the cable is understood to be always in tension.

The kite has mass m and is subject to the action of the wind, summarized by the total aerodynamic force $\boldsymbol{F}^{aer}$, the gravitational force $m\boldsymbol{g}$ and the tether tension $\boldsymbol{T}$. The vehicle of mass M is subject to the gravitational force $M\boldsymbol{g}$, the same tension $-\boldsymbol{T}$ acting on the other end of the rope, the sliding friction force $\boldsymbol{F}^{\mu}$ and the normal reaction of the ground/sea surface $\boldsymbol{N}$ (see Fig. 1.1)

The positions of the kite $\boldsymbol{x}_k$ and of the vehicle $\boldsymbol{x}_v$ obey the Newton's law

$$\begin{cases} m\ddot{\boldsymbol{x}}_k = \boldsymbol{F}^{aer} + m\boldsymbol{g} - \boldsymbol{T} \\ M\ddot{\boldsymbol{x}}_v = \boldsymbol{F}^{\mu} + \boldsymbol{N} + M\boldsymbol{g} + \boldsymbol{T} \end{cases} \tag{1.1}$$

subject to the constraints of fixed tether length $|\boldsymbol{x}_k - \boldsymbol{x}_v| = R$ and that the vehicle remains anchored to the ground $z_v = 0$.

The friction force has amplitude $F^{\mu} = \mu N = \mu(Mg - T_z)$ where T_z is the vertical component of the tension applied to the vehicle. Its direction is opposed to the vehicle velocity $\dot{\boldsymbol{x}}_v$.

The aerodynamic forces are customarily decomposed in *lift* and *drag*: $\boldsymbol{F}^{aer} = \boldsymbol{L} + \boldsymbol{D}$. Both components depend on the relative velocity of the kite with respect to the wind $\boldsymbol{v}_{rel} = \boldsymbol{v}_k - \boldsymbol{u}$, where $\boldsymbol{u}$ is the wind velocity and $\boldsymbol{v}_k = \dot{\boldsymbol{x}}_k$.

The amplitudes of lift and drag are $L = \frac{1}{2}\rho A C_L(\alpha)|\boldsymbol{v}_{rel}|^2$ and $D = \frac{1}{2}\rho A C_D(\alpha)|\boldsymbol{v}_{rel}|^2$ respectively, where ρ is the air density, A is the kite surface area and $C_L(\alpha)$, $C_D(\alpha)$ are the lift and drag coefficients. The latter depend on the *attack angle* of the kite α (fig. 1.1b), which is the angle between the longitudinal axis of the kite and the relative velocity. By changing the attack angle the kite can rise or dive. The values of the coefficients are measured empirically for different airfoils and here we used the ones given in Ref. [24] for our simulation.

As for the direction, the drag is anti-parallel to the relative velocity whereas the lift lies in the plane perpendicular to it. Its exact orientation depends on the *bank* angle ψ (fig. 1.1c), which can be controlled and allows for the kite to make turns. Namely, if we take $\boldsymbol{e}_r = (\boldsymbol{x}_k - \boldsymbol{x}_v)/R$ to be the unit vector which identifies the direction of the tether and define the two unit vectors perpendicular to the relative velocity $\boldsymbol{e}_t = \boldsymbol{e}_r \times \boldsymbol{v}_{rel}/|\boldsymbol{e}_r \times \boldsymbol{v}_{rel}|$ and $\boldsymbol{e}_n = \boldsymbol{v}_{rel} \times \boldsymbol{e}_t/|\boldsymbol{v}_{rel} \times \boldsymbol{e}_t|$ we have $\boldsymbol{L} = L\left(\boldsymbol{e}_t \sin(\psi) + \boldsymbol{e}_n \cos(\psi)\right)$ (see Ref. [16] and Appendix A for more details on the dynamics).

We assume that the kite can be controlled by changing its attack and bank angles - and consequently its orientation in space - by operating on the lines that are connected to the sides of the airfoil.

1.2.2 The turbulent environment

We simulate the wind dynamics in the atmospheric boundary layer by solving the incompressible Navier-Stokes equations

$$\begin{cases} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u} \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \tag{1.2}$$

in a cubic domain of side ≈ 100 m. No-slip boundary conditions are imposed at the bottom and a fixed velocity $U_W = 30$ m/s, directed along the streamwise direction x, is imposed at the top. Periodic boundary conditions are applied in the stream-wise x and span-wise y directions. This configuration is usually dubbed a turbulent Couette flow. The Reynolds number $Re = \frac{U_W \delta}{2\nu}$, where δ is the half-height of the channel and ν is the kinematic viscosity, is $Re = 16400$ which indicates the presence of a turbulent flow. The simulation uses a Spectral Element Method as implemented in the free CFD software Nek5000 [25]. At the steady state, the statistical properties of the flow are consistent with those observed in previous works [26] (see Appendix A for more details).

1.3 Learning to control the kite

As anticipated in the Introduction, we have approached the problem of maximising the traction power of a kite in a turbulent flow by means of Reinforcement Learning. In this

Section we give a brief description of the algorithm that we have used and how it has been implemented in the case at hand.

1.3.1 Reinforcement Learning

The objective of RL is to optimize the performance of a controller, or agent, in a generic goal-directed task (see Ref. [7] for a comprehensive introduction to the subject). At each time step, the controller receives an observation S that provides some information about the current, possibly hidden, full state of the system X and takes an action A according to a *policy* π, that is a probability distribution over actions which depends on the history of previous observations. The system then evolves in a new state X' and the process repeats itself until the end of an *episode* of duration T. At each time step, the agent receives a reward, or reinforcement signal, R. The objective is to find a policy that maximizes the expected sum of future rewards, known as the *return* $G_t = \sum_{k=t}^{T} R_k$.

Importantly, the agent has no *a priori* knowledge of the dynamics of the hidden states $X \to X'$ and of the structure of the rewards. It has to learn how to associate past observations with profitable actions just by interacting with the environment.

Finding a policy that depends on the full history of observations is computationally very demanding. Here we will settle for the comparatively less ambitious goal of searching for good *reactive policies*, that is strategies that choose to take an action a based only on the last observation s according to a policy $\pi(a|s)$. If the observations were the actual Markov states for the system, i.e. $S = X$, then the knowledge of past observations is superfluous and one can consider only reactive policies. However, this nearly never happens in practice.

Approximately optimal reactive policies can be found with limited computational effort when observations and actions can be represented by a finite discrete set. In this case tabular time-difference learning algorithm such as SARSA (the acronym of State-Action-Reward-State-Action) can find good strategies by interacting with the environment.

SARSA starts off with an estimate of the expected return that a policy can achieve starting with an observation s and an action a: $\hat{Q}(s, a)$. Then, at each step the algorithm improves the estimate and the policy as follows:

i) Derive a policy from the current estimate $\hat{Q}$ with $\epsilon-$*greedy* exploration: given an observation S pick the action which has the largest estimated return $A = \text{argmax}_a \hat{Q}(S, a)$ with probability $1-\epsilon$ and any possible action at random otherwise;

ii) After having observed S' and taken action A' according to the same policy, update the estimate as $\hat{Q}(S, A) \leftarrow \hat{Q}(S, A) + \eta(R + \hat{Q}(S', A') - \hat{Q}(S, A))$. All the other entries of the matrix $\hat{Q}$ are left unchanged.

This procedure is iterated until convergence is obtained.

In the expressions above, the learning rate η and the exploration probability ϵ can in general depend on the current state-action pair as well as on the total number of iterations. With a proper scheduling of these parameters, SARSA converges with probability

one to the optimal control strategy if observations are Markov states for the system. Otherwise, it converges to the best possible reactive strategy for the given choice of observables.

In order to couch our AWE optimization problem in the language of reinforcement learning we now turn to define the observables S, the actions A and rewards R for our AWE system.

1.3.2 Actions

As discussed earlier, the kite can rise or dive by changing the angle of attack α and it can turn by modifying the bank angle ψ. Therefore an intuitive and minimal way to control the kite is to decrease, leave unchanged or increase both α and ψ by some fixed amount:

$$\begin{array}{ll} \alpha \to \alpha + A_\alpha \Delta\alpha & A_\alpha \in \{-1, 0, 1\} \\ \psi \to \psi + A_\psi \Delta\psi & A_\psi \in \{-1, 0, 1\} \end{array} \tag{1.3}$$

The total number of possible actions $A = (A_\alpha, A_\psi)$ is 9 and the values of the increments $\Delta\alpha$ and $\Delta\psi$ are chosen based on the aerodynamic characteristic of the kite (see Appendix A).

1.3.3 The choice of observables

The actual state X of an AWE device immersed in a turbulent flow is an extremely high-dimensional object as it includes all the degrees of freedom of the kite and the vehicle, as well as the wind field at any point in space and time. A direct approach is clearly unfeasible. It is therefore necessary to identify some relevant features, that is to map the full state X into a smaller set of observables S, which can summarize the state of the system without losing valuable information.

The selection of such observables could be in principle performed automatically. However this procedure requires very large training datasets that are very expensive to obtain. In addition, the selected features so obtained might be difficult to interpret in physical terms. Here we take the opposite approach of choosing observations based on some physical intuition about which variables matter the most in achieving good control of the system. This approach requires expert domain knowledge and may fail if relevant information is overlooked. However, in our experience so far, the benefits of a physics-informed approach in terms of speed of learning, data parsimony and interpretability largely outweigh the cost of reduced performance.

Following this idea, we expect that the aerodynamic forces are a determinant factor for the ability of controlling the kite. As discussed in the previous section, lift and drag depend on the angle of attack α, on the bank angle ψ, and on the relative velocity $\boldsymbol{v}_{rel}$ and it is therefore natural to consider them as relevant observables. In a further effort to compress the input we will consider just the orientation of the relative velocity with respect to the ground, given by the angle $\beta = \arcsin(v^z_{rel}/|\boldsymbol{v}_{rel}|)$. These variables are

then discretized to obtain a finite set of observables

$$\begin{aligned} S_\alpha &\in \{\alpha_{min} + i\Delta\alpha \,; i = 0, \ldots, N_\alpha\} \\ S_\psi &\in \{\psi_{min} + j\Delta\psi \,; j = 0, \ldots, N_\psi\} \\ S_\beta &\in \{\beta_{min} + k\Delta\beta \,; k = 0, \ldots, N_\beta\} \end{aligned} \tag{1.4}$$

with a total number of observables $S = (S_\alpha, S_\psi, S_\beta)$ equal to $(N_\alpha + 1)(N_\psi + 1)(N_\beta + 1)$ which in our learning experiments amounts to a few hundreds (see Appendix A).

In spite of the drastic reduction of dimensionality of the state space that is operated by this choice of observations, we will see that this information is sufficient to learn a very effective control strategy.

1.3.4 The reward structure

The last key ingredient is the reward, which is the numerical signal issued by the environment that provides feedback about the consequences of the performed actions. As the goal to be pursued is to maximise the traction power that we can extract from the wind, we use as a reward a close proxy that is the distance travelled along the mean wind in a time step $|\dot{x}_v|\,\Delta t$. In the Appendix A we explore different reward structures not necessarily aligned with the main component of the wind. When the kite crashes to the ground it receives a penalty, i.e. a negative reward, whose magnitude may depend on the time elapsed from the start (for instance, early falls are penalized more than late ones). In addition, to avoid situations in which the kite flies dangerously close to the ground, we reduce the reward obtained whenever z_k goes below a threshold height z_{low}. The detailed parameters used in the simulations are given in the Appendix A.

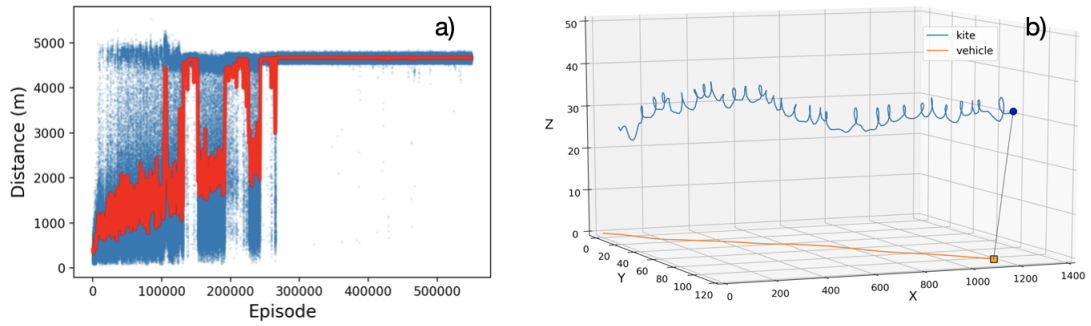

Figure 1.2: Discovering effective control strategies with Reinforcement Learning. a) Horizontal distance covered by the vehicle as learning progresses. The blue dots refer to single episodes while the red line is a moving average over 500 episodes. Figure b) represents a sample of learned motion in the turbulent Couette channel. The kite displays a helical motion adapted to the fluctuations of the wind flow. Note that the vehicle moves also in the y direction even if it is not directly rewarding since only the distance covered along x is accounted for in the return.

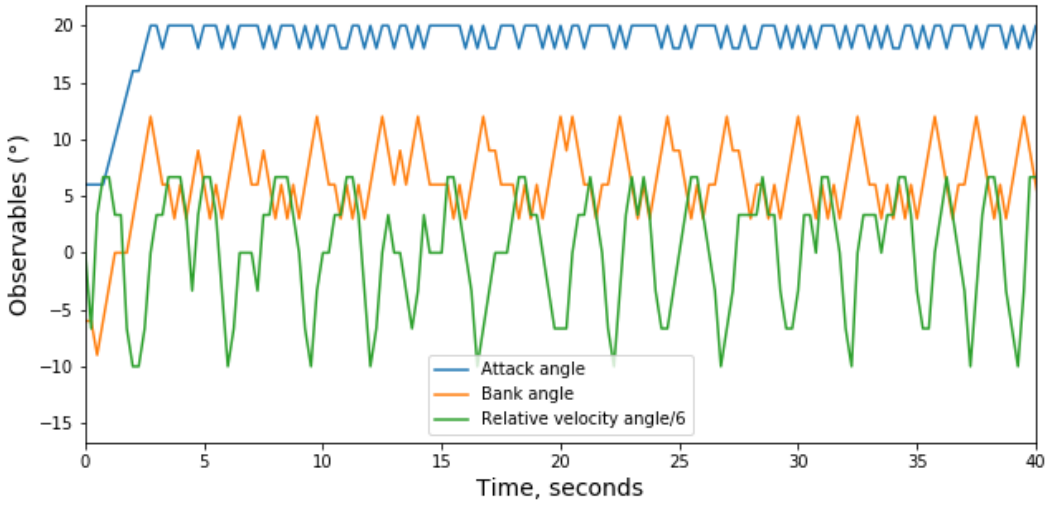

Figure 1.3: Dynamical behavior of the observables under the learned policy. Two stages emerge: a very brief transient where the kite reacts to the initial conditions, and an approximately periodic pattern where the attack angle remains close to its maximum values whereas the bank angle and the relative wind velocity angle oscillate out of phase.

1.4 Results

In this section we present the results obtained by implementing the Reinforcement Learning algorithm that described above with special emphasis on the interpretation of the learned strategy in terms of simple decision rules.

1.4.1 Learning effective control strategies

The training is divided into episodes that terminate either when the kite crashes or after a sufficiently long time. At the beginning of each episode the vehicle is randomly initialized in a different point on the ground and the kite is at a given relative position from it. Random control angles are chosen in order to sample different flow conditions and kite postures, in order to obtain robust strategies and avoid overfitting. The system is motionless at time $t = 0$ for every episode, with fixed initial angles of the tether θ and ϕ. Different take-off configurations can be considered depending on the specifics of the system at hand.

The initial estimates of the return $\hat{Q}(s, a)$ for each state-action pair are chosen optimistically in order to favor exploration of the state-action space [7].

The best results were obtained by scheduling the learning rate depending on the number of visits of the current state-action pair. This ensures faster updates of state-action values that have been visited less and vice versa (see Appendix A for details).

From Fig. A.4a we can see that after a few hundreds of thousands of episodes the distance covered along the x-axis converges to a stable value and maintains this

performance for the remainder of the episodes. We have then evaluated the learned policy on a sequence of test episodes that are different from the training ones and, importantly, last longer. The fact that the performance is unaltered confirms that the learned strategy is able to generalize to previously unseen wind configurations (see Appendix A).

As shown in Fig. A.4b the trajectory of the kite has an approximately helical shape which changes over time depending on the local wind speed and direction. The towed vehicle moves along an approximately straight path with a sideways component with respect to the mean wind. We now turn our attention to the learned control strategy, uncover its main properties and distil a simple control strategy that achieves comparable performance.

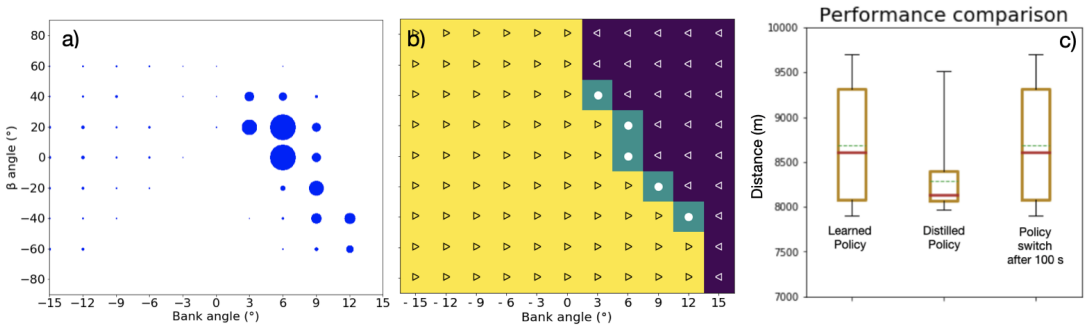

Figure 1.4: A distilled control strategy. a) Number of visits in the β_{t-1}, ψ_t space highlighting the anticorrelation between the relative wind velocity direction and the bank angle. b) The distilled policy keeps the attack angle fixed and changes the bank angle as shown. Right arrows, left arrows and circles correspond to increase, decrease or maintain the bank angle, respectively. c) The performance of the learned policy and of the distilled one are compared, together with a mixed strategy that switches from the first to the second after 100 s. Medians are in red, averages in dotted blue, the boxes are the quartiles.

Inspecting the evolution of the system under the learned policy, it is possible to identify two stages (Fig. A.11): a short transient that lasts from a few to some tens of seconds where the kite reacts to the initial motionless condition of the system, followed by a stable and approximately cyclic behavior in which bank angle ψ and β angles oscillate out of phase whereas the attack angle is nearly constant and close to its maximum value.

This observation suggests that the basic control mechanism underlying the helical motion of the kite can be actually explained in terms of simple control rules. A straightforward correlation analysis between β_{t-1} and ψ_t, gives a Pearson correlation coefficient $\simeq -0.82$. The time delay between β_{t-1} and ψ_t hints at the fact that the control reacts to changes in the direction of relative wind velocity by adapting the bank angle accordingly. The anticorrelation is also conspicuous when looking at the occurrence of visits in the β_{t-1} and ψ_t space, shown in Fig. 1.4a. Building up on these considerations we defined a *distilled policy* that keeps the attack angle fixed to the highest admissible value and changes the bank angle in such a way to reproduce the same pattern of visits as the

one of the learned policy. This distilled control strategy is displayed in Fig. 1.4b. This simple policy turns out to guarantee a performance comparable to the learned one (Fig. 1.4c), The small reduction in distance traveled can be explained by the different ways in which the learned and the distilled policy manage the initial transient phase. Indeed, implementing an improved version in which the agent switches from the learned policy to the distilled one after some time achieves a performance that is indistinguishable from the fully learned one.

1.5 Discussion

We have shown how Reinforcement Learning can discover effective control strategies to manoeuver a kite to the end of providing traction power. The kite-vehicle system is immersed in a simulated environment that comprises the essential and unavoidable effect of atmospheric turbulence.

Our results have been obtained by an algorithm which only takes into account the control angles and the orientation of the relative wind velocity as observables. The learned control can be interpreted in terms of simple rules:

i) keep the attack angle constant and as large as possible,

ii) given a certain measurement of the relative wind velocity angle β, increase or decrease the bank angle ψ in order to reach a target value that depends on β in an approximately linear way, with a negative proportionality coefficient (see Fig. 1.4b).

How this strategy generalizes to other simulated environments with different velocity statistics remains an open question that we want to address in the near future.

In our approach we selected the relevant observables based on our physical intuition. Another possibility would be to delegate the choice of the most appropriate features to the algorithm itself, for instance approximating the return $\hat{Q}$ by means of an artificial neural network as in Deep Q-Learning [27]. The latter approach could in principle discover more appropriate inputs and lead to even better performance. However, this class of algorithms are infamously known to be very data thirsty. In addition, their results are often hard to interpret in terms of human-readable rules. Here we deliberately resolved the trade off in favor of rapid training and increased explainability rather than performance. It would nonetheless be of great interest to explore alternate approaches.

The application of our method beyond the simulated environment is a tantalizing perspective. However, several challenges lie ahead when training takes place in the real physical world. Among those, a prominent necessity is finding algorithms that learn faster. Encouraging results from robotics and unmanned aerial navigation, e.g. [22], offer some hope that these challenges can be overcome and that Reinforcement Learning can become an important algorithmic tool for AWE applications.

Chapter 2

Why is SARSA so effective?

After focusing on the employment of Airborne Wind Energy for ship-towing, in this chapter we take under consideration the main usage of this technology, that is energy production. This shift in architecture may serve as a proof for the generality of the control algorithm that we deployed on one hand and it will be functional to discover more details about its success on the other, providing fruitful insights in order to answer the question about SARSA effectiveness reported in the title.

2.1 Introduction

As previously mentioned both in the introduction and in chapter 1, the term Airborne Wind Energy usually refers to a heterogeneous set of technologies and devices which share the capability of transforming the kinetic energy of wind into electrical energy by means of a generator and a flying object (in most applications, a power kite). This innovative machinery promises to address most of the issues of traditional wind turbines: the smaller, lighter structures needed for AWE lead to much lower material costs and environmental impact if compared to traditional wind energy generation and, apart from the possibility of exploiting stronger winds at higher altitudes, AWE is more flexible with respect to wind conditions, since the harvesting position can be adjusted continuously to optimize energy extraction.

However, this technology is operationally more complex than traditional turbines requiring a strong automatic control pretty much alike the case of ship-towing. In fact, in the following we show how the same Reinforcement Learning algorithm implemented in chapter 1 proves valuable in controlling this system as well. Therefore, we compare its features and performance with a new algorithm that includes a deep network.

AWE systems can be classified as ground-gen systems or fly-gen systems [9], based on where the generation step happens: in the case of ground-gen systems the ropes connecting the flying device transmit mechanical energy and a generator placed on the ground transforms it into electrical energy. On the contrary, in the case of fly-gen systems the generator is placed directly on the flying object and the ropes (in this case, electric wires) transmit electrical energy to the ground station.

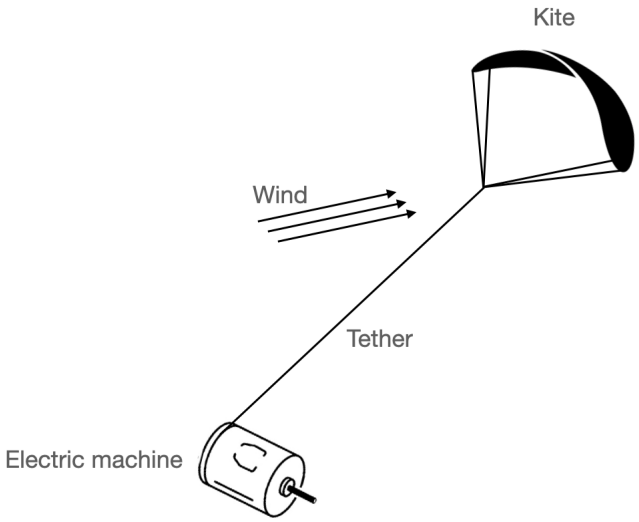

Figure 2.1: Schematic representation of an Airborne Wind Energy system.

In this work we concentrate on fixed-ground-station systems, which happen to be the most studied implementations of AWE. The basic elements of these systems are an electric machine, a kite (or another flying device) and a tether that links the two, wound on a winch connected to the shaft of the electric machine (fig. 2.1). Energy is produced by continuously performing a working cycle made of a traction phase and a passive phase. In the traction phase, the kite flies and uses wind power to unroll the lines, putting into rotation the shaft of the electric machine, which acts as a generator. During the passive phase instead, the machine acts as a motor and spends energy to rewind the tether, preparing the system for a new traction phase [11]. This architecture is referred to as the *yo-yo configuration* [24].

The switch between the traction phase and the passive phase can happen at different moments: the most obvious one is when the maximum tether length is reached, but in principle one could dynamically decide to stop production and prepare for a new productive cycle if the wind is not sufficiently strong or if potentially dangerous wind conditions arise. To make the whole cycle profitable in terms of power generation, the energy consumed during the passive phase to rewind the lines must be significantly lower than the energy produced during the traction phase. This objective can be pursued by designing two controllers: one for the traction phase that has the objective of maximizing the production of energy and one for the passive phase that has the objective of minimizing the consumption of energy. The control system operates on the lines connecting the kite and adjusts the trajectory by modifying the attack and bank angles of the kite, making it soar and turn just like an airplane.

We test two Reinforcement Learning algorithms to control the traction phase, leaving the passive one to future work. Both approaches work in a model-free setting, meaning

that in principle they would not require any *a priori* knowledge of the system, building empirical awareness from experience.

Nevertheless, like we did in paragraph 1.2.1 for the ship-towing case, we provide a model (based on the one presented by [24]) for the yo-yo configuration AWE production system. The reason is that, even if the control algorithm is designed using a model-free technique, a model is necessary to simulate the system and to construct the observations needed for the learning process. This step would not be needed if we had access to data from a physical version of the system (i.e. a kite, a tether and an electric generator).

2.2 Modeling

The structure of the system is very resembling to the kite-vehicle system studied in chapter 1, only this time the position of the ground station is fixed while the length of the cable can grow (fig. 2.2). The dynamics of the system is analyzed in detail in appendix B.

One thing to keep in mind is the different origin of power production with respect to the ship-towing architecture: while in the previous case we were monitoring the entity of the vehicle displacement as the measure of success of the control strategy, now we focus directly on the tension exerted by the cable on the ground station. The resulting instantaneous power will have the form $P(t) = T(t)\dot{r}(t)$, where $\dot{r}$ is the the unrolling velocity of the tether.

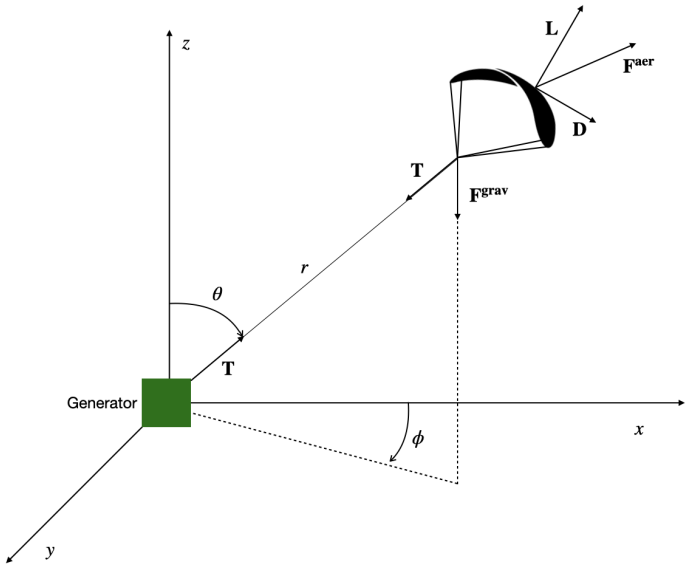

Figure 2.2: Model diagram.

Control angles are very close to the ones deployed for the ship-towing case since we

assume again to be able to change the attack and bank angle of the kite by adjusting the length of the ropes connected to its sides. We maintain the same kite features, translating into analogous lift and drag coefficients. As for the bank angles, we resort to a smaller range of values ($N_\psi = 6$) according to the different dynamics of the system in study. The complete table regarding the attack angles and the values of the bank angles can be found in appendix B, containing a higher number of possible attack angles ($N_\alpha = 14$) as reported in [24].

In this chapter we will deal with simpler flows, since we are not exclusively interested in the performance of the algorithms. We test them in two different virtual environments, such as the constant horizontal wind analyzed in Appendix A and a linearly increasing flow, where the horizontal velocity grows with altitude. We show how especially in these simplified contexts we can obtain useful information about which observables are relevant for the learning process, thanks to the analysis of the neural network employed by the deep-Reinforcement Learning algorithm we are going to introduce in the next section.

2.3 The Reinforcement Learning framework

In the following we are going to introduce the two learning algorithms working on the same task, which is the maximization of the traction power of the tether exploiting the wind field. We will highlight their differences and common points, focusing on their structure and their inputs.

The confrontation between the two algorithms is made fair by the fact that both have access to the same information about the environment and can perform the same actions described in chapter 1. Once again the states S used by the learning algorithms do not coincide with the full state of the system X, marking the difference between this approach and the optimal control framework, which is instead bound to the knowledge of the variables of the model dynamics.

Moreover, in both Reinforcement Learning algorithms the found strategies are restricted to the domain of reactive strategies, since they select actions only taking into account the last made observation and not the entire history of the system, assuming that

$$\pi(A_t|S_t) \simeq \pi(A_t|S_t, A_{t-1}, S_{t-1}, ..., A_0, S_0). \tag{2.1}$$

What differs is how the estimate of the total extractable power is built up and how the observations of the system are given as an input.

2.3.1 Tabular SARSA

The first algorithm is the SARSA algorithm described in chapter 1 and resumed in alg. 1: it consists in a tabular temporal-difference algorithm working on a discrete space of states $\mathcal{S}$ which adjusts the prediction about the expected return according to the

perceived reward. When we work with a limited set of states $|\mathcal{S}|$ and actions $|\mathcal{A}|$, this translates into approximating the entries

$$Q(S, A) = \mathbb{E}[\sum_{k=t}^{T} R_k | S_t = S, A_t = A] \tag{2.2}$$

of a table in which to each state-action pair corresponds an estimate of the future obtainable return starting from a given state-action pair (S, A), with $S \in \mathcal{S}$ and $A \in \mathcal{A}$. Such estimate $\hat{Q}(S, A)$ is further refined with experience:

$$\hat{Q}(S, A) \leftarrow \hat{Q}(S, A) + \eta(R + \hat{Q}(S', A') - \hat{Q}(S, A)). \tag{2.3}$$

At each decision step the current estimate is corrected by a factor called *temporal difference error* $\delta = R + \hat{Q}(S', A') - \hat{Q}(S, A)$, where R is the reward perceived after performing action A and $\hat{Q}(S', A')$ is the expected return in the new state-action pair (S', A'). This results in a bootstrapping technique where the *target* $R+\hat{Q}(S', A')$ towards which the learning is moving consists in the combination of the witnessed reward and the current estimate in the subsequent state-action pair.

The policy employed to select the action is the ϵ-greedy one introduced in 1.3.1, which chooses the action $a = \arg\max_b \hat{Q}(S, b)$ almost always except for a portion ϵ of times in which another random action in taken.

Algorithm 1 SARSA

Initialize $\hat{Q}(s, a)$ arbitrarily $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$
repeat(for each episode)
  Initialize S
  Choose A from S using ϵ-greedy policy derived from $\hat{Q}$
  repeat(for each step of episode):
    Take action A, observe reward R and next state S'
    Choose A' from S' using policy derived from Q
    $\hat{Q}(S, A) \leftarrow \hat{Q}(S, A) + \eta[R + \gamma\hat{Q}(S', A') - \hat{Q}(S, A)]$
    $S \leftarrow S'$
    $A \leftarrow A'$
  until S is terminal
until last episode is over

2.3.2 Deep Q-Learning

The second algorithm enforces the universal function approximator property of neural networks and lets a feed-forward net estimate the expected return. It follows that $\hat{Q} = \hat{Q}(S, A, \boldsymbol{w})$ will depend on the weights of the network $\boldsymbol{w}$. Instead of acting directly on $\hat{Q}$, the focus shifts from learning the entries of a table of size $|\mathcal{S}| \times |\mathcal{A}|$ to learning a set of weights of size d. The network will receive as an input the present state of the system

- not necessarily discretized this time - and will output the estimated values of $\hat{Q}$ for each action.

The standard approach to neural network training includes some variant of gradient-based optimization of a properly defined loss function, which quantifies for a certain input vector $\mathbf{x}$ the difference (according to a metric d) between a reference vector $\mathbf{r}$ and the output $\mathbf{y}$ of the network:

$$\mathcal{L}(\mathbf{w}, \mathbf{x}) = d(\mathbf{y}(\mathbf{w}, \mathbf{x}), \mathbf{r}(\mathbf{x})). \tag{2.4}$$

This training technique relies on gradients and it is made possible by the introduction of ways to quickly compute gradients such as the backpropagation algorithm [28].

The weights of the network are updated iteratively according to a rule that, in the simplest case (plain gradient descent) runs as follows:

$$w \leftarrow w - \eta \nabla_w \mathcal{L} \quad \forall w \in \mathbf{w}. \tag{2.5}$$

Once again, the hyperparameter η goes by the name of learning rate and it defines the size of the step to be taken in the parameter space when updating the weights.

The algorithm that marked the success of Deep Learning approaches to RL is Deep Q-Learning (DQL), presented in 2015 [29]. Q-Learning update rule slightly differs from SARSA one and its flow is reported in alg. 2. Basically, the algorithm always selects the action with the highest estimate $\max_a \hat{Q}(S', a)$ at the subsequent step, detaching from the ϵ-greedy policy adopted to select actions in the present step.

DQL and its later variants allowed RL researchers to achieve outstanding success in very demanding scenarios, such as complex board games or video games, that are now played at human or super-human level. In particular, as one could expect, the neural approach to RL is particularly effective when the state to elaborate is an image (or a video), since neural architectures such as convolutional neural networks are extremely effective at processing this kind of input.

Algorithm 2 Q-Learning

Initialize $\hat{Q}(s, a)$ arbitrarily $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$
repeat(for each episode):
  Initialize S
  repeat(for each step of episode):
    Choose A from S using ϵ-greedy policy derived from $\hat{Q}$
    Take action A, observe reward R and next state S'
    $\hat{Q}(S, A) \leftarrow \hat{Q}(S, A) + \eta[R + \gamma \max_a \hat{Q}(S', a) - \hat{Q}(S, A)]$
    $S \leftarrow S'$
  until S is terminal
until last episode is over

However, in this work we deal with a much simpler, numerical set of inputs, so we can rely on a straightforward application of neural approximation to Q-Learning, by

means of a standard feed-forward neural network.

To do so, one can notice that the update rule of Q-Learning is equivalent to:

$$\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} \mathcal{L}(\hat{Q}(S, A, \mathbf{w}), R + \gamma \max_a \hat{Q}(S', a, \mathbf{w})). \tag{2.6}$$

Then, it is sufficient to define a loss function that keeps into account the difference between the target $R + \gamma \max_a \hat{Q}(S', a, \mathbf{w})$ and the estimated Q-value $\hat{Q}(S, A, \mathbf{w})$ and to minimize it with respect to the parameters $\mathbf{w}$ using gradient-based optimization techniques.

The key element of DQL is a Q-Network, the neural network used to approximate Q values. In common practice this network takes as input the state S and outputs a vector of Q values, one for each action A in $\mathcal{A}$.

At each step of the learning procedure, the parameters of the network are updated according to equation 2.5. Assuming that the most common distance metric, the l_2 distance, is used as a loss function, the loss at step t would be:

$$\mathcal{L}_t = (R + \gamma \max_a \hat{Q}(S', a, \mathbf{w}_t) - \hat{Q}(S, A, \mathbf{w}_t))^2. \tag{2.7}$$

The gradient descent step would then read:

$$w_t = w_{t-1} - \eta \nabla_{w_t} \mathcal{L}_t \quad \forall w \in \mathbf{w}. \tag{2.8}$$

Some alternative approaches have been proposed to improve the stability of the training process, such as semi-gradient optimization and the use of a target network.

In a semi-gradient setting, the loss at step t becomes

$$\mathcal{L}_t = (R + \gamma \max_a \hat{Q}(S', a, \mathbf{w}_{t-1}) - \hat{Q}(S, A, \mathbf{w}_t))^2, \tag{2.9}$$

the only difference from 2.7 being that the target value is supposed to be constant with respect to the parameters $\mathbf{w}_t$. In this case, the update step is usually written as

$$w_t = w_{t-1} + \eta \delta \nabla_{w_t} \hat{Q}(S, A, \mathbf{w}_t) \quad \forall w \in \mathbf{w} \tag{2.10}$$

where δ is the temporal-difference error $(R + \gamma \max_a \hat{Q}(S', a, \mathbf{w}_{t-1}) - Q(S, A, \mathbf{w}_t))$.

Another approach which is gaining popularity in recent works is the one relying on the use of a Double Deep Q-Network (DDQN) [30]. In this setting learning stability is sought by resorting to a different Q-Network (called target network) for the computation of the target (with the same architecture and different parameters). The parameters of the main network are then copied into the target network at fixed intervals.

Last but not least, the experience replay technique presented in [29] also constitutes an interesting upgrade in the context of Deep Reinforcement Learning, since it allows to store in memory a buffer of past rewards and then use an averaged out batch of them to upload the estimate $\hat{Q}$, providing with better stability and preventing from the so-called phenomenon of *catastrophic forgetting*.

However, on our specific control task we found the simplest choice to prevail on all the others: in fact, *true* gradient descent approach showed to converge faster and to yield

better results in terms of cumulative returns with respect to semi-gradient optimization, DDQN and experience replay as well.

2.3.3 An informed approach to deep-Reinforcement Learning

Feed-forward networks are usually employed with a black-box approach, favoring performance and giving up the understanding about the configuration of the weights inside the net. We instead resort to DQL as a tool to extract side information on the learning success.

As it was briefly anticipated, one great advantage of approximating the expected return with a neural network is the chance of feeding it with continuous inputs, which is not possible using tabular SARSA. Moreover, the net can in principle deal with multidimensional inputs finding the features to better represent them on its own. On the contrary tabular SARSA requires an a priori discretization of the input demanding for an approximation whenever the observable is in fact continuous.

Basically, DQL is able to deal with a dense set of states by performing dimensionality reduction towards the space of weights of dimension d, avoiding suffering of the curse of dimensionality. The meaningful information on how this feature-extraction procedure is performed lies in the weights themselves: the first layer of weights for example, connecting the input neurons to the first hidden ones, might play the role of a filter, in the sense that if one observable given as input is useless, then the weights associated to it could go to zero. This is sufficient for the specific observable to be ignored by the decision-making procedure, but it's not always true that redundant input is eliminated by the first layer.

Another question that we pose is how information about the input is encoded in the final hidden layer of neurons. This encoding is comparable to what in tabular SARSA has to be done by hand, discretizing the state of the system before feeding it to the algorithm. In fact, we will show in the following that from the analysis of the activations of the last layer we can understand intuitively how the choices operated with SARSA work out well for obtaining good results.

2.3.4 State and control variables

Since the observation S fed to the learning algorithm does not necessarily coincide with the full state of the system, we have to select the observables that we want our algorithms to monitor. Of course there is a trade-off between data abundance and fast convergence.

Both SARSA and DQL are shown to work fine even with a very narrow set of observations, which are chosen to to coincide with the control variables α and ψ, namely the attack angle and the bank angle. As in 1.3.2, we assume to be able to increase and decrease by one level the two angles, by means of actuation on the lines connecting the kite. This leaves us with 3 actions per control variable (increase, decrease, keep still) and 9 (3×3) total actions.

There is also a third observable that we monitor, which is the orientation β of the relative velocity of the wind with respect to the kite. We will discuss in the following

how the presence of the angle β can be understood to be in fact superfluous in a constant wind environment, simply by analyzing the way it is processed by the network.

Notably, while in tabular SARSA information about β has to be discretized, the deep network can directly process continuous signals without the need for dividing it into rigid sectors.

2.3.5 Reward structure

Reinforcement Learning is a goal-directed task and in Airborne Wind Energy systems the objective is to maximise the power extraction from the wind. Consequently, it is crucial to assign rewards proportionally to the success of the control strategy in wind harvesting.

The parameter of this success was measured by the covered distance in the case of ship-towing, while in this case we focus directly on traction power. Therefore, we chose to assign to the agent a reward equal to the maximum extractable energy during the next learning interval. This energy is approximated using the formula:

$$E(t) = T(t)\dot{r}(t)\Delta t = R_t. \tag{2.11}$$

Therefore, in case of a critical failure, the agent is assigned a penalty instead of a reward, regardless of what was the cause of failure (unexpected landing or numerical error). Such penalty should have the same order of magnitude of the average obtained return in order to clearly distinguish the scenario in which the kite crashes from the desired one in which everything works smoothly. The controller will then accordingly decrease the value of the state-action pair that led it to the failure, keeping memory of the mistake in the following repetitions of the task.

2.4 Simulations and results

We now proceed with displaying the results of the simulations we ran using SARSA and DQL. We start with the constant horizontal wind environment and then move to the linear wind gradient. The learning is episodic, so at the beginning of each episode the kite is initialized in a fixed position in mid-air in order to avoid dealing with the take-off phase. The initial attack and bank angle are randomized instead.

Once the kite is released we consider the cable to fully extend along the episode, assuming its length to be virtually infinite. Another solution consists in fixing the maximum length of the tether instead of the duration of the episode and letting it finish when the cable has reached full extension. We will employ this second option for the linear wind gradient case.

2.4.1 Results in a constant wind pattern

We showcase the results obtained with a constant wind directed along x with speed $u = 10\,m/s$ in two trainings of 8000 episodes each.

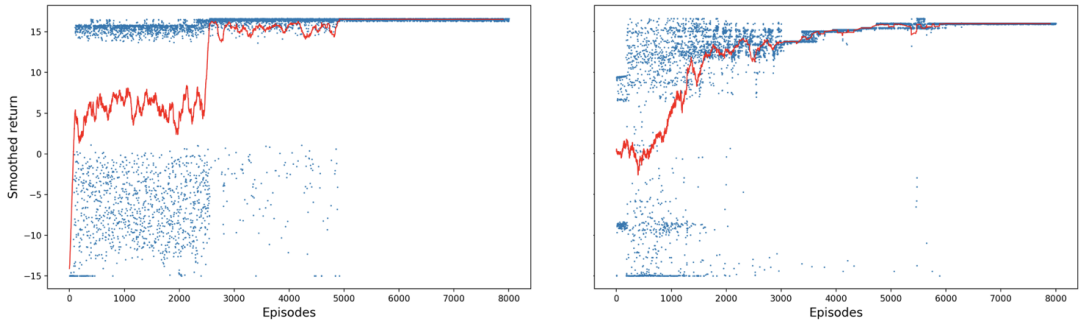

Figure 2.3: Curves of the return for SARSA (left) and DQL (right). Each blue dot correspond to the outcome of a single episode while the red line is a moving average over 100 episodes. The units are in 0.1 kWh and the penalty is equal to 15, determining the visible gap between successful flights and unfortunate ones, especially in the SARSA learning.

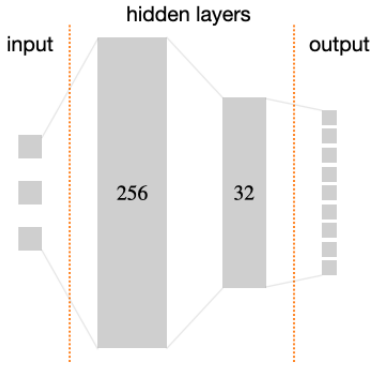

Figure 2.4: A scheme of the Q-network employed for the learning.

Both the learning and exploration rate are scheduled towards zero along the training and the corresponding thresholds and decays are listed in Appendix B. This guarantees that the strategy converges to the best possible reactive strategy for the given choice of observables.

As potraited in fig. 2.3, the performances of the two algorithms are quite similar as they succeed in converging to the highest possible return. Notably, both SARSA and DQL are able to achieve this result without the need of knowing the angle β and just by keeping track of the control angles. This confirms the findings reported in Appendix A for the ship-towing case, showing that – even with a different architecture providing the data – information about the orientation of the relative velocity is in fact redundant with a constant wind blowing.

However, it is remarkable to point out that if we indeed add β to the space of observations S which the learning algorithms are able to witness, the performance of SARSA and DQL diversifies, since SARSA has to deal with an augmented space $\mathcal{S}$

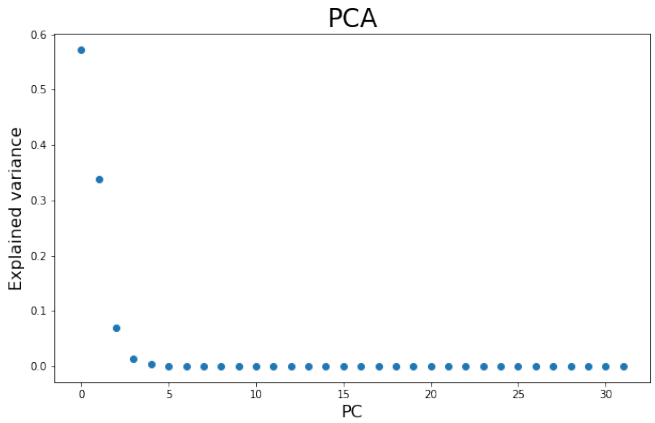

Figure 2.5: Explained variance of the final layer for a 2-dimensional input consisting only of the control angles. As expected, there are 2 relevant variables accounting for almost all the variance.

leading to a number of entries of table Q that goes from $(N_\alpha+1)(N_\psi+1)$ to $(N_\alpha+1)(N_\psi+1)(N_\beta+1)$. The time needed for the algorithm to converge has to scale consequently. This does not happen when the deep network is involved, since the number of weights remains almost constant with the addition of one input neuron.

We can look deeper into how the new input observation affects the learning process inside the network. The net is composed by an input layer, with 2 or 3 neurons depending whether β is there or not, two hidden layers respectively of 256 and 32 neurons and the output one, with 9 exits corresponding to the 9 possible actions (fig. 2.4). The 32-neuron layer goes under the name of *final* layer and plays the role of feeding to the output the input processed by the neural architecture. In some sense it is analogous to the grid which is used by SARSA to map states into actions, only this time the network is free to re-elaborate the data by collecting it into clusters if needed, individuating the best features to represent it independently.

If we analyze the activations of the neurons of the final layer we move in a 32-dimension space which maps the information from the input layer which is at most 3-dimensional. In order to understand the intrinsic dimension of the data in input as it appears to the network, we can perform Principal Component Analysis (PCA) on the final layer, whose results are shown in fig. 2.5 for a 2-dimensional input.

As one could expect, the intrinsic dimension turns out to be 2 since the analysis highlights 2 components accounting for 90% of the variance. It consequently appears that both control angles are important for the decision-making problem we want to solve.

If we represent the space spanned by this two components fig. 2.6(left) we get a quite regular grid that reproduces the spacing of the input points: with a 2-dimensional input we get $|\mathcal{S}| = (N_\alpha + 1) \times (N_\psi + 1) = 15 \times 7 = 105$ points arranged on a rectangular grid with no clustering.

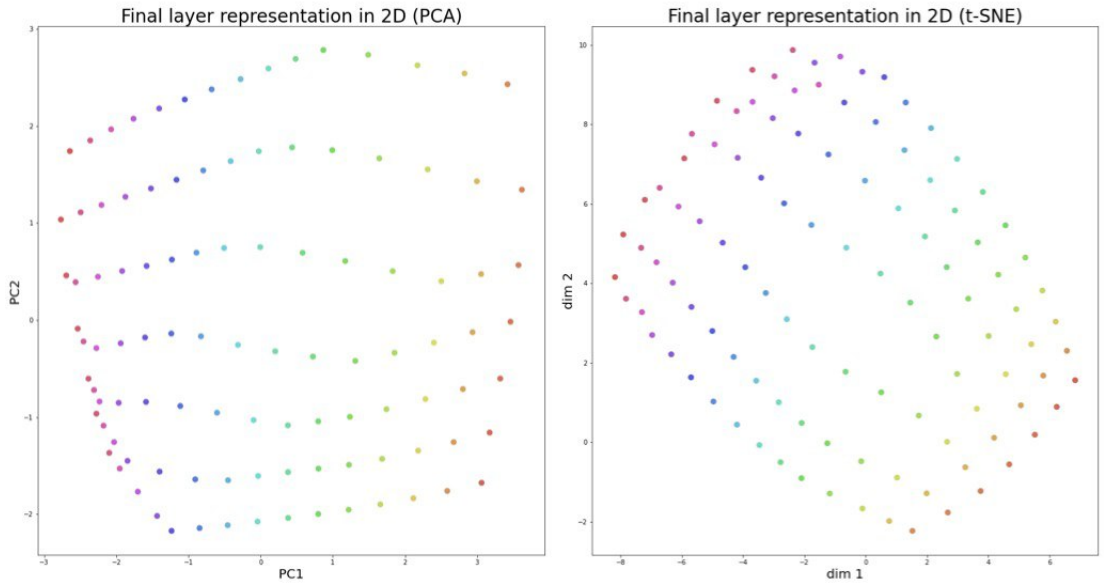

Figure 2.6: Projection onto the space of the first 2 components of the PCA (left) and of t-SNE analysis (right) performed on the final layer starting from a 2-dimensional input.

Another dimensionality-reduction method we can apply on the final layer is T-distributed Stochastic Neighbor Embedding (t-SNE) [31] which is best suited to represent high-dimensional data on a 2-dimensional map. Also in this case the analysis results with a fairly regular 15×7 grid (fig. 2.6 right) suggesting that there is no simpler representation of the input data than the one provided to SARSA, therefore giving motif to the success of the tabular algorithm in learning the most fruitful strategy in order to maximise traction power.

When we add the angle β to the incoming observation, it is interesting to acknowledge how this further information changes the scenario. If we look at fig. 2.7 we can see that the intrinsic dimension retrieved in the final layer is again equal to 2, as the first two components explain roughly 93% of the variance, seemingly regardless of the new variable β, in line with the continuity in performance between the 2-dimension and 3-dimension input.

Now, if we project again onto the space of the first two components (fig. 2.8) we find again the same grid as before both with PCA and t-SNE, only this time we can recognize clusters where previously we had single points: those clusters are given by the little influence of the new variable β, which is fed into the network as a continuous value and later discretized in 10 bins in order to perform the analysis.

This is an explainatory example of the potential represented by the deep network in autonomously recognize which observations are meaningful, allowing us to work with the lightest possible structure, polished from superfluous inputs. Moreover, it is useful to stress how tabular SARSA is actually extremely efficient in performing this task, making use of the smallest number of parameters and therefore even overcoming the speed of

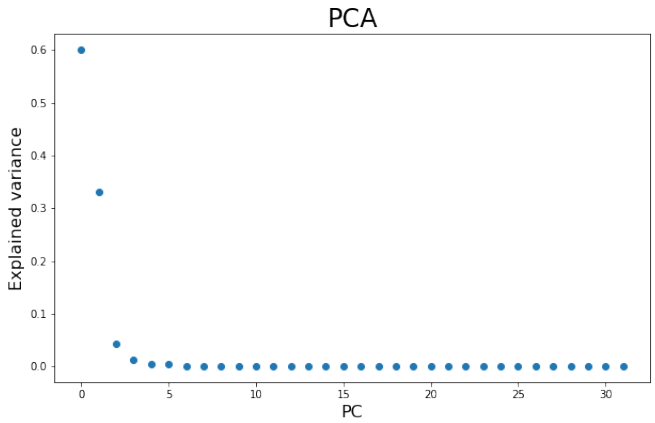

Figure 2.7: Explained variance of the final layer for a 3-dimensional input consisting of the control angles and relative velocity orientation. The relevant variables continue to be 2, accounting for 93% of the variance.

DQL in learning the optimal strategy.

Both SARSA and DQL agree on which is the best strategy in order to pull the tether in a constant wind environment and it is similar in essence to the one discovered for ship-towing, which consists in flying crosswind performing helices (fig. 2.9). Differently from the kite-vehicle system now the station on the ground is fixed and the tether elongates along the duration of the whole episode, dealing with a slightly modified dynamics which still leads to the same solution, in line with what studies on AWE predict about the optimality of crosswind motion [32].

2.4.2 Results with a linear wind gradient

We briefly recap what happens when the horizontal wind increases with height, guaranteeing better results whenever the kite harvests higher altitude winds, in an environment that recalls in a simplified way what happens in the boundary layer of the atmosphere. We constrain the kite with a limitation on the length of the tether and show that SARSA algorithm is able to learn to soar in order to exploit stronger winds, this time making good use of the information on the relative velocity orientation. Fig. 2.10 shows successful results over a longer strand of smaller episodes, which reach their end when the tether reaches its maximum extension (fixed to $950\,m$). Even in this shorter amount of time, the kite is able to harvest more energy starting from the same conditions of the constant wind set-up, since the velocity at the height at which the kite is initialized is the same in both environments.

If we analyze the learned trajectory we indeed find out that the kite still moves crosswind while climbing till it reaches the optimal height over which the wind would become too violent to be advantageous (fig. 2.11).

The same performance cannot be easily obtained with DQL, which fails in finding an

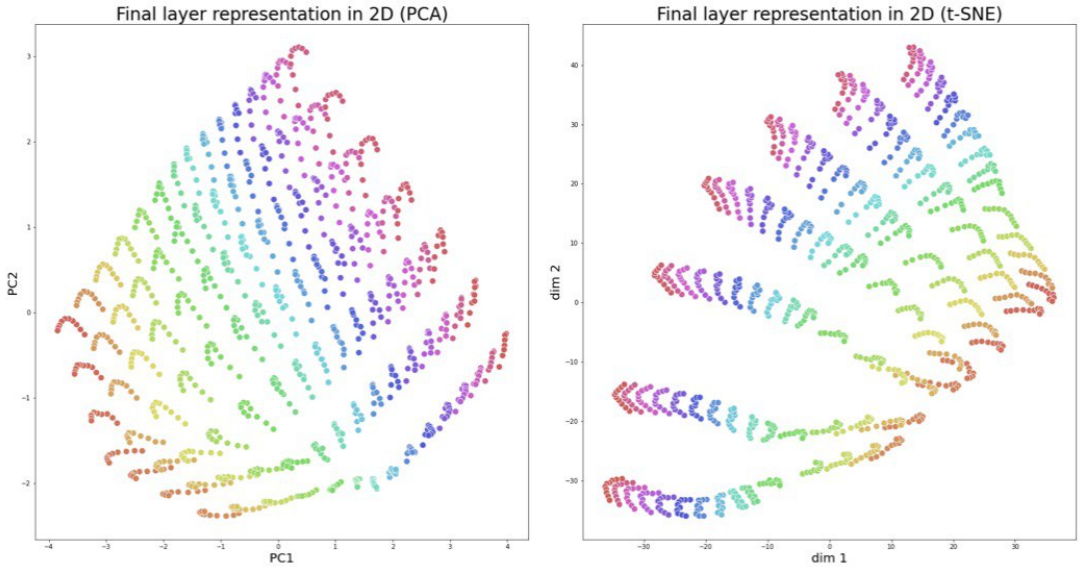

Figure 2.8: Projection onto the space of the first 2 components of the PCA (left) and of t-SNE analysis (right) performed on the final layer starting from a 3-dimensional input.

efficient strategy to solve this task, underlining once again the effectiveness of SARSA in wind harvesting jobs.

2.5 Conclusions and future perspectives

In this chapter and in the previous one we showed that Reinforcement Learning is a suitable alternative to optimal control for AWE systems, both in simple and complex environments. Moreover, we showed that even simple, tabular algorithms like SARSA can be very effective at addressing this problem, even in realistic and challenging conditions like a Couette channel flow.

In this chapter we introduced Deep Reinforcement Learning as an alternative method to control a kite in a windy environment, showing that it can provide valuable insights about which observables can be relevant in the learning process.

However, when compared with SARSA and disposing of the same information, it turns out DQL models are more difficult to train as the search for the correct hyperparameters can be lengthy and the time needed to reach convergence is longer as well with respect to tabular algorithms working on a limited set of states such as the one that we showed here. Also when combined with the experience replay technique we did not witness improvement in terms of performance. Nevertheless, DQL remains always a profitable choice whenever the amount of input data increases rapidly dismissing the tabular alternative. Yet, it turns out that in this specific case the intuition about the interest on the orientation of the relative velocity is enough to obtain solid results also in complex wind conditions.

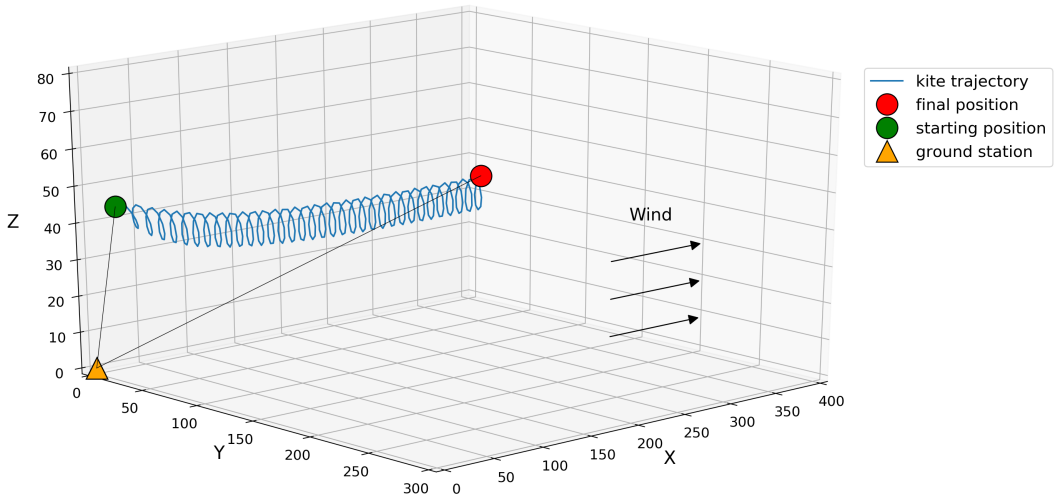

Figure 2.9: Sample trajectory of a kite connected to a generator on the ground pulling a virtually infinite cable for 300 s in a constant wind environment.

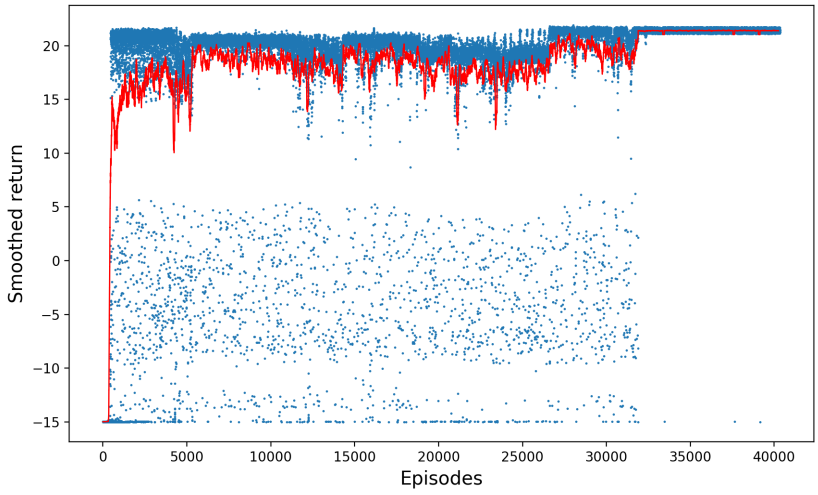

Figure 2.10: Curve of the return for learning with a linear gradient wind in units of 0.1 kWh with a penalty equal to 15.

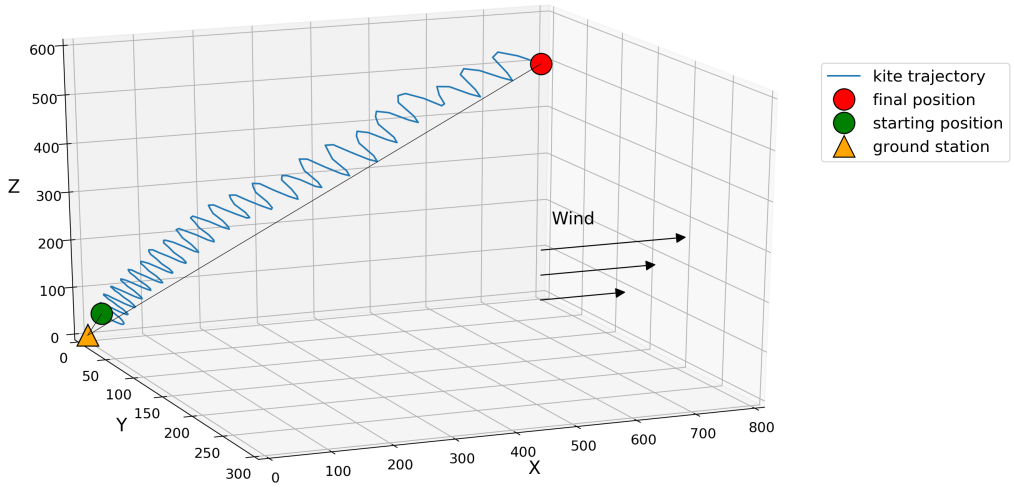

Figure 2.11: Sample trajectory of a kite connected to a generator on the ground pulling a cable with maximum length equal to 950 m in a linear gradient wind.

Of course there is still long way to go in terms of mastering the full control of AWE systems: one direction we want to head towards is the optimization of the passive phase. In this sense, we see two main approaches: one is the design of two separate control algorithms, one for the traction phase (such as the ones proposed in this thesis) and one for the passive phase, with a switch between the two happening at a fixed time or by human intervention. Another possibility, more challenging but also more interesting from our point of view, is the design of a single control algorithm, which automatically decides when it is the right moment to stop energy production and to start rewinding the lines.

This second approach would significantly increase the complexity of the problem at study, but it enables a new degree of freedom to exploit in the optimization of the system: it could be very profitable to have a system that is able to understand if at some point the wind conditions make it preferable to stop production and rewind the lines, preparing for a new traction phase.

Finally, we would like to test our approach in a fully realistic setting, using a physical version of the AWE system that we modeled and simulated for this work. This step would lead us to face the difficulties and the complexity of a real world scenario but it would also make us fully exploit the power of a completely model-free approach to this problem.

Chapter 3

Spider Ballooning as a decision-making process

This chapter is devoted to the study of a peculiar animal behaviour which goes under the name of spider ballooning. Again we are looking at the phenomenon of taking flight and becoming airborne, only this time the objects of our research are way smaller and lighter than the kites used for energy harvesting. Yet the principle is the same and relies in exploiting the wind to move fast away from the starting point. In fact, spiders are able to enforce the wind drag acting on their silk lines and use them effectively as a wing, allowing them to fly.

What results particularly interesting to us is the decision-making process that stands behind this phenomenon: it turns out that spider species capable of ballooning first inquire the environment that surrounds them for a few seconds and then understand if the conditions are actually favourable for them to take flight or if it is better to weight for another more suitable moment. This decision capability clearly hides an optimization process which has taken place over course of evolution.

Even if spider ballooning remains a highly risky practice since there is no control on the direction of motion once becoming airborne, there must be some recognizable features of the local wind field which can discriminate between bad and good take-off times. Along this chapter we address this problem, developing a novel model for the dynamics of ballooning spiders and then testing it in a synthetic wind environment in order to extract the most relevant characteristics of local wind sequences required to maximise the probability of success.

3.1 Introduction

Dispersal is a crucial mechanism for survival in some animal species and consists in moving away from the birth-place in order to avoid overpopulation and consequently starvation [33]. Unlike migration, dispersal is usually a one-way trip to an unknown destination. Dispersal traits and mechanisms are thus wide and varied across taxa, even for organisms that disperse passively through air or water currents.

Spiders represent one taxon that undergoes multiple dispersal pathways: aside from walking from site to site, spiders can take advantage of their silk lines in different modes. One method, *bridging*, is to cast a line into the breeze and to climb out on it when it catches on a distant object [34]. Another one is to balloon: when the extruded thread and the spider get enough buoyancy from updrafts (usually thermals or vertical wind-velocity gradients), the spider will be lifted off the substrate and carried through the air [35].

This behaviour is common across small species of spiders ($< 5\,mg$ of body mass), but it has been observed also in larger ones [36]. The spider generally climbs up to a high point and showcases what is called *tiptoe* behaviour, straightening its legs, balancing on the tips of its tarsi [37]. From that position it senses if the conditions allow for take-off, occasionally also raising the two anterior legs to actively estimate wind velocity in the place it is located [38]. If the outcome of this preliminary analysis comes out positive, the spider raises its abdomen, releasing one or more silken draglines in the air. Wind then allows for drag-induced lift of the whole body. Otherwise it gives up in order to hide and wait for more suitable conditions.

Once airborne, individuals have little control over the direction and distance of displacement [39]; rather, they join other floating life forms collectively known as *aerial plankton* [40], which are subject to air currents.

While there have been attempts to observe ballooning distances visually, which suggest that spiders move no more than a few hundred metres in any one attempt [41], it also may be inferred from anecdotal evidence that spiders make journeys of several hundred kilometres, reaching up to 5 km altitudes above the ground. Charles Darwin for example noticed this phenomenon and wrote down on his *Beagle Diary* about some spiders landing on the ship after a 60 mile journey off the coast of Buenos Aires.

The decision whether taking flight or not is fairly complex, as it is inevitably based on partial observation of the surrounding environment: the spider looks out for clues regarding the success of its flight collecting local information without knowing how the conditions are going to evolve once it has become airborne. There is still open debate on the individuation of the most relevant physical properties of the environment that favour ballooning. Multiple field observations suggest that spiders balloon during daylight hours, under sunny and clear skies, and at wind speeds less than $3\,m/s$ [42, 43]. In these conditions the vertical movement of warmer air up and cooler air down results in static instabilities in the atmosphere and leads to the formation of a vertical, turbulent layer filled with vortices, called thermals, that can be used as a lifting force by ballooning spiders. However, some researchers have conjectured that the negative surface charge density of the Earth may play a role in spider ballooning, making electrostatic field is a necessary condition for take-off [44], even though empirical findings have shown that thermal currents could provide all the necessary lift for ballooning.

Again, we are faced with the problem of predicting the behaviour of the wind in order to extract from it the power needed to move away from the original location. As we pointed out at the beginning, this task can be phrased as an optimization process. More specifically, we will show it is straight-forward to cast it as an instance of *bandit*

algorithms [45], which stand as a subclass of Reinforcement Learning problems. We will draw our first conclusions by addressing it using basic statistical tools coming from logistic regression.

The first issue that we encounter in our treatment is the need for data, whereas spider ballooning is a rather difficult phenomenon to keep track of, since for example radar technology fails to resolve ballooners [46]. In absence of experimental data to examine, we start by developing a novel physical model of spider ballooning that draws inspiration from the existing ones and takes under consideration more than one dragline. We proceed by testing it in a synthetic wind field, providing with the data to later elaborate in the optimization framework. After that, we examine the preliminary results of our prediction framework and set the stage for further data production and more developed analysis.

3.2 Model of the environment

In this section we introduce our own physical model of a ballooning spider and then describe the virtual wind environment that we present to the spider.

3.2.1 Physics of the ballooning spider

The first analytical model for spider ballooning dates back to 1987 was built by Humphrey [47], who studied the phenomenon with a simple fluid mechanical model. In Humphrey's model, the spider is represented as a solid sphere. Attached to the solid sphere is a rigid, inextensible, cylindrical rod that is used to approximate a silk dragline, thus producing a *lollipop* appearance. The rod is considered to be massless relative to the spider. This crude approximation worked as conceptual foundation of the physical constraints under which ballooning spiders must operate, obtaining some empirical confirmations [48]. However, the physical properties and dimensions in Humphrey's model were not validated.

A major refinement to Humphrey's model comes by the works of Reynolds's group in 2006/07 [49, 50]. Instead of the rigid rod, they modeled the silk dragline as a chain of springs and spheres that resist stretching but not bending. This model leads to better agreement with observations of ballooning behaviour, yet it showcases the problem of silk entanglement, which is not actually reported to be an issue in nature.

A punctual review of state of the art quantitative modeling on spider ballooning up to 2017 can be found at [51], including, together with the two that we mentioned, other studies focused on diffusion and on electrostatics.

Our model for the mechanics of spider ballooning takes Reynolds scheme as a baseline and introduces some new ingredients. The first one consists in allowing for more than one dragline connected to a single spider. This results in multiple chains of springs connected by beads departing from the spider's body. Each bead is uniquely defined by its position $\boldsymbol{q}_{i,j}$, where i is the bead number inside a specific chain while j is the index of the chain. Since we neglect electrostatic interactions between chains, we let the j index

fall in the following to deal with a lighter notation.

The force acting on each bead is the sum of 3 contributions, leading to the following Newton equation:

$$\ddot{\boldsymbol{q}}_i = \frac{1}{m_i}(\boldsymbol{F}_i^{el} + \boldsymbol{F}_i^{aer} + \boldsymbol{F}_i^{rig}) \tag{3.1}$$

where m_i is the mass of bead i and $\ddot{\boldsymbol{q}}_i$ its acceleration. We proceed by analyzing the forces in detail.

- $\boldsymbol{F}_i^{el}$ is the elastic force acting on node i and standing for the resistance to stretching of the chain. Elastic force is already present in Reynolds model and its energetic contribution to the total energy of the chain reads:

$$\mathcal{H}^{el} = \frac{1}{2}\kappa \sum_{i=1}^{N}(|\boldsymbol{q}_i - \boldsymbol{q}_{i-1}| - s_0)^2 \tag{3.2}$$

  where s_0 is the resting length of the spring, which we consider to be the same for all the springs as for the elastic constant κ. It follows that the force on the i-th bead is

$$\boldsymbol{F}_i^{el} = -\boldsymbol{\nabla}_i \mathcal{H}^{el} = \kappa(s_{i,i-1}\boldsymbol{p}_{i,i-1} + s_{i,i+1}\boldsymbol{p}_{i,i+1}) \tag{3.3}$$

  where $s_{i,i-1} = |\boldsymbol{q}_{i-1} - \boldsymbol{q}_i| - s_0$ and $\boldsymbol{p}_{i,i-1} = \frac{\boldsymbol{q}_{i-1}-\boldsymbol{q}_i}{|\boldsymbol{q}_{i-1}-\boldsymbol{q}_i|}$ and similarly for bond between i and $i+1$.

- $\boldsymbol{F}_i^{aer}$ is the i-th component of the aerodynamic force acting on the filament. It is the contribution that determines the possibility to exploit the wind power in order to take-off and remain airborne. In this matter we apply a refinement to Reynolds model, taking into account the anisotropy of the silkline. We still consider the aerodynamic force to scale linearly with the relative velocity of the bead with respect to the fluid $\boldsymbol{v}_{rel} = \dot{\boldsymbol{q}}_i - \boldsymbol{u}_i$, which is customary for objects moving at relatively slow speeds in fluids with no turbulence. However, previously $\boldsymbol{F}_i^{aer}$ was taken to be anti-parallel to the relative velocity (as it was the case for the drag in chapters 1 and 2, leading to a purely-drag motion), while now we decouple $\boldsymbol{v}_{rel}$ into 2 components: one parallel to the spring, therefore exerting lower resistance to the fluid, and one perpendicular, responsible for higher force. Such difference is not true for the spider's body, which we model as a sphere and consequently aerodynamics affects it isotropically.

- $\boldsymbol{F}_i^{rig}$ is the second main update to the Reynolds model, introducing a rigidity force that limits the unrealistic bending possibility of the dragline. Similarly to what is done in [52], we consider an energy term $\mathcal{H}^{KP}$ which resembles the *Kratky-Porod* model in polymer physics [53], yielding

$$\mathcal{H}^{KP} = J \sum_{i=1}^{N-1} \boldsymbol{p}_{i,i-1} \cdot \boldsymbol{p}_{i,i+1} \tag{3.4}$$

resulting in a penalty whenever two adjacent springs are not parallel. The bending resistance acting on node i is given by

$$\boldsymbol{F}_i^{rig} = -\boldsymbol{\nabla}_i \mathcal{H}^{KP} = -J\boldsymbol{\nabla}_i \sum_i \frac{(\boldsymbol{q}_{i-1} - \boldsymbol{q}_i) \cdot (\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)}{|\boldsymbol{q}_{i-1} - \boldsymbol{q}_i|\,|\boldsymbol{q}_{i+1} - \boldsymbol{q}_i|} \tag{3.5}$$

depending on the position of the adjacent beads, yielding $\boldsymbol{F}_i^{rig} = \boldsymbol{F}_i^{rig}(\boldsymbol{q}_{i-1}, \boldsymbol{q}_i, \boldsymbol{q}_{i+1})$.

The same set of forces acts on the spider's body as well, summed over all the silklines attacked to it. The spider also perceives the action of gravity, whose effect is neglected on the lighter filaments in line with previous models.

3.2.2 The virtual wind environment

We test this new mechanical model in a controlled environment and generate a first database that works as a benchmark to understand the predictive power of optimization algorithms regarding spider ballooning.

We define a synthetic wind field with a horizontal velocity component linearly increasing with height combined with a stochastic component which acts primarily on the vertical direction. This gives rise to the following velocity field:

$$\begin{cases} u_x = kz - f(t)\ \mu(x)z \\ u_y = 0 \\ u_z = f(t)\mu'(x)\frac{z^2}{2}. \end{cases} \tag{3.6}$$

Incompressibility of the flow is preserved as $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$. $\mu'(x)$ is a spatial gate function which localizes the effect of the stochastic component close to the starting point and $\mu(x)$ is its primitive function. $f(t)$ is the temporal function responsible for the randomness. It consists of small Gaussian bumps acting on the vertical velocity mimicking the behaviour of brief ascending or descending currents.

This simplified field is conceived in such a way that the outcome of the flights is highly diversified, making it easier to distinguish between good and bad moments for take-off.

We are now provided with all the ingredients to create a database of flown distances and corresponding perceived wind sequences. Simulations of spider flight are organized as follows:

- the spider is located on a starting point with little elevation with respect to the ground;
- from there it witnesses a certain wind sequence for a limited amount of time, while letting its draglines fly but remaining attached to the floor with its body;
- after that it takes flight and we simulate its dynamics until it touches the ground at a certain distance from the starting point.

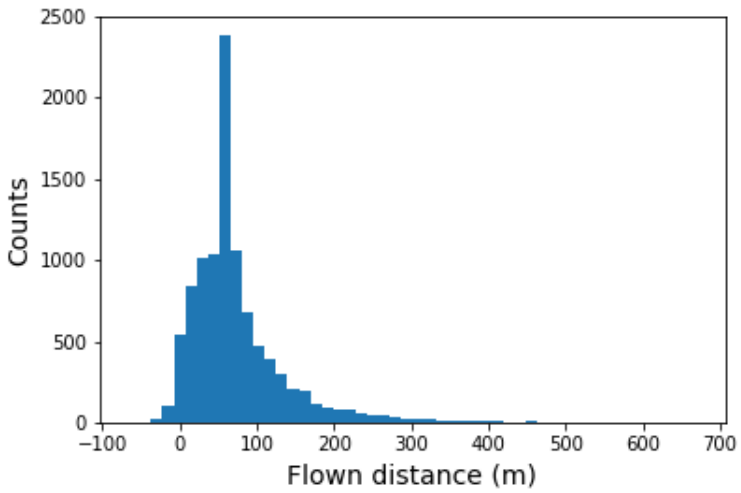

Figure 3.1: Histogram of the flown distances of a batch of 10000 spiders corresponding to 10000 different wind sequences.

A histogram of flown distances is reported in fig. 3.1. The peak is due to all those sequences characterized by no bumps close to the take-off moment, resulting in an almost deterministic fall.

We want to investigate the connection between the wind sequence witnessed by the spider in its original location right before take-off and the corresponding flown distance: is it possible to predict the outcome of the flight from the observations provided before taking off? This will be the topic of the next section.

3.3 The decision-making algorithm

The presence (or absence) of a relation between the observations collected at the initial point and the outcome of the flight makes the case for the possibility to decide if it is convenient to take off or not. Of course if randomness prevailed, there would be no room for prediction and ballooning would uniquely rely on probabilistic considerations, as suggested by [39]. We will show in the following that this is not the case in the stochastic virtual environment that we built up.

As in the previous chapters we face the problem of defining the relevant observables to monitor in order to make well-informed decisions. In principle this can be a very complicated question to answer, especially in a context in which even phenomenological studies do not completely agree on the physical motivations of spider ballooning. In practice in this analysis we restrict our field of investigation to wind velocities exclusively, knowing that this could result insufficient when dealing with more complex flows or experimental databases.

This task can be cast as a *two-armed bandit* problem. It is a simpler version of the

Reinforcement Learning problem where the agent-environment loop is replaced with a choice between two actions as in fig. 3.2. After sensing the wind velocity for a certain time interval, the agent – the simulated spider – has to choose whether to take-off or to give up. The first action leads to a stochastic reward which is proportional to the landing distance, while the second one has the sure result of giving zero reward. The comparison between the two possibilities changes according to where we decide to put the threshold between what we judge as a successful attempt and an unsuccessful one. This threshold can be considered as a compensation for the cost of performing ballooning, which means an expenditure in terms of energy for the spider that is rewarding only if it actually seizes a substantial displacement.

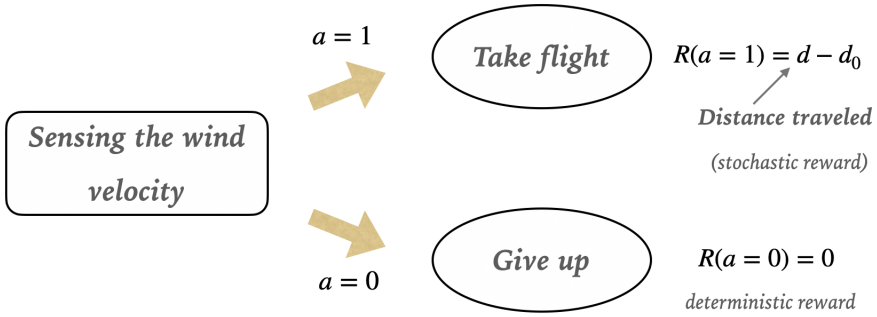

Figure 3.2: Spider ballooning as a two-armed bandit problem.

Once we have fixed the threshold distance d_0, the goal reduces to predict whether the traveled distance will be $d > d_0$, making take-off more convenient, or $d < d_0$, preferring giving up. We assign outcome $y = 1$ to the former case and $y = 0$ to the latter and we proceed by classifying wind sequences.

In this set-up we focus on the vertical velocity presented to the spider before taking off, but in principle we could extend our analysis also to other wind properties over time. We let the agent witness the development of vertical velocity $u_z(t)$ for a time interval comparable to the duration of the tip-toeing behaviour of spiders in nature (we select $\Delta t = 10\,s$ to be a reasonable waiting time with respect to the observational study [36]). After that, we ask to predict the outcome of the flight. To do this, we train a simple logistic regression model.

We sample the vertical velocity in N points every δt with $\Delta t = N\delta t$, then we feed it as an input to the logistic function, reading

$$p_\beta(y = 1|u_z(t_1), ..., u_z(t_N)) = \frac{1}{1 + \exp\left[\beta_0 + \sum_{i=1}^{N} \beta_i u_z(t_i)\right]}. \tag{3.7}$$

This returns us the estimate of the probability of success by taking off as a function of a vector of weights $\boldsymbol{\beta} = (\beta_0, ..., \beta_N)$. We can further refine this estimate by adjusting the weights seeking to maximize the log-likelihood of the model, for example using gradient

ascent.

This process leads to an approximation of the outcome which is far from being precise, as it is portraited in fig. 3.3. It appears that the data is not linearly-separable, resulting in low accuracy. However, it is still possible to draw useful conclusions on the features of wind sequences that undergo the classification task.

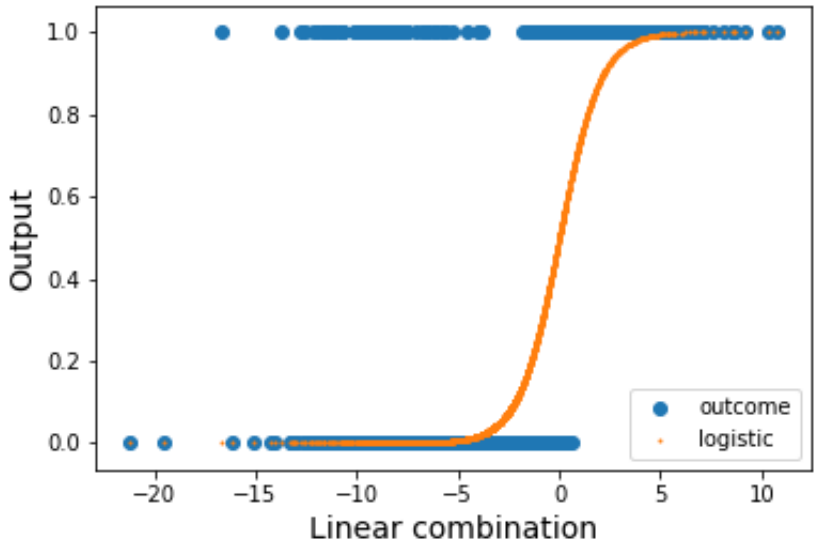

Figure 3.3: Fit of the logistic function between the two outcomes given by the data. The abscissa represents the values of the linear combination $\sum_{i=0}^{N-1} w_i u_z(t_i) + b$ while the ordinate reports the probabilities and the outcomes 0 and 1. The resulting logistic function is plotted in orange while the actual outcomes are in blue.

Let us choose d_0 to be one standard deviation over the average value of the distribution of flown distances of fig. 3.1. If we look at the average of wind sequences weighted by $p_\beta(y = 1)$ and by $p_\beta(y = 0) = 1 - p_\beta(y = 1)$ respectively, we obtain the mean profiles reported in 3.4a. As expected, the logistic regression model learns to associate with successful outcomes wind sequences characterized by a positive vertical component close to the moment of take-off. An ascending wind plume starting before the moment in which the spider has to choose whether to take off is the best suited situation in order to maximize the distance of dispersal. Very similar profiles can be obtained if we directly average the data, manually dividing it according to their outcome (fig. 3.4b).

Finally, we look at the resulting weight distribution reported in fig. 3.5 with $N = 100$. Again we face confirmation regarding the fact that the model learns to give higher importance to information close to the end of the wind sequence, assigning bigger weights to the velocity samples right before take-off. The relevance of information decays going back with time.

The results of this preliminary analysis on a first database are promising and agree

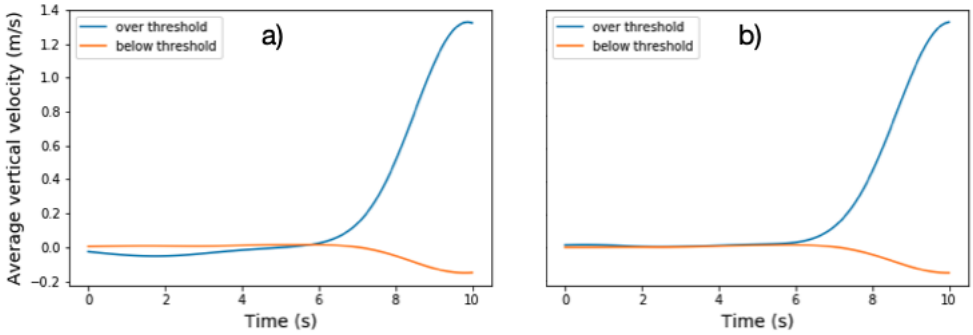

Figure 3.4: a) Average value of the vertical velocity over time weighted by the probability of success $p_\beta(y = 1)$ (blue curve) vs. probability of failure $p_\beta(y = 0)$ (orange curve). b) Average value of the vertical velocity for sequences resulting in successful flights (blue curve) and unsuccessful ones (orange curve).

with the expectations. This leaves room for further development both in data production and in algorithmic refinement, as we are going to discuss in the next section.

3.4 Conclusions and future perspectives

In this chapter we showed how to frame the behaviour of ballooning spiders as a decision-making process. The scheme reported in fig. 3.2 is quite general and remains true across different environments and choices of observables.

We tackled this problem with simple tools coming from statistics, being able to draw some preliminary conclusions on the predictive power of this analysis. From the employment of logistic regression on this data-set, we can state that:

- at least in this simple wind environment, it is possible to extract the main features of the wind velocity sequences that will lead to the goal, that is the maximisation of the dispersion distance;
- those features are intuitive and understandable and confirm the reasonable expectation of individuating wind sequences characterized by a positive vertical component close to take-off time;
- the limits of regression are highlighted by the low accuracy in discriminating between good and bad outcomes, signaling the presence of non-linearity effects that are not captured by this analysis.

This stands as a proof of concept of the validity of this approach. Further refinement moves in two directions: on one hand there is the algorithmic development, that necessarily moves from regression towards non-linear architectures and online analysis; on the other we have the need for data coming from less controlled environments, where fluctuations are not localized in space and more realistic flows are contemplated.

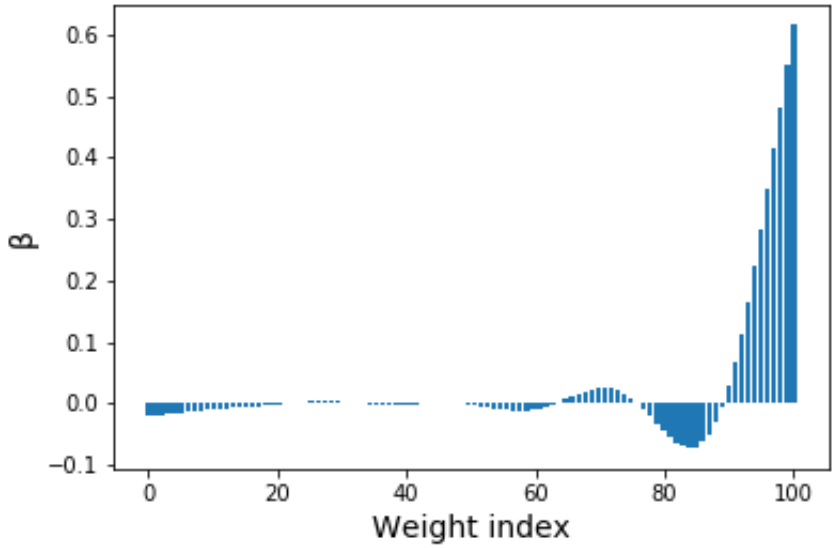

Figure 3.5: Final weight values for 10 s sequences sampled every 0.1 s.

As far as the first is concerned, we did not actually employ Reinforcement Learning techniques to solve the decision problem. In fact, the analysis that we perform works on static data, pretty much as a supervised learning algorithm: we can visualize logistic regression as learning with a perceptron with one output neuron reproducing the probability of success. The natural evolution of this task is the decision of the take-off time, similarly to what happens in reality: as information from the environment flows in, the agent waits for the best time to take flight and become airborne. The analysis on the incoming signal can be performed for example using temporal logic features [54], which are particularly suited for online data monitoring.

With the refinement of the learning algorithm designed to solve the decision problem in study, we can plan to address more complex environments inevitably producing noisier data. We have at our disposal a mechanical model of a ballooning spider which reproduces credible results in a linear gradient field and we can test it in more realistic flows like the Couette channel flow employed in chapter 1. This would represent a challenge for the learning algorithm in order to understand how much its power of prediction can be extended towards the real-world scenario.

Conclusions

Throughout the course of this thesis work we have inspected three instances of optimization processes that involve the exploitation of wind power. We introduced the field of Airborne Wind Energy, which represents an innovative and sustainable alternative for the future of energy production and of shipping transport. The need for control of the motion of the kite employed for energy harvesting constitutes a very interesting problem, which we addressed with techniques in the framework of Reinforcement Learning, arguing that they can represent a valid alternative to the traditional control methods that are currently employed on this task.

The results that we obtained by experimenting this novel approach to Airborne Wind Energy are promising, as we have at our disposal one algorithm that showcased a solid performance both when used to tow a vehicle and when applied to energy extraction. The advantages of this solution lie in its model-free nature, allowing for plasticity and possible application to different architectures. Its deployment on two different tasks like we did in chapters 1 and 2 works as a proof for its versatility. This asset is particularly valuable in a field of technology characterized by the development of many different prototypes, each of them trying to tackle the challenge of power maximization while limiting the risk of crashes.

Another strength of our method is its interpretability: it turns out that even in a complex turbulent environment the learning process comes out with a strategy that can be translated into a simple set of rules which are easily reproducible and can constitute a proper benchmark for on-field testing. The main requisite for interpretability lies in the limited need for data by this algorithm, which is able to reach a satisfying performance having access to very few pieces of information regarding the underlying dynamics. This keeps the learning extremely light and transparent at the same time.

We also tried a different approach employing a deep-Reinforcement Learning algorithm to optimize Airborne Wind Energy extraction, in principle working at the opposite side of the spectrum, providing with the possibility to deal with more articulate state spaces. However, we encountered some difficulties in searching for hyper-parameters values allowing to converge to the best strategy. Nonetheless, the analysis of the neural network grants further depth to the choice of the relevant observables to be monitored while controlling the kite.

We remain confident that this novel approach to Airborne Wind Energy could improve the quality of the control currently exerted on these systems, possibly filling the

gap that there still exists between the state of the art technology and its large-scale commercialization.

Regarding the phenomenon of spider ballooning, we addressed a problem that shared many similarities with AWE, starting from the essential feature of using wind as a source of energy, in this case to favour the spider dispersal in space. Differently from the work on AWE, where we adopted existing models to mimic the dynamics of the system, we developed a new mechanical model of a ballooning spider. We tested it in a simplified wind environment characterized by stochastic gusts on the vertical component of the wind velocity, diversifying the outcome of the spider flight according to the moment of take-off. By doing that we distilled an instance of the decision-making process that spiders appear to undergo any time they feel the urge to perform ballooning.

By analyzing the data resulting from the simulated flights in this synthetic wind field, we obtained a proof of concept regarding the validity of this approach, being able to identify the main features to look for in order to guarantee a displacement farther than a given distance.

Again this approach seems promising both in terms of explaining a peculiar instance of animal behaviour and in drawing inspiration from it in order to optimize the feature selection process that leads to the prediction of the motion of the air flow starting from local and partial observation.

Appendices

Appendix A

Supplementary Information to Chapter 1

A.1 Detailed model dynamics

In this section we provide all the calculations regarding the integration of the equations of motion of the kite-vehicle system.

Starting from the equations of motion and the relative constraints as follows:

$$\begin{cases} m\ddot{\boldsymbol{x}}_k = \boldsymbol{F}^{aer} + m\boldsymbol{g} - \boldsymbol{T} \\ M\ddot{\boldsymbol{x}}_v = \boldsymbol{F}^{\mu} + \boldsymbol{N} + M\boldsymbol{g} + \boldsymbol{T} \\ z_v = 0 \\ |\boldsymbol{r}(t)| = R \quad \text{where } \boldsymbol{r} = \boldsymbol{x}_k - \boldsymbol{x}_v \end{cases} \tag{A.1}$$

where we explicit the fixed tether length and the position of the vehicle on the ground.

From the equilibrium of the forces on the z axis for the vehicle, the modulus of the normal reaction of the ground on the vehicle can be written as:

$$N = Mg - T_z.$$

The friction force is exerted between the vehicle and the ground, and is proportional to the sum of the forces acting on the vehicle on the vertical direction:

$$\begin{aligned} |\boldsymbol{F}^{\mu}| &= \mu N \\ &= \mu(Mg - T_z) \end{aligned}$$

where μ is the friction coefficient.

To solve the system, let us compute the ratio of the first two equations of system A.1 by the respective masses and subtract the resulting equations:

$$\ddot{\boldsymbol{r}} = \frac{\boldsymbol{F}^{aer} - \boldsymbol{T}}{m} - \frac{\boldsymbol{F}^{\mu} + \boldsymbol{N} + \boldsymbol{T}}{M}, \tag{A.2}$$

Let's now compute the derivative of $|\boldsymbol{r}(t)|$ with respect to time (the time dependence in the following passages is suppressed for clarity):

$$\frac{1}{2}\frac{d}{dt}|\boldsymbol{r}|^2 = |\boldsymbol{r}|\boldsymbol{e}_r \cdot \dot{\boldsymbol{r}} = \boldsymbol{r} \cdot \dot{\boldsymbol{r}} = 0,$$

where $\boldsymbol{e}_r$ is the unit vector directed along the cable, labeled by $\boldsymbol{r}$. Deriving a second time we end up with:

$$\ddot{\boldsymbol{r}} \cdot \boldsymbol{r} = -\dot{\boldsymbol{r}}^2$$

We can now equate $-\dot{\boldsymbol{r}}^2$ to the product of equation A.2 with $\boldsymbol{r}$, inserting the constraint of fixed length of the tether indirectly in the system:

$$|\boldsymbol{T}|R(m^{-1} + M^{-1}) - \frac{\boldsymbol{F}^{aer} \cdot \boldsymbol{r}}{m} + \frac{(\boldsymbol{F}^{\mu} + \boldsymbol{N})}{M} \cdot \boldsymbol{r} = \dot{\boldsymbol{r}}^2,$$

This equation can be used to find out the modulus of tension force along the rope direction, by writing explicitly both $\boldsymbol{N}$ and $\boldsymbol{F}^{\mu}$ as functions of $|\boldsymbol{T}|$:

$$|\boldsymbol{T}|R(m^{-1} + M^{-1}) = \frac{\boldsymbol{F}^{aer} \cdot \boldsymbol{r}}{m} - \frac{(\boldsymbol{F}^{\mu} + \boldsymbol{N})}{M} \cdot \boldsymbol{r} + \dot{\boldsymbol{r}}^2$$

The form of the friction force depends on the status of motion of the vehicle. There are three possible situations:

- The vehicle is moving. Then the friction force opposes the speed of the vehicle:

$$\boldsymbol{F}^{\mu} = -\mu N \frac{\boldsymbol{v}_v}{|\boldsymbol{v}_v|} = -\mu \frac{N}{|\boldsymbol{v}_v|}(v_{v,x}\hat{\boldsymbol{x}} + v_{v,y}\hat{\boldsymbol{y}}).$$

  The equation for the tension then becomes:

$$T = \frac{\frac{\boldsymbol{F}^{aer} \cdot \boldsymbol{r}}{m} + \dot{\boldsymbol{r}}^2 - g\left[z_k - \frac{\mu}{|\boldsymbol{v}_v|}(v_{v,x}r_x + v_{v,y}r_y)\right]}{R\frac{m+M}{mM} - \frac{\cos(\theta)}{M}\left[z_k - \frac{\mu}{|\boldsymbol{v}_v|}(v_{v,x}r_x + v_{v,y}r_y)\right]}.$$

- The vehicle is still and $|\boldsymbol{T}_{xy}| \geq \left|\boldsymbol{F}^{\mu}_{Max}\right|$. Then the friction force opposes the tether tension on the xy plane:

$$\boldsymbol{F}^{\mu} = -\mu N \frac{\boldsymbol{T}_{xy}}{|\boldsymbol{T}_{xy}|} = -\mu \frac{N}{|\boldsymbol{T}_{xy}|}(T_x\hat{\boldsymbol{x}} + T_y\hat{\boldsymbol{y}})$$

  yielding

$$T = \frac{\frac{\boldsymbol{F}^{aer} \cdot \boldsymbol{r}}{m} + \dot{\boldsymbol{r}}^2 - g\left[z_k - \mu(\cos(\phi)r_x + \sin(\phi)r_y\right]}{R\frac{m+M}{mM} - \frac{\cos(\theta)}{M}\left[z_k - \mu(\cos(\phi)r_x + \sin(\phi)r_y)\right]},$$

- The vehicle is still and $|\boldsymbol{T}_{xy}| < \left|\boldsymbol{F}^{\mu}_{Max}\right|$. Then the friction force is equal and opposite to the tether tension on the xy plane

$$\boldsymbol{F}^{\mu} = -\boldsymbol{T}_{xy}$$

The modulus of the tension in this case becomes:

$$T = \frac{\frac{\boldsymbol{F}^{\boldsymbol{aer}}\cdot\boldsymbol{r}}{m} + \dot{\boldsymbol{r}}^2 - gz_k}{R\frac{m+M}{mM} - \frac{\sin(\theta)}{M}(\cos(\phi)r_x + \sin(\phi)r_y) - \frac{\cos(\theta)}{M}z_k}$$

Expressing the components for the accelerations of kite and vehicle in the three dimensions, one can write:

$$\begin{cases} m\ddot{x}_k = F^{aer}_x - T_x \\ m\ddot{y}_k = F^{aer}_y - T_y \\ m\ddot{z}_k = F^{aer}_z - T_z - mg \\ M\ddot{x}_v = F^{\mu}_x + T_x \\ M\ddot{y}_v = F^{\mu}_y + T_y \\ M\ddot{z}_v = 0. \end{cases} \tag{A.3}$$

These equations are integrated using Euler method with integration step $\Delta t = 0.001\,s$. Since the constraint $|\boldsymbol{r}(t)| = R$ is inserted indirectly in the dynamics, the distance between vehicle and kite varies after the integration. To avoid this, the kite position is adjusted according to:

$$\boldsymbol{x}'_k = \boldsymbol{x}_k + \frac{\boldsymbol{x}_k - \boldsymbol{x}_v}{|\boldsymbol{x}_k - \boldsymbol{x}_v|}R.$$

A.2 Turbulent flow structure

The turbulent Couette Channel flow is a shear-driven motion of an incompressible fluid bounded by two parallel walls in relative motion. The governing equations of the velocity field are the incompressible Navier-Stokes equations, that in dimensional form read

$$\begin{cases} \partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla\cdot(\nabla\mathbf{u}) \\ \nabla\cdot\mathbf{u} = 0, \end{cases} \tag{A.4}$$

where ρ is the fluid density and ν is the kinematic viscosity.

In order to deal with dimensionless equations we introduce the following variables: $\mathbf{x}^* = \frac{\mathbf{x}}{L}, \mathbf{u}^* = \frac{u}{U}, t^* = \frac{t}{L/U}, p^* = \frac{p}{\rho U^2}$, where U and L are the characteristic velocity and length of the problem in study. In our case we take the height 2δ of the channel to be the characteristic length and the upper wall velocity U_W as the characteristic speed. The resulting dimensionless equations are

$$\begin{cases} \frac{\partial \mathbf{u}^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla \mathbf{u}^* = -\nabla p^* + \frac{1}{Re} \nabla \cdot (\nabla \mathbf{u}^*) \\ \nabla \cdot \mathbf{u}^* = 0 \end{cases} \tag{A.5}$$

where the Reynolds number $Re = \frac{2\, U_W \delta}{\nu}$ remains the only defining parameter of the flow, determining the ratio between the weight of the inertial force term and the one of the viscous term. As the magnitude of the inertial term continues to increase and eventually becomes much larger than the viscous term, there results a "runaway" flow instability whereby the damping effect is marginalized. Therefore, 3D instabilities form and become rampant as Re continues to increase. From this critical point on ($Re > 1500$), the flow starts to show the trademarks of turbulence.

The simulation is initially run in a small box with dimensions $2\pi \times \pi \times 2\, m^3$, respectively corresponding to the length of the x, y and z axes. It follows that the characteristic length for this system $\delta = 1\, m$.The upper wall velocity in this configuration is $U_W = 3\, m/s$ and the kinematic viscosity $\nu = 1/10935\, m^2/s$. This results in a Reynolds number $Re = 65610$, giving rise to a highly turbulent flow.

Dimensions and wall velocity are later re-scaled in post-processing, resulting in the $32\pi \times 32\pi \times 100\, m^3$ channel that we use for running the learning simulation, together with $U_W = 30\, m/s$.

In fig. A.1 and A.2 we report the analysis we performed on the final flux, both along time and on a single frame of the flow. Results are in line with those predicted in ref. (3).

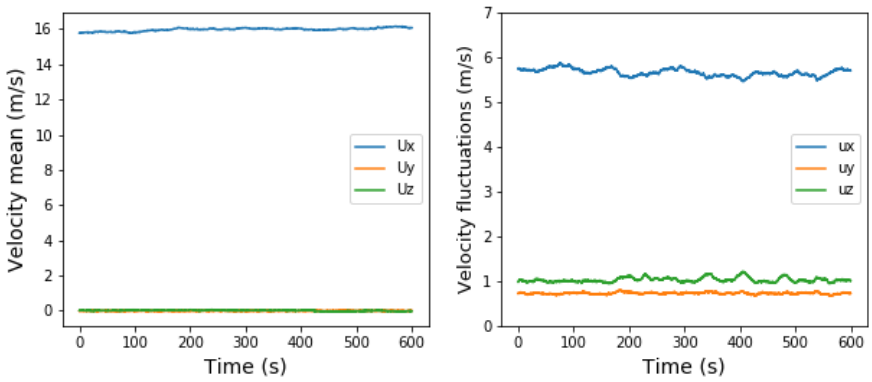

Figure A.1: Behaviour of the overall mean and standard deviation of the three components of the velocity plotted with respect to time in seconds from 0 to the entire duration of the flow, being $600s$. The mean and the standard deviation are computed by averaging over all the points of the grid at each time step. Each snapshot of the flow is taken at a $0.2s$ distance from the previous one, resulting in 3000 total frames.

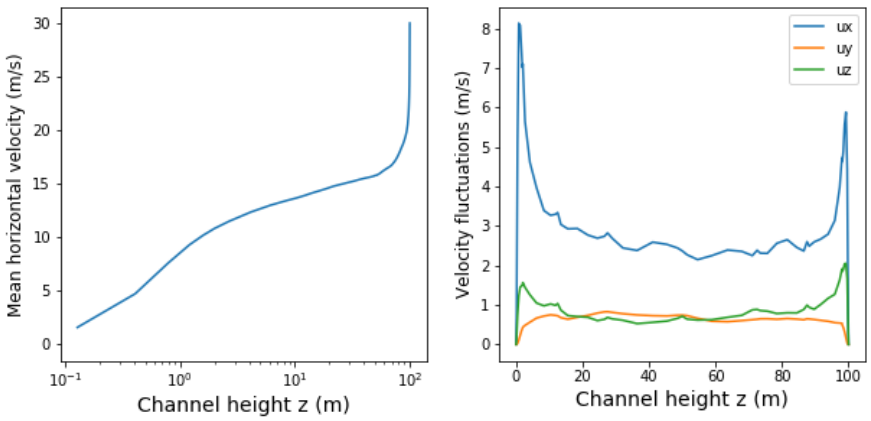

Figure A.2: Mean and fluctuations at $t = 300s$ obtained by averaging over all the grid points of each plane at a fixed channel height z. As expected the mean velocity scales linearly with height in the viscous sub-layer and then logarithmically afterwards. We portrait just half the channel height in the left plot and the entire spectrum in the right one, in order to highlight the regularity in the mean fluctuations close to the lower and upper walls.

A.3 Simulation parameters

Speaking about the values of the parameters employed to simulate the behaviour and the learning of the kite-vehicle system, we first have to disclaim between the *environmental* parameters, such as the physical characteristics of the kite, the duration of the episodes and the magnitude of the penalty, and the *learning* parameters, regarding the magnitude and the scheduling of learning and exploration rate, and the length of the training.

The former ones are resumed in table A.1 and are kept constant in all simulations.

Particular attention must be kept on the control angles α and ψ and the perceived angle β. For every simulation we used the discretized attack angles listed in table A.2 with the relative lift and drag coefficients. As for the bank angles we employed $\psi \in [-15°, 15°]$ and the spacing between adjacent banks is $\Delta\psi = 3°$. In the turbulent simulation we account for the angle β between the relative velocity and the xy-plane as well. The latter is binned uniformly between with $\beta \in [-80°, 80°]$ with $\Delta\beta = 20°$. This amounts to $8 \times 11 \times 9$ total observables (which represent the states of the learning process) in a turbulent flow, while it is limited to 8×11 for simpler flows as we will see later.

We now shed light on the values of the learning parameters we used for training. Values and scheduling of learning rate and exploration rate are reported in table A.3 and are valid both for trainings through the turbulent channel and for simpler settings. What can be changed is the total number of learning steps N_{steps}, which is higher for the turbulent trainings as the space of observables is larger and the learning is more

Parameter	Symbol	Value
Vehicle mass	M	$40\,kg$
Kite mass	m	$1\,kg$
Cable length	R	$50\,m$
Friction coefficient	μ	0.4
Kite area	A	$5\,m^2$
Integration time	Δt	$0.001\,s$
Penalty value	P	1000
Threshold time for P	T_{pen}	$200\,s$
Penalty threshold for z	z_{low}	$10\,m$
Training ep. duration	T	$350\,s$
Evaluation ep. duration	T_{eval}	$600\,s$
Decision time	$\Delta t_{decision}$	$0.25\,s$

Table A.1: Values of the employed environmental parameters for the kite-vehicle system.

Attack angle α	$C_L(\alpha)$	$C_D(\alpha)$
6	0.65	0.05
8	0.75	0.07
10	0.82	0.09
12	0.9	0.1
14	1.0	0.13
16	1.08	0.18
18	1.1	0.18
20	1.05	0.21

Table A.2: Lift coefficient C_L and drag coefficient C_D as functions of the attack angle α. The employed values listed in the table are taken from ref. [24].

difficult.

The scheduling used for the learning rate is

$$\eta_j(S_t, A_t) = \frac{\eta_0}{1 + \left(\frac{n_j(S_t,A_t)}{N_0}\right)^\gamma}$$

and is function of the number of visits $n_j(S_t, A_t)$ to each observation-action couple. The exploration rate is instead function only of the training time j and has an initial constant interval N_{burn} and then shows a power-law decay as follows

$$\epsilon_j = \frac{\epsilon_0 \epsilon_c}{\epsilon_c + (j - N_{burn})^\delta}.$$

A graphic representation of these two power law decays can be found in fig. A.3 with $N_{steps} = 6 \cdot 10^8$.

As far as the penalty for crashing is concerned, it does not remain the same along one episode: crashing is penalized only during the first 200 s of each episode, with $P = 1000$ and then is not penalized anymore. The penalty for flying low instead remains the same along the whole episode (and we select $z_{low} = 10\,m$ as reported in table A.1). This hard scheduling enhances the finding of a reliable policy to overcome the transient from the initial conditions to the stable pattern.

All the values of the parameters of the simulation with turbulent wind are reported in table A.3. Time values are expressed in terms of number of learning steps. One learning step follows the other by 0.25 s.

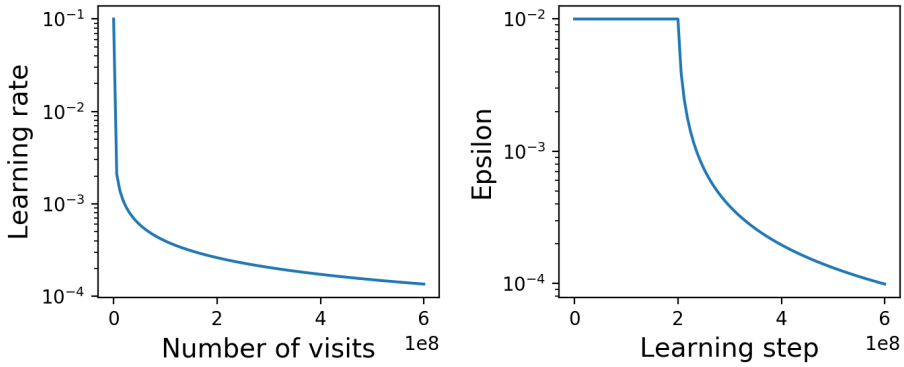

Figure A.3: Graphical representation of the scheduling of $\eta(S_t, A_t)$ and ϵ_t. On the left the abscissa represents the number of visits to a certain observation-action couple $n(S_t, A_t)$ while on the right it is shown the dependence between the exploration rate and the learning step. In this simulation we resort to $6 \cdot 10^8$ learning steps.

Parameter	Value
η_0	0.1
N_0	10000
γ	0.6
ϵ_0	0.01
ϵ_c	$N_{steps}/150$
N_{burn}	$N_{steps}/3$
δ	1

Table A.3: Values of the training parameters for any training. N_{steps} refers to the total number of learning steps of the training and is equal to $6 \cdot 10^8$ for the turbulent simulations, while $N_{steps} = 10^7$ for simpler set-ups.

A.4 Learning in simpler set-ups and with different reward distribution

As anticipated in section A.3, we performed various simulations in a simpler environment as well, both as a test bench for the more complex turbulent problem and also in order to try out different reward structures. In this scheme we consider wind velocity to be constant along the x-axis and uniform. This results in a shorter search for the best strategy, as a much smaller N_{steps} is needed with respect to the one employed for the trainings in a turbulent environment.

We find out that the system is able to learn effectively (fig. A.4) only knowing the value of the control angles. Therefore we can discard the information regarding the angle β, whose knowledge does not improve the performance of the algorithm. This allows us to work with an even more restricted observation-space.

The maximum witnessed return is comparable with the one we witness in the turbulent flow but the parallel is not particularly meaningful either: we selected a constant wind velocity similar to the average velocity measured in the turbulent flow over all the channel, but it is also true that the fraction of the field witnessed by the kite has usually smaller velocity values.

The resulting trajectory is a stable helix-like motion along the episode as portraited in fig. A.7. Also in this case the strategy is stable enough to make the kite remain airborne also for longer periods of time with respect to the ones it has been trained on. As for the turbulent case, we train the kite on episodes of length $T = 350\,s$ and then test the policy on longer episodes with $T_{eval} = 600\,s$ (cf. table A.1 for all the time intervals employed in the simulations).

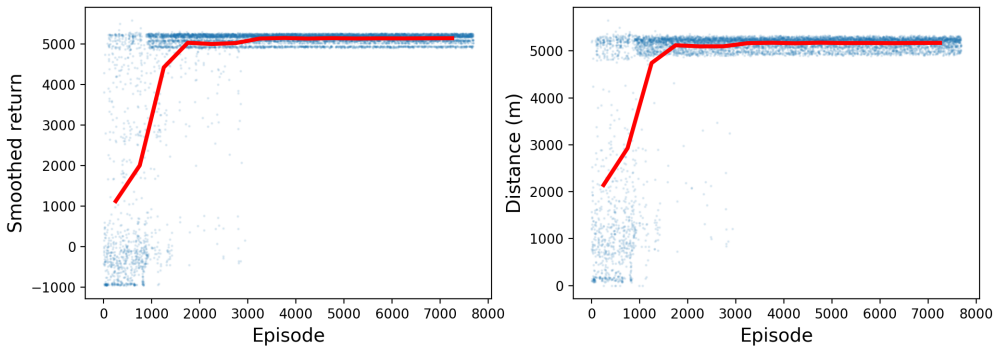

Figure A.4: Curve of the return (left) for a training with $N_{steps} = 10^7$ with constant wind velocity equal to $15\,m/s$ along x, episodes of length $T = 350s$ and corresponding traveled distances (right), which differ from the return values by the falling penalty and by smaller rewards for $z_k < z_{low} = 10\,m$.

The first test that we perform is the analysis of the best strategy learned at the end of the training. The policy is now easily representable and is reported on the left

part fig. A.5, where each entry stands for an observation S and the corresponding arrow is the performed action. Even if this scheme is pretty noisy, we can recognize two stable points with $\alpha = 8°$ and $\psi = \pm 12°$ respectively, which act as attractors of the observation-action dynamics. This is made more evident by the right part of fig. A.5: if we evaluate the learned strategy, we notice immediately that the two stable points are visited exponentially more times than all the others, resulting in absorbing points of the dynamics.

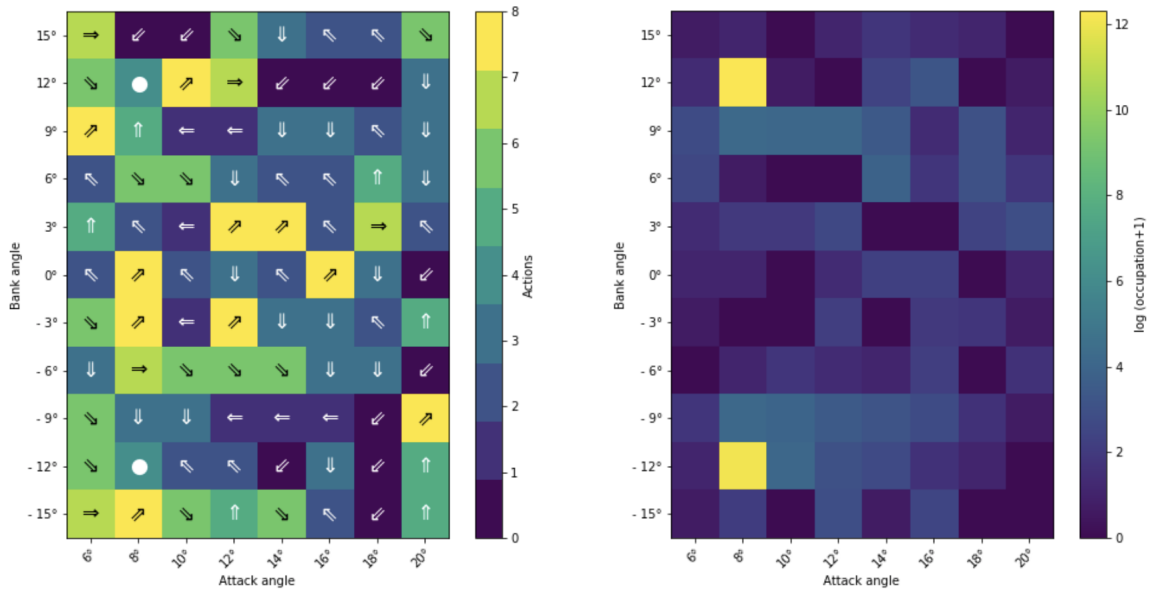

Figure A.5: A picture of the best policy learned at the end of the training (left) and the corresponding observation occupation after 166 evaluation episodes (right). Arrows and circles represent the best action to be taken according to the maximization of the learned Q. The circle stands for keeping attack and bank angles fixed. This figure highlights the presence of two grid-points being the attractors of the observation-action dynamics.

We can then resort to gather this information and build a simplified policy which drives the dynamics directly towards these two absorbing points avoiding the more intricate cycles portraited by fig. A.5 (left). The new policy that we design is represented in fig. A.6.

If we proceed by testing this new policy we find out that the performance is the same as the one obtained at the end of the training, suggesting that the relevant information stands in locating the absorbing points and all the actions that do not point directly there are just the result of noisy learning.

This conclusion is almost analogous to the one drawn in the chapter with respect to the turbulent flow learning, where the performance of the distilled policy and of the learned one are comparable, nonetheless in that case the transient is responsible for a higher variance.

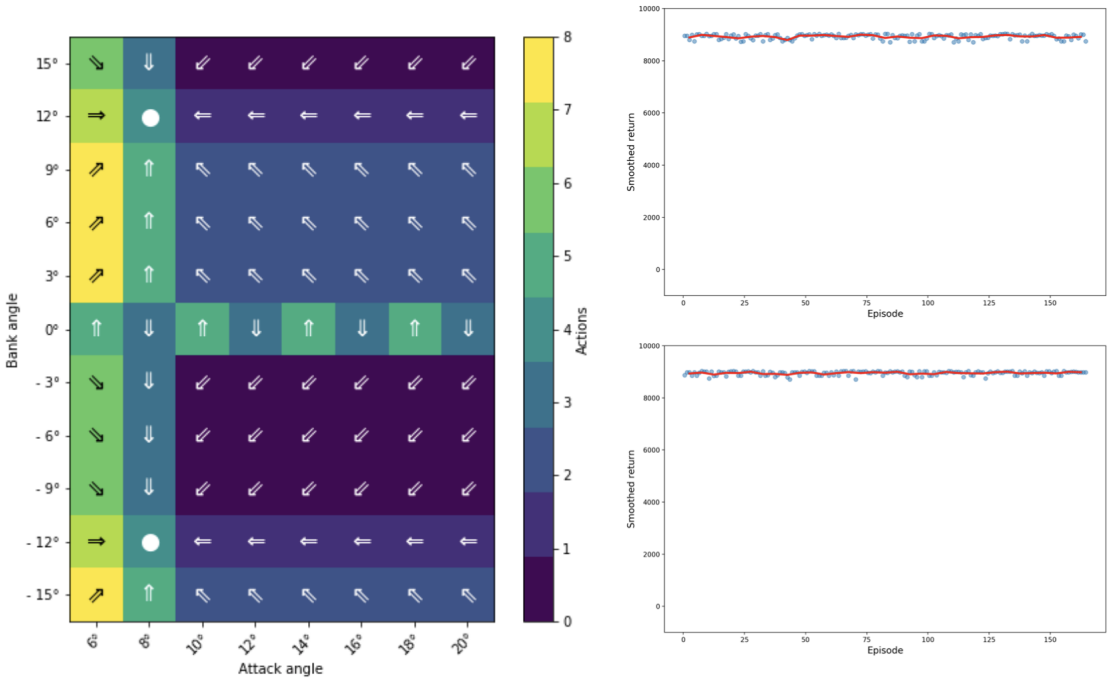

Figure A.6: On the left we have the resulting schematized policy obtained by combining the information coming from fig. A.5. We drew the shorted path towards the attractors and the performance of the policy is invariate: on the right there are the two plots of the returns collected during the evaluation of the learned policy and of the distilled one for episodes of length $T = 600\,s$.

The other test that we ran in this simple environment is the study of the learning in contexts where the reward structure is different from the original one. If in the main work we award the vehicle for moving as far as possible in the x direction, here we tried to enquire whether it is possible to make the system move also in other directions even if the wind continues to remain directed along x.

The obtained results are summarized in fig. A.8 and show sensibly smaller returns with respect to the one of fig. A.4. This happens since the vehicle is not able to learn efficiently how to exploit the horizontal wind in order to move upwind: it comes out that the followed strategy remains very similar to the original one, with one stable attractor around $\alpha = 8°$ and $\psi = 12°$ (fig. A.9). The negative absorbing state vanishes since we privilege moving in the positive direction of the y-axis.

This suggests that the dynamics is not rich enough in order to allow the due plasticity required to the policy in order to make the system move efficiently in a direction which does not correspond to the one given by the wind.

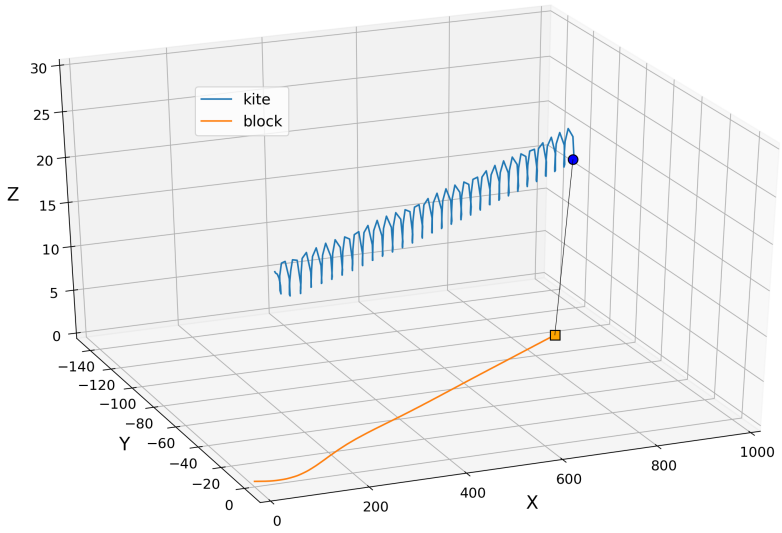

Figure A.7: An example of the initial trajectory of the kite-vehicle system evaluated using the learned policy. The helix motion is much more regular than in the turbulent case as the environment is completely predictable and fluctuations are absent.

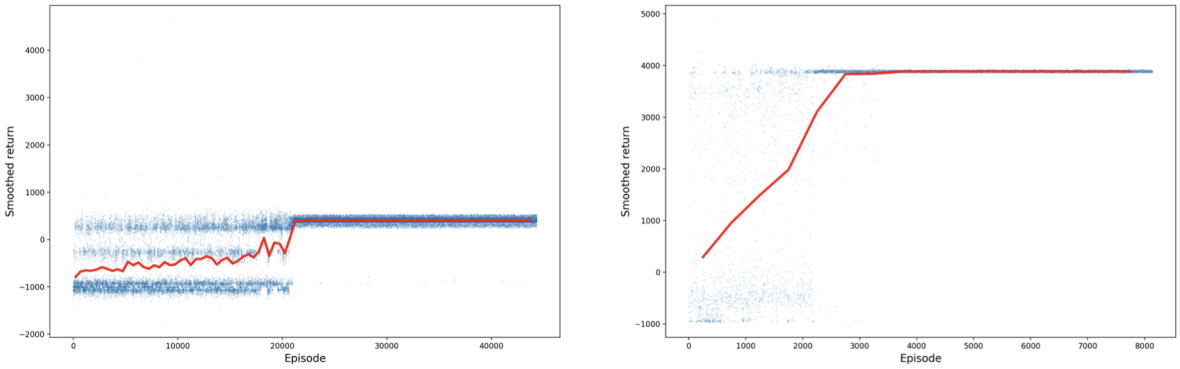

Figure A.8: Return growth in a constant wind environment along the x-axis with a reward that awards the space travelled along y (left) and the space along the bisectrix of the xy-plane (right). In both cases the obtained return is sensibly smaller than in the usual reward scheme.

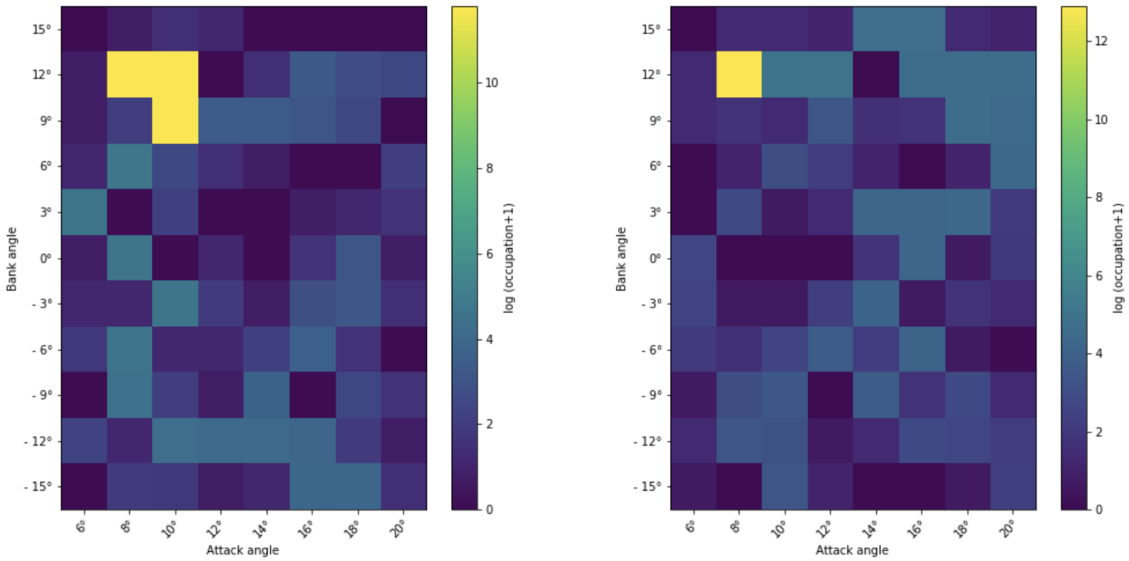

Figure A.9: Learning state occupation after evaluation for the case with the reward given along the y-axis (left) and the reward assigned for the space traveled along the bisectrix of the xy-plane (right).

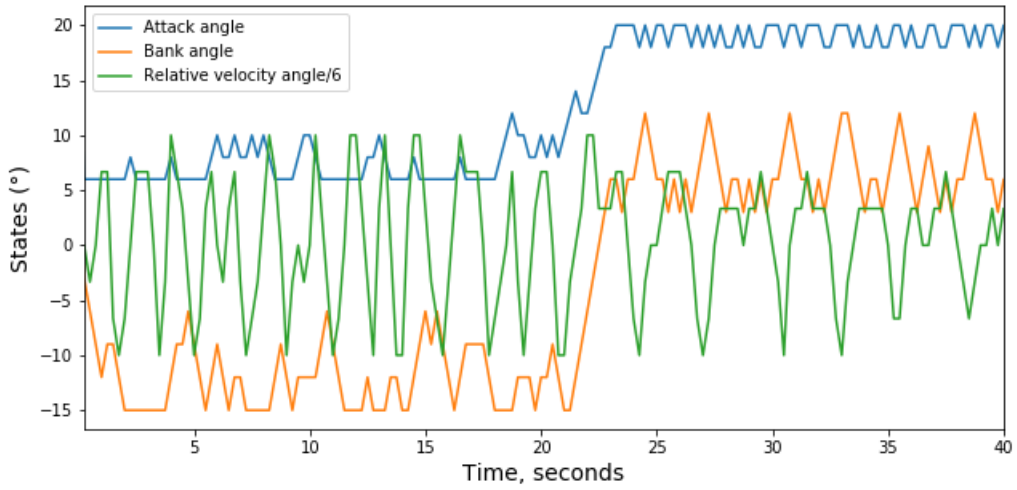

Figure A.10: Another example of learning state-action trajectory portraiting the beginning of an evaluation episode: in this case the transient phase is much longer and lasts around 20 s before moving to the stationary one.

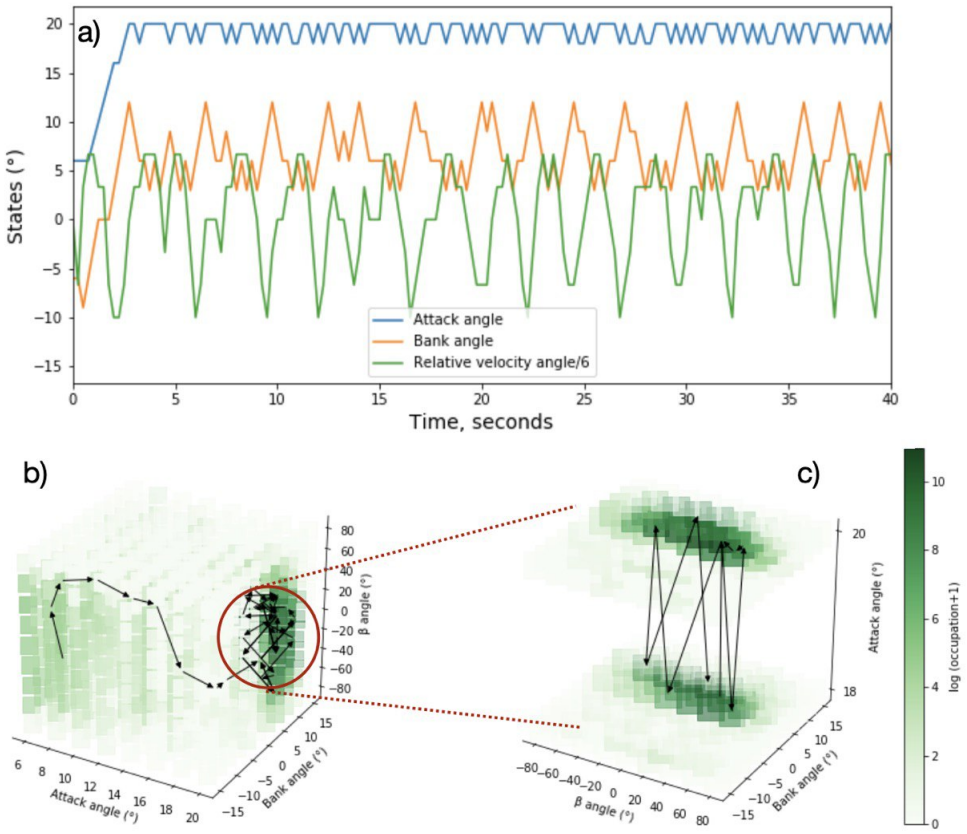

Figure A.11: Figure a) portraits the states visited during the first 40 seconds of flight according to the learnt policy. It is possible to individuate two phases of behaviour: a very brief transient one where the kite reacts to the initial conditions it happens to face, and a stationary one where the attack angle remains close to its maximum values and the bank angle reacts to changes in the angle β reaching its peaks right after wells in β. The same action pattern is represented inside the state space in b); arrows represent actions and their consequences on the environment in terms of β. The transient is mapped into the initial motion towards the states where the stationary phase of the policy is performed. A glance of the stationary behaviour is given in c), showing a complex cycle taking place between $\alpha = 18°$ and $\alpha = 20°$ and that slightly changes according to the fluctuations perceived in the environment. The darkness of the colour stands for $\log[n(s,a)+1]$, $n(s,a)$ being the state-action occupation during the evaluation phase.

Appendix B

Supplementary Information to Chapter 2

B.1 Mathematical model

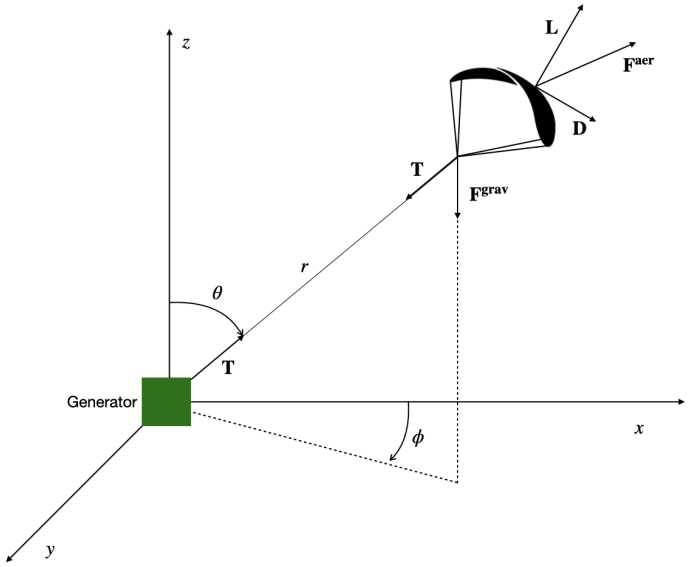

Figure B.1: Model diagram

Let us consider a cartesian coordinate system (x, y, z) centered in the electric machine, with the x axis aligned with the wind speed. In this system, the position of the kite can be expressed using spherical coordinates (θ, ϕ, r), with r being the distance from the origin (the length of the tether) and θ and ϕ the two angles shown in figure B.1.
Then, we can obtain a local coordinate system with unit vectors $(\mathbf{e}_\theta, \mathbf{e}_\phi, \mathbf{e}_\mathbf{r})$, centered in

the position of the kite. These unit vectors can be expressed in the (x, y, z) system as:

$$\begin{pmatrix} \mathbf{e}_\theta \\ \mathbf{e}_\phi \\ \mathbf{e_r} \end{pmatrix} = \begin{pmatrix} \cos(\theta)\cos(\phi) & \cos(\theta)\sin(\phi) & -\sin(\theta) \\ -\sin(\phi) & \cos(\phi) & 0 \\ \sin(\theta)\cos(\phi) & \sin(\theta)\sin(\phi) & \cos(\theta) \end{pmatrix} \tag{B.1}$$

Applying Newton's laws of motion we obtain the following equations:

$$\ddot{\theta} = \frac{F_\theta}{mr} \tag{B.2}$$

$$\ddot{\phi} = \frac{F_\phi}{mr\sin(\theta)} \tag{B.3}$$

$$\ddot{r} = \frac{F_r}{m} \tag{B.4}$$

where m is the mass of the kite.
The total force $\mathbf{F}$ that acts on the kite is the sum of gravity force $\mathbf{F_{grav}}$, apparent force $\mathbf{F_{app}}$, aerodynamic force $\mathbf{F_{aer}}$ and line tension $\mathbf{T}$ (whose only nonzero component is along negative r axis):

$$\mathbf{F} = \mathbf{F^{grav}} + \mathbf{F^{app}} + \mathbf{F^{aer}} - \mathbf{T} \tag{B.5}$$

With the simplifying assumption of a massless tether, the gravity force can be simply expressed as:

$$\mathbf{F^{grav}} = \begin{pmatrix} F_\theta^{grav} \\ F_\phi^{grav} \\ F_r^{grav} \end{pmatrix} = \begin{pmatrix} mg\sin(\theta) \\ 0 \\ mg\cos(\theta) \end{pmatrix} \tag{B.6}$$

The apparent force that comes into play is the centrifugal force, computed as:

$$\mathbf{F^{app}} = \begin{pmatrix} F_\theta^{app} \\ F_\phi^{app} \\ F_r^{app} \end{pmatrix} = \begin{pmatrix} m(\dot{\phi}^2 r\sin(\theta)\cos(\theta) - 2\dot{r}\dot{\theta}) \\ m(-2\dot{r}\dot{\phi}\sin(\theta) - 2\dot{\phi}\dot{\theta}r\cos(\theta)) \\ m(r\dot{\theta}^2 + r\dot{\phi}^2\sin^2(\theta)) \end{pmatrix} \tag{B.7}$$

B.1.1 Aerodynamic force

The derivation of the equations for aerodynamic force requires some additional care. This force depends on the relative wind speed, which is computed as:

$$\mathbf{W_r} = \mathbf{W_w} - \mathbf{W_k} \tag{B.8}$$

where $\mathbf{W_w}$ is the wind speed and $\mathbf{W_k}$ is the kite speed, both with respect to the ground. In the local coordinate system $(\mathbf{e}_\theta, \mathbf{e}_\phi, \mathbf{e_r})$, $\mathbf{W_k}$ can be expressed as:

$$\mathbf{W_k} = \begin{pmatrix} \dot{\theta}r \\ \dot{\phi}r\sin(\theta) \\ \dot{r} \end{pmatrix} \tag{B.9}$$

We define now a kite wind coordinate system with basis vectors $(\mathbf{x_w}, \mathbf{y_w}, \mathbf{z_w})$. $\mathbf{x_w}$ is aligned with $\mathbf{W_r}$ and points from the trailing edge to the leading edge of the kite, $\mathbf{z_w}$ is contained in the kite symmetry plane and points from the top surface to the bottom surface of the kite, $\mathbf{y_w}$ completes the right-handed system.
By its definition, $\mathbf{x_w}$ can be simply computed as:

$$\mathbf{x_w} = -\frac{\mathbf{W_r}}{W_r} \tag{B.10}$$

To obtain an equation for $\mathbf{y_w}$ we first define one of our control inputs ψ, the bank angle of the kite, which we suppose we can control directly by acting on the length of the lines connecting the kite:

$$\psi = \arcsin \frac{\Delta l}{d} \tag{B.11}$$

Then, if we define:

$$\mathbf{e_w} = \frac{\mathbf{W_r} - \mathbf{e_r}(\mathbf{e_r} \cdot \mathbf{W_r})}{|\mathbf{W_r} - \mathbf{e_r}(\mathbf{e_r} \cdot \mathbf{W_r})|} \tag{B.12}$$

and

$$\eta = \arcsin\left(\frac{\mathbf{e_r} \cdot \mathbf{W_r}}{|\mathbf{W_r} - \mathbf{e_r}(\mathbf{e_r} \cdot \mathbf{W_r})|} \tan(\psi)\right) \tag{B.13}$$

We obtain:

$$\mathbf{y_w} = \mathbf{e_w}(-\cos(\psi)\sin(\eta)) + (\mathbf{e_r} \times \mathbf{e_w})(\cos(\psi)\cos(\eta)) + \mathbf{e_r}\sin(\psi) \tag{B.14}$$

Finally, the unit vector $\mathbf{z_w}$ can be computed as:

$$\mathbf{z_w} = \mathbf{x_w} \times \mathbf{y_w} \tag{B.15}$$

The aerodynamic force is the sum of drag and lift, that can be computed as:

$$\mathbf{D} = -\frac{1}{2} C_D A \rho W_r^2 \mathbf{x_w} \tag{B.16}$$

$$\mathbf{L} = -\frac{1}{2} C_L A \rho W_r^2 \mathbf{z_w} \tag{B.17}$$

where ρ is the air density, A is the kite characteristic area and C_D and C_L are respectively the drag and lift coefficients. These coefficients are nonlinear functions of the angle of attack of the kite, which we take as a control input, and will be discussed in section B.2. Finally, $\mathbf{F^{aer}}$ is computed as:

$$\mathbf{F^{aer}} = \mathbf{D} + \mathbf{L} \tag{B.18}$$

B.1.2 Tension

In our system energy is produced by the unwinding of a tether wound on a winch, which puts into rotation the drum of an electric generator. Then, since we do not make any particular assumptions about the unwinding velocity (in [24] it is controlled to be around a reference value), we have to integrate the physics of the drum into our model. To do so, we perform a simplified analysis in which we consider the interaction between drum and tether to happen on a plane orthogonal to the axis of rotation of the drum.

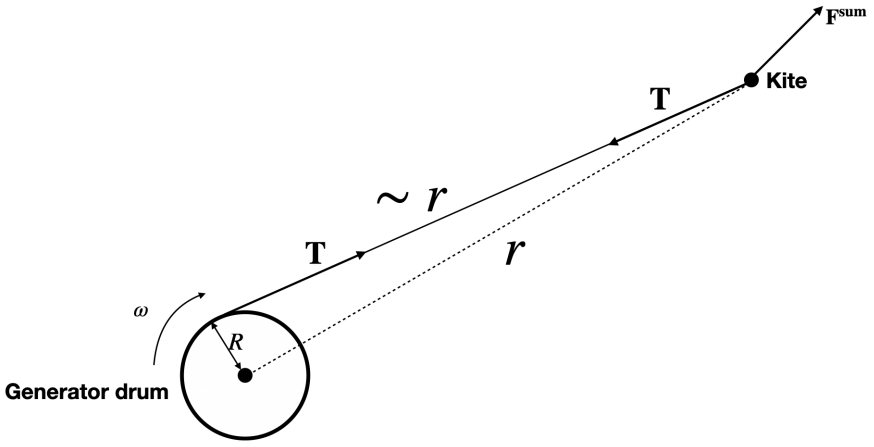

Figure B.2: 2D diagram for tension computation.

When the radius R of the generator drum is much smaller than the distance r between generator and kite, the tension $\mathbf{T}$ is always directed along axis r and points towards the generator. To compute its intensity T, we can apply the rotational form of Newton's laws on the shaft of the generator:

$$T_d - T_r = I\dot{\omega} \tag{B.19}$$

where ω is the angular velocity of the drum, T_d is the driving torque due to the tether tension, T_r is the resistance torque (that we approximate as a viscous friction, proportional to ω), and I is the moment of inertia of the drum.
The angular velocity of the drum is connected to the linear expansion velocity of the tether by the equation:

$$\omega R = \dot{r} \tag{B.20}$$

where R is the radius of the drum.
Since the driving torque applied to the drum is due to the tension, Newton's equation can be rewritten as:

$$TR - k\omega = I\dot{\omega} \tag{B.21}$$

and, substituting ω:

$$TR - \frac{k}{R}\dot{r} = \frac{I}{R}\ddot{r} \tag{B.22}$$

However, we recall that:

$$\ddot{r} = \frac{F_r}{m} \tag{B.23}$$

where F_r is the sum of all forces acting on the kite along the radial axis. This net force can be split into two terms: one is the tether tension T that we are computing and the other is the net force F_r^{sum} resulting from gravity forces, apparent forces and aerodynamic forces acting on axis r:

$$F_r = F_r^{sum} - T \tag{B.24}$$

Then, we obtain:

$$TR - \frac{k}{R}\dot{r} = \frac{I}{R}(F_r^{sum} - T) \tag{B.25}$$

The moment of inertia of a drum can be written in terms of the mass M of the drum as:

$$I = \frac{1}{2}MR^2 \tag{B.26}$$

And, solving the equation for T, we obtain:

$$T = \frac{MF_r^{sum}R + 2m\frac{\dot{r}}{R}k}{2mR + MR} \tag{B.27}$$

B.2 Attack and bank angles

We assume to be able to control the attack and bank angle of the kite by adjusting the length of the ropes connecting the kite to the ground.
These angles allow us to control the trajectory of the kite during the traction phase (and in the subsequent passive phase) in order to optimize energy extraction just like the trajectory of an airplane can be adjusted by the pilot by operating on ailerons and elevators.

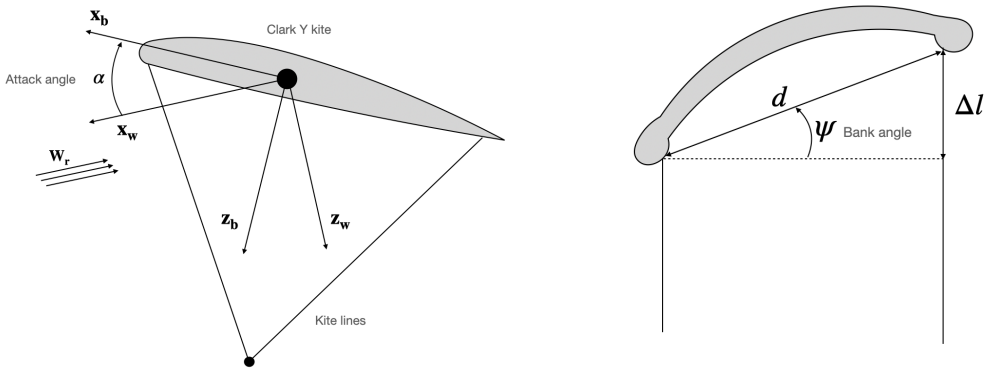

Figure B.3: Attack angle and bank angle.

The bank angle was introduced in section B.1.1 and is depicted in figure B.3. By controlling this variable we can make the kite turn along its longitudinal axis (the axis that connects the trailing edge to the leading edge).
The attack angle (shown in figure B.3) is involved in making the kite soar or glide and it appears in the computation of the aerodynamic forces (drag and lift), since it is one of the main factors that influence the lift and drag coefficients C_L, C_D. To properly define this angle, we have to introduce a kite body coordinate system $(\mathbf{x_b}, \mathbf{y_b}, \mathbf{z_b})$, centered in the center of mass of the kite. The unit vector $\mathbf{x_b}$ is contained in the kite symmetry plane and points from the trailing edge to the leading edge, $\mathbf{z_b}$ points down and is perpendicular to the kite surface, while $\mathbf{y_b}$ completes the right-handed coordinate system. Then, the attack angle α can be defined as the angle between the vectors $\mathbf{x_w}$ and $\mathbf{x_b}$.
The dependence of lift and drag coefficients on the attack angle is quite complex and strongly influenced by the profile of the airfoil. We assume to work with a Clark-Y kite: the values for lift and drag coefficients for such a profile have been estimated in [24] for 15 values of α ranging from $-8°$ to $20°$ and are reported in the following plot and table:

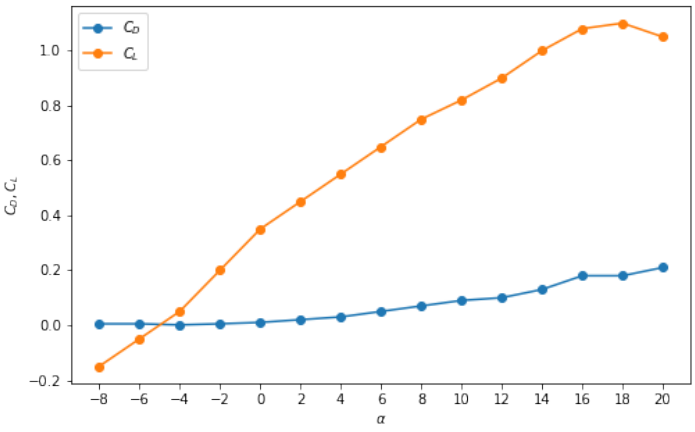

Figure B.4: Lift and drag coefficients as functions of the attack angle.

Attack angle α	C_D	C_L
-8	0.005	-0.15
-6	0.005	-0.05
-4	0.001	0.05
-2	0.005	0.2
0	0.01	0.35
2	0.02	0.45
4	0.03	0.55
6	0.05	0.65
8	0.07	0.75
10	0.09	0.82
12	0.1	0.9
14	0.13	1
16	0.18	1.08
18	0.18	1.1
20	0.21	1.05

Table B.1: Lift coefficient C_L and drag coefficient C_D as functions of the attack angle α. The employed values listed in the table are integrally taken from ref. [24].

B.3 Model configuration

The physical model used throughout all our simulations is a ground-gen fixed-ground-station AWE system in a yo-yo configuration, in which the tether is pulled by a kite with a Clark-Y profile. We suppose the tether to be infinitely long. The parameters of the kite and of the electric machine were kept fixed across all simulations and they are reported in the following table:

Parameter	Value
Kite mass m	1kg
Kite area A	$10m^2$
Drum mass M	10kg
Drum radius R	20cm
Friction coefficient k	$10\frac{\text{Nm}}{\text{rad/s}}$

Table B.2: Kite parameters.

B.4 Details on the simulations

As already explained in A.3, it is important to schedule towards 0 the learning rate η. This can be done by keeping the learning rate constant until a certain time T_η and then using a power law scheduling with respect to the absolute learning time step (i.e. accumulating learning steps across distinct episodes), with an exponent e_η:

$$\eta(t) = \begin{cases} \eta_0 & \text{if } t \leq T_\eta \\ \frac{\eta_0}{(t-T_\eta)^{e_\eta}} & \text{if } t > T_\eta. \end{cases} \tag{B.28}$$

The same kind of scheduling, with a different threshold T_ϵ and exponent e_ϵ is used for the exploration rate ϵ, employed in the ϵ-greedy policy. In B.5 an example of power law scheduling for η and ϵ is shown, with $\eta_0 = 0.02$, $\epsilon_0 = 0.01$, $T_\eta = 7000$, $T_\epsilon = 8000$, $e_\eta = 0.6$, $e_\epsilon = 0.8$.

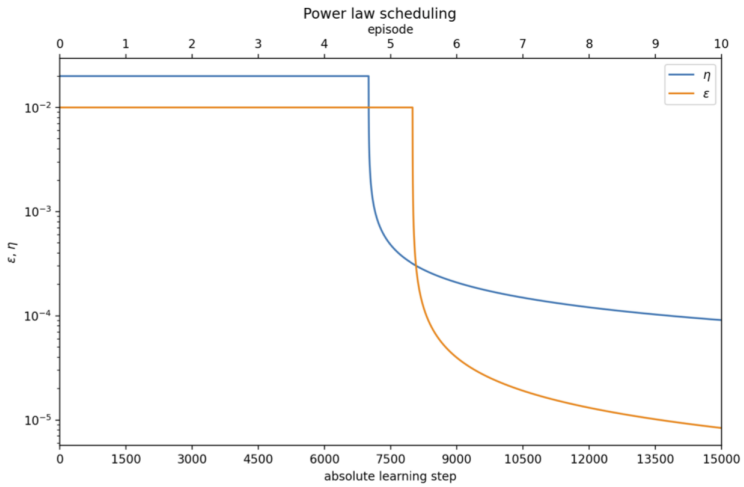

Figure B.5: Example of power law scheduling for learning rate and exploration rate

Here we report in detail the parameters used for the constant wind pattern both for SARSA and DQL:

SARSA	DQL
Number of episodes	8000
Wind speed	10m/s
$\eta_0 = 0.01$	$\eta_0 = 0.001$
$\epsilon_0 = 0.01$	$\epsilon_0 = 0.001$
$T_\eta = 8 \cdot 10^6$	$T_\eta = 10^6$
$T_\epsilon = 6.4 \cdot 10^6$	$T_\epsilon = 5 \cdot 10^6$
$e_\eta = 0.9$	$e_\eta = 0.9$
$e_\epsilon = 1.3$	$e_\epsilon = 1.3$

Bibliography

[1] Uwe Ahrens, Moritz Diehl, and Roland Schmehl. *Airborne Wind Energy*. Green Energy and Technology. Springer, 2013.

[2] Clarence D. Cone. Thermal soaring by migrating starlings. *The Auk*, 85(1):19–23, 1968.

[3] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. 1951.

[4] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6:366–422, 1960.

[5] L. S. Pontryagin. *The mathematical theory of optimal processes and differential games*, volume 169. 1985.

[6] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[7] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[8] Philip Bechtle, Mark Schelbergen, Roland Schmehl, Udo Zillmann, and Simon Watson. Airborne wind energy resource analysis. *Renewable Energy*, 141:1103–1116, 2019.

[9] Antonello Cherubini, Andrea Papini, Rocco Vertechy, and Marco Fontana. Airborne wind energy systems: A review of the technologies. *Renewable and Sustainable Energy Reviews*, 51:1461–1476, 2015.

[10] J.F. Wellicome. Some comments on the relative merits of various wind propulsion devices. *Journal of Wind Engineering and Industrial Aerodynamics*, 20(1):111–142, 1985.

[11] Massimo Canale, Lorenzo Fagiano, Massimo Ippolito, and Mario Milanese. Control of tethered airfoils for a new class of wind energy generator. pages 4020 – 4026, 01 2007.

[12] Robert Haffner. Study on challenges in the commercialisation of airborne wind energy systems. 11 2018.

[13] A. Ilzhöfer, B. Houska, and M. Diehl. Nonlinear mpc of kites under varying wind conditions for a new class of large-scale wind power generators. *International Journal of Robust and Nonlinear Control*, 17(17):1590–1599, 2007.

[14] Massimo Canale, Lorenzo Fagiano, Massimo Ippolito, and Mario Milanese. Control of tethered airfoils for a new class of wind energy generator. pages 4020 – 4026, 01 2007.

[15] Michael Erhard and Hans Strauch. *Automatic Control of Pumping Cycles for the SkySails Prototype in Airborne Wind Energy*, pages 189–213. 04 2018.

[16] Paul Williams, Bas Lansdorp, and Wubbo Ockels. Optimal cross-wind towing and power generation with tethered kites. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 31:81–93, 01 2008.

[17] S. Joe Qin and Thomas A. Badgwell. An overview of nonlinear model predictive control applications. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 369–392, Basel, 2000. Birkhäuser Basel.

[18] I. Argatov, P. Rautakorpi, and R. Silvennoinen. Estimation of the mechanical energy output of the kite wind generator. *Renewable Energy*, 34(6):1525–1532, 2009.

[19] Boris Houska and Moritz Diehl. Optimal control for power generating kites. *2007 European Control Conference, ECC 2007*, 01 2006.

[20] Michael L Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015.

[21] Damien Ernst, Mevludin Glavic, Florin Capitanescu, and Louis Wehenkel. Reinforcement learning versus model predictive control: A comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2009.

[22] Gautam Reddy, Jerome Wong-Ng, Antonio Celani, Terrence J Sejnowski, and Massimo Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, 2018.

[23] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77–82, 2020.

[24] Massimo Canale, Lorenzo Fagiano, and Mario Milanese. High altitude wind energy generation using controlled power kites. *Control Systems Technology, IEEE Transactions on*, 18:279 – 293, 04 2010.

[25] Anthony T Patera. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics*, 54(3):468–488, 1984.

[26] V. Avsarkisov, S. Hoyas, M. Oberlack, and J. P. García-Galache. Turbulent plane couette flow at moderately high reynolds number. *Journal of Fluid Mechanics*, 751:R1, 2014.

[27] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[28] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[30] H. van Hasselt, A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-learning". *arXiv:1509.06461*, 2015.

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[32] Miles L. Loyd. Crosswind kite power. *Journal of Energy*, pages 106–111, 1980.

[33] Michael D. Breed and Janice Moore. Chapter 8 - movement: Search, navigation, migration, and dispersal. In Michael D. Breed and Janice Moore, editors, *Animal Behavior*, pages 219–252. Academic Press, San Diego, 2012.

[34] Matjaž Gregorič, Ingi Agnarsson, Todd A. Blackledge, and Matjaž Kuntner. How did the spider cross the river? behavioral adaptations for river-bridging webs in caerostris darwini (araneae: Araneidae). *PLOS ONE*, 6(10):1–6, 10 2011.

[35] Joh R. Henschel, Jutta Schneider, and Yael D. Lubin. Dispersal mechanisms of stegodyphus (eresidae): Do they balloon? *The Journal of Arachnology*, 23(3):202–204, 1995.

[36] Moonsung Cho, Peter Neubauer, Christoph Fahrenson, and Ingo Rechenberg. An observational study of ballooning in large spiders: Nanoscale multifibers enable large spiders' soaring flight. *PLOS Biology*, 16(6):1–27, 06 2018.

[37] Dries Bonte, Isra Deblauwe, and J. P. Maelfait. Environmental and genetic background of tiptoe-initiating behaviour in the dwarfspider erigone atra. *Animal Behaviour*, 66:169–174, 2003.

[38] Jan O. Washburn and Libe Washburn. Active aerial dispersal of minute wingless arthropods: Exploitation of boundary-layer velocity gradients. *Science*, 223(4640):1088–1089, 1984.

[39] Robert B. Suter. An aerial lottery: The physics of ballooning in a chaotic atmosphere. *The Journal of Arachnology*, 27(1):281–293, 1999.

[40] P. A. Glick. The distribution of insects, spiders, and mites in the air. Technical Bulletins 168268, United States Department of Agriculture, Economic Research Service, 1939.

[41] Jutta Schneider and Roos Schneider. Dispersal of stegodyphus dumicola (araneae: Eresidae): They do balloon after all! *Journal of Arachnology*, 29:114–116, 04 2001.

[42] Eric Duffey. Aerial dispersal in a known spider population. *Journal of Animal Ecology*, 25(1):85–111, 1956.

[43] M H Greenstone. Meteorological determinants of spider ballooning: the roles of thermals vs. the vertical windspeed gradient in becoming airborne. *Oecologia*, 84(2):164–168, September 1990.

[44] Peter Gorham. Ballooning spiders: The case for electrostatic flight. 09 2013.

[45] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[46] James R. Bell, David A. Bohan, Richard Le Fevre, and Gabriel S. Weyman. CAN SIMPLE EXPERIMENTAL ELECTRONICS SIMULATE THE DISPERSAL PHASE OF SPIDER BALLOONERS? *The Journal of Arachnology*, 33(2):523 – 532, 2005.

[47] J.A.C. Humphrey. Fluid mechanic constraints on spider. *Oecologia*, 73:469–477, 1987.

[48] Robert B. Suter. Ballooning in spiders: results of wind tunnel experiments. *Ethology Ecology & Evolution*, 3(1):13–25, 1991.

[49] A M Reynolds, D A Bohan, and J R Bell. Ballooning dispersal in arthropod taxa with convergent behaviours: dynamic properties of ballooning silk in turbulent flows. *Biol. Lett.*, 2(3):371–373, September 2006.

[50] Andy Reynolds, David Bohan, and James Bell. Ballooning dispersal in arthropod taxa: Conditions at take-off. *Biology letters*, 3:237–40, 07 2007.

[51] Kimberly Sheldon, Longhua Zhao, Angela Chuang, Iordanka Panayotova, Laura Miller, and Lydia Bourouiba. *Revisiting the Physics of Spider Ballooning*, pages 163–178. 08 2017.

[52] Longhua Zhao, Iordanka Panayotova, Angela Chuang, Kimberly Sheldon, Lydia Bourouiba, and Laura Miller. *Flying Spiders: Simulating and Modeling the Dynamics of Ballooning*, pages 179–210. 08 2017.

[53] O. Kratky and G. Porod. Röntgenuntersuchung gelöster fadenmoleküle. *Recueil des Travaux Chimiques des Pays-Bas*, 68(12):1106–1122, 1949.

[54] Laura Nenzi, Simone Silvetti, Ezio Bartocci, and Luca Bortolussi. A robust genetic algorithm for learning temporal specifications from data. 05 2018.