SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Physics Department
PhD in Physics

Thesis For The Degree Of Doctor Of Philosophy

Machine learning applications to dynamical and multi-agent systems

Relatori
Prof. Angelo Vulpiani
Dr. Massimo Cencini

Francesco Borra
1841917
matricola

Academic Year MMXIII-MMXXI (XXXIV cycle)

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Angelo Vulpiani for welcoming me in the TNT group, for supporting me and for giving me the opportunity to learn a lot and reshaping the way I see physics. My deepest gratitude goes to my co-supervisor, Dr. Massimo Cencini, who has guided me in this journey: not only for all the scientific discussions and for everything he has taught me, but also for always being supportive, kind, patient, for providing so many invaluable recommendations, and for educating me in the soft skills needed to navigate the scientific world with his constructive criticisms and inputs.

I would also like to thank Prof. Luca Biferale for giving me the possibility to work with him, for all the support and for all the opportunities he has provided me. I would like to thank Prof. Antonio Celani for introducing me to multi-agent systems and for allowing me to greatly expand the scope of my knowledge and interests.

I would like to thank Marco Baldovin, with whom I had the pleasure to work with, also for all the interesting scientific conversations and pieces of advice. I would also like to thank all members of the TNT group, especially Andrea Plati and Lorenzo Caprini, for the time spent together. I would also like to thank Alessandro Sozza and Chiara Calascibetta with whom I have worked during the last part of my Phd. I would also like to thank the GPU-AI CINECA group of Rome both for their support and for the availability of high performance computing resources. I would like to thank Fabio Bonaccorso for the support with GPU machine learning computations. I would like to thank Prof. Andrea Cavagna for the inspiring and great lectures in statistical mechanics. In addition, I would like to honour the memory of Prof. Bruno Bassetti, whose contagious enthusiasm and great teaching skills originally inspired me to work in this field.

I would like to thank my friends, who have been so important during my PhD journey and the overlapping pandemic, and, in particular: Giulio, with whom I have shared so much of my adventure in physics, in happy and sad times alike; Silvia, with whom I shared my experience as a PhD student in Rome; and Matteo, who has always been so close, even from across the ocean.

Finally, I would like to thank my parents, Flores and Massimo, for believing in me and supporting me in this journey throughout the years, in happy and difficult moments, from the first time I expressed interest in science to the end of my Physics PhD.

Francesco Borra, Rome, october 2021

Contents

Acronyms

ADAM ADAptive Moment estimation

ESRNN Echo State Recurrent Neural Network

FFNN Feed-Forward Neural Network

FSLE Finite Size Lyapunov Exponent

LSTM Long-Short Term Memory

MDP Markov Decision Process

OC Optimal Control

POMDP Partially Observable Markov Decision Process

RC Reservoir Computing

RL Reinforcement Learning

RNN Recurrent Neural Network

Introduction

In the last two decades, machine learning has transitioned from being just an ordinary field of research amongst many to being a fundamental topic with universal applications in science – from physics to medicine – which managed to cross the boundaries of the academic world, reaching industry, everyday life and even geopolitics or popular culture. In this thesis, on the other hand, we will mainly focus on machine learning as a modelling tool for dynamical problems.

Despite early academic enthusiasm around machine learning in the last century, the first breakthroughs in machine learning application took place outside the standard scope of hard sciences, and were relative to language and image processing and classification. For instance, a famous early machine learning accomplishment concerns the archetypal task of classifying – sloppily – handwritten digits, which was pioneered by LeCun in 1989 [148] and became mainstream with the so called MNIST dataset [149], assembled in 1998 and still employed today, as a benchmark, either in its original or extended versions [65]. Over the years, machine learning techniques have been repeatedly shown to be capable of classifying such items with near-human accuracy. This successful story has been replicated in increasingly complex scenarios with growing efficiency and sophistication, such as in image-classification, a field where stunning results have been achieved with famous architectures such as those from Google's Inception project [235]. It might not have been obvious, at first, that similar techniques could be easily transferred to quantitative scientific research. However, despite some initial skepticism from some communities, machine learning made its way into hard science and it is now widely applied, for instance in robotics [10], genetics [152], medicine [105], and physics [46]. Just to name a few physical cases, it is employed in high energy physics (e.g. [5, 265]) where it is commonly used for event identification of other inferential problems; in biophysics (e.g. [59, 246]), where it allows to study and predict complex structures such as proteins (e.g. Google's AlphaFold [127]).

Machine learning is actually a vague umbrella word. It is used to describe linear regressions as much as decision trees or neural networks, and it applies to virtually any algorithm which allows to learn to solve a task from examples, on statistical basis. Neural networks probably represent, so far, the pinnacle of success of machine learning. The core idea, which has been around since at least early works from the middle of the past century [113, 219], is to exploit networks of nodes, inspired from brain neurons, in order to build versatile computing systems. The whole learning procedure consists in tuning inter-neuron connections in such a way that the whole network can compute the solution of a desired problem.

In traditional computational approaches, a human operator designs an algorithm which solves a certain task by building the whole logical structure, addressing any conditional decisions, sub-tasks or exceptions. It is no wonder that this framework, while ideal for certain delicate tasks, immediately appears to be problematic for others. The aforementioned problem of classifying images is the perfect example: how does one even defines procedurally what a "3" is? Where does a "3" cease to be a "3" if I change it pixel by pixel? On the other hand, it is quite obvious for a human that a "3" is a "3" when they see one. Machine learning (neural networks specifically) is the toolbox which allows to approach this simple-yet-difficult problems in a way that is similar to how humans do. We learn to tell a "3" by *generalizing* the idea of "3" from a finite number of examples. Likewise, an artificial network can learn to recognize a never-seen-before "3" image after being exposed to a number of examples, without direct human intervention[1] in the feature extraction process; only an appropriate

[1] In spite of some narratives, neural networks and machine learning techniques still have to be tailored – occa-

learning-algorithm and an good dataset are needed. The possibility to design algorithms to *learn from examples* is one of the features machine learning is named after.

In this sense, neural networks are designed to mimic human learning and intuition. Note that, despite striking and inspiring similarities, neural networks do not necessarily learn in the same way as human brains do. Indeed, artificial neural networks can solve certain tasks (e.g. image classification) with more-than-human accuracy and even simulate creativity but they are not equivalent to biological brains. At a structural level, this is true for the simple reason that artificial neurons are not generally designed to operate in the same way biological ones do. For instance, we can mention that some networks may be deceived with relatively simple tricks, called adversarial attacks [79, 82]: by selectively modifying certain pixels – in ways a human would never be fooled – one can cause a network to catastrophically fail its task [231].

Learning from examples is not the only possible learning framework: a related but different idea is to learn a task from *trials and errors*, which is the domain of a branch of machine learning called *reinforcement learning* [233]. The word "reinforcement" refers to the possibility to design algorithms which select optimal behaviours in order to achieve a certain goal by designing an appropriate reward/penalty system: behaviours which achieve the best outcomes are rewarded the most and, therefore, encouraged (reinforced) trial after trial. In this sense, as much as neural networks are inspired by biological neural systems, reinforcement learning is inspired by animal learning. While it is not strictly necessary, in order to encode such strategies, neural networks may be employed as powerful function approximators. Reinforcement learning is especially suitable for dynamic tasks, in which one has to account for long term consequences of an action, and may be used for several purposes, such as playing games, such as chess, or GO (it is worth mentioning the celebrated algorithm AlphaGO); building learning robots; designing algorithm for automatically controlling aircrafts or gliders [214]; finding the shortest path towards a target [29] and so on.

One of the reasons why machine learning has emerged after so many decades since it was proposed is not that theoretical breakthroughs were lacking but, rather, that two key ingredients were missing: computational power and data. Indeed, powerful CPUs, RAMs, large amounts of data and computational times are necessary for these kind of tools to make the difference. Our world is entering what some people call *Big Data* revolution: the possibility to story massive amount of data combined with widely available and unprecedented computational power is reshaping the way we think science. In this context, some prophet of the new age of Big Data paradigm have theorized that, thanks to unstoppable progresses of machine learning, the scientific method we are familiar with and the whole idea of building models and theories will soon become obsolete [7], as powerful artificial intelligence will be able to build a new kind of science grounded on empirical correlations alone. While this idea appears naive for several reasons, including the possibility that machines themselves would try to build theories to understand reality, it is certainly possible that, in some future, machines will outsmart humans to the point of replacing us as scientists for good. However, irrespectively of whether one would hail or grieve this moment, it is not this day. On the contrary, machine learning is opening new scenarios in physics – and science in general – and the topic of this thesis revolves around the application of machine learning to dynamical and biologically-inspired

sionally fine-tuned – to the problem one is trying to solve. A long term goal may be to overcome this issue, with general-purpose, self-setting techniques. While this may sound like an utopian dream, some exceptionally versatile techniques may become reality in the foreseeable future.

multi-agent systems.

In physics, machine learning is a topic of interest for several reasons. For instance, there is an ongoing theoretical investigation (with ideas from statistical physics, dynamical systems, information theory etc.) which aims at understanding why machine learning works so well, and at exploiting such knowledge in enhancing existing methods or designing new ones [181]. In this thesis, on the other hand, we neither addressed the theoretical foundations of this discipline, nor we aimed at testing machine learning techniques in extreme conditions but, rather, we tried to apply them to interesting and controlled problems where we could focus more on the physics and its modelling, rather than on the technique itself. Accordingly, we will mainly keep the focus on the physical problems under exams rather than the tools themselves.

This thesis is divided in two parts. The first part of this thesis is dedicated to machine learning applications to multiscale and chaotic dynamical systems [51, 186]. Such systems have been subject to extensive investigation both for their theoretical relevance and for their practical importance. Many relevant physical phenomena display both a chaotic behaviour and a multiscale structure: for instance in climate, weather, turbulence, astrophysics or geophysics [51, 86, 261]. Informally, a system is chaotic when very small errors on the initial conditions expand exponentially in time. This feature alone is challenging for the sake of modelling, since, when attempting long term forecasts, any error made at any given time will be amplified exponentially in time thus affecting the quality of subsequent predictions. A multiscale structure may make the modelling task even harder. It is very common for physical systems, to feature variables charachterized by different scales, either spatially or temporally. The span of such scales may be extreme, such as in climate [8, 140]: the same system, our planet, displays hourly or daily weather variations as much as a very slow, but otherwise much more dramatic, switching between glacial and inter-glacial periods over tens of thousands of years. Several climate-related phenomena have intermediate or even larger scales, such as vegetation growth (decades-centuries), carbon cycle (decades to million years), oceans (decades-millennia) and so on. While both fast and slow phenomena may be interesting, in this thesis we are mainly interested in the slower ones, which generally are associated with larger spatial scales or variables with greater magnitude. For this reason, they may be referred as macroscopic phenomena.

When trying to describe or model macroscopic phenomena, one should choose how to address the presence of smaller/faster variables, the microscopic scales. A first possibility may be to model a system of interest completely with both fast and slow variables. However, it is clear that this approach is problematic for several reasons: microscopic variables may be so many that simulations are too slow, computationally expensive or outright unfeasible[2]. Moreover, in this scenario, all microscopic interactions should be modelled correctly – or, at least, correctly enough – since, due to the typical nonlinear character of such systems, modelling errors might spoil the description of macroscopic variables we want to study. On the contrary, it is indeed well understood by now that, in several scenarios, the best way to study macroscopic variables is to account for fast variables through an effective description, a *coarse graining*. Building effective equations may be a delicate task and, so far, there are no systematic approaches [20, 21]. In one of the most favorable scenarios, the scale separation between fast and slow scale is so large that one can apply asymptotic methods

[2] A model cannot both account for atmospheric turbulence lasting from seconds to a hours and, say, glacial and inter-glacial events.

(such as multiple scale expansions [195, 223]), but this is not true in general. Finally, we should point out that it may not even be clear what relevant slow degrees of freedom one should include in the effective equations. In this context, machine learning provides a resourceful toolbox for tackling these problems or improving traditional techniques; in particular, it allows to build effective equations without significant prior knowledge about the problem under exam, the so called "model free" framework. In a nutshell, this method relies on the possibility to use large amount of data to fit desired functions (in our case, mainly the equations of motion) in a very efficient way, by exploiting classes of functions which behaves as universal approximators. Such techniques are already known to work very well but a physicist's work may be to critically study their performances in physically relevant and controlled scenarios; in particular, in the first work "Effective models and predictability of chaotic multiscale systems via machine learning" [39] (in collaboration with Massimo Cencnini and Angelo Vulpiani) presented in this thesis, we have focused on equation reconstruction in a chaotic multiscale system, using an echo state neural networks as a tool.

The physical model we have studied is the so called two-scale Lorenz system [34, 51], which is obtained by coupling of two chaotic subsystems with different intrinsic timescales. A noteworthy feature of this model is that the scale separation can be tuned, allowing to explore different scenarios. For the purpose of this work, we have used a kind of recurrent neural networks called *echo state neural network* or *reservoir computer* [120, 121, 161]. The most remarkable characteristic of echo state networks is that most neural connections are intialized randomly and never modified, while only a very small amount of links (the *output* or *readout layer*) are trained. As a result, the training procedure is much faster and straightforward (a simple linear regression) than the usual backpropagation-based one, which, in general, is used to iteratively modify all connections. Such networks have gained attention in the dynamical systems community since they are interesting dynamical systems themselves and, most importantly, they have been shown to be capable of modelling chaotic systems with great efficiency [159, 191], allowing both for short/long term forecasts and for the reproduction of statistical properties – also called *climate* [192]. Specifically, we have used this tool to build effective equations for the slow part of the two-scale Lorenz system and compared its predictive performance with that of other effective models constructed with simpler techniques. We found that, in the slow dynamical regime (when the error expansion is driven by the slow subsystem), in all cases we studied, the dynamical reconstruction provided by the network efficiently captures the slow regime. Moreover, by performing a sort of reverse engineering, we could conclude that, when the scale separation is strong, the networks essentially build an adiabatic model, but it still works when scale separation is weak and such model fails. Finally, we focused on non-model free extension of the reservoir approach, the so-called *hybrid scheme* [194], which consists in "assisting" the network predictions with some forecasts provided by an imperfect model. We could show that this approach allows to achieve the same performance as in the standard scheme but with a smaller network and with less performance fluctuations. However, in both cases, the performance plateaus at a certain network size.

In a second work [36], in collaboration with Marco Baldovin, we approached a similar problem, but from a different perspective. Instead of working with a system with hard-wired scale separation, we have focused on a system of *globally coupled maps* [132, 133]: a collection of non-linearly interacting nodes, which are known to display a non-trivial macroscopic – i.e. of the mean-field – dynamics; such collective dynamics displays, for appropriate parameter choices, a multiscale and chaotic be-

haviour [53]. Therefore, such multiscale structure is an emergent property and cannot be deduced in some obvious way from the equations. Hence, these kind of systems are valuable because, in spite of their simplicity, they present some complex macroscopic phenomenology [53, 128, 133] whose features are reminiscent of those of real high dimensional systems such as groups of neurons, biological agents or turbulence. In this work, we studied the possibility of modelling such emergent collective dynamics. We focused on the setting studied in ref. [53], in which the system displays chaos at macroscopic level. As a first attempt, in order to probe the hardness of the problem, we tried two non-machine learning approaches aimed at building a second order effective dynamics. Since both failed to provide a convincing reconstruction, we could conclude that the macroscopic dynamics was complex enough to be suitable for a machine learning approach. Therefore, in this work, we used a feed-forward neural network in order to build effective equations from data. We did not just tried to build a model for the sake of making predictions, but we also attempted to understand whether the effective model obtained via machine learning could help us gaining some insight into the underlying physics of the system. The answer is positive: by performing a careful analysis and modulating the information available to the network, we could provide a rather detailed characterization of the macroscopic dynamics of our coupled maps system. The procedure consists in building both a stochastic and a deterministic model for the delay vector of the macroscopic dynamics. We showed that the stochastic modelling, by accounting for the unresolved degrees of freedom, could provide an excellent reproduction of the system macro-dynamics; the deterministic model, on the other hand, had an acceptable performance only for very large number of maps. Thanks to our machine learning analysis, we could claim that the systems displays long memory effects, which explain why a second-order description was doomed to fail. Moreover, we could extract information about the existence of a rich multiscale structure, with different possible level of coarse graining. Furthermore, we backed our machine learning-based exploration with some standard analysis, such as the Grassberger and Procaccia procedure for computing the correlation dimension of the macroscopic attractor. Hence, we explored the deep connection between modelling and theoretical investigation: while not being a substitute for standard and rigorous techniques, machine learning modelling can be a frontline investigative tool for understanding dynamical systems.

The second part of this thesis deals with modelling a peculiar kind of physical systems, which are biologically inspired multi-agents systems, where "agents" should be thought of as some living organisms, from bacteria to birds or even humans. In some sense, such physical systems are akin to conventional dynamical systems, but with a major difference: while standard physical systems evolve according to a set or rules or given interactions, it is very natural to assume that biological agents – say through deliberate actions in the case of superior organisms or through evolutionary hardwired instincts – behave in a way that benefits them in some sense. Mathematically, this means that, in many cases, agents try to "optimize" their behaviours according to some metrics. The previous argument makes it clear that there is a conceptual connection between biology – self interested agents – and robotics – artificial agents engineered to achieve a certain goal. Note that, in multi-agent settings, the optimal behaviour adopted by individual agents may be non obvious even when starting from simple premises, since the "best" strategy of an individual depends on those of other self-interested agents. Therefore, such problems are naturally connected to optimization formalisms such as optimal control theory [26], game theory [185] and reinforcement learning [233].

The first work, done in collaboration with Massimo Cencini, Luca Biferale e Antonio Celani, pre-

sented in this second part of the thesis deals precisely with an application of reinforcement learning to the two-agent interaction between a prey and a predator in an hydrodynamic environment. The core idea is to frame this problem as a zero-sum adversarial game where a first agent – a predator or pursuer – should try to capture its opponent in the shortest possible time, while the second one should try to avoid encounter as long as it can; this pursuit-evasion problem is loosely inspired to ref. [16], where two competing teams of virtual agents learned to play a hide-and-seek game using various tools from a simulated the environment. In our setting, there are no tools, but agents are immersed in a two-dimensional low-Reynolds number hydrodynamic environment, which idealizes the conditions of small organisms – called "micro-swimmers" in literature [77] – living in non-turbulent or still waters. Therefore, the motion of one agent generates a disturbance in the medium which affects the other one. Moreover, we assumed the agents are blind and do not directly perceive their respective positions or direction of motion, but can only extract clues from the perturbations of the fluid. This is a realistic assumption for organisms either with poor/absent eyesight or living in dark or murky environments; many aquatic organisms do possess senses which are alternative or complementary to eyesight. For instance, fishes generally display the so called lateral lines [32]: arrays of hydrodynamic sensors located on the side of their bodies. On the other hand, smaller animals such as arthropods may perceive the disturbances of the fluid through antennae or setae [139]. In our idealized modelling, just like in the real world, it is not an easy task to infer the position of the other agent through hydrodynamic signals alone, a feature called "partial information" in the machine learning community. We have employed a reinforcement learning algorithm to let agents learn the appropriate behaviours (called *policies*). The policies that emerge from the simulations are visually appealing but, most importantly, we have been able to explain the rationale behind the most relevant of them – both in the case of the predator and the prey – and even to provide an analytical description the observed behaviours. While the specific policies likely depend on the specific setting, our understanding of the policies allows us to conjecture that the core elements of such strategies are likely preserved in more general scenarios. For instance, we observed that the predator, while incapable of directly locating the prey (due to the symmetries of the hydrodynamic signals), behaves in such a way to reduce the dimension of the coupled two-agents dynamics, greatly increasing the chance of a random encounter. On the other hand, the prey exploits the hydrodynamics as a defence in two complementary ways: by taking advantage of direct repulsive flows at short range and, on the other hand, by making good use of the ambiguity of the information and turning the predator's attack strategy against itself, when the two agents are far from each others.

The last work presented in this thesis, done with Massimo Cencini and Antonio Celani, does not deal with machine learning in a strict sense, but it addresses the same problem of optimal behaviour in multi-agent systems. While in the previous work the interaction between the two agents was purely adversarial, here we have considered a fully cooperative problem of optimal collective motion of an idealized swarm of agents. Such problems are usually investigated with statistical mechanics either by reverse engineering biological interactions between individuals or by capturing the essence of them in idealized schemes as in Vicsek [255, 256] or Kuramoto [144] models. However, it is not obvious that such models are "optimal" in some sense: to introduce a notion of optimality in this framework would be both relevant from a theoretical point of view, and in biological or robotic applications. In our work, we adopted optimal control formalism for this purpose – specifically, Todorov's formulation [243] – and we chose to idealize the swarm as a system of active Brownian

particles [218] which are trying to avoid colliding with each others by controlling their heading directions with a torque; particles are also subject to rotational noise. In order to define a cost function, we introduce a cost paid by particles for each collision. Since the goal is to minimize the total cost, with these premises, the solution would be rather trivial, as the particles might apply infinite control to avoid any collisions. However, in realistic scenarios, infinite control is meaningless, since control itself has a cost, which can be interpreted for instance as energy consumption or as a cognitive effort. In either cases, there exists a tradeoff between avoiding collisions and reducing control costs. The choice of a quadratic control cost allows to map the optimal control problem into an egeinstate equation for a quantum many-body system. In order to proceed analytically, we introduced two mean-fields assumptions – spatial homogeneity and agent-wise factorization – which physically mean that we are focusing on a large uniform region within the bulk of the swarm. As a consequence, we were able to find an explicit solution for the problem and characterize a mean-field second-order phase transition in the polarization order parameter (the average direction of the swarm); the transition depends on the relative importance between collision and control costs. Through a careful analysis, we were able to show that, for any choices of the problem parameters, it is always possible to construct a Vicsek inspired model [60, 61, 201] which is nearly optimal (and truly optimal in a certain limit) by any reasonable metrics. Hence, this work both shows how optimal control may be successfully applied to collective behaviour problems and suggests that simple existing models might be employed in conditions close to optimality.

As already mentioned, the thesis is divided into two parts, each containing three chapters. Each part has an introduction, which outlines the relevant methodology and two chapters, each describing the main results obtained in the works published or submitted during the PhD.[3].

[3] An additional work, concerning a machine learning application to intermittency in the shell model (a simplified model for turbulence) could have been added to the first part of thesis, but was not inserted since it was still under preparation by the submission deadline.

Part I

Machine learning and multiscale chaotic systems

Chapter 1

Machine learning techniques for dynamical systems: key ideas and techniques

The aim of this chapter is to provide a framework for machine learning applications to dynamical systems. In principle, we might focus both on machine learning and on the basics of the theory of dynamical systems. Since the latter is part of the background of physicists, in this chapter, we focus on machine learning alone and we refer to the literature (see for instance [51, 186]) for a systematic introduction to dynamical systems. However, key theoretical ideas will be introduced whenever they are needed.

1.1 The problem of dynamical reconstruction

The landscape of machine learning techniques is vast and rapidly expanding and it would be both unfeasible and pointless to provide a complete introduction. Instead, we will offer a self-contained overview of the general problem of machine learning for modelling and forecasting dynamical variables within the scope of our concern, and then we will proceed by outlining some techniques which will be relevant to the original results presented in this thesis.

In the real world, virtually any complex dynamical signals that one may want to describe – weather, motion of planets, polls dynamics, monetary inflation or the number of people infected with SARS-COV2 – comprise just some of the many interacting degrees of freedom of a larger environment. Clearly, the existence of such an environment cannot be neglected, but can otherwise be very difficult to be accounted for. The degrees of freedom may be too many for a complete description and some of them may not even be observable: for instance, an astrophysicist who wants to forecast the solar activity cannot rely on direct observations of the convective currents inside the Sun, which would otherwise provide useful information. In other cases, it is not even obvious what the relevant variables are.

While, traditionally, much of the work of physicists consists in understanding what the correct variables are and how to infer the unobservable ones [19, 21, 259], machine learning provides both the tools for selecting relevant variables and for making prediction from data even without much physical insight – the so called *model-free* approach.

Formally, we can assume that our time-dependent signal $x_t \in \mathbb{R}^k$ is some observable function $M : \mathbb{R}^n \to \mathbb{R}^k$ of the state of the environment $s_t \in \mathbb{R}^n$, so that $x_t = M(s_t)$. Since, in general, M is not invertible (typically $n > k$), the state s_t cannot be inferred exactly from a single measurement of x_t. We can either assume that s_t evolves in continuous or discrete time but, for presentation simplicity, we will limit the discussion to the discrete formulation, without loss of generality. Ideally, the unobservable environment, may be assumed to be subject to a memoryless evolution, since it contains all degrees of freedom by construction. It evolves, either stochastically or deterministically, with some law $S : A_s \to A_s$

$$s_{t+1} = S(s_t), \tag{1.1}$$

where $A_s \subset \mathbb{R}^n$ is an $S-$invariant and bounded manifold. Let us now assume we have a sequence of data $x_{t:t+T} = \{x_t, ..., x_{t+T}\}$: the core idea is that the dynamics of x_t, including the contribution from unobservable variables s_t, can be inferred from it. In a seminal work, Takens proved [51, 116, 182, 237] that, under some hypotheses[1], for deterministic chaotic systems, the *embedding vector* or *delay vector* $x_{t-m+1:t}$ constitutes an embedding of the invariant manifold A_s of s_t into $\mathbb{R}^m$, provided that $m > 2\, d_{bc}$, where d_{bc} is the box counting dimension [51] of A_s. The key point is that, in order to describe the dynamics of variable x_t we do not need to reconstruct the whole universe s_t, but we can just focus on the history of the variable x_t itself.

In this sense, we can look for a non-Markovian description of the variable x_t. Namely, in the prediction problem, we can look for some function F, either stochastic or deterministic, such that

$$x_t = F(x_{t-T:t-1}), \tag{1.2}$$

with T possibly being $T = -\infty$ for some pathological cases. While standard techniques struggle with the task of reconstructing such functions F for large delay T, machine learning tools have proven to be very efficient in this regard, as long as the problem is not too hard. The rest of the chapter will be dedicated to the description of some relevant machine learning techniques for the problem of dynamical reconstruction.

1.2 Feed-forward neural networks

1.2.1 Statistical learning

Following the ideas from the previous section, the simplest way to forecast a dynamical variable x_t is to construct some function F, either stochastic[2] or determinisitc, such that the map

$$\hat{x}_{t+1} = F(x_{t-n:t}), \tag{1.3}$$

yields a prediction $\hat{x}_{t+t}$ for variable x at time $t+1$ from a piece of history $x_{t-n:t}$ of x of length n. If the whole environment is fully observable ($x_t = s_t$) or if we can assume that x_t is Markovian, it is possible to simply use $n = 1$. Identifying an appropriate function F is the nontrivial part. One should first choose a class of test functions $\mathcal{F}$ and a metric or loss function L. Given a set of

[1] Besides usual smoothness assumptions on the dynamics S, the measurement function $M : s \to x$ should be twice differentiable with full rank, and it must be a typical function, e.g. it must not be a constant function.

[2] Strictly speaking, this means that $\hat{x}_t$ is a random variable distributed with probability density function $F(x_{t-n:t})$.

data Ω, the loss function $L(F, \Omega)$ describes how well a given test function F performs over such a set. In our case, the performance can be understood as the accuracy of a function F in guessing the output $Y(= x_{t+1})$ given an input $X(= x_{t-n:t})$. A set of data can therefore be interpreted as a collection of M input-output pairs $\Omega = \{(X_j, Y_j)\}_{j=1}^{M}$ and a natural loss may be defined as the mismatch between the prediction and the actual value: for instance the mean squared error

$$L(F, \Omega) = \frac{1}{M} \sum_{j=1}^{M} \|F(X_j) - Y_j\|^2. \tag{1.4}$$

Since we are defining a "correct" labelling[3] system, we are dealing with a case of so-called *supervised learning* [168], which is far from being the most general scenario. In the case of image classification, for instance, X would correspond to images themselves and Y to their captions; in the MNIST case, e.g. in the MNIST[4], black-and-white handwritten digits and integers between 0 and 9, respectively.

Training corresponds to finding the function which performs the best in fitting our unknown target function F. Therefore, given a training set Ω_{train}, one can define the best function as

$$\hat{F} = \arg\min_{F \in \mathcal{F}} L(F, \Omega_{train}). \tag{1.5}$$

The function $\hat{F}$ does not necessarily represent the best candidate as a solution to the problem. Indeed, we have only optimized its performance over the training set $L_{train} = L(\hat{F}, \Omega_{train})$, but what we really want is to make predictions about never-seen-before data. The ability of a machine learning technique to correctly handle new data – learning from examples – is called *generalization* [181] and it is the ultimate measure of success of machine learning, which elevates it beyond some simple fitting technique. For this reason, what one really wants to minimize is the *generalization error* (statistical theory of learning) or *evaluation loss* (machine learning community), namely

$$L_{gen} = \mathbb{E}[L(F, \cdot)], \tag{1.6}$$

which is the expected loss w.r.t. to the "true" distribution of data. In practice, one can employ a *validation or test set* Ω_{test}, statistically independent of Ω_{train}, and compute the following estimator for the generalization error:

$$\hat{L}_{gen} = L(F, \Omega_{test})]. \tag{1.7}$$

While it is generally safe to assume that a larger training set lowers L_{gen}, an important role is played by the class of functions $\mathcal{F}$: a class usually contains functions which have the same functional shape but different parameters. A small number of parameters makes the training from data easier for obvious reasons but limits the *expressivity* of the neural network, i.e. the ability to fit complicated functions. Conversely, in principle, a large number of parameters may allows for a finer fitting – more expressivity[5] – but could make training more difficult: more data are needed and, most importantly, large amounts of parameters may be overfitted over training data and thus generalize poorly. The bottom line is that both too many or too few parameters may be detrimental. This qualitative argument can be approached rigorously within statistical theory of learning framework, for instance

[3] A label is another name for the output variable in supervised learning.
[4] For information on MNIST database see [65, 149]
[5] Warning: one should not assume that, in general, expressivity and the number of parameters are equivalent.

with Vapnik-Cervonenkis' theory [251, 252], or by studying the bias-variance tradeoff [91]. The latter[6] represents a possible tradeoff between accurate average estimation (bias) and the sensitivity to data (variance). It should be noted that not all such mathematical results (especially worst-case-scenario analyses) are directly relevant to real-world applications, where the structure of data is important [220] and very large network do not necessarily show increased variance [180]. In practice, in spite of some theoretical pessimistic arguments for fine tuning of the number of parameters, very large number of parameters are often used successfully and, in order to avoid overfitting, an number of techniques, called *regularizations*, have been proposed. It is worth mentioning a few, such as dropout [18], batch normalization [117], algorithms targeting large flat minima [17, 57] and many others. In order to provide an illustrative example, one could regularize the loss (1.4) by introducing a Bayesian prior p over the space $\mathcal{F}$ and update the loss as

$$L(F, \Omega) = \sum_{j=1}^{M} \|F(X_j) - Y_j\|^2 - \beta \log p(F), \tag{1.8}$$

in order to penalize "strange" solutions (small $p(F)$) unless data really point in that direction. Indeed, equation (1.8) is proportional to (minus) the log of the posterior probability $P(F|\Omega) = P(\Omega|F)\, p(F)/P(\Omega)$ if $F(X) - Y$ are assumed to be Gaussian distributed.

1.2.2 Feed-forward neural networks: a classic

Main ideas

One of the most relevant classes of approximating functions are, of course, neural networks [73, 109, 168, 181]. The simplest architecture is the so called Feed-Forward Neural Network (FFNN). Since such networks are a widely known, we will just provide a minimal overview. In general, a FFNN is a map

$$F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_N} \tag{1.9}$$

where n_0 is the dimension of the input and n_N is the dimension of the output. The network can be written as the composition of single-layer functions $F^{(i)}$:

$$F = F^{(N)} \circ F^{(N-1)} \circ ... \circ F^{(1)} \tag{1.10}$$

with

$$F^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i} \quad \forall i = 0, ..., N \tag{1.11}$$

and

$$F_a^{(i)}(\boldsymbol{x}) = \sigma_i \left(\sum_{b=1}^{n_{i-1}} w_{ab}^{(i)} x_b + \theta_a^{(i)} \right). \tag{1.12}$$

[6] Let us consider two random variables x and y and assume we want to estimate $P(y|x)$ with a model $M(\cdot; \Omega)$, estimated with a dataset Ω. Then, it can be shown that [91]

$$\mathbb{E}_\Omega[(M(x;\Omega) - \mathbb{E}_y[y|x])^2] = \underbrace{(M(x;\Omega) - \mathbb{E}_y[y|x])^2}_{\text{bias}} + \underbrace{\mathbb{E}_\Omega[(M(x;\Omega) - \mathbb{E}_\Omega[M(x;\Omega)])^2]}_{\text{variance}}.$$

There could be a tradeoff in minimizing these two terms.

The i^{th} layer, containing of n_i neurons, is described by the matrix-array tuple $\Theta^{(i)} = (w^{(i)}, \theta^{(i)})$ along with the activation function σ_i. Intuitively, the minimal unit of a neural network is a "neuron" or node. Each neuron possesses a number of incoming and outcoming connections (links/synapses). When the network is working, a scalar value (state x_a) is associated to a neuron a. The node "fires" its signal through the outcoming synapses to the neurons it is connected to. Each synapsis is described by a scalar value (weight). The signal from neuron a to neuron b is the product of the synaptical weight and the state of the firing neuron a: $x_a\, w_{ba}$. All signals coming into b are summed into one $\sum_{a \text{ connected to } b} w_{ba} x_a$ and shifted by a *bias* θ_b. The signal is then processed by a non-linear activation function σ so that the state of b is updated to $x_b = \sigma\left(\sum_a w_{ba}\, x_a - \theta_b\right)$. Several choices of activation function are possible, including tanh and the so called *rectified linear unit* (ReLU)

$$\mathrm{ReLU}(x) = \max(0, x), \tag{1.13}$$

with the latter being among popular ones when using deep architectures (i.e. with many layers) for reasons that will be mentioned later. Activation functions belonging to different layers need not to be the same. Also, layers need not to be fully connected and, on the contrary, networks with highest performances are often obtained with well-tailored structures, such as convolutional ones, which are suited for image processing. A special mention should go to the last layer. By choosing σ_N to be the identity, one can map a generic input $x \in \mathbb{R}^{n_0}$ into another generic $y = F(x) \in \mathbb{R}^N$ but more specific choices are possible. It is worth mentioning the softmax activation function

$$\sigma_j = \frac{1}{Z} \exp(H_j) \;\; \forall j = 1, ..., n_N \tag{1.14}$$

with $Z = \sum_{j=1}^{n_N} \exp(H_j)$ and $H_a = \sum_{a=1}^{n_{N-1}} w_{ab}^{(N)} F_b^{(N-1)} + \theta_a^{(N)}$. This specific choice ensures that, for any given input, elements of the output $\sigma(x)$ are non negative and sum to one for any input x. Hence $\sigma_i(x)$ can be interpreted as the probability of the i^{th} output neuron: this is one of the techniques used to make probabilitstic predictions. In the archetypical MNIST case, each neuron corresponds to a possible digit: the network receives an image as input and returns the probability that it represents a certain digit. This architecture will be used in chapter 3.

Gradient descent

As it is clear from the previous description, a given architecture – connectivity structure, number of neurons, activation functions – is compatible with different choices of parameters $\Theta = \{\Theta^{(i)}\}_{i=1}^N$. The parameters are chosen as those which minimize the loss function; such optimization can be achieved with one of the many techniques revolving around the idea of gradient descent. Namely, parameters $\Theta(k+1)$ at step $k+1$ are updated from step k as

$$\Theta(k+1) = \Theta(k) - \eta\, \nabla_\Theta L(F_{\Theta(k)}, \Omega_{train}) \tag{1.15}$$

where η is the learning rate. Algorithm (1.15) is a deterministic search and, therefore, may be trapped in local minima if the loss in not convex (as it is in the general case). The solution is usually to add some effective noise (*stochastic gradient descent*) in order to allow the dynamics to cross barriers and enhance exploration, a problem extensively studied from a theoretical physics point of view (e.g. in Refs. [171, 274]). This can be done in several ways: the most common is

to divide the dataset into batches, to compute the loss gradient with each batch and iteratively update parameters in this way; after the last batch has been used, one typically starts again from the first one, possibly after shuffling the data. Each complete set of batches is called *epoch*. While the expected direction of the batch gradient does not change, the randomness of individual batches generates some noise which could unstucks the dynamics. Another trick is to add momentum to the gradient descent, such as it is done in the celebrated ADAptive Moment estimation (ADAM) (which is not free of flaws and has its detractors [268]). Another issue, which is not necessarily specific to FFNN in gradient descent algorithms is the problem of *exploding or vanishing gradients*. This issue, which will also be relevant in recurrent neural networks, can be easily understood in the case of deep neural networks. The gradient of the loss with respect to parameters of a certain layer, say i, is given by the chain rule and may be written as

$$\nabla_{\Theta^{(i)}} L = \nabla_{\Theta_i} F^{(i)} \cdot ... \cdot \nabla_{F^{(N-1)}} F^{(N)} \cdot \nabla_{F^{(N)}} L. \tag{1.16}$$

which can be efficiently evaluated through a procedure called *back-propagation* [263]. As a matter of fact, the product of gradients can easily make the overall gradient update for deep layers to vanish exponentially, especially for certain choices of the activation function, such as the tanh.

Universal approximation theorems

One of the fundamental mathematical results concerning machine learning is that FFNNs are universal approximators: any regular function (in a sense which depends on the specific formulation of the problem) can be approximated with arbitrary accuracy by a FFNN with a single hidden layer [114, 224]. Clearly, the existence of a solution to the fitting problem does not imply that learning is possible, since one should make sure that it may be possible to reach it and that generalization properties are good. Amongst the many possible references, for a classic physical approach to the problem see [181], for a mathematical approach see for instance [124].

1.3 Recurrent neural network: from vanilla to LSTM

Recurrent Neural Network (RNN) are a kind of networks specifically designed to address time-dependent problems. The goal is to find a way to describe the evolution of a variable x_t as $x_{t+1} = F(x_{-\infty:t})$, as discussed before[7]. The procedure is akin to building of a hidden Markov chain: it is general knowledge that memory effects may derive from – and can in turn may be accounted for – the presence of unobserved degrees of freedom. Then, the core idea in modelling is to introduce an auxiliary variable r_t which evolves along with x_t, so that we have an augmented dynamical variable (x_t, r_t). The interactions between the two variables – physical and auxiliary – should be chosen in such a way that the dynamics[8] of x_t fits the data[9]. In this framework, the variable r_t ought to account for those (relevant) degrees of freedom belonging to the hidden state s_t that cannot be inferred from a single-time snapshot of the dynamics of variable x_t. See fig. 1.1 for a pictorial representation of these ideas.

[7] $x_{-\infty:t}$ simply denotes an infinite left sequence $\{x_s\}_{s=-\infty}^{t}$

[8] This means that we evolve the system (x_t, r_t) and we look at the subsystem x_t.

[9] Note that, with an abuse of notation, we are using the same letter x_t both for the the empirical time series and the dynamical variable which evolving with the recurrent neural network dynamics.

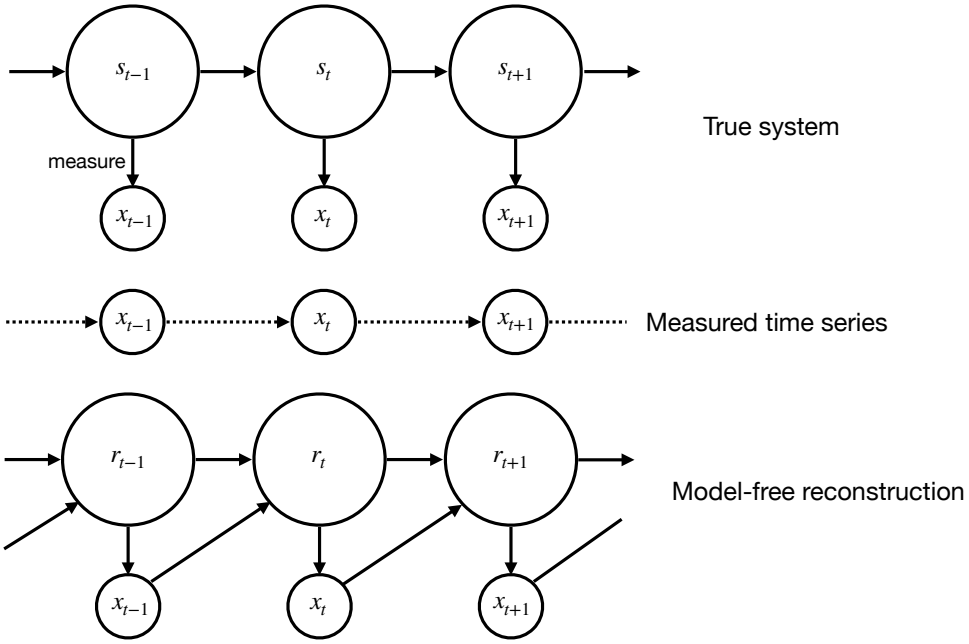

Figure 1.1: Main ideas of dynamical reconstruction with recurrent neural network of a signal x_t. Variable s_t evolves in time, without memory effects; however, only the time series of variable $x_t = M(s_t)$ may be measured. To compensate for unmeasured degrees of freedom, a solution consists in generating ancillary degrees of freedom r_t which co-evolve with x_t is such a way that the dynamics of x_t may reconstruct the original signal (denoted with the same letter).

In order to express the idea more formally, we say that variable r_t evolves according the following law

$$r_{t+1} = G(r_t, x_t). \tag{1.17}$$

Moreover, we can assume that r_t contains the most relevant information about past history $x_{-\infty:t}$ of the variable x_t itself. As much as we assume that x_t can be forecast by its past history, we can extract a prediction $\hat{x}_{t+1}$ of x_{t+1} from r_t: e.g. we write

$$\hat{x}_{t+1} = H(r_{t+1}); \tag{1.18}$$

whenever needed, we will use the hat $\hat{x}$ to denote forecasts of x, but notice that, with an abuse of language, we will drop this notation when it may become redundant. If we combine equations (1.17) and (1.18), we can write an equation for r_t alone

$$r_{r+1} = G(r_t, H(r_t)). \tag{1.19}$$

Thus, r_t is a dynamical variable describing an autonomous system which can run for arbitrarily long times, allowing for long term predictions of x_t. Therefore, the machine learning problem can be understood as the problem of finding two functions H and G and to fix an initial condition for the auxiliary variable r_t. The natural question is how one can do that in practice.

As a final point, it is worth mentioning that RNN have much broader applications than our narrow dynamical systems-oriented description (1.17) and (1.18). Indeed, the input x_t and the

output y_t need not to be the same:

$$\begin{cases} r_{t+1} = G(r_t, x_t) & (1.20) \\ y_t = H(r_t). & (1.21) \end{cases}$$

This scheme allows to generate any element of a sequence $y_{1:T+1}$ from a sequence $x_{0:T}$. The case $x_{0:T} \mapsto y_{T+1}$ is dubbed *many-to-one* case and may be used to compute physical observables from trajectories. In contexts different from the problem of predictions of dynamical systems, this scheme may be adopted for sentence classification or sentiment analysis, where a final label is assigned to a time series. Different choices are possible, as the *many-to-many* scenario $x_{0:T} \mapsto y_{1:T+1}$. This is the approach that will be described in the open-loop phase in the echo-state section. A last case is the *one-to-many*: one tries to forecast a sequence $y_{1:T}$ from a value x_0 (e.g. closed loop mode in echo-state neural networks, see sec. 1.4) or even just from r_1. A standard example is text generation from a seed word or the generation of a caption for an input image.

1.3.1 Recurrent neural networks: main schemes

Vanilla Recurrent neural networks

One of the simplest machine-learning ideas is to build a recurrent neural network, similarly to the feed-forward scheme. We can define the simplest – or "vanilla" – RNN as

$$r^i_{t+1} = (1-\alpha)\, r^i_t + \alpha\, \sigma\left(\sum_j W_{ij} r^j_t + \sum_j W^{in}_{ij} x^j_t + b^i \right) \tag{1.22}$$

$$y^i_t = \sum_j W^{out}_{ij}\, r^j_t, \tag{1.23}$$

where upper indices in x_t, y_t and r_t indicate the component; σ is the activation function and α is the so called leakage rate, which in the following will always be set to 1, unless otherwise specified; y is the predicted variable which, in our specific scheme corresponds to forecast on x, i.e. $y_t = \hat{x}_t$. It is easy to realize that the system (1.22) (1.23) is compatible with equations (1.17) (1.18). Universal approximation theorems exist in this case too, for instance see Refs [126, 225].

Neural networks of this kind can be trained with the same gradient descent techniques as in the feed-forward case. Assume we want to use the network to predict x_{T+1} from a piece of history $x_{0:T}$ and consider initializing the network with state r_0, typically $r_0 = 0$. In order to make a prediction, one should iteratively apply equation in (1.22) until one obtains r_T and, then, the prediction is computed as $\hat{x}_T = W^{out} r_T$ with (1.23). Since the same layer is applied recursively more than once, this seemingly shallow network is equivalent in practice to a feed-forward deep one with as many layers as the time steps T, with the fundamental detail that all layers have the same weights (see fig. 1.2). The latter difference does not change the fact that, when differentiating some cost function with respect to the weights, the chain rule (as a basis for backpropagation) still applies. Hence, the RNNs are heavily effected by the vanishing gradients issue described in the previous section.

Long-Short Term Memory

The vanishing gradients problem can be circumvented in at least two ways. The first is the *reservoir computing approach*, which will be explained in sec. 1.4 and the second is the so called Long-Short

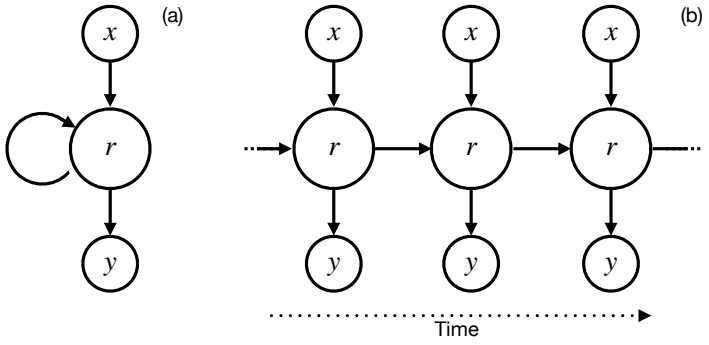

Figure 1.2: RNN in its equivalent recurrent (a) and *unrolled* (b) representations.

Term Memory (LSTM) scheme for RNNs. There exist several kinds of LSTMs but the core idea is to introduce a layer containing special hidden variables (a subset of r_t), the "cell", that are passed from time t to time $t+1$ without being processed by activation functions. Cell variables form a sort of train of data which is designed to store information for an indefinite amount of time. At each time step, information bits are kept, updated, or forgotten or extracted depending on the input x_t (as well as on the other hidden variables, if there is any) through a system of special multiplicative/additive layers, called *gates*. The absence of activation functions acting on the cell partially overcomes the problem of vanishing gradients, thus potentially saving the RNN scheme from one of its most dramatic flaws. LSTM are not amongst the most user-friendly architectures and a more detailed description would be lengthy and beyond the scope of this thesis, so that we refer to the original paper [110], to keras/tensorflow documentation [1] and to the several excellent data-science online articles for further details.

1.4 Recurrent neural networks: reservoir computing or echo state neural networks

1.4.1 Reservoir computing: the power of random neural networks

Reservoir computers: main ideas

In order to bypass the vanishing gradient problem, a possible approach is to adopt a framework known as Reservoir Computing (RC) [226, 254] (see also ref. [241] for a recent review). The reservoir approach is inspired by RNN and, while time series are its natural field of application, it can be employed in a wide number of problems.

In training standard neural networks, all neural connections are tuned in order to solve a certain task. The RC approach starts from a different point of view: *randomly connected neurons are enough* to process information, provided you *have enough of them*; all is needed is to tune a minimal amount of connections, i.e. *readout* connections.

In order to further illustrate this abstract claim, assume we have an input vector, say $X \in \mathbb{R}^{d_i}$, and we want to extract some information from it, say $Y = Y(X) \in \mathbb{R}^{d_o}$, as the desired output. The main ingredient is a very high dimensional (with respect to the input) auxiliary system, called the *reservoir*, whose state can be called $r \in \mathbb{R}^{d_r}$ with $d_r \gg d_i, d_o$. The reservoir should be understood informally as a large, complex and disordered system which exhibits non-trivial responses to external

perturbations. For instance, it can be a collection of randomly connected neurons, whose links are fixed and will not be trained at all. The reservoir should be initialized with some initial condition r_0 and then coupled to the signal X: as a result, the state of the reservoir is updated and to a new state $r = r(r_0, X)$, which is the result both of reservoir-to-reservoir and input-reservoir interactions. After this process is complete, the information contained in the input signal X has spread over many degrees of freedom: so many that, under proper conditions, the state $r = r(r_0, X)$ may be (even linearly) separable. This means that r can be mapped into Y, ideally with a linear function $Y \approx W^{out} r + b$. This last operation is called *readout* and it is the only direct optimization that one carries directly. Hence, the whole machine learning problem reduces to building a good reservoir and performing a linear regression in order to tune appropriately the connections between the reservoirs and the output layer. The separation of these two tasks is fundamental feature of RC and makes the optimization much easier and faster than in the standard machine learning approach.

Clearly, not every large system is suited as a reservoir. A good reservoir system must be *nonlinear*, for the trivial reason that the composition of linear maps, no matter how complicated, cannot fit nonlinear functions. Moreover, in order for the readout to be reliable, the final state $r(r_0, X)$ must essentially depend on X alone, and memory of the initial condition r_0 should be lost when the reservoir processes the signal. For instance, if X is a temporal signal, and the final state of the reservoir is only a continuous function of the most recent part of the signal, we say the network has the "fading memory" property.

Finally, the system must be complex enough that different inputs X and X' produce distinguishable final states of the reservoir, a condition called separability [93, 273].

The RC framework is not only interesting from a theoretical point of view, since it entails the idea that complex computations can be performed by near random systems, but has some promising hardware applications. Indeed, reservoir neural networks may be implemented with networks of physical neuron-like units, which, with minimal tuning could solve a vast array of problems. Moreover, reservoir computers have been implemented with a number of physical supports [101, 143, 170, 226, 254], including biological neurons [72], quantum systems [88, 92], e.g. photons, as the reservoirs [43]. Therefore, such implementations may allow for extremely fast computations and constitute a promising direction in experimental physics.

Echo state neural networks

Echo State Recurrent Neural Network (ESRNN)s are perhaps the most prominent example of the RC framework and were introduced by Jaeger about 20 years ago [119, 121–123, 162]. This approach to recurrent networks, which is gathering much attention for model-free and data-driven predictions of chaotic systems [159, 179, 191, 192, 258], involves rethinking the usage of the vanilla scheme (see eqs. (1.22) (1.23)) rather than introducing novel layers as in the LSTM case. The equations describing the network are almost the same as in (1.22) (1.23):

$$r^i_{t+1} = (1-\alpha)\, r^i_t + \alpha \tanh\left(\sum_j W_{ij} r^j_t + \sum_j W^{in}_{ij} x^j_t + b^i\right) \tag{1.24}$$

$$y^i_t = \sum_j W^{out}_{ij}\, R^j_t \tag{1.25}$$

or, in continuous time,

$$\dot{r} = -\alpha\, r + \tanh\left(\sum_j W_{ij} r^j + \sum_j W^{in}_{ij} x^j + b^i\right) \tag{1.26}$$

$$y^i = \sum_j W^{out}_{ij} R^j. \tag{1.27}$$

where, for now, we are keeping input vector $x \in \mathbb{R}^{d_i}$ and $\mathbb{R}^{d_o}$ distinct, for a more general formulation, which will be relevant in presenting the original results of this thesis. As for the vector R, it is obtained by some element-wise non-linear function $\hat{R}^i$ such that $R^i = \hat{R}^i(r^i)$. For instance, in this thesis, we will use

$$R^i = \hat{R}^i(r^i) = \begin{cases} r^i \text{ for even } i \\ [r^i]^2 \text{ otherwise,} \end{cases} \tag{1.28}$$

for reasons that will be explained later. The rationale for introducing the mapping $r \mapsto R$ before the linear readout is that this additional non-linearity may improve the performance, but it can also be employed to enforce known symmetries, as will be made clear in the application [191, 192]. Note that the discrete formulation with $\alpha = 1$ is a common choice.

The matrix $W \in \mathbb{R}^{d_r} \times \mathbb{R}^{d_r}$ describes the internal connections of the reservoir. As hinted in the discussion of the general RC framework, such connections are chosen randomly and never trained in a supervised fashion. Moreover, it is common practice to use sparse connections, in order to make the computations faster. Note that some unsupervised training of these connections might boost the performance [119, 227, 273]. The matrix $W^{in} \in \mathbb{R}^{d_i} \times \mathbb{R}^{d_r}$, on the other hand, describes the incoming connections from the input layer to the reservoir. These connections are created randomly as well and never optimized. Finally, $W^{out} \in \mathbb{R}^{d_r} \times \mathbb{R}^{d_o}$ maps the state of the reservoir to the prediction layer. This last matrix is the only object which will be determined by an explicit optimization (as detailed below).

The network can be used in two different modes. The first is called *open loop* [217] or, as in Pathak et al. [159, 191] language, *listening* (see fig. 1.3). It essentially consists in updating the state $r_{t-1} \mapsto r_t$ using (1.24) with elements x_t from a given sequence of inputs. In this case, r_t is a non-autonomous dynamical variable. The second mode (see fig. 1.3) can be called *closed loop* or *prediction mode* and consists in connecting the output layer at time t with the input layer at time $t+1$ (as in passing from eqs. (1.17) (1.18) to eq. (1.19)), by assuming $x_t = K(y_t)$ for some function K. In the simplest case, we can choose $y_t = x_t$ ($K = \mathbb{I}$), such as in eq. (1.19). In this scheme, we get an autonomous system:

$$\begin{cases} r^i_{t+1} = (1-\alpha)\, r^i_t + \alpha\, \tanh\left(\sum_j W_{ij} r^j_t + \sum_j W^{in}_{ij} x^j_t + b^i\right) \\ y^i_t = \sum_j W^{out}_{ij} R^j_t \\ x^i_t = K^i(y_t). \end{cases} \tag{1.29}$$

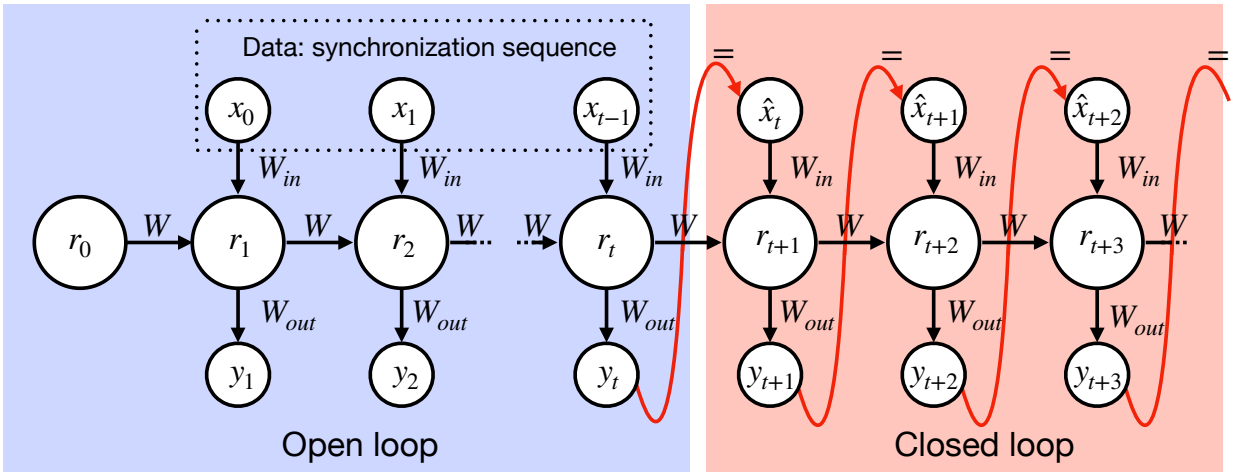

Figure 1.3: A scheme showing the basics of echo-state neural networks: open and closed loop. In closed loop, the output at a certain timestep is used an input in the next: long term predictions are possible.

Properties of a good reservoir

As outlined in the general discussion, the role of the reservoir state is to store and process input information. Consider a given sequence of data $x_{0:T}$, an initial condition of the reservoir $r_\star$ and write equation (1.24) as

$$r_{t+1} = F(r_t, x_t) \tag{1.30}$$

(compare with equation (1.17)). By applying T times the network (1.24) in open loop mode, we can write

$$r_T(r_\star, x_{0:T-1}) = [F(\cdot, x_{T-1}) \circ F(\cdot, x_{t-2}) \circ ... \circ F(\cdot, x_0)](r_\star). \tag{1.31}$$

Our goal is to extract, via the readout layer, a prediction from r_T, under the assumption that r_T represents the history of x_t. In particular, the reservoir should only encode information about the input and not not being contaminated by spurious information about the initial condition $r_\star$: this feature is formalized with the *echo state property* and the *fading memory property*. They can be rigorously stated in several ways (e.g. see ref. [119, 164]), and both revolve around the idea that, in open loop mode, past information is forgotten when a sufficiently long driving sequence is provided. However, in spite of their names, they are distinct concepts in the literature: the former refers to the requirement that the input history uniquely determines the state of the reservoir in open loop; the latter is a property of continuity of final state with respect to the recent input history in open loop. Therefore, they are an existence-and-uniqueness and continuity conditions, respectively.

We can informally write the echo-state property in the following way: for any initial condition $r_\star$ and for any bounded sequence $x_{-\infty:t}$, then the limit

$$r_t = \lim_{T\to\infty} r(r_\star, \hat{x}_{-T:t}) \tag{1.32}$$

exists and does not depend on $r_\star$; r_t is called *echo state*. The way this property is usually written is however the following: given any left sequence $r_{-\infty:0}$ there exists a unique compatible input sequence $x_{-\infty:0}$ such that $r_{t+1} = F(r_t, x_t)$; accordingly, we can define *the echo state function* $\mathcal{E}$ as

the function which assigns the echo state to a certain left sequence

$$r_0 = \mathcal{E}(x_{-\infty:0}). \tag{1.33}$$

Another useful formulation of this property is the following: if we initialize the same network with different initial conditions, the states of the two reservoirs converge in time as long as the two reservoir are fed the same history $x_{0:t}$. Namely, we can require that

$$\lim_{t\to\infty} \|r(r_\star, x_{0:t}), r(r'_\star, x_{0:t})\| = 0 \quad \forall r_\star, r'_\star. \tag{1.34}$$

This property is reminiscent of the concept of generalized synchronization in dynamical systems, as it will be discussed later.

A network has the *fading memory property* [40, 119, 163] if is the final state of the reservoir is continuous with respect to the *recent* input history: informally, this means that sequences which are similar in their terminal part produce similar final reservoir states. Namely[10], an echo state network defined by its echo state function $\mathcal{E}$ has the fading memory property if, for any[11] left sequence $x_{-\infty:0}$ and for any $\epsilon > 0$, there exist some $\delta > 0$ and $T \in \mathbb{N},\ T > 0$ such that, for any other left sequence $x_{-\infty:0}$

$$\|x_{-t} - x'_{-t}\| < \delta\ \forall t = 0, ..., T \implies \|\mathcal{E}(x_{-\infty:0}) - \mathcal{E}(x'_{-\infty:0}))\| < \epsilon. \tag{1.35}$$

Notice that this is analogous to the standard $\epsilon - \delta$ definition of continuity.

Note that the previous conditions are trivially satisfied in some unwanted situations, for instance if $F(r, x) = \text{const}\ \forall x, r$. Indeed, different input stories should be encoded into different reservoir states, a property dubbed separability. This can be studied with the separation ratio [93] by considering the relation between $\|r_t - r'_t\|$ and $\|x'_t - x_t\|$ for different pairs of input histories $\{x_t\}$ and $\{x'_t\}$ and their associated reservoir state histories $\{r_t\}$ and $\{r'_t\}$. If large values $\|x'_t - x_t\|$ do not correspond to large values of $\|r_t - r'_t\|$, then the reservoir is likely to perform poorly. Note that, to the best of my knowledge, separability is more used as a powerful empirical test than as a mathematical requirement, and pathological situations are generally already ruled out by the hypotheses of the theorems in the mathematical literature.

1.4.2 Implementing echo-state neural networks

In specific settings, the echo state state property can be enforced in several ways. In the case of the reservoir given by equation (1.24), a common recipe consists in tuning the spectral radius $\rho(W)$ (which is the greatest eigenvalue, in absolute value) of the reservoir-to-reservoir connectivity matrix W. For instance, $\rho(W) < 1$ is sometimes used as a guess but, as a matter of fact, it is neither a sufficient nor a necessary condition: extensive analyses on how to define and enforce the echo state property can be found in Refs. [89, 119, 125, 188, 272].

As example, we mention that, in the case of the standard reservoir computer given by (1.24), it

[10] This is a simplified formulation in a special case, see ref. [40] for the original definition.
[11] Bounded and belonging to an appropriate domain.

can be shown that if $\alpha = 1$ (the case we will be using) then[12] that

$$\|F(r, x) - F(r', x)\| \leq \Lambda \|r - r'\| \; \forall r, r', x \tag{1.36}$$

with Λ being the leading singular value of the reservoir-to-reservoir connectivity matrix W. Therefore, if $\Lambda < 1$, any two initial conditions would converge at least exponentially, implying the echo state property. Notice that this a sufficient but not necessary condition. In this thesis, we will use the spectral radius as an hyper-parameter and we will fix it with an empirical analysis, as explained later in chapter 2. Note that we are interested in having good long term prediction performances: for this purpose, the echo state property is a desirable feature but does not guarantee high quality closed loop forecasts [217].

Optimization

The only part of the ESRNN that is directly optimized is the readout layer. For this purpose, we need an input and an output training trajectories, $x_{-T:0}$ and $y_{-T:0}$, which reduce to a single trajectory $x_{-T:0}$ if the input and output variables coincide. Let us consider a random initial condition for the reservoir variables $r_{-T} = r_\star$. By exploiting the open loop equation (1.24) (see fig. 1.3), we compute a sequence of reservoir states $r_{-T:0}$.

Owing to the echo state property, we know that, input after input, the network forgets the initial condition and starts synchronizing its internal state with the input history. The length T of the trajectory should be long enough to allow full synchronization and for the echo state to be found: one can even discard the first part of the sequence and use it for the sole purpose of fixing the initial condition as an echo state.

Then, we can apply a possible nonlinear mapping and get a new sequence as $r_{-T:0} \mapsto R_{-T:0}$ (for instance, as in eq. (1.28)). Now we must find the optimal W^{out} that maps R_t to y_t. For this purpose, we have to introduce a loss function, that, in the ESRNN framework, is typically chosen to be the regularized mean square error:

$$L = \frac{1}{T+1} \sum_{t=-T}^{0} \|W^{out} R_t - y_t\|^2 + \beta \, \mathrm{tr}([W^{out}]^T W^{out}) \tag{1.37}$$

where the term proportional to β is called Tikhonov regularization [31] and it essentially penalizes large entries in the output matrix W^{out} (this is also called ridge regression). In order to minimize the loss function (1.37), we have to impose

$$\frac{d}{dW^{out}} L = 0, \tag{1.38}$$

which yields the standard expression for the slope in a least-square linear regression

$$W^{out} = \langle y \otimes R \rangle \, [\langle R \otimes R \rangle + \beta \mathbb{I}]^{-1} \tag{1.39}$$

[12] Observe that $|z - z'| \geq |\tanh(z) - \tanh(z')| \; \forall z, z' \in \mathbb{R}$. Then

$$\sum_i [F^i(r, x) - F^i(r', x)]^2 \leq (r - r')^T W^T W \, (r - r') \leq \Lambda^2 \|r - r'\|^2$$

with last passage following from the definition of leading singular value.

where

$$\langle f \rangle = \frac{1}{T+1} \sum_{t=0}^{T} f_t \tag{1.40}$$

is the empirical time averages along the training trajectory. Expression (1.39) further clarifies the role of the Tikhonov regularization. Matrix $\langle R \otimes R \rangle$ may have a very large ratio between its largest and smallest eigenvalues (in absolute value). The smallest eigenvalue can behave as numerical zero and destabilize the matrix inversion. Note that, if the size of the reservoir d_r is larger than T, then $\langle R \otimes R \rangle$ has at least $d_r - T$ null eigenvalues. This can be easily understood by looking observing that $\Pi_{R_t} = R_t \otimes R_t$ is proportional to a projector along the R_t direction and, therefore, it has a single non-null eigenvalue. Thus, the kernel of the sum of operators $\sum_{t=0}^{T} \Pi_{R_t}$ is $\{R_0, ..., R_T\}^{\perp}$ and has dimension $d_r - T$, provided all R_ts are linearly independent.

Closed loop mode: forecasts and climate

Once W^{out} has been computed, if there exists a way to map the output y_t into x_t, say $x_t = K(y_t)$ from eq. (1.29), we can close the loop in (1.24) (1.25) (see fig. 1.3) and run the network as an autonomous system. From the way we have presented the training procedure, one might deduce that the terminal element, say x_T, of the training sequence $x_{0:T}$, is the only possible initial condition for making predictions and that, in order to use a different initial condition, a new training is needed. This is not the case, though: any initial condition is viable, provided that the reservoir is synchronized and that the training trajectory length was long enough to produce a "good" W^{out}. The possibility to use different initial conditions can be regarded as the generalization property in the context or ESRNN.

Assume, for instance, we want to make predictions from an initial condition $x_\star$ at time $t = 0$. The main issue is that we need to fix the appropriate initial condition r_0 for the reservoir at time $t = 0$ as well. For this purpose, we need an appropriate synchronization sequence, say $x_{-T:0}$, generated by the correct dynamics, such that $x_0 = x_\star$. First, choose a random initial condition $r_\star$ for the reservoir. Then, after feeding the sequence to the network in open loop mode, we can get the synchronized reservoir state $r(r_\star, x_{0:T})$. Thanks to the echo state property, if the synchronization sequence is long enough, then $r_\star$ is the echo state, which "represents" $x_{-T:0}$ and, therefore, we can safely close the loop and make predictions. See fig. 1.4 for a pictorial representation of such process.

The necessity to use a synchronization sequence before starting to make closed loop predictions may seem like a week spot in the method. This may indeed be the case if the signal x_t is Markovian itself, since the synchronization sequence does not add any information that the initial condition $x_\star$ does not carry. However, in the general case in which x_t is the observable part of a larger non-observable system s_t, this feature turns out to be advantageous. Indeed, the usage of a synchronization sequence may allow the reservoir state r_0 to represent s_0, which would correspond to the best possible reconstruction.

It should be remarked that, as it is often the case in machine learning applications to time series, the network has been optimized to make one step forecasts, i.e. to extract a prediction for x_{t+1} from r_t but not for predicting long sequences. This is an important point, since the ability to make accurate one step forecasts in open loop mode does not necessarily result into a good statistical performance in making medium-long term forecasts [39, 191, 217] in closed loop or into reproducing

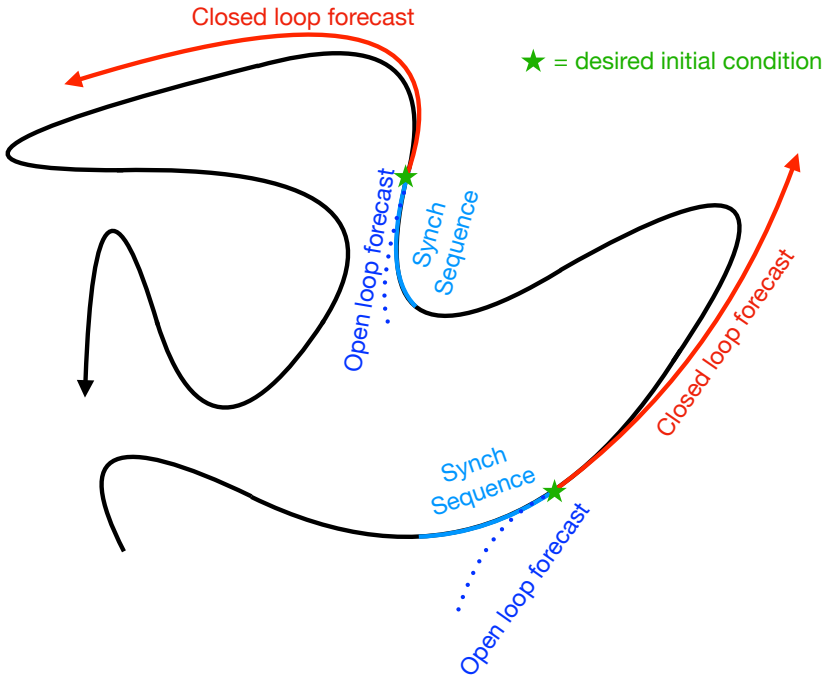

Figure 1.4: Making forecasts about the black trajectory with a trained ESRNN. Suppose we want to use initial conditions corresponding to the green stars. We need synchronization sequences in light blue which terminates with such desired points. Such sequences are fed to the network in open loop mode: as the net synchronizes, open loop one-step forecasts (blue dotted lines) become and more and more accurate. After synchronization, we can close the loop and attempt long term forecasts (red lines).

the statistical properties of the original dynamics in the long run (the so called *climate* [192]). The main problem is that, even if one-step prediction errors are small, they may be in the *wrong direction* and bring the forecast far from the manifold. In dynamical system framework, it means that the system may not be structurally stable [33] and the variable may leave the attractor, since there may be a positive spurious expanding direction transversal to the attractor itself [158]. As a result, after the optimization, one may need to separately check the quality of long term predictions and possibly choose hyper-parameters suited for such purpose.

Universal approximation theorems

In order to be theoretically sound, any neural network technique should be backed by universal approximation theorems. We restate that such theorems guarantee the possibility to build a neural network that solves a certain task with arbitrary precision, but do not necessarily imply that a solution can be found at all within reasonable computation time and that the generalization properties are good. These two last points require separate analysis.

Indeed, there exists approximation theorems for ESRNNs, which however, to the best of my knowledge, do not fully explain computational results yet and are still a work in progress. It is important to remark that the problem of approximation and generalization in random recurrent networks is especially hard (and fascinating), since a good performance is needed both in the open loop and the closed loop modes (see fig. 1.3) after training. The first functioning mode is somehow easier to study, but open loop stability does not fully characterize (see ref. [217]) the closed loop

dynamics. Moreover, since networks are random, different realizations may perform differently: for the ESRNN approach to be reliable, good properties must be true for the typical random neural network generated with a certain statistics. For these reasons, we feel it is worth providing a minimal and informal overview of some relevant mathematical literature.

A universal approximation theorem has been formulated in the open loop case by Grigoryeva and Ortega [96, 99] by exploiting the fading memory property. Define a filter[13] as a function $\mathcal{F} : (\mathbb{R}^{d_i})^{\mathbb{Z}} \to (\mathbb{R}^{d_o})^{\mathbb{Z}}$ so that $\mathcal{F} : x_{-\infty:\infty} \mapsto y_{-\infty:\infty}$; notice that a filter naturally defines a map between left sequences $\mathcal{F} : x_{-\infty:t} \mapsto y_{-\infty:t}$. A filter is causal[14] if, given two sequences $x = x_{-\infty:\infty}, x' = x'_{-\infty:\infty} \in (\mathbb{R}^{d_i})^{\mathbb{Z}}$ such that $x_{-\infty:t} = x'_{-\infty:t}$, then $\mathcal{F}_t(x') = \mathcal{F}_t(x)$ and time invariant if it commutes with time translation operator. If the echo state property holds, a network $r_{t+1} = F(r_t, x_t)$ (see eq. (1.30)) is associated to a unique filter $\mathcal{F}_F$. Then, a theorem [96, 99] state that, for any causal, time invariant filter $\mathcal{F}$ with fading memory[15]; for any Lipschitz-continuous, non constant and bounded activation function; $\forall \epsilon > 0$; there exists an echo state network F, satisfying both echo state and fading memory properties, so that its associated filter $\mathcal{F}_F$ is such that

$$|||\mathcal{F}_F - \mathcal{F}|||_\infty < \epsilon \tag{1.41}$$

with

$$|||\mathcal{F}|||_\infty = \sup_{x_{-\infty:0}} \sup_{t \leq 0} \|\mathcal{F}_t(x_{-\infty:0})\|. \tag{1.42}$$

Let us explain this informally. Assume we have a function (a filter) $\mathcal{F}$ that maps input sequences $x_{-\infty,\infty}$ into output sequences $y_{-\infty,\infty}$. We require that the image y_t at position t only depends on x_s for $s < t$ (causality); that it is a continuous function of the recent history of x_s, $s < t$ (fading memory); and time-translation invariance of the filter (stationarity). Then, the filter can be approximated uniformly with arbitrarily high accuracy by a recurrent neural network which displays both echo state and fading memory properties. Note that, this result, however beautiful, implies the existence of a specific open loop recurrent architecture, but tackles neither randomness of the reservoir connections nor closed loop stability.

Such two issues have been studied by Hart et al Refs. [104]. The authors directly tackle the problem of connection randomness, by proving a Random Universal Approximation Theorem, which extends the very well-known universal approximation theorems for FFNNs [114, 224], by stating that any smooth function on a compact set can be approximated with probability arbitrarily close to 1 by a FFNN with a single hidden layer randomly connected to the input layer. Since this theorem is both relevant and very useful in guiding intuition, we report it formally. Let $f : [0:1]^n \to \mathbb{R}$ be a C^1 function; let $\sigma : \mathbb{R} \to \mathbb{R}$ be a differentiable function such that $\int dx\, |d\sigma/dx| < \infty$; let $\{g_j\}, \{v_{jk}\}$ be sequences of i.i.d. random variables with full support (probability positive almost everywhere). Then, $\forall \epsilon > 0,\ q \in (0,1)\ \exists n \in \mathbb{N}$ and $w \in \mathbb{R}^N$ such that

$$P\left(\|f - f_{net}\|_\infty + \|f' - f'_{net}\|_\infty < \epsilon\right) > q \tag{1.43}$$

[13] In this context, it is a function between sequences.

[14] Causality means that, if two sequences x, x' from the filter domain are equivalent from $-\infty$ up to a position t, then their image sequences $\mathcal{F}(x)$ and $\mathcal{F}(x')$ coincide at position t.

[15] The authors use a more general definition than the one presented in this thesis.

with

$$f_{net}(u) = \sum_{j=1}^{N} w_j \sigma \left(\sum_{k=1}^{n} v_{jk} u_k + g_j \right). \tag{1.44}$$

Informally, if we look at the ESRNN structure (eqs. (1.24) and (1.25) with $R = r$ and $\alpha = 1$), we may realize that the output may be written as $y_t^i = \sum_j W_{ij}^{out} \tanh\left(\sum_k W_{jk} r_t^k + \sum_a W_{ja}^{in} x^a + b_j\right)$. Hence, by looking at eq. (1.44), we may identify $(W, W^{in}) \to w$, $b \to g$, $(r, x) \to u$, $\sigma \to \tanh$: we now have a hint on why an ESRNN may be good approximators for regular dynamics.

The context in which the authors employ this theorem is the following. They consider[16] a discrete time structurally stable dynamics, given by some diffeomorphism $S : A_s \to A_s$ with $A_s \subset \mathbb{R}^n$ and assume one dimensional observations $M : A_s \to \mathbb{R}$. The authors discuss the possibility to build an embedding of A_s into the space of the reservoir variables[17]. First, take any $s \in A_s$ (say at time 0) and run the (invertible) dynamics S backward by t steps; collect the sequence $\{S^\tau(s)\}_{\tau=-t}^0 = s_{-t:0}$ and get the sequence of observations $\{M(S^\tau(s))\}_{\tau=-t}^0 = x_{-t:0}$. Now we may define[18]

$$z_t(s, r_\star) := r(r_\star, x_{-t:0}) = r(r_\star, \{M(S^\tau(s))\}_{\tau=-t}^0) \tag{1.45}$$

for any initial condition $r_\star$, by running the open loop dynamics forward. If the echo state property holds, the limit $\mathcal{R}(s_0) = \lim_{t\to\infty} z_t(s_0, r_\star)$ exists and is independent of $r_\star$. Hence, the function $\mathcal{R}$ is a mapping between A_s and the reservoir space, which assigns a unique echo state $\mathcal{R}(s)$ to each possible state of the environment $s \in A_s$; therefore, $\mathcal{R}$ is the right candidate for an open-loop embedding of the dynamics S. Note that this procedure is in fact a form of generalized synchronization (see below).

The authors proceed in proving, among other results, that, with appropriate hypotheses, a class of ESRNN satisfies what they call the ESRNN approximation theorem. Thanks to the universal random approximation theorem, they show that, *if* an ESRNN embeds a structurally stable dynamics S in the reservoir space (with the above procedure), for any $q > 0$, there exists a number of hidden neurons and a readout matrix W^{out} such that, with probability q, the closed loop dynamics of the network[19] is structurally stable and topologically conjugate to the target S. There are two important caveats: first, the authors construct the connectivity matrices W and W^{in} in a somehow non-standard way; second, the topological conjugation between the closed loop echo state dynamics and S does not directly provide the usual $\epsilon - \delta$ approximation scheme employed in universal approximation theorems. Therefore, while being insightful, this result is not immediately relevant for assessing predictions in a standard sense. We refer to the paper for further details.

Echo state as generalized synchronization.

It has been proposed (see ref. [158]) that the functioning of ESRNN may be interpreted as a form of generalized synchronization. Generalized synchronization [205, 221, 275] between a master dynamical variable m_t and a slave variable z_t happens when, for large times, z_t is fully determined by m_t. More precisely, say that $m_{t+1} = f_m(m_t)$ and $z_{t+1} = f_z(z_t, m_t)$: then z_t synchronizes to m_t

[16] We use a notation compatible with that from sec. 1.1. Consistently, A_s is the invariant manifold of the environment dynamical variable s, while x is the observation.

[17] They formulate a conjecture about this.

[18] See subsection "Properties of a good reservoir" from sec. 1.4.1 for the notation.

[19] Of the reservoir variable.

in a generalized sense if, for appropriate initial conditions, there exists some function ϕ such that $\lim_{t\to\infty} \|\phi(m_t) - z_t\| = 0$. Therefore, the manifold $\phi(m) - z$ is stable and attractive. While finding ϕ is, in general, a difficult task; generalized synchronization may be verified by checking numerically that, given the same driving trajectory $m_{0:\infty}$, trajectories of $z_{t:\infty}(z_\star)$ with different initial conditions - say $z_\star$ and $z'_\star$- all converge to the same value, i.e. $\lim_{t\to\infty} \|z_t(z_\star) - z_t(z'_\star)\| = 0$.

In the case of ESRNN, one may recognize the analogy between the fading memory property and generalized synchronization. The conjecture in ref. [158] is that the state of the reservoir r_t synchronizes to the environment state s_t (as in sec. 1.1). Therefore, we may assume that there exists a function ϕ such that $r_t = \phi(s_t)$. Ideally, ϕ should be invertible or at least carry as much information as needed to make forecasts. In the special case in which $s_t = x_t$, then it should be possible to map r_t into x_t with an invertible function. Note that this argument is essentially a "physical rephrasing" of the mathematical formulations from ref. [104], which was mentioned in the previous section about universal approximation theorems.

Deep and multi-reservoir echo state recurrent neural networks

So far, we have given an overview of the features of echo state neural networks as given by eqs. (1.24) (1.25). Such architecture can clearly be regarded as shallow from the point of view of standard neural network-based machine learning. Indeed, state of the art machine learning often employs stacked LSTM layers with possible multi-layered readouts. These deep structures still operate as RNN, while gaining the perks of deep neural networks in terms of feature extraction. In the case of time series, stacked recurrent layers may improve the description of multiple time scales. It is therefore pretty natural to wonder whether such deep structures are compatible with the reservoir computing framework. The answer is of course affirmative. The main idea is to build a network of interconnected reservoirs, which do not need to have the same hyper-parameters - e.g. spectral radius, leakage rate, number of neurons. Reservoirs can be connected in several ways, for instance in parallel, in a convolution-like structure [191], or in a deep sequential manner, such as in ref. [153], where this idea has been employed to forecast rare events. A thorough and clean overview of the topic can be found in ref. [273], while for a more mathematical analysis we refer to ref. [90]. Perhaps unsurprisingly, carefully tuned deep echo state neural networks seem to outperform shallow ones. See fig. 1.5 for a sketch about multi-reservoir architectures.

Hybrid-scheme

It is often the case that, alongside with data, one has some insight into the physics of a system of interest, and can write some approximated equations. In this scenario, one does not necessarily have to choose between model-based or model-free approaches, but may opt for the so called *hybrid scheme*, which is a general framework for machine learning, not limited to ESRNN [172, 260, 264, 267]; the hybrid approach essentially consists in assisting model-free forecasts with some physical information.

In the context of ESRNN (see [194]), the idea is simply to augment the input vector

$$x_t \mapsto (x_t, \tilde{x}_{t+1}), \tag{1.46}$$

i.e. to provide, besides the variable x_t at time t, also a physically informed prediction $\tilde{x}_{t+1}$ at time

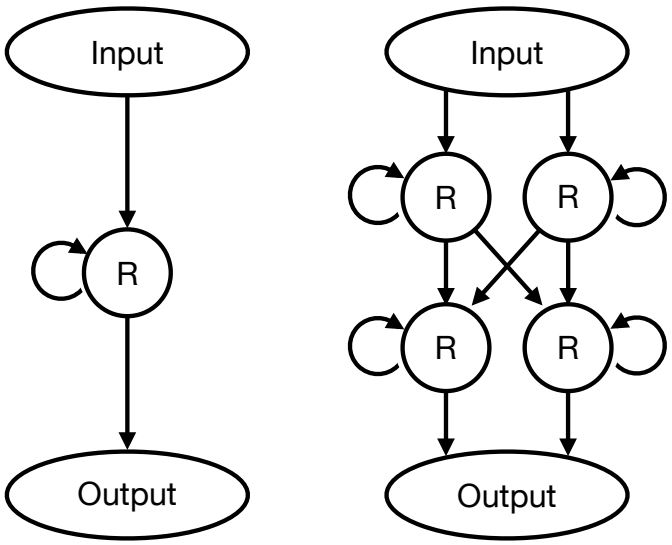

Figure 1.5: Single reservoir scheme (on the left) and a scheme with multiple interconnected reservoirs

$t+1$, while, at the same time, increasing the size of the input matrix $W^{in} \mapsto (W^{in}, \tilde{W}^{in})$. This is a very flexible scheme, which can be employed both in single or multiple reservoir settings, and details may depend on the specific implementation but, in general, the more accurate the auxiliary model is, the more the a neural network may rely on it rather than on data-driven forecasts and the other way round.

1.5 About the choice of the best model, hyperparameters, and the rationale behind machine learning application in this thesis.

Without any claim of having covered anything but a small fraction of the rapidly expanding machine leaning landscape, we have presented three different approaches to neural network-based dynamics reconstruction: FFNN, LSTM and ESRNN. ESRNN and LSTM are based on the same recurrent scheme and can be written with almost the same equations, but belong to otherwise very different paradigms. On the other hand, in some sense, LSTMs and FFNNs may be viewed as having a strong affinity from an algorithmic point of view, since they both are backpropagation-based mainstream machine learning techniques, which can be easily implemented in tensorflow [1]. It is very natural to wonder what is the most performing technique in modelling chaotic systems. While noting that this question was tackled, in a specific case, in ref. [56], in the rest of this section we will try to argue that this is a tricky question that we cannot really answer, at least within the scope of this thesis.

The first simple consideration is that the best technique depends on the goal. For instance, consider the ESRNNs: at least in the mainstream framework, they are naturally suited for handling deterministic trajectories, while even standard tensorflow libraries provides a wide array of techniques for stochastic problems. But, in a broader context, the best technique depends on whether one is aiming at accuracy, theoretical insight, computational efficiency, good inference from noisy data, anything else or any combination of the above. For instance, consider reservoir computing. It

may be argued that it is probably underwhelming when aiming at raw accuracy, since any quenched random neural connectivity structure is available, in principle, through an optimization of all connections (as in the case of LSTMs or FFNNs), while the converse is not true: there is no reason why the optimal connectivity (w.r.t. a certain task) structure should be available through a random realization from a fixed probability distribution (with certain hyperparameters). Of course, this is the reason why construction of good reservoir (i.e. good probability distribution for connections) is an active field of research but, at present, to the best of my knowledge, there exists no recipe for a truly universal reservoir. In this sense, the need to fix hyperparameters in ESRNNs is equivalent to a hidden optimization, which can be either done with some search technique, by hand or, in the best-case scenario, from theoretical considerations. On the other hand, once hyperparameters have been chosen, ESRNN have many positive features. The optimization becomes very simple and fast; moreover, if one uses sparse networks, ESRNNs may be very fast at computing predictions even if the network is very large. Even more importantly, as already mentioned, ESRNNs have promising hardware implementations and, finally, they are very interesting from a theoretical point of view since understanding how randomly connected networks allow computation is a fascinating topic itself.

There is another point which should be highlighted: unlike many techniques traditionally employed by physicists, advancements in applied machine learning are – with some notable exceptions – strongly driven by the data science community, while many in the physics communities have embraced this field only recently. This is unsurprising, since machine learning has several non-academic applications by now. Moreover, high-performance techniques are complex; they require experience and specific knowledge which may not generally belong to the traditional physicist's background, though this seems to be changing. Furthermore, since machine learning owes its success to the possibility of efficiently handling large amount of data, there is a pressure to develop hardware-friendly and scalable techniques. This implies that techniques co-evolve with hardware and software development rather than just theoretical understanding.

Even if we set all the previous issues aside, any comparison between different techniques implies that one is able to choose good hyperparameters in each case. This is not necessarily easy since there are many of them: just to name a few, we can mention the number of layers, the number of neurons per layer, the connectivity structure, the activation functions, the optimization algorithm, the regularization and so on. While, in some cases, there are recipes and techniques for searching the best hyperameters, the space to explore remains huge.

We can now finally articulate the implicit question in this subsection: "How do we know that the works presented in this thesis employ the best techniques? How do we know that the hyperparameters are the most performing ones?". The honest answer is "we don't". However, that doesn't mean it is hard to find *good* hyperparmeters. Furthermore, if the parameters need to be excessively fine tuned, we could conclude that a technique is not robust. Indeed, the idea behind the works presented in this thesis is to select a reasonably good technique, based on the problem we are examining and study the physics of the problem. For this purpose, we have chosen problems and settings which allowed us to investigate modelling and physics without worrying too much about possible machine learning failures. Hence, we opted for simple techniques, as much as we could, while giving reasonable but not paramount importance to the choice of hyperparameters.

Therefore, the rationale for the validity of our choices is the following. If our machine learning

approaches had not worked, anything could have been possible: maybe the tool was wrong, maybe the hyperparameters were poorly chosen, maybe the problem was physically too hard. However, the fact that the machine learning techniques we have chosen have yielded physically meaningful results validates our methods.

For completeness sake, it is worth mentioning that the problem of hyperparameters has played a way deeper role in the presented works and has required way more effort than it may appear in this thesis. However, this is most likely something that the reader would not be interested in, and will be mostly omitted.

Chapter 2

Effective models and predictability of chaotic multiscale systems via reservoir computers

Multi-scale systems, either in a spatial or temporal sense, are paramount in physics and, in general, in any quantitative science, both for theory and applications. As an example, stars have historically being used as a fixed reference, since their perceived motion is order of magnitude slower than many phenomena of interest on Earth. Alternatively, consider biology: one can study ecological interactions of macroscopic organisms, interactions between cells, molecules or atoms. From the point of view of physicists, the existence of multiple scales allows for descriptions at different levels, each requiring to account effectively for lower and/or higher scales, if needed. Furthermore, the multi-scale structure is also a challenge, since one may want to find the way to connect descriptions at different scales.

In this thesis, we will consider the narrower scope of multi-scale dynamical systems. As a classical example pertaining to this context we can mention turbulent phenomena which can easily span over 4/6 decades in temporal/spatial scales [261] or climate, whose dynamics involves daily weather as much as slow variations which evolve over millennia [198, 200].

Among dynamical systems, a very interesting class of systems is given by chaotic systems [51, 186]. Since such systems have been studied for decades in great detail, it would be beyond the scope of this thesis to provide a comprehensive introduction to the topic – which is generally familiar to physicists – thus, here, we will only present the basic formalism to provide some context for our works.

Consider a dynamical variable x_t, with either continuous or discrete time. Informally, such system is chaotic if infinitesimal errors on the initial condition expand exponentially in time over a infinite amount of time as $\delta x_t \sim \delta_0 \, e^{t \, \lambda_{max}}$ where λ_{max} is the maximum Lyapunov exponent (see below). More formally, in the discrete case, to fix ideas, let x_t evolve as

$$x_{t+1} = F(x_t). \tag{2.1}$$

An infinitesimal error dx_t evolves as

$$dx_{t+1} = (dx_t \cdot \nabla)\, F(x_t), \tag{2.2}$$

which is the tangent space equation associated to the system, with ∇F being called stability matrix. This means that the amplitude of the error obeys the law

$$\|dx_t\|^2 = \|dx_0 \cdot \nabla F^t(x_0)\|^2 \tag{2.3}$$

with $F^t = \overbrace{F \circ F \circ ... \circ F}^{t}$. Accordingly, the largest Lyapunov λ_{max} exponent is defined as the largest eigenvalue of

$$V(x_0) = \lim_{t\to\infty} \frac{1}{2\,t} \ln\left([\nabla F^t(x_0)]^T \, \nabla F^t(x_0) \right). \tag{2.4}$$

The existence of the previous limit is granted almost everywhere by the Oseledec theorem and does not depend on the initial condition x_0 as long as ergodicity[1] holds. If $\lambda_{max} > 0$, the system is chaotic. The other eigenvalues of the same matrix describe the rest of the Lyapunov spectrum [25, 51] and there may be more than one positive exponents, but the sign of the largest one is enough to say that the system is chaotic. An equivalent definition for the maximum Lyapunov exponent is

$$\lambda_{max} = \lim_{t\to\infty} \lim_{\|\delta x_0\|\to 0} \frac{1}{t} \ln \frac{\|\delta x_t\|}{\|\delta x_0\|}. \tag{2.5}$$

This is a suitable mathematical definition, however, it is physically problematic in the case of systems with a multiscale structure, as it may fail to capture part of the relevant physics of the system. Since a more quantitative insight will be provided in the next subsection, we give here a more qualitative motivation. By definition, the Lyapunov exponent is valid for infinitesimal errors (see (2.5)) and, therefore, it may only describe the behaviour of the fastest degrees of freedom of the system, which typically appertain to the smallest scale: the fact that small variables are also the fast ones is a common feature in real world scenarios. For instance, in climatic models, the fast subsystem would represent small hydrodynamic eddies or the weather, which are not necessarily important when trying to study slow climatic variables. Indeed, very large errors on weather forecasts may still be small from the point of view of climate and their expansion rate has little impact on the expansion of errors of slow and large-scale phenomena.

The above discussion illustrates the intuitive reason why the expansion rate of the errors cannot be decoupled from the scale at which phenomena of interest happen. Hence, the Lyapunov exponent alone may not be appropriate to describe the expansion of the error on macroscopic variables. A solution was proposed with the introduction of the Finite Size Lyapunov Exponent (FSLE) [13, 14, 34, 51, 55, 157] as an alternative to the maximum Lyapunov exponent in characterizing error growth. Given a certain error size, we can define the FSLE by computing the time it takes for such

[1] We say [51] a system is ergodic if 1) there exists a dynamically invariant measure μ such that, for any measurable set A, $\mu(A) = \mu(F^{-1}(A))$ and 2) for any measurable function S, we have

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} S(F^t(x_0)) = \int d\mu(x)\; S(x)$$

for almost all initial conditions x_0. Informally, temporal averages do not depend on the initial condition and are equivalent to expectation values w.r.t. some invariant measure μ.

error to increase by a factor r. For instance, consider the continuous dynamical variable $u(t)$ and let $\delta(t)$ be the error on such variable at time t. The FSLE at size δ with threshold r may be defined as

$$FSLE(\delta, r) = \frac{1}{\langle T(\delta \to r\,\delta)\rangle} \ln r \tag{2.6}$$

where $T(\delta \to r\,\delta)$ is the time it takes for $\delta(t)$ to grow from δ to $r\,\delta$. Clearly, the $FSLE$ reduces to λ_{max} for $\delta \to 0$. Note that, since this observable – unlike the Lyapunov exponent – is not defined in the tangent space, there are some subtleties which ought to be accounted for. We should mention, for instance, that the $FSLE$ may depend on r, may be a sensible to the specific way it is computed and, in general, may depend on the norm one uses in evaluating the error magnitude. In particular, when the dimension of the system is larger than 1, the initial error size δ underspecifies the initial condition of the error: for the observable to be meaningful, the orientation of the error vector should be aligned along the local unstable direction. A safe way [51, 55] is to let the dynamics itslef select such direction. Algorithmically, one starts with a very small error of size $\delta_0 \ll \delta$ so that, by the time the error has grown to size δ, details on the initial error direction are lost and the error is naturally oriented in the direction of the instability. Moreover, if the dynamics is discrete – or discretized – then eq. (2.6) should be corrected to account for[2] the fact that the errors will not in general be exactly δ or $r\,\delta$.

Therefore, the FSLE approach not only allows to describe the error expansion outside the limited scope of the linear regime (infinitesimal errors), but it can also be used to uncover the multiscale structure of a system, by associating different error-expansion regimes to different scales. A mathematically rigorous way to approach this same problem is to employ the covariant Lyapunov vector formalism [87, 95], which allows to probe the problem of coexistence of fast and slow modes in the same system.

Besides the characterization of the instabilities of a multiscale system beyond the linear regime, an important issue in coping with multiscale systems is the need of modeling. Indeed, it is often impossible, even computationally, to account for all the scales of motion and typically one needs to develop effective models for the degrees of freedom of interest, usually the slow ones. This is precisely the subject of this chapter and it is worth illustrating it with some details. A simple structure for a multiscale system is the following:

$$\begin{cases} \dot{s} = F_s(s) + F_{f\to s}(s, f) \\ \dot{f} = c\,(F_f(f) + F_{s\to f}(s, f)) \end{cases} \tag{2.8}$$

where s and f represent slow and fast variable, respectively. Constant c is the scale separation factor: time passes $c-$times faster for variable f than for variable s and, therefore, we have scale separation for $c > 1$. Building effective equations means finding a closure for the first equation in (2.8) and replacing the interaction term $F_{f\to s}(s, f)$ with an effective one $\tilde{F}_{f\to s}(s)$ which only depends on s:

$$\dot{s} = F_s(s) + \tilde{F}_{f\to s}(s). \tag{2.9}$$

[2] Define

$$FSLE(\delta, r) = \left\langle \frac{1}{T([\delta]_+ \to [r\,\delta]_+)} \ln \frac{[r\,\delta]_+}{[\delta]_+} \right\rangle \tag{2.7}$$

where $[\delta]_+$ and $\delta_+(r\,\delta)$ are first values the error assumes after crossing thresholds δ and $r\,\delta$ respectively; and $T([\delta]_+ \to [r\,\delta]_+)$ is the time elapsed between these two events. This is the algorithm we have employed whenever needed.

The delicate issue here is that, due to the nonlinear character of system, improper modeling of the fast scales usually results in underperforming models of the slow variables. These effects may be especially dramatic in chaotic systems, where perturbations and errors are expanded exponentially. In the following sections, we will implement this ideas for a specific model that we have studied through machine learning in ref. [39], which is the article presented in this chapter.

2.0.1 Two-scale Lorenz system

The two scale Lorenz system is a toy model and for ocean-atmosphere interactions and it is a simple example of a chaotic system with two timescales. It was introduced by Boffetta et al. in [34, 51] and it is obtained by coupling together two standard Lorenz systems [156] with different intrinsic timescales and adding interaction terms. The equations for the two-scale Lorenz system are the following (notice the similarities with eqs. (2.8)):

$$\begin{cases} \dot{X} = a\,(Y - X) \\ \dot{Y} = R_s\,X - Z\,X - Y - \epsilon_s\,x\,y \\ \dot{Z} = X\,Y - b\,Z \end{cases} \tag{2.10}$$

$$\begin{cases} \dot{x} = c\,a\,(y - x) \\ \dot{y} = c\,(R_f\,x - z\,x - y) + \epsilon_f\,Y\,x \\ \dot{z} = c\,(x\,y - b\,z)\,, \end{cases} \tag{2.11}$$

where $U = (X, Y, Z)$ are slow variables and $u = (x, y, z)$ are fast ones. We keep the authors' original setup: $a = 10$, $b = 8/3$, $R_s = 28$, $R_f = 45$, $\epsilon_s = 10^{-2}$ and $\epsilon_f = 10$. Factor c is the tunable scale separation factor, representing the ratio between characteristic timescales of slow and fast variables, respectively. In the following, we will use $c = 10$ to enforce strong time-scale separation and $c = 3$ for small scale separation. To picture this, consider that, in the non interacting case $\epsilon_f = \epsilon_s = 0$, the maximum Lyapunov exponent of the slow subsystem is $\lambda_s^0 \approx 0.906$, while the that of the fast subsystem is $\lambda_f^0 = c\,\lambda_s^0 \approx 9.06$. When interaction is introduced (in the $c = 10$ case), the Lyapunov exponent of the whole system $\lambda = 11.5$ is close to that of fast subsystem. However, the the system presents a multiscale structure, since it has two regimes, characterized by two values of the FSLE: a large one λ_f, which is observed for small perturbations and which is about the size of the maximum Lyapunov of the whole system or of the fast subsystem, corresponding to the fast regime; and a small one λ_s, which is observed for small perturbations and which is about the size of maximum Lyapunov exponent of the slow subsystem, corresponding to the slow regime. We can picture this either in time or in error size; before going further, we should explain how to describe the distinct behaviours of the slow and fast subsystems. We define slow and fast errors as $\|U - U'\|$ and $\|u - u'\|$, respectively, where we may regard the primed variables as corresponding to the the reference trajectories. When both $\|U - U'\|$ and $\|u - u'\|$ are vanishing small, they both expand exponentially in time with the large exponent λ_f. This is bound to happen because, when errors are so small that their dynamics may be described by tangent space equations, the behaviour of the error does not depend on the norm[3]. We can rationalize this by observing that, in this regime, the error on slow scales is driven by the rapidly expanding fast error in the fast subsystem. At some point, the error on fast scales saturates and grows no more: fast variables are

[3]Notice that, strictly speaking, $\|U - U'\|$ and $\|u - u'\|$ are not true norms in the 6 dimensional space (U, u).

now decorrelated. This happens around the slow error value 10^{-2} (which is obviously true only for this specific parameter choice and $c = 10$), which is the *crossover scale* between the fast and slow regime of error expansion. Note that the crossover scale is much smaller than the typical size of the slow variables: in this sense, fast variables are also "small", not because the relation $\langle \|U\| \rangle \gg \langle \|u\| \rangle$ holds (it does not), but because the fast-to-slow interaction is small[4]. Let us now go back to the error expansion dynamics: after a transient, as soon as the error on slow scales has outgrown the the typical size of interaction with fast scales, it starts growing exponentially with the slow FSLE until it saturates as well. Average trajectories describing this behaviour are shown in fig. 2.1, which is the same as what can be found in ref. [34]. Similarly[5], in fig. 2.2, we can see the FSLE (computed with slow variables) as a function of the slow error size: there is a clear crossover around 10^{-2}. Note that, the behaviour of the total error $\|(U, u) - (U', u')\|$ would be similar to that of the slow one (not shown).

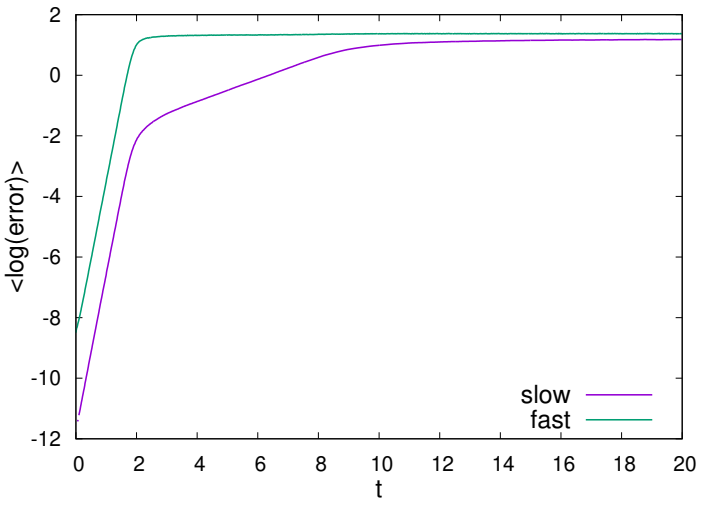

Figure 2.1: Expansion of the average slow (defined as $\langle \log_{10}(\|U - U'\|) \rangle$) and fast (defined as $\langle \log_{10}(\|u - u'\|) \rangle$) errors in time, as in [34]. When the error on fast variables saturates, the slope of the average log-error changes slope, entering the slow regime. This happens when the slow log-error is around -2, which is the crossover scale.

In the following, we will use a ESRNN to make forecasts about the slow part U of the systems, using information about U alone, thus employing the network with optimized output as an effective model for the slow dynamics. Before doing this, though, we should present a standard approximation (exact in the $c \to \infty$ limit) which serves the same purpose: it will we useful both as comparison and to better understand how the neural network operates.

[4] As explained earlier in typical multiscale systems, the fast variables are also smaller in amplitude than the slow ones. In this respect the coupled Lorenz system is not realistic. Anyway, this is not a severe limitation, as the time scale separation is enough for the present discussion.

[5] Note that, in more complicated cases, the slope of the trajectory $E(t) = \langle \log(\text{error}(t)) \rangle$ does not yield the FSLE at scale $e^{E(t)}$, which should be computed with the appropriate algorithm. The reason is that there could be strong fluctuations in the scale of the error size E at time t: there is no way to associate a certain scale to a certain time, not even by fixing the amplitude of the error on the initial condition. In this sense, the previous narrative of the average error expansion is not general.

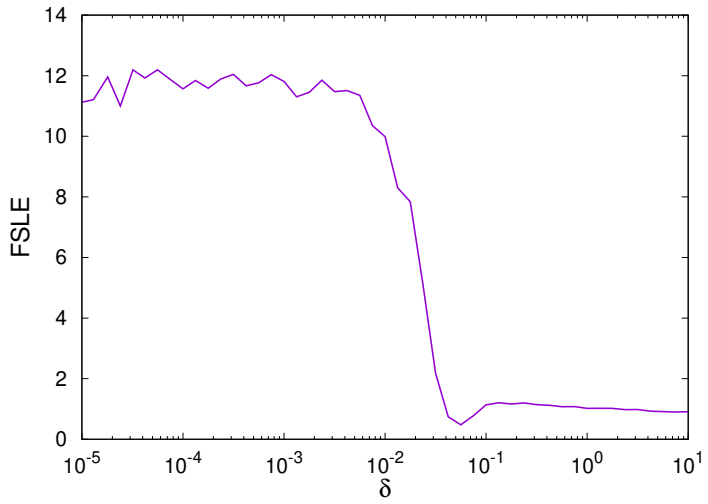

Figure 2.2: FSLE computed for slow variables (i.e. using the norm $\|U - U'\|$) as a function of the error size δ.

2.0.2 Adiabatic and truncated models

In building an effective model for the slow variables of the two-scales Lorenz system, we may observe that, in the slow equations (2.10), the only contribution from the fast subsystem appears in the second equation (for Y) and it is proportional to $x\,y$. Hence, in order to write an effective equation we have to replace $x\,y$ with some function of the slow variable U, thus obtaining a closure for the system (2.10).

In this section we explain how to approach this problem by introducing the adiabatic principle which, in the context of multiscale techniques [195, 223], is a well known framework to build effective models for slow variables. The idea is to assume infinite scale separation (i.e. $c \to \infty$ in (2.11)), i.e. that fast variables are so fast that, while they evolve, the value of the slow ones is essentially fixed. Conversely, from the point of view of slow variables, fast variables can be accounted for statistically i.e. in terms of averages with fixed slow variables – the adiabatic principle. To be more precise, consider the following scheme, equivalent to eq. (2.8), which can be viewed as a a qualitative sketch for a more rigorous approach to scale separation [47],

$$\dot{s} = F_s(s, f) \tag{2.12}$$

$$\dot{f} = F_f(s, f), \tag{2.13}$$

with f and s being fast and slow variables, respectively. Consider some observable $g(s, f)$: we may define the conditional average $\langle g(s, f)|s\rangle$ as the expected value of g with respect to the fast variables f, conditioned to the slow ones s. $\langle g(s, f)|s\rangle$ may be computed numerically by evolving the whole system ((2.12)-(2.13)) and conditioning to s. Alternatively, if we assume ergodicity, one can compute the conditional expected values with Bayes theorem, by using stationary probabilities $p(s, f)$ associated to the coupled slow-fast evolution: since $p(f|s) = p(s, f)/\int df\, p(s, f)$, we may write

$$\langle g(s, f)|s\rangle = \int df \frac{p(s, f)}{\int df' p(s, f')} g(s, f) = \lim_{T\to\infty} \frac{\int_0^T dt\, g(s(t), f(t))\, \delta(s(t) - s)}{\int_0^T dt\, \delta(s(t) - s)}. \tag{2.14}$$

On the other hand, the adiabatic averages are computed only evolving the fast equations (2.13), while keeping s fixed. If we call the associated stationary probability $p_s(f)$, then we write adiabatic averages as

$$\langle g(s,f)\rangle_s = \int df\, p_s(f)\, g(s,f) = \lim_{T\to\infty} \frac{1}{T}\int_0^T dt\, g(s,f(t)). \tag{2.15}$$

In the case of strong scale separation, we have $\langle g(s,f)\rangle_s \approx \langle g(s,f)|s\rangle$ for well behaved systems. Notice that, by construction, $\langle \dot{f}\rangle_s = 0$. With these premises in mind, we define an adiabatic model for the two-scale system as

$$\dot{s} = \langle F_s(s,f)\rangle_s \tag{2.16}$$

Let us now apply this principle to the Lorenz model. The effect of fast variables on slow ones in eq. (2.10) is given by $x\,y$. On the other hand if we take the adiabatic average in the third equation in the system (2.11), we can write

$$\langle x\,y\rangle_U = b\,\langle z\rangle_U. \tag{2.17}$$

This last equation contains the average we are looking for and, therefore, we shall focus on $\langle z\rangle_U$. Notice that the equations for the fast part (2.11) describe a single scale Lorenz systems where the presence of slow variables results in a shift of the Rayleight number R_f, namely

$$R_f \mapsto R_f + \frac{\epsilon_f}{c}\, Y, \tag{2.18}$$

which depends on the $Y \in U$ variable alone. Therefore, since adiabatic averages only depend on the variable Y, our problem is greatly simplified. Moreover, we can observe that, in the standard Lorenz system,

$$\langle z\rangle_R \approx (R-1)\,\Theta(R-1). \tag{2.19}$$

This relation is exact if $R < R_\star \approx 24.74$ since, in this case, there exists two stable fixed points with that $z-$coordinate, i.e. $(x_\star, y_\star, z_\star) = (\pm\sqrt{b\,(R-1)}, \pm\sqrt{b\,(R-1)}, R-1)$. Remarkably, this is still a good approximation for R above the critical value $R_\star$ (see fig. 2.3) and, more precisely, a direct numerical computation gives:

$$\langle z\rangle_R \approx \begin{cases} 0 & R < 1 \\ R-1 & 1 \le R \lesssim 24.74 \\ 0.976R - 3.614 & R \gtrsim 24.74. \end{cases} \tag{2.20}$$

Hence, we can write an approximation from eqs. (2.19), (2.17) and (2.18)

$$\langle xy\rangle_Y = b(R_f - 1 + (\epsilon_f/c)Y)\Theta(R_f - 1 + (\epsilon_f/c)Y)\,. \tag{2.21}$$

which can be refined numerically as

$$\langle xy\rangle_Y = (107.5+26.04Y/c)\,\Theta((107.5+26.04Y/c) \tag{2.22}$$

which is the model we are going to use to replace the interaction term in eq. (2.10).

Now that we have presented an effective model, we should ask whether it is good or not. Indeed, if we compare adiabatic and conditional averages (see fig. 2.4) we can realize that they overlap for

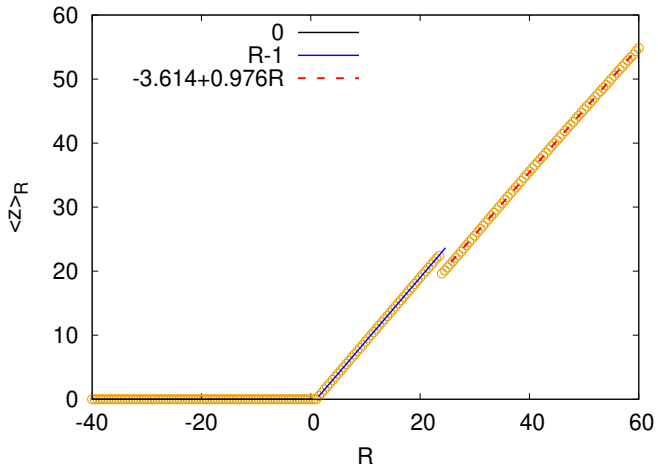

Figure 2.3: The expected value of z as a functin of R in the one-scale Lorenz model.

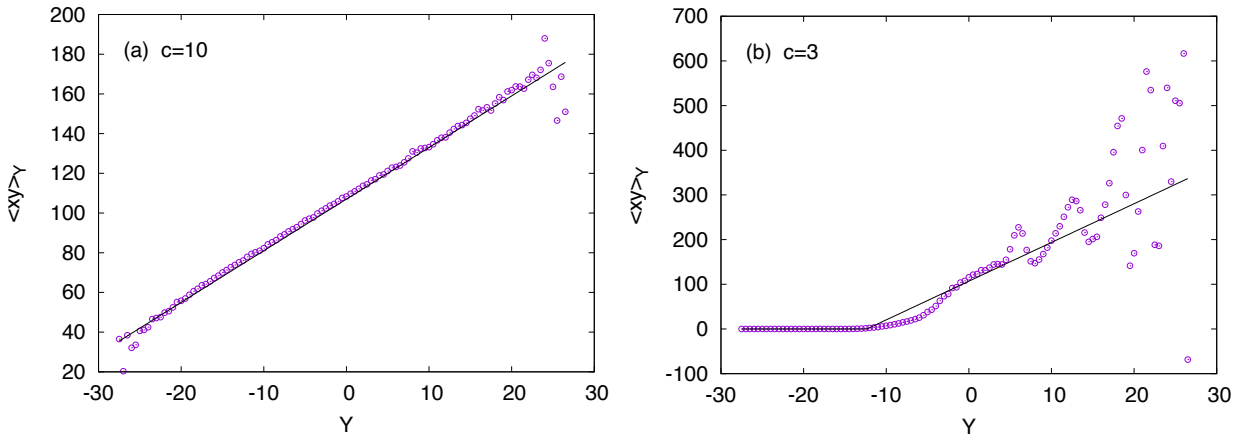

Figure 2.4: $\langle x\, y\rangle$: adiabatic prediction (2.22) (black line) vs empirical conditional average with respect to Y (purple circles) in cases $c = 10$ (panel (a)) and $c = 3$ (panel (b)).

$c = 10$ but not for $c = 3$. As a consequence, we can trust this approximation only in the strong separation case, but not the other, as we expect. As a more reliable test, we can compute the FSLE of the adiabatic model with respect to the original system. This procedure consists in computing the FSLE by comparing a reference trajectory obtained with the correct equations and a control trajectory obtained with adiabatic equations with the same initial condition. By looking at fig. 2.5, we can realize that, for strong scale separation, above the scale crossover, i.e. in the slow regime, the error expands with the same FSLE as the exact equations. This means that this approximation is excellent, but only if the error on the initial condition is large enough to fall above the crossover threshold. This is not surprising: all fast effects are averaged out in this approximation. For the sake of completeness, in fig. 2.5 we also introduce the FSLE of what we call *truncated model* in which we simply drop fast variables, i.e. we use the replacement $x\, y \mapsto 0$ in (2.10). The truncated model clearly performs far worse that the adiabatic one, meaning that the fast subsystem is relevant and cannot simply be ignored. As already stated, this is a quite generic feature: when fast and slow degrees of freedom are non-linearly coupled, a poor modelling of the fast scales results in a poor modelling of the slow ones as well.

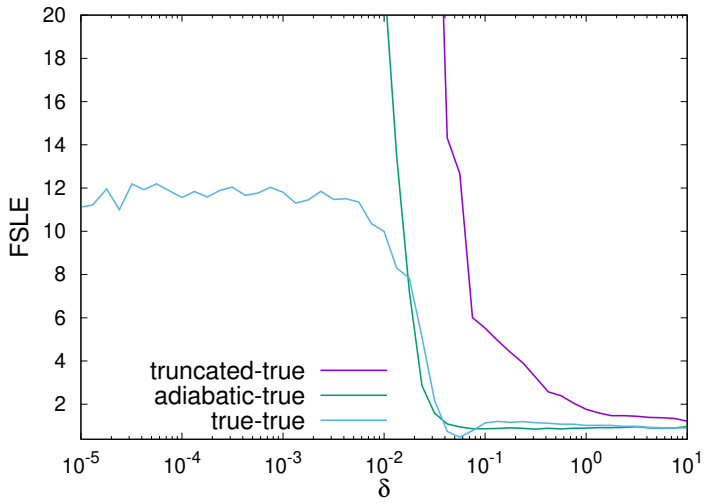

Figure 2.5: FSLE on the slow variables from the original system ("true") with respect to the following models: the original system, the truncated model and the adiabatic model. Notice the scale crossover, as the FSLE true-true drops around 10^{-2}. The true-adiabatic FSLE is similar to the true-true one above the crossover, but becomes way larger below: any small error below that scale would expand very rapidly until such threshold is reached. The truncated model fails at all scales, but is somehow better above the crossover.

2.1 Effective slow dynamics through echo state neural networks

2.1.1 Applying ESRNN to the slow part of the two-scale Lorenz system: definitions and setting

In this section, we will briefly outline how to apply an ESRNN to the slow variables of the two-scale Lorenz system (2.11) (2.10) with either $c = 3$ or $c = 10$.

Training setting

We are interested in building a predictive model for variables U alone, assuming the fast part u cannot be simulated. We generate some long training trajectories $\{(u(t), U(t))\}_{t=-T}^{0}$ and we drop the fast variables from it so that we are left with the training set $\{U(t)\}_{t=-T}^{0}$. Now we should choose a sampling time[6]: we either pick $\Delta t = 10^{-1}$ or $\Delta t = 10^{-2}$. Notice that the first choice is approximately the characteristic timescale of the fast dynamics. We can proceed with a temporal sampling and build a discrete-time dataset $\Omega = \{(U(n\,\Delta t))\}_{n=-N_T}^{0}$ with $N_T = T/\Delta t$. For simplicity, we define

$$\mathcal{U}_n = U(n\,\Delta t). \tag{2.23}$$

and, form now on, will indifferently use the U or $\mathcal{U}$ notation.

In building an ESRNN, we mostly follow the works by Pathak et al. [159, 191, 192] and further details on the implementation are provided in Appendix 2.A: the reader can find the list of the hyperparameters in tab. 2.2 and the network architecture in sec. 1.4, specifically eqs. (1.24) and (1.25) but with no bias. Note that the choice of the function $\hat{R}$ as eq. (1.28) is appropriate in this context, since it respects the symmetries of the system, as suggested in ref. [159, 192]. We can optimize the network with the training set Ω, by following the procedure described in sec. 1.4.2. Since we

[6] We could also have used continuous time but, for simplicity, we employ time sampling.

want to compute statistical properties for our study, we need to make forecasts from multiple initial conditions (see fig. 1.4 and the associated discussion). For this purpose, we need to build additional test sets[7] of the form $\Omega_{test} = \{U(n\,\Delta t)\}_{n=-N_s}^{N_p}$: the first N_s data points are used to synchronize the reservoir in open loop mode and reach the echo state at timestep $n = 0$, while the latter segment of N_p steps is used as a reference trajectory for evaluating the performance of the network in closed loop setting (cfr fig. 1.3).

Open loop and synchronization error

Before discussing the results, we introduce both our terminology and illustrate what happens in the typical case after training the network.

Consider the test set $\Omega_{test} = \{U(n\,\Delta t)\}_{n=-N_s}^{N_p}$, which we need to make forecasts with initial condition $U_\star = U_0 \in \Omega_{test}$. We randomly initialize the reservoir with initial condition $r_\star$ and then start to feed $\{U(n\,\Delta t)\}_{n=-N_s}^{0} \subset \Omega_{test}$ in open loop. To check if synchronization is proceeding well, we plot the one-step forecast error as a function of the time step n (see fig. 2.6, gray shaded area). We define the log-error as

$$E_n = \log(\|\hat{\mathcal{U}}_n - \mathcal{U}_n\|), \tag{2.24}$$

where $\hat{\mathcal{U}}_n = W^{out} R_n$ is the forecast. The error drops exponentially in time as the reservoir synchronizes, until it reaches a plateau, whose value can be called *synchronization error* E_S. A possible way to define E_S is

$$E_S = \lim_{N_s \to \infty} \frac{1}{N_s} \sum_{n=1}^{N_s} \log(\|\hat{\mathcal{U}}_n(r_\star, \mathcal{U}_{0:n-1}) - \mathcal{U}_n\|), \tag{2.25}$$

where $\hat{\mathcal{U}}_n(r_\star, \mathcal{U}_{0:n-1})$ is the n^{th} forecast after using the $\mathcal{U}_{0:n-1}$ sequence in open loop, and using reservoir initial condition $r_\star$. Provided that the echo state property holds, the limit should be well defined.

The synchronization error has a twofold interpretation. It is the minimal average log-error on forecast that can be obtained by synchronizing the reservoir. In this sense, it is the average log-error on the equations of motion measured on the attractor. However, as already stated in the general case, a small error on the equations is not a sufficient condition for structural stability of the dynamics in closed loop (fig. 1.3). It is not hard to build networks with very small E_S which fail catastrophically in closed loop mode. As a second interpretation, E_S can also be viewed as the minimum average log-error on initial conditions (technically, after one-step). Since, in chaotic systems, the error on the initial condition sets a statistical upper bound to the time elapsing before the predicteds trajectory and its reference totally de-correlate. In case there is a single Lyapunov exponent, the decorrelation time reads:

$$T_{decorr} \approx \frac{1}{\lambda} \log(\sqrt{2\,C, \langle U \cdot U - \langle U \rangle \cdot \langle U \rangle \rangle}/\delta), \tag{2.26}$$

where λ is the maximum Lyapunov exponent and $C \leq 1$, $C = O(1)$ fixes an arbitrary threshold; $\delta \approx e^{E_S}$ is the error on the initial condition; $2\,\langle U \cdot U - \langle U \rangle \cdot \langle U \rangle \rangle$ is the average square distance between two uncorreleated trajectories. Estimate (2.26) is clearly overly pessimistic in multiscale

[7] This is an abuse of language, since we are referring to a single trajectory as a test set. The actual test set is composed of 10000 of such trajectories.

scenario, since it assumes the largest Lyapunov exponent dominates at all scales. In our two scale scenario, if δ is larger than the crossover scale, we may write

$$T_{decorr} \approx \frac{1}{FSLE_{slow}} \log(\sqrt{2\,C\,\langle U \cdot U - \langle U\rangle \cdot \langle U\rangle\rangle}/\delta). \tag{2.27}$$

Either ways, the synchronization error is an important quantity and we will elaborate further on it.

Closed loop

After reaching the echo state, we can close the loop and attempt long term forecasts (see eqs. (1.24) (1.25) and fig. 1.3). As can be seen in fig. 2.6, after closing the loop, the log-error E_n starts increasing again until it saturates once the predicted trajectory is fully decorrelated from the reference one. This is the most delicate part and, since the network is, strictly speaking, optimized in open loop mode, there is no guarantee that it will succeed, and it may fail in several ways instead. Among the most common issues, the predicted trajectory collapses on either a spurious periodic orbit or fixed point. A necessary test consists in checking that the average error asymptotically reaches $2\langle U \cdot U - \langle U\rangle \cdot \langle U\rangle\rangle$, but it is far from a safe quality indicator. A more sound approach is to directly test the quality of the predictions. In fig. 2.6, we have reported a successful training case: the error expands with the slow FSLE (cfr. fig. 2.5) as if we were using the exact equations. In the following section, we will explore this topic in greater depth by performing averages over many training trajectories in order to evaluate the predictability in statistical terms.

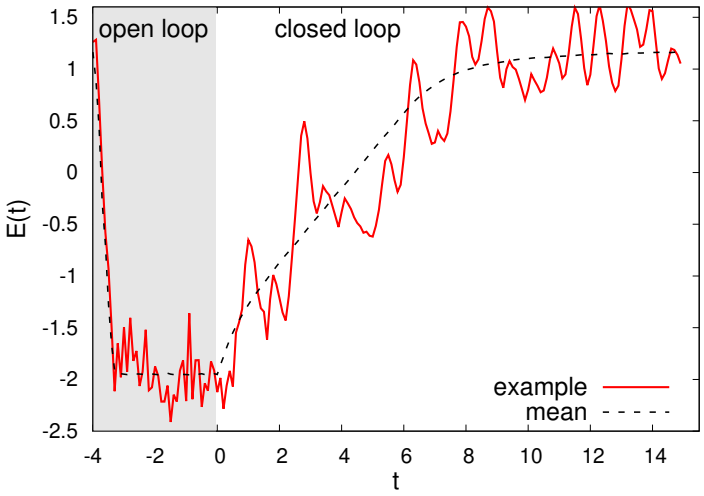

Figure 2.6: Typical example of prediction with a successfully trained ESRNN. From time -4 to 0 (sampling time $\Delta t = 0.1$, $c = 10$) the networks runs in open loop: the log_{10} error $E(t)$ (in the text, for simplicity, we used the natural logarithm instead) drops until it reaches a plateau (synchronization, i.e. echo state, is reached). After closing the loop, it expands again until it saturates. Notice that the linear expansion in log scale should be interpreted as finite-size chaotic behaviour. The red curve shows an example of the $log_{10}-$error trajectory, while the black line is the average over 10000 initial conditions. Note that the exponential expansion is compatible with the slow FSLE from fig. 2.6. Other relevant parameters are in the figure legends.

2.1.2 Comparison between the network model and other models.

In order to properly evaluate the performance of the network, we should compare it with other models. In order to make a fair comparison, though, all models should be initialized with the same initial conditions. This can indeed be done by taking as initial condition the first forecast $\hat{\mathcal{U}}_1$ after synchronization (i.e. at the end of the open loop regime). Such initial condition is characterized by an average log-error equal to E_S.

Models can be divided into two classes: models with or without error on equations, i.e. imperfect and perfect models, respectively. Imperfect models are characterized by some inexact modelling of the fast-to-slow interaction term $x\,y$ in (2.10). Besides the network itself, which pertains to this class, we choose the adiabatic model given by $xy \mapsto \langle xy \rangle_s$ and the truncated model, given by $xy \mapsto 0$ (see sec. 2.0.2 for both). For completeness sake, we also introduce the *average model*, where the interaction term is replaced by its global average $x\,y \mapsto \langle x\,y \rangle$.

It is also important to compare the network predictive performance with that of the exact equations – perfect models with errors on the initial condition. There is an ambiguity here, though. The first prediction after the open loop phase only includes slow variables and does not provide any information on how to initialize the fast variables, since they are not part of the neural network-based modelling at all. Since we have no assumption-free way to choose them, we use the two extreme options. We define the *twin model* as the model with exact equations and *exact fast variables* (i.e. those of the reference trajectory) *in the initial condition* (this corresponds to the best one can do). We define the *random twin model* as the model with exact equations and *random fast variables in the initial condition*: they are drawn from the stationary distribution associated to the fast dynamics in eq. (2.11) with fixed slow variables $U = \mathcal{U}_1$. Table 2.1 summarizes the models we are going to compare.

	Name	$x\,y$ modelling	fast scale initial condition
Inexact equations	ESRNN	?	echo-state?
	adiabatic model	$\langle x\,y \rangle_Y$	none
	mean model	$\langle x\,y \rangle$	none
	truncated model	0	none
Exact equations	twin model	exact	exact
	random model	exact	random

Table 2.1: Summary of models for two-scale Lorenz equations (2.10).

In fig. 2.7, we show the average log-error for different models in close loop mode for $c = 3, 10$ and $\Delta t = 10^{-1}, 10^{-2}$. In fig 2.8, some examples of predicted vs reference trajectories are shown. In all cases, the network outperforms imperfect models and the random twin model, which is a remarkable demonstration that the network is very accurate in reconstructing the dynamics. Moreover, the network does worse than the twin model in all but in the $c = 10,\ \Delta t = 10^{-1}$ case. This last exception may look surprising but it can be easily rationalized without assuming the neural network is doing anything special, and will be briefly discussed in the subsection about the scale crossover.

Finally, observe that, in the strong scale-separation case $c = 10$, the performance of the adiabatic model is comparable to that of the network while in the $c = 3$ case it is not: this matches our priors. But can we learn anything deeper from this comparison? In the following section, we try to explain

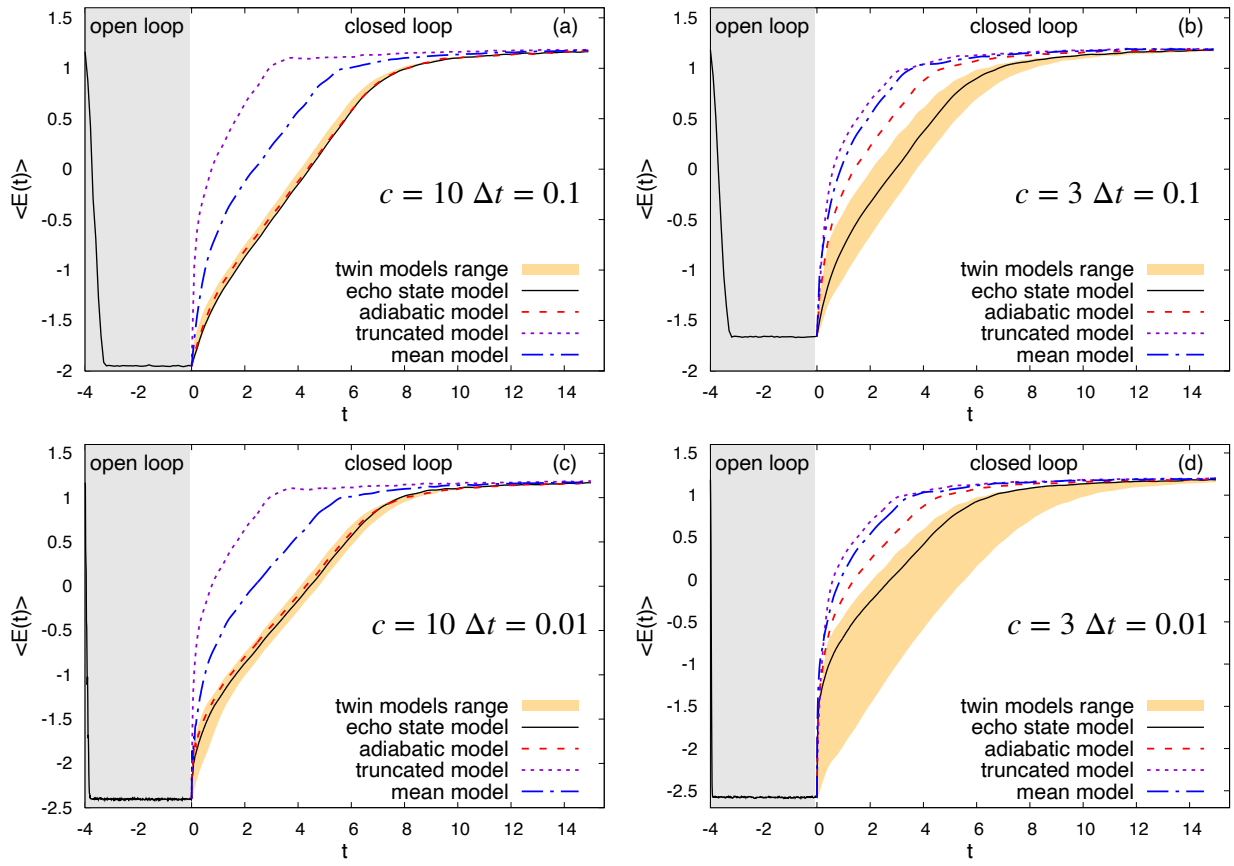

Figure 2.7: The panels show the evolution of the average log-error $E(t)$ as a function of time for different parameter choices. The shaded area represents the closed loop or synchronization phase with $d_r = 500$. Averages have been computed over 10000 initial conditions.

the functioning of the network in terms of the adiabatic model.

2.1.3 Relation between ESRNN and adiabatic model

When the scale separation is large (c=10), the adiabatic and the network-based models have a very similar performance: it is therefore natural to wonder how they are related.

While we may reasonably expect that the network can reconstruct the slow part of the dynamics accurately, the main unknown is how the fast oscillations are handled. Let us start by remembering that the fast-to-slow interaction term only appears in the Y derivative and that it can be written as

$$\dot{Y} - \dot{Y}_T = \epsilon_s \, x \, y \tag{2.28}$$

where $\dot{Y}_T = R_s \, X - Z \, X - Y$ is, by construction, the derivative of Y in the truncated model. However, it is true in general that, for any model that correctly reconstructs the slow part, the fast-to-slow interaction term can be computed as $\dot{Y}_{model} - \dot{Y}_T$; for instance $\dot{Y}_{adiabatic} - \dot{Y}_T = \epsilon_s \, \langle x \, y \rangle_Y$. Unfortunately, we cannot compute this quantity directly for the network-based model, since derivatives are not computed explicitly by the ESRNN because time is discreet. However, we can choose a small sampling time $\Delta t = 10^{-2}$ such that derivatives are well approximated by finite differences.

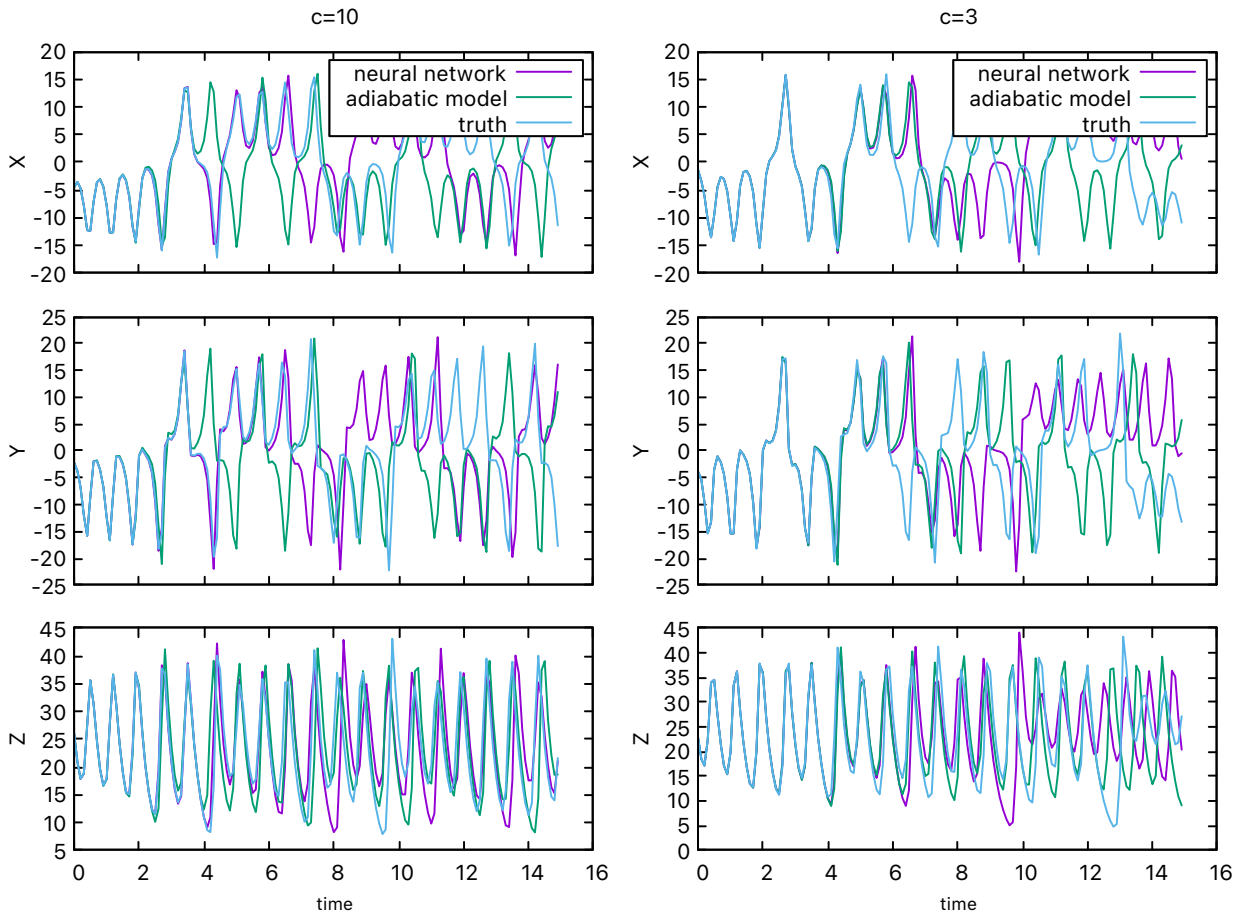

Figure 2.8: Slow-variable trajectories: reference, adiabatic forecast and ESRNN-forecast in closed loop mode.

Then, we define

$$\Delta\widetilde{\dot{Y}}_{model} = \frac{Y(t+\Delta t) - Y(t)}{\Delta t} - \frac{Y_T(t+\Delta t) - Y(t)}{\Delta t} = \frac{Y(t+\Delta t) - Y_T(t+\Delta t)}{\Delta t} \tag{2.29}$$

as a surrogate observable for $\dot{Y}_{model} - \dot{Y}_T$. The quantity $\Delta\widetilde{\dot{Y}}_{model}$ can be computed along closed loop trajectories generated by the network. In fig. 2.9, we plot $\Delta\widetilde{\dot{Y}}_{model}$ as a function of Y for *model* =adiabatic, network, two-scale Lorenz. We observe the wide fluctuations of $\Delta\widetilde{\dot{Y}}_{Lorenz}$ in the case of the original system, while a straight line represents $\Delta\widetilde{\dot{Y}}_{adiabatic}$ in the case of the adiabatic model. Remarkably, the estimated network interaction term $\Delta\widetilde{\dot{Y}}_{net}$ closely resembles the adiabatic one, implying that, at least for this choice of parameters ($c = 10,\ \Delta t = 10^{-2}$), the network model can be interpreted as an accurate adiabatic one. We tried to extend the analysis to the $c = 3$, but results were inconsistent and, moreover, with such a weak scale separation, we have no legitimate argument for assuming that the interaction term should depend on Y alone (on the contrary, we can argue it should not).

Since the network still performs well in the $c = 3$ case, it is a more general and versatile technique than the adiabatic approximation..

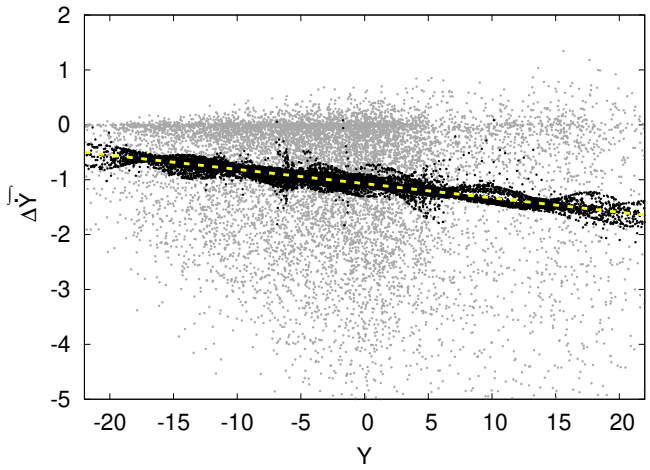

Figure 2.9: We show an empirical scatter plot $(Y, \Delta\widetilde{Y}_{model})$ for the exact model (grey dots), the adiabatic (yellow line) and network (black dots) in the case $\Delta t = 10^{-2}$ and $c = 10$.

2.1.4 Hybrid echo-state neural network

In this section, we will analyze the application of the hybrid framework [194, 266] (see sec. 1.4.2 and Appendix 2.B) to the multiscale chaotic setting. It consists in assuming that the network has access to imperfect forecasts $\tilde{U}((n+1)\,\Delta t))$ and, consistently, we augment the input vector as

$$U(n\,\Delta t) \mapsto (U(n\,\Delta t), \tilde{U}((n+1)\,\Delta t)) \tag{2.30}$$

where $\tilde{U}((n+1)\,\Delta t))$ is an available forecast, which we choose to get from the truncated model, the worse among the available ones. Hence, the closed loop equations are

$$\begin{cases} r_n = \tanh\left(\begin{bmatrix} W & W^{in} & \tilde{W}^{in} \end{bmatrix} \begin{bmatrix} r_{n-1} \\ U((n-1)\,\Delta t) \\ \tilde{U}(n\,\Delta t) \end{bmatrix}\right) \\ U(n\,\Delta t) = W^{out}\, R_n \\ \tilde{U}(n\,\Delta t) = U((n-1)\,\Delta t) + \displaystyle\int_{(n-1)\,\Delta t}^{n\,\Delta t} dt'\; F_{\text{truncated}}(U(t')) \end{cases} \tag{2.31}$$

where $F_{\text{truncated}}$ represents the truncated model equations and $\tilde{W}^{in}$ is an additional input matrix.

In the following, we are interested in understanding whether the hybrid formulation can enhance the performance of the basic scheme (from now on, called *reservoir-only* as in [194]). For this purpose, we need a metric to make an assumption-free assessment of the quality of long term forecasts of the network. Following ref. [194], we use the *predictability time* T_{pred}, defined as the average first time for which the closed loop prediction error exceeds an arbitrary threshold $\Delta = 0.4\,\sqrt{\langle U \cdot U \rangle}$, which approximately represents the size at which the slow exponential regime ends in fig. 2.1. Namely:

$$T_{pred} = \Delta T\; \langle \min\{n \text{ such that } \|\hat{\mathcal{U}}_n - \mathcal{U}_n\| \geq \Delta\} \rangle. \tag{2.32}$$

where the brackets $\langle \cdot \rangle$ denote the average over initial conditions and the beginning of the closed loop forecasting correspond to $n = 0$ (after the usual synchronization sequence).

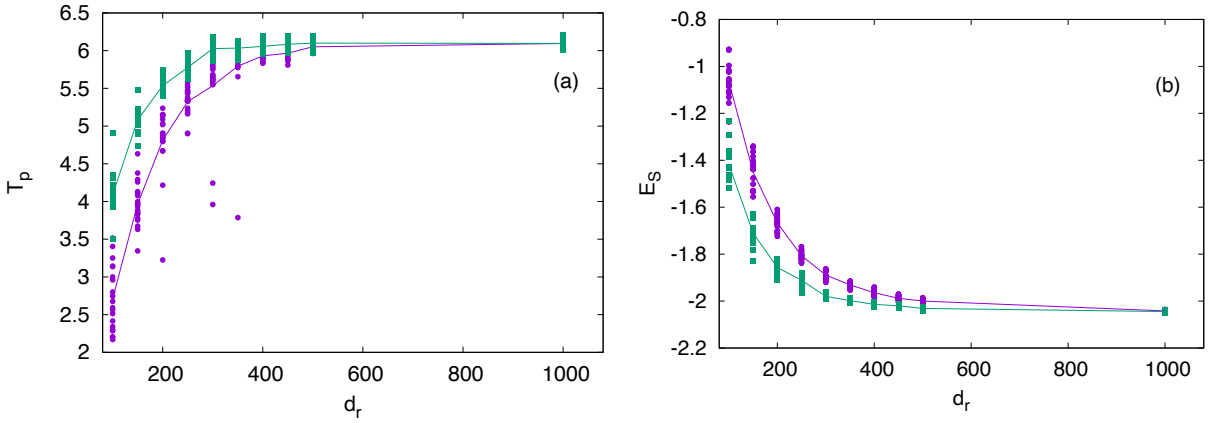

Figure 2.10: Case $c = 10$, $\Delta t = 0.1$. Panel (a). Predictability time in the case of the hybrid (green) and reservoir only (purple) schemes. Panel (b) Synchronization error E_S with same color coding. In both cases, the dots represent individual experiments, while the curve represents the average.

So far, we have never mentioned a fundamental ingredient in our technique: the size of the network d_r. As one may expect intuitively, larger networks tend to outperform smaller ones. Therefore, any reservoir-only vs hybrid comparison should be done as a function of the reservoir size. In ref. [194] it was found that, in the single scale Lorenz system, the hybrid scheme is advantageous at any size. In the case study $\Delta t = 10^{-1}$, $c = 10$, our multi-scale results show a more nuanced picture, as can be deduced by looking at fig 2.10(a). Indeed, both in reservoir-only and hybrid schemes, T_{pred} is a non-decreasing function of the size d_r, and the performance of the latter implementation is always greater or equal than that of the former one; however, in both cases, T_{pred} reaches a plateau at the smae value for large sizes ($d_r \gtrsim 500$). This can be either formulated as a positive or a negative result. One could say that, in general, the hybrid scheme allows to achieve the same performance as the reservoir-only but with a smaller size. However, one could also say that that, if the network is large enough, the hybrid scheme does not boost the performance. This result clearly opens some questions and an explanation for these observations will be provided in the next section.

Finally, observe fig. 2.10(a): in the hybrid scheme, the predictability times associated to different network realizations are close to the mean value, while there are some outliers characterized by smaller predictability times in the reservoir-only case. These anomalous values should be interpreted as failed realizations of the network (remember that the reservoir is randomly assembled and no tuning is performed on the internal links). Hence, the hybrid scheme is not just better on average, but also more robust against failures. As a final remark, note that such failures cannot be detected by merely looking at the synchronization error in fig. 2.10(b), which turns out to be an unreliable metric in evaluating the performance of the network, providing no more of an upper bound to the performance rather than carrying any real predictive power in this sense.

2.1.5 The problem of the scale crossover

The last point of the previous subsection is worth some discussion. Why does T_{pred} reach a plateau in d_r? As we have already mentioned, the time elapsing before decorrelation of predicted and reference trajectories (see eq. (2.26)) is statistically bound by the error on the initial condition. Therefore, a lower bound on the error on the initial condition – the synchronization error E_S – would put an upper bound on the predictability time T_{pred}, which is exactly the case, as shown in fig. 2.10(b).

However, the converse is not true, since, as it will be expanded below, a low synchronization error E_S is a poor predictor of a high predictability time T_p.

We should now try to understand why the synchronization error does not go lower than a certain value. This last point can be readily answered with physics. Indeed, the minimum of the synchronization error is about $e^{E_S} \approx 10^{-2}$ which correspond to the scale crossover (in the $c = 10$ case, see fig. 1.20). Therefore, the natural conclusion is that the network failed to synchronize with the fast oscillations, which have been accounted for in an effective way. Note that this is perfectly consistent with the notion that the neural network operates as an adiabatic model in the strong scale separation case. Indeed, unresolved fast variables behave as noise with short correlations, whose effective amplitude is related to the scale crossover[8]. Therefore, if the error on the initial condition is lower than the threshold, it will not expand exponentially, but "jump" to the threshold as if driven by white noise (since Δt is finite, steep but continuous expansion may not be observed). Note that this is consistent with the FSLE fig. 2.5.

This picture also explains why the neural network could slightly outperform the twin model in the case of strong scale separation. Both the twin and network are initialized with an error whose size is compatible with the crossover. Hence, the error on the slow part of the dynamics forces almost instant decorrelation on the exact fast variables in the twin model, nullifying its core advantage. From that point onward, fast oscillations of the model add up to those of the reference, effectively increasing the variance of the noise[9]. The network model, on the other hand, is subject only to the noise of the reference from the beginning.

Note that, in order to push the error below the crossover scale, synchronization is not enough. Indeed, by looking at fig. 2.7(c), one may realize that, in the $c = 10$, $\Delta t = 10^{-2}$ case, the synchronization error e^{E_S} is lower than 10^{-2}, implying that part of the fast dynamics may be synchronized in open loop. However, the error immediately jumps to the threshold as soon as we close the loop. Indeed, in order to make predictions below the crossover, not only correct synchronization, but also accurate reconstruction below the threshold is needed, which was unsuccessful, as we can deduce from the figure.

As a final remark, we point out that, in spite of their efficiency, of ESRNN are not necessarily the most flexible machine learning techniques. For this reason, the behaviour of the network, e.g. the similarity with the adiabatic model, may not necessarily be reproduced by different neural networks, say an LSTM, which could, in principle, have a better performance. In our very specific example, it is likely that, by using a different technique (e.g. deep ESRNN or LSTM) we could have been able to reconstruct the fast dynamics as well, since it involves only 3 degrees of freedom, after all (see [240], for instance). Note that, however, the case we have been focusing on – residual non-resolvable degrees of freedom – is very relevant in real world scenarios: in general, the fast sub-dynamics is very hard to describe and cannot fully be accounted for.

[8] This relation is more complicated that it may appear from this minimal description, and it would be possible to give a way more satisfactory description, but, regrettably, it is beyond the scope of this work and possibly of this thesis.

[9] As a minimal sketch, say the reference trajectory evolves with effective stochastic equation $dU = f(U)\,dt + \sqrt{2\,D}\,d\xi$ where ξ is white noise, representing fast variables. Even with exact initial conditions, a model with no noise, say $\dot{U} = f(U)$, would be more accurate, in the short term, than a perfect model, since forecasts point in the average direction but with less variance. This is true as long as we use the Euclidean distance between the reference and the forecast as a metric, but would be false if, for instance, we were investigating statistical properties.

2.1.6 Conclusions

We have studied, in a controlled case study, how an ESRNN behaves when trying to learn the slow dynamics in multi-scale chaotic system. We found out that the slow sub-dynamics can be well reconstructed: when scale separation is strong ($c = 10$), the technique is neat in building an adiabatic model. Despite checking, we could not provide an equivalently clear picture for the $c = 3$ case but that most likely involves some averaging of fast degrees of freedom, which may be guessed by considering how the technique is constructed. In this sense, ESRNNs turn out to be an effective and automatic coarse graining technique. Finally, in the hybrid scheme, the reservoir computing approach can be improved by blending it with an imperfect predictor, boosting the performance of smaller reservoirs. While we have obtained these results with a relatively simple two-time scale model, given the success of previous applications to spatially extended systems [191], we think the method should work also with more complex high dimensional multiscale systems. In the latter, it may be necessary to consider multi reservoir architectures [48] in parallel [191] (see sec. 1.4.2). Moreover, reservoir computing can be used to directly predict unobserved degrees of freedom [160]. Using this scheme and the ideas developed in this work it would be interesting to explore the possibility to build novel subgrid schemes for turbulent flows [169, 222] (see also [266] for a very recent attempt in this direction based on reservoir computing with hybrid implementation), preliminary tests could be performed in shell models for turbulence for which physics-only approaches have been only partially successful [30].

Appendix 2.A Details on the implementation

Intra-reservoir (R-to-R) connectivity matrix W

The intra-reservoir connectivity matrix, W, is generated by drawing each entry from the same distribution. Each element is the product of two random variables $W^0_{ij} = a * b$: a being a real uniformly distributed random number in $[-1,\ 1]$ and b taking values 1 or 0 with probability $P_d = d/d_R$ and $1 - P_d$, respectively. Consequently, each row has, on average, d non zero elements. Since $d_r \gg d$, the number of non null entries per row is essentially distributed according to a Poisson distribution. As a last step, the maximal eigenvalue (in absolute value), $\rho_{\max}(W)$ of the resulting matrix W is computed and the matrix is rescaled element wise so that its new spectral radius matches the target value ρ, i.e.:

$$W_R = W^0 \frac{\rho}{\rho_{\max}(W^0)}$$

Input-to-reservoir (I-to-R) connectivity matrix W_{in}

The input to reservoir matrix W_{in} is generated in such a way that each reservoir node is connected to a single input. For this purpose, for each row j, a single element n_j, uniformly chosen between 1 and the input dimension D_I, is different from zero. This means that the reservoir node j is only connected to the n_j^{th} input node. The connection strength is randomly chosen in $[-\sigma, \sigma]$ with uniform distribution.

Optimization

The optimization is described in sec. 1.4.2.

Synchronization time and length of the training input trajectory

All results presented in this article have been obtained using training trajectories of length $T_t = 500$. We remark that using values $100 \leq T_t \leq 1000$ one can hardly notice qualitative differences. At low training times, failures can be very diverse, ranging from tilted attractors to periodic orbits or spurious fixed points. The chosen values of T_t have been tested to be in the range that guarantees long term reconstruction of the attractor with proper hyperparameters. As for the the synchronizing length, we have chosen $T_s = 4$. Such value is about four times larger than the time actually needed to achieve best possible synchronization indeed, as shown in the gray shaded areas of figs. 2.7, the error E saturates to E_S in about a time unit.

Numerical details

The whole code has been implemented in *python3*, with linear algebra performed via *numpy*. Numerical integration of the coupled Lorenz model were performed via a 4^{th} order Runge Kutta scheme.

Fixing the hyperparameters

The architecture of a generic network is described by a number of parameters, often dubbed hyperparameters, e.g.: the number of layers, activation functions etc. While a proper design is always

crucial, in the reservoir computing paradigm, this issue is especially critical due to the absence of global optimization via backpropagation. The reservoir-to-reservoir and input-to-reservoir connectivity matrices, as discussed above, are quenched stochastic variables, whose distribution depends on four hyperparameters:

$$\text{Net} \sim P(\sigma, \rho, d, d_r)\,,$$

namely, the strength of the I-to-R connection matrix σ, the spectral radius ρ of the R-to-R connection matrix, the degree d of the R-to-R connection graph, and the reservoir size d_r. Once the distribution is chosen, there are two separate issues.

The first is that, for a given choice of (σ, ρ, d), the network should be self-averaging if its size d_r is large enough. Indeed, we see from fig. 2.10 that the variability between realizations decreases with d_r, as expected.

The second issue is the choice of the triple (σ, ρ, d). In general, the existence of large and nearly flat (for any reasonable performance metrics) region of suitable hyperparameters implies the robustness of the method. As for the problem we have presented, such region exists, even though, in the case $\Delta t = 0.1$, $c = 10$, moderate fine tuning of the hyperparameters did improve the final result, allowing to even (moderately) outperform the fully informed twin model, as shown in fig. 2.7a.

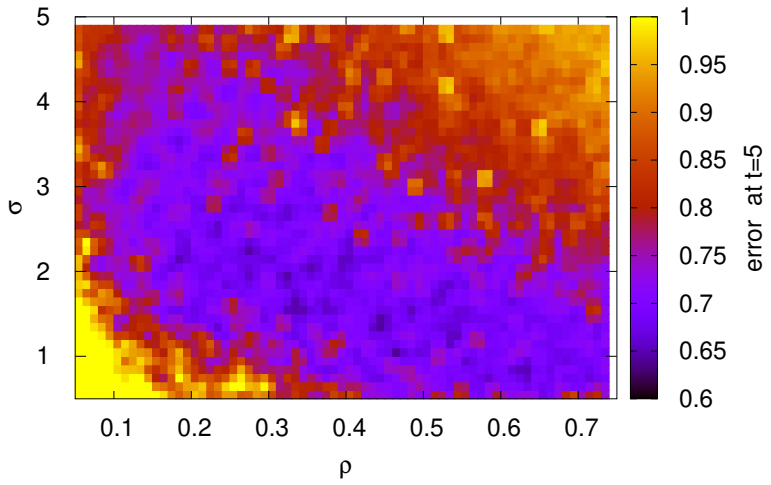

Figure 2.11: Performance grid for $c = 3$, $\Delta t = 0.1$, $d_r = 350$, $d = 5$. Colors code error between forecast and reference trajectories at time $t = 5$ after closing the loop, which if the metrics here used $f = \|U^{forecast}(t = 5) - U^{true}(t = 5)\|$ (averaged over 100 points of the attractor) for a single realization of the network for a given value of parameters (ρ, σ). To highlight the suitable parameter region, a cutoff on has been put at $f = 1$.

It is important to remark that the characteristics of the regions of suitable hyperparameters depend on the used metric. Here, we have focused on medium term predictability, i.e. we evaluate the error between forecasted and reference slow variables at a time (after synchronization) that is much larger than one step Δt but before error saturation (corresponding to trajectories completely uncorrelated). Requiring a too short time predictability, as discussed in ref. [159], typically is not enough for reproducing long time statistical properties of the target system (i.e. the so called climate), as the learned attractor may be unstable even if the dynamical equations are locally well approximated. If both short term predictability and climate reproduction are required, the suitable hyperparameter region typically shrinks. The metric we used typically led to both predictability

and climate reproduction, at least for reservoir sizes large enough.

In order to fix the parameters, two techniques have been employed. The first is the standard search on a grid (for a representative example see fig. 2.11): a lattice is generated in the space of parameters, then each node is evaluated according to some cost function metrics. If such function is regular enough, it should be possible to detect a suitable region of parameters. While this default method is sound, it may require to train many independent networks, even in poorly performing regions. Each network cannot be too small for two reasons: the first is that small networks suffer from higher inter realization fluctuations, the second is that we cannot exclude that optimal (σ, ρ, d) have a loose dependence on the reservoir size d_r. As further discussed below we found a mild dependence on the network degree d, provided it is not too large, thus in Fig.2.11 we focused on the dependence on ρ and σ.

The second technique is the no gradient optimization method known as particle swarming optimization (PSO) [137]. PSO consists in generating n (we used $n = 10$) tuples of candidate – the particles – parameters, say $p_i = (\rho_i, \sigma_i, d_i)$ $i = 1, ..., n$. At each step, each candidate is tested with a given metrics f. Here, we used the average (over $50 - 100$ initial conditions) error on the slow variables after $t = 2, 4, 5$ (depending on the parameters) in the close loop configuration. Then, at each iteration k of the algorithm, each candidate is accelerated towards a stochastic mixture of its own best performing past position

$$p_i^*(k) = \arg\min_{p_i(s)}\{f(p_i(s))|s < k\}$$

and the overall best past performer

$$p^*(k) = \arg\min_{p_i^*(k)}\{f(p_i^*(k))|i = 1, ..., n\}.$$

Particles are evolved with the following second order time discrete dynamics

$$\begin{aligned} p_i(k+1) &= p_i(k) + v_i(k) \\ v_i(k+1) &= w v_i(k) + \phi_i^1(k)\,(p_i^*(k) - p_i(k)) \\ &+ \phi_i^2(k)\,(p^*(k) - p_i(k)) \end{aligned}$$

with $\phi_i^j(k) \in [0, 1]$ being random variables and $w \in [0, 1]$ representing a form of inertia, as implemented in the python library *pyswarms*. After a suitable amount of iterations, p^* should be a valid candidate. The advantage of PSO is that, after a transient, most candidate evaluations (each of which require to initialize, train and test at least one network) should happen in the good regions. It is worth pointing out that, unless self-averaging is achieved thanks to large enough reservoir sizes, inter network variability adds noise to limited attractor sampling when evaluating f and, therefore, fluctuations may appear and trap the algorithm in suboptimal regions for some time. Moreover, the algorithm itself depends on some hyperparameters that may have to be optimized themselves by hand.

In our study, PSO has been mainly useful in fixing parameters in the ($\Delta t = 0.1$, $c = 10$) case and to observe that d is the parameter which affects the performance the least. Some gridding (especially in ρ and σ) around the optimal solution is useful, in general, as a cross check and to

highlight the robustness (or lack thereof) of the solution.

In Table 2.2 we summarize the hyperparameters used in our study.

	$\Delta t = 0.1$	$\Delta t = 0.01$
c=3	$d = 5$ $\sigma = 2$ $\rho = 0.35$	$d = 5$ $\sigma = 2.5$ $\rho = 0.25$
c=10	$d = 5$ $\sigma = 1.8$ $\rho = 0.34$	$d = 5$ $\sigma = 0.8$ $\rho = 0.68$

Table 2.2: (Color online) Hyperparameters used in the simulations: Δt is the sampling time, c is the time scale separation of the multiscale scale Lorenz model Eqs. (2.10-2.11), σ is the input-to-reservoir coupling strength, ρ is spectral radius of the reservoir-to-reservoir connectivity matrix and d its degree. For the hybrid implementation, discussed in Sec. 2.1.4 and Appendix 2.B we used the same hyperparameters.

Appendix 2.B Discussion on various hybrid schemes implementations

The hybrid scheme discussed in Sec. 2.1.4 allows for highlighting the properties of the reservoir, but it is just one among the possible choices. Here, we briefly discuss three general schemes for the hybrid approach.

Let us assume, for simplicity, that our dynamical system, with state variables $s = (s_1, ..., s_n)$, is described by the equation $s(t+1) = f(s(t))$, which is unknown. Here, without loss of generality, we use discrete time dynamics and that we want to forecast the whole set of state variables, this is just for the sake of simplicity of the presentation. Provided we have an imperfect model, $s(t+1) \approx f_m(s(t))$, for its evolution, we have basically three options for building a hybrid scheme.

A first possibility is to approximate via machine learning only the part of the signal that is not captured by the model $f_m(s(t))$. In other terms, one writes a forecast as

$$\hat{s}(t+1) = f_m(s(t)) + \delta_n(s(t)) \tag{2.33}$$

where the residual δ_n is given by the network, and can be learned from a set of input-output pairs $s(t), s(t+1) - f_m(s(t))\}_{t=-T}^{0}$ according to some supervised learning algorithm. In our framework, the hybrid network should be trained with the usual input but with target output given by the difference between the true value of $s(t+1)$ and the model forecast $f_m(s(t))$.

A second possibility is to add the available model prediction $f_m(s(t))$ to the input $s(t)$, obtaining an augmented input $(s(t), f_m(s(t))$ for the network. In this case, the forecast reads as

$$\hat{s}(t+1) = f_n(s(t), f_m(s(t)). \tag{2.34}$$

Clearly, if the model based prediction is very accurate, the network will try to approximate the identity function. The network should be trained with a set of input-outputs pairs $\{(s(t), f_m(s(t))), s(t+1)\}_{t=-T}^{0}$. This is the approach we have implemented in this article, in order to evaluate the performance of the reservoir.

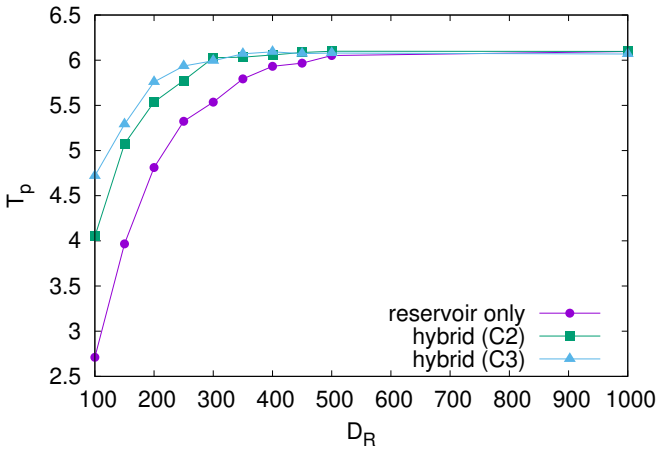

Figure 2.12: (Color online) Average (over 10^4 initial conditions) predictability times are shown for reservoir only and two hybrid implementations ($\Delta t = 0.1$ and $c = 10$). The green line corresponds to the hybrid scheme (2.34), blue lines to the hybrid scheme (2.35) and purple lines to the reservoir only baseline.

A third possibility is to combine the two previous options, which is the approach followed in ref. [194]. In this case, the forecast is obtained as:

$$\hat{s}(t+1) = \mathbb{A}\, f_m(s(t)) + \mathbb{B}\, \delta_n(s(t), f_m(s(t)))\,, \tag{2.35}$$

where the matrices $\mathbb{A}$ and $\mathbb{B}$ should be optimized, along with δ_n. This last option is a special case of the second scheme, describing a residual multilayered neural network with a linear output layer.

For the sake of completeness, in fig. 2.12 we show how this last architectures compares with the one we used in fig. 2.10a in terms of predictability. It consists in taking the optimized linear combination of the predictions from the hybrid net and the imperfect model. Namely, one augment the r^* array as $\tilde{r}^* = (r^*, f_m)$ and then optimizes $\mathbb{W}_O$ to achieve $v(t+1) = \hat{s}(t+1) \approx \mathbb{W}_O \tilde{r}^*(t)$. As one can see, the main effect is to slightly shift the predictability-vs-size curve leftward, meaning that optimal performance can be achieved with a slightly smaller network. However, the improvement quickly disappears when the reservoir size increases.

Chapter 3

Modelling and understanding the macroscopic dynamics of a system of coupled maps through machine learning

The two scale Lorenz system we have used in the previous chapter has two main features: it is a well-studied system which features multiple timescales by construction. In many interesting real world scenarios, though, the multiscale structure emerges from collective interactions which do not have explicit multiscale behaviours. Likewise, the physics of the system is not known or, at least, is not known exactly.

There is a conceptual difference in applying machine learning technique to a system when one already understands its physics and when one does not, with many analyses focusing on the former case, as we have done in the multiscale Lorenz system. However, the latter case may be more interesting and challanging. Indeed, many machine learning techniques have been both employed to model unknown system from data, but also to understand new physics. The purpose of this work is to explore the tight connection between these two aspects in the framework of dynamical systems and, specifically, of collective dynamics of high dimensional chaotic systems. As a case study, in reference [36], on which this chapter is based, we have explored a system of coupled maps.

3.0.1 Exploring the connection between modelling and understanding

"Coupled maps" is an umbrella term referring to a class of high dimensional systems featuring a large number of interacting units. Such systems have long been studied since they are suitable toy models for high dimensional (typically) chaotic systems and they are of interest in several branches of physics, including turbulence, neural networks and biophysics [35, 132, 133]. A remarkable feature of such systems is that relatively simple inter-unit interactions result in complex collective behaviours. Depending on the topology of unit-to-unit connections and on the nature on interactions, several behaviours are possible; coupled maps on lattice, for instance, can display pattern formation [128, 187]. Even fully connected networks of units can produce non-trivial macroscopic behaviours due to the emergence of coherent structures which are the result of synchronization between subsets of units: as a result, macroscopic observables are not trivially self averaging but display a variety of behaviours. Such collective dynamics, which may also be chaotic [238], have been studied with

a number of techniques, such as FSLE [13, 34, 53, 228] or Frobenius-Perron equation [206, 239] and covariant Lyapunov vectors [47, 87, 95, 239]. It has been argued that the emergence of a slow collective dynamics should be understood in terms of an effective low dimensional attractor. This problem may be studied through the computation of covariant Lyapunov vectors [47, 87, 95, 239]: the presence of covariant Lyapunov vectors involving an extensive number of microscopic degrees of freedom [238] implies the presence of a collective modes. Through this kind of analysis, one may estimate, in some cases, the effective dimension of the macroscopic attractor [271].

While the possibility of deriving some macroscopic properties from microscopic ones through covariant Lyapunov vectors is crucial to the analysis of the global behaviour of the system, a purely macroscopic dynamical reconstruction from macroscopic data is also relevant for a number of reasons. Indeed, Lyapunov analysis is computationally expensive for large systems, and it requires excellent knowledge of the microscopic structure, so that its application may result unfeasible in many cases. Most importantly, in many realistic physical scenarios one only has access to global observables: one is then interested in building a low-dimensional, coarse-grained model from data, with the purpose of making reliable predictions at the macroscopic level. In the coupled maps framework, this problem displays features which are common to several physical problems, such as turbulence or climatology: the low-dimensional collective behaviour, the multiscale structure and the difficulty to distinguish high-dimensional deterministic dynamics from noise. Many approaches have been developed to face this kind of problem, most of which are based on the assumption that a relatively simple stochastic differential equation properly describes the dynamics. The coefficients can be found by mean of a careful analysis of conditioned moments [85, 230], even in the case of memory kernels [150], or via a Bayesian approach [83]. This kind of strategy has been successfully employed in many fields of physics, including turbulence, soft matter and biophysics [21, 42, 85, 199]. In recent years many different machine-learning approaches were also developed. While even pure model-free methods can be very efficient [39, 160, 191], other approaches aim to blend physical information with data-only techniques [151, 194, 266]. An important role is played by those machine learning methods attempting to extract an effective low-dimensional dynamics, for instance by employing autoencoder based networks [154, 193].

One of the simplest but also well studied models of coupled maps was introduced by Kaneko [129] and can be written as follows

$$y_{t+1}^i = (1-\varepsilon) g_a(y_t^i) + \frac{\varepsilon}{N} \sum_{j=1}^{N} g_a(y_t^j) \quad i = 1, ..., N \,, \tag{3.1}$$

where the N variables $\{y^i\}$ belong to the interval $[0, 1)$ and $g_a : [0, 1) \mapsto [0, 1)$ is a chaotic map, whose behaviour is determined by a control parameter a. Some choices for g_a include the logistic map, the circle map and the tent map [130]. We chose the last option which reads

$$g_a(y) = a \left(\frac{1}{2} - \left| \frac{1}{2} - y \right| \right) \,, \tag{3.2}$$

with $1 < a < 2$. The interaction strength in the dynamics (3.1) is can be tuned via the coupling parameter ε: in the limit case $\varepsilon = 0$ all maps evolve independently and, therefore, after a transient, they may be trivially described as independent and identically distributed random variables (for

typical initial conditions). The opposite condition $\varepsilon = 1$ forces perfect synchronization on all units, so that the system is given by a trivial one-dimensional chaotic variables. Intermediate values in the range $0 < \varepsilon < 1$ lead to a rich and complex phenomenology, investigated in a series of papers in the 1990s [69, 130, 131, 134]. Synchronisation, violation of the law of large numbers and tree-structure organization of the chaotic attractors are among the features emerging from the opposite effects of synchronizing interaction and chaos.

We choose the mean value

$$z_t = \frac{1}{N} \sum_{i=1}^{N} y_t^i \,, \tag{3.3}$$

as the macroscopic observable describing the collective behaviour of the N – the coupled maps – microscopic variables which evolve as (3.1). The microscopic evolution is clearly chaotic as soon as $a(1-\varepsilon) > 1$ as explained in Refs. [53, 228]. For suitable choices of the parameters a and ε, the macroscopic dynamics can be regarded as chaotic as well, since it is characterized by a small positive FSLE [13, 34] ("macroscopic chaos"). While the microscopic attractor dimension is typically extensive in N, the macroscopic attractor has a low dimension, almost independent of N, as long as N itself is large enough for the collective modes to prevail over microscopic fluctuations.

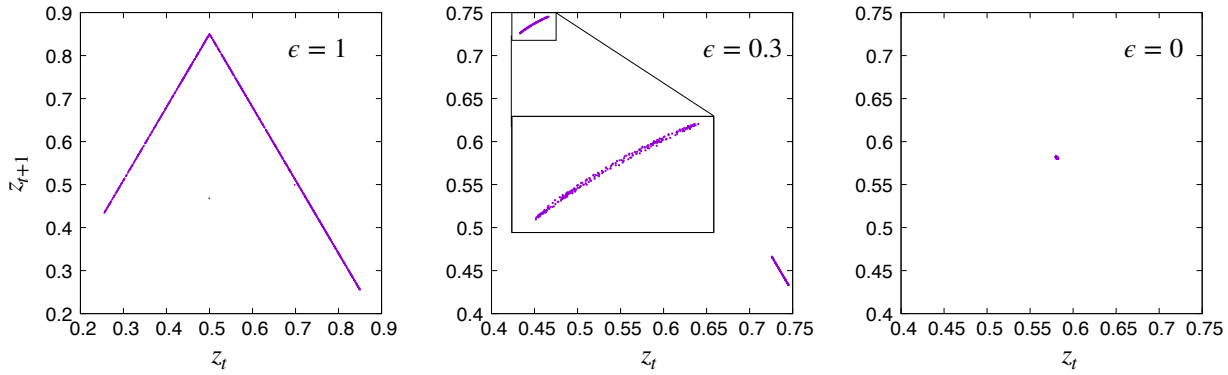

Figure 3.1: The three panels show the z_{t+1} vs z_t (see definition (3.3)) plot for $a = 1.7$, for three values of ε. For $\varepsilon = 0$, maps are independent and the macroscopic variable z is trivially self averaging (for typical initial condition); for $\varepsilon = 1$, maps are fully synchronized and the macroscopic dynamics correspond to a that of a single tent map; for $\varepsilon = 0.3$ some complex macroscopic behaviour is displayed, as a result between competition between chaos and synchronization.

In this chapter, we show that it is possible to build coarse-grained stochastic models for the collective signal generated by the maps, using a stochastic machine-learning method, which allows us to account both for the existence of a low-dimensional effective manifold and for the presence of residual high-dimensional dynamics. This stochastic approach is in line with theoretical arguments that stochastic modelling is the correct coarse graining procedure in hydrodynamical-like systems [78, 189, 190]. The results of our analysis are carefully compared to those achievable by mean of a direct numerical computation of transition probabilities. Remarkably, such an approach to model reconstruction also yields relevant physical information about the underlying physical process, implying that the quality of forecasts and the physical insight are strongly interconnected. We therefore explore the possibility to overcome the separation between a purely result-oriented approach and theoretical investigation, which is a promising direction for machine learning based approaches [23, 44, 67, 154].

3.1 Macroscopic motion in globally-coupled maps

In the following, we will consider the parameter pair

$$a = 1.7 \quad \varepsilon = 0.3\,, \tag{3.4}$$

investigated by Cencini et al. [53]. In this example the dynamics of z_t is characterized by a quasi period 4 dynamics, whose origin is related to the two-band structure of the maps distribution at a given time [134], discussed in some detail in Appendix 3.A. For this reason, the variable

$$x_t := z_{4t}\,, \tag{3.5}$$

looks very similar to a continuous process. The dynamics of x_t strongly depends on the number of maps composing the system, as clear from fig. 3.2, displaying a trajectory segment fro different values of N. By increasing the value of N, the trajectories look more regular and almost display a periodic behaviour with period $\simeq 10^2$. This observation may suggest the presence of a second-order dynamics, a possibility also seemingly corroborated by the shape of the two-dimensional projection of the dynamical attractor in the $N = 10^6$ case (fig. 3.2(d)).

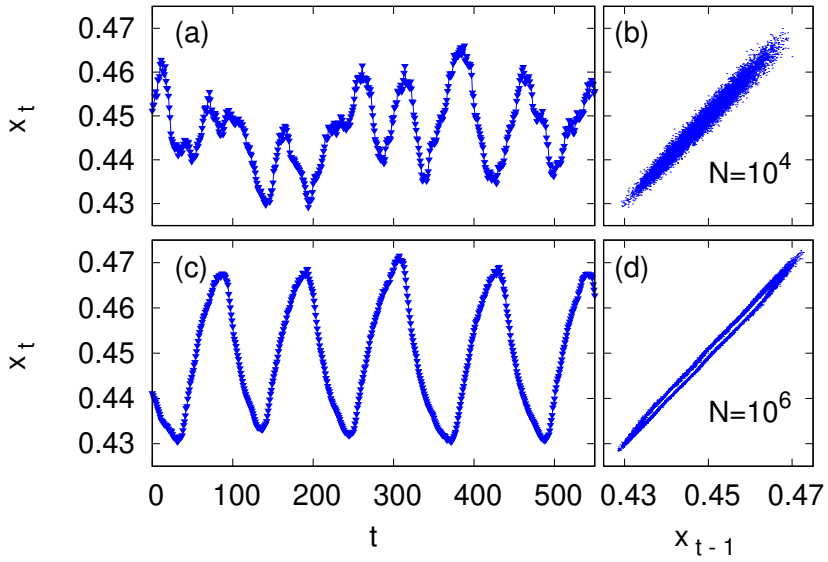

Figure 3.2: Dynamics of x_t (eq. (3.5)). For two different choices of the number of maps (top $N = 10^4$, bottom $N = 10^6$), sample trajectories of x_t are shown (panels (a) and (c)), as defined in Eq. (3.5). In panels (b) and (d) the two-dimensional projections of the corresponding dynamical attractors are displayed.

The standard Grassberger-Procaccia correlation dimension analysis [51, 98] provides a better insight into the dimensionality of the macroscopic dynamics. The correlation dimension provides a measure the fractal dimension of a chaotic attractor, which is connected to the number of variables one needs to reconstruct an unknown dynamics, as suggested by Takens[237]. Let us define the m-dimensional embedding vector (also called "delay vector") as

$$X_t^{(m)} = (x_t, x_{t-1}, ..., x_{t-m+1})^T\,, \tag{3.6}$$

and define the stationary measure $\mu : \mathbb{R}^m \mapsto \mathbb{R}$ associated to the dynamical variable $X_t^{(m)}$. The correlation integral at length-scale L is

$$G_m(L) = \int d\mu(X^{(m)})\, d\mu(Y^{(m)})\, \mathbb{H}(L - \|X^{(m)} - X^{(m)}\|)\,, \tag{3.7}$$

where $\mathbb{H}$ is the Heavyside step-function. The correlation integral can be numerically estimated as:

$$G_m(L) \simeq \frac{2}{T(T-1)} \sum_{t=1}^{T-1} \sum_{s=i+1}^{T} \mathbb{H}(L - \|X_t^{(m)} - X_s^{(m)}\|)\,, \tag{3.8}$$

where T is the length of the empirical trajectory. The log-log slope of $G_m(L)$, namely

$$D_{GP}(L) = \frac{d[\ln G_m]}{d[\ln L]}, \tag{3.9}$$

defines the correlation dimension of the attractor measured at scale L, which is always lower than m.

Figure 3.3a suggests that D_{GP} is characterized by (at least) two different regimes: for small values of L we observe $G_m(L) \sim L^m$, i.e. D_{GP}, in each curve, tends to its maximum value m. Conversely, at larger scales, all curves with $m > 1$ reach a slope ~ 1.3 in the log-log plot, meaning that at macroscopic scales the attractor has a low-dimensional structure. Let us also stress that the cross-over length L_{cross} between the two regimes depends on the number of maps N, as shown in fig. 3.3(b).

In the proposed dynamical reconstruction, we assume that at least part of the fluctuating signal seen in fig. 3.3, originally produced by a deterministic mechanism, may be described as stochastic noise, in line with other probabilistic approaches to deterministic systems [9, 33, 210]. Hence, our purpose is to study the nontrivial macroscopic dynamics described above from data. In particular, we are interested in a Markovian[1] stochastic description of the process. A Marokv process is defined in terms of the conditional probability

$$p_n(x_{t+1}|x_t, ..., x_{t-n+1}) = p_n(x_{t+1}|X_t^{(n)}) \tag{3.10}$$

for the value assumed by the variable at a generic step, once n previous steps are known. As it will discussed in the following sections, one of the main difficulties in this approach is represented by the fact that we do not know in advance what value of n should be considered and, in principle, we do not even know whether such value exists. This is a general issue which is typically encountered when trying to infer a model from data [19] which is faced in a variety of ways, including machine learning techniques such as the one we have employed in this work, or other techniques such as the false neighborhood method [45, 106] typically used for estimating the appropriate size of the embedding vector for dynamical reconstruction.

The absence of an underlying low-dimensional model for x_t makes this problem an appropriate benchmark for the machine-learning approach. Moreover, as already mentioned, the quasi-periodic oscillations of x_t suggest that a second-order like stochastic modelling ($n = 2$) may be enough

[1] Markovianity will always be referred to the delay vector dynamics $X_t^{(n)}$ and not to the macroscopic variable x_t itself. Indeed, specifying $P(x_{t+1}|X_t^{(n)})$ is equivalent to specifying $P(X_{t+1}^{(n)}|X_t^{(n)})$.

(though we can anticipate that it is not). Such minimal and intuitive guess can be implemented by a direct numerical computation of the transition probabilities and offers a benchmark in evaluating the performance of the machine learning approach.

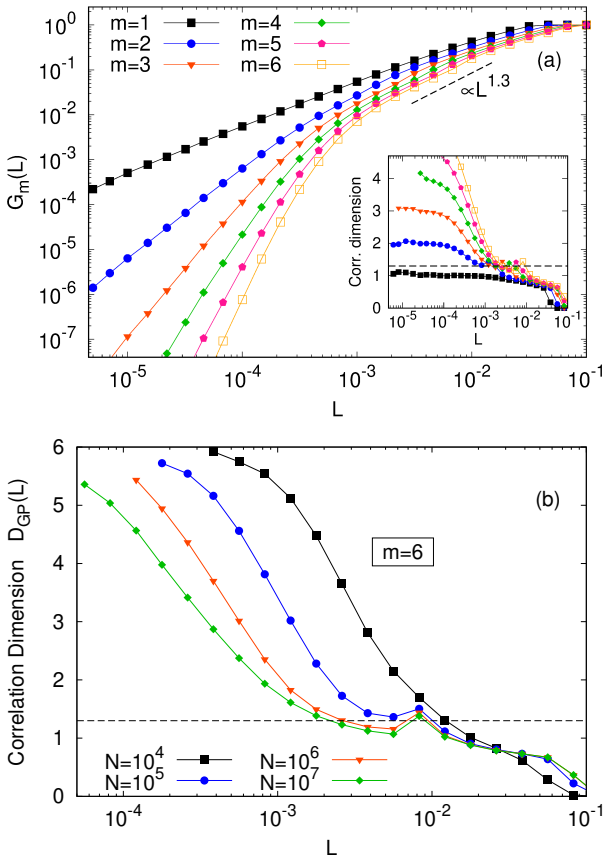

Figure 3.3: Grassberger-Procaccia analysis. Panel (a): correlation integral defined by Eq (3.8), for different values of the embedding dimension m. Here $N = 10^6$. In the inset, correlation dimension defined by Eq. (3.9) as a function of L; the dashed line is a guide for the eyes, corresponding to dimension 1.3. Panel (b): dependence of $D_{GP}(L)$ on the number of maps N, for $m = 6$.

3.2 Stochastic dynamics from data

The rationale of the analysis is the assumption that the trajectory of x_t can be fairly approximated by a relatively short-memory Markovian stochastic process. For this purpose, we should determine by the conditional probability (3.10) and, in doing so, we adopt a purely data-drinven approach, only exploiting computer-generated macroscopic trajectories, while discarding any additional physical insight about the underlying N-dimensional dynamics. This procedure can be interpreted as a form of coarse graining, and yields an effective description for the dynamics of x_t. In the following, we present both an intuition-based version of this approach and its machine learning counterpart.

3.2.1 Validation methods

Before we introducing our two ways of inferring a Markov process from data, we should present the metrics that we have used in order to evaluate the quality of the reconstruction. The choice of such metrics is not neutral, since, in general, short and long term predictions are not equivalent tasks

and their performances, while generally correlated, may still be different as explained in the case of ESRNN. Even if short term errors are very small, the variable could still eventually leave the attractor, if this is unstable [33]; conversely, long term accuracy does necessarily require high short-term resolution. In the same spirit of the ESRNN approach discussed in the previous chapter, we trained the model to produce short-term forecast, with the implicit actual goal to achieve long-term predictions too and, then, we validated the quality of the reconstruction in both cases.

A good test for the quality of the reconstruction is the analysis of the Fourier power spectrum, defined as

$$S(f) = \Big| \frac{1}{2\pi T} \sum_{t=0}^{T} e^{2\pi i f t} x_t \Big|^2 , \tag{3.11}$$

where T is the length of the empirical trajectory. The comparison of the spectrum of the original dynamics with that of the reconstructed one constitutes a powerful tool to evaluate the quality of the latter at any time scale. Since our approach is based on the optimization of one-step predictions, it is reasonable to expect a good matching at high frequencies f. On the other hand, a good match at low f too would be a reliable indication that the reconstructed attractor is stable and that the climate is reproduced correctly.

In order to evaluate the quality of the method on the short-time scales, where randomness may matter the most, we use the average cross entropy. This metrics is a natural choice, since it is commonly employed as loss function in classification problems, under the name of "categorical crossentropy". This metrics will have the twofold role of being the loss function in training the neural network and of the validation metrics in the intuition-based approach. It is also employed in sec. 3.3 to determine the best setting for both methods. Assume time is discrete and let us call x_t a generic continuous-valued dynamical variable. Let $p_n(x_t|x_{t-1}, ..., x_{t-n-1})$ be the true conditional probabilities of the process. Conversely, let $q_n(x_t|x_{t-1}, ..., x_{t-n-1})$ be the conditional probability associated to the model dynamics. Bearing in mind the definition of the delay vector given by Eq. (3.6), we introduce the *average* conditional cross entropy between the true and the model distribution as

$$C_n = -\int d\mu(X^{(n)}) \int dx\, p_n(x|X^{(n)})\ \ln q_n(x|X^{(n)}) = \langle H[q_n|p_n] \rangle \,, \tag{3.12}$$

where $\mu(X^{(n)})$ is again the natural measure on the true dynamical attractor and $\langle \cdot \rangle$ denotes the corresponding average. The lower the value of C_n is the more similar p_n are q_n are: the minimum of C_n with respect to the possible choices of q_n is achieved when setting $p_n = q_n$. Consistently, the distribution q_n that minimizes C_n also minimizes the average Kullback-Leibler divergence [68]:

$$\langle D[q_n \| p_n] \rangle = -\int d\mu(X^{(n)}) \int dx\, p_n(x|X^{(n)})\ \ln \left(\frac{q_n(x|X^{(n)})}{p_n(x|X^{(n)})} \right) = \langle H[q_n|p_n] - H[p_n] \rangle \,. \tag{3.13}$$

Notice that the two quantities $\langle D[q_n \| p_n] \rangle$ and $\langle H[q_n|p_n] \rangle$ differ by a term not depending on q_n. It should remarked that this metrics only detects discrepancies in one-step forecasts.

It is interesting to oberserve that, if the system is Markovian after $\tilde{n}$ steps, then the minimum possible value of the average cross entropy is the Shannon entropy rate, given by $q_n = p_n$ for any

$n \geq \tilde{n}$:

$$h = -\int d\mu(X^{(n)}) \int dx\, p_n(x|X^{(n)})\, \ln p_n(x|X^{(n)}) \text{ with } n \geq \tilde{n}\,. \tag{3.14}$$

If we assume that our model provides an explicit expression for q_n, then, given any dataset of input-output pairs $\Omega_{test} = \{X^{(n)}_{(i)}, x_{(i)}\}_{i=1}^{T_{test}}$ (the sequence obtaining by appending an output to its input sequence, namely $(X^{(n)}_{(i)}, x_{(i)})$, is a piece of a trajectory consisting of $n+1$ elements), the average cross entropy (3.12) is the expected value of the following empirical quantity:

$$C_n(\Omega_{test}) = \sum_{i=1}^{T_{test}} \log q_n(x_{(i)}|X^{(n)}_{(i)})\,. \tag{3.15}$$

3.2.2 Position-velocity Markov process (p-vMP): a physics-inspired approach

The intuition based approach is based on the previous observation that the oscillatory behaviour of x_t trajectories (see fig. 3.2) is reminiscent of a second-order dynamics. Qualitatively, its time evolution may resemble that generated by an underdamped harmonic oscillator, at least for the cases in which N is large. At first sight, one might expect a reasonable approximation of the dynamics to be achieved by means of simple Langevin equations of the form

$$\begin{cases} x_{t+1} &= x_t + v_{t+1} \\ v_{t+1} &= u(x_t, v_t) + \sqrt{2B}\,\eta_t\,, \end{cases} \tag{3.16}$$

where $u : \mathbb{R}^2 \mapsto \mathbb{R}$ is some smooth function, B is a constant (possibly depending on N) and η_t is a zero-mean Gaussian noise such that $\langle \eta_t \eta_s \rangle = \delta_{t,s}$ and $\langle \eta_t \rangle = 0$. In the above equations, the variable

$$v_t = x_t - x_{t-1} \tag{3.17}$$

may be regarded as the "velocity" of the "position" variable x_t. If intuition is correct, under the assumption that

$$\langle v^2 \rangle - \langle v \rangle^2 \ll \langle x^2 \rangle - \langle x \rangle^2 \tag{3.18}$$

(i.e. the dynamics is close to the "continuous limit"), one could exploit a data-driven approach to build a discrete-time Langevin system of equations [20, 21, 85, 199]; this strategy is based on the possibility to infer an approximate functional form for $u(x, v)$ by considering the small-time limit of suitable conditioned moments. Unfortunately, the model constructed with this method (not shown) is severely unsuccessful since the predicted trajectory does not remain bounded in the correct domain. Therefore, a model like eq. (3.16) does not provide a viable path.

Despite the practical difficulties in deriving a model of the form (3.16) from data, a second order stochastic dynamics involving x_t and v_t remains a viable option. Guessing that that failure of the model (3.16) is due to the Gaussian assumption being too restrictive, we tried a more general ansatz with the same underlying idea of a second order dynamics with a random force. The idea is that the model should retain memory only of two time steps, i.e.

$$p_n(x_{t+1}|X^{(n)}_t) = p_2(x_{t+1}|X^{(2)}_t) \qquad \forall n \geq 2\,, \tag{3.19}$$

consistently with a two-dimensional structure of the dynamical attractor at the typical length-scales

of the dynamics of our chosen test system, suggested by fig. 3.3(a) (we recall that $D_{GP} \approx 1.3$ for "large" L). A simple change of variable from (x_{t-1}, x_t) to (v_t, x_t) allows to recover the position-velocity Markov process (p-vMP) akin to eq. (3.16) but defined in terms of a generic transition probability

$$p(v_{t+1}|x_t, v_t), \tag{3.20}$$

which may be fitted from data. The complete procedure is the following. First compute variable ranges $[x_{min}, x_{max}]$ and $[v_{min}, v_{max}]$ from data in order to construct a rectangle in the phase space containing the macroscopic dynamics. Choose a integer N_b corresponding to the number of bins per dimension and create a partition Γ of $[x_{min}, x_{max}] \times [v_{min}, v_{max}]$ by dividing the rectangle in N_b^2 bins of equal sizes. Each cell - say bin j - in Γ is associated to N_b additional bins corresponding to the interval $[v_{min}, v_{max}]$: these bins correspond to discretized predictions of v_{t+1} conditioned to the fact that (x_t, v_t) belongs to bin j. One should consider a long empirical trajectory and compute the histogram of the measured values of v_{t+1}, conditioned to the state represented by each cell in Γ.

In fig. 3.4, we provide a graphical representation of this method. In the main plot, the (discretized) (x_t, v_t) phase space is represented; the color of each cell is given by the average value of v_{t+1}. Plots (a)-(d) show the normalized histograms of v_{t+1} for selected states, corresponding the empirical pdfs $q(v_{t+1}|j, t)$. It is interesting to note the qualitative difference between plot (b) and plot (d): in both cases $\langle v_{t+1} \rangle \simeq 0$, but in plot (d) the distribution features a single peak centered around zero, whereas in plot (b) the histogram is clearly bimodal. The presence of two peaks in the latter case may be regarded as the superposition of two different distributions, corresponding to predictions from two very different states of the original system which fall inside the same bin; it is something one expects when projecting a dynamics to a lower dimension. Even if cases as that of plot (b) are quite rare, this is a first hint that our "state identification" by mean of x_t and v_t only is not appropriate.

Still, if the trajectory employed to build the histograms is long enough, and the binning is sufficiently refined, we may hope that $q(v_{t+1}|j, t)$ is a fair approximation of the "true" stationary conditional probability $p(v_{t+1}|x_t, v_t)$. The rule

1. extract v_{t+1} according to $q(v_{t+1}|j, t)$;
2. evolve x_t by imposing $x_{t+1} = x_t + v_{t+1}$

defines the reconstructed position-velocity Markov process (p-vMP) on the (discretized) states of the system. In the following section, we will discuss how good this reconstruction is, and we will use it as a benchmark for the results of a machine-learning based approach.

Let us notice that our method here is purely "frequentistic", meaning that we do not take advantage of any prior knowledge or assumption about the functional form of $q(v_{t+1}|j, t)$. If, by any chance, the reconstructed trajectory reaches a state which in the original trajectory has been explored very few times (less than a fixed threshold, 10 in our simulations), so that it is not possible to get a reliable prediction about v_{t+1}, the state of the system is re-initialized, according to the empirical distribution. However, the frequency of this kind of events becomes negligible as soon as the length T of the original trajectory is long enough (e.g. less than once in 10^4 steps for $T \simeq 10^6$). Notice that, in principle, nothing prevents from us from extending this approach and incorporate larger memory effects, but the amount of data that would be needed makes it unfeasible.

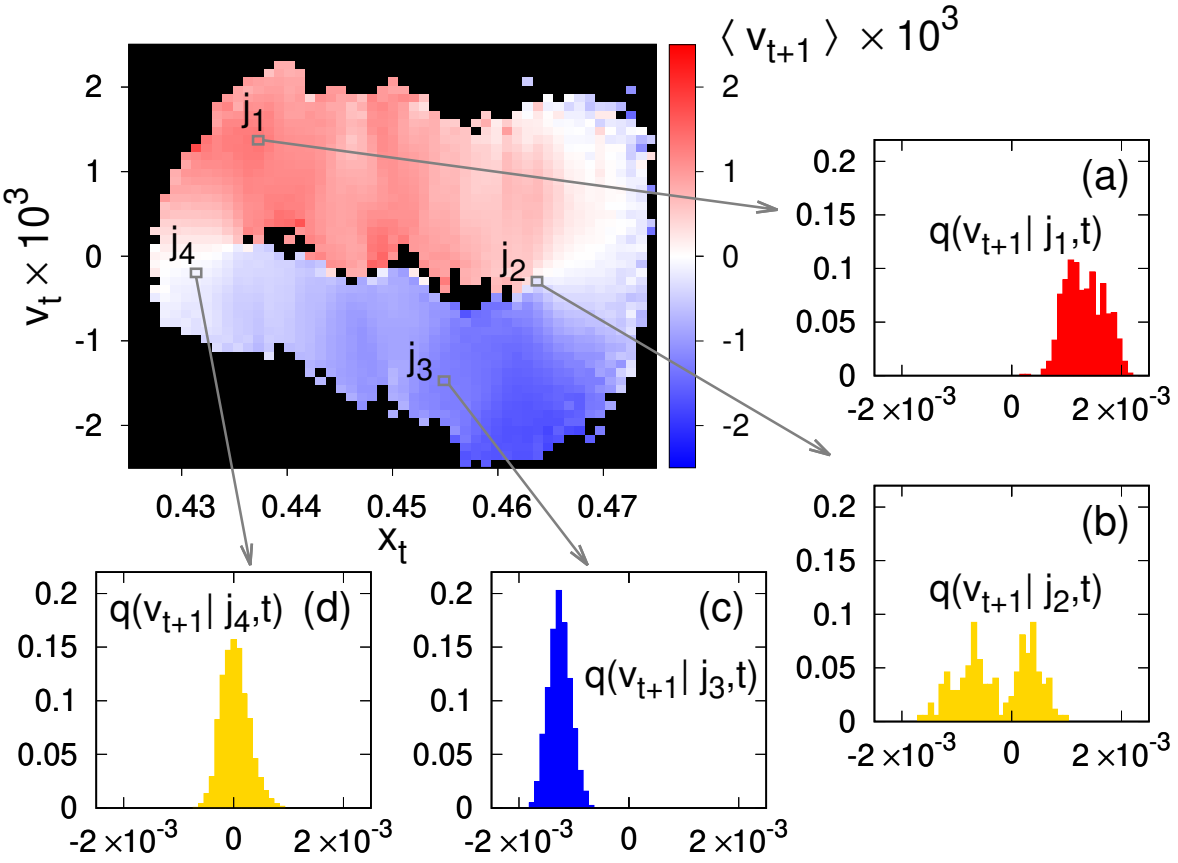

Figure 3.4: Modeling a Markov dynamics. Main plot: for each cell of the (discretized) (x_t, v_t) phase space, the average $\langle v_{t+1} \rangle$ is represented with the color code reported on the right side of the plot. Plots (a) - (d) show examples of the conditional empirical pdf $q(v_{t+1}|j,t)$, as discussed in the main text. Here, $N = 10^6$. The trajectory used to make this plot has a length $T = 10^6$, the number of bins for the histograms and for the discretization of the phase space is $N_b = 50$.

3.2.3 Machine learning (ML) approach

In this section, we outline a machine learning approach to the construction of a stochastic model for the macroscopic dynamics. The key assumption is that the distribution of the variable x_t is a function of the past history of the variable itself:

$$x_t \text{ drawn from } p(\cdot|\{x_s\}_{s<t}) \tag{3.21}$$

just as one assumes when building the two-steps empirical p-vMP. Our purpose is to approximate p with an empirical distribution q given by a neural network. We remark that the role of the neural network is simply to provide a class of distributions (over the space of past histories $\{x_s\}_{s<t}$) which is large enough to fit any non pathological distribution p in an efficient way (as ensured by the universal approximation theorems) [168]. There are at least two relevant differences between the machine learning approach and the p-vMP. The first is that, unlike for the specific Markovian method presented before, the input values of the network-based recontruction of p are (machine) continuous vectors. The second is that neural networks are naturally suited for processing long time series, and we can easily condition a forecast for x_t to very long pieces of trajectories $\{x_s\}_{s<t}$. The reason is that the network is optimised to discard irrelevant information and, therefore, it may fit

functions of high-dimensional inputs with relatively low amount of data. Certain kind of neural networks can deal with indefinitely long input sequences in principle, but there is no reason to expect that arbitrarily long sequences of data can or need to be exploited. For this reason, in this work we always base predictions on delay vectors with definite size.

Likewise, even though we are working with time series, the most suitable choice for our purpose is a feedforward neural network, as described in the following (see sec. 1.2.2 for a minimal introduction to this kind of networks). Other tools, such as recurrent neural networks, (e.g. LSTM [110, 151]), can perform well but would not have allowed the efficient manipulation of the input vectors $X_t^{(n)}$ that we need to investigate physical properties.

Our goal is to estimate the true probability

$$p_n(x_t|X_t^{(n)}) \tag{3.22}$$

from data. Given p_n, we define a Markov chain for $X^{(n)}$,

$$\begin{bmatrix} x_{t-1} \\ x_{t-2} \\ ... \\ x_{t-n} \end{bmatrix} \mapsto \begin{bmatrix} \hat{x}_t \\ x_{t-1} \\ ... \\ x_{t-n+1} \end{bmatrix}, \tag{3.23}$$

with

$$\hat{x}_t \text{ drawn from } p_n(\cdot|X_{t-1}^{(n)}), \tag{3.24}$$

with which we can approximate the true dynamics. In general, we expect that the reconstruction improves[2] when increasing n. We can hope that, for some n, this procedure can provide a Markovian coarse grained model for the true dynamics. Notice that the $n = 2$ case is conceptually equivalent to the p-vMP discussed in the previous section.

The goal is to approximate the distribution p_n with a probability function defined on a finite number of elements with a feedforward neural network. Instead of binning the random variable x itself, it is convenient to use the same approach as in the position-velocity Markov process by defining and attempting to approximate $p_n(v_t|X_{t-1}^{(n)})$. This allows us to achieve greater precision with fewer bins, since the typical scale of velocities is much smaller than the scale of positions, but it is not an otherwise mandatory passage.

In order to achieve probabilistic predictions, the activation function of the neural network is chosen as the softmax (see 1.2.2), so that the neural network output is the following probability function Q_n (for delay n and with an abuse of notation)

$$Q_n(j|X^{(n)}) := \int_{v_{min}+(j-1)\,\Delta v}^{v_{min}+j\,\Delta v} dv\; q_n(v|X^{(n)}) = \frac{1}{Z}\exp(H_j(X^{(n)}))\;\; \forall j = 1, ..., N_b + 1, \tag{3.25}$$

where $H_a = \sum_{b=1}^{n_{M-1}} w_{ab}^{(M)} F_b^{(M-1)} + \theta_a^{(M)}$ are functions involving M hidden layers of the neural network (see sec. 1.2.2 for details and notation). $\Delta v = (v_{max} - v_{min})/N_b$ is the bin size, where the extreme values v_{max} and v_{min} have to be computed empirically. Q_n depends on a set of parameters

[2]Note that this procedures, contains two approximations: the first is given by fixing n which is potentially too small; the second is given by the model error in reconstructing p_n.

$\Theta = \{(\theta, w)\}$ which can be fixed by minimizing a cost function with a gradient descent algorithm (see sec. 1.2.2). A standard procedure for fitting the empirical distribution Q_n is the minimization of the empirical cross entropy between the target distribution P_n (which is the discretization of p_n) and Q_n,

$$\hat{C}_n(\Omega_{train}) = \frac{1}{T_{train}} \sum_{i=1}^{T_{train}} \ln Q_n(I(v_{(i)})|X^{(n)}_{(i)}) \tag{3.26}$$

for a given training set $\Omega_{train} = \{X^{(n)}_{(i)}, v_{(i)}\}_{i=1,\dots,T_{train}}$ where the operator I assigns to every v in the interval (v_{min}, v_{max}) the integer label of the corresponding bin, ranging from 1 to N_b. The expected value of C is

$$\langle \hat{C}_n(\Omega) \rangle_\Omega = \int d\mu(X^{(n)}) \sum_{j=1}^{N_b} P_n(j|X^{(n)}) \ \ln Q_n(j|X^{(n)}). \tag{3.27}$$

This procedure is equivalent to minimizing the quantity

$$C_n = \int d\mu(X^{(n)}) \sum_{j=1}^{N_b} \Delta v \ \frac{P_n(j|X^{(n)})}{\Delta v} \ \ln \frac{Q_n(j|X^{(n)})}{\Delta v} \tag{3.28}$$

since

$$C_n = \hat{C}_n + \log(\Delta v). \tag{3.29}$$

From now on, we will refer to (3.28) when mentioning the cost function - unless otherwise specified - which will be always estimated empirically as in (3.26) on a test set $\Omega_{test} = \{X^{(n)}_{(i)}, v_{(i)}\}_{i=1,\dots,T_{test}}$, statistically independent from Ω_{train}.

For Δv small, under physically reasonable assumptions, C_n reduces to the relative entropy (3.12) (see sec. 3.2.1), introduced as cost function for continuous-valued distribution, if we replace

$$\sum_{j=1}^{N_b} \Delta v \mapsto \int_{v_{min}}^{v_{max}} dv \quad \text{and} \quad \frac{P_n}{\Delta v}, \ \frac{Q_n}{\Delta v} \mapsto p_n, q_n. \tag{3.30}$$

In order to be sure that the statistical approach is not a redundant machinery which can be replaced with a deterministic network, we can build a network for deterministic predictions and compare the outcomes of the two approaches. The latter network only differs from its stochastic counterpart in the last layer, where we use the identity as activation function (linear layer, see sec. 1.2.2). The network correspond to a function S_n

$$\hat{v} = S_n(X^{(n)}) \tag{3.31}$$

which can be trained with a least square error procedure, i.e. minimizing

$$C_n(\Omega_{train}) = \frac{1}{T_{train}} \sum_{i=1}^{T_{train}} \|v_{(i)} - S_n(X^{(n)}_{(i)})\|^2, \tag{3.32}$$

with training set $\Omega_{train} = \{X^{(n)}_{(i)}, v_{(i)}\}_{i=1,\dots,T_{train}}$.

All results here presented have been obtained with a feedforward neural network with two

intermediate layers, each one with 300 hidden neurons. We also tried some different sizes but could not detect relevant differences in performance. The layers were fully connected, and we employed the ReLU activation function everywhere but for the output layer (deterministic case→linear layer; stochastic case→softmax layer). We trained the network with the ADAM [138] gradient descent algorithm.

3.3 Implementation details

The methods introduced in the previous section allow us to build effective dynamics from the analysis of trajectories. The outcomes of these protocols crucially depend on the setting of parameters such as the bin size and the length of the analyzed trajectory: in order to obtain meaningful results it is important to verify that our choices are sensible, and this task can be accomplished by a careful analysis of the cost function introduced in sec. 3.2.1. An important question is whether our results are descriptive of the system itself or whether they are only meaningful within the narrow context of our particular tool choice. With a careful analysis of the dependence of our methods on the setting parameters, we can definitively attribute results to the physics, with conventional limitations in confidence that are intrinsic in numerical studies.

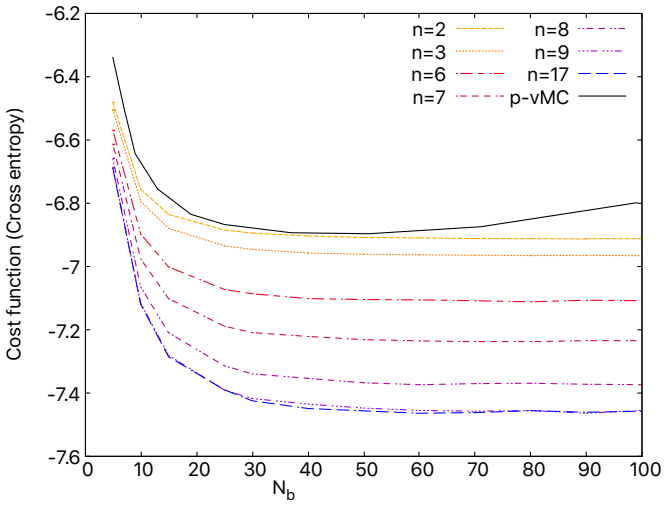

Figure 3.5: Cross entropy dependence on the number of bins. Black solid line shows the dependence of $C_{p\text{-}v}$ (eq. (3.33)) on N_b in a p-vMP extracted from a trajectory of $T = 10^6$ steps in a system of $N = 10^6$ coupled maps. The optimal resolution is obtained for $N_b \simeq 50$. Neural network: the cost function (relative entropy C_n in eq. (3.28)), computed on a test set of $T_{test} = 5 \cdot 10^4$, reaches a plateau as the number of bins increases. The size of a bin is $\Delta v = (v_{max} - v_{min})/N_b = 9.64 \cdot 10^{-5}/N_b$. The chart is has been obtained with $N = 10^6$ maps; the training set size is $T_{train} = 10^6$. Different delays n are shown; notice the saturation for $n > 9$.

3.3.1 The choice of binning

The choice of the number of bins N_b is not just a technical point, but is related to the physics of the system. Indeed, $\Delta v = (v_{max} - v_{min})/N_b$ fixes the lowest scale at which increments $x_{t+1} - x_t$ can be resolved or, equivalently, the order of magnitude of the typical one-step forecasting error: all phenomenology taking place below this scale is implicitly discarded by the model. This is

particularly relevant when modelling systems with multiple spatial scales, such as the one we are examining [53].

In the p-vMP approach the number N_b of bins per linear dimension plays an important role. On the one hand, this parameter determines our ability to "resolve" the state of the system, i.e. the point of the position-velocity phase-space in which the system is found. On the other hand, N_b is also the number of bins used for the discretization of the conditional pdf $p(v_{t+1}|j,t)$, and therefore it determines the precision of our forecasting.

If one had access to arbitrarily long trajectories (and arbitrary machine precision), the quality of the reconstruction would be directly determined by N_b: the larger the number of bins, the better our the resolution would be. However, in practice, trajectories are have a finite size and, when the value of N_b is too large, the performance is reduced: indeed, for a proper reconstruction, it is crucial that most cells along the attractor are visited several times by the training trajectory in order to compute a reliable histogram for the conditional probability $p(v_{t+1}|j,t)$; if the bin size is too small, this is clearly not possible.

The above reasoning can be verified quantitatively by looking at the cross entropy eq. (3.12) and (discussed in sec. 3.2.1), adapted to this case:

$$C_{p\text{-}v} = -\int d\mu(x_t, v_t) \int dv_{t+1}\, p(v_{t+1}|x_t, v_t)\ \ln q(v_{t+1}|x_t, v_t)\,, \tag{3.33}$$

which is equivalent, by construction, to C_2. Figure 3.5 (black solid curve) shows the value of $C_{p\text{-}v}$ as a function of N_b, when a trajectory of $T = 10^6$ time steps is considered. We observe that the optimal number of bins is around 50, since after that value, the cross entropy starts increasing. This is a clear hint that the quality of the reconstruction is decreasing.

The network-based model requires a binning on the velocities (but not on the state of the system $X_t^{(n)}$), just like the p-vMP approach. In order to analyse the role of Δv, assume our training set is large enough (e.g. $T_{train} \approx 10^6$) to avoid significant dependence on its size - the dependence on T_{train} will be discussed later. Then, for any fixed n, we can compute the cost function C_n as a function of $\log(\Delta v)$. We can see from fig. 3.5 that if Δv is small enough (i.e. N_b is sufficiently large), C_n reaches convergence. This means that the sum appearing in the cost function (see eq. (3.28)) is converging - in a subset of the attractor of $X^{(n)}$ whose probability is close to 1 - to a well-defined integral in Riemann-sense.

Let us notice that in this case there is no upper limit on the choice of N_b, at variance with the p-vMP approach, since now N_b does not play any role in the identification of the state. This explains the difference between the curves representing C_2 and $C_{p\text{-}v}$ in fig. 3.5: they are qualitatively similar for $N_b < 50$; then the former reaches a plateau, the latter starts increasing for the reasons discussed above. Let us notice that the optimal values reached by these two curves are very similar: this is not surprising, since conditioning on of x_t and x_{t-1}, as it is done in the ML approach with delay $n = 2$, should be equivalent to conditioning on x_t and $v_t = x_t - x_{t-1}$.

Figure 3.5 is the first confirmation that our stochastic approach is more appropriate than a deterministic one. Indeed, it shows that the fraction of bins with non-zero probability (for any given $X^{(n)}$) does not decreases with N_b. The latter case, which would correspond to C_n being a decreasing function of Δv, would have been observed for a deterministic dynamics (or a distribution with several very thin peaks) where, for any $X^{(n)}$, there should be single "active" bin with non null

probability, provided that n is large enough that we can write $x_t = F(X_t^{(n)})$ with some deterministic function F. However, it is important to consider that the plateau (of C vs N_b) is not to be interpreted as evidence that the observed dynamics is truly stochastic. The smoothness of q_n may be a result of n being too short, not having enough data or our machine learning procedure not being good enough. We know by construction that this is our scenario: if we could use a delay n comparable to N, an implausible large neural network (layer sizes of order N) and a prohibitive long training length, we could have unveiled the decreasing behaviour C_n (with respect to Δv) associated to the deterministic nature of the coupled maps system. One might also notice that, in fig. 3.5, curves with $n = 9$ and $n = 17$ overlap, implying the existence of a second plateau in C vs n: this will be throughly discussed later.

The ML analysis shown in the following sections has been done by adopting $N_b = 50$, for the sake of consistency with the p-vMC. Further refinement, say $N_b = 100$, does not bear much greater computational hardness. Our analysis suggests that this binning was sufficient for good model building.

As a broader methodological note, it is important to highlight that, in the general scenario of modelling a D-dimensional dynamical variable, the number of bins would scale as N_b^D (N_b=bin number per linear dimension), which is not out reach for up to date machine learning technology, as long as D is reasonable low and one has enough data, but could otherwise result in the curse of dimensionality.

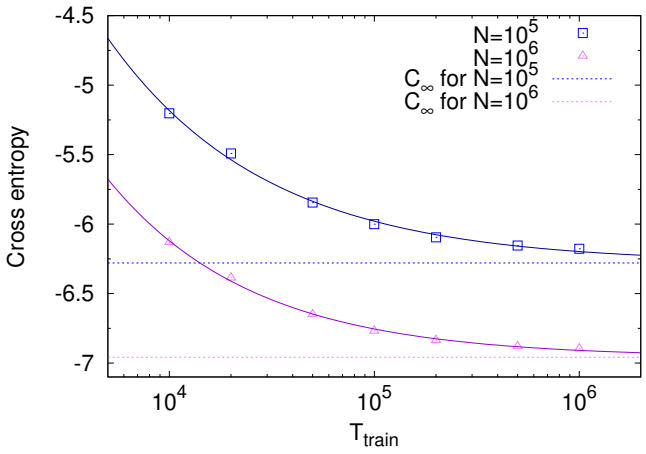

Figure 3.6: Dependence on T_{train} in the p-vMP analysis. For two different choices of N, the behaviour of the cost function (3.33) as a function of the length of the analysed trajectory is shown. Solid curves are obtained with a fit of the functional form (3.34), whereas dashed lines represent the inferred value of C_∞. Here $N_b = 50$. From the power-law fit we find $C_\infty = -6.28$, $\alpha = -0.562$ for $N = 10^5$ and $C_\infty = -6.96$, $\alpha = -0.615$ for $N = 10^6$.

3.3.2 Dependence on the training trajectory and asymptotic extrapolations

The approaches presented in sec.å 3.2 are based on the extrapolation of relevant information from data, in order to make reliable predictions. The quality of the results clearly depends on the amount of available data, i.e. on the length T_{train} of the original trajectory employed to infer the conditional probabilities of the p-vMP and to optimize the internal weights of the neural network during the "training" phase. The value of the cost function defined by Eq. (3.12) is thus expected to decay

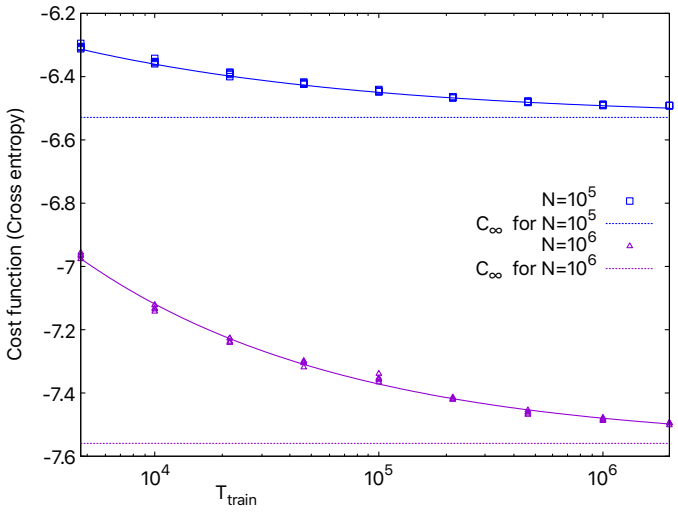

Figure 3.7: Dependence on T_{train} in the ML analysis. The cost function decreases in T_{train} as a power law. We have grouped values for different $n > \tilde{n}$ both in $N = 10^5$ and $N = 10^6$ cases. Dashed lines show extrapolated asymptotic values of the cost function C_∞. We have used $T_{test} = 10^6$ and $N_b = 50$. From the power-law fit we find $C_\infty = -6.53$, $\alpha = -0.325$ for $N = 10^5$ and $C_\infty = -7.59$, $\alpha = -0.368$ for $N = 10^6$.

with T_{train}. We observe that in all considered cases such decay is well described by a power law

$$C_n(N, N_b) \approx C_\infty(n, N, N_b) + \frac{c(n, N, N_b)}{T_{train}^{\alpha(n,N,N_b)}}, \tag{3.34}$$

where C_∞, c and $\alpha > 0$ are parameters which will depend, in general, on N_b, N and n, but we will drop dependencies for simplicity (as before, the cost function defined by Eq. (3.33) can be seen as equivalent to C_2).

Figure 3.6 shows the case of the p-vMP, for two choices of N. Assuming that eq. (3.34) holds asymptotically, we expect that no significant improvement in the reconstruction would be achieved by considering values of T_{train} larger than 10^6 (at least with this choice of the binning). Consistently, in the following we will always keep $T_{train} = 10^6$.

In order to control for the effect of the training trajectory length in the ML approach, we can plot the cost function as a function of T_{train} for different values of N (see fig. 3.7 and 3.8 for more details on $N = 10^6$ case). While $\alpha(n, N, N_b)$ and $c(n, N, N_b)$ are likely to be strongly dependent on the machine learning algorithm, the extrapolated $T_{train} \to \infty$ asymptotic value $C_\infty(n, N, N_b)$ should be an estimator for a procedure-independent observable, with a (hopefully small) bias deriving from the specific machine learning protocol: we know that, for fixed N, $C_n(N, N_b)$ reaches a plateau when N_b is large and $n > \tilde{n}$ (we will see that $\tilde{n} \approx 10$). We can interpret this as evidence of an underlying effective low-dimensional macroscopic Markovian structure in the original system. This Markovian structure is identified by a transition probability distribution which is well approximated by the empirical function q_n as defined in (3.28). If this is true, then the quantity (see sec. 3.2.1):

$$C_\infty \approx -\int d\mu(X^{(n)}) \int dx\, p_n(x|X^{(n)})\, \ln q_n(x|X^{(n)}) \text{ for } N \gg n > \tilde{n} \tag{3.35}$$

is a physical observable describing the cost function of the coarse grained process with respect to the true dynamics.

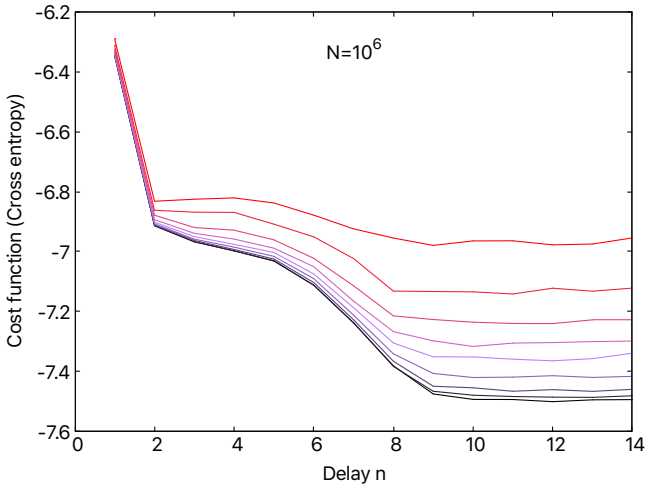

Figure 3.8: Cross entropy as a function of the delay n, for different lengths of the training set T_{train}. The value of $\tilde{n} \approx 10$ emerges as T_{train} increases. From top to bottom line, training times are $T_{train} = 10^{4+k/3}$ with $k = -1, ..., 7$. Here, $T_{test} = 10^6$ and $N = 10^6$.

Note that, depending on the value of n, the $T_{train} \to \infty$ limit we have mentioned is physically misleading. Indeed, we have no reason to believe that our results will hold for an arbitrary large neural network (for instance a network with $\gg N$ many neurons) and in the actual $T_{train} \to \infty$ limit, in case we used very long delay size, since that would allow to reconstruct the true deterministic dynamics, thanks to Takens theorem. Indeed, it is argued [52, 80] that the apparent nature of a system depends of the length of observations. In particular, one may not distinguish periodic dynamics with very large period, genuine chaos or a stochastic dynamics without enough data; pseudo-convergences of the Shannon entropy rate can hide the true nature of the system. This is our case, since we derived a stochastic system from equations which are deterministic by construction. Nevertheless, this is not a problem, since we are not aiming at reconstructing the "true" system but rather to provide a coarse grained characterization. If we are confident that the reconstruction is efficient, then we can say that $p \approx q$, so that, if $N \gg n \geq \tilde{n}$

$$C_\infty \approx h = \int d\mu(X^{(n)}) \int dx\, p_n(x|X^{(n)})\ \ln p_n(x|X^{(n)}), \tag{3.36}$$

which represents the Shannon entropy rate of the coarse grained dynamics in the Markovian approximation and quantifies how much information we are losing, in terms of one-step forecasts, by coarse graining. Therefore, we stress that this machine learning method, by providing an explicit - albeit approximated - expression for p_n, can overcome standard difficulties in quantifying the Shannon entropy rate of the coarse grained process.

An empirical confirmation supporting the physical interpretation of machine learning results will be given in spectrum analysis section.

3.4 Results and discussion

The ML approach introduced in sec. 3.2.3 aims at mimicking the dynamics produced by Eq. (3.5) by the analysis of time series of data, without any additional information about the generating model.

In this section we show the outcomes of this approach, and we compare them to the results of the purely frequentistic p-vMP method. In this way, on the one hand, it will be possible to compare the predictive power of our ML approach with respect to a "benchmark" simpler and more direct strategy and explore the advantages of a machine learning procedure.

As anticipated in Section 3.2.1, the main tools for our validation tests are the cross entropy and the analysis of Fourier frequency spectra of trajectories generated by the considered methods.

3.4.1 Cost function vs delay and model building

In the ML approach, the dependence of the test cost function on the delay n is expected to be, in principle, the result of two effects pointing in opposite directions. As n increases, the dimension of the input increases and so does the amount of training data needed: this effect alone would make the cost function an increasing function of n. Conversely, as n increases, more information can be extracted from past history of the dynamic variable and, hence, this would make the cost function a non-increasing function of n. In practice, as shown in fig. 3.9, with training length chosen $T_{train} = 10^6$, the first effect is barely noticeable and we can focus on the second. For small values of n, the function C_n decreases; after some value $\tilde{n}$ it reaches a plateau. Remarkably, the value of $\tilde{n}$, which is found to be $\simeq 8-10$, is very similar for all different number of maps we have examined, i.e. $N = 10^4,\ 10^5,\ 10^6,\ 10^7$.

If we exclude strong effects by either finite training length or network size, the plateau after $\tilde{n}(N)$ implies that no more information can be extracted by increasing the delay, given our resolution. This implies that

$$P_n(x_t|X_{t-1}^{(n)}) \approx P_m(x_t|X_{t-1}^{(m)}) \ \text{ if } m, n > \tilde{n}, \tag{3.37}$$

i.e. the system appears to be well described by a Markov process at our level of coarse graining. Hence, for any $n \geq \tilde{n}$, we define the Markovian machine learning model as the Markovian dynamics given by transition matrix Q_n as defined in sec. 3.2.3. This is, by construction, the best coarse grained model, in terms of one-step likelihood, we could obtain with our procedure. We will show later that this model is efficient in reproducing the dynamics on all timescales and for any N we have considered.

In order to obtain all figures in this section, we have trained a single neural network for each n and N. For each pair n, N, we have kept the best performing set of neural-network parameters found in 100 epochs (the number of epochs is, informally, the number of times the network is optimized over the same training set, see sec. 1.2.2. The performance has been evaluated with a test set of size $T_{test} = 10^6$ for all N.

In order to further confirm that the delay $\tilde{n}$ pertains to the dynamics and it is not a machine-learning construct, in fig. 3.10 we plot the (square root of) mean-square-error cost function (for $N = 10^6$), calculated with the deterministic network described above: the behaviour is the same and the plateau begins around the same values of n. Note that the plateau height is $\approx 10^{-3}$, which is the scale crossover to high-dimensional dynamics as highlighted by Grassberger-Procaccia analysis (see fig. 3.3(b), in which such a regime starts approximately at $L \approx 10^{-3}$).

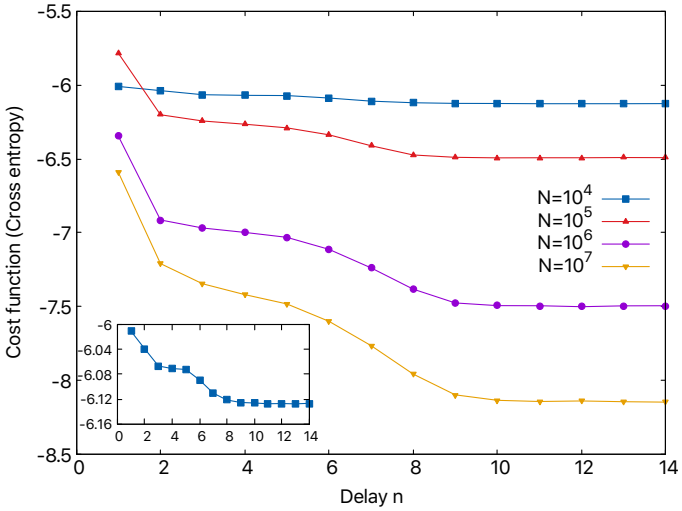

Figure 3.9: The cost function (cross entropy in eq. C_n (3.28), stochastic case), computed on a test set of 10^6, decreases with n for all N. After $\tilde{n} \approx 10$, a plateau is visible for any value of N. The training set size is $T_{train} = 2 \cdot 10^6$ for $N = 10^4$, 10^5, N^6 and 10^6 for $N = 10^7$. The inset shows a magnification of $N = 10^4$ curve.

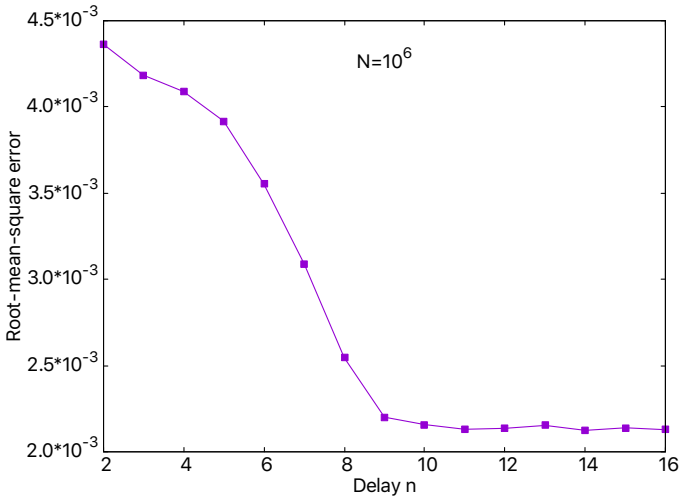

Figure 3.10: Deterministic neural network model. Root mean square error (square root of C_n^{det} in eq. (3.32)) decreases with n, shown for $N = 10^6$. After $\tilde{n} \approx 10$, a plateau is visible. Test set size $T_{test} = 50000$ while train set size $T_{train} = 2 \cdot 10^6$. Compare plateau height with scale crossover in fig. 3.3.

3.4.2 Spectral analysis

For evaluating the long-term reconstruction of the dynamics, we have compared the Fourier spectra of the original system against that of the proposed models; the results are shown in fig. 3.11. For the ML approach with $n = 2$ or $n = 3$ and the position-velocity Markov process, we get a fair agreement at high frequency f, and a correct detection of the peak at $f \simeq 10^{-2}$, if the number of maps is sufficiently large ($N \gtrsim 10^6$). For $N = 10^5$ these methods are not able to catch the oscillatory behaviour of the dynamics, as can be understood from the absence of the above mentioned peak in the spectrum; similarly, the low-frequency dynamics is poorly reproduced.

Conversely, for the stochastic neural-network based model, we can see that for $n \geq \tilde{n}$, good reconstruction has been achieved at all frequencies (see fig. 3.11). Notably, a delay of $n = 6$ seems

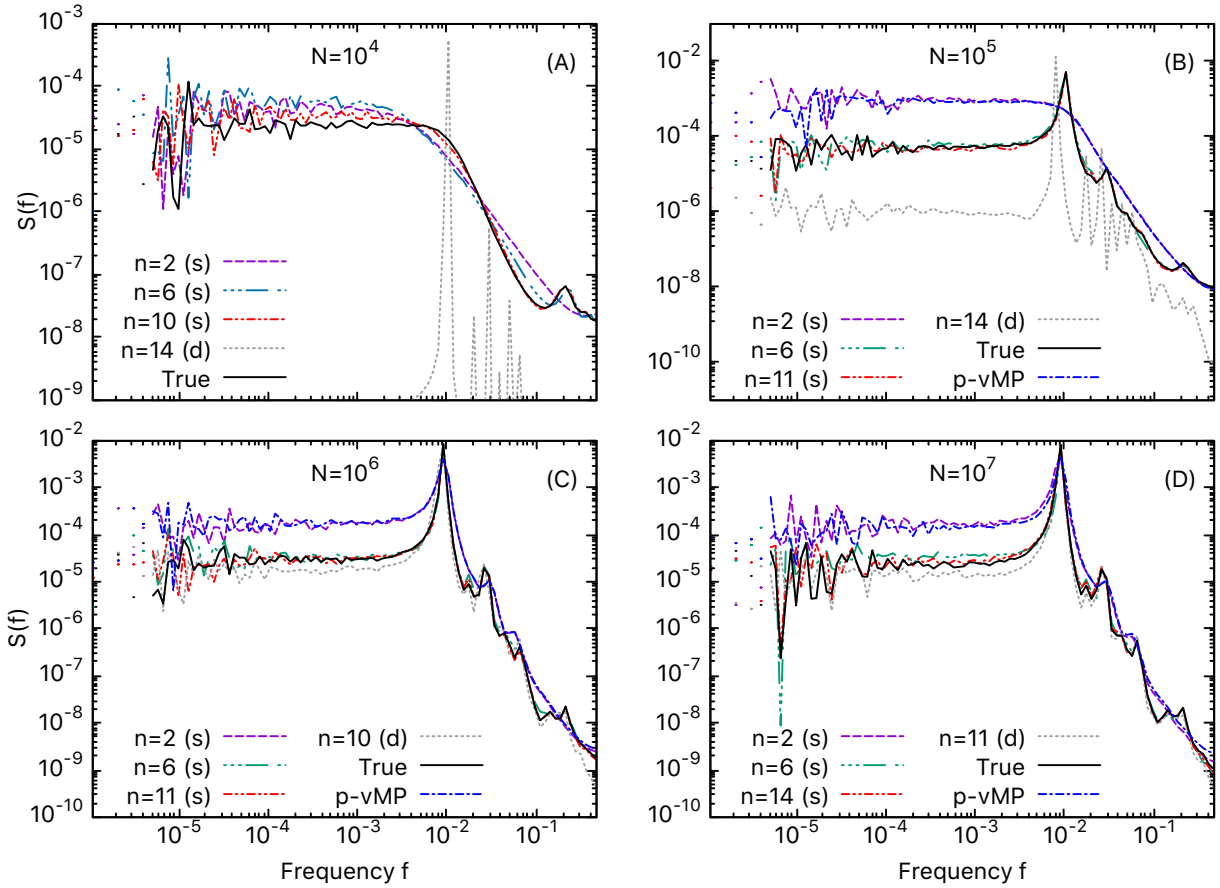

Figure 3.11: Fourier spectra for different numbers of maps $N = 10^4,\ 10^5,\ 10^6,\ 10^7$ in panels (A), (B), (C) and (D) respectively. In each panel, the true spectrum is compared to its counterparts from ML and p-vMP (but for panel (A), in which the p-vMP method is too unstable to allow for a trajectory reconstruction). Stochastic model (s): the frequency spectrum of the delay $n = 2$ neural network model almost perfectly overlaps with the position-velocity Markov process. They both miss the main frequency peak for $N = 10^5$, even if they are able to catch the main features of the dynamics for larger values of N (see sec. 3.4.4 for a discussion on this point). With delay $n \gtrsim 10$, the reconstruction is always good. Deterministic model (d): imperfect reconstruction, to some degree, can be seen at all scales even with $n \geq \tilde{n}$, implying that the stochasticity is a key feature; on the other hand, the deterministic model captures most frequency peaks. Moreover, it greatly improves with N and around $N = 10^6$. For all charts $T_{train} = T_{test} = 10^6$.

to be enough - if $N \geq 10^6$ - for the correct modelling of the low frequencies. For shorter delays, there seems to be not enough information and the model overestimates stochasticity, generating spurious long oscillations which are not observed in the original coupled maps dynamics. The deterministic models, on the other hand, always underperform their stochastic counterparts: most frequency peaks are captured, while low frequency oscillations are damped. Reasonably, a deterministic description efficiently captures quasi-periodic behaviours but fails to account for high-dimensional residual dynamics, which - in an effective description - acts as noise by widening peaks and generating rich low frequency dynamics. The case $N = 10^4$ is instructive: the original dynamics lacks any quasi-periodic behaviour and the model critically fails by collapsing unresolved noise into a single peak. Still, as N increases, the deterministic description greatly improves and, for $N = 10^6$, the performance is good enough to provide an alternative option to its stochastic counterpart, at least at large scales.

As a technical remark, in identical settings (n, N, T_{train} etc.), even the same short term performances do not always imply comparable long term identical results (we remind that the result of the training process is not deterministic). The variability is low for the stochastic nets but noticeable for the deterministic one. Since we were only interested in showing that the problem can be solved but not in machine learning algorithms per se, we only displayed performing cases in fig. 3.11. This is a very well known issue, and the underlying reason is probably the choice of hyper-parameters: a more efficient machine learning technique can most likely solve the issue, but it is beyond the scope of this work.

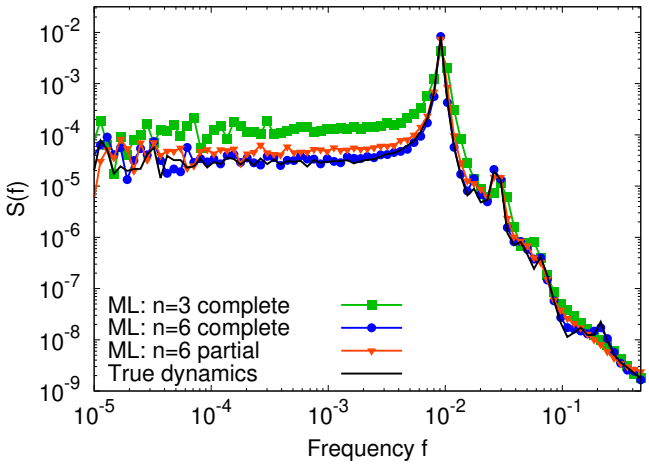

Figure 3.12: Proper choice of the variables. The spectrum of the actual dynamics (black solid line) is compared to those obtained with a ML approach with delay $n = 3$ (green squares) and $n = 6$ (red circles). Blue triangles show the result of a reconstruction obtained by considering only the "partial" embedding vector $(x_{t-1}, x_{t-2}, x_{t-6})$. The good agreement of the last curve with the original dynamics shows that the entire knowledge of the $n = 6$ delay vector is redundant. Parameters of the reconstruction: $n = 10$, $T_{train} = 10^6$, $T_{test} = 10^6$, $N = 10^6$.

3.4.3 Proper selection of the variables

The results discussed above clearly show that the stochastic ML approach is able to catch the essential low-frequency features of the original trajectories when $n \gtrsim 6$, a threshold which is seemingly independent of N. Apparently, this fact seems to suggest that the true dynamics needs, at least, a 6-dimensional description to be correctly reproduced. On the other hand, there is actually no valid reason to believe that *all* elements of the embedding vector are equally important to the reconstruction of the dynamics, and it is even possible, in principle, that the knowledge of some of them is redundant. Some hint on this point may be gained by a careful inspection of fig. 3.9. The decrease rate of the cross entropy is not constant along the curve: the inclusion of some specific elements seems to be particularly relevant. For instance, passing from $n = 5$ to $n = 6$ improves the reconstruction "more" than passing from $n = 3$ to $n = 4$ (if we use the cross entropy as a metric for the quality of the reconstruction).

It is natural to wonder what happens if one does not consider the *entire* embedding vector, i.e. if only some elements are passed to the neural network. Figure 3.12 shows the spectrum that is obtained if the vector $(x_{t-1}, x_{t-2}, x_{t-6})$ is used to infer the probability distribution of x_t: its agreement with the original dynamics is basically as good as that obtained with the complete $n = 6$

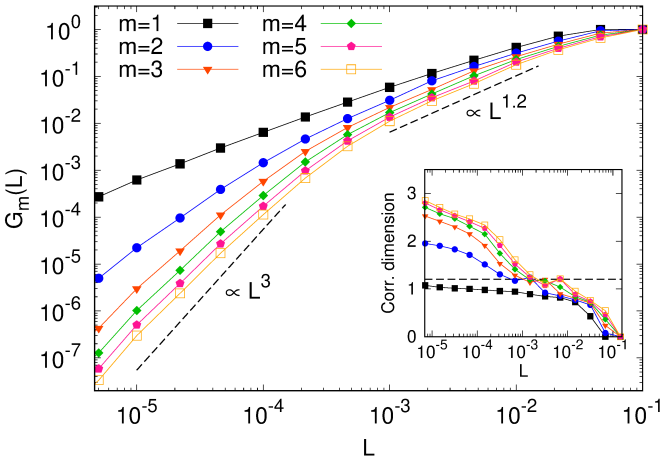

Figure 3.13: Grassberger and Procaccia analysis of a (deterministic) ML-generated trajectory. As in fig. 3.3(a), in the main plot we show the correlation (3.7) as a function of the lengthscale L, for different values of the embedding dimension; the inset reports the corresponding values of the attractor dimension. At least two different regimes can be individuated: a small-scale behaviour with dimension between 2 and 3 and a large-scale one, similar to that shown in fig. 3.3. Parameters of the reconstruction: $n = 10$, $T_{train} = 10^6$, $T_{test} = 8 \cdot 10^6$, $N = 10^6$.

embedding vector. This fact seems to suggest that the effective dimension of the attractor of the coarse-grained dynamics is actually close to 3 in the case under study.

3.4.4 Discussion

By solely looking at fig. 3.3, reporting the Grassberger-Procaccia analysis of the original trajectory, one may expect that a coarse-grained dynamics with dimension $\simeq 1.3$ would suffice to catch the macroscopic behaviour of the considered model. Indeed, as discussed in sec. 3.1, the attractor is found to have that dimensionality when the typical length scales of the dynamics are considered. Results shown in the previous sections, on the contrary, seem to depict a more complex scenario.

When we apply the stochastic ML analysis described in sec. 3.2.3 using low embedding vector dimension ($n = 2$, $n = 3$) the results are not so different from those obtained by a more straightforward, physics-inspired analysis as that introduced in sec. 3.2.2. This is quite reasonable, if one considers that the p-vMP approach relies on the determination of the conditional pdf $q(v_{t+1}|x_t, v_t)$, which embodies the same amount of information as the $q(v_{t+1}|x_t, x_{t-1})$ investigated by the ML method. Based on the above conjecture that the attractor of the effective macroscopic dynamics has dimension $\simeq 1.3$, we might expect that such second-order dynamics is enough to catch its main features. Figure 3.11(b), which reports the Fourier spectrum of the original and of the reconstructed dynamics for $N = 10^5$, clearly shows that this is not the case: indeed the p-vMP procedure and the ML approach with $n = 2$ even fail to catch the oscillating behaviour of the dynamics, and they miss the peak at $f \simeq 10^{-2}$.

The reason for this failure can be understood by looking again at fig. 3.4(b). In that case, the histogram of v_{t+1} based on the knowledge of x_t and v_t is given by the superposition of two peaks with opposite signs, a strong hint that, in that case, such knowledge does not completely identify the actual state of the system. This mismatch is due to the presence of "deterministic noise", whose amplitude is larger when N is smaller (see sec. 3.1), which brings to an overlap, in the phase space,

between two branches of the attractor characterized by opposite velocities (see fig. 3.4, main plot). Consistently, for values of N large enough ($N \gtrsim 10^6$), when the occurrence of cases as the one described in fig. 3.4(b) becomes negligible, the two-variables methods give a fair approximation of the dynamics, and they catch the oscillatory behaviour.

The above scenario seems to reinforce the initial conjecture that the macroscopic dynamics can be actually described by two variables only, and that the failure of our methods in the $N = 10^4$ and $N = 10^5$ cases is only due to a wrong identification of the state, i.e. to a wrong choice of the two variables to use for the description of the system. However, even when the number of maps is very large (e.g. $N = 10^7$), such methods fail to reproduce the low-frequency part of the Fourier spectrum, and we need to switch to a stochastic ML approach with larger n in order to recover sensible results at all scales. Very good results are observed for $n \geq 6$ (see fig. 3.11), so that one might be tempted to conclude that the actual dimension of the attractor is around that value. Bearing in mind the discussion in section 3.4.3, it is conversely clear that 3 dimensions should be enough to get a satisfying low-frequency description, once the variables are properly chosen.

An indirect evidence of the validity of this conclusion can be achieved by mean of the *deterministic* ML approach discussed at the end of sec. 3.2.3. We consider the case $N = 10^6$ and $n = 10$: as shown in fig. 3.11, this choice provides a very good reconstruction of the dynamics, even if the protocol employs a deterministic approach, because in the large-N limit the stochastic noise is negligible. We can thus expect that this neural network finds a proper way to approximate the true dynamics (whose real microscopic attractor has a dimension which is likely extensive in N) with a deterministic process involving 10 variables. The GP analysis can then be used to get a hint on the actual dimension of this deterministic dynamics (i.e. the minimum number of variables that would be needed to reproduce it). The results are shown in fig. 3.13: as expected, at the typical lengthscales of the macroscopic dynamics, we find again that the effective dimension is $\simeq 1.3$; at smaller scales, however, the attractor is found to have a larger correlation dimension, between 2 and 3, confirming our previous conjecture built on empirical bases (cfr fig.3.12).

By combining all previous observations, we can gain an insightful picture of the multi-scale structure, the memory and dimensionality of the system, at different level of coarse graining, constructed as Markovian processes for the embedding vectors.

- Delay $n = 2$ for $N \geq 10^6$. Approximately one dimensional (≈ 1.3) nearly periodic oscillations, involving a single frequency peak at $f \approx 10^{-2}$

- Delay $n = 6$ for $N \geq 10^5$. This level of coarse graining captures the "climate" of the system: low frequencies and main spectrum peaks are fully reconstructed. Minor discrepancies are visible in the high frequency spectrum reconstruction and can be fully appreciated with the cross entropy (cost function). The dimension of the system is approximately 3 and has a main deterministic contribution for $N \geq 10^6$.

- Delay $\tilde{n} \approx 10$. Both "climate" and "weather" (i.e. the short term dynamics) are reconstructed. To our available computational power, this level of coarse graining allows the best reconstruction of both low and high frequencies (as measured with cross entropy cost function). This works also for the "problematic" case $N = 10^4$, which lacks a clear deterministic backbone in the attractor.

- Delay $n \geq \tilde{n} \approx 10$. Residual effect of the high-dimensional deterministic dynamics. Information cannot be effectively extracted from it and should be modelled as noise in a coarse grained description. It becomes less relevant for $N \geq 10^6$. From machine learning we can estimate the Shannon entropy rate of this unresolved fast dynamics.

This non-trivial structure of the macroscopic dynamics was not evident from either equation or data.

3.5 Conclusions

The study of macroscopic dynamics from data in systems with many degrees of freedom is a challenging problem, and it has a wide range of applications, from turbulence to climate physics. In the work presented in this chapter, we have employed a ML approach to investigate the non-trivial macroscopic dynamics observed in a system of coupled chaotic maps; the outcomes have been compared to those of a more straightforward, physics-inspired analysis. Our aim was twofold: on the one hand, we wanted to test the ability of a stochastic ML approach in reproducing such non-trivial dynamics; on the other hand, we aimed at understanding whether the ML analysis could shed some light on the physics of the original model, for instance by determining the number of dimensions of its macroscopic dynamics.

Let us notice that the two purposes of understanding and forecasting are strongly interwoven. Predictions either fail or succeed: in the former case, it is hard to reach confident conclusions, since the failure may be either due to poor variable choice or to an inefficient machine learning approach; if predictions are successful, though, we can state that certain variables or information are *sufficient* (not necessary though) for building a model with a given degree of coarse graining. We can even control the level coarse-graining by employing different metrics. While indeed many powerful black-box tools exist to tackle forecasting problems, our choice of a direct technique allows for great control and easy manipulation of what variables and information (in our case, the delay vector) the network is using in building a certain model.

Our analysis shows that the stochastic ML approach can reproduce the dynamics with remarkable accuracy: when the embedding vector passed to the neural network has dimension $n = 2$, the results are comparable with those obtained by a direct reconstruction of the stochastic position-velocity dynamics. By increasing the length of the delay vector, the quality of the reconstruction gets even better, until the Fourier spectrum of the original and of the reproduced dynamics can be barely distinguished. This is indeed remarkable, since in the training process no direct information about the low-frequency behaviour of the dynamics is provided to the neural network. fig. 3.14 ketches a summary of the coarse-graining structure we have identified.

Our systematic analysis also gives some hints about the dimensionality of the macroscopic dynamics. The value of n at which the reconstruction starts to be reliable at every scale can be seen as an upper bound to the dimension of the attractor of the macroscopic dynamics. In this sense, the proposed procedure is not only able to determine an upper bound to the quantity of variables that are actually needed to describe the original trajectory. The idea of "compressing" the signal and only keeping relevant information is not new in machine learning (and specific machine learning approach to attractor size estimation can be found in literature [154]) but, unlike some automatic dimensional-reduction approaches (e.g. autoencoder neural networks), the technique employed in

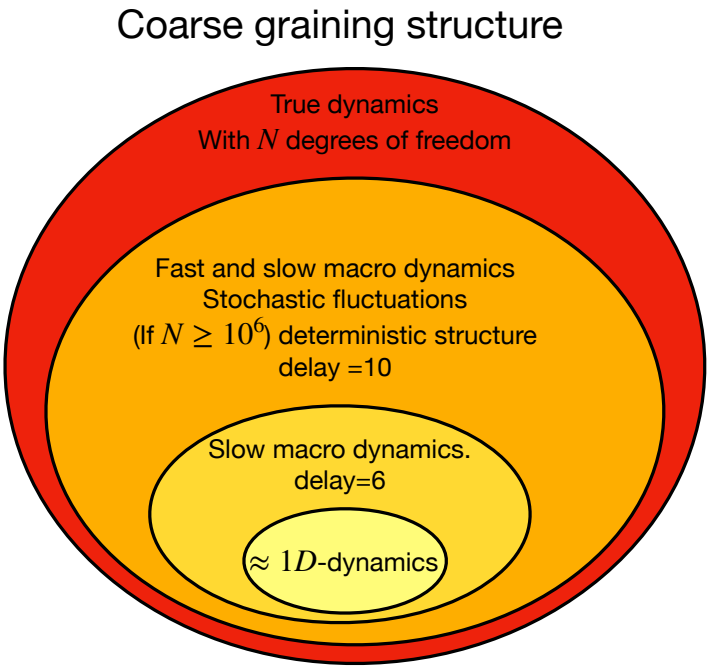

Figure 3.14: A sketch of the possible levels of coarse graining we have identified.

this work also provides some direct hint on *which* variables may be expected to be relevant in modelling the macroscopic dynamics.

We have thus shown that our approach can extract information ranging from multi-scale structure, memory effects, entropy rates, effective dimension and even variable selection. As a side note, let us also stress that once we are confident enough about the physical reconstruction, the network-based model can even be used as a data augmentation. In our case, we could produce a very long deterministic trajectory ($T = 10^7$) to run Grassberger-Procaccia algorithm, something that would have been computationally critical, e.g., for $N = 10^6$. In this perspective, an exploration-oriented machine-learning approach may probably be optimal when used alongside with standard physical techniques and powerful black-boxes tools for maximizing both insight and performance.

Appendix 3.A Two-band structure in model (3.1)

In this Appendix, we discuss some fundamental macroscopic properties of model (3.1), providing some insight into its qualitative behaviour discussed in sec. 3.1.

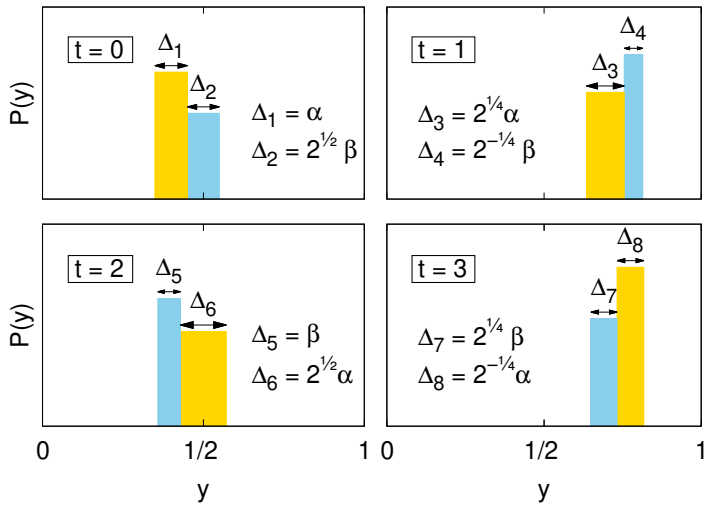

Figure 3.15: Scheme of the 4-steps dynamics obtained with model (3.1) under conditions (3.39). Each panel represents the distribution of the maps at a given time.

We can start by observing that the quantity $a(1-\varepsilon)$ (see eq. (3.1)) determines the "expansion rate" of the separation between between two units, say y_t^i and y_t^j, if they are, at a certain time t, both located at the same (right or left) side of $1/2$ in the the (0,1) interval. As a consequence, $a(1-\varepsilon)$ is a key quantity in describing the evolution of the distribution $p(y)$ of the maps. Specifically, we may focus on the cases in which $a(1-\varepsilon) = 2^{1/n}$, with $n \in \mathbb{N}$, $n \geq 0$: in these cases, it is usually possible to write piece-wise constant distributions which are mapped into themselves after n time steps when $N \to \infty$. A simple example is

$$p(y) = \begin{cases} 1/\alpha & \text{if} \quad \frac{1-\alpha}{2} < y < \frac{1+\alpha}{2} \\ 0 & \text{otherwise}\,, \end{cases} \tag{3.38}$$

which is mapped into itself after 2 steps if $a(1-\varepsilon) = \sqrt{2}$ and $\alpha = 2 - 4a^{-1} + 2a^{-2}$. A more relevant case is given by

$$a(1-\varepsilon) = 2^{1/4}\,. \tag{3.39}$$

Let us consider the distribution

$$p(y) = \begin{cases} \gamma/\alpha & \text{if} \quad \frac{1}{2} - \alpha - \frac{\beta}{\sqrt{2}} < y < \frac{1}{2} - \frac{\beta}{\sqrt{2}} \\ (1-\gamma)/\alpha & \text{if} \quad \frac{1}{2} - \frac{\beta}{\sqrt{2}} < y < \frac{1}{2} + \frac{\beta}{\sqrt{2}} \\ 0 & \text{otherwise}\,, \end{cases} \tag{3.40}$$

which is illustrated in the first panel of fig. 3.15. Parameter $\gamma \in (0,1)$ weighs the relative areas of the two rectangular "bands" of $p(y)$, whose widths are determined by the constant parameters α and β. For every couple of parameters (a, ε) satisfying the relation (3.39), there exist infinite triplets of parameters (α, β, γ) such that the distribution of maps $p(y)$ is mapped into itself after 4

steps of the dynamics, as sketched in fig. 3.15.

To see this, one has to observe that, if at times $t = 2$ and $t = 4$ the center of the right band is placed at $y = 1/2$, then all rectangular bands are mapped into new rectangles by the dynamics: this is sufficient for preserving the two band structure. Since there are two constraints but three free parameters, there is a residual degree of freedom and there are infinite distributions $p(y)$ which satisfy the 4-steps recurrency condition shown in fig. 3.15. Notice that, as a consequence, the expected value of y, i.e. $Z = \int dy\, p(y)$ has an exact period 4 and, therefore, if we consider $N \gg 1$ maps, the macroscopic variable $z_t = \sum_i y_t^i$ (eq. (3.3)) has the same period, up to finite size effects.

The previous argument provides a rationale for the quasi-period 4 of the macroscopic dynamics of z_t (eq. (3.3)) associated to (3.39) with parameters a, ε used in this work (see (3.4)). Indeed, $a\,(1-\varepsilon) = 1.19$ almost satisfies condition (3.39), since $2^{1/4} \simeq 1.1892$. Hence, since the system given by parameters (3.4) is very "close" – in parameter space – to another system with macroscopic period 4, it is no surprise that it is quasi 4-periodic itself[3]: hence, we can rationalize the smooth-looking and oscillatory behaviour of the filtered variable x_t (see (3.5)). Furthermore, the states visited in the dynamics given by (3.4) closely resemble eigenstates from dynamics (3.39).

[3]The system is periodic, in the $N \gg 1$ limit, only if we look at macroscopic variables or at distributions, but not at the state vector $\{y_t^i\}$.

Part II

Optimal behaviours in multi-agent biological systems

Chapter 4

Behaviour optimization methods for multi-agent systems

In the first part of this thesis, we have presented some applications of machine learning techniques to dynamical systems. There exists a particular class of non-conventional dynamical systems which, in recent years, are being investigated by physicists with increasing interest: multi-agent systems. Such "agents" should be intended either as biological organisms – bacteria, genes, plants, animals etc. – or artificial agents – robots, drones etc. A first prominent feature shared by many of such systems is their out-of-equilibrium nature; this is the case of "active matter" [213], an umbrella term for self propelled agents, whose description has to account for the agent-wise injection of energy into the system. Another important observation is that multi-agent systems are naturally connected to the idea of optimization: it is fairly obvious that artificial agents, built for a certain task, should behave in such a way to achieve their goal according to some optimality criterion (e.g. minimization of execution time, of energy saving etc etc). On the other hand, also biological agents generally adopt behaviours which have been selected by evolution – and which are therefore fit in an evolutionary sense – or that have been learned within a lifetime, as in the case of higher organisms such as birds and mammals or any animal equipped with a (even simplified) nervous system. The ideas of fitness or goal-driven behavior are naturally associated to optimization, from a formal point of view. Therefore, in this framework, it is an interesting challenge to uncover the biological role of natural agents' behavior, and to understand whether it is optimal in some sense; from a physical perspective, a particularly interesting scenario is the collective behaviour emerging from individual agents' interactions, since it involves connecting the "single-particle" description with the "macroscopic" one.

Amongst the techniques that may be employed in approaching these ideas, we may mention reinforcement learning [233], optimal control theory [75] and game theory [185]. In this introductory chapter, we will provide a minimal framework for the articles which are presented in the following chapters, concerning two cases of multi-agent dynamics, a cooperative one [38] (chapter 6) and an adversarial one [37] (chapter 5).

4.1 Optimization in dynamical systems: reinforcement learning and optimal control theory

Optimal Control (OC) theory and Reinforcement Learning (RL) are dynamical optimization frameworks, which allow to find the best way to influence a system in order to achieve a certain goal. The class of problems they may be applied to is rather general, and ranges from economics and engineering to robotics and biology [209]. We may mention three possible tasks as suitable examples (that will be used throughout this section) for the application of OC and RL techniques: finding the shortest path on a map, winning a chess match, finding the best command sequence to make a glider soar. All these examples have in common a well defined goal (e.g. finding the path, winning, soaring), a specific and repeated possibility of intervening (e.g. moving in a certain direction, playing a certain move, modifying the asset of the wings) in order to influence, at least partially, the state of the system (e.g. one's location, chessboard configuration, wing configuration+altitude+weather conditions+...).

While RL and OC theory are closely related, they differ in terminology, applicability and conceptual background: OC is the mathematical framework which deals with formalizing and solving this kind of problems, either analytically or numerically; reinforcement learning is a branch of machine learning which, as opposed to supervised or unsupervised learning, is grounded on the idea of learning to solve such tasks from trials and errors. OC is, in some sense, the mathematical description of planning a strategy; RL is the conceptualization of the way animals learn [233], i.e. by remembering which of their own behaviours benefited them and which did not. See fig. 4.2 for a biologically inspired representation of the two different approaches.

There exists a rich zoology of dynamical optimization problems; without indulging too much in classifying all possible settings, we may mention a few important distinctions. Any non trivial problem is set in an *environment* (RL term) or *system* (OC term) which, in general, evolves according some dynamics, either stochastic or deterministic. The environment may either have discrete states (e.g. chessboard configuration) or continuous states (e.g. the position on a map). Likewise, time may be discrete (e.g. turns in chess) or continuous (e.g. moving in a map). Another important distinction is the time horizon: the task may last for an infinite amount of time i.e. it could have an infinite time horizon (continuing task, in RL notation); or it could last for a finite amount of time (which may be a random variable) i.e. it could have finite time horizon (*episodic task*, in RL terminology). This latter case may be due to the existence of a maximum time before an episode ends (e.g. a glider may have a fixed amount of time to soar before landing) or due to the existence of a *terminal state*; in the latter case, if the system reaches a terminal state, the episode ends (say checkmate or a stalemate in the game of chess; or when an agents reaches a certain target location, such as in [203]).

In order to design a strategy to achieve a certain goal, one has to define an optimization problem. In general, one introduces a cost function to minimize (OC) or a reward to maximize (RL); the reward system should be designed in such a way that an optimal (in the cost/reward sense) strategy will fulfill the desired goal. Notice that there may exist different reward systems to achieve a specific goal and some may make learning faster than others. Costs/rewards may either be collected in time or given at the end and of an episode but, in either cases, the fundamental idea is not to adopt a short-sighted perspective but a global one: the agent should not be interested in the immediate

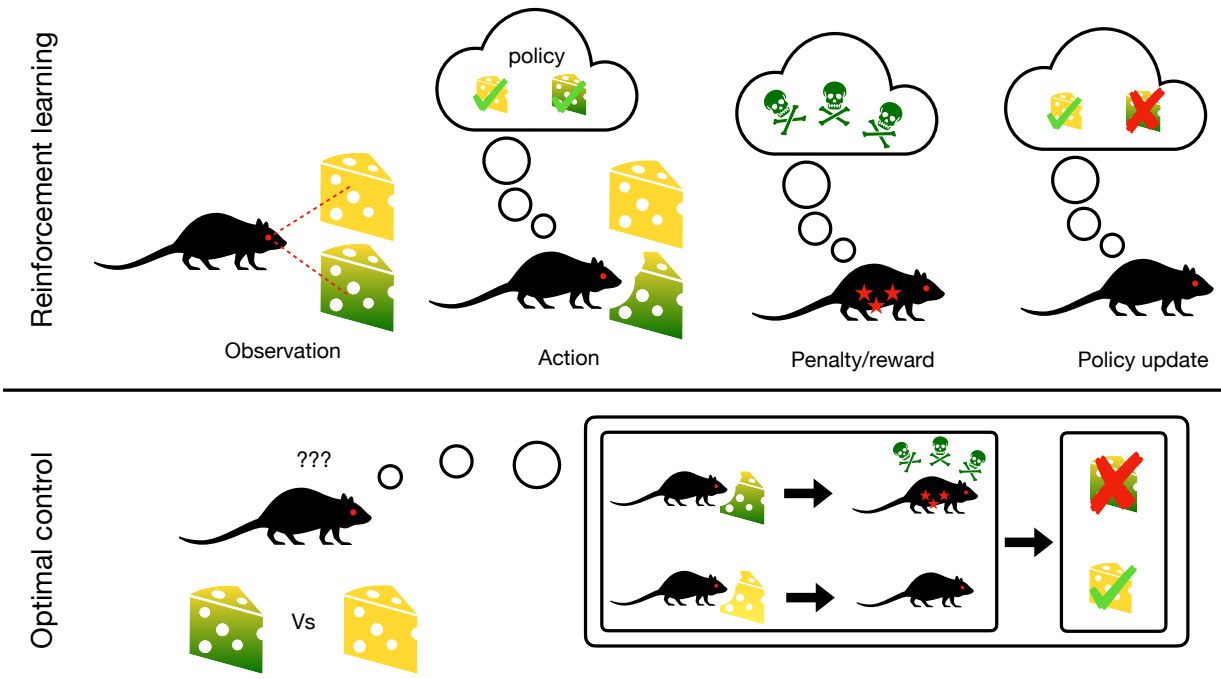

Figure 4.1: Pictorial representation of reinforcement learning and optimal control. In the former case, the optimal behaviour is learned from trials and errors: the agent observes the environment; it picks an action according to its current knowledge of cause-effect relations between actions and rewards; it receives a penalty/reward; finally, it updates its behaviour i.e. it updates its belief about the most appropriate action given a certain environmental observation. In the case of OC, the problem of maximizing the reward is solved by planning the best strategy by assuming perfect knowledge of the consequences of each action.

reward obtained after a certain action but, rather, in understanding the long term consequences of each choice, so that the global reward is maximized. The nature of the global reward depends on the setting. In the case of finite horizon, the rewards may stack in time, so that in the end the sum of all rewards should be maximized. It is also possible to introduce a time-discount [233], which is the "better an egg today than a hen tomorrow" principle: future rewards have lesser value than the today's because the future is uncertain[1]. If the horizon is infinite, one generally maximizes the average reward obtained in time.

While many problems may be solved with either techniques, the preferred approach depends on how much information is available. Any non-trivial problem is set in an environment or system, which has a certain *state* and evolves according to a certain law or dynamics. The setting in which both the state of system and the dynamics are exactly known is called MDP (it will be briefly described in the RL section, for conceptual clarity): in principle, this would allow to design an optimal strategy, either numerically (e.g. with dynamic programming [26, 233]) or analytically, in the OC framework. The setting in which the dynamics is known but the state is not is called POMDP: generally, the state should be inferred through observations with Bayesian updating of the beliefs [142, 174]. For instance, consider the problem of the glider: it may be able to measure local wind velocity and temperature, but the surrounding state of the atmosphere may not be directly measureable; however, if the measured wind speed has been low for some time, it may be concluded that no strong winds are to be expected anytime soon. In the case in which the

[1]It is equivalent to assuming that at each time there is a certain probability that the episode will end, but we will not go into details.

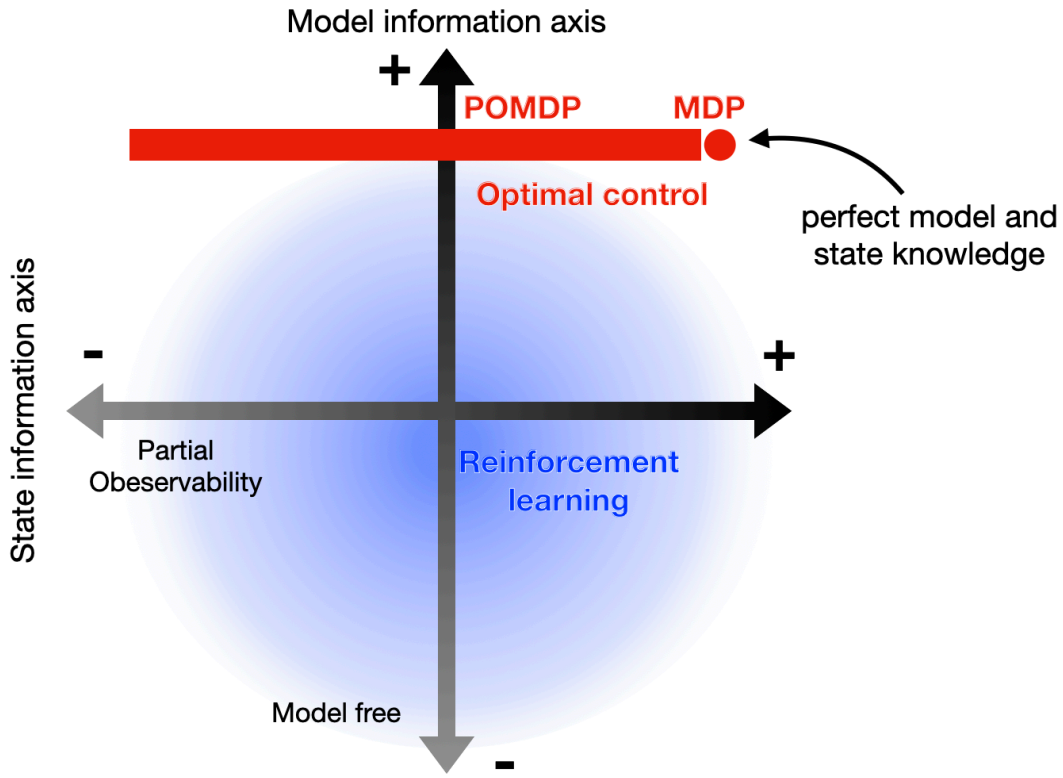

Figure 4.2: A conceptual scheme (inspired by figure 1 from [204]) for dynamical optimization based on available information. Markov Decision Process (MDP) correspond to perfect knowledge about both the model and the state of the system and may be studied with optimal control theory. Likewise, Partially Observable Markov Decision Process (POMDP) are characterized by imperfect state knowledge and may be studied with OC and Bayesian updating. Other cases, where analytic or systematic approaches are harder or unfeasible, are mainly the domain of RL.

dynamics is unknown, a *model free* approach is needed; OC formalism cannot be applied and RL may be used instead, since RL allows to select the best option in a given circumstance based on previous experience (i.e. from the collected information concerning past observations, actions and rewards), even without any knowledge of the environment dynamics. Consider again the glider problem: the exact equations governing flight might not be known but, in the same way a pilot may use their intuition to fly a plane, a RL algorithm may infer the appropriate manoeuvres to keep an aircraft airborne. Notice that, in some cases, even MDPs are better studied with RL rather than OC. In the chess game, for instance, both the state and the dynamics are known, but the number of possible game trajectories is so large that it is impractical to solve the game through OC, while RL, which focuses on a subset of strategies, has been shown to be extremely performing [209] . This example illustrates how RL may be used to face the *curse of dimensionality* in dynamical optimization. See fig. 4.2 for a scheme about applicability of optimization techniques depending on the available information.

Table 4.1 summarizes the RL-OC definition correspondences between definitions which have or will be presented.

In the following section, we give a brief introduction to the theory of OC. For practical reasons,

Reinforcement learning	Optimal control
reward r	cost c
environment s	system x
maximize	minimize
action a	control u
state-value function V	cost-to-go V

Table 4.1: RL and OC definition correspondence.

we will only address the continuous states-continuous time case in the optimal control section, while focusing on the discrete-discrete scenario in the reinforcement learning section.

4.2 Optimal control theory: some key ideas

To illustrate the main ideas of the theory of optimal control, we consider a simplified scheme of a stochastic dynamical system with continuous states and finite time horizon, governed by the stochastic differential equation

$$dx = f(x,u)\,dt + \sqrt{2D}\,d\xi, \tag{4.1}$$

where ξ is white noise and D is a diffusivity constant. The state of the system is $x(t)$ and u(t), usually dubbed *control*, is the only variable the agent can directly act upon in order to alter the system evolution. We define a policy[2] π as a function $\pi(x(t)) = u(t)$ which maps the state x into a control u that, in general, may be time-dependent. We can define the expected cost-to-go at time t and associated to a policy π as

$$V_\pi(t,x) = \mathbb{E}\left[V^{end}(x_T) + \int_t^T d\tau\; c(\pi(x(\tau)), x(\tau))\middle| x_t = x\right], \tag{4.2}$$

where T is the time horizon of the episode; V^{end} is a function which assigns a cost to a terminal state; $c(u,x)$ is the cost per time unit; $T < \infty$ is the time horizon. In such context, the optimal cost-to-go is given by

$$V_*(t,x) = \min_\pi V_\pi(t,x), \tag{4.3}$$

and may be achieved through one or more *optimal policies* which may be computed with OC theory in different ways.

The first way to obtain such policies is to write the so called Bellman equation [209], which, in the continuous-time formulation is called Hamilton-Bellman-Jacobi equation [250]. In a nutshell, one may notice that eq. (4.3) may be written as

$$\begin{aligned} V_\star(t,x) &= \min_{\{u(s)\}_{s=t}^{t+dt}} \left\{\mathbb{E}\left[\int_t^{t+dt} c(u(t),x(t)) + V_\star(t+dt, x(t+dt))\middle| x(t) = x\right]\right\} \\ &\approx \min_{u(t)} \left\{dt\; c(u(t),x(t)) + \mathbb{E}\left[V_\star(t+dt, x(t+dt))|x(t)=x\right]\right\}. \end{aligned} \tag{4.4}$$

Observe that, according to stochastic calculus[3], $\mathbb{E}\left[V_\star(t+dt, x(t+dt))|x(t)=x\right] - V_\star(t,x) \approx dt\,(\partial_t +$

[2] Note that this is slightly different from the reinforcement learning definition.

[3] For instance, one may use Ito lemma.

$\mathcal{L}^\dagger)V_\star(t,x)$ where $\mathcal{L} = -\partial_x f + D\partial_x\partial_x$ is the evolution operator[4] associated to eq. (4.1). Hence, by using the above expressions, one can easily derive the Hamilton-Bellman-Jacobi equation

$$-\partial_t V_\star = \min_u \left\{ c + \mathcal{L}^\dagger V_\star \right\}; \tag{4.5}$$

with boundary conditions $V_\star(T,x) = V^{end}(x)$. Notice that one can obtain an even simpler equation in the deterministic[5] case. By solving the Hamilton-Bellman-Jacobi, one may derive the optimal policy either analytically or, for instance, with dynamic programming [26], which is an efficient computational strategy that builds up the optimal strategy by iteratively solving optimization sub-problems from the end backward to the initial condition.

A different way of approaching the problem of OC is to employ the Pontryagin principle [208]. Essentially, one solves the equations in a larger space in which configuration and control trajectories are independent and their connection – the dynamics i.e. (4.1) or the Fokker Planck equation – is imposed as a constraint. For instance, in our problem with finite horizon, we may write a Lagrangian function

$$\mathcal{H} = \int_0^T dt \int dx \left[\rho(x,t) \left(c(x,u,t) + \delta(t-T) V^{end}(x) \right) + \phi(x,t) \left(\partial_t - \mathcal{L} \right) \rho(x,t) \right], \tag{4.6}$$

where ϕ is formally a Lagrange multiplier. In order to solve the problem, we should impose stationarity of $\mathcal{H}$ with respect to ρ and u (actually, we should minimize in u),

$$\frac{\delta \mathcal{H}}{\delta u} = 0 \qquad \frac{\delta \mathcal{H}}{\delta \rho} = 0, \tag{4.7}$$

with boundary conditions $\rho(x,0) = \delta(x - x_0)$, $\delta\rho(x,0) = 0$ and $\delta\phi(x,T) = 0$. With the above procedure, a necessary condition for optimality is obtained. Since this approach will be described in detail (albeit in a slightly different case) in chapter 6, we do not go into further details. However, it is worth mentioning the formal connection with the Bellman equation. Note that $\frac{\delta \mathcal{H}}{\delta \rho} = 0$ implies

$$c + \left(\partial_t - \mathcal{L}^\dagger \right) \phi = 0 \tag{4.8}$$

and $\phi(x,T) = -V^{end}(x)$. Now, assume we have found the optimal control u_* and call $\mathcal{L}_\star$ the associated evolution operator: then, the distribution $\rho_\star(x,t)$ corresponding to such control satisfies $\left(\partial_t - \mathcal{L}_\star \right) \rho_\star = 0$ for $0 < t < T$. Thus, if we average eq.(4.8) with $\rho_\star$ and integrate between 0 and T,

[4] In the sense that $\partial_t \rho = \mathcal{L}\rho$ where ρ is the probability density function of x, which is the dynamical variable evolving according to eq. (4.1). Operator $\mathcal{L}^\dagger = f\,\partial_x + D\partial_x\partial_x$ is its adjoint under proper boundary conditions.

[5] Start from the first equality in (4.4). Since the system is deterministic, remove the expectation value $\mathbb{E}$ and observe that $V_\star(t+dt, x(t+dt)) - V_\star(t,x(t)) \approx dt\,(\partial_t + f \cdot \partial_x) V_\star(x(t), u(t))$. One immediately obtains

$$-\partial_t V_\star = \min_u \left\{ c + f \cdot \partial_x V_\star \right\}.$$

Alternatively, one may set $D = 0$ in the stochastic equations to get the same result.

we get[6]

$$\phi(x_0,0) = -\int_0^T dt \int dx\, c(u_\star, x; x_0)\rho_\star(x,t;x_0) + \int dx\, \rho_\star(x,t;x_0)\, V^{end}(x) = -V_\star(0,x_0). \quad (4.9)$$

Therefore, as we will find out in a specific case in chapter 6, there is a connection between equation (4.8) for the multiplier ϕ and the Hamilton-Bellman-Jacobi equation (4.5). Equivalently, there is a connection between the multiplier and the cost-to-go.

While equations (4.5) and (4.8) are non-linear, there exists a class of problems for which the Hamilton-Bellman-Jacobi equation can be linearized with either formulations (4.5) or (4.8). Such possibility has been highlighted for continuous time problems [136] and put in a broader context by Todorov [243]. Specifically, one should assume functional compatibility between the cost and the dynamics. For instance, consider the case of a Brownan particle whose deterministic part, the velocity, is the control itself ($f = u$ in eq.(4.1); this is a simplified example, see [243] for a throughout discussion)

$$dx = u\, dt + \sqrt{2D}\, d\xi, \quad (4.10)$$

and the cost is chosen as a quadratic function of the control

$$c = \alpha\, u^2/2 + q(x) \quad (4.11)$$

where $\alpha > 0$ is some constant and q is a function. Assuming eqs. (4.10) and (4.11), the Largrangian (4.6) reads

$$\mathcal{H} = \int_0^T dt \int dx \, \left[\rho\,(q + \alpha\, u^2/2) + \phi\, \partial_t\rho + \phi\, \partial_x(u\,\rho) - D\,\phi\,\partial_x^2\rho\right], \quad (4.12)$$

and the corresponding stationarity conditions (4.7) become

$$u = \frac{1}{\alpha}\partial_x\phi \quad (4.13)$$

$$\partial_t\phi = D\,\partial_x^2\phi - q + \frac{1}{2\,\alpha}(\partial_x\phi)^2. \quad (4.14)$$

By introducing the Hopf-Cole transformation

$$\phi = 2\,\alpha\, D\, \ln z, \quad (4.15)$$

one can directly check that the Hamilton-Bellman-Jacobi equation (4.14) becomes linear:

$$\partial_t z = D\,\partial_x^2 z - \frac{q}{2\,\alpha\, D}\, z, \quad (4.16)$$

where z is called desirability [243]. Notice that, under this formulation, the control reads

$$u = 2\, D\, \partial_x \ln z, \quad (4.17)$$

[6]

$$\int_0^T dt\, dx\, \rho_\star\, (-\partial_t - \mathcal{L}_\star^\dagger)\, \phi = \int_0^T dt\, dx\, \phi\, (\partial_t - \mathcal{L}_\star)\, \rho_\star - \int dx\, \phi(x,T)\, \rho_\star(x,T) + \int dx\, \phi(x,0)\, \rho_\star(x,0).$$

Now observe that $(\partial_t - \mathcal{L}_\star)\,\rho_\star = 0$ and use $\rho(x,0) = \delta(x - x_0)$ and $\phi(T,x) = -V^{end}(x)$.

which means that the dynamics correspond to a gradient ascent of the desirability function.

The quadratic part of the cost (4.11) may be interpreted as the cognitive cost of deviating from the default (no control) strategy [243]. Indeed, assume the state of the system is $x(t)$ at time t and consider the probability distribution of $x(t+dt)$ under the evolution given by (4.10): as long as $dt \ll 1$, we may write

$$P_u(x(t+dt) - x(t) = dx|x(t)) \propto \exp\left[-\frac{(dx - u\,dt)^2}{2\,D\,dt}\right]. \tag{4.18}$$

On the other hand, if no control is applied, we have

$$P_0(x(t+dt) - x(t) = dx|x(t)) \propto \exp\left[-\frac{(dx)^2}{2\,D\,dt}\right]. \tag{4.19}$$

It is easy to see that the Kullback–Leibler divergence[7] between the two distributions is

$$KL[P_0||P_u] \propto dt\ u^2. \tag{4.20}$$

Therefore, the quadratic cost may be interpreted as the distance between the distributions obtained via the controlled and uncontrolled dynamics.

4.3 Reinforcement learning: some key ideas

4.3.1 From the Bellman equation to function approximation techniques

Reinforcement learning is a branch of machine learning which formalizes the idea of learning how to perform a certain task through trials and errors. As it was explained in the sec. 4.1 about similarities and differences with OC, RL is suitable for problems featuring uncertainty on the dynamics of the environment (see also fig. 4.2), very high dimensional systems or extremely complicated tasks. In order to provide an example, we may imagine placing a robot with limited sensory capabilities in an unknown environment and wanting it to collect information but, at the same time, to return before its batteries run out. That would be a complicated and sloppy task, unsuitable for mathematical rigor but still manageable with heuristic approaches and with a RL approach. A RL algorithm, by learning effective probabilistic cause-effect relations between specific actions and long-term outcomes, could engineer an efficient solution to the task. With these motivations in mind, in the following, we will give some fundamental ideas about RL and, finally, explain in more detail the specific algorithm[8] that will be used in chapter 5.

In order to explain some basic ideas, it is useful to introduce MDPs, which are useful to illustrate the basic concepts and notation and, also, to understand the link between OC and RL. The core idea is to consider an agent which is interacting with the environment (see fig. 4.3): the environment (an umbrella term for anything not contained within the "mind" of the agent) may change in time with its own laws but it is also influenced by the agent's actions. At the same time, the agent is assumed to be able to observe the state of the environment. And, finally, the agent may receive rewards or

[7] Note the Kullback–Leibler may still be employed as a cost for discrete systems and, while the cost itself ceases to be quadratic, most remarkable properties are preserved [243].

[8] As noted by Sutton [234], it is not really possible to disentangle theory from algorithms, when discussing RL.

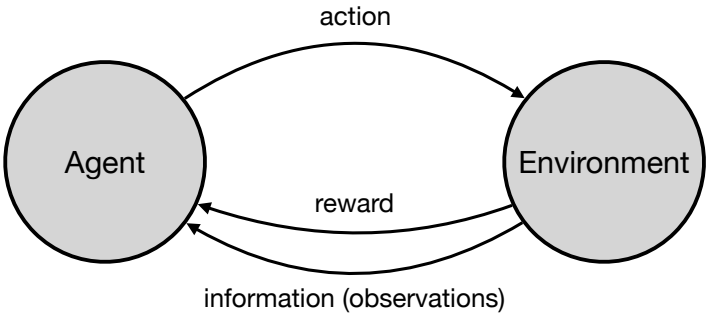

Figure 4.3: Idealize RL scheme: an agent impacts the state of the environment and modifies it with its actions; in return, it receives information on the state of the system and rewards to judge the quality of its actions.

penalties from the environment, depending on the state on the environment itself and/or on the actions taken. Formally, the state of the environment at time t is described by a variable s_t which, in the MDP context, is fully known to the agent. The agent may use any information extracted from s_t in order to pick an action a_t, according to its own policy π: $\pi(a_t|s_t)$ is the probability of selecting a certain action a_t given a certain state s_t. Such action may influence the dynamics of the environment, which changes from state s_t to s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$; at the same time, the agent receives a reward $r(s_t, a_t, s_{t+1})$ (the reward could be stochastic but, for the sake of notation simplicity, we will ignore such a possibility). We assume that the goal of the agent is to maximize the expected total reward, measured as

$$J = \mathbb{E}\left[\sum_{t=1}^{T} r_t\right]. \tag{4.21}$$

As in the OC section, we will limit the description to the finite time horizon case, i.e. episodic tasks in RL jargon.

It is useful to introduce the *state-value function* $V(s)$ (equivalent to the cost-to-go in optimal control theory) and *action-state value function* $Q(s, a)$. The former is defined as the expected future reward given a certain state and a certain policy

$$V_\pi(s) = \mathbb{E}\left[\sum_{k=0} r_{t+k} \middle| s = s_t, \pi\right]. \tag{4.22}$$

The latter has a similar definition but the conditioning is done both with respect to the action and the state:

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{k=0} r_{t+k} \middle| s = s_t, a = a_t, \pi\right]. \tag{4.23}$$

These two quantities are related in the following ways:

$$V_\pi(s) = \sum_a \pi(a|s)\, Q_\pi(s, a) \tag{4.24}$$

and

$$Q_\pi(s, a) = \sum_{s'} P(s'|s, a)\, [r(s, a, s') + V_\pi(s')]. \tag{4.25}$$

Using these definitions, one can say that policy $\pi \geq \pi'$ (policy π performs equal or better than π') if $V_\pi(s) \geq V'_\pi(s)\ \forall s$. It can be shown that, in MDP with finite many states, there exists at least one optimal policy $\pi_\star$, which satisfies $\pi_\star \geq \pi\ \forall \pi$ (we call $V_\star$ and $Q_\star$ the associated value functions). Such *optimal policy* solves a Bellman equation, either

$$V_\star(s) = \max_a \sum_{s'} P(s'|s,a) \left[r(s,a,s') + V_\star(s') \right] \tag{4.26}$$

or

$$Q_\star(s,a) = \sum_{s'} P(s'|s,a) \left[r(a,s,s') + \max_{a'} Q_\star(s',a') \right]. \tag{4.27}$$

Note that such equations show that the optimal policy may always be chosen as deterministic, since, under policy $\pi_\star$, after observing s, the best action is always $a \in \arg\max_{a'} Q_\star(s,a')$. The underlying reason is that the environment is non-adversarial. Conversely, in adversarial situations (typical of multi-agent systems), where the environment itself has an antagonistic strategy which may change over time, the optimal policy may be strictly stochastic since not all games admit equilibria with deterministic policies [185].

In the MDP case, where an optimal policy is known to exist, techniques such as dynamic programming can be employed; if information is incomplete, though, the problem becomes more challenging. Specifically, if we assume that the state s_t is not known and only some observation $o_t = o_t(s_t)$ of the system may be observed, we clearly cannot solve the Bellman equation directly, and we enter the domain of *partial observability*. The MDP becomes a POMDP and the state s_t may only be inferred from the history of actions and observations $\{o_t, a_t, o_{t+1}, a_{t+1}, ...\}$ through Bayesian updating (since such approaches will not be used in this thesis, we will not expand this topic any further, see for details [142, 174]). Conversely, the state s_t may be known, while the laws $P(s_{t+1}|s_t, a_t)$ may be not: this is the *model free* scenario. The most general and realistic scenario combines a model-free setting with partial observabilty (see fig. 4.2).

In order to learn an optimal strategy, exploration is need so that, through trials and errors, one may determine the best action for any possible observation. However, one cannot really explore while focusing on collecting the reward according to the best known policy: by definition, exploring implies attempting new strategies, accepting the risk to fail and, at least temporarily, deviating from current behaviour. More precisely, one may need to leave behind some locally optimal behaviours in search for a better optimum, but without even knowing if there is any at all. Therefore, there exists a well known trade-off between exploitation and exploration. In this context, two classes of algorithms are available: *on policy* or *off policy*. In the former case, the policy one learns is also the one uses to explore the environment; in the latter, an agents explores through the *behaviour policy* while learning the optimal one, the *target policy*.

As an insightful example of the way RL works in a model-free framework, it is worth providing a short illustration of the ϵ-greedy Q-learning[9], which is a simple off-policy algorithm to learn the action-state value function through an iterative procedure. For illustrating the problem, assume we know the action-state value function $Q_\star$ of an optimal policy. With any behaviour policy π, we may

[9]For convergence, one should require [233, 262] that $\sum_{t=0}^{\infty} \eta_t = \infty$ but $\sum_{t=0}^{\infty} \eta_t < \infty$.

generate trajectories $\{(s_t, r_t, a_t)\}$ and define a quantity $\hat{Q}$ which is updated as:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \eta \, [Q_\star(s_t, a) - \hat{Q}(s_t, a_t)]. \tag{4.28}$$

where η is the learning rate. One may guess that $\hat{Q}$ would converge to $Q_\star$ for any a, s: however, the target $Q_\star$ is precisely what we do not know and, therefore, we need to replace it with an estimator. The target used in Q-learning is the biased estimator $r_t + \max_a \hat{Q}(s_{t+1}, a)$ (note that $\mathbb{E}[r_t + \max_a Q_\star(s_{t+1}, a)|s_t, a_t, \pi_\star] = Q_\star(s_t, a_t)$) so that the update rule becomes

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \eta \, [r_t + \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)]. \tag{4.29}$$

Since we are comparing estimates from $t + 1$ and t, this technique is called *temporal difference* and it stands at the basis of a large class of very efficient RL algorithms [233]. Moreover, note that the algorithm uses a previous estimate of a state (corresponding to s_{t+1}) to update the Q-value of another one (corresponding to s_t): this is called *bootstrapping* and is another key idea in reinforcement learning. One can verify that, at the fixed point, $\hat{Q}$ converges [262] to $Q_\star$ as given by the Bellman equation (4.27) (notice that eq. (4.28) itself is a way to approximate the Bellman equation (4.27)): the optimal greedy policy $a = \arg\max_{a'} Q_\star(s, a)$ is therefore learned as the target policy. As for the behaviour policy, the ϵ-greedy formulation balances exploration and exploitation: one exploits the greedy policy $a = \arg\max_{a'} \hat{Q}(s, a)$ with probability $1 - \epsilon$ and explores other possibilities by picking a random action with probability ϵ.

As powerful as Q-learning (or similar algorithms) might be, it is a tabular method, which involves constructing a matrix containing the values of action-state pairs: this approach is unfeasible in several cases, for instance when the environment is described by continuous variables (though one may successfully attempt tiling techniques) or when it is simply too large. For this reason, functional approximation techniques should be employed. They generally consist in using parametrized functions for approximating V, Q or even the policy itself π. Consider, for instance, the state value function $V(s)$. One may employ a basis of functions $\{w_i(s)\}$ and write the exact expansion $V = \sum_{i=1}^{\infty} \alpha_i \, w_i(s)$. Since an infinite sum may not be employed, one may truncate the series $V = \sum_i^n \alpha_i \, w_i(s)$ and keep n elements only. In this approximation scheme, $\{w_i(s)\}_{i=1}^n$ are called features or percepts. Alternatively, one may use a neural network to approximate $V(s)$. Another advantage of the function approximation approach is that it is suitable for partially observable environments[10]. In the following subsection, we will outline some important ideas about function approximation, while informally presenting a specific algorithm which is used in chapter 5.

4.3.2 Natural-actor critic

In this subsection, we give an introduction to an on-policy function approximation scheme, known as natural actor critic [27, 100, 202], adapted to a partially observable environment [118].

As in the previous cases, the goal is to maximize the average total reward per episode J from eq. (4.21), which we may write as

$$J = \sum_s \mu_\pi(s) \sum_a \pi(a|o(s)) \, Q(a, s) = \mathbb{E}_\pi[Q] \tag{4.30}$$

[10]Even without using memory for Bayesian updating as in partially observable Markov decision processes.

where $\mu_\pi(s)$ is frequency of visit of s under policy π. Since the true state s may be non-observable, it is convenient to write the action-observation value function $q(o,a)$, i.e. the expected future reward given a certain observation o, defined as

$$q(o,a) = \sum_s P(s|o)\, Q(a,s) \tag{4.31}$$

where $P(s|o)$ is the probability of being in s while observing o (in principle, $P(s|o)$ might depend on the policy π). We can also write the frequency of visit of s as

$$\mu_\pi(s) = \sum_o P(s|o)\, P_\pi(o), \tag{4.32}$$

where $P_\pi(o)$ is the probability of observing o while following policy π. Hence

$$J = \sum_o P_\pi(o) \sum_a \pi(a|o)\, q(a,o) = \mathbb{E}_\pi[q]. \tag{4.33}$$

In order to approximate the optimal policy, the idea is to define a class of functions $\pi_\xi(a|o)$ which depend on a set parameters ξ. Such parameters may be learned through a gradient ascent algorithm. Ignoring partial observability for a moment, a fundamental result for function approximation techniques is known as *policy gradient theorem* [233, 234] and can be stated as follows:

$$\nabla_\xi J = \sum_s \mu_\pi(s) \sum_a \pi(a|s)\, Q(a,s)\, \nabla_\xi \ln \pi(a|s) = \mathbb{E}_\pi[Q\, \nabla_\xi \ln \pi]. \tag{4.34}$$

Notice that this theorem does not require a particular parametrization of the policy, neither a specific dependene of the policy on the state. Therefore, eq. (4.34) remains true even if we assume to work with such class of functions for which $\pi(a|o(s))$, i.e. in the case of partial observability. In this sense, one does not even need to introduce a specialized notation or formalism for the partially observable setting. For the sake of clarity, though, we proceed by explicitly accounting for the presence of observations: hence, by assuming that π only depends on o and by using (4.31) and (4.32), we may write (see [15] for a more rigorous approach)

$$\nabla_\xi J = \sum_o P_\pi(o) \sum_a \pi(a|o)\, q(a,o)\, \nabla_\xi \ln \pi(a|o) = \mathbb{E}_\pi[q\, \nabla_\xi \ln \pi]. \tag{4.35}$$

As many machine learning algorithms, actor-critic algorithms employ an estimator for $\nabla_\xi J$ instead of updating parameters with exact gradient ascent rule. As in Q-learning, the estimator used in this case allows an agent to learn the optimal policy online – while playing. At each time-step t, the agent observes the environment (observation o_t) and chooses an action a_t according to its current policy π. It receives its reward r_t and then observes the environment again, getting o_{t+1}; the update $\xi(t) \mapsto \xi(t+1)$ is based on the idea of temporal difference.

It has been shown [233] that, in order to reduce the variance of the stochastic gradient ascent, it is possible to rescale the action-value function in equation (4.35) by subtracting a "baseline" function which must be independent of a. This shift does not change the value of $\nabla_\xi J$, but it is useful in making algorithms faster. The natural baseline is the observation-value function $v(o)$, i.e. the expected future reward given a certain observation o. In a nutshell the idea is to substitute q

with the so-called advantage function A defined as

$$A(a, o) := q(a, o) - v(o), \tag{4.36}$$

which measures the advantage of using action a when observing o with respect to the value of o undert the current policy. Therefore

$$\nabla_\xi J = \mathbb{E}_\pi[A \, \nabla_\xi \ln \pi]. \tag{4.37}$$

Since $\mathbb{E}_\pi[r_t + v(o_{t+1})|o_t, a_t] = q(a_t, o_t)$, the quantity

$$\hat{A}_t = r_t + v(o_{t+1}) - v(o_t) \tag{4.38}$$

is an estimator for the advantage function A which exploits again the idea of temporal differences (TD). Hence, if $\hat{v}$ is an estimator - a function approximation - of v, then

$$\delta_t = r_t + \hat{v}(o_{t+1}) - \hat{v}(o_t), \tag{4.39}$$

called TD error, is also an estimator for A. Therefore, it is possible to use the following update step

$$\xi(t + 1) = \xi(t) + \eta_A \, \delta_t \, \nabla_\xi \ln \pi \tag{4.40}$$

where η_A is the "actor" learning rate. The rationale is that the update (4.40) moves, on average, in the direction of the gradient (4.37) but with a possible bias. By definition, to compute δ_t we need to learn an approximation $\hat{v}$ of the function v, which is, in general, not known. The full actor-critic algorithm consists in simultaneously learning the policy π (actor) and the state-value function $\hat{v}$ (critic), whose parameters are called κ; these two steps, corresponding to policy improvement and policy evaluation, are carried in parallel. Let us now focus on the critic step: in order to estimate $\hat{v}$, we have to minimize

$$C = \frac{1}{2}\mathbb{E}_\pi[(v - \hat{v})^2]. \tag{4.41}$$

This minimization can be carried with a gradient descent, where the gradient may be computed as

$$\nabla_\kappa C = \sum_o P_\pi(o)[v(o) - \hat{v}(o)] \, \nabla_\kappa \hat{v}(o). \tag{4.42}$$

Therefore, we could update the parameters κ along a trajectory with the rule

$$\kappa(t + 1) = \kappa(t) - \eta_C \, (\hat{v}(o_t) - v(o_t)) \, \nabla_\kappa \hat{v}(o_t) \tag{4.43}$$

with η_C being the critic learning rate. Since the target function $v(o_t)$ is obviously not known, we can replace it with $\hat{v}(o_{t+1}) + r_t$, exploiting temporal differences once again and replace $\hat{v}(o_t)$ with $r_t + \hat{v}(o_{t+1})$. Hence, we get

$$\kappa(t + 1) = \kappa(t) - \eta_C \, \delta_t \, \nabla_\kappa \hat{v}(o_t). \tag{4.44}$$

Note that, since the parameters κ now appear in the target too, this is no longer a true gradient

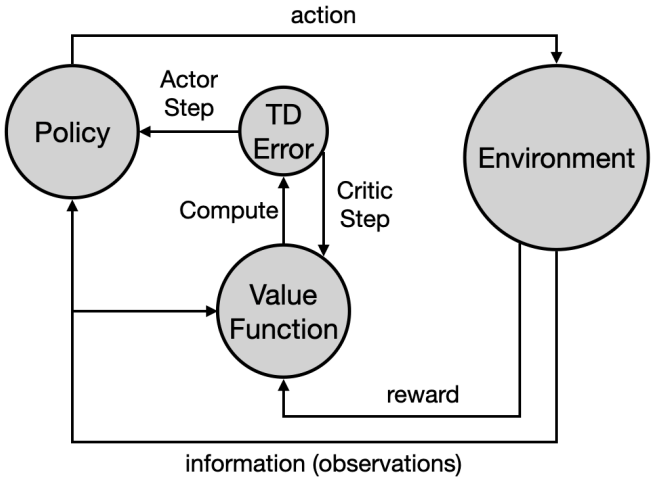

Figure 4.4: Actor-critic scheme. The environmnt provides information (an observation) and a reward. With the reward, the observation and the previous estimate of the value function, it is possible to compute the temporal difference error (eq. (4.46)). The TD error allows both to update the value function estimate (critic step, second eq. in (4.45)) and the policy (actor step, first eq. in (4.45)). Finally, a new action is selected with the updated policy.

descent[11], but it is a case of *semi-gradient* algorithm.

In order for the whole algorithm to converge, the update process of the value function $\hat{v}$ should be much faster than that of the policy π ($\eta_C \gg \eta_A$), so that the value-function updating is always close to convergence w.r.t. to the policy updating timescale.

By combining previous observations, one gets the full algorithm. As we mentioned, the agent, after observing o_t, picks action a_t according to its current policy π. Then it observes the "consequences" o_{t+1} and receives its reward r_t. Finally it updates both its state value function estimation parameters $\kappa(t)$ and its current policy parameters $\xi(t)$ as

$$\begin{cases} \xi_{bi}(t+1) = \xi_{bi}(t) + \eta_A \ \delta_t \ \nabla_{\xi_{bi}} \ln \pi(a_t|o_t) \\ \kappa_j(t+1) = \kappa_j(t) - \eta_C \ \delta_t \ \nabla_{\kappa_j} \hat{v}(o_t) \end{cases} \tag{4.45}$$

with

$$\begin{cases} \delta_t = -v(o_t) & \text{if } s_{t+1} \text{ is terminal} \\ \delta_t = r_t + \hat{v}(o_{t+1}) - \hat{v}(o_t) & \text{otherwise.} \end{cases} \tag{4.46}$$

A sketch of the actor-critic scheme can be found in fig. 4.4.

What was described so far is a standard actor critic algorithm. In the following, which is going to be a little technical, we proceed in outlinig the an improved version of this algorithm, the natural actor critic [27, 202], which can be obtained by replacing standard gradients with natural ones. *Natural gradients* are covariant gradients in the sense of differential geometry and play an important role in information geometry and machine learning [6]. A natural metric in the space of parameters is the Fisher-information [202], which, in the space of policy parameters ξ, is given by

$$G(\xi) := \sum_s \mu_\pi(s) \sum_a \pi(a|o(s)) \ \nabla_\xi \ln \pi(a|o(s)) \otimes \nabla_\xi \ln \pi(a|o(s)), \tag{4.47}$$

[11] We have replaced $(\hat{v}(o_t) - v(o_t)) \ \nabla_\kappa \hat{v}(o_t)$ with $(\hat{v}(o_t) - \hat{v}(o_{t+1}) - r_t) \ \nabla_\kappa \hat{v}(o_t)$. Since $\hat{v}(o_{t+1})$ depends on the same parameters as $\hat{v}(o_t)$, we have introduced an additional dependence on the parameters, which breaks the gradient structure of the update rule.

which can easily be rewritten in terms of observations alone. In general, the Fisher information measures the sensitivity of parameter estimation to observations and it is a fundamental quantity[12] in information and estimation theory [68]. The Fisher information may be obtained [244] as the (symmetric) second order term in the expansion of Kullback-Leibler divergence between two arbitrary probability distributions p_ξ and $p_{\xi+d\xi}$ with slightly different parametrizations ξ and $\xi + d\xi$, respectively:

$$D[p_\xi \| p_{\xi+d\xi}] = \mathrm{tr}[d\xi \otimes d\xi \underbrace{\mathbb{E}[\nabla_\xi \ln \pi_\xi \otimes \nabla_\xi \ln \pi_\xi]}_{G(\xi)}] + o(\|d\xi\|^2), \tag{4.48}$$

where $\otimes$ is the tensor product. Since $D[p_\xi \| p_{\xi+d\xi}]$ does not depend on the parametrization, then $\mathrm{tr}[d\xi \otimes d\xi\, G(\xi)]$ is a scalar with respect to reparametrization[13] and, therefore, $G(\xi)$ transforms as metric tensor in the sense of differential geometry (one may check other metric properties are satisfied as well).

In our case, if we define a trajectory as $\tau = ((s_0, a_0, r_0), (s_1, a_1, r_1), ...)$ and $\mathcal{P}_\xi(\tau)$ as its probability (given the policy parameters ξ), then eq. (4.47) is equivalent to [6]

$$G(\xi) := \mathbb{E}_\xi[\nabla_\xi \ln \mathcal{P}_\xi \otimes \nabla_\xi \ln \mathcal{P}_\xi]. \tag{4.49}$$

In practice, in order to switch to the natural-gradient actor-critic algorithm, we need to replace standard gradients with natural ones:

$$\xi(t+1) = \xi(t) + \eta_A\, \delta_t\, G^{-1}\, \nabla_\xi \ln \pi(a|o). \tag{4.50}$$

In the previous equation, $G(\xi)^{-1}\, \nabla_\xi$ is the natural/covariant gradient while the remaining terms correspond to the standard "actor" part of the algorithm (see eq. (4.40)).

Since computing and inverting the Fisher information matrix G can slow down the process, it is convenient to adopt a variant of the algorithm which relies on an auxiliary variable $g = \{g_{ai}\}$, which is updated alongside with the actor and critic steps as:

$$g(t+1) = g(t) - \eta_G\, [\nabla_\xi \ln \pi(a_t|o_t) \otimes \nabla_\xi \ln \pi(a_t|o_t)]\ \cdot g(t) + \eta_G\, \delta_t \nabla_\xi \ln \pi(a_t|o_t). \tag{4.51}$$

Since we want to use g to estimate the gradient $\mathbb{E}[\delta_t \nabla_\xi \ln \pi]$, the previous update should essentially be allowed to converge with fixed policy parameters ξ: hence, we require $\eta_A \ll \eta_G$. On the other hand, since the update (4.51) contains δ_t, the critic convergence should be faster than that of g: hence, we require $\eta_C \gg \eta_G$. Overall $\eta_A \ll \eta_G \ll \eta_C$. The average update of g is described by the equation

$$g(t+1) = g(t) - G\, g(t) + \eta_G\, \mathbb{E}[\delta_t \nabla_\xi \ln \pi] \tag{4.52}$$

whose fixed point is

$$g^* = G^{-1} \mathbb{E}[\delta_t \nabla_\xi \ln \pi]. \tag{4.53}$$

[12] For instance, it appears the important Cramér-Rao inequality [68], which provides a lower bound to the variance of unbiased estimators.

[13] Consider a diffeomorphism f such that $\xi' = f(\xi)$ and $\xi' + d\xi' = f(\xi + d\xi)$ and define $\tilde{p}_{f(\xi)} = p_\xi\ \forall \xi$. Moreover, define $A = \frac{\partial \xi'}{\partial \xi}$. Since, by construction, $D[p_\xi \| p_{\xi+d\xi}] = D[\tilde{p}'_\xi \| \tilde{p}_{\xi'+d\xi'}]$ and $d\xi' = a\, d\xi$, then

$$\mathrm{tr}[d\xi \otimes d\xi\, G(\xi)] = \mathrm{tr}[d\xi' \otimes d\xi'\, G'(\xi')] \implies G_{ij}(\xi) = A^l_i A^m_j G'_{lm}(\xi').$$

Alternatively, covariance may be directly deduced from the r.h.s of (4.48).

i.e. the quantity needed for updating the actor parameter (see eq. (4.50)). Therefore, the following update rule is equivalent (in the sense that they have the same stable fixed points) to (4.50)

$$\begin{cases} \kappa(t+1) = \kappa(t) - \eta_C \; \delta_t \; \nabla_\kappa \hat{v} \\ g(t+1) = g(t) - \eta_G \left[\nabla_\xi \ln \pi(a_t|o_t) \otimes \nabla_\xi \ln \pi(a_t|o_t) \right] g(t) + \alpha_G \, \delta_t \nabla_\xi \ln \pi(a_t|o_t) \\ \xi(t+1) = \xi(t) + \eta_A \; g(t+1), \end{cases} \tag{4.54}$$

but it has the advantage of not requiring direct computation of the Fisher information matrix. This algorithm is used in chapter 5, in an adversarial multi-agent context with partial observability.

Chapter 5

Reinforcement learning for pursuit and evasion of microswimmers at low Reynolds number

Aquatic organisms can detect moving objects in their environment by sensing the induced hydrodynamic disturbances [236, 245, 248]. Such an ability is crucial in navigation, especially in murky or dark waters as in the case of the blind Mexican cavefish [155]; it is also important in interactions among individuals both from the same and different species, such as in the case of prey-predator interactions. For this purpose, fishes have developed the lateral line, a mechanosensory system sensitive to water flows and pressure gradients [32, 135, 175]. At smaller scales, some planktonic microorganisms possess antennae and setae [139] to sense hydrodynamic disturbances in low-Reynolds-number environments, and, for instance, they can detect predators and preys [71, 139]. In engineering, bioinspired mechanosensors that can sense the hydrodynamic fields are used in underwater robots employed for search and recovery, surveillance and ship inspection [84, 141]. Thus, understanding how to exploit hydrodynamic cues is of interest both for explaining animal behavior and for designing smart underwater robots.

Even in abstract terms, i.e. ignoring specific mechanisms developed by aquatic organisms or deployed for robots, the problem of pursue-evasion in microswimmers guided by hydrodynamic cues poses substantial difficulties that are rooted in the physics of the ambient medium. At low Reynolds numbers[1], flow disturbances are generally weak and characterized by symmetries [103] that induce ambiguities about the location (and direction of motion) of the signal source, especially when it is located far from the receiver [139, 236, 248]. Moreover, hydrodynamics has dynamical effects, since

[1]The Reynolds number weighs the relative importance between inertial and viscous forces in a fluid and characterizes its degree of turbulence. It is defined as

$$Re = \frac{U\,L}{\nu},$$

where U is the large-scale flow velocity, L is the characteristic length-scale and ν is the kinematic viscosity. Consider a flow $\boldsymbol{u}$: the (momentum) Navier-Stokes equation for an incompressible ($\nabla \cdot \boldsymbol{u} = 0$) fluid is

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\,\boldsymbol{u} - \nu\,\Delta \boldsymbol{u} + \Delta p = \boldsymbol{f}$$

where p is the pressure and $\boldsymbol{f}$ is the forcing. By dimensional analysis, one can verify that, in the $Re \ll 1$ limit, the equation reduces to the Stokes equation

$$-\nu\,\Delta \boldsymbol{u} + \Delta p = \boldsymbol{f}.$$

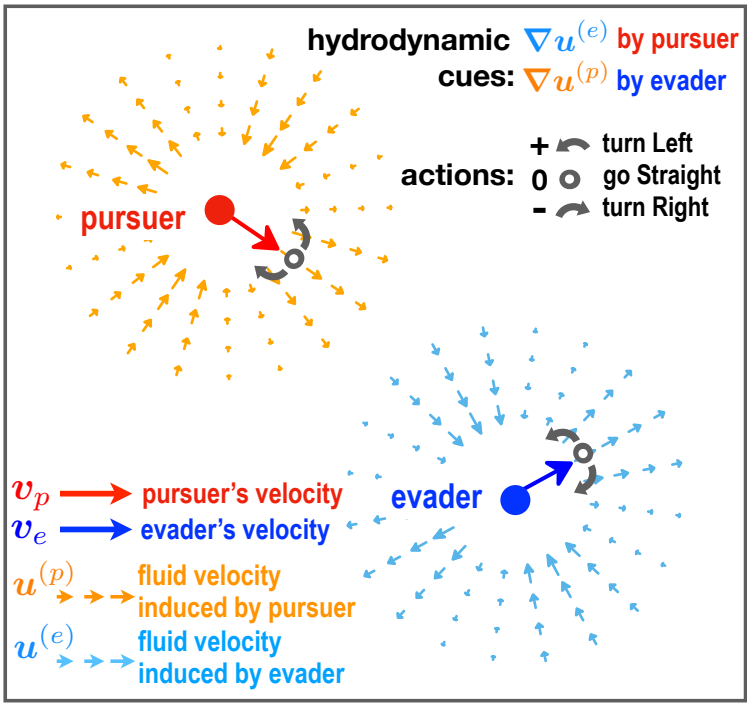

Figure 5.1: Model illustration. The pursuer (p, red)/ evader (e, blue) goal is to min/maximize the time their distance reaches the capture value R_c within a given time horizon. Agents move in the plane with speed $v_{p/e}$; every τ time-unit they choose to maintain or turn left/right their heading direction. By swimming an agent generates a velocity disturbance, $\boldsymbol{u}^{(p/e)}$, which drags the other and offers a cue to the other agent on the relative position and orientation via its gradients, $\boldsymbol{\nabla u}^{(p/e)}$.

the disturbances generated by one swimmer alter the motion of others. For the aforementioned reasons, hydrodynamic interaction between moving agents may be complex due to the combined effects of dynamical interactions and sophisticated sensory mechanisms.

In the work [37] presented in this chapter, we investigate possible prey-predator strategies in a low Reynolds aquatic environment [211], where we assume that swimming agents may only rely on hydrodynamic clues which provide only partial information about the state of system, and, in our case, about the location of the other swimming agent. We frame this problem in the broader context of pursuit evasion literature [178] but, unlike other studies, which mainly focus only on the prey/predator strategies while fixing the behaviour of the opponent, we employ a game theoretic framework [111]: by designing a zero sum game, we are able to study competing strategies between non-coopertating agents. Inspired by recent applications of RL to hide-and-seek simulated contests [16, 58], in order to identify appropriate chasing and escaping adversarial strategies at low Reynolds number, we employ Multi-Agent Reinforcement Learning [233] as a general model-free framework. The potential of RL for navigation in complex and dynamic fluid environments has already been demonstrated in various tasks, both in simulations [4, 29, 64, 66, 173, 212, 214, 253] and experiments [177, 215].

In the following sections, we will first introduce the setup and then the results.

5.1 Pursuit and evasion games for microswimmers

5.1.1 Modelling microswimmers

Dynamics and hydrodynamics

Hydrodynamics at low Reynolds numbers [211] is simpler that its complete counterpart. It can be shown through dimensional analysis that, in this limit, non-linear terms in the Navier-Stokes equations are vanishing (see note from previous page): consequently the effective equations are linear and Green functions formalism can be employed. Furthermore, it can be assumed that the timescales of disturbance propagation are much faster than the typical timescales in which agents move. Hence, due to scale separation, fluid perturbations can be approximated to travel with infinite speed and the medium can always be considered in a steady state.

We model the two agents or microswimmers as point-like oriented discoid objects, whose motion is affected both by self propulsion and by hydrodynamic effects. Each agents generates a flow while swimming, is dragged by the local field flow velocity and rotated by the vorticity generated by its opponent. In the absence of an external flow (as assumed here for the sake of simplicity) the coupled dynamics of the two agents is the following:

$$\dot{\boldsymbol{x}}_e = v_e\, \boldsymbol{n}(\theta_e) + \boldsymbol{u}^{(p)}(\boldsymbol{x}_e) \tag{5.1}$$

$$\dot{\theta}_e = \frac{1}{2}\omega^{(p)}(\boldsymbol{x}_e) + \Omega_e \tag{5.2}$$

$$\dot{\boldsymbol{x}}_p = v_p\, \boldsymbol{n}(\theta_p) + \boldsymbol{u}^{(e)}(\boldsymbol{x}_p) \tag{5.3}$$

$$\dot{\theta}_p = \frac{1}{2}\omega^{(e)}(\boldsymbol{x}_p) + \Omega_p, \tag{5.4}$$

where e/p =evader/pursuer (i.e. prey/predator); $\boldsymbol{x}_\alpha$ and θ_α are the position and the orientation of agent $\alpha = e, p$ respectively; v_α and Ω_α are the linear and angular velocities of agent α, respectively; $\boldsymbol{u}^{(\alpha)}$ and $\boldsymbol{\omega}^{(\alpha)} = \nabla\times\boldsymbol{u}^{(\alpha)}$ are the velocity and vorticity flow felt by β and generated by α, respectively. There could be other possible flow terms (because of the linearity, the combined effect of flows from different sources is given by their superposition): the interaction of an agent with its own disturbance and an external field; we can neglect the former in our coarse grained description (it would be important in a closed domain) and we did not include the latter for simplicity. Notice that, angular and linear speeds are the only quantities that an agent may directly control; for simplicity, we assumed that v_α is fixed.

In this framework, the disturbances generated by small swimming agents are commonly modelled as force dipoles: while we will not indulge in detailed derivations from microscopic hydrodynamic equations (see [77]), it is easy to outline the intuitive reasons why this is true. First, we should say that there are different kinds of swimmers [70]: pushers, pullers, neutral and intermediate cases. We focus on the first two possibilities. Pushers are organisms like sperm cells or *Escherichia coli* which are capable of self propelling by pushing the medium behind their rear, for instance with a flagellum, a tail or a bundle of filaments, so that a flow is generated from their back; at the same time, their front end pushes the fluid forward, generating a flow from their head. If we assume incompressibility, since there are no sinks or sources in the medium, there must be an incoming flow from the sides of their body. Pullers, e.g. *Chlamydomonas reinhardtii* algae, on the other hand, swim by dragging water from their front end to their sides with appropriate appendices such

as flagella; as a result, water flows in from back and front, and out from the sides. The previous argument allows to rationalize that either flows – of pullers and pusher – may appear as force dipoles, with opposite signs. A more rigorous[2] way to understand the origin of the force dipole is that, in overdamped conditions, there are not forces acting on the fluid since the mass of an agent is negligible. However, it is easy to guess that such modelling is only accurate at large distances from the source: exact dipoles have a singular point (the source) and the near field features of the flow must depend both on the shape of the swimmer and on the details of its propulsion mechanism. In our work, we opted to model our agents as pushers[3] and, since we do not want to fine tune our description after a specific setting, we decided to treat the disturbances generated by either agents as pure force dipoles, which is the only universal backbone of the perturbations. Note that we do not need to worry about the singularity, since agents never move below a given distance R_c, which will be introduced later.

A second important point is the number of dimensions of the water environment. While the most natural choice would be to set the problem in three dimensions, it is common practice to study such problems in the plane in order to make results easier to interpret. Unfortunately, 2D hydrodynamics is well known to be rather pathological. For instance, the fundamental solution generated by a force point F in the fluid – the Stokeslet – corresponds to a perturbation whose strength increases with distance, in the opposite direction of F itself. Due to issues like this, it is not uncommon to use three dimensional solutions of swimming-induced flows projected in a plane in which agents are constrained to lay [173]. Nontheless, it must be noted that the dipole solution in 2D is not very pathological, it preserves most fundamental qualitative features of its three dimensional counterpart (though it decays as $\sim 1/r$ instead of $\sim 1/r^2$) and preserves incompressibility[4]. For these reasons, we opted to keep a purely two dimensional description.

With these premises, we can derive the field generated by a swimmers moving in direction $\boldsymbol{n}$ located in the origin; the field generated from any other point in space can be obtained by applying a translation. The dipole can be obtained with a pair of point forces pushing the fluid in opposite directions. The fundamental solution associated with a point force $\boldsymbol{F} = F\,\boldsymbol{n}$, the Stokeslet, solves the linear Stokes equation

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \nu\,\Delta \boldsymbol{u} - \boldsymbol{\nabla} p = \boldsymbol{F}\,\delta(\boldsymbol{x})\,, \end{cases} \tag{5.5}$$

where $\boldsymbol{u}$ and p are the velocity and pressure fields, and ν the fluid viscosity. In two dimensions, the solution of Eq. (5.5) reads

$$\boldsymbol{u}(\boldsymbol{x}) = G(\boldsymbol{x})\,\boldsymbol{F}, \tag{5.6}$$

where G is the Green function:

$$G_{ij}(\boldsymbol{x}) = \frac{1}{4\pi\nu}\left[-\delta_{ij}\ln\left(\frac{|\boldsymbol{x}|}{L}\right) + \frac{x_i x_j}{|\boldsymbol{x}|^2}\right], \tag{5.7}$$

[2]The previous argument is not a proof: the problem of swimming at Low Reynolds is non-trivial and somehow counter-intuitive. See [77, 211] for details.

[3]We explored scenarios with pullers, but we could not find strong qualitative differences, except for the fact that the phenomenology is not as rich as in the pusher-pusher scenario. Hence, we decided to stick to pushers for the sake of consistency and just occasionally commenting on other possibilities.

[4]Notice that, in general, an incompressible 3D flow is no more incompressible when projected in 2D. One may directly check that the force dipole used in [173] is compressible in 2D: this would be a bad feature for us, since flow sinks or sources would generate uncontrolled attraction/repulsion between agents.

with L being an arbitrary length. The pressure field is $p(\boldsymbol{x}) = \boldsymbol{F} \cdot \boldsymbol{x}/(4\pi|\boldsymbol{x}|^3) + p_\infty$, where p_∞ is a constant.

We can consider two parallel but opposite point forces $\boldsymbol{F}^\pm = \pm F\boldsymbol{n}$ in two points located along the direcion of force: $\boldsymbol{x}^\pm = \pm\epsilon\boldsymbol{n}$, with some small displacement epsilon ϵ. The velocity field given by the superposition of the resulting Stokeslet is

$$\boldsymbol{u}(\boldsymbol{x}) = G(\boldsymbol{x} - \boldsymbol{x}^+)\,\boldsymbol{F}^+ + G(\boldsymbol{x} - \boldsymbol{x}^+)\,\boldsymbol{F}^- \simeq -2\,\epsilon\,F\,(\boldsymbol{n}\cdot\boldsymbol{\nabla})\,G(\boldsymbol{x})\,\boldsymbol{n} \tag{5.8}$$

where $F_i^+ = -F_i^- = Fn_i$ and we retained only the first order in ϵ, in order to obtain a far-field approximation under the assumption that $|\boldsymbol{x}| \gg \epsilon$. With some algebra, we get:

$$\boldsymbol{u}(\boldsymbol{x}) = \frac{D}{|\boldsymbol{x}|}\left[2\left(\frac{\boldsymbol{n}\cdot\boldsymbol{x}}{|\boldsymbol{x}|}\right)^2 - 1\right]\frac{\boldsymbol{x}}{|\boldsymbol{x}|} = \frac{D}{|\boldsymbol{x}|}\cos(2\phi - 2\theta)\begin{pmatrix}\cos\phi\\ \sin\phi\end{pmatrix} \tag{5.9}$$

where $D = F\epsilon/(2\pi\nu)$ is the dipole strength ($D > 0$ for pushers and $D < 0$ for pullers [147]). In the second equality we have introduced the angular representation so that $\boldsymbol{x} = |\boldsymbol{x}|\,(\cos\phi, \sin\phi)^T$ and $\boldsymbol{n} = \boldsymbol{n}(\theta) = (\cos\theta, \sin\theta)^T$.

Therefore, in order to get the velocity field $\boldsymbol{u}^{(\alpha)}$ generated by agent α in $\boldsymbol{x}_\alpha$ and affecting agent β in $\boldsymbol{x}_\beta$, take eq. (5.9) and set $\boldsymbol{x} = \boldsymbol{x}_\beta - \boldsymbol{x}_\alpha$ or, equivalently, $\theta = \theta_\alpha$ and $\phi = \arg(\boldsymbol{x}_\beta - \boldsymbol{x}_\alpha)$.

The corresponding vorticity field is given by

$$\omega = \frac{2D_\alpha}{R^2}\sin(2\phi - 2\theta). \tag{5.10}$$

A few important angles

Since we are interested in understanding how the perturbation generated by one agent, say β, affects the other, say α, we need to evaluate the field produced by the former in the frame of reference of the latter. As a definition, we require that an agent is always at rest and oriented along the horizontal axis in its own frame of reference. We can give a few definitions (see fig. 5.2):

$$\theta_\alpha = \arg(\boldsymbol{n}_\alpha) \tag{5.11}$$

$$R = \|\boldsymbol{x}_\alpha - \boldsymbol{x}_\beta\| \tag{5.12}$$

$$\Theta_\alpha = \arg(\boldsymbol{n}_\alpha) - \arg(\boldsymbol{n}_\beta) \tag{5.13}$$

$$\Phi_\alpha = \arg(\boldsymbol{x}_\alpha - \boldsymbol{x}_\beta) - \arg(\boldsymbol{n}_\beta) \tag{5.14}$$

where the function arg (commonly called atan2) is defined as $\boldsymbol{v} = |\boldsymbol{v}|\,(\cos(\arg(\boldsymbol{v})), \sin(\arg(\boldsymbol{v})))^T \;\; \forall \boldsymbol{v} \in \mathbb{R}^2$. In other words, θ_α and ϕ_α is the absolute orientation of α; Θ_α and Φ_α are the orientation and the angular position of α in the frame of reference of β, respectively. Note that the following relations hold:

$$\Theta_e = -\Theta_p \tag{5.15}$$

$$\Phi_e = \Phi_p + \pi \tag{5.16}$$

In pursuit-evasion literature, angle Φ_α is commonly referred to as *bearing angle* [24], while we call Θ_α the relative heading.

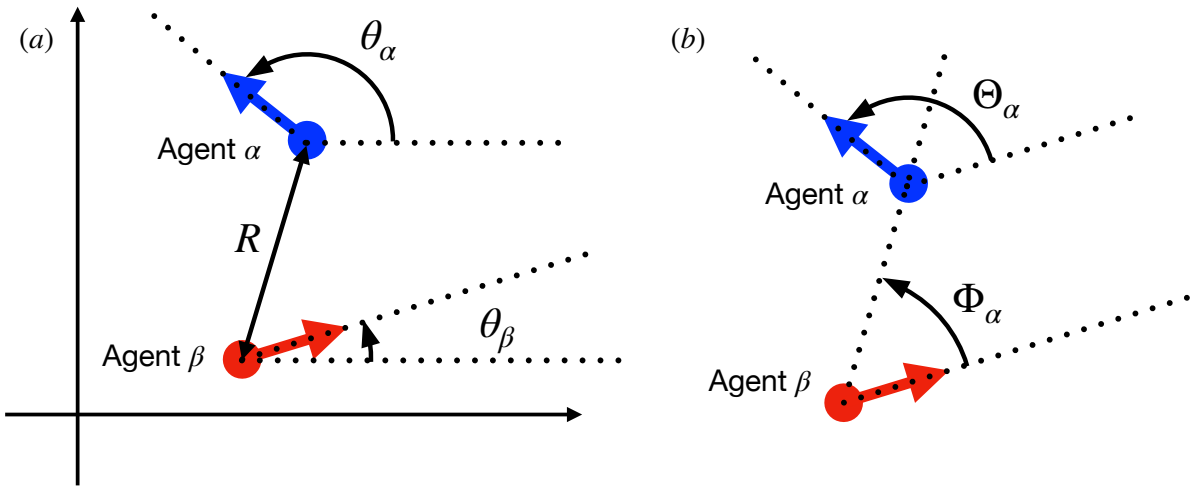

Figure 5.2: Panel (a) shows angles θ_a and θ_b describing the heading directions of both agents in a given frame of reference. Panel (b): angles Φ_α and Θ_α are, respectively, the angular position of agent α w.r.t. the heading direction of heading β (the bearing angle) and heading direction of agent α with respect to that of agent β.

5.1.2 Modelling the hydrodynamic information

Within our model, agents can base their strategy on hydrodynamic cues alone. Since they are dragged by the flow, it is unrealistic that they could measure the velocity field itself. For this reason, we assumed that they could perceive gradients of the field flow[5], as it is believed to happen for copepods [139]. In order to understand what kind of information can be extracted from the flow, it is convenient to decompose the gradient $\nabla \boldsymbol{u}$ in three physical components, the longitudinal and shear strains and the vorticity. Since agents have no notion of an external frame of reference, we assume that they perceive the gradients in their own frame of reference, i.e. projected along the swimming direction. The components of gradients of the field generated by β, in the frame of reference of α, read:

$$\omega^{(\beta)} = \partial_x u_y^{(\beta)} - \partial_y u_x^{(\beta)} = \frac{2D_\beta}{R^2}\sin(2\Phi_\beta - 2\Theta_\beta) \tag{5.17}$$

$$\mathcal{L}^{(\beta)} = \partial_x u_x^{(\beta)} = -\partial_y u_y^{(\beta)} = -\frac{D_\beta}{R^2}\sin(4\Phi_\beta - 2\Theta_\beta) \tag{5.18}$$

$$\mathcal{S}^{(\beta)} = \frac{1}{2}(\partial_x u_y^{(\beta)} + \partial_y u_x^{(\beta)}) = \frac{D_\beta}{R^2}\cos(4\Phi_\beta - 2\Theta_\beta)\,. \tag{5.19}$$

Note that the complete description of the system at any given time (assuming no memory), from the point of view[6] of α is encoded in the triplet $(R, \Phi_\beta, \Theta_\beta)$, which features relative distance, position an heading direction of the opponent. It is important to highlight that $(R, \Phi_\beta, \Theta_\beta)$ cannot be obtained from the accessible triplet $(\omega^{(\beta)}, \mathcal{L}^{(\beta)}, \mathcal{S}^{(\beta)})$. This is due to the presence of symmetries. In particular, the triplet $(\omega^{(\beta)}, \mathcal{L}^{(\beta)}, \mathcal{S}^{(\beta)})$ is invariant under the two following transformations:

[5]Note that this is not the only possible choice, for instance, agents could perceive time derivatives of the flow, pressure gradients, or frequencies of the flow oscillations.

[6]The whole system has (2(position)+1(velocity))*2(agents)=6 degrees of freedom. But, due to rotation (-1) and translation (-2) symmetries, only 3 true degrees of freedom are left. Thus, by adopting the point of view of either agents, we are eliminating spurious dimensions.

- Head-tail symmetry $\Theta_\beta \mapsto \Theta_\beta + \pi$. This is a property of the dipole flow and it is due to the nematic and non-polar nature of dipoles.
- Parity symmetry $\Theta_\beta \mapsto \Theta_\beta + \pi,\ \Phi_\beta \mapsto \Phi_\beta + \pi$. This is a general property[7] of the gradients of the velocity field.

Overall, the gradients are invariant under two independent transformations $\Theta_\beta \mapsto \Theta_\beta + \pi$ and $\Phi_\beta \mapsto \Phi_\beta + \pi$, separately. Therefore, any given value of the gradients is compatible with four configurations of the system. The head-tail symmetry implies that the orientation of a swimming dipole is ambiguous, the parity that any signal may be generated by two different positions of the opponent. Note that such uncertainties in the determination of the position of the source of the signal can be observed in nature: we can mention the 180^o ambiguity occurring in fish hearing [270]. Such uncertainties make it impossible to implement standard pursuit-evasion strategies such as the ones based on visual cues [24, 178]. Memory of past gradient detections and/or additional hydrodynamic cues could mitigate the ambiguities which, however, typically persist at larger distances [229, 236, 245].

Finally, let us note that there is a convenient 1:1 mapping between the gradients $(\omega^{(\beta)}, \mathcal{L}^{(\beta)}, \mathcal{S}^{(\beta)})$ and the triplet

$$(\hat{R} = D/R^2, \gamma = 2\Phi_\beta - \Theta_\beta, \omega_\beta) \tag{5.20}$$

which are the fundamental hydrodynamical features that we assume will be accessible to the agents. To see this, observe that $\hat{R} = \sqrt{\mathcal{S}^2 + \mathcal{L}^2} \propto 1/R^2$ and $2\Phi_\beta - \Theta_\beta = \arg((\mathcal{S}, \mathcal{L}))/2$.

5.1.3 Game structure and goal

Game structure

Now that we have outlined the dynamical aspects of our system of interacting swimming agents, we can describe how to implement the prey-predator behaviors. As it was anticipated, we formalize such adversarial interaction as a zero-sum game. This is a very important feature of our work, since it allows to study the stability of a strategy against response from the opponent, which is paramount in true biological setting, when competing strategies coevolve [107].

The simulation consists in repeated games, each called episode (using RL jargon). When an episode starts, agents are placed at an initial distance d_0 (which is approximately chosen in the order of the effective range of hydrodynamic effects) and have random orientations. They move according to equations (5.1)-(5.4) with velocity and vorticity fields given by (5.9) and (5.10) until either the predator catches the prey or time reaches a maximum allowed value T_{max}. The former ending scenario is defined as the first time that

$$\|\boldsymbol{x}_e - \boldsymbol{x}_p\| < R_c \tag{5.21}$$

with R_c being the capture radius. Below this scale, we assume the pursuer is able to capture the evader with certainty, e.g.by using its body parts to directly attack the evader or adopt some other move. The choice of not modelling what happens below this scale is consistent with our goal of

[7] It is a simple consequence of the tensorial nature (in the sense of field theory) of $\nabla \boldsymbol{u}$. Call $\mathcal{P}$ the parity transformation. We know that $\mathcal{P}\boldsymbol{u}(\mathcal{P}\boldsymbol{x}) = -\boldsymbol{u}(x)$ since $\boldsymbol{u}$ is a vector. Likewise $\mathcal{P}(\nabla \boldsymbol{u})(\mathcal{P}\boldsymbol{x}) = \mathcal{P}(\nabla \boldsymbol{u})(-\boldsymbol{x}) = \nabla \boldsymbol{u}(x)$, which is exactly our point.

not fine tuning too much our problem into a specific setting and allows us to use pure dipoles as hydrodynamic flows and to ignore short range details. The time cap T_{max} ensures that episodes have an ending in the first place, since agents could otherwise wander in plane indefinitely without ever meeting.

The goal of the predator is to catch the prey as soon as it can, which means it has to make duration of the episode as short as possible. Consistently, we can assign it a total reward $-T$ per episode, where T is the duration of the episode itself. Conversely, the prey ought to avoid capture, which means it tries to delay ending as long as it can. As a consequence, we assign it a total reward of T for each episode. Note that, during each episode, the total reward of the pursuer is exactly minus the reward of the evader: this defines a zero sum game [111] in the framework of game theory and implies that there is no room for cooperation between agents; whatever benefits one player, it damages the other.

Learning scheme

Agents learn their strategies through RL; in this framework, the reward is not assigned at the end of an episode, but accumulated in time. The game goes as follows. Every τ time units, at what we call decision time, each agent observes the environment (measures the local flow gradients) and chooses the action which, according to what it has learned, will mostly benefit it in maximizing the total reward. We assume that a swimmer has three available actions, which correspond to three possible rotational velocities $a_t^\alpha \in \{\Omega^\alpha = \pm\Omega_0^\alpha, \Omega^\alpha = 0\}$ for each agent α, while we assume the linear speeds are fixed. An action is drawn from a policy $\pi(a_t|o_t)$, which is the probability of picking a certain action a_t given a certain observation o_t. Then, the equations of motion (5.1)-(5.4) are integrated for time τ, after which the prey receives a reward $r = +1$ and the predator a reward $r_t = -1$, unless a capture event happens; in which case, they both receive a null reward and the episode ends. Provided the episode is still running, agents observe the environment again. In this way, agents can evaluate the consequences ($o_{t+\tau}$ and r_t) of action a_t after observing o_t and update the policy accordingly, through the learning algorithm, which is described in [27] (see also sec. 4.3.2). Finally, a new actions is picked and a new iteration begins.

In this setting, agents may learn simultaneously adversarial strategies; while this would be a legitimate choice, we found it is problematic in terms of interpretation. In a non adversarial setting, a RL algorithm ensures policy convergence to the optimal one as long as one decreases the learning rate appropriately episode after episode. In the learning process, the convergence is generally not straightforward, but exploration phases are generally present: the policy occasionally leaves local optima in search for new ones, in order to eventually converge to a global optimum (at least in principle). Thus, during exploration phases, the performance is bound to deteriorate even in non adversarial cases. In the adversarial setting, with simultaneous learning processes, since both policies are changing, the target policy of either agents is not fixed, so that exploration is even more important and convergence should not be given for granted. Note that an equilibrium may be reached if the algorithm is able to select it as a fixed point of the adversarial learning (we will comment on this in when commenting on the choice of the algorithm); it is not a priori obvious that, for an adversarial algorithm, this can happen either in theory or in practice. With these premises, even if an equilibrium exists a very long time may be needed to find it, with several intermediate events in which the two agents lose, win and explore, just like in [16]. Since we want to understand

what happens in this complicated dynamical process, we need to be able to tell when an agent is deploying a successful strategy, for the sake of physical interpretation: the main issue (a very practical one, that we did encounter) is that, if an agent, say the predator, is winning, it is nearly impossible to tell whether 1) the predator has found a strategy that dominates that of the prey 2) the prey is starting to explore, abandoning an otherwise successful strategy[8].

Because of all the aforementioned reasons, we have decided to structure the game in learning turns, which make it much easier to gain insight in the process. The turns consists of N_{ep} consecutive episodes; during one turn, only one agent updates its policy, while the other sticks on the policy found in the previous turn (or the initial one in the first turn). At the beginning of the game, all parameters are set to zero, so that the agents start with fully randomized policies in which each action is taken with prob $1/3$ at each decision time.

It is important to highlight that our turn-based setting, in which essentially agents play one move per turn and asynchronously, is not bound to have the same equilibria as the case in which they play simultaneously.

The following section provides details the RL procedure and more information about the implementation.

5.2 Implementation of the reinforcement learning algorithm

Policy parametrization and the choice of features

In the case of actual biological organisms, environmental cues are processed by the nervous system which encodes the behavioural responses to stimuli (the "policy"), for example by means of specific neurons that control escape reactions in fishes [76]. Such neural processing may be emulated by artificial neural networks [11]. Here, for the sake of explainability, we opted for an explicit parameterization of the policy in terms of a few features of the observations: we modelled the policy of both agents in the same way and, more specifically, we have chosen a function approximation scheme (see 4.3.1) to the reinforcement learning problem [233], which means that, instead of looking for the best performing policy among all possible ones, we restrict our search to a class of functions. We define the policy as a soft-max

$$\pi(a|o) = \frac{1}{Z} \exp\left(\sum_j \mathcal{F}_j(o)\, \xi_{aj} \right), \tag{5.22}$$

where $\mathcal{F}_j$ are *features* or *percepts* of the observable quantities o (local field gradient), and ξ are trainable parameters. Note that if the set of percepts $\{\mathcal{F}_j\}$ were a complete basis of functions on the space of gradients o, parametrization (5.22) would allow to approximate any (non mathematically pathological) policy $\pi(a|o)$: such basis would be composed of infinite many elements, which is obviously not possible in practice. We tested different choices of the features and the results presented correspond to the choice of $N_F = 13$ features, summarized in Table 5.1 (see (5.20)).

While the first six features are clearly related to the information that can be extracted from gradients and, specifically, triplet (5.20), the features $i = 7, 12$ are introduced to provide the agents

[8] Note that, too much exploration should be regarded as a pathological non convergence rather than a physiological phase of the learning process.

$\mathcal{F}_1(o_t) = \hat{R}(t)$
$\mathcal{F}_2(o_t) = \sin(\gamma(t))$
$\mathcal{F}_3(o_t) = \cos(\gamma(t))$
$\mathcal{F}_4(o_t) = \sin(2\,\gamma(t))$
$\mathcal{F}_5(o_t) = \cos(2\,\gamma(t))$
$\mathcal{F}_6(o_t) = \omega(t)$
$\mathcal{F}_i(o_t) = (1-\mu)\,\mathcal{F}_i(t-\tau) + \mu\,\mathcal{F}_{i-6}(t-\tau) \quad \text{for} \quad i = 7, 12$
$\mathcal{F}_{13}(o_t) = 1$

Table 5.1: Implemented features, derived from triplet (5.20). Note that features $7-12$ provide some memory of the values of the previous features; in the implementation we chose $\mu = 0.3$ to retain some memory about the last $2-4$ past observations approximately). $\hat{R}$, γ and ω are derived by the triplet (5.20).

with some weak memory, which may mitigate the symmetry-induced ambiguities to some degree; while we were expecting it might have made the difference (e.g. to remove at least partially the ambiguities on the agents positions), we concluded it played a minimal role, if any, in our game. Finally, the 13^{th} feature is unrelated to the gradients and it is chosen to allow the agents to adopt strategies which are independent of the percepts – a common choice in such kind of approaches.

Choice of the algorithm

In the simplest reinforcement learning setting, a single agent is interacting with static environment. In our case, however, there are two agents with opposite goals: from the point of view of a single agent, therefore, the environment is not only non-static, but adversarial. There are multiple consequences: first of all, many mathematical results about convergence of algorithms should not be given for granted; hence, stochastic policies may dominate deterministic ones and the algorithms may require specific adjustments. In particular, there should be two policies, one for each agent, and two learning sub-algorithms that should be run independently. We opted for the natural actor-critic algorithm (presented in sec. 4.3.2) because of its theoretical soundness in partially observable settings and its connection with evolutionary game theory [108, 111]. Specifically, while it is known that RL techniques do not always allow to find equilibria, the natural actor critic we have employed has been shown to reproduce the dynamics of the replicator equation [108] (the equation describing the evolution of two populations competing in the sense of evolutionary game theory) in the stateless case; one should be careful though, since combined effects of partial observability and instabilities due to implementation may interfere with the possibility of finding such equilibria.

A description of the natural actor critic algorithm, inspired by ref. [27], and further details may be found in Appendix 5.A. Following [27], we opted for a linear parametrization of the estimator for the observation value function:

$$\hat{v}(o) = \sum_j \mathcal{F}_j(o)\,\kappa_j \tag{5.23}$$

where κ is the set of value-function parameters and $\mathcal{F}_j$ are the same features as those used for the policy. Note that each agent has its own estimator $\hat{v}(o)$ and set of parameters κ.

5.3 Results

5.3.1 Parameter setting

The game described in the previous sections has been implemented with different parameter settings (see table 5.2), corresponding to different dynamical properties of the agents. In all simulations we have used capture radius $R_c = 0.05$ and interval between decision times $\tau = 0.1$. Moreovoer, for the sake of simplicity, we have used a fixed value of the dipole strength $D = 0.03$ in all settings, for both agents. Finally, we have used turns of $N_{ep} = 5000$ episodes and set a time cap $T_{max} = 500$.

The simplest scenario is given by setting A, in which the agents are identical, having the same linear speed $v_p = v_e = 0.1$ and angular velocity $\Omega_e = \Omega_p = 3$. Since the predator is the player with the hardest task (it has to find a small moving target in a plane), in setting B, we gave the predator a greater velocity $v_p = 0.3$, while other parameters were left unchanged. Note that, in this case, the agents still have same angular speed and, as a consequence, the curvature radius of the predator is larger than prey's. Since this may result in a penalty, we prepared a third scenario, C, with $v_p = 1.5$ and $\Omega_p = 4.5$, so that the curvature radii of both agents are the same. Since scenario A is very specific case featuring extreme fine tuning, we mainly focused on B and C. However, setting A is very instructive; to see why this is the case, though, we should first study the more general scenarios.

For each set of parameters, we can compute the relevant scales of the system: the typical distance R_h at which the hydrodynamic interactions start to overcome agents control can be found by imposing $D/(R_h\, v) = O(1)$, from which $R_h \approx D/v \approx 10^{-1}$. This is similar for all settings but it should be noted it is a very rough estimate. The key point is that, in all settings, hydrodynamics dominates at scales which is greater but comparable to the capture radius, which is consistent with the idea that agents can only produce small perturbations.

Setting A	$v_p = 0.1$	$v_e = 0.1$	$\Omega_p = 3$	$\Omega_e = 3$
Setting B	$v_p = 0.3$	$v_e = 0.1$	$\Omega_p = 3$	$\Omega_e = 3$
Setting B	$v_p = 0.15$	$v_e = 0.1$	$\Omega_p = 4.5$	$\Omega_e = 3$

Table 5.2: Parameter settings, where Ω_α are angular velocities and v_α are linear speeds, with $\alpha = e, p$ (evader, pursuer).

5.3.2 Analysis of emerging strategies

Different settings could generate different behaviours, in principle. Of course, such differences appear but, remarkably, it is possible to provide a unitary picture for them; therefore, in the following, they will be described together. Note that, for simplicity, of the many simulations we have run, only a few are shown in this thesis.

All game sequences start with the predator's learning turn, during which the prey uses a random policy, i.e. it picks any of the three actions with probability 1/3; the predator also starts with a random policy so that, at the very beginning of the game, the chance of a capture event is the probability of a random encounter in the plane. Different seeds for settings C (this is the setting we focused most attention on) and a seed for setting B are shown in figs. 5.4 and 5.3, respectively: in all cases, the normalized average duration of an episode T/T_{max} drops from the initial value

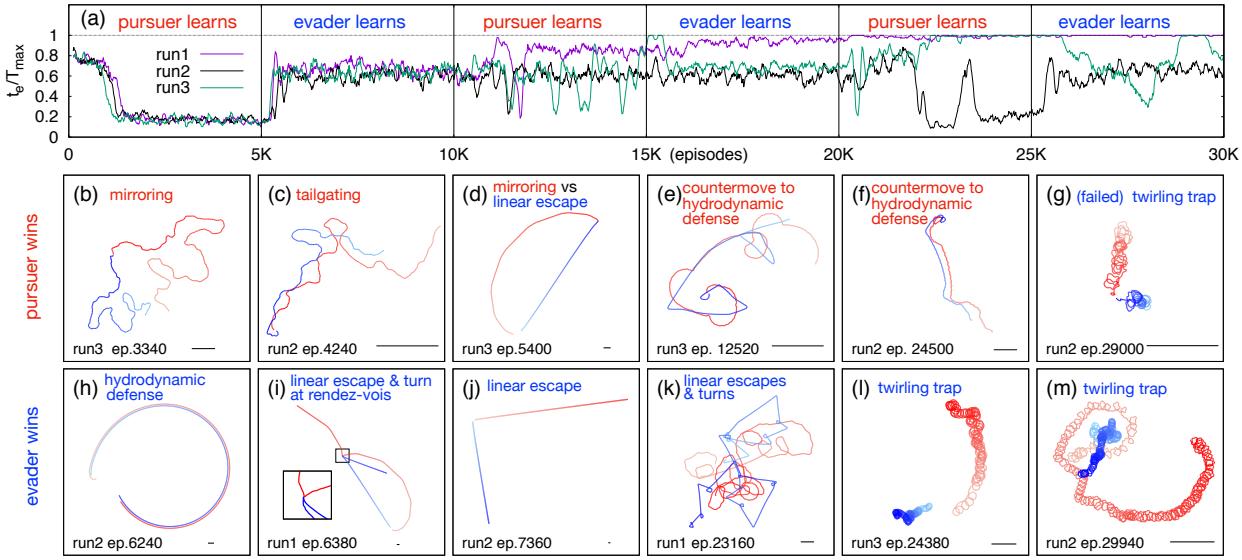

Figure 5.3: Setting C. History of first 6 training cycles and coevolving strategies. $(v_e, \Omega_e) = (0.1, 3)$ and $(v_p, \Omega_p) = (0.15, 4.5)$. (a) Running average (over 100 episodes) of normalized episode duration T/T_{max} for 3 realizations of the learning process. (b-g) Winning pursuing strategies: (b) mirroring, (c) tailgating, (d) mirroring vs linear escape with a rendez-vous, (e,f) tailgating with different countermoves to hydrodynamic defense, (g) failing twirling on mirroring. (h-m) Winning evasion strategies: (h) hydrodynamic defense, (i) linear escape with turn and hydrodynamic collision at rendez-vous, (j) linear escape against mirroring, (k) linear escapes and turns inducing pursuer switches between mirroring at distance, (l,m) twirling trap. Red/Blue denotes pursuer/evader trajectories, episode time runs from lighter to darker color; run and episode are labeled on each panel; the black segment on the bottom right displays the unit length.

and reaches a plateau, which is approximately the same for all seeds in the same setting. This indicates that the algorithm is both working and it has found a stable optimum of the learning problem in all cases. After 5000 episodes, it is the prey's turn: the average duration of an episode rapidly grows and reaches a new higher plateau: the evader has found a strategy to counter the pursuer's one. What happens from the third learning turn onward is less clear and different seeds show some variability. From a purely algorithmic point of view, it should be mentioned that agents occasionally lose in their own learning turn, implying that they may quit a good policy for less performing ones. Philosophically, we can call this explorations, but it is more correct to point out that such behaviours are likely indicators of poor stability of the algorithm in some phases[9]; we cannot also rule out imperfect tuning of the learning rates or of the duration of the learning phases, which might cause some algorithmic instabilities. With these warnings in mind, it still seems that, in a given setting, the (running) average duration of the episodes $\langle T/T_{max} \rangle$ in turns 3 onward oscillates around a certain values, which may be reminiscent of equilibrium values; we should not over-interpret it, though. The main reason is that not much can be said from the values of the $\langle T/T_{max} \rangle$ alone: similar pursuer-evader combinations of strategies may result in different $\langle T/T_{max} \rangle$ values and, most importantly, there are different combinations of policies associated to deceptively

[9]In interpreting this, we should not forget that, in spite of the possibility to identify strategies from their most distinctive qualitative features, policies are the result of the interactions of several parameters (the ξ, but also g and κ, see Appendix 5.A), many of which are likely redundant, in the sense that the same policy may effectively be obtained from several combinations of them; such underlying difference, combined with parameter fluctuations, may generate accidental micro-features of a policy which may not be easily visible to the human eye but could have a non-trivial impact on the performance or even be directly exploited by the enemy agent. For instance, in ref. [16], agents learned to exploit glitches in the implementation of the simulated physical laws to win against the competing team.

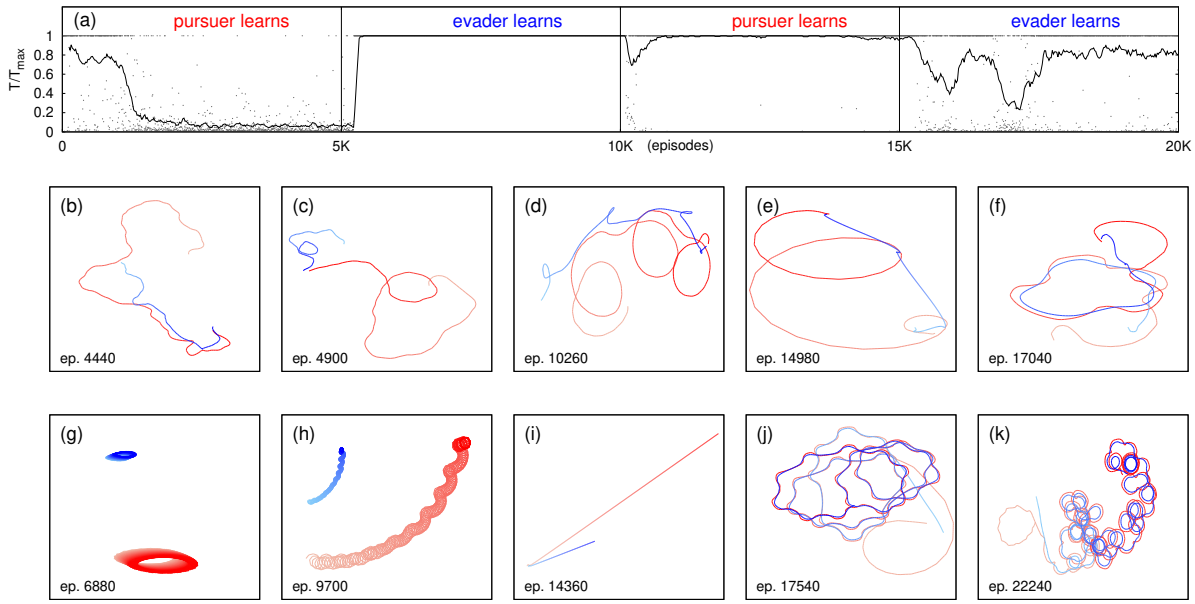

Figure 5.4: Setting B. History of first 6 training cycles and coevolving strategies, one seed. $(v_e, \Omega_e) = (0.1, 3)$ and $(v_p, \Omega_p) = (0.3, 3)$. We observed reproducibility of the first two learning cycles and variability in the other ones (not shown). Panel (a) shows the normalized episode duration, T/T_{max}, as learning progresses. The dots represent individual realizations, while the line is the running average over 100 episodes. Panels (b-f) describes cases in which the seeker wins. (b) and (c) correspond to tailgating and mirroring respectively; (d) shows some complicated behaviour (most likely a combination of mirroring and some form of defence); (e) shows a capture arch (mirroring) countered by sudden defence turns (or, possibly, hydrodynamics collisions); (f) a successful untrapping from an hydrodynamic defense of the prey (likely, with a form of tailgating). Panels (g-k) correspond to the evader winning: in (g) it copes against mirroring by twirling as this induces a twirling at distance of the seeker; (h) is similar but with a drift. In (i) the prey goes straight, generating a extremely long mirroring capture arch, essentially a straight line. (j) and (k) show instances of hydrodynamic defense along with twirling.

similar $\langle T/T_{max} \rangle$.

A way more compelling picture can be drawn by looking at which strategies are practically implemented by the agents. The most remarkable point is that what happens in the first two turns is very stable from a qualitative and quantitative point a view. In all settings (A, B and C), the predator always learns the same fundamental strategy in turn one, to which the prey responds in specific ways (the evader response is somehow more dependent on the setting). The nice feature of these early turns is that we could gain a deep – qualitative and quantitative – understanding of the observed behaviours, which are both visually appealing and physically insightful. Indeed, by looking at figs. 5.3 and 5.4 we may notice how the predator manages to win by producing trajectories which are geometrically similar to those of the prey; on the other hand, in its own learning phase, the evader quickly learns to turn around, run in a straight line and use hydrodynamic repulsion as a defence mechanism. Remarkably, strategies deployed from turn 3 onward are variations over such fundamental strategies for the most part.

In the following, we will explain these strategies and their significance.

Predator strategies. Mirroring and tailgating: two role of partial information.

During the first turn, while the prey moves randomly, the predator's strategy may, at first, appear puzzling: depending on the episode, the pursuer either deploy a behaviour we call "mirroring" or

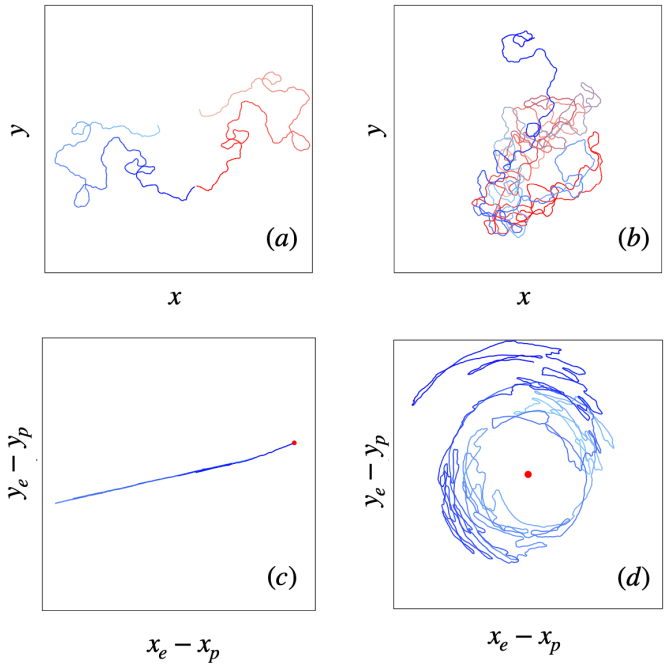

Figure 5.5: Setting A. Red=pursuer, blue=evader. A case of mirroring and tailgating in panel (a) and (b), respectively. Panels (c) and (d) are the same same trejectories respectively, but in a frame of reference where the predator is at rest. It is clear that mirroring, in reducing the joint dynamics to a line (panel (c)), is a case of dimensionality reduction. Notice how, in panel (b) the predator fails to approach the prey; if (5.27) was enforced exactly, the evader would draw an exact circle around the pursuer while in this case some diffusion is visible due to approximate implementation of this policy. Many factors may contribute to this, such as hydrodynamic effects, finite decision time, finite manoeuvrability etc. See fig. 5.9 for more details about this last comment.

what we call "tailgating". The two behaviours are easy to identify and to distinguish visually. When adopting the former (fig. 5.3(b) in setting C and fig. 5.4(c) in setting B), the predator's trajectory appears as a distorted mirror image of prey's one. The greater the difference between the two agents' speed is, the larger the distortion becomes: if $v_p = v_e$ (setting A, fig. 5.5(a)), no deformation is left and the symmetry between trajectories is remarkably precise with respect to a well defined axis of symmetry. When the pursuer adopt tailgating (fig. 5.3(c) in setting C 5.4(b) in setting B), on the other hands, it moves behind the back of its target and then runs forward to capture it. This latter scenario is noteworthy because, by construction, the pursuer does not have access to the prey's position. In scenario A ($v_e = v_p$, fig. 5.5(b)), however, tailgating degenerates in maximally inefficient behaviours which practically *never* ends in a capture event. In all cases, one of these two strategies is selected in a seemingly random way with uneven frequencies (depending on the setting, the tailgating vs mirroring frequency ratio may vary from 0.5 up to even 0.8). Note that these two behaviours are not produced by successive phases of the learning process; they are generated by the very same policy parameters (which we checked by freezing the pursuer's parameters at the ending of the first learning turn). Besides wondering how two seemingly incompatible behaviours may coexist and why they have been selected, one may ask how the predators enforce them: contrary to what the reader may guess, no memory is used is uttering such strategies (they appear even when removing all delayed percepts from the predator's ones). In order to solve this puzzle, we need to give a quantitative descriptions of such behaviours. Note that the policies that we are going to

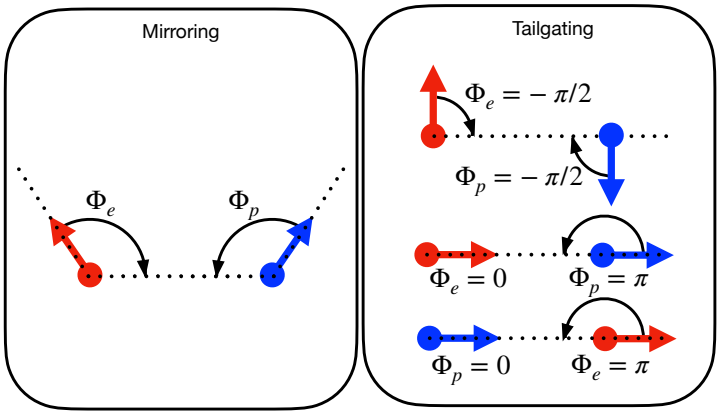

Figure 5.6: The two panels illustrate relations for mirroring (5.24) and tailgating (5.27). Note that, in the latter case, the pursuer may paradoxically run away from the prey.

describe work as long as hydrodynamic interactions do not dominate; at short range, the pursuer may change behaviour. Close interactions are somehow less interesting and will be described later while, in the following, we will focus on how the predator may locate the prey.

By looking at the trajectories, one can define mirroring as pursuer's behaviour such that, at any given time, the pursuer (defined by its position and heading direction) is the reflected image of the evader w.r.t. to a given axis of symmetry (which, in general, is time dependent). It is not hard to formally translate this in the following equation (see fig 5.6)

$$\Phi_e = -\Phi_p. \tag{5.24}$$

With some algebra, we can rewrite eq. (5.24) in the frame of reference of the pursuer:

$$2\Phi_e - \Theta_e = \pi. \tag{5.25}$$

Hence, in order to achieve mirroring, the pursuer has to make sure that (5.25) is enforced at all times. Note that (5.25) fully specifies the ideal control $\Omega_p(t)$ at all times. Below, more details will be given on how this is achieved in practice. We just observe that this strategy is often effective in capturing the prey, while leaving a more detailed explanation later.

The next thing we should ask is what the equation for tailgating is. Guessing such equation by simply looking at the trajectories without any clues is not as easy as for the mirroring. However, we can readily get it if we understand the connection between tailgating and mirroring. Let us look at relation (5.25); in order for the pursuer to enforce it, it does not need to measure Φ_e and Θ_e separately (we know that it cannot!) but it must somehow extract information about $2\Phi_e - \Theta_e$ from the gradients. By looking at the equations for the strains (5.19) and (5.18), we realize this is where such angle come from and, indeed, $2\Phi_e - \Theta_e$ is part of the triplet (5.20). We have already mentioned that gradients-derived clues suffers from symmetries $\Phi_e \mapsto \Phi_e + \pi$ and $\Theta_e \mapsto \Theta_e + \pi$. While the mirroring equation (5.25) is invariant w.r.t. the former, the latter maps (5.25) into

$$2\Phi_e - \Theta_e = 0. \tag{5.26}$$

or (see eq. (5.6))

$$\Phi_e + \Phi_p = \pi. \tag{5.27}$$

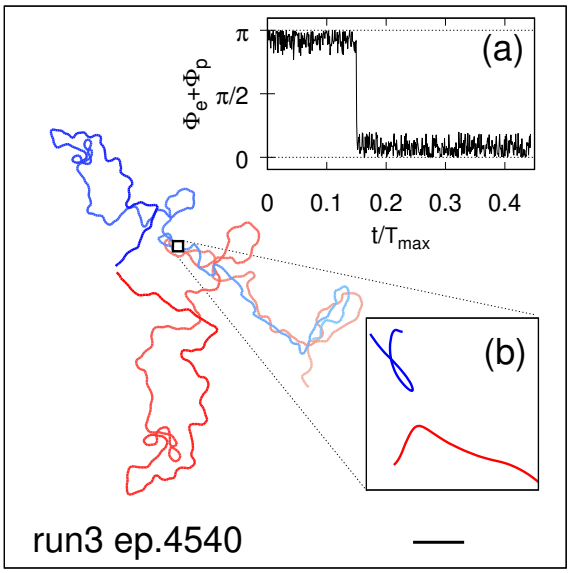

Figure 5.7: Setting B. Switching between tailgating to mirroring strategies. Inset (a) Sum of bearing angles $\Phi_e + \Phi_p$ vs normalized time, notice the switching from $\approx \pi$ (tailgating) to ≈ 0 (mirroring) at $t/T_{max} \approx 0.15$ corresponding to the close encounter and the evader turning shown in inset (b).

Therefore, unless some smart memory effect is exploited, the agent cannot distinguish between (5.26) and (5.25): any strategy which preserves the former, will preserve the latter as well. Depending on the initial condition, the pursuer will randomly select either ones. It is now easy to guess that (5.26) describes tailgating, as can be readily checked numerically by plotting the angles. We can even write a joint equation

$$2\Phi_e - \Theta_e = \gamma_\pm \tag{5.28}$$

with $\gamma_+ = \pi$ for mirroring and $\gamma_- = 0$ for tailgating. Note that, while the predator will try stick to either strategies, in some cases, it can switch from one to the other. This is shown in fig. 5.7: the predator, after a hydrodynamic collision in which it fails to catch the prey, loses control over its current strategy and seeks again to enforce one in eqs. (5.28); in the case shown, the new strategy happens to be different from the original one: this a very neat way both to illustrate and to validate our argument. With some straightforward algebra [10], we can write down the joint prey-predator

[10] The equations of motion are $\dot{x}_\alpha = v_\alpha \boldsymbol{n}(\theta_\alpha)$ and $\dot{\theta}_\alpha = \Omega_\alpha$ $(\alpha = p, e)$ with the constraint (5.28), which we may write as

$$2\,\arg(x_e - x_p) - \theta_p - \theta_e = \gamma_\pm$$

so that

$$2\,\frac{d}{dt}\arg(x_e - x_p) - \Omega_p - \Omega_e = 0 \qquad (\star)$$

The constraint (5.28) cuts the degrees of freedom of the joint dynamics from 3 to 2, so that the system may be described in the (R, Φ_e) cylinder (or the (Θ_e, Φ_e) torus).

A simple differentiation of $R = \|\boldsymbol{x}_p - \boldsymbol{x}_e\|$ reveals that (use eq. (5.28))

$$\dot{R} = v_e \cos(\theta_e - \arg(x_e - x_p)) - v_p \cos(\theta_p - \arg(x_e - x_p)) = v_e \cos(\Phi_e - \gamma_\pm) - v_p \cos(\Phi_e)$$

from which we get the first equation in (5.29).

Now compute

$$\frac{d}{dt}\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) = \frac{[v_p\,\boldsymbol{n}(\theta_p) - v_e\,\boldsymbol{n}(\theta_e)] \times (\boldsymbol{x}_e - \boldsymbol{x}_p)}{R^2} = \frac{v_p\,\sin(\Phi_e) + v_e\,\sin(\Phi_e - \gamma_\pm)}{R} \qquad (\star\star).$$

By plugging eqs.$(\star)$ and $(\star\star)$ into the definition $\dot{\Phi}_e = \frac{d}{dt}\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) - \Omega_p$, we obtain the second equation from (5.29).

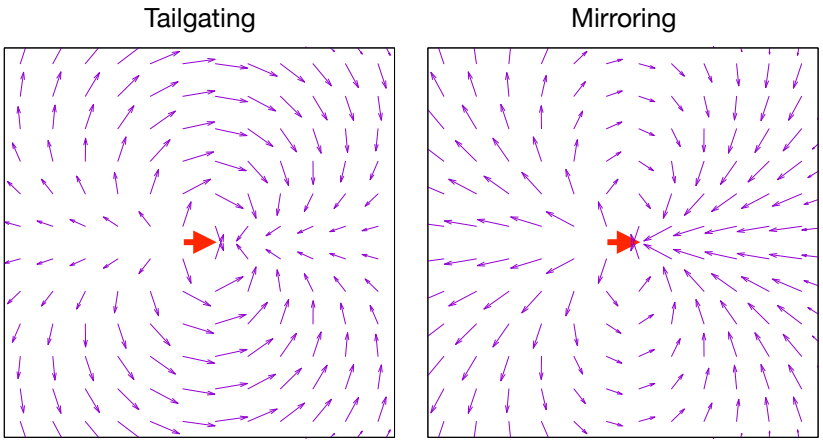

Figure 5.8: Mirroring and tailgating vector fields affecting the prey, from the point of view of a predator fixed in the origin (red arrow) and oriented to the right. They are obtained from (5.29) with $\Omega_e = 0$. Note that Ω_e induces a rotational behaviour in the plane.

equations of motion in the frame of reference of the predator: they read

$$\begin{cases} \dot{R} &= -(v_p \pm v_e)\cos(\Phi_e) \\ \dot{\Phi}_e &= \Omega_e - \dfrac{1}{R}(v_p \mp v_e)\sin(\Phi_e)\,, \end{cases} \tag{5.29}$$

for mirroring (upper sign) and tailgating respectively (lower sign). Note that eq. (5.29) is true for any prey strategy $\Omega_e(t)$, but we clearly have to neglect hydrodynamics (far field equations) and assume the strategy is implemented exactly. Still, eqs. (5.29) are remarkably accurate in when agents are sufficiently far apart from each other and still very useful at medium-short range. We can plot the vector field associated with both cases in fig. 5.8 in the pursuer's reference frame. We can immediately see that, if $\Omega_e = 0$ (prey moving in a straight line), any trajectory leads eventually to the origin, a fact that will become important later. In the first turn, though, Ω_e is random and plays the role of rotational noise (note that it does not depend on R!). Let us assume that $v_p > v_e$. Then, in far field conditions, as long as $R \gg (v_p \pm v_e)/\Omega_0$, noise dominates and the prey's trajectory is given by wide arcs with random orientation and length; Φ_e is essentially random and, therefore, in this limiting case, $\dot{R}$ is zero on average. The dynamics is still biased towards the origin. Indeed, when $R \leq (v_p \pm v_e)/\Omega_0$, noise becomes less relevant and the deterministic part dominates. Then, $\Phi_e = 0$ becomes attractive since $\dot{\Phi}_e \propto -\sin(\Phi_e)$. Note that $\Phi_e = 0$ implies that the heading direction of the pursuer points towards the evader and, therefore, we have $\dot{R} < 0$ (consistently with the first equation in (5.29)). There is a huge qualitative difference between mirroring and tailgating, though. In the mirroring case $\Phi_e = 0 \implies \Theta_e = \pi$ and, therefore, the predator would attack the prey from the front; however, note that $\Phi_e = 0$ is attracting in a weaker way ($v_p - v_e < v_e + v_p$) and, therefore, little noise suffices for attacks to come from sides instead of front. On the contrary, in the tailgating case, $\Phi_e = 0$ is way more attracting and $\Phi_e = 0 \implies \Theta_e = 0$ so that attacks mostly come from behind, as it is observed.

It is natural to wonder how the predator can enforce either strategies. In fig. 5.9 we show the average action, defined as $\langle a \rangle$ with $a = 0, \pm 1$ (± 1 is the sign of the rotational velocity) as a function of (Φ_e, Θ_e). There are clear bands with sharp transitions, corresponding to "go straight","go left" and

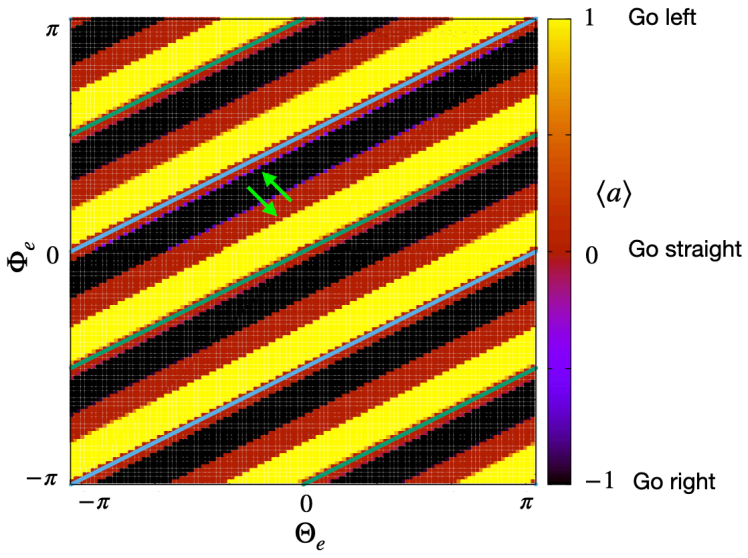

Figure 5.9: Setting A, where delayed features have been removed (agents have no memory). The average action as a function of Φ_e, Θ_e. Green and blue lines correspond to tailgating and mirroring, respectively. At large distance, the predator can move in the directions shown by the light green arrows and thus it can cross stripes. Note that the yellow and black stripes stabilize only red ("go straight") bands which correspond to tailgating and mirroring. Moreover, it is noteworthy that the policy is almost deterministic, since transitions between bands are sharp. Note that the "go straight" stripes corresponding to mirroring (5.25) and tailgaing (5.26) have finite width so that these relations are enforced with a certain tolerance, which explains the features of trajectories in fig. 5.5. In order to draw the figure, we placed agents at distance 1 and rotated the two angles in order to explore the torus. Different distances do not yield significantly different results.

"go right" zones. Notice that half of the "go straight" lines correspond the mirroring and tailgating sets. The surrounding stripes are organized in such a way that any deviation from the desired regions is compensated by an appropriate counter-maneuver. In this way, the mirroring and tailgating sets are dynamically stable behaviours. Note that, with a still prey and in far field conditions, it is easy to see[11] that $\dot{\Phi}_e \approx -\Omega_p = -\dot{\Theta}_e$ so that the predator can move along diagonal lines in the (Φ_e, Θ_e) space. From this picture, we can also guess that, in idealized conditions, mirroring and tailgating would be selected with $50\% - 50\%$ probabilities since there are no preferred initial conditions in the (Φ_e, Θ_e) torus. This is not true in practice: hydrodynamic effects skew such probability towards one of either cases, depending on the setting, unless the agents are put at very large distances when the game starts.

The previous discussion has been mainly centered around settings B and C. It is now time to focus on case A, which is different and insightful; for the sake of clarity, we have removed memory from the percepts in this setting. By looking at equations (5.29), we can readily see that, if $v_p = v_e$, then $\dot{R} = 0$ in the tailgating case. Therefore, if this strategy is implemented exactly, the predator would never reach the prey which would remain on a circle of constant radius drawn around the pursuer (in the pursuer frame of reference). Note that, in practice (fig. 5.5(d)), the radius is not exactly constant since the tailgating strategy is not implemented perfectly[12] and nothing compensates radial diffusion or drifts. In setting A, tailgating becomes a trap, but, in order to occasionally achieve mirroring, the predator still seeks the tailgating/mirroring strategy. In this sense, we have provided

[11] $\frac{d}{dt}\Phi_e = \frac{d}{dt}\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) - \Omega_p$ with $\frac{d}{dt}\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) = \boldsymbol{n}(\theta_p) \times (\boldsymbol{x}_e - \boldsymbol{x}_p)/R^2 \sim 1/R$.
[12] This would be impossible due to finite decision time τ and limited angular/linear manoeuvrability.

neat example of the effects of partial observability. It would not be unreasonable to assume that an animal or a robot may do something similar: unable to locate an object, it could accept the risk of missing the target in about half of its attempt, as long as, during the other half, its strategy proves effective. Mirroring becomes also very interesting in setting A. We can write eq.(5.25) as

$$2\Phi_e - \theta_e + \theta_p = 0, \tag{5.30}$$

and observe that, if we set $v_p = v_e$ in (5.29), we get $\dot{\Phi}_e = \Omega_e$. Then, obtain

$$\dot{\theta}_p = -\dot{\theta}_e = -\Omega_e. \tag{5.31}$$

Now, remember that $\Phi_e = \arg(\boldsymbol{x}_e - \boldsymbol{x}_p) - \theta_p$: by differentiating (5.30) and inserting (5.31), we get

$$\frac{d}{dt}\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) = 0, \tag{5.32}$$

Therefore, if we look at the trajectory in a frame of reference in which the predator always sits in the origin (but axes have fixed direction), the prey moves along the line defined by the angle $\arg(\boldsymbol{x}_e - \boldsymbol{x}_P)$ (fig. 5.5(c)). Now consider (5.29) again: if Ω_e is a random variable, so is $\cos(\Phi_e)$ in the equation for $\dot{R}$. Hence, the prey is performing a random walk in one dimension. We can provide a rationale for mirroring in this setting. If the predator cannot locate the prey, by default, the chance of a capture event is a the chance of a random encounter in the plane. On the contrary, if mirroring is employed, the pursuer's search is still random but is enhanced by a *reduction of dimensionality* [3], since a capture event can be reframed as a random encounter in 1D, which is a much likelier event. This description is strictly true only in setting A but it is not hard to argue that it still qualitatively applies to the general case: the predator strategy is to reduce the effective volume in which the prey is diffusing, in order to increase the average first passage time for $R < R_c$.

Prey strategies. Information and hydrodynamic defences

After outlining the predator's fundamental strategies in the first turn, we can focus on the second turn and prey's responses. Such responses can be classified by looking at the distances and the role of the hydrodynamics.

The first defence is mainly found in setting B (fig. 5.4(g,h,j,k)) and A (not shown), but also C (fig.5.3 (g,l,m)). It is an information-based defence that works at long range. Basically, the prey starts turning around, designing circles in the planes. By deploying mirroring or tailgating, the predator is trapped by its own policy and starts turning around as well. This strategy in nearly impenetrable in setting A, but it seems that the predator can otherwise occasionally manage to drift towards the prey (there could be several ways that could explain this: hydrodynamic interactions, imperfections in the prey's implementation or even just slow kinematic effects we did not explore) and partially overcome this.

A second, most notable strategy (setting B fig. 5.4(f,j,k); setting C fig. 5.3(h,e,f)) is employed at short range and involves direct hydrodynamic interactions, usually after a attempt to fly away in a straight line (setting B fig. 5.4(i,e); setting C fig. 5.3(d,j)), since essentially the evader may adopt this defence by sticking to the "go straight" action. If the pursuer manages to reach the prey (with tailgating), it tries to attack it from behind as we have described. Once the predator is behind it,

the prey tries to keep its heading direction aligned with the prey's one. As a result, two combined effects keep the predator at distance: the repulsive wake of the evader pushes back the predator and the repulsive flow coming from the head of the predator pushes the prey forth (see fig. 5.10). This fully repulsive interaction is too strong for the predator to overcome it: as a result, the pursuer follows in the evader's trail without ever catching it. In later turns, the predator occasionally learns a way to weaken such defence (setting B fig: 5.4(f); setting C fig: 5.3(f)): for instance, by shrinking the "go straight" band it starts wiggling while tailgating and, therefore, it makes it harder for the evader to maintain correct alignment. This strategy is less effective against mirroring; since the prey generally moves in a straight line, clearly $\dot{\Omega}_e = 0$ (setting B fig. 5.4(i,e); setting C fig. 5.3(d,j)). However, as we have anticipated while describing (5.29), this leads to the prey eventually meeting the predator, which is clear in (setting B fig. 5.4(e); setting C fig. 5.3(d)), where the predator's designs what we call "capture arch" in the plane. As we have already discussed, at the encounter point, the prey does not attack from behind and, therefore, a capture event is likely to happen. Note that, depending on the initial condition, such arch may be extremely large, to the point that the time needed to reach the target may be way longer than T_{max}. For this reason, it may make sense, from the point of view of the evader, to counter tailgating and hope that, in case of mirroring, the capture arch will fail. In some turns, the prey occasionally learns to counter successful arches by performing a sudden turn (setting B fig. 5.4(e); setting C fig. 5.3(i,k)) when the predator is getting too close: as a result, the pursuer must plan a new arch to meet the prey again. If the evader can dodge all successive attacks before time expires, it has a chance never to get caught.

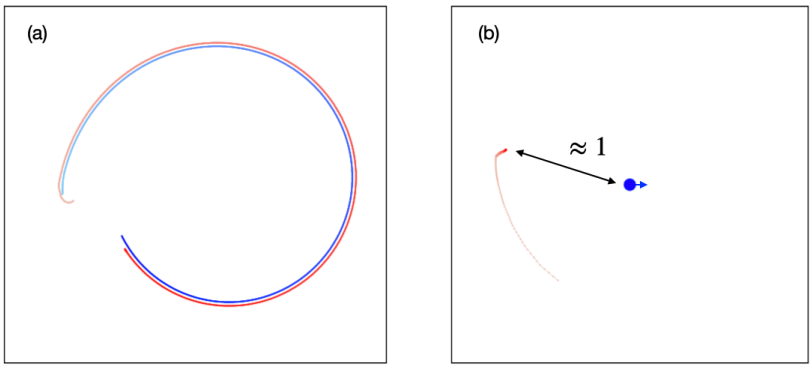

Figure 5.10: Hydrodynamic defence, setting C. The predator approaches the prey from behind (tailgating) but is repelled by the hydrodynamic interaction. In panel (a), the trajectories are shown (a magnification of panel (h) from fig 5.3.). In panel (b), the same trajectories are shown in the reference frame of the prey: the predator remains distant, trapped in the repulsive wake of its opponent.

Note that, in setting B, the two main defence strategies are likely combines in an impenetrable defence see (fig. 5.4(j,k)) which exploits circling, hydrodynamics defence and the larger curvature radius of the predator's trajectories which makes it nearly impossible for the predator to close in on the prey.

5.3.3 Comparison with known pursuit strategies and quality assessment

There are a number of well known pursuit strategies from both biological and robotic literature. They generally exploit visual information, so that they do not constitute a reliable baseline. Just like

in the mirroring and tailgating case, such strategies are described by angular relations. Figure 5.11 shows such comparisons.

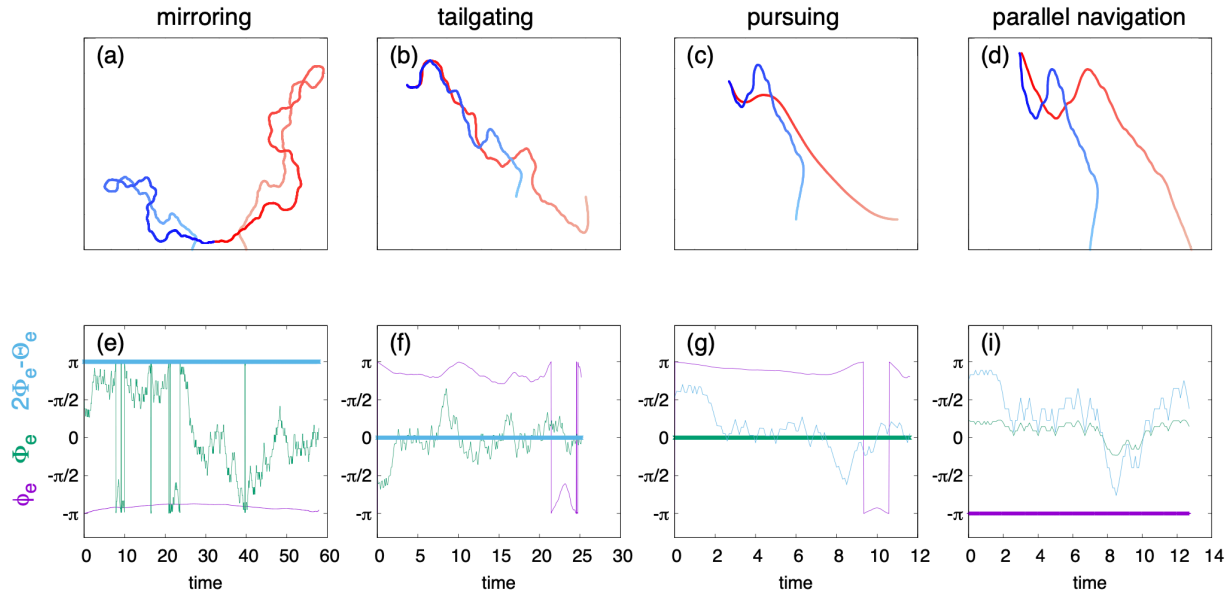

Figure 5.11: Comparison between mirroring and tailgating strategies and standard visual-based pursuit strategies: (a) mirroring, (b) tailgating, (c) pure-pursuit, (d) parallel navigation. (a-b) have been discovered by Reinforcement Learning, (c-d) are known standard strategies based on the knowledge of the line of sight between pursuer and evader agents. Red=predator; blue=prey. Panels (e-h) show the time evolution of the line of sight $\phi_e = \arg(\boldsymbol{x}_e - \boldsymbol{x}_p)$ w.r.t. to a fixed frame of reference (purple curves), of the bearing angle Φ_e (green curves) and the angle $2\Phi_e - \Theta_e$ (light blue curves). Each strategy keeps fixed some angle during the evolution: (e) $2\Phi_e - \Theta_e = \pi \mod 2\pi$ for mirroring, (f) $2\Phi_e - \Theta_e = 0 \mod 2\pi$ for tailgating, (g) $\Phi_e = 0$ for pure-pursuit, (h) $\phi_e = const$ for parallel navigation. The evader trajectory (blue) is obtained with a random policy $\pi_e(a|o) = 1/3$, while the seeker one (red) is obtained by numerically integrating the kinematic equations [24], neglecting the hydrodynamics, with the appropriate constraint on the angle. Data refer to the case $v_p = 0.15$, $v_e = 0.1$ and $\Omega_e = 3$. The prey (initialized in the origin with random orientation) chooses the angular velocity as in main text, while the predator (initialized in $(1, 0)$) heading direction is geometrically imposed as appropriate for the strategy. In particular, parallel navigation is obtained by imposing $\theta_p = \phi_e + \arcsin(v_e/v_p \sin(\theta_e - \phi_p))$ as described in [24]. Notice that when the seeker performs mirroring the angle ϕ_e w.r.t. a fixed frame of reference is almost constant as in parallel navigation (it becomes exactly constant in the case $v_p = v_e$), but unlike parallel navigation there are situations in which the seeker moves away from the hider, it thus corresponds to an imperfect parallel navigation. Notice also that, in tailgating, $\Phi_e \approx 0$, holds at late time (just before a capture event happens).

We may mention, as the simplest strategy, pure pursuit (fig. 5.11(c,h)) defined as $\Phi_e = 0$, as (e.g. bats or some fishes appear to adopt it [63, 145]) which is what tailgating reduces to once the predator has moved behind its prey. Note that they are not otherwise equivalent: if the pursuer is aligned with the evader but is located in front of it, it will paradoxically flee forward and run through a huge arch until it finds itself behind its target.

"Parallel navigation" (fig. 5.11(e,j)) is a a strategy which consists in keeping the line-of-sight direction $\arg(\boldsymbol{x}_e - \boldsymbol{x}_p)$ constant with respect to an inertial frame of reference (it has been conjectured that dragonflies follow this class of strategies for predation purposes [184]) and, in this sense, it is loosely related to mirroring. The picture is most clear in setting A: in this case, both strategies are characterized by the condition $\arg(\boldsymbol{x}_e - \boldsymbol{x}_p) = const$. More precisely, parallel navigation is given by $\Phi_p = -\sin^{-1}(\sin(\Phi_e)v_p/v_e)$ which becomes $\Phi_p = -\sin^{-1}(\sin(\Phi_e))$ when $v_p = v_e$: in parallel

navigation, only the solution $\Phi_p \in [-\pi/2, \pi/2]$ is taken (the pursuer always approaches its target), while in mirroring $\Phi_e = -\Phi_p$ is used. Therefore, mirroring may be viewed as an imperfect parallel navigation. For completeness sake we can also mentioned deviated pursuit [167], which is given by $\Phi_e = const \in (-\pi/2, \pi/2)$.

Overall, both tailgating and mirroring are clearly underwhelming w.r.t. visual strategies, as they should. However, at short range, the performance of the former is not that different from that of its visual counterparts, which is remarkable. Moreover, a fair comparison should account for partial information. Any no-memory visual strategy which one could design would still suffer from 180° position ambiguity: for instance, assuming the pursuer should pick either position randomly at the beginning of the episode and chase it[13], it would fail half of the time. As a result, the best visual strategy (compatible with this assumptions) that one could design would still achieve $T \geq T_{max}/2$ which is larger than the mirroring/tailgating pair. We could even go further and assume that the prey may pick a random target between the two possible prey positions every N_p (persitency) time steps and chase it with pure pursuit: in this case, $\langle T/T_{max} \rangle$ has a minimum around $N_p \approx 400$ (setting C) and its value is approximately 0.4 (see fig. 5.12 for more details), which is still way larger than the tailgating/mirroring pair. Therefore, RL strategies perform even better than pure pursuit adapted to partial observability (at least in this way).

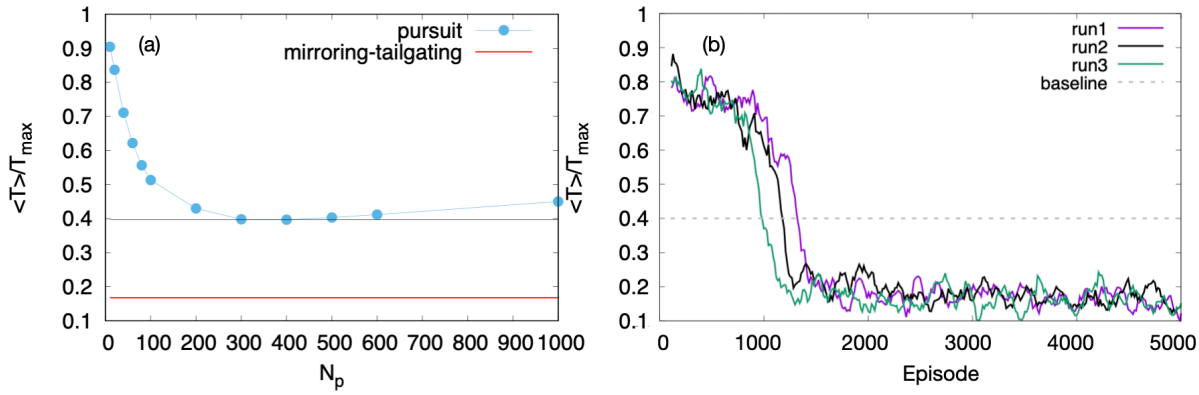

Figure 5.12: An heuristic baseline for pursuer's policies with partial observability, setting C. We assume the predator choose either of the two possible location of the prey every N_p (persistency time) decision times and uses pure pursuit to chase such target. In panel (a), we scanned different values of N_p, identifying a minimum of $\langle T/T_{max} \rangle \approx 0.2$ at $N_p \approx 400$; such minimum is way above the typical duration of an episode where tailgating-mirroring is used against a random prey. In panel (b), we show $\langle T/T_{max} \rangle$ for three seeds in the first turn, along with the baseline.

Note that it is easy to argue that, through some smart way of manipulating memory, it would be possible to design superior performances (for example with deep reinforcement learning or a decision tree). For instance, consider the following algorithm: aim at one of two possible positions with pure pursuit; if the hydrodynamic signal does not increases over times, switch to the other possible location with pure pursuit. In spite of this, note that, at least in the first turn of settings B and C, partial observability is mostly solved with the mirroring/tailgating pair. No matter what the initial condition is, either sub-strategies will work to some degree; this is remarkable since it allows for easy and robust implementation and does not require complicated decision structures.

[13] the phantom image would keep run far away

5.4 Conclusions and perspectives

We have shown that, through reinforcement learning, it is possible to simulate prey-predator interactions at low Reynolds number. Our work is set in a highly idealized scenario as is intended as a proof of concept for machine learning applications in this field. As such, since realistic details are missing, it is unlikely to have yielded results of direct biological or robotic relevance. However, real-world strategies are likely dependent on the details (size and shape of the animals, their propulsion system, sensory capability, memory and intelligence and so on) of system, to some degree. On the contrary, we have been able to identify some relevant behavioural patterns in partial observable hydrodynamic environment which we conjecture may be way more general than our specific setting. Let us consider the predator first; we have shown that, in setting A and B, the predator can overcome information uncertainty in a very efficient way. Instead of trying to identify the position of the target, it adopts a tactic which produces an ambiguous behaviour (mirroring or tailgating; the pursuer cannot distinguish one from the other): the predator does not know which behaviour is picking but, whatever it is, it efficiently approaches the target in one way or the other. In setting A, we have seen a more difficult scenario: it is not possible to reproduce the scheme of settings B and C (tailgating does not work anymore), and we are left with a single strategy that works half of the times, depending on the initial condition; this may be a realistic feature in problems with hard partial observability. We have also shown that, in this scenario, the predator's policy may be interpreted as a random search with dimensional reduction, which is a common strategy in biology.

As for the prey strategy, we have shown the possibility of engineering defences either exploiting information or direct hydrodynamic effects (or both). Strategies by both agents appears to be robust to the degree to which an agent may slightly modify them to weaken an enemy's countermeasures (predator's wiggling, prey's close range turns).

It is important to highlight that, with a more powerful function approximation scheme, such as deep reinforcement learning, different and better strategies may emerge. However, we tried to add several percepts to our scheme and no qualitative difference could be noticed: this suggest that the strategies we have presented are probably quite robust, even if likely sub-optimal.

We may add that tailgating and mirroring are found even if either or both agents are turned into pullers (not shown), but their effectiveness changes at short range. Alongside with these strategies, a new pair of conjugated policies appears: $\Phi_p + \Phi_e = \pm\pi/2$ (note the π symmetry). With either signs, the performance is similar to mirroring and their rationale is the same as well, so that we do not provide further details. It would be interesting to test if such strategies emerge with slightly different agent descriptions, such as ellipsoidal agents (which would be rotated by the strain, not just by vorticity, via de Jeffery equations). Moreover, it may be worth trying to modify the geometry of the arena in which the game takes place, for instance by adding boundaries. We have some preliminary results concerning a circular arena with no-slip boundary conditions, so that agents interacts also with their own perturbation, which is reflected by the boundaries. The phenomenology is less clean to analyze and agents spend most of their time near the boundaries; remarkably, we have noticed some traces of tailgating and mirroring, even in this setting.

In this sense, our study also incorporates the adversarial setting which has been somehow neglected in previous studies about prey/predators interactions; this is important, since robustness and effectiveness of behaviours may not be abstracted from the adversarial context in which they

are set. Hence, we have shown how reinforcement learning may be a reliable tool for approaching these problems. In conclusion, because of the idealization, we see our work as a preliminary step towards further research on the use of reinforcement learning algorithms with a twofold goal: rationalizing observed prey-predator interactions between aquatic organisms, and training underwater robots to accomplish complex tasks – e.g. artificial fishes imitating escape responses [165]. This approach may be extended beyond hydrodynamic environments and to collective pursue strategies like wolf-packing, and collective escape responses such as hydrodynamic cloaking [173].

Appendix 5.A Algorithm details, parameters and summary

Algorithm description

The natural actor critic algorithm (see subsec. 4.3.2) still suffers from a relevant issue in an adversarial setting: as the learning process progresses, the norm of the policy parameters ξ generally grows roughly linearly in time, at least in our setting. This is somehow problematic since, as the adversarial game proceeds, agents have to update their policy in response to changes in the enemy's one; unfortunately, due to the linear divergence, it becomes increasingly difficult to change a strategy so that agents are soon locked in their policies. Hence, we have to apply a regularization scheme, i.e. we need to constrain the space available to parameters in order to put an upper value to their norm. The most rigorous way would imply accounting for the parameter space curvature, an idea which translates to implementing a covariant boundary for parameters [242]. Since covariant corrections are computationally expensive, we opted for a non-covariant heuristic strategy to avoid indefinite parameter growth. Our choice implies that, even in non a adversarial setting, the fixed points of the gradient-ascent dynamics do not correspond to local optima of the cost function [242]; nontheless, in a careful implementation, we expect the bias not to be too dramatic. We chose the following regularization for (4.54):

$$\xi_{bi}(t+1) = \xi_{bi}(t) + \eta_A \, w_{bi}(t+1) - \left(\xi_{bi}(t)/\xi_0\right)^3 \tag{5.33}$$

The cubic term constrains the parameter dynamics inside a soft hyper-cube: it offers a steep barrier for $\|\xi\| \gg \xi_0$ but it has little effect when $\|\xi\| \ll \xi_0$. The rationale for this choice is that it can partially accommodate for the different intrinsic scales of the features. Our regularization has turned out to be more efficient - at least in our case - than an L^2 regularization or than a hard clipping (imposing $|\xi_{ai}| < \xi_0 \quad \forall i, a$), which is particularly disruptive of policy performance. We chose $\xi_0 = 20$.

5.A.1 Summary

Here, we provide a pseudocode for the algorithm.

Parameter initialization

$\xi^{p/e} \leftarrow 0\,; \qquad \kappa^{p/e} \leftarrow 0\,; \qquad g^{p/e} \leftarrow 0$

$\eta_C^p = \eta_C^e \leftarrow \eta_C\,; \qquad \eta_G^p = \eta_G^e \leftarrow \eta_G$

Loop on learning cycles: $(c = 1, \ldots)$

IF c is ODD $\quad \eta_A^p \leftarrow \eta_A \quad \eta_A^e \leftarrow 0 \quad$ i.e. pursuer learns

ELSE $\quad \eta_A^p \leftarrow 0 \quad \eta_A^e \leftarrow \eta_A \quad$ i.e. evader learns

Loop on episodes: $(e = 1, M)$

Initialize agents' positions and swimming orientations

observations $o_0^{p/e}$

Loop on time t: $(\{t \leq T_{max}$ OR $\quad t|R(t) \leq R_c\})$

pick action $a_t^{p/e} \sim \pi^{p/e}(a_t^{p/e}|o_t^{p/e})$

advance dynamics using Eqs. (1)-(2) of main text from t to $t + \tau$

observations $o_{t+\tau}^{p/e}$

Learning updates based on $o_t^{p/e}, o_{t+\tau}^{p/e}, a_t^{p/e}$:

Compute temporal difference:

$$\delta^{p/e} = \begin{cases} -\hat{v}^{p/e}(o_t^{p/e}) & \text{if the state is terminal} \\ r_t^{p/e} + \hat{v}^{p/e}(o_t^{p/e}) - \hat{v}^{p/e}(o_{t+\tau}^{p/e}) & \text{otherwise.} \end{cases}$$

update parameters:

$$\kappa_i^{p/e} \leftarrow \kappa_i^{p/e} + \eta_C^{p/e}\delta^{p/e}\nabla_{\kappa_i}\hat{v}^{p/e}$$

$$g_{ib}^{p/e} \leftarrow g_{ib}^{p/e} + \eta_G^{p/e}\left[\delta^{p/e} - \sum_{jc} \nabla_{\alpha_{jc}^{p/e}} \ln \pi^{p/e}(a_t^{p/e}|o_t^{p/e}) g_{jc}^{p/e}\right] \nabla_{\alpha_{ib}^{p/e}} \ln \pi^{p/e}(a_t^{p/e}|o_t^{p/e})$$

$$\xi_{ib}^{p/e} \leftarrow \xi_{ib}^{p/e} + \eta_A^{p/e} g_{ib}^{p/e} - (\xi_{ib}^{p/e}/\xi_0)^3 \qquad (\star)$$

$t \leftarrow t + \tau$

In table 5.3 we summarize the parameters used in the definition of episodes and turns, and the learning rates, which as shown in ref. [27] should be chosen such that $\eta_A \ll \eta_G \ll \eta_C$ to ensure convergence.

The dynamics (5.1)-(5.4) is integrated with a 4^{th} order Runge-Kutta scheme with time-step $dt = 0.02$ time unit, while the decision time is $\tau = 0.1$.

$T_{max} = 500$	$N_{ep} = 5000$	$\eta_A = 10^{-5}$	$\eta_G = 10^{-4}$	$\eta_C = 10^{-3}$	$\xi_0 = 20$

Table 5.3: Parameters used in the reinforcement learning.

Chapter 6

Optimal collision avoidance in swarms of active Brownian particles

Awe-inspiring examples of organized collective motions abound in a number of biological problems [183, 256] from simple microorganisms such as bacteria [269], to insects and higher animals which display deliberate social behaviors such as insect swarming [49, 232] bird flocking [12, 50], and fish schooling or shoaling [196, 207]. Most impressively, large and dense groups of animals can organize themselves in complex coordinated motions avoiding collisions while flying or swimming at close distance. Several models have been proposed to model the origin of such phenomena in terms of simple behavioral rules. A first intuition of the basic ingredients came from computer graphics [216] and entered the domain of statistical physics with the Vicsek model [255]: the core idea is that each animal in the group needs to align its heading direction with the mean direction of its neighbors. If such local alignment interactions are strong enough with respect to the unavoidable noise (in our case, on the heading directions) a transition from a disordered phase to a collective-order phase, characterized by global orientational order (alignment) of the heading directions, may occur. This idea was also applied, for instance, to the control of groups of artificial agents such as robots, where collision avoidance is crucial [247, 257]. Overall, these approaches generally build agent-based models aimed at generating certain collective behaviors starting from intuition, observations or by reverse-engineering natural phenomena via data analysis; this often leads to biomimetic algorithms for robotics.

Within this thesis we approached the problem of collision avoidance from a different perspective. We do not aim at modeling the interactions that underlie a certain collective behavior, but instead we consider a simple model of swarming agents and explicitly set the goal of avoiding collisions in the form a cost function and ask the following questions: What is the optimal choice of control which minimizes the collective cost? How does the OC compare with known agent-based models which lead to collision avoidance?

The natural setting to answer the above questions is that of optimal control theory [243] (see chapter 4 for a broader introduction) and mean-field game formalism [146, 249], in which an individual agent is assumed to interact with the mean behaviour of other agents, in a self consistent way. Similar approaches, indeed, have already been shown to yield promising results. For instance, for collective search problems the optimal control reduces in some limit to a well known model of chemo-

taxis [203], for the problem swarming agents in one-dimensional disordered environments [112], and also for flocking problems with the aid of reinforcement learning techniques [74].

In what follows, we start by introducing the setting of the problem in Sec. 6.1.1 where also the optimal control formalism is presented.

We model agents as active Brownian particles [22, 218] which move in two-dimensions and whose heading direction is subject to rotational noise. They try to avoid collisions by exerting some control on their heading direction, in the form of a torque.

Collisions lead to a cost, but also the control itself is not free of charge, and for that we assume a quadratic dependence in the angular velocity. The cost for control can either be understood in terms of power dissipation, physical limitations of an animal/robot, or in terms of the cognitive cost of deviating from free spontaneous behavior [243], as discussed in sec. 4.2, specifically eq. (4.20). Therefore, an agent is interested in applying a non-trivial control to its motion only to the extent to which the gain outweighs the cost: the optimal strategy emerges from this tradeoff. This cost minimization problem can be exactly mapped into a quantum many-body problem which is unfortunately hard to solve in general. For this reason, we introduce a mean-field approximation (Sec. 6.1.2) that reduces the many-body problem to a quantum pendulum, which is exactly solvable.

Under the mean field-approximation, agents are assumed to be homogeneously distributed. While this is unrealistic under many respects, it can suitably describe the optimal behavior over an approximately uniform region in the bulk of a swarm. We show that all relevant parameters combine into a fundamental tradeoff parameter h, which effectively accounts for the balance between the collision and control costs. Upon increasing such tradeoff parameter, the system displays a second order phase transition in terms of the polar order parameter (a measure of the mean-field alignment) at a certain critical value h_c. Then, we study some relevant observables, such as the cost, the polar order and the susceptivity – which is known to be important in collective motions [176] – both near the critical point (Sec. 6.1.3) and in the strong coupling regime (large h, corresponding to collisions costs dominating, see Sec. 6.1.4). Remarkably, in both limits, the optimal control is well approximated by a sinusoidal function of the difference between the individual heading direction (angle) and the mean one Interestingly, the sinusoidal control is a distinctive trait of the well-known Vicsek-like models [62, 81, 256] and its mean-field versions [60, 61, 201]. This observation motivates a *vis à vis* comparison between the optimal and sinusoidal control. For a sound comparison, we first find the sinusoidal control which minimizes the total cost (sec. 6.2.1). Remarkably, such best sinusoidal model is controlled by the same tradeoff parameter, and the polar order turns out to display a second order transition at the same critical point as for the optimal model; with analytical tools, we explore this regime along with the strong coupling one. Finally, we proceed with a systematic comparison (Sec. 6.2.2) in the whole range of the tradeoff parameter, showing how the optimal solution, while close to its sinusoidal approximation, can better manage the collisions, and that the sinusoidal model becomes the exact optimum in the strong coupling (large h) limit. We end the chapter with some discussions and perspectives.

6.1 Optimal solution of the collision problem

6.1.1 Collision minimization as an optimal control problem

We consider a group of N agents in two dimensions whose goal is to swarm together while avoiding collisions with each other (see fig. 6.1). We model the agents as active Brownian particles [22, 218]: each agent i is a self-propelled particle moving with a constant speed u_0 in a direction identified by an angle θ_i (or, equivalently, by the associated unitary vector $\boldsymbol{n}(\theta_i) = (\cos\theta_i, \sin\theta_i)$) which randomly changes due to rotational noise with diffusivity D. Each agent, to avoid collisions, can exert some control, f_i, on its angular velocity and possibly contrast the rotational noise. The controls f_i are, in the most general case, functions of all positions, $\boldsymbol{x}_j$, and heading directions, θ_j, of all agents ($j = 1, \dots, N$). Generalizing eq. (4.1) from sec. 4.2, the dynamics of agent i thus reads

$$\begin{cases} d\boldsymbol{x}_i = u_0\, \boldsymbol{n}(\theta_i) dt \\ d\theta_i = f_i(\boldsymbol{x}_i, \theta_i; \{\boldsymbol{x}_j, \theta_j\}_{j\neq i})\, dt + \sqrt{2\,D}\, d\xi_i\,, \end{cases} \tag{6.1}$$

where the noise term in the angular dynamics is a zero mean, $\langle d\xi_i(t)\rangle = 0$, Gaussian process with correlation $\langle d\xi_i(t)\, d\xi_j(t')\rangle = \delta_{ij}\, \delta(t-t')\, dt$. We assume periodic boundary conditions, since we are only interested in the bulk interactions within the swarm. This choice will not be relevant for the rest of the paper.

When particle pairs, say i and j, collide they pay a cost $G_{ij} = \delta(\boldsymbol{x}_i - \boldsymbol{x}_j)\, \mathcal{G}(\theta_{ij})$ with $\mathcal{G}(\theta_{ij})$ representing the functional dependence of cost on the collision angle, $\theta_{ij} = (\theta_i - \theta_j)$. By expanding $\mathcal{G}(\theta) = g_0 + g_1 \cos\theta + g_2 \cos(2\theta) + \dots$ into (even) harmonics and truncating after the second term, we obtain $G_{ij} = \delta(\boldsymbol{x}_i - \boldsymbol{x}_j)\, (g_0 - g_1\, \boldsymbol{n}(\theta_i) \cdot \boldsymbol{n}(\theta_j)) = \delta(\boldsymbol{x}_i - \boldsymbol{x}_j)\, (g_0 - g_1 \cos\theta_{ij})$. As we will see, the cost per contact $g_0 > 0$ will be somehow unimportant but its relationship with angular cost $g_1 > 0$ allows for different model interpretations. For instance, if $g_0 = 0$ we have a pure alignment

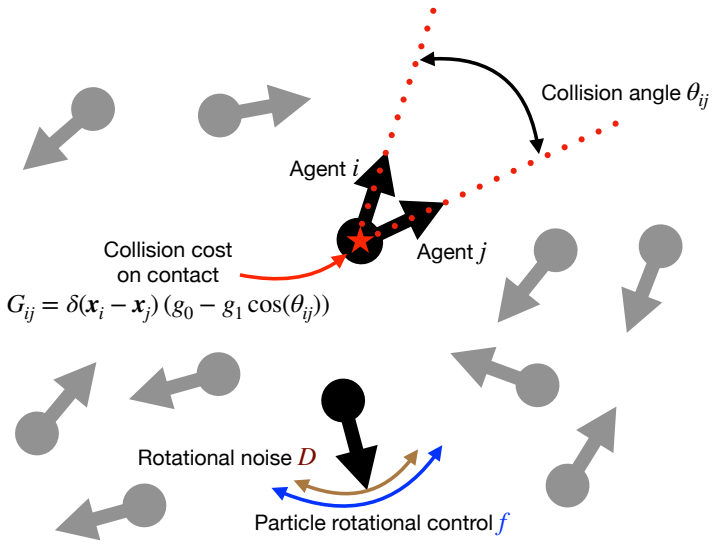

Figure 6.1: Sketch of the swarming active Brownian particles. The black particle on the bottom illustrates the angular dynamics influenced by rotational noise (brown arrows) and the control (blue arrows). The couple of black particles on the top, denoted i and j, illustrates the cost paid for each contact, depending on the collision angle. The agents can avoid the collisions by controlling their angular velocity, but pay a cost for it.

problem, while for $g_1 = g_0$ we have a pure collision-based model, as in the latter case the collision cost is proportional to the relative velocity which is exactly zero when velocities are aligned.

Agent i can partially control its heading direction by imparting an angular velocity f_i but it pays a cost $\alpha\, f_i^2/2$. The quadratic choice for the cost of control, besides being quite natural when interpreted in terms of power dissipation, has an information-theoretical foundation as the cost (measured in terms of the Kullback-Leibler divergence) of deviating from a random control strategy [243] (see sec. 4.2). The total cost per unit time - the sum of individual costs - reads

$$C(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N) = \frac{\alpha}{2} \sum_i f_i^2 + \frac{1}{2} \sum_{i \neq j} \delta(\boldsymbol{x}_i - \boldsymbol{x}_j)\, (g_0 - g_1\, \boldsymbol{n}(\theta_i) \cdot \boldsymbol{n}(\theta_j))\,. \tag{6.2}$$

Notice that the parameter $\alpha > 0$ can be reabsorbed in the definition of g_0 and g_1, since we are only interested in the optimal strategy, while it would have played a role in risk-sensitive scenarios [75, 115, 203].

The agents collective goal is to choose the controls that minimize the average total cost $\bar{C} = \int \prod_{k=1}^N d\boldsymbol{x}_k\, d\theta_k\; C(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N)\; P(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N)$, with P being the stationary joint probability density of particles positions and angles. The non-trivial point is that P itself depends on the controls $\{f_i\}_{i=1}^N$ and should be determined as part of the solution. In particular, the joint probability P, besides the normalization constraint $\int \prod_{k=1}^N d\boldsymbol{x}_k\, d\theta_k\; P(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N) = 1$, must be the stationary solution of the Fokker-Planck equation associated to Eq. (6.1), which reads

$$\sum_{i=1}^N \left[-u_0\, \partial_{\boldsymbol{x}_i}\, \boldsymbol{n}(\theta_i) - \sum_i \partial_{\theta_i}\; f_i + D\; \sum_i \partial_{\theta_i}^2 \right] P = \sum_{i=1}^N \mathcal{L}_i\, P = \mathcal{L}_{(N)} P = 0\,, \tag{6.3}$$

where $\mathcal{L}_i$ is the single-agent linear Fokker-Planck operator and $\mathcal{L}_{(N)} = \sum_i \mathcal{L}_i$ the N-bodies one. By minimizing the total cost, we are looking for a cooperative solution to the problem. As illustrated in sec. 4.2, the solution of the constrained minimization can obtained by a generalized Lagrange-multipliers technique, or namely, by finding the stationary points of the auxiliary functional[1] (Pontryagin principle [208])

$$\mathcal{H} = \lambda + \int \prod_{i=1}^N d\boldsymbol{x}_i\, d\theta_i\; [C - \lambda - \Phi\, \mathcal{L}_N]\, P\,. \tag{6.4}$$

The normalization and dynamical constraints are obtained by imposing stationarity w.r.t. (with respect to) the multipliers λ and $\Phi(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N)$, respectively.[2] The non-trivial results come from the request of stationarity w.r.t. P and f_i, which yields

$$\begin{cases} \dfrac{\delta \mathcal{H}}{\delta f_i} = 0 \implies f_i = \partial_{\theta_i} \Phi & (6.5) \\ \dfrac{\delta \mathcal{H}}{\delta P} = 0 \implies C - \lambda - u_0 \displaystyle\sum_i \boldsymbol{n}(\theta_i) \cdot \partial_{\boldsymbol{x}_i}\, \Phi - \sum_i f_i\, \partial_{\theta_i}\, \Phi - D\; \sum_i \partial_{\theta_i}^2 \Phi = 0\,. & (6.6) \end{cases}$$

Equation (6.6) is the Hamilton-Jacobi-Bellman equation associated to the optimal control problem. It can be linearized via the Hopf–Cole transform (cfr sec. 4.2), $\Phi = 2D \log Z$, by introducing the

[1] The minus signs in Eq. (6.4) are chosen for the convenience of notation.

[2] Notice that Φ is a function because $\mathcal{L}_N P = 0$ must be imposed for all angles and positions.

desirability function $Z(\boldsymbol{x}_1, \theta_1; ...; \boldsymbol{x}_N, \theta_N)$ [243]. Then the control (6.5) becomes

$$f_i = 2\,D\,\partial_{\theta_i} \ln Z\,, \tag{6.7}$$

that is a gradient ascent towards more desirable configurations, hence the name. Thanks to the Hopf–Cole transform, eq. (6.6) becomes the linearized Bellman equation

$$\frac{\lambda}{2\,D}\,Z - \frac{1}{4D}\sum_{i\neq j}\delta(\boldsymbol{x}_i - \boldsymbol{x}_j)\,(g_0 - g_1\,\boldsymbol{n}(\theta_i)\cdot\boldsymbol{n}(\theta_j))\,Z + u_0\sum_i \boldsymbol{n}(\theta_i)\cdot\partial_{\boldsymbol{x}_i}\,Z + D\,\sum_i \partial^2_{\theta_i}\,Z = 0\,. \tag{6.8}$$

which is formally identical to the stationary Schrödinger equation of N identical, interacting bosons. We should solve both for the ground-state eigenvalue $\lambda/2D$, which can be shown to be proportional to the total cost[3], and the eigenfunction Z, requiring Z to be real and positive. To our knowledge, this quantum many-body problem has no known general solution for generic N; therefore, we will seek for the optimal control in an approximate mean-field setting.

6.1.2 Mean-field approximation

To simplify the problem and make it exactly solvable, we proceed with a mean-field approximation based on two hypothesis: first we assume agent-wise factorization of the desirability Z and then we assume spatial homogeneity - no preferred points in space, only preferred directions. The agent-wise factorization excludes direct pairwise interactions – agents cannot directly dodge each other – but, rather, each agent interacts with the joint probability of the remaining $N-1$ ones in a self consistent manner. This approximation is rather strong since in animal collective behavior it would make more sense to consider local interactions [50]. It must also be remarked that the homogeneity assumption excludes from the description many interesting phenomena related to heterogeneities. Notwithstanding these limitations, we can still assume that this treatment could be relevant to describe agents within a uniform bulk region of the swarm.

With the factorization and homogeneity assumptions, the desirability can be written as

$$Z(\boldsymbol{x}_1, \theta_1, ..., \boldsymbol{x}_N, \theta_N) = \prod_{i=1}^{N} \zeta(\theta_i)\,, \tag{6.9}$$

and, equivalently $\Phi(\boldsymbol{x}_1, \theta_1, ..., \boldsymbol{x}_N, \theta_N) = \sum_{i=1}^{N}\phi(\theta_i)$. As a consequence, the probability P is factorized as $P(\boldsymbol{x}_1, \theta_1, ..., \boldsymbol{x}_N, \theta_N) = \prod_{i=1}^{N} p(\boldsymbol{x}_i, \theta_i)$ and, owing to spatial homogeneity, we can write $p(\boldsymbol{x}_i, \theta_i) = \frac{1}{V}\rho(\theta_i)$, with V being the area where the swarm moves.

We define the agents' average heading direction $\bar{\theta}$ and the polar order parameter (alignment parameter or polarization) m as

$$m\,\boldsymbol{n}(\bar{\theta}) = \int d\theta'\;\boldsymbol{n}(\theta')\,\rho(\theta')\,, \tag{6.10}$$

in terms of which the average agent speed reads $\langle\dot{\boldsymbol{x}}\rangle = m\,u_0\,\boldsymbol{n}(\bar{\theta})$; here and in the sequel, since all particles are equivalent by mean-field ansatz, we drop particle indices.

[3]Note that, formally, the HBJ equation (6.6) can be written as $C - \lambda - \mathcal{L}^\dagger\Phi = 0$ with $\mathcal{L}^\dagger$ being the adjoint of the Fokker-Planck operator. Taking the average with respect to P we get $\bar{C} - \lambda - \int P\mathcal{L}^\dagger\Phi = 0$. Since, at the stationary point, $\mathcal{L}P = 0$ holds, we can deduce that $\int P\mathcal{L}^\dagger\Phi = \int \Phi\mathcal{L}P = 0$ from which $\bar{C} = \lambda$ follows.

By defining the parameter $\delta = (N-1)/V$, which is the particle density measured by a reference agent, and $C_0 = \delta\, g_0/2$, with a few straightforward passages, we write the average per agent cost as

$$\bar{C} = C_0 + \int d\theta\ \rho(\theta)\ \left[-\frac{\delta\, m\, g_1}{2}\ \cos(\theta - \bar{\theta}) + \frac{1}{2} f^2\right], \tag{6.11}$$

and the functional (6.4) as

$$\mathcal{H} = \lambda + \bar{C} - \int d\theta\ [\lambda + \phi(\theta)\ \mathcal{L}\,]\rho(\theta)\,, \tag{6.12}$$

with $\mathcal{L} = -\partial_\theta f + D\,\partial_\theta^2$ being the single-particle Fokker-Planck operator as in Eq. (6.3). Proceeding analogously to the general case and by exploiting the Hopf-Cole transform with the factorized desirability (6.9), we derive the control to be

$$f(\theta) = 2\,D\,\frac{d}{d\theta}\ln\zeta\,. \tag{6.13}$$

Plugging the last expression into the stationary Fokker-Planck equation, $\mathcal{L}\,\rho = 0$, with periodic boundary conditions, yields

$$\rho = \zeta^2\,, \tag{6.14}$$

with $\int d\theta\ \zeta^2 = 1$. Therefore, the OC problem boils down to solving the self-consistent system of equations

$$\begin{cases} m = \displaystyle\int d\theta\ \cos\theta\ \zeta^2 & (6.15) \\ \left[\dfrac{\lambda}{2D} + \dfrac{\delta\, m\, g_1}{2D}\ \cos\theta\right]\ \zeta + D\,\partial_\theta^2\zeta = 0\,. & (6.16) \end{cases}$$

Equation (6.15) is just Eq. (6.10) where we used (6.14) and fixed $\bar{\theta} = 0$ with no loss of generality, as the rotational symmetry can be broken in an arbitrary direction, while Eq. (6.16) is the mean-field linearized Bellman equation with $\bar{\theta} = 0$. Formally, Eq. (6.16) is the stationary Schrödinger equation for a quantum pendulum also known as Mathieu equation that, for consistency with literature [41, 102], we rewrite in the canonical form

$$[a - 2\,q\,\cos(2\,y)]\ \zeta + \zeta'' = 0. \tag{6.17}$$

with $y = \theta/2$, $a = -\lambda/(2\,D^2)$, $q = -\delta\, m\, g_1/D^2$, and $''$ denoting the second derivative with respect to y. Equation (6.17) must be solved both for the eigenvalue $-a$ (which corresponds to solving for λ) and the eigenfunction $\zeta = \zeta_m$, which depends parametrically on m. The solutions are the so-called Mathieu functions, as briefly recalled in Appendix 6.A. For periodic boundary conditions, the Mathieu ground state eigenfunction, denoted as ce_0 in the literature, is an even function with a single maximum in $y = 0$ and a minimum in $y = \pi/2$. The associated eigenvalue is called characteristic Mathieu function $a = a(q)$, which is non-positive and takes the asymptotic expressions $a(q) \sim -q^2/2$ (see Eq. (6.23)) and $a(q) \sim 2\,q + 2\,\sqrt{-q}$ (see Eq. (6.36)) for $q \to 0$ and $q \to -\infty$, respectively.

In order for an eigenfunction ζ_m to be a solution of the optimization problem, it must also satisfy

Parameter	Description
D	rotational diffusivity
g_0 and g_1	positional and angular collision cost
α	weight of the cost of control
m	polar order parameter/polarization
$\delta = (N-1)/V$	density of particles
$h = \delta\, g_1/D^2$	tradeoff parameter
$q = -m\, h$	Mathieu equation parameter

Table 6.1: Summary of main parameters.

Mean-field glossary	
ζ	desirability
ρ	single particle angular distribution
$\rho = \zeta^2$	desirability & angular distribution relation
$f = 2D\frac{d}{d\theta}\ln\zeta$	optimal control
$m = \mathcal{F}(m)$	self-consistency equation

Table 6.2: Summary of functions and main relations for the mean-field model.

the self-consistency condition (6.15) which, using Eq. (6.16), reads

$$m = \int d\theta\, \cos\theta\, \zeta_m^2 = \mathcal{F}(m). \tag{6.18}$$

In the sequel we will drop the subscript in ζ_m whenever that would not hinder clarity.

The function $\mathcal{F}(m)$, shown in fig. 6.2a, depends on the parameter (see Table 6.1 for a handy

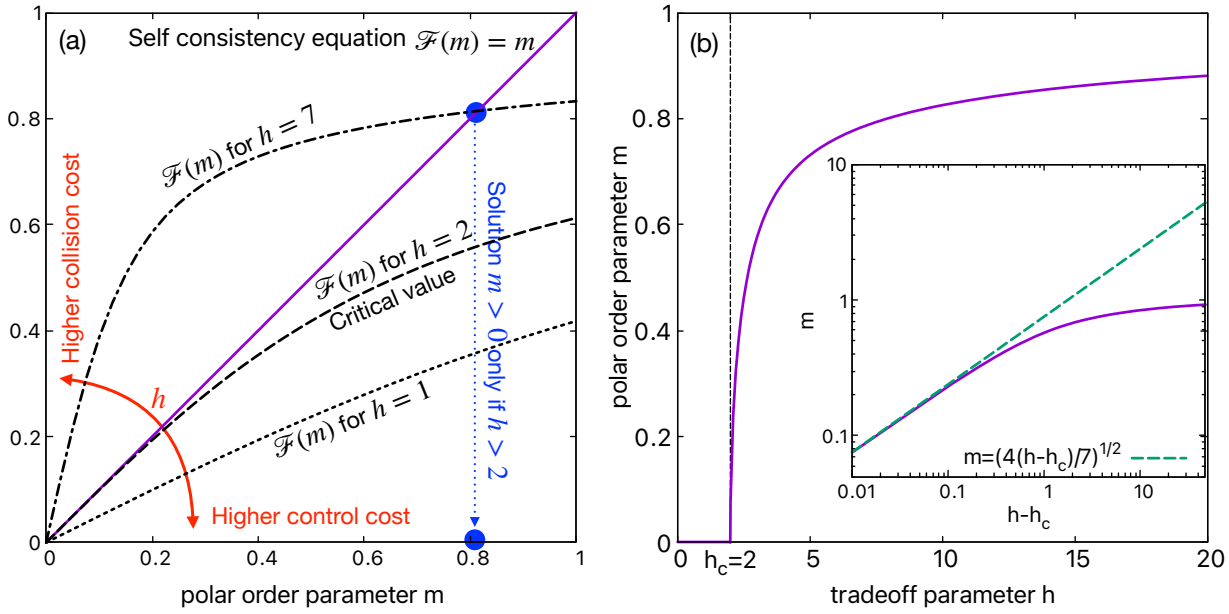

Figure 6.2: Self-consistency equation and polar order parameter: (a) graphical solution of Eq. (6.18), $\mathcal{F}(m) = m$ can only be satisfied with $m > 0$ if $\mathcal{F}' > 1$, which requires the tradeoff parameter $h = (N-1)\, g_1/(D^2\, V)$ to be larger than $h_c = 2$; (b) polar order parameter m as a function of h, a second order phase transition takes place at $h = h_c$. The inset displays the asymptotic approximation (6.26) (green dashed line) showing the critical exponent to be 1/2.

summary of all the parameters of the problem and Table 6.2 for main relations and functions)

$$h = -\frac{q}{m} = \frac{\delta\, g_1}{D^2}\,, \tag{6.19}$$

which, besides containing all dependencies on the problem parameters N, g_1, D, V, has a natural interpretation as the ratio between the parameters which control the importance of collision and control costs. For a clearer picture, we reintroduce α and write as $h = (g_1\,\delta)/(\alpha\, D^2)$: at the numerator, g_1 is rescaled with particle density δ while, at the denominator, α is rescaled with diffusivity D. Note that the solution does not depend on g_0, as follows from the homogeneity assumption. As graphically illustrated in fig. 6.2a, Eq. (6.18) admits only the trivial solution with no polar order $m = m(h) = 0$ for $0 \leq h \leq h_c = 2$, while a non-trivial solution emerges, via a second order transition to non-zero alignment ($m > 0$) for $h > h_c$ (the critical value $h_c = 2$ is derived in the next subsection by a perturbative expansion of the self-consistency equation). The numerically computed function $m(h)$ is shown in fig. 6.2b. The polar oder parameter m is zero up to the critical tradeoff value $h_c = 2$, meaning that the alignment benefit outweighs the cost of control only for $h > h_c$. Therefore, for $h < h_c$, no control is applied and the system remains isotropic so that the average cost $\bar{C}$ (6.11) is equal to $C_0 = \delta g_0/2$ while, for $h > h_c$, $\bar{C}$ is equal to

$$\bar{C} = C_0 + \frac{1}{2} D^2\, \left[h\, m^2(h) + a(-m(h)\, h)\right]. \tag{6.20}$$

To see this observe that the mean-field costs (6.11) can be written as

$$\bar{C} - C_0 = -\frac{D^2\, h}{2}\, m \int d\theta\, \zeta^2\, \cos\theta + 2\, D^2 \int d\theta\, \zeta^2\, (\partial_\theta \ln \zeta)^2 = -\frac{D^2\, h}{2}\, m^2 + 2\, D^2 \int d\theta\, (\partial_\theta \zeta)^2. \tag{6.21}$$

We can apply partial integration to the control cost term $\int d\theta\, (\partial_\theta \zeta)^2 = -\int d\theta\, \zeta\, \partial_\theta^2 \zeta$. By using the Mathieu equation (6.55), we can write $\int d\theta\, (\partial_\theta \zeta)^2 = a/4 + m^2\, h/2$. Equation (6.20) follows from a straightforward substitution.

Note that, as anticipated, the constant C_0 is irrelevant to the optimization process (as a consequence of the mean-field assumptions), while the remaining part depends only on the tradeoff parameter h, up to the D^2 prefactor.

In the following sections, we will give a more detailed description of both the critical behavior (for $h \to h_c = 2$) and the strong coupling (large h) regime.

6.1.3 Critical behavior

As clear from fig. 6.2, at the critical point $h = h_c = 2$, there is a second order phase transition in the polar order parameter with exponent $1/2$, a classical mean-field value, which can be derived as follows. When $h = h_c^+$, we can solve Eqs. (6.15) and (6.16) perturbatively near the critical point. We start from the Mathieu equation (6.55) and we observe that $m \to 0$ implies $q = -h\, m \to 0$ (see Table 6.1) for finite h. We solve the equation perturbatively by expanding the solution ζ and the eigenvalue a in power series of q, by writing

$$\begin{cases} \zeta = \zeta_0 + \zeta_1 + \zeta_2 + \zeta_3 + \zeta_4 + o(q^4) \\ a = a_0 + a_1 + a_2 + a_3 + a_4 + o(q^4). \end{cases} \tag{6.22}$$

where $a_k = O(q^k)$ and $\zeta_k = O(q^k)$ (for instance in C_∞ norm). At any order k, we impose $\zeta'' + (a - 2\,q\,\cos(2\,y))\,\zeta = o(q^k)$ and $\int_{-\pi/2}^{\pi/2} dy\,\zeta^2 = 1 + o(q^k)$, starting from order 0 to 4. Clearly, for any k, ζ_k and a_k only depend on $\{\zeta_s\,, a_s\}_{s<k}$. From this procedure, we find that the eigenvalue is

$$a(q) = -\frac{1}{2}\,q^2 + \frac{7}{128}\,q^4 + o(q^4), \tag{6.23}$$

while the eigenfunction can be written as the following expansion in harmonics

$$\zeta = \frac{1}{\sqrt{\pi}}\left[1 - \frac{q^2}{16} + \left(-\frac{q}{2} + \frac{11\,q^3}{128}\right)\cos(2\,y) + \frac{q^2}{32}\,\cos(4\,y) - \frac{q^3}{1152}\,\cos(6\,y)\right] + o(q^3). \tag{6.24}$$

From expansion (6.24), the self-consistency condition (6.18) can be written as

$$m = \int_{-\pi/2}^{\pi/2} dy\,\zeta_0^2\,\cos(2\,y) = \frac{h\,m}{2} - \frac{7\,(m\,h)^3}{64} + o(m^3)\,, \tag{6.25}$$

which has 3 solutions: the trivial one $m = 0$, an unphysical solution $m < 0$ and

$$m \approx \sqrt{(4/7)\,(h - h_c)}\,, \tag{6.26}$$

which is well defined only for $h > h_c = 2^+$, where a second order phase transition occurs. This yields the asymptotic behavior and the critical exponent 1/2. As shown in the inset of fig. 6.2b, eq. (6.26) fully captures the critical behavior. From eq. (6.24) we can see that, in this regime, the desirability ζ can be approximated at leading order in m as

$$\zeta(\theta) = \frac{1}{\sqrt{2\,\pi}}\left[1 + \frac{h\,m(h)}{2}\,\cos\theta\right]\,. \tag{6.27}$$

Since the angular probability density function satisfies eq. (6.14), agents directions are uniformly distributed but for a tiny $O(m)$ cosine modulation. Plugging eq. (6.27) into eq. (6.13), the optimal control at the critical point reads

$$f = -D\,h_c\,m\,\sin(\theta) + o(m)\,. \tag{6.28}$$

Moreover, exploiting the asymptotic expressions (6.23) for a into eq. (6.20), we can obtain the cost close to the critical point $h - h_c = 0^+$:

$$\bar{C} - C_0 = -(D^2/7)\,(h - h_c)^2 + o((h - h_c)^2)\,. \tag{6.29}$$

We conclude the investigation of the critical properties by studying the susceptivity to external perturbations, which is an important observable in multi-agent systems, both when considering artificial swarms control and biological collective behaviors, such as bird flocks or insect swarms. Indeed, the susceptivity describes the crowd sensitivity to fluctuations and/or external stimuli [28, 49]. In order to define the susceptivity, we need to specify an external field, and then compute the derivative of m with respect to it. Consistently with the optimization setting we are considering, the external field is represented by a small collective nudge of intensity ϵ in the direction $\hat{\theta}$, which

formally amounts to adding a small per-agent reward in the cost function

$$\delta G_i = -\epsilon \, \cos(\theta_i - \hat{\theta}) \,, \tag{6.30}$$

for aligning along the direction $\hat{\theta}$. The perturbation (6.30) breaks the isotropy favoring an average alignment in the preferred direction $\hat{\theta}$ (which, with no loss of generality, can be set to 0). Since $\hat{\theta}$ breaks the isotropy, and the system has no intrinsic preferred direction, we can deduce that the system will polarize along the preferred direction $\bar{\theta} = \hat{\theta}$. Therefore, susceptivity is then defined as

$$\chi(h) = \left. \frac{\partial m}{\partial \epsilon} \right|_{\epsilon=0} . \tag{6.31}$$

Essentially, the additional (negative) cost (6.30) induces the shift

$$q \mapsto q + \frac{2\epsilon}{D^2} \,, \tag{6.32}$$

which, plugged into Eq. (6.18), implicitly defines m as a function of ϵ, for any h. Then, by a straightforward application of Dini's implicit function theorem (see Appendix 6.B), we obtain the following result: when $h < h_c$, $\chi = 2/[D^2\,(h_c - h)]$ holds exactly; close to the critical point ($h \to h_c^+$), $\chi = 1/[D^2\,(h - h_c)] + o(1/(h - h_c))$. Therefore, near the critical point, we have that the susceptivity diverges with $h \to h_c$ as

$$\chi \sim |h - h_c|^{-1} \,, \tag{6.33}$$

we notice that the critical exponent 1 for the susceptivity is also quite standard in mean-field theories. On the other hand, exactly at the critical point $h = h_c$, m depends on ϵ as $m \sim 2 \left[\frac{\epsilon}{7\,D^2}\right]^{1/3}$ and, hence, m is a continuous but non differentiable function of ϵ at the critical point and, consequently, the susceptivity diverges as

$$\chi \sim \epsilon^{-2/3} \,. \tag{6.34}$$

The absence of a first order discontinuity implies that, at the critical point, an infinitesimal nudge is not enough to induce finite polarization in this kind of system. This a natural consequence of the continuous symmetry which is spontaneously broken even in absence of external perturbations. On the other hand, any small nudge is enough to fix the direction of the polarization vector.

6.1.4 Strong coupling

The strong coupling regime is identified by the condition $h \gg h_c$, which can be achieved with high density ($\delta = (N-1)/V \gg 1$), with a high collision cost coefficient g_1 or with low noise D. In the large h limit, we can analytically solve Eqs. (6.15) and (6.16). For this purpose, we assume $m \approx 1$, from which it follows that $q(h) = -h\,m(h)$ becomes large negative. For large q, then both ζ and ρ are peaked around $y = 0$ and, therefore, we can assume a regime of small oscillations $|y| \ll 1$. Then, by expanding $\cos(2\,y) \approx 1 - 2\,y^2$, as one would expect from elementary physics courses, Eq. (6.55)) reduces to the well-known Schrödinger equation of the quantum harmonic oscillator:

$$-\frac{1}{2}\zeta'' + \frac{1}{2}\underbrace{(-4\,q)}_{\omega_0^2}\, y^2\, \zeta = \underbrace{\frac{a - 2\,q}{2}}_{E}\, \zeta. \tag{6.35}$$

From the ground state eigenvalue solution $E_0 = \omega_0/2$, we get the large q approximation of the characteristic function a

$$a(q) \approx 2\,q + 2\,\sqrt{-q} \text{ for } q \ll -1\,. \tag{6.36}$$

Conversely, from the ground state eigenfunction $\exp(-\omega_0/2\,y^2)$, we get

$$\zeta(\theta) \sim \left(\frac{2\sqrt{h\,m(h)}}{\pi}\right)^{1/4} \exp(-\sqrt{h\,m(h)}\,\theta^2)\,, \tag{6.37}$$

where we have extended the domain of y from $[-\pi/2, \pi/2]$ to $(-\infty, \infty)$ by enforcing normalization $\int_{-\infty}^{\infty} dy\ \zeta^2(y) = 1$. We can then rewrite Eq. (6.18) as $\int_{-\infty}^{\infty} dy\ \zeta^2(y)\,\cos(2\,y) = m$ which becomes

$$m = e^{-\frac{1}{2\sqrt{hm}}}. \tag{6.38}$$

Hence, if $h \to \infty$, then $m \to 1$, validating our small-oscillations ansatz. More precisely

$$1 - m \sim \frac{1}{2\,\sqrt{h}} \text{ for } h \to \infty\,. \tag{6.39}$$

The associated asymptotic control is linear in θ

$$f = -D\,\sqrt{h\,m(h)}\,\theta\,, \tag{6.40}$$

where $m \approx 1$ from (6.39). Note that, since this solution has been obtained in the small oscillations regime, Eq. (6.40) is only accurate around $\theta = 0$ which is, on the other hand, the only region which matters, since the probability of visiting other regions is exponentially suppressed. Exploiting the large h asymptotics for the Mathieu characteristic function (6.36), one can easily obtain that the cost decreases approximately linearly with h as $\bar{C} - C_0 = D^2\,[-h/2 + (3/4)\,\sqrt{h}] + o(\sqrt{h})$. Using the same scheme of the previous section, we can compute the susceptivity in this regime as well obtaining that χ (6.31) vanishes for large h as $\chi \sim 1/[2\,D^2\,h^{3/2}\,m^{1/2}]$ (see Appendix 6.B). Therefore, as we might have expected, the polar order strength m is little responsive to external nudges, when it is already close to its maximum value 1.

6.2 Sinusoidal control vs optimal solution

The optimal solution (6.28) to the collision avoidance problem in swarming active Brownian particles suggests that close to the critical point the optimal control is sinusoidal at leading order, i.e.

$$f = -D\,K\ \sin(\theta - \bar{\theta})\,, \tag{6.41}$$

with $K \approx h_c m$; notice that in Eq. (6.28) the mean field direction was put to zero ($\bar{\theta} = 0$) for the sake simplicity and here restored for clarity. Interestingly, also the strong coupling optimal solution (6.40) is well approximated by the sinusoidal control (6.41): owing to the small oscillations property $|\theta - \bar{\theta}| \ll 1$, we can replace $\theta - \bar{\theta}$ with $\sin(\theta - \bar{\theta})$ in Eq. (6.40), obtaining $K \approx \sqrt{hm}$. Therefore, in both asymptotics ($h \to h_c^+$ and $h \to \infty$), Eq. (6.41) approximates the optimal control with a rescaled control strength K, which only depends on the parameter h in both cases. This is quite

noteworthy as the search of the optimal control is done without any constraints on the functional form of the control.

Remarkably, the control (6.41) is reminiscent of the mean field versions of the Kuramoto model [144] with zero natural frequencies, which is a paradigm for synchronization [2], and of the (time-continuous) stochastic Vicsek model [60–62, 201], which is a variant of one of the most popular models used for describing collective motions and swarming of self-propelled agents [94, 256]. In the mean-field version of the latter [60, 62, 201], individual agents are driven by the approximate control $f_V = -R\, m\, \sin(\theta - \bar{\theta})$, corresponding to Eq. (6.41) upon defining

$$R = D\, K/m \tag{6.42}$$

whenever the polar order parameter $m = m(K, D, g_1, V)$ is non zero.

Given the similarity, both in the critical and in the strong coupling limit, of the optimal control with the classical models discussed above it is worth to compare *vis a vis* the optimal control solution with the class of "sinusoidal control models" defined by Eq. (6.41). In particular, we aim at comparing the optimal control with the "best" sinusoidal control, defined by Eq. (6.41) with $K = K_\star$, with $K_\star$ being the value of K that minimizes the average cost, given the parameters of the problem (N, g_1, V, D). As we will see, actually best sinusoidal model will depend on the familiar combination $h = (N-1)g_1/(VD^2)$, so that $K_\star = K_\star(h)$ is a function of the tradeoff parameter h only.

In the following subsection, after obtaining the best sinusoidal control, we briefly discuss the critical and strong coupling regimes and, then, we compare the optimal and best sinusoidal control for arbitrary tradeoff parameter values.

6.2.1 Derivation of the best sinusoidal model

The dynamics of the heading direction for a generic agent with the sinusoidal control (6.41) is $d\theta = -K\, D\, \sin\theta dt + \sqrt{2D}\, d\xi$, where ξ is the usual Wiener noise and where, again, we assume $\bar{\theta} = 0$ for the sake of notation simplicity. Note that the above dynamics also describes the orientation of gravitactic (bottom-heavy) microorganisms in two dimensions, where $1/KD$ is the time scale with which the organism orients vertically upward oppositely to gravity [54, 197]. The Fokker-Planck equation associated with such dynamics, $\partial_\theta(-KD\sin\theta - D\partial_\theta)\rho_s = 0$ is solved by the Fisher-Von Mises distribution [166]

$$\rho_s(\theta) = \frac{1}{2\,\pi\, I_0(K)} e^{K\,\cos\theta}\,, \tag{6.43}$$

where $I_\alpha(z)$ is the modified Bessel function of the first kind of order α (briefly surveyed in Appendix 6.C) and where the subscript s is used to remind that it pertains to the sinusoidal control. Once the distribution is known, we can compute the polar order parameter as

$$m_s(K) = \int d\theta \rho_s(\theta)\cos\theta = \frac{I_1(K)}{I_0(K)}\,. \tag{6.44}$$

As stated, the best sinusoidal model is given by Eq. (6.41) with such $K_\star$ that minimizes the average cost (6.11). By plugging Eqs. (6.43) and (6.41) into Eq. (6.11), after a few trigonometric passages

combined with the identity (6.67), the sinusoidal control cost $\bar{C}_s$ can be written as

$$\bar{C}_s[K, D] = \int d\theta\, \rho_s\, C_s = C_0 + D^2 \left[-\frac{h}{2}\, m_s^2(K) + \frac{K}{2}\, m_s(K) \right], \tag{6.45}$$

from which one can easily deduce that $K_\star = K_\star(h)$, as anticipated. Unfortunately, it is impossible to minimize the cost (6.45) analytically with respect to K, so we proceeded numerically. In fig. 6.3a we show both $K_\star(h)$ as a function of the tradeoff parameter h.

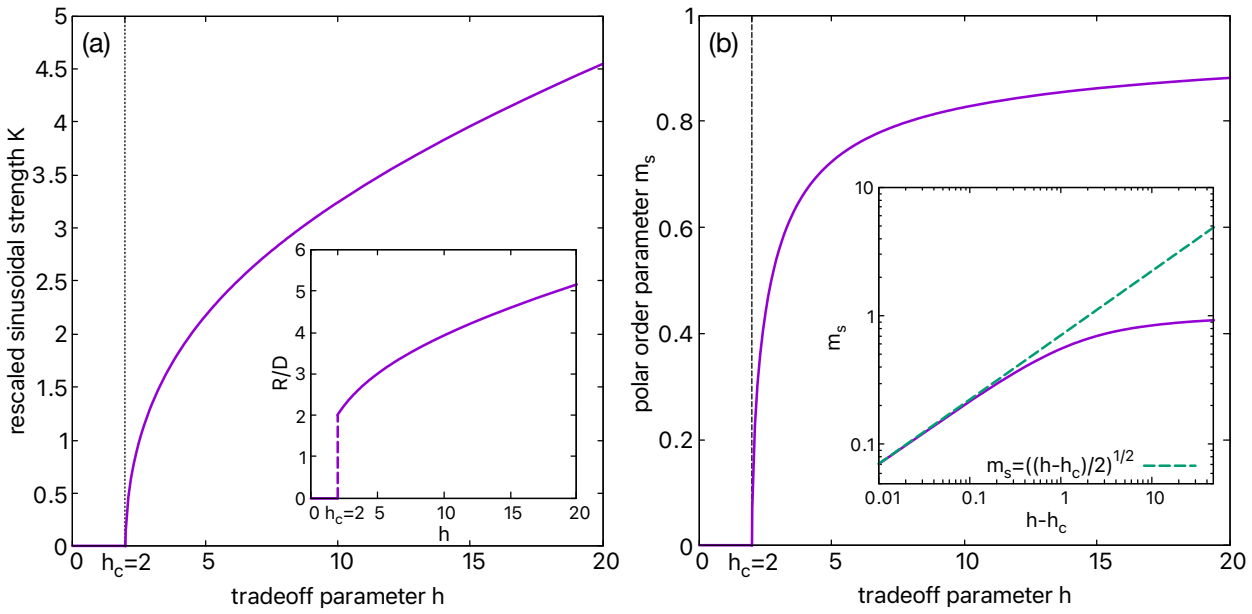

Figure 6.3: (a) Rescaled best sinusoidal control strength $K_\star$ as a function of the tradeoff parameter h. It is non-zero only for $h > h_c = 2$, where a second order transition takes place. The inset shows the control strength $R_\star(h) = K_\star(h)/(m_s D)$ of the associated Vicsek mean-field model, see Eq. (6.42). (b) polar order parameter m_s as a function of h for the best sinusoidal control. A second order phase transition is visible at $h = h_c = 2$ with critical exponent $1/2$. The inset shows the asymptotic approximations (6.49) (green dashed line) valid close to the critical point. Notice the similarity with fig. 6.2b.

Figure 6.3b displays the behavior of the polar order parameter $m_s = m_s(h)$ as a function of the tradeoff parameter h. Like in optimal case, the polar order parameter displays a second order phase transition at the same critical point $h = h_c = 2$ and with the same critical exponent $1/2$. To study the critical regime analytically , we follow the same logic as for the optimal model. The main difference is that, instead of expanding for small q, we expand for small K. In the end, the procedures appear to be equivalent as they are both expansion in $\sqrt{h - h_c}$. First, we can expand (6.43) and get

$$\rho_s = \frac{1}{2\,\pi} \left(1 + K \cos\theta + \frac{K^2}{4}(2\,\cos^2\theta - 1) + \frac{K^3}{12}(2\,\cos^3\theta - 3\cos\theta) + \frac{K^4}{192}(9 - 24\,\cos^2\theta + 8\,\cos^4\theta) \right) + o(K^4) \tag{6.46}$$

Then, by using Eqs. (6.44) and (6.46), the average total cost (6.45) can be written explicitly as

$$\bar{C}_s - C_0 = D^2 \left[\left(\frac{1}{4} - \frac{h}{8} \right) K^2 + \frac{h-1}{32}\, K^4 \right] + o(K^4) \tag{6.47}$$

A positive $K = K_\star$ which minimizes the previous expression exists only for $h > h_c = 2$ and its asymptotic expression near h_c is

$$K_\star \sim \sqrt{2\,(h-2)}\,, \tag{6.48}$$

which plugged into Eq. (6.44) yields (see also inset of fig. 6.3b)

$$m \sim \sqrt{\frac{1}{2}(h-h_c)}\,. \tag{6.49}$$

It follows from Eqs. (6.48) and (6.47) that

$$\bar{C}_s - C_0 = -\frac{D^2}{8}\,(h-h_c)^2 + o((h-h_c)^2)\,. \tag{6.50}$$

Note that the sinusoidal control (6.41), near the critical point, features a rescaled strength $K_\star \approx h_c m_s$ which is similar to the optimal one (6.28). However, m and m_s are not fully equivalent. The previous expressions also imply that the best coupling $R_\star$ (Eq. (6.42)) from the Vicsek interpretation displays a first order discontinuity, as shown in the inset of fig. 6.3a.

In spite of the similarities, the asymptotic dependence of m_s on h (for $h \to h_c^+$) differs by a pre-factor with respect to the optimal control: indeed comparing Eqs. (6.49) and (6.26) we have different prefactors $\sqrt{1/2}$ and $\sqrt{4/7}$, respectively which differ by a mere 6%. At a first glance this difference may seem surprising, but it can actually be rationalized by observing that the sinusoidal control is just a first order approximation to the optimal one while, as detailed below, the polar order parameter prefactor at criticality is determined by the first sub-leading order.

For the optimal control, we have derived the optimal critical behavior by expanding the self-consistency condition (6.18) $\mathcal{F}(m) = m$. The latter can be rewritten as $m = \langle \cos\theta \rangle_m$, which plays the same role as Eq. (6.44) for the sinusoidal model; $\langle [\ldots] \rangle_m$ is to remind that the probability density depends on the polarization itself as typical in self-consistent problems. As the control is odd w.r.t. the transformation $\bar{\theta} \mapsto \bar{\theta} + \pi$, it turns out that $\langle \cos\theta \rangle_m$ is odd in m. Consequently, close to the critical point (i.e. form small m) we can write $\langle \cos\theta \rangle_m = c_1\, hm + c_k\,(h\,m)^k + o((hm)^k)$ with $k > 1$ being an odd integer, which we know to be 3 (see e.g. Eq. (6.25)). Now, imposing the self-consistency condition $m = \langle \cos\theta \rangle_m$ yields

$$m \approx \left[\frac{1}{h^k\, c_k}\,(1 - c_1\, h)\right]^{\frac{1}{k-1}}\,, \tag{6.51}$$

from which we deduce that the critical point $h_c = c_1^{-1}$ is solely determined by the first order and is, therefore, a leading order effect; the critical exponent is fixed by the value of k (ordinal number of the first non-vanishing sub-leading order) as $1/(k-1)$ (1/2 in our case as $k = 3$) and the prefactor is given by the sub-leading order prefactor $-c_1/(c_k\, h^3)$. Since the optimal solution is sinusoidal at leading order, we expect $h_c = 2$ to hold for both the optimal and best sinusoidal controls, by construction. The value of $k = 3$ and, therefore, the critical exponent 1/2, is fixed by the symmetry and should be the same for both models. However, any further equivalence is not obvious and, in particular, there is no specific reason for c_3 to be the same: they are actually different and this explains the difference in the prefactors discussed above.

Now we briefly discuss the strong coupling regime. For large h (which implies large K), the

Von-Mises distribution (6.43) is well approximated by the Gaussian

$$\rho_K = \sqrt{\frac{K}{2\pi}}\, \exp\left(-\frac{K}{2}\theta^2\right) , \tag{6.52}$$

and $\cos\theta \approx 1 - \theta^2/2$. With such an approximation the self-consistency condition (6.44) yields $m_s = 1 - \frac{1}{2K} + o(1/K)$, which inserted into the cost (6.45) and minimizing with respect to K gives

$$K_\star = \sqrt{h} + o(\sqrt{h})\,. \tag{6.53}$$

Consequently, the polar order parameter in the large h limit is given by $m_s \approx 1 - 1/2\sqrt{h}$ as for optimal case (6.39).

The above discussion establishes a connection between sinusoidal and optimal model in the asymptotic regimes, but provides little insights into the intermediate region. In the next section, we further investigate the differences and similarities between optimal and best sinusoidal control.

6.2.2 Comparison between optimal solution and best sinusoidal model

Critical case comparison For both models, below the critical point, $h < h_c$, no control is exerted as it is too expensive. Consequently, the stationary distribution of agents orientation is uniform in both cases. Near the critical point, $h - h_c = 0^+$, the angular distribution remains approximately uniform and collision costs remain high, since collisions of anti-aligned particles are common. However, in this regime, tiny deviations of the optimal control from the sinusoidal one (see fig. 6.4a) allow to slightly squeeze the distribution towards the alignment with the mean direction $\bar{\theta} = 0$ in fig. 6.4b and 6.5a. Therefore, the optimal model pays slightly more in the control cost to achieve a reduction of the collision cost w.r.t. the sinusoidal one, overall reducing the average cost, as shown in fig. 6.6. In particular, close to the critical point, the cost difference between the

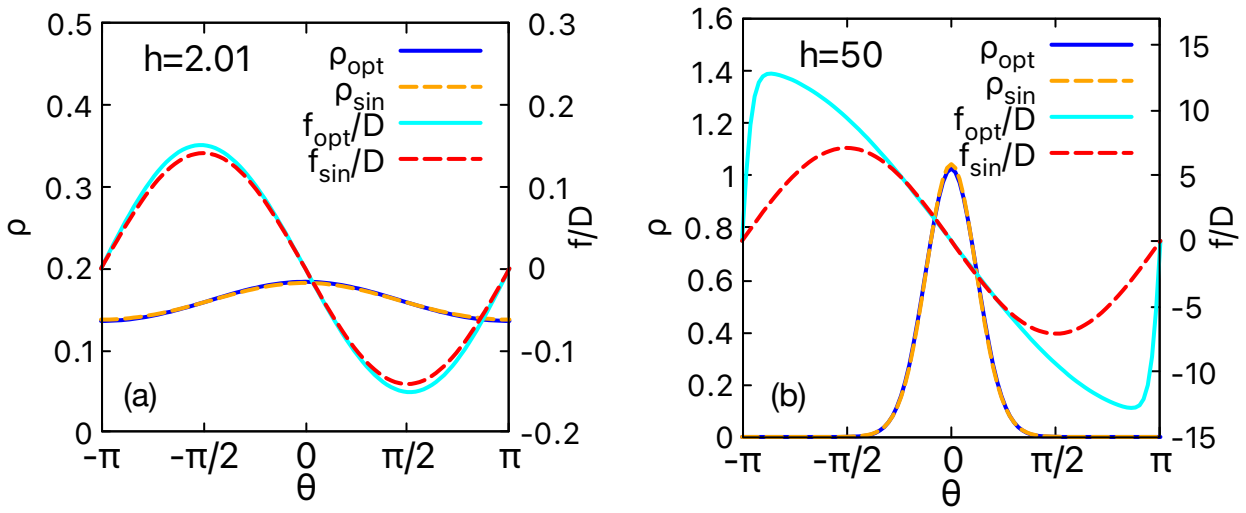

Figure 6.4: Comparison for the optimal and best sinusoidal model between the angular probability density function (left scale) and the control f/D (right scale) for (a) $h = 2.1$, i.e. close to the critical point, and (b) $h = 50$, corresponding to the strong coupling regime. See the figure legend for the various curves.

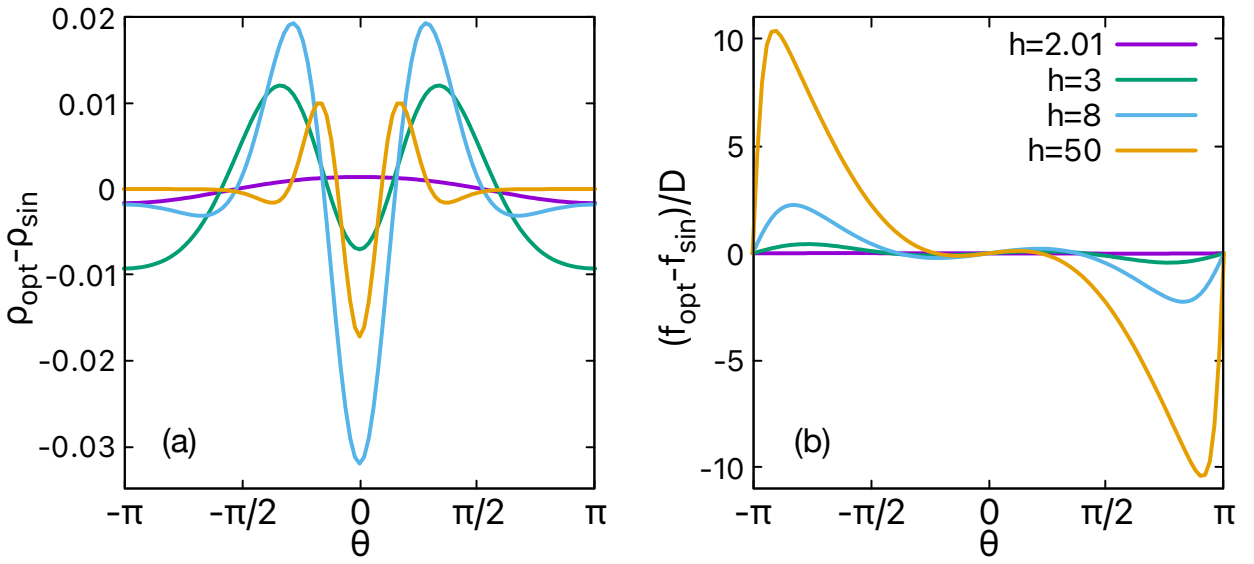

Figure 6.5: Differences between the optimal and best sinusoidal model: (a) angular probability density and (b) rescaled control, $(f(h) - D\,K_\star(h)\,\sin(\theta))/D$, for different h as in figure legend.

optimal and sinusoidal model is

$$\Delta\bar{C} = \bar{C} - \bar{C}_s = -\frac{D^2}{56}\,(h - h_c)^2 + o((h - h_c)^2)\,, \tag{6.54}$$

as obtained by subtracting Eq. (6.50) from Eq. (6.29).

Intermediate and strongly interacting regimes After having discussed the critical region we now move to a comparison between the two models in the intermediate and strong coupling regime. As h grows away from the critical point, both distributions shrink towards the origin, but

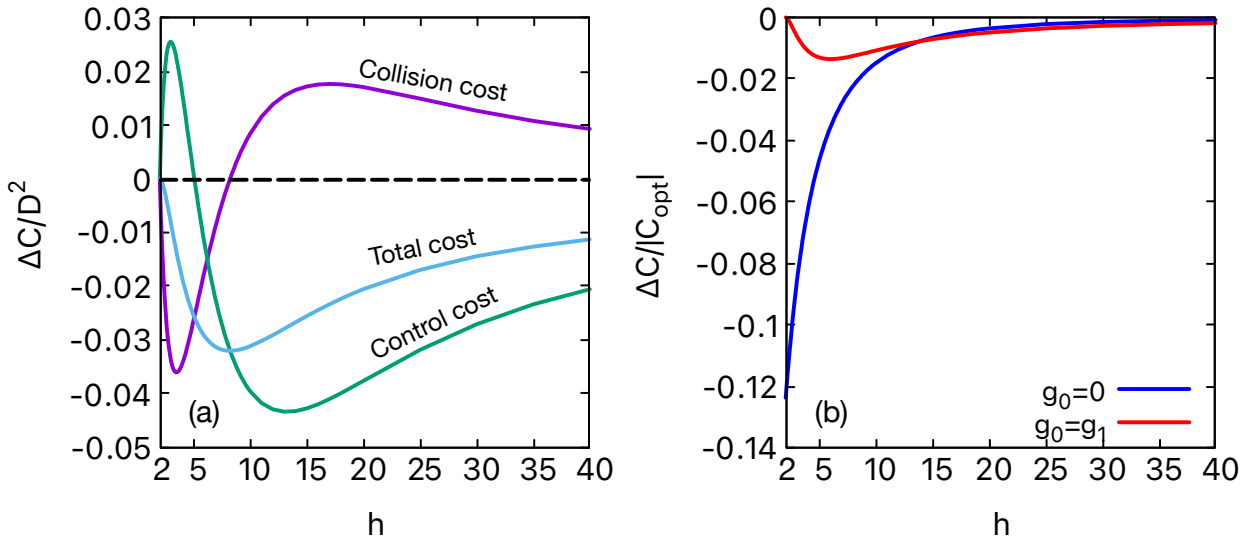

Figure 6.6: Cost comparison between optimal and best sinusoidal model: (a) rescaled average cost differences $\Delta\bar{C} = (\bar{C} - \bar{C}_s)/D^2$ - total (cyan curve), collision (purple) and control (green)- plotted as universal functions of the tradeoff parameter h; (a) the relative cost difference $\Delta\bar{C}/|\bar{C}|$, as a function of the tradeoff parameter h in two notable cases $g_0 = 0$ (pure alignment) and $g_0 = g_1$ (pure collision).

not exactly in the same way. The best sinusoidal model distribution is more peaked both around the origin (strong alignment) and around $\pm\pi$ (strong anti-alignment), while intermediate values are less probable. This difference, highlighted in fig. 6.5a, is the largest around $h = \hat{h} \approx 8$, which also corresponds to the the region where the difference between the total costs $\Delta\bar{C}$ is the largest, as shown in fig. 6.6a. Just before $\hat{h}$, the optimal model outperforms the sinusoidal one in both control and collision costs. However, the collision cost advantage rapidly declines and, for large h, the edge of the optimal solution is preferred only due to lower control costs, while the collision cost is higher. To understand the origin of these differences, we should look at the shape of the optimal control, which starts sinusoidal at $h - h_c = 0^+$ and then approaches a sawtooth shape for $h \gg 1$ (see fig. 6.5b and 6.4b). The strong control near $\pm\pi$ makes little difference in terms of costs, because the probability of exploring such region is exponentially suppressed. Indeed, both optimal and best sinusoidal distributions converge to the same Gaussian distribution with vanishing variance for large h (see fig. 6.4b). In other terms, the two seemingly different controls (fig. 6.5b) only contribute with their linear approximation near the origin and are therefore equivalent, consistently with derivation of the previous section. We now move to a discussion of the cost difference.

We have provided a detailed description of the differences between the two models. We should remark that all discrepancies we have highlighted are somehow small, since the two distributions never show significantly different shapes. Moreover, at the critical point, the critical exponents are the same and, finally, the polar order parameters are closely related for all h. The most delicate point is the cost difference: the universal behavior of the average cost difference as a function of h (fig. 6.6a) only emerges when rescaling with such difference with the factor D^2. In other terms, such difference can be made arbitrarily large as can be deduced, for instance, by rescaling $g \mapsto z\, g$ and $D \mapsto D\sqrt{z}$. Under this transformation, while h does not change, the cost difference does, as $\Delta\bar{C} \mapsto z\, \Delta\bar{C}$.

The above observation implies that a sound analysis should take into account the relative difference $\Delta\bar{C}/\bar{C}$. Note, though, that such quantity depends on the constant C_0 and thus on g_0 (which otherwise play no role in the optimization process). However, as discussed g_0 relates to the interpretation of the model, thus it is not possible to give a universal description. We can briefly consider some notable cases: $g_0 = 0$ and $g_0 = g_1$ which represent a pure alignment and collision problem, respectively. In the former case, $\Delta\bar{C}/|\bar{C}|$ is 0 for $h < h_c$ and then (first order discontinuity) jumps to $-1/8$ at $h = h_c$ (see fig. 6.6b) and finally vanishes to zero for $h \to \infty$. Note, however, that the maximal relative cost difference at $h = h_c^+$ is obtained in a limit in which both $\Delta\bar{C}$ and, mostly important, $\bar{C}$ itself vanishes (see Eqs. (6.54) and (6.29) with $C_0 = 0$). On the other hand, in the $g_0 = g_1$ case, we have that $\bar{C} = C_0 \neq 0$ at $h = h_c$ meaning that the even the cost of isotropic random collisions is not zero. Consequently, since the denominator never vanishes, we have $\Delta\bar{C}/|\bar{C}| \to 0$ for $h \to h_c^+$. Similarly, for $h \to \infty$, it is easy to see that $\Delta\bar{C}/|\bar{C}| \to 0$, as the two models are equivalent at leading order. Therefore, the maximal relative difference is realized for some $h \in (h_c, \infty)$: a numerical test shown in fig. 6.6b reveals that this is realized at $h \approx 6$ with $\Delta\bar{C}/|\bar{C}| \approx -0.0135$. By considering both the rescaled and the relative cost difference analysis, we can conclude that the sinusoidal model is always a good approximations for the optimal solution in realistic scenarios.

6.3 Conclusions and perspectives

We have shown by means of optimal control techniques, that the optimal behavior of active particles for collision-avoiding active particles can be characterized by a tradeoff parameter h between collision and control costs, and that the polarization of the system undergoes a phase transition in the mean-field regime. The possibility of approaching this problem analytically, albeit in an approximate form, provided insights into the features of the optimal solution and a comprehensive statistical characterization. Moreover, we found that the optimal behavior, both close to the transition and for large tradeoff values, is well approximated by a mean-field version of the kinetic Vicsek model [60, 62, 201] which also displays a second order transition. Therefore, such model, whose short range version was mainly derived from from phenomenological considerations, turns out to be a quasi-optimal solution for the collision-avoidance task, upon choosing the appropriate parameters. Clearly, when working with task-oriented agents, as in biological systems or robotics, being close to optimality is a highly desirable feature. The OC framework is therefore a very valuable tool not just for discovering new optimal models, but also for assessing the quality of existing ones with respect to some performance criteria. Moreover, in accordance with previous studies, e.g. ref. [203] where known chemotactic behaviors were found to be optimal solutions to target search problem, our findings suggest that OC formalism can, at least in some cases, provide a theoretical ground to interpret some biological solutions in situations where specific tasks need to be solved.

While our analysis was restricted to a single scenario, the same approach could be successfully carried to different settings to explore other classes of collective behaviors, for instance allowing for linear acceleration or for particles of finite sizes. Also, remaining in the context of mean-field Vicsek-like models, it would be interesting to explore how different kinds of noise can influence the optimal solution. Another interesting outlook would be to go beyond the mean-field approximation, either analytically or numerically, by introducing a spatial structure.

Appendix 6.A Mathieu functions

Mathieu functions are solutions of eigenvalue Schrödinger equation for the quantum pendulum, which is customarily written as

$$\zeta'' + (a - 2\,q\,\cos(2\,y))\,\zeta = 0\,, \tag{6.55}$$

with $y \in [-\pi/2, \pi/2]$, and where q is some constant and a is (minus) the eigenvalue (the "energy"). For any q, the eigenvalues $a_k(q)$ depend on a parameter k and are called Mathieu characteristic function. By imposing boundary conditions, $k > 0$ becomes and integer representing the ordinal number of the energy level: if $q < 0$ then $-a_k(q) < -a_{k+1}(q)$. The eigenfunctions m_k are either even $\mathit{ce}_k = \mathit{m}_{2k}$ or odd $\mathit{se}_k = \mathit{m}_{2k+1}$. Since we are only interested in the ground state with periodic boundary conditions, we focus on $k = 0$. The eigenvalue is $a(q) := a_0(q)$ (main text notation) and the eigenfunction $\zeta = \mathit{ce}_0(a(q), q)$. For further details, see for instance refs. [41, 97, 102].

$(-\infty, \infty)$ by enforcing

Appendix 6.B Proofs of the expressions for the susceptivity

Consider an external field as defined in Eq. (6.30). The self consistency condition (6.18) implicitly defines the polar oder parameter $m = m(h, \epsilon)$ as

$$\mathcal{F}(m(h,\epsilon)) = m(h,\epsilon). \tag{6.56}$$

Upon defining $\tilde{\mathcal{F}} = m - \mathcal{F}$, Dini implicit function theorem allows to compute the susceptivity as

$$\chi(h) = \left.\frac{\partial m}{\partial \epsilon}\right|_{\epsilon=0} = -\frac{\left.\frac{\partial \tilde{\mathcal{F}}}{\partial \epsilon}\right|_{\epsilon=0}}{\left.\frac{\partial \tilde{\mathcal{F}}}{\partial m}\right|_{\epsilon=0}}. \tag{6.57}$$

though, in some cases, there is a shorter procedure. We consider the following four scenarios.

- $h < h_c$. In this case, we can assume that, unless there is some discontinuity, $\lim_{\epsilon\to 0} m(h,\epsilon) = 0$. Hence, assuming both ϵ and m small, Eq. (6.56) can be expanded as

$$-\frac{(h-2)\,m + 2\,\epsilon/D^2}{2} + \frac{7\,(m\,h + 2\,\epsilon/D^2)^3}{64} + o((m\,h + 2\,\epsilon/D^2)^3) = 0\,. \tag{6.58}$$

  Then we can either use Dini's theorem or simply observe that $\mathcal{F} = 0$ at leading order implies $m = 2\epsilon/[D^2\,(2-h)]$ and, therefore

$$\chi = \frac{2}{D^2\,(h_c - h)}\,, \tag{6.59}$$

  holds exactly in this region.

- $h = h_c$. We can again use the ansatz $\lim_{\epsilon\to 0} m(h,\epsilon) = 0$ along with Eq. (6.58). Then, either applying Dini's theorem, or observing that, since $h-2$ is zero exactly, leading order in ϵ must match leading order in m, which is m^3. Hence, we immediately get

$$m \sim 2\,\left[\frac{\epsilon}{7\,D^2}\right]^{1/3}\,, \tag{6.60}$$

  thus, at the critical point, $m(\epsilon)$ in continuous in ϵ but non differentiable.

- $h = h_c^+$. As in the critical regime, it remains valid that the nudge selects the direction in which the symmetry is broken since other choices would be sub-optimal. Here $\lim_{\epsilon\to 0} m(h,\epsilon) = m(h) > 0$, however, as long as h is close to h_c, we can still use expansion (6.58). From Eq. (6.26) we can write $m(h) = \sqrt{(4/7)\,(h-2)} + \delta m$, with $\delta m \ll 1$ and match leading order of δm and ϵ. As a result, we get

$$\chi \sim \frac{1}{D^2\,(h_c - h)}\,. \tag{6.61}$$

- $h \gg h_c$. Using Eq. (6.38), the self consistency equation (6.18) becomes

$$m - e^{-\frac{1}{2\sqrt{h\,m + 2\,\epsilon/D^2}}} = 0\,, \tag{6.62}$$

and a straightforward application of Dini's theorem yields

$$\chi \sim \frac{1}{2\,D^2\,h^{3/2}\,m^{1/2}}\,. \tag{6.63}$$

Appendix 6.C Modified Bessel function of the first kind

Modified Bessel functions of the first kind are non decreasing solution of the modified Bessel equation:

$$z^2\,I_n'' + z\,I_n' - (z^2 + n^2)\,I_n = 0 \tag{6.64}$$

We report a few identities we have used in the main text

$$I_0(z) = \int_{-\pi}^{\pi} d\theta\, e^{z\cos\theta} \tag{6.65}$$

$$I_0'(z) = I_1(z) = \int_{-\pi}^{\pi} d\theta\, \cos\theta\, e^{z\cos\theta} \tag{6.66}$$

$$I_0''(z) = I_0(z) - \frac{1}{z}\,I_1(z) = \int_{-\pi}^{\pi} d\theta\, \cos^2\theta\, e^{z\cos\theta} \tag{6.67}$$

For further details, see [97].

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, and R. Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Rev. Mod. Phys*, 77(1):137, 2005.

[3] G. Adam and M. Delbrück. Reduction of dimensionality in biological diffusion processes. *Nat. Struct. Mol. Biol.*, 198:198–215, 1968.

[4] J. K. Alageshan, A. K. Verma, J. Bec, and R. Pandit. Machine learning strategies for path-planning microswimmers in turbulent flows. *Phys. Rev. E*, 101(4):043110, 2020.

[5] K. Albertsson, P. Altoe, D. Anderson, M. Andrews, J. P. A. Espinosa, A. Aurisano, L. Basara, A. Bevan, W. Bhimji, D. Bonacorsi, et al. Machine learning in high energy physics community white paper. In *J. Phys. Conf. Ser.*, volume 1085, page 022008. IOP Publishing, 2018.

[6] S.-I. Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, 1998.

[7] C. Anderson. The end of theory: The data deluge makes the scientific method obsolete. *Wired magazine*, 16(7):16–07, 2008.

[8] S. Anna, P. Ashwin, C. D. Camp, M. Crucifix, H. A. Dijkstra, P. Ditlevsen, and T. M. Lenton. Quantification and interpretation of the climate variability record. *Global and Planetary Change*, page 103399, 2020.

[9] H. Arbabi and T. Sapsis. Data-driven modeling of strongly nonlinear chaotic systems with non-gaussian statistics. *arXiv preprint arXiv:1908.08941*, 2019.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Sys.*, 57(5):469–483, 2009.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Proces. Mag.*, 34(6):26–38, 2017.

[12] A. Attanasi, A. Cavagna, L. Del Castello, I. Giardina, T. S. Grigera, A. Jelić, S. Melillo, L. Parisi, O. Pohl, E. Shen, and M. Viale. Information transfer and behavioural inertia in starling flocks. *Nature Phys.*, 10(9):691–696, 2014.

[13] E. Aurell, G. Boffetta, A. Crisanti, G. Paladin, and A. Vulpiani. Growth of noninfinitesimal perturbations in turbulence. *Phys. Rev. Lett.*, 77(7):1262, 1996.

[14] E. Aurell, G. Boffetta, A. Crisanti, G. Paladin, and A. Vulpiani. Predictability in systems with many characteristic times: The case of turbulence. *Phys. Rev. E*, 53(3):2337, 1996.

[15] K. Azizzadenesheli, Y. Yue, and A. Anandkumar. Policy gradient in partially observable environments: Approximation and convergence. *arXiv preprint arXiv:1810.07900*, 2018.

[16] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019.

[17] C. Baldassi, E. M. Malatesta, M. Negri, and R. Zecchina. Wide flat minima and optimal generalization in classifying high-dimensional gaussian mixtures. *J. Stat. Mech. Theory Exp.*, 2020(12):124012, 2020.

[18] P. Baldi and P. J. Sadowski. Understanding dropout. *Adv. Neural. Inf. Process. Syst.*, 26:2814–2822, 2013.

[19] M. Baldovin, F. Cecconi, M. Cencini, A. Puglisi, and A. Vulpiani. The role of data in model building and prediction: a survey through examples. *Entropy*, 20(10):807, 2018.

[20] M. Baldovin, F. Cecconi, and A. Vulpiani. Effective equations for reaction coordinates in polymer transport. *J. Stat. Mech. Theory Exp.*, 2020(1):013208, 2020.

[21] M. Baldovin, A. Puglisi, and A. Vulpiani. Langevin equations from experimental data: The case of rotational diffusion in granular media. *PloS one*, 14(2):e0212135, 2019.

[22] C. Bechinger, R. Di Leonardo, H. Löwen, C. Reichhardt, G. Volpe, and G. Volpe. Active particles in complex and crowded environments. *Rev. Mod. Phys.*, 88(4):045006, 2016.

[23] G. Beintema, A. Corbetta, L. Biferale, and F. Toschi. Controlling Rayleigh-Bénard convection via reinforcement learning. *arXiv preprint arXiv:2003.14358*, 2020.

[24] F. Belkhouche, B. Belkhouche, and P. Rastgoufard. Parallel navigation for reaching a moving goal by a mobile robot. *Robotica*, 25(1):63–74, 2007.

[25] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory. *Meccanica*, 15(1):9–20, 1980.

[26] D. P. Bertsekas. Dynamic programming and optimal control 3rd edition, volume ii. *Belmont, MA: Athena Scientific*, 2011.

[27] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[28] W. Bialek, A. Cavagna, I. Giardina, T. Mora, E. Silvestri, M. Viale, and A. M. Walczak. Statistical mechanics for natural flocks of birds. *Proc. Natl. Acad. Sci. U.S.A.*, 109(13):4786–4791, 2012.

[29] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson. Zermelo's problem: Optimal point-to-point navigation in 2d turbulent flows using reinforcement learning. *Chaos*, 29(10):103138, 2019.

[30] L. Biferale, A. A. Mailybaev, and G. Parisi. Optimal subgrid scheme for shell models of turbulence. *Phys. Rev. E*, 95(4):043108, 2017.

[31] Å. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[32] H. Bleckmann and R. Zelick. Lateral line system of fish. *Integr. Zool.*, 4(1):13, 2009.

[33] G. Boffetta, A. Celani, M. Cencini, G. Lacorata, and A. Vulpiani. The predictability problem in systems with an uncertainty in the evolution law. *J. Phys. A Math. Theor.*, 33(7):1313, 2000.

[34] G. Boffetta, P. Giuliani, G. Paladin, and A. Vulpiani. An extension of the Lyapunov analysis for the predictability problem. *J. Atmos. Sci.*, 55(23):3409–3416, 1998.

[35] T. Bohr, M. H. Jensen, G. Paladin, and A. Vulpiani. *Dynamical systems approach to turbulence*. Cambridge University Press, 2005.

[36] F. Borra and M. Baldovin. Using machine-learning modeling to understand macroscopic dynamics in a system of coupled maps. *Chaos*, 31(2):023102, 2021.

[37] F. Borra, L. Biferale, M. Cencini, and A. Celani. Reinforcement learning for pursuit and evasion of microswimmers at low reynolds number. *arXiv preprint arXiv:2106.08609*, 2021.

[38] F. Borra, M. Cencini, and A. Celani. Optimal collision avoidance in swarms of active brownian particles. 2021(8):083401, aug 2021.

[39] F. Borra, A. Vulpiani, and M. Cencini. Effective models and predictability of chaotic multiscale systems via machine learning. *Phys. Rev. E*, 102(5):052203, 2020.

[40] S. Boyd and L. Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on circuits and systems*, 32(11):1150–1161, 1985.

[41] C. Brimacombe, R. M. Corless, and M. Zamir. Computation and applications of Mathieu functions: A historical perspective. *arXiv preprint arXiv:2008.01812*, 2020.

[42] D. B. Brückner, P. Ronceray, and C. P. Broedersz. Inferring the dynamics of underdamped stochastic systems. *Phys. Rev. Lett.*, 125(5):058103, 2020.

[43] D. Brunner, M. C. Soriano, and G. Van der Sande. Photonic reservoir computing. *De Gruyter*, 8:19, 2019.

[44] M. Buzzicotti, F. Bonaccorso, P. C. Di Leoni, and L. Biferale. Reconstruction of turbulent data with deep generative models for semantic inpainting from TURB-Rot database. *arXiv preprint arXiv:2006.09179*, 2020.

[45] L. Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D*, 110(1-2):43–50, 1997.

[46] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91(4):045002, 2019.

[47] M. Carlu, F. Ginelli, V. Lucarini, and A. Politi. Lyapunov analysis of multiscale dynamics: The slow manifold of the two-scale Lorenz'96 model. *arXiv preprint arXiv:1809.05065*, 2018.

[48] Z. Carmichael, H. Syed, and D. Kudithipudi. Analysis of wide and deep echo state networks for multiscale spatiotemporal time series forecasting. In *Proc. 7th Annual Neuro-inspired Comput. Elements Workshop*, pages 1–10, 2019.

[49] A. Cavagna, D. Conti, C. Creato, L. Del Castello, I. Giardina, T. S. Grigera, S. Melillo, L. Parisi, and M. Viale. Dynamic scaling in natural swarms. *Nature Phys.*, 13(9):914–918, 2017.

[50] A. Cavagna, I. Giardina, and T. S. Grigera. The physics of flocking: Correlation as a compass from experiments to theory. *Phys. Rep.*, 728:1–62, 2018.

[51] M. Cencini, F. Cecconi, and A. Vulpiani. *Chaos: from simple models to complex systems.* World Scientific, 2010.

[52] M. Cencini, M. Falcioni, E. Olbrich, H. Kantz, and A. Vulpiani. Chaos or noise: Difficulties of a distinction. *Phys. Rev. E*, 62(1):427, 2000.

[53] M. Cencini, M. Falcioni, D. Vergni, and A. Vulpiani. Macroscopic chaos in globally coupled maps. *Physica D*, 130(1):58 – 72, 1999.

[54] M. Cencini, M. Franchino, F. Santamaria, and G. Boffetta. Centripetal focusing of gyrotactic phytoplankton. *J. Theor. Biol.*, 399:62–70, 2016.

[55] M. Cencini and A. Vulpiani. Finite size lyapunov exponent: review on applications. *J. Phys. A Math. Theor.*, 46(25):254019, 2013.

[56] A. Chattopadhyay, P. Hassanzadeh, K. V. Palem, and D. Subramanian. Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ann, and RNN-LSTM. *CoRR*, abs/1906.08829, 2019.

[57] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *J. Stat. Mech. Theory Exp.*, 2019(12):124018, 2019.

[58] B. Chen, S. Song, H. Lipson, and C. Vondrick. Visual hide and seek. In *Artificial Life Conference Proceedings*, pages 645–655. MIT Press, 2020.

[59] J. Cheng, A. N. Tegge, and P. Baldi. Machine learning methods for protein structure prediction. *IEEE reviews in biomedical engineering*, 1:41–49, 2008.

[60] A. Chepizhko and V. Kulinskii. The kinetic regime of the Vicsek model. In *AIP Conf. Proc.*, volume 1198, pages 25–33, 2009.

[61] A. Chepizhko and V. Kulinskii. On the relation between Vicsek and Kuramoto models of spontaneous synchronization. *Physica A*, 389(23):5347–5352, 2010.

[62] O. Chepizhko, D. Saintillan, and F. Peruani. Revisiting the emergence of order in active matter. *Soft Matter*, 17(11):3113–3120, 2021.

[63] C. Chiu, P. V. Reddy, W. Xian, P. S. Krishnaprasad, and C. F. Moss. Effects of competitive prey capture on flight behavior and sonar beam pattern in paired big brown bats, eptesicus fuscus. *J. Exper. Biol.*, 213(19):3348, 2010.

[64] F. Cichos, K. Gustavsson, B. Mehlig, and G. Volpe. Machine learning for active matter. *Nature Mach. Intel.*, 2(2):94, 2020.

[65] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[66] S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale. Flow navigation by smart microswimmers via reinforcement learning. *Phys. Rev. Lett.*, 118(15):158004, 2017.

[67] A. Corbetta, V. Menkovski, R. Benzi, and F. Toschi. Deep learning velocity signals allows to quantify turbulence intensity. *arXiv preprint arXiv:1911.05718*, 2019.

[68] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[69] A. Crisanti, M. Falcioni, and A. Vulpiani. Broken ergodicity and glassy behavior in a deterministic chaotic map. *Phys. Rev. Lett.*, 76(4):612, 1996.

[70] A. Daddi-Moussa-Ider, B. Nasouri, A. Vilfan, and R. Golestanian. Optimal swimmer can be puller, pusher, or neutral depending on the shape. *arXiv preprint arXiv:2104.13607*, 2021.

[71] M. Doall, J. Strickler, D. Fields, and J. Yen. Mapping the free-swimming attack volume of a planktonic copepod, euchaeta rimana. *Mar. Biol.*, 140(4):871, 2002.

[72] M. R. Dranias, H. Ju, E. Rajaram, and A. M. VanDongen. Short-term memory in networks of dissociated cortical neurons. *J. Neurosci.*, 33(5):1940–1953, 2013.

[73] K.-L. Du and M. N. Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013.

[74] M. Durve, F. Peruani, and A. Celani. Learning to flock through reinforcement. *Phys. Rev. E*, 102(1):012601, 2020.

[75] K. Dvijotham and E. Todorov. A unified theory of linearly solvable optimal control. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, volume 1, pages 25–34, 2011.

[76] R. C. Eaton, R. K. K. Lee, and M. B. Foreman. The mauthner cell and other identified neurons of the brainstem escape network of fish. *Progr. Neurobiol.*, 63(4):467–485, 2001.

[77] J. Elgeti, R. G. Winkler, and G. Gompper. Physics of microswimmers—single particle motion and collective behavior: a review. *Rep. Prog. Phys.*, 78(5):056601, 2015.

[78] G. L. Eyink and D. Bandak. A renormalization group approach to spontaneous stochasticity. *arXiv preprint arXiv:2007.01333*, 2020.

[79] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[80] M. Falcioni, L. Palatella, S. Pigolotti, and A. Vulpiani. Properties making a chaotic system a good pseudo random number generator. *Phys. Rev. E*, 72(1):016220, 2005.

[81] F. D. C. Farrell, M. C. Marchetti, D. Marenduzzo, and J. Tailleur. Pattern formation in self-propelled particles with density-dependent motility. *Phys. Rev. Lett.*, 108(24):248101, 2012.

[82] R. Feng, J. Chen, E. Fernandes, S. Jha, and A. Prakash. Robust physical hard-label attacks on deep learning visual classification. *arXiv preprint arXiv:2002.07088*, 2020.

[83] F. Ferretti, V. Chardès, T. Mora, A. M. Walczak, and I. Giardina. Building general Langevin models from discrete datasets. *Phys. Rev. X*, 10(3):031018, 2020.

[84] B. A. Free, J. Lee, and D. A. Paley. Bioinspired pursuit with a swimming robot using feedback control of an internal rotor. *Bioinsp. Biomim.*, 15(3):035005, 2020.

[85] R. Friedrich, J. Peinke, M. Sahimi, and M. R. R. Tabar. Approaching complexity by stochastic methods: From biological systems to turbulence. *Physics Reports*, 506(5):87–162, 2011.

[86] U. Frisch. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[87] G. Froyland, T. Hüls, G. P. Morriss, and T. M. Watson. Computing covariant Lyapunov vectors, Oseledets vectors, and dichotomy projectors: A comparative numerical study. *Physica D*, 247(1):18–39, 2013.

[88] K. Fujii and K. Nakajima. Harnessing disordered-ensemble quantum dynamics for machine learning. *Phys. Rev. Appl.*, 8(2):024030, 2017.

[89] C. Gallicchio. Chasing the echo state property. *arXiv preprint arXiv:1811.10892*, 2018.

[90] C. Gallicchio, A. Micheli, and L. Silvestri. Local lyapunov exponents of deep echo state networks. *Neurocomputing*, 02 2018.

[91] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[92] S. Ghosh, T. Krisnanda, T. Paterek, and T. C. Liew. Realising and compressing quantum circuits with quantum reservoir computing. *Comm. Phys.*, 4(1):1–7, 2021.

[93] T. E. Gibbons. Unifying quality metrics for reservoir networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2010.

[94] F. Ginelli. The physics of the Vicsek model. *Eur. Phys. J. Spec. Top.*, 225(11):2099–2117, 2016.

[95] F. Ginelli, H. Chaté, R. Livi, and A. Politi. Covariant Lyapunov vectors. *J. Phys. A Math. Theor.*, 46(25):254005, 2013.

[96] L. Gonon and J.-P. Ortega. Fading memory echo state networks are universal. *Neural Netw.*, 138:10–13, 2021.

[97] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Academic press, 2014.

[98] P. Grassberger and I. Procaccia. Characterization of strange attractors. *Phys. Rev. Lett.*, 50(5):346, 1983.

[99] L. Grigoryeva and J. Ortega. Echo state networks are universal. *CoRR*, abs/1806.00797, 2018.

[100] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man. Cybern. Part C*, 42(6):1291–1307, 2012.

[101] X. X. Guo, S. Y. Xiang, Y. H. Zhang, L. Lin, A. J. Wen, and Y. Hao. High-speed neuromorphic reservoir computing based on a semiconductor nanolaser with optical feedback under electrical modulation. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(5):1–7, 2020.

[102] J. C. Gutiérrez-Vega, R. Rodrıguez-Dagnino, M. Meneses-Nava, and S. Chávez-Cerda. Mathieu functions, a visual approach. *Am. J. Phys.*, 71(3):233–242, 2003.

[103] J. Happel and H. Brenner. *Low Reynolds number hydrodynamics: with special applications to particulate media*, volume 1. Springer Science & Business Media, 2012.

[104] A. Hart, J. Hook, and J. Dawes. Embedding and approximation theorems for echo state networks. *Neural Netw.*, 128:234–247, 2020.

[105] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, and K. Zhang. The practical implementation of artificial intelligence technologies in medicine. *Nature Med.*, 25(1):30–36, 2019.

[106] R. Hegger and H. Kantz. Improved false nearest neighbor method to detect determinism in time series data. *Phys. Rev. E*, 60(4):4970, 1999.

[107] A. M. Hein, D. L. Altshuler, D. E. Cade, J. C. Liao, B. T. Martin, and G. K. Taylor. An algorithmic approach to natural behavior. *Curr. Biol.*, 30(11):R663–R675, 2020.

[108] D. Hennes, D. Morrill, S. Omidshafiei, R. Munos, J. Perolat, M. Lanctot, A. Gruslys, J.-B. Lespiau, P. Parmas, E. Duenez-Guzman, and K. Tuyls. Neural replicator dynamics. *arXiv:1906.00190 [cs.LG]*, 2019.

[109] J. A. Hertz. *Introduction to the theory of neural computation.* CRC Press, 2018.

[110] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[111] J. Hofbauer and K. Sigmund. *Evolutionary games and population dynamics.* Cambridge University Press, 1998.

[112] M.-O. Hongler. Mean-field games and swarms dynamics in gaussian and non-gaussian environments. *J. Dyn. Games*, 7(1):1, 2020.

[113] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.*, 79(8):2554–2558, 1982.

[114] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural netw.*, 2(5):359–366, 1989.

[115] R. A. Howard and J. E. Matheson. Risk-sensitive markov decision processes. *Manage. Sci.*, 18(7):356–369, 1972.

[116] J. Huke. Embedding nonlinear dynamical systems: A guide to takens' theorem. 2006.

[117] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[118] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Adv. Neural. Inf. Process. Syst.*, volume 8, page 345. Morgan Kaufmann Publishers, 1995.

[119] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. *GMD-Report 148, German National Research Institute for Computer Science*, 01 2001.

[120] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *German Nat. Res. Center Infor. Tech. GMD Technical Report*, 148(34):13, 2001.

[121] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[122] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 05 2004.

[123] H. Jaeger and H. Jaeger. Short term memory in echo state networks. gmd-report 152. In *GMD - German National Research Institute for Computer Science (2002), http://www.faculty.jacobs-university.de/hjaeger/pubs/ STMEchoStatesTechRep.pdf.*

[124] A. Jentzen and T. Welti. Overall error analysis for the training of deep neural networks via stochastic gradient descent with random initialisation. *arXiv preprint arXiv:2003.01291*, 2020.

[125] J. Jiang and Y.-C. Lai. Model-free prediction of spatiotemporal dynamical systems with recurrent neural networks: Role of network spectral radius. *Phys. Rev. Res.*, 1(3):033056, 2019.

[126] L. Jin, M. M. Gupta, and P. N. Nikiforuk. Universal approximation using dynamic recurrent neural networks: discrete-time version. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 1, pages 403–408. IEEE, 1995.

[127] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[128] K. Kaneko. Spatiotemporal intermittency in coupled map lattices. *Prog. Theor. Phys.*, 74(5):1033–1044, 1985.

[129] K. Kaneko. Chaotic but regular posi-nega switch among coded attractors by cluster-size variation. *Phys. Rev. Lett.*, 63:219–223, Jul 1989.

[130] K. Kaneko. Globally coupled chaos violates the law of large numbers but not the central-limit theorem. *Phys. Rev. Lett.*, 65:1391–1394, Sep 1990.

[131] K. Kaneko. Mean field fluctuation of a network of chaotic elements: Remaining fluctuation and correlation in the large size limit. *Physica D*, 55(3):368 – 384, 1992.

[132] K. Kaneko. Overview of coupled map lattices. *Chaos*, 2(3):279–282, 1992.

[133] K. Kaneko. Theory and applications of coupled map lattices. *Nonlinear. Anal. Theory. Methods. Appl.*, 1993.

[134] K. Kaneko. Remarks on the mean field dynamics of networks of chaotic elements. *Physica D*, 86(1):158 – 170, 1995.

[135] M. J. Kanter and S. Coombs. Rheotaxis and prey detection in uniform currents by lake michigan mottled sculpin (cottus bairdi). *J. Experm. Biol.*, 206(1):59, 2003.

[136] H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys. Rev. Lett.*, 95(20):200201, 2005.

[137] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[138] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[139] T. Kiørboe and A. W. Visser. Predator and prey perception in copepods due to hydromechanical signals. *Mar. Ecol. Progr. Ser.*, 179:81, 1999.

[140] R. Knutti and M. A. Rugenstein. Feedbacks, climate sensitivity and the limits of linear models. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 373(2054):20150146, 2015.

[141] A. G. P. Kottapalli, M. Asadnia, J. Miao, and M. Triantafyllou. Soft polymer membrane micro-sensor arrays inspired by the mechanosensory lateral line on the blind cavefish. *J. Intell. Mat. Syst. Struct.*, 26(1):38, 2015.

[142] V. Krishnamurthy. *Partially observed Markov decision processes*. Cambridge university press, 2016.

[143] M. S. Kulkarni and C. Teuscher. Memristor-based reservoir computing. In *2012 IEEE/ACM international symposium on nanoscale architectures (NANOARCH)*, pages 226–232. IEEE, 2012.

[144] Y. Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics*, pages 420–422. Springer, 1975.

[145] B. S. Lanchester and R. F. Mark. Pursuit and prediction in the tracking of moving food by a teleost fish (acanthaluteres spilomelanurus). *J. Exper. Biol.*, 63(3):627, 1975.

[146] J.-M. Lasry and P.-L. Lions. Mean field games. *Japanese J. Math.*, 2(1):229–260, 2007.

[147] E. Lauga and T. R. Powers. The hydrodynamics of swimming microorganisms. *Rep. Progr. Phys.*, 72(9):096601, 2009.

[148] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Adv. Neural. Inf. Process. Syst.*, 2, 1989.

[149] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database, 2010.

[150] H. Lei, N. A. Baker, and X. Li. Data-driven parameterization of the generalized Langevin equation. *Proc. Natl. Acad. Sci. U.S.A.*, 113(50):14183–14188, 2016.

[151] Y. Lei, J. Hu, and J. Ding. A hybrid model based on deep LSTM for predicting high-dimensional chaotic systems. *arXiv preprint arXiv:2002.00799*, 2020.

[152] M. W. Libbrecht and W. S. Noble. Machine learning applications in genetics and genomics. *Nature Rev. Genet.*, 16(6):321–332, 2015.

[153] S. H. Lim, L. Theo Giorgini, W. Moon, and J. S. Wettlaufer. Predicting critical transitions in multiscale dynamical systems using reservoir computing. *Chaos*, 30(12):123126, 2020.

[154] A. J. Linot and M. D. Graham. Deep learning to discover and predict dynamics on an inertial manifold. *Phys. Rev. E*, 101(6):062209, 2020.

[155] E. Lloyd, C. Olive, B. A. Stahl, J. B. Jaggard, P. Amaral, E. R. Duboué, and A. C. Keene. Evolutionary shift towards lateral line dependent prey capture behavior in the blind mexican cavefish. *Develop. Biol.*, 441(2):328, 2018.

[156] E. N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20(2):130–141, 1963.

[157] E. N. Lorenz. Predictability: A problem partly solved. In *ECMWF Seminar Proceedings on Predictability*, volume 1. ECMWF, Reading, UK, 1995.

[158] Z. Lu, B. Hunt, and E. Ott. Attractor reconstruction by machine learning. *Chaos*, 28, 06 2018.

[159] Z. Lu, B. R. Hunt, and E. Ott. Attractor reconstruction by machine learning. *Chaos*, 28(6):061104, 2018.

[160] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos*, 27(4):041102, 2017.

[161] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Comp. Sci. Rev.*, 3(3):127–149, 2009.

[162] M. Lukoševičius and H. Jaeger. Jaeger, h.: Reservoir computing approaches to recurrent neural network training. comput. sci. rev. 3, 127-149. *Computer Science Review*, 3:127–149, 08 2009.

[163] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[164] G. Manjunath and H. Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural computation*, 25(3):671–696, 2013.

[165] A. D. Marchese, C. D. Onal, and D. Rus. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. *Soft Robot.*, 1(1):75, 2014.

[166] K. V. Mardia and P. E. Jupp. *Directional statistics*, volume 494. John Wiley & Sons, 2009.

[167] M. J. McHenry, J. L. Johansen, A. P. Soto, B. A. Free, D. A. Paley, and J. C. Liao. The pursuit strategy of predatory bluefish (pomatomus saltatrix). *Proc. Royal Soc. B*, 286(1897):20182934, 2019.

[168] B. Mehlig. Artificial neural networks. *arXiv preprint arXiv:1901.05639*, 2019.

[169] C. Meneveau and J. Katz. Scale-invariance and turbulence models for large-eddy simulation. *Annu. Rev. Fluid Mech.*, 32(1):1–32, 2000.

[170] C. Mesaritakis, V. Papataxiarhis, and D. Syvridis. Micro ring resonators as building blocks for an all-optical high-speed reservoir-computing bit-pattern-recognition system. *JOSA B*, 30(11):3048–3055, 2013.

[171] F. Mignacco, F. Krzakala, P. Urbani, and L. Zdeborová. Dynamical mean-field theory for stochastic gradient descent in gaussian mixture classification. *arXiv preprint arXiv:2006.06098*, 2020.

[172] M. Milano and P. Koumoutsakos. Neural network modeling for near wall turbulent flow. *J. Comput. Phys.*, 182(1):1–26, 2002.

[173] M. Mirzakhanloo, S. Esmaeilzadeh, and M.-R. Alam. Active cloaking in stokes flows via reinforcement learning. *J. Fluid Mech.*, 903, 2020.

[174] G. E. Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16, 1982.

[175] J. C. Montgomery, C. F. Baker, and A. G. Carton. The lateral line can mediate rheotaxis in fish. *Nature*, 389(6654):960, 1997.

[176] T. Mora and W. Bialek. Are biological systems poised at criticality? *J. Stat. Phys.*, 144(2):268–302, 2011.

[177] S. Muiños-Landin, A. Fischer, V. Holubec, and F. Cichos. Reinforcement learning with artificial microswimmers. *Sci. Robot.*, 6(52), 2021.

[178] P. J. Nahin. *Chases and escapes: the mathematics of pursuit and evasion*. Princeton University Press, 2012.

[179] K. Nakai and Y. Saiki. Machine-learning inference of fluid variables from data using reservoir computing. *Phys. Rev. E*, 98(2):023111, 2018.

[180] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[181] H. Nishimori. *Statistical physics of spin glasses and information processing: an introduction*. Number 111. Clarendon Press, 2001.

[182] L. Noakes. The takens embedding theorem. *Int. J. Bifurcat. Chaos*, 1(04):867–872, 1991.

[183] A. Okubo. Dynamical aspects of animal grouping: swarms, schools, flocks, and herds. *Adv. Biophys.*, 22:1–94, 1986.

[184] R. M. Olberg, A. H. Worthington, and K. R. Venator. Prey pursuit and interception in dragonflies. *J. Compar. Physiol. A*, 186(2):155, 2000.

[185] M. J. Osborne et al. *An introduction to game theory*, volume 3. Oxford university press New York, 2004.

[186] E. Ott. *Chaos in dynamical systems*. Cambridge university press, 2002.

[187] N. B. Ouchi and K. Kaneko. Coupled maps with local and global interactions. *Chaos*, 10(2):359–365, 2000.

[188] M. C. Ozturk, D. Xu, and J. C. Principe. Analysis and design of echo state networks. *Neural computation*, 19(1):111–138, 2007.

[189] T. Palmer. Stochastic weather and climate models. *Nature Reviews Physics*, 1(7):463–471, 2019.

[190] T. Palmer, A. Döring, and G. Seregin. The real butterfly effect. *Nonlinearity*, 27(9):R123, 2014.

[191] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120(2):024102, 2018.

[192] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos*, 27(12):121102, 2017.

[193] J. Pathak, M. Mustafa, K. Kashinath, E. Motheau, T. Kurth, and M. Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations. *arXiv preprint arXiv:2010.00072*, 2020.

[194] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos*, 28(4):041101, 2018.

[195] G. Pavliotis and A. Stuart. *Multiscale methods: averaging and homogenization*. Springer, 2008.

[196] D. Pavlov and A. Kasumyan. Patterns and mechanisms of schooling behavior in fish: a review. *J. Ichthyol.*, 40(2):S163, 2000.

[197] T. J. Pedley and J. Kessler. The orientation of spheroidal microorganisms swimming in a flow field. *Proc. R. Soc. B*, 231(1262):47–70, 1987.

[198] J. Pedlosky. *Geophysical fluid dynamics*. Springer, 2013.

[199] J. Peinke, M. R. Tabar, and M. Wächter. The Fokker–Planck approach to complex spatiotemporal disordered systems. *Annu. Rev. Condens. Matter Phys.*, 10:107–132, 2019.

[200] J. P. Peixoto and A. H. Oort. *Physics of climate*. New York, NY (United States); American Institute of Physics, 1992.

[201] F. Peruani, A. Deutsch, and M. Bär. A mean-field theory for self-propelled particles interacting by velocity alignment mechanisms. *Europ. Phys. J. Spec. Top.*, 157(1):111–122, 2008.

[202] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[203] A. Pezzotta, M. Adorisio, and A. Celani. Chemotaxis emerges as the optimal solution to cooperative search games. *Phys. Rev. E*, 98(4):042401, 2018.

[204] A. Pezzotta et al. Optimal search processes in physics and biology, phd thesis. 2018.

[205] A. Pikovsky, J. Kurths, and M. Rosenblum. *Synchronization: a universal concept in nonlinear sciences*, volume 12. Cambridge university press, 2003.

[206] A. S. Pikovsky and J. Kurths. Do globally coupled maps really violate the law of large numbers? *Phys. Rev. Lett.*, 72(11):1644, 1994.

[207] T. J. Pitcher. Heuristic definitions of fish shoaling behaviour. *Anim. Behav.*, 31:611–613, 1983.

[208] L. S. Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[209] W. B. Powell. From reinforcement learning to optimal control: A unified framework for sequential decisions. In *Handbook of Reinforcement Learning and Control*, pages 29–74. Springer, 2021.

[210] M. Pulido, P. Tandeo, M. Bocquet, A. Carrassi, and M. Lucini. Stochastic parameterization identification using ensemble Kalman filtering combined with maximum likelihood methods. *Tellus A: Dyn. Meteorol. Oceanogr.*, 70(1):1–17, 2018.

[211] E. M. Purcell. Life at low reynolds number. *Am. J. Phys.*, 45(1):3–11, 1977.

[212] J. Qiu, N. Mousavi, L. Zhao, and K. Gustavsson. Active gyrotactic stability of microswimmers using hydromechanical signals. *arXiv preprint arXiv:2105.12232*, 2021.

[213] S. Ramaswamy. The mechanics and statistics of active matter. *Annu. Rev. Condens. Matter Phys.*, 1(1):323–345, 2010.

[214] G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola. Learning to soar in turbulent environments. *Proc. Nat. Acad. Sci.*, 113(33):E4877, 2016.

[215] G. Reddy, J. Wong-Ng, A. Celani, T. J. Sejnowski, and M. Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, 2018.

[216] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[217] A. Rivkind and O. Barak. Local dynamics in trained recurrent neural networks. *Phys. Rev. Lett.*, 118(25):258101, 2017.

[218] P. Romanczuk, M. Bär, W. Ebeling, B. Lindner, and L. Schimansky-Geier. Active brownian particles. *Europ. Phys. J. Spec. Top.*, 202(1):1–162, 2012.

[219] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65(6):386, 1958.

[220] P. Rotondo, M. Pastore, and M. Gherardi. Beyond the storage capacity: Data-driven satisfiability transition. *Phys. Rev. Lett.*, 125(12):120601, 2020.

[221] N. F. Rulkov, M. M. Sushchik, L. S. Tsimring, and H. D. Abarbanel. Generalized synchronization of chaos in directionally coupled chaotic systems. *Phys. Rev. E*, 51(2):980, 1995.

[222] P. Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer, 2006.

[223] J. A. Sanders, F. Verhulst, and J. Murdock. *Averaging methods in nonlinear dynamical systems*, volume 59. Springer, 2007.

[224] F. Scarselli and A. C. Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Netw.*, 11(1):15–37, 1998.

[225] A. M. Schäfer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In *International Conference on Artificial Neural Networks*, pages 632–640. Springer, 2006.

[226] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proc. 15th Europ. Symp. Artif. Neural Netw.*, pages 471–482, 2007.

[227] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, 2008.

[228] T. Shibata and K. Kaneko. Collective chaos. *Phys. Rev. Lett.*, 81:4116–4119, Nov 1998.

[229] A. B. Sichert, R. Bamler, and J. L. van Hemmen. Hydrodynamic object recognition: when multipoles count. *Phys. Rev. Lett.*, 102(5):058104, 2009.

[230] S. Siegert, R. Friedrich, and J. Peinke. Analysis of data sets of stochastic systems. *Physics Letters A*, 243(5-6):275–280, 1998.

[231] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.*, 23(5):828–841, 2019.

[232] R. T. Sullivan. Insect swarming and mating. *Fla. Entomol.*, 64(1):44–65, 1981.

[233] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[234] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Adv. Neural. Inf. Process. Syst.*, pages 1057–1063, 2000.

[235] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[236] D. Takagi and D. K. Hartline. Directional hydrodynamic sensing by free-swimming organisms. *Bull. Math. Biol.*, 80(1):215, 2018.

[237] F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[238] K. A. Takeuchi, H. Chaté, F. Ginelli, A. Politi, and A. Torcini. Extensive and subextensive chaos in globally coupled dynamical systems. *Phys. Rev. Lett.*, 107(12):124101, 2011.

[239] K. A. Takeuchi, F. Ginelli, and H. Chaté. Lyapunov analysis captures the collective dynamics of large chaotic systems. *Phys. Rev. Lett.*, 103(15):154103, 2009.

[240] G. Tanaka, T. Matsumori, H. Yoshida, and K. Aihara. Reservoir computing with diverse timescales for prediction of multiscale dynamics. *arXiv preprint arXiv:2108.09446*, 2021.

[241] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Netw.*, 115:100–123, 2019.

[242] P. S. Thomas, W. Dabney, S. Giguere, and S. Mahadevan. Projected natural actor-critic. In *NIPS*, pages 2337–2345, 2013.

[243] E. Todorov. Efficient computation of optimal actions. *Proc. Natl. Acad. Sci. U.S.A.*, 106(28):11478–11483, 2009.

[244] M. K. Transtrum, B. B. Machta, K. S. Brown, B. C. Daniels, C. R. Myers, and J. P. Sethna. Perspective: Sloppiness and emergent theories in physics, biology, and beyond. *J. Chem. Phys.*, 143(1):07B201_1, 2015.

[245] M. S. Triantafyllou, G. D. Weymouth, and J. Miao. Biomimetic survival hydrodynamics and flow sensing. *Ann. Rev. Fluid Mech.*, 48:1, 2016.

[246] J. Tubiana, S. Cocco, and R. Monasson. Learning protein constitutive motifs from sequence data. *Elife*, 8:e39397, 2019.

[247] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin. Self-organized flocking in mobile robot swarms. *Swarm. Intell.*, 2(2):97–120, 2008.

[248] L. J. Tuttle, H. E. Robinson, D. Takagi, J. R. Strickler, P. H. Lenz, and D. K. Hartline. Going with the flow: hydrodynamic cues trigger directed escapes from a stalking predator. *J. Royal Soc. Interface*, 16(151):20180776, 2019.

[249] D. Ullmo, I. Swiecicki, and T. Gobron. Quadratic mean field games. *Phys. Rep.*, 799:1–35, 2019.

[250] B. van den Broek, W. Wiegerinck, and B. Kappen. Stochastic optimal control of state constrained systems. *International Journal of Control*, 84(3):597–615, 2011.

[251] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[252] V. N. Vapnik. An overview of statistical learning theory. *IEEE trans. neural netw.*, 10(5):988–999, 1999.

[253] S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Nat. Acad. Sci.*, 115(23):5849–5854, 2018.

[254] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Net.*, 20(3):391–403, 2007.

[255] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Phy. Rev. Lett.*, 75(6):1226, 1995.

[256] T. Vicsek and A. Zafeiris. Collective motion. *Phys. Rep.*, 517(3-4):71–140, 2012.

[257] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, and T. Vicsek. Flocking algorithm for autonomous flying robots. *Bioinspir. Biomim.*, 9(2):025012, 2014.

[258] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. Royal Soc. A*, 474(2213):20170844, 2018.

[259] A. Vulpiani and M. Baldovin. Effective equations in complex systems: from langevin to machine learning. *J. Stat. Mech. Theory Exp.*, 2020(1):014003, 2020.

[260] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[261] Z. Warhaft. Turbulence in nature and in the laboratory. *Proc. Nat. Acad. Sci.*, 99(suppl 1):2481–2486, 2002.

[262] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[263] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990.

[264] G. D. Weymouth and D. K. P. Yue. Physics-based learning models for ship hydrodynamics. *J. Ship Res.*, 57(1):1–12, 2013.

[265] S. Whiteson and D. Whiteson. Machine learning for event selection in high energy physics. *Eng. Appl. Artif. Intell.*, 22(8):1203–1217, 2009.

[266] A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos*, 30(5):053111, 2020.

[267] A. Wikner, J. Pathak, B. R. Hunt, I. Szunyogh, M. Girvan, and E. Ott. Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components<? a3b2 show [editpick]?>. *Chaos*, 31(5):053114, 2021.

[268] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[269] C. W. Wolgemuth. Collective swimming and the dynamics of bacterial turbulence. *Biophys. J.*, 95(4):1564–1574, 2008.

[270] R. J. Wubbels and N. A. M. Schellart. Neuronal encoding of sound direction in the auditory midbrain of the rainbow trout. *J. Neurophysiol.*, 77(6):3060, 1997.

[271] H.-l. Yang, K. A. Takeuchi, F. Ginelli, H. Chaté, and G. Radons. Hyperbolicity and the effective dimension of spatially extended dissipative systems. *Phys. Rev. Lett.*, 102(7):074102, 2009.

[272] I. B. Yildiz, H. Jaeger, and S. J. Kiebel. Re-visiting the echo state property. *Neural Netw.*, 35:1–9, 2012.

[273] D. K. Zachariah Carmichael, Humza Syed. Analysis of wide and deep echo state networks for multiscale spatiotemporal time series forecasting. 7 2019.

[274] L. Zdeborová. Understanding deep learning is also a job for physicists. *Nature Phys.*, 16(6):602–604, 2020.

[275] G. Zhang, Z. Liu, and Z. Ma. Generalized synchronization of different dimensional chaotic dynamical systems. *Chaos. Solit.*, 32(2):773–779, 2007.