Control of Dynamical Systems with Intrinsic Nonadiabatic Time Dependence using Deep Reinforcement Learning

This thesis is submitted in partial fulfillment of the requirements

for the degree of

B.Sc. in Theoretical Physics

by

Georgi Aleksandrov

(student ID: 13152)

Theoretical Physics Department
Faculty of Physics
Sofia University

July 16, 2023

Approved by

Thesis Supervisor: Marin Bukov, PhD

Co-advisor: Assoc. Prof. Dr. Andon Rangelov

Referee: Friederike Metz, PhD

defense date: 24.07.2023

Certificate of Originality

*This is to certify that this thesis, entitled "**Control of Dynamical Systems with Intrinsic Nonadiabatic Time Dependence using Deep Reinforcement Learning**", submitted in partial fulfillment of the requirements for the degree B.Sc. in Theoretical Physics at Sofia University, is my own work and contains original results obtained by myself with the support and assistance of my supervisor.*

In particular, I certify that results and ideas obtained by third parties that are described, published, or authored elsewhere, are to a sufficient extent referred to, properly cited, and acknowledged in the thesis, in accordance with the requirements for respecting intellectual property, and the academic ethics and standards.

I am aware that, should plagiarism or scientific misconduct be established, the Thesis Committee reserves the right to reject this thesis, while the Committee for Academic Ethics at the Ministry of Science and Education has the legal right to retroactively revoke the degree conferred.

I declare that the results contained in the present thesis have not been submitted at any other university, institute, or higher-education institution in (partial) fulfillment of the requiements for an academic degree.

Sofia
July 16, 2023

Georgi Aleksandrov
student ID: 13152
Faculty of Physics
Sofia University

Abstract

Reinforcement Learning (RL) is a branch of Machine Learning that has made rapid inroads into various fields of science in recent years. It finds a natural application in the control of dynamical systems. If an RL agent possesses the knowledge of the state of a system and is given a reward, which is maximized when the system is in a given target state, the agent finds increasingly better control policies on its own. If the dynamics of the system change over time, but the changes are adiabatic (sufficiently slow), the RL agent can gradually adapt to the changed system. On the other hand, if the system dynamics change too quickly – within the training episode, traditional RL agents could fail. Their decisions would have to depend not only on the state of the environment but also on time.

In this thesis, we first review selected concepts of Hamiltonian Mechanics, which is the framework in which we will later consider some classical dynamical systems. We then look at the basics of Quantum mechanics, with an emphasis on two-level systems (qubits). After that, we introduce the most important concepts in Reinforcement Learning (RL), as well as the basic algorithm Policy Gradient (PG), which we will use thereafter.

In the next part of the thesis, we propose a way in which the framework of Reinforcement Learning can be extended, as the RL agent will also possess information about the time at which it acts on the system. We consider 3 particular classical systems – a particle contained in different time-dependent potentials which are not known by the agent in advance. The agent controls the particle by applying a small external force. Finally, we consider a quantum two-level system with time-dependent decay of the state. There, the agent can act on the quantum state with unitary transformations (quantum gates). We demonstrate that the RL agent learns a better strategy when it has information about time.

Контрол на динамични системи с явна неадиабатична времева зависимост чрез дълбоко обучение с утвърждение

Дипломна работа представена за частично покриване на изискванията
за образователно-квалификационна степен
бакалавър "Квантова и космическа теоретична физика"

от

Георги Мирославов Александров
(факултетен номер: 13152)

Катедра Теоретична Физика
Физически Факултет
Софийски университет „Св. Климент Охридски“

16 юли 2023 г.

Одобрена от

ръководител: д-р Марин Буков

консултант: доц. дфзн Андон Рангелов

рецензент: д-р Фридерике Метц

дата на защитата: 24.07.2023

Декларация за оригиналност и автентичност

Аз, ***Георги Мирославов Александров****, студент от Физически Факултет на Софийски университет „Св. Климент Охридски“, декларирам, че представената от мен за защита дипломна работа на тема:* ***„Контрол на динамични системи с явна неадиабатична времева зависимост чрез дълбоко обучение с утвърждение”****, за присъждане на образователно-квалификационна степен бакалавър "Квантова и космическа теоретична физика" е оригинална разработка и съдържа оригинални резултати, получени при проведени от мен научни изследвания (с подкрепата и/или съдействието на научния ми ръководител).*

Декларирам, че резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани, при спазване на изискванията за защита на авторското право и на академичната етика и стандарти.

Уведомен/а съм, че в случай на констатиране на плагиатство или недостоверност на представените научни данни, Комисията по защитата е в правото си да я отхвърли, а Комисията по академична етика към Министерство на образованието и науката е в законното си правото да анулира придобитата образователно-квалификационна степен.

Декларирам, че настоящият труд не е представян пред други университети, институти и други висши училища за придобиване на образователна и/или научна степен.

София
16 юли 2023 г.

Георги Мирославов Александров
факултетен номер: 13152
Физически Факултет
Софийски университет „Св. Климент Охридски“

Абстракт

Обучението с утвърждение – Reinforcement Learning (RL) – е дял на машинното обучение, който през последните години бурно навлезе в различни области на науката. То намира естествено приложение е в контрола на динамични системи. Ако RL-агент притежава знанието за състоянието на една система и му бъде зададена награда, която е максимална в дадено целево състояние на системата, той самостоятелно намира все по-добри стратегии за контрол на системата. Ако динамиката на системата се променя с времето, но промените са достатъчно бавни (адиабатични), RL-агентът може постепенно да се адаптира към променената система. Но ако динамиката на системата се променя много бързо – в рамките на епизода на тренирането, то традиционните RL-агенти биха могли да се провалят. Техните действия ще трябва да зависят освен от състоянието на средата, и от времето.

В тази дипломна работа първо правим обзор на избрани части от хамилтоновата механика, в рамките на която ще разглеждаме класически динамични системи. След това разглеждаме основите на квантовата механика, с акцент върху системите от две нива (кюбити). После въвеждаме и най-важните понятия от обучението с утвърждение (RL), както и основният алгоритъм Policy Gradient (PG), който ще използваме по-нататък.

В следващата част от дипломната работа предлагаме начин, по който може да се разшири рамката на обучението с утвърждение, при което RL-агентът разполага и с информация за момента от време, в който трябва да вземе решението си. Разглеждаме с 3 конкретни класически системи – материална точка, намираща се в различни потенциали, зависещи от времето по закон, който не е предварително известен на агента. Агентът контролира частицата чрез прилагане на малка външна сила. Освен това разглеждаме и една квантова система с 2 нива, в която е наличен времезависим разпад на състоянието. В нея агентът може да действа върху състоянието с унитарни трансформации (гейтове). Демонстрираме, че RL-агентът се научава на по-добра стратегия, когато разполага с информация за времето.

Acknowledgments

I would like to express my deepest gratitude to my advisor, **Marin Bukov, PhD**, for his guidance, support, and expertise throughout the process of writing this thesis. He was the person who introduced me to the world of research and without his encouragement, this thesis would not have been possible.

I would also like to thank everyone who constantly asked about my progress and expected me to be ahead of schedule.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI Artificial Intelligence 1, 3, 4

MC Monte Carlo 27, 52–54, 57

PG Policy Gradient iii, ix, 6, 26, 27, 30, 33, 38, 44, 52–54, 57

RL Reinforcement Learning iii, viii, 3–6, 21, 23, 26, 30, 32, 33, 35, 37, 38, 40, 41, 44, 45, 47, 49, 50, 55, 56

Chapter 1

Introduction

1.1 Dynamical Systems

Consider a system of particles. We can describe its state with a set of coordinates. A dynamical system is one whose state evolution is given by ordinary differential equations. These equations contain the coordinates, their time derivatives, and in some cases, time explicitly.

In classical mechanics, dynamical systems are deterministic – by knowing the initial state of the system and its equations of motion, in theory, we can deduce its evolution as the equations of motion have a unique solution. In practice, for simpler systems, it is indeed possible to solve the equations of motion analytically. However, more complicated systems cannot be analytically solved. In this case, we could at best obtain approximate solutions with various methods, including computer simulations. There are some dynamical systems that exhibit chaos – even our computer simulations can fail to predict the long-term behavior of such systems, since small differences in the initial state can result in exponentially growing differences in the subsequent dynamics.

As various dynamical systems can be directly applied in engineering, they need to behave predictably. There should be mechanisms in place to alter their evolution in a desired way. That is why, in the 19th century naturally emerged **Control theory**, the field that studies how we can create models that govern the behavior of the dynamical system. One valuable approach for the control of dynamical systems, which we will use in this thesis, is **Reinforcement Learning**.

1.2 Machine Learning

In the last decade, our technological progress reached the tipping point which allowed Machine learning to succeed in various tasks with the potential of changing our life.

Machine learning is a subfield of Artificial Intelligence (AI). AI is the science and engineering of making intelligent computer programs [1]. "Intelligent" means that they do not simply follow hard-wired instructions in order to achieve a certain task. Instead, the program should possess some kind of understanding of the task and act in such a way as to reach a certain goal which was set by programmers. The purpose of Machine

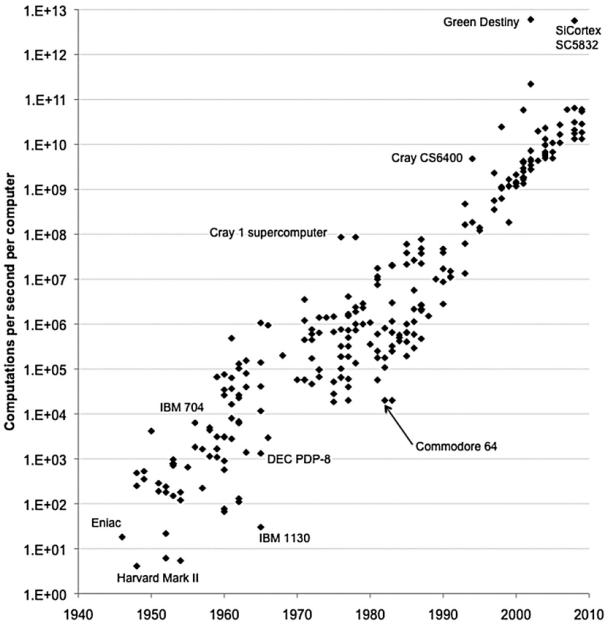

Figure 1.1: Moore's law-like exponential increase of the computational capacity over time (computations/second per computer). Figure from Ref. [2]

learning is to develop algorithms which use "training" data in order to "learn" how to act. "Learning" means that they are improving their performance by receiving data.

Machine learning has yielded impressive results due to two main factors. Firstly, the computational power of machines is increasing exponentially. According to Moore's law, the number of transistors per integrated circuit doubles every two years, in turn resulting in exponential increase of the computational capacity of supercomputers (see Fig. 1.1). Secondly, the Internet provided us with large amounts of information. "Big data" are large datasets containing petabytes of information on a certain topic. They are difficult for traditional analysis, but required for building intelligent systems.

Machine learning can be divided in three main paradigms – Supervised learning, Unsupervised learning and Reinforcement learning.

In Supervised Learning, the algorithm is trained using a set of **training examples**, consisting of inputs ("features") and desired outputs ("labels"). The algorithm has to infer a function that matches each input to the output by maximizing a "reward" function /alternatively, minimizing a "cost" function/ that depends on the deviation between the correct output and the one produced by the algorithm.

For example, to train an algorithm to recognize handwritten digits, each training example will consist of an image containing a handwritten digit and a label – the digit corresponding to the image. The program, during training, gradually learns the dependence between images and digits.

In Unsupervised Learning, the program receives unlabelled input data. The goal of such a program could be to find similarities between the data points it encounters. It would either try to cluster or order the data, or it would have a goal to produce new data that mimics the one it was trained on.

In Reinforcement Learning, the program functions as an "agent" which interacts with the environment. Its actions affect the environment. We do not input pre-labeled desired

actions to the agent. Instead, a reward is given to the agent, in the form of a number which should be maximized, depending on the state of the environment and its action. The reward function is set in such a way as to be higher when a certain objective is accomplished.

This training is similar to classical conditioning of intelligent animals. For example, if we want to teach a dog to perform a certain trick, we could give it an award (e.g. food or praise) each time the dog moves the right way. Over time, the dog will associate its behavior with the positive "reinforcement", performing the trick better with time.

1.2.1 Machine Learning Milestones

Although we have not yet developed artificial general intelligence, AI has outperformed human intelligence in various tasks where it would be very difficult to program specific instructions. In 1997, Deep Blue surpassed human-level chess by using a combination of large databases of grandmasters' openings and endgames, as well as brute force allowed by its high computing power. The currently best playing chess systems, Stockfish and Leela Chess Zero are using neural networks, which gives them additional improvement in performance compared to Deep Blue, achieving an estimated ELO-rating of above 3500[1], compared to below 2900 for the current best human player[2].

As there is an average of 35 possible moves in Chess and the average length of a game is 80 moves [3], algorithms based on brute force only are inefficient. However, there is a game with even more possibilities which survived 18 years longer as a fortress of human intelligence. In the game Go, there are around 250 legal moves and the game lasts about 150 moves, making it impossible for any brute force-based algorithm. Humans navigate through this vast number of possibilities not only usng logic, but also creativity and intuition, which are difficult concepts for a computer. However, in 2015, the program AlphaGo won against the best human Go player. It uses a combination of tree search and deep neural networks, which were trained using a combination of supervised learning, using information from expert human players and Reinforcement Learning, by playing on its own [4]. The current best Go engines, KataGo and Leela Zero, reached even higher level of performance, with Leela Zero not requiring any human player input, and KataGo requiring human games only for rating evaluation.

In the last years, we have witnessed major progress in even more human tasks such as image and text analysis and generation. China actively uses facial recognition software in many areas of life such as digital payments, hospital waiting rooms, housing complexes, transportation systems and urban policing [5]. Image generation is currently in a state where unsuspecting humans cannot differentiate between real and "deepfake" images [6]. Text generation programs can pass exams in various fields of study, producing sensible natural human-like text and following instructions written in natural text [7].

Machine Learning is not only an interesting game-playing or text-writing tool. It has a real impact on scientific and technological progress. Recently, a new sorting algorithm was discovered by a self-learning RL agent, leading to an update to the standard C++ sort library, which is estimated to be used trillions of times a day [8]. In our quest for using nuclear fusion as a renewable energy source, it could prove beneficial to use RL to

[1] The ratings of the current best chess engines are listed at https://ssdf.bosjo.net/

[2] The ratings of the best human chess players are available on the International Chess Federation site: https://ratings.fide.com/id.phtml?event=1503014

control tokamak plasmas [9].

1.3 Reinforcement Learning in Physics

We will now take a look at some of the contemporary applications of Reinforcement Learning in Physics.

1.3.1 Reinforcement Learning in Quantum Mechanics

The scope of AI applications in quantum technologies spans from interpreting measurement data, estimating the properties of quantum systems, discovering quantum control algorithms and discovering new quantum circuits and algorithms for error correction [10].

Preparation of ground states with Reinforcement Learning

In Quantum Mechanics, an important task we should learn to achieve is the preparation of a many-body system in its ground state. This is required in various quantum computing algorithms, as well as in quantum simulators, in order to study the fundamental properties of many-body quantum systems. Reinforcement Learning can help in this area.

In the last years, researchers have used RL to prepare ground states of the quantum Ising chain, whose state is represented as Matrix product state [11], search for the correct parameters of the Quantum Approximate Optimization Algorithm (QAQA) [12], as well as generalizing the algorithm (CD-QAQA) [13], optimizing it via RL for preparation of spin chains.

In Ref. [14], a near-optimal driving protocol in a system of interacting qubits was found using RL even in a state manipulation phase where this process is exponentially harder. In Ref. [15], various quantum states were prepared using RL in a system of a quantum harmonic oscillator coupled to an ancilla qubit. Their research can be experimentally applied in trapped ions platforms. RL algorithms have been demonstrated to outperform non-Machine Learning methods in preparing a desired quantum state when the problem is discretized and scaled up [16].

RL was also used in the area of coherent transport, where it discovered a control sequence that outperforms the "counterintuitive control sequence" [17]. Also, in Floquet systems, RL has shown to take advantage of the micromotion dynamics which in the traditional analysis remain neglected [18].

Reinforcement learning in Quantum Computing

We are approaching the inevitable end of Moore's law for classical computers, as the sizes of transistors approach the size of individual atoms. The next step of our technological development is the construction of quantum computers. Currently, the greatest setback of quantum computers is quantum decoherence, which causes unpredictable errors in quantum computations. RL looks promising in this area as well, with numerous research showing considerable progress. For instance, in Ref. [19], RL simultaneously optimizes the

speed and fidelity of quantum computation against both leakage and stochastic control error.

The development of error correction systems needs to take into account that we do not possess the full information about errors. If we try to measure errors completely, this would destroy the quantum information. Instead, we have access to so called "syndromes" – partial error information, which is more difficult to interpret. In Ref. [20], an RL-based decoder can suggest the best error correction to perform for any given syndrome, matching the performance of hand-made algorithms. In simplified settings, RL-based decoders for topological error-correcting codes were developed in Ref. [21].

However, quantum error correction generates new errors itself, which may make it unpractical. This year, by implementing multiple innovations including RL, a real-time quantum error correction was achieved at a level that the speed of error corrections is now faster than the speed of new error generation [22].

Another area with active development in quantum computing is the design of new sets of gates. The gate design can help minimize quantum errors. In Ref. [23], RL agents design qubit gates which outperform hardware default gates and exhibit superior calibration-free performance. In Ref. [24], researchers use RL to limit the number of required quantum gates, which improves accuracy and simulation times in the presence of experimental imperfections.

1.3.2 Reinforcement Learning in Dynamical Systems

Reinforcement Learning is generally well-suited for control of dynamical systems. Its framework gives a natural way to frame control problems as RL problems. In control problems, we need an external mechanism to act on a physical system. In Reinforcement Learning, an RL agent produces actions on an environment [25]. This means that we can directly identify the external machinery with an RL agent, and the physical system as an environment.

However, for the traditional training of RL agents to control a dynamic system, there is a requirement that the system itself does not change with time – the evolution of the system state should depend on its current state only. If the environment changes with time, i.e., its evolution has explicit time dependence, we may need to generalize the current RL algorithms, as will be discussed in this thesis.

In Ref. [26] there are shown the various ways current RL algorithms have difficulties in time-dependent environments. Typically, time-dependent environments tackled by RL depend on time **adiabatically**. That means that the time dependence is slow enough, much slower than the time of one training episode, making the agent gradually adapt to the newer environment. In Ref. [27], for example, a clever way of designing an "optimistic" agent is suggested. It tries to predict the direction of change in the environment in future training episodes. Similarly, Ref. [28] suggest a "prognosticator" – an RL agent that optimizes its future performance. In Ref. [29], non-stationary Markov decision processes are discussed, with the hypothesis of slowly changing environments. There are other efforts in the theoretical development of ε-Markov decision process models, which rely on the hypothesis that the environments have small variations from being static [30].

Apart from these examples, in this thesis, we are interested in generalizing Reinforcement

Learning to systems with **intrinsic nonadiabatic** time dependence. This time dependence manifests itself in the scope of a single episode, making it impossible for the agent to adapt to a "modified environment" slowly. Instead, it has to learn the intrinsic time dependence of the environment and produce optimal policies, taking into account both the state of the system and time.

This thesis is organized as follows. In Chapter 2, we outline the foundations of Hamiltonian mechanics, starting from Newtonian mechanics, then introducing the Lagrangian formalism, and finally, the Hamiltonian formalism. In Chapter 3, we introduce some elements of quantum mechanics, starting from the foundations – states, observables, and measurement, and lay down the notion of a two-level system. In Chapter 4, we introduce Reinforcement Learning (RL) and the Policy Gradient algorithm in particular. In Chapter 5, we apply RL to control four dynamical systems. The problems in 5.1 are about controlling Hamiltonian systems, and the problem in 5.2 is a quantum problem, namely to prepare a two-level system in a desired state.

Chapter 2

Overview of Classical Mechanics

This chapter introduces the Lagrangian and Hamiltonian formalism in classical mechanics, as we will be trying to control Hamiltonian systems in Chap. 5.1 and we will need different coordinate systems and coordinate transformations.

2.1 Newtonian Dynamics

To describe the motion of a classical particle in 3D Euclidean space, we need a *frame of reference* containing an origin we consider stationary, a coordinate system attached to it, and a clock measuring time t. The simplest coordinate system is the Cartesian (x, y, z).

Then, the position vector of a particle is $\vec{r} = (x, y, z)$. The velocity of the particle is defined as $\vec{v} = \dot{\vec{r}} = (\dot{x}, \dot{y}, \dot{z})$, and the acceleration is $\vec{a} = \dot{\vec{v}} = (\ddot{x}, \ddot{y}, \ddot{z})$. The "dot" means derivative with respect to time.

In *Newtonian mechanics*, all particles in the Universe interact with forces $\vec{F}$, which follow Newton's laws of motion:

1. In an *inertial frame of reference*, if no forces are acting on a particle, its velocity vector is constant.
2. A force $\vec{F}$ acting on a particle causes acceleration $\vec{a} = \vec{F}/m$, where m is the mass of the particle.
3. All pairs of particles (i, j) interact with forces of equal magnitude and opposite directions.

We can define the **momentum** of the particle as $\vec{p} = m\vec{v}$. In this way, $m\vec{a} = \dot{\vec{p}}$. Therefore, Newton's second law can be reformulated in this way: forces cause a change in the particle momentum. Also, Newton's third law implies that any interaction between particles does not change the total momentum of the system of particles.

We define the **kinetic energy** of the particle as

$$T = \frac{m\vec{v}^2}{2}. \tag{2.1}$$

Using this definition, the change of the kinetic energy is

$$\delta T = m\vec{v} \cdot \mathrm{d}\vec{v} = m\vec{a} \cdot \vec{v}\mathrm{d}t = \sum_{i=1}^{n} \vec{F}_i \cdot \mathrm{d}\vec{r} = \delta W, \tag{2.2}$$

which is the work done on the particle by all forces $\vec{F}_i$ acting on it.

A force $\vec{F}$ is conservative if the work W done on the particle by the force $\vec{F}$ is a function of the initial $\vec{r}_1$ and final $\vec{r}_2$ positions of the particle, and does not depend on the intermediate trajectory. For all conservative forces, we can define the **potential energy** $V(\vec{r})$, such that $W(\vec{r}_1, \vec{r}_2) = V(\vec{r}_1) - V(\vec{r}_2)$. In this way, if we only exert conservative forces on a particle, its total energy $E = T + V$ will be constant:

$$T_2 - T_1 = W_{12} = V_1 - V_2 \Rightarrow T_1 + V_1 = T_2 + V_2. \tag{2.3}$$

2.2 Lagrangian Mechanics

As $\delta W = -\mathrm{d}V = \vec{F} \cdot \mathrm{d}\vec{r}$, the force can be expressed as a gradient of the potential energy:

$$\vec{F} = -\vec{\nabla} V, \tag{2.4}$$

$$F_x = -\frac{\partial V}{\partial x}, \quad F_y = -\frac{\partial V}{\partial y}, \quad F_z = -\frac{\partial V}{\partial z}. \tag{2.5}$$

The momentum can be expressed as the derivative of the kinetic energy with respect to the components of the velocity:

$$\frac{\partial T}{\partial \dot{x}} = m\dot{x} = p_x; \quad \frac{\partial T}{\partial \dot{y}} = p_y; \quad \frac{\partial T}{\partial \dot{z}} = p_z. \tag{2.6}$$

Since the V depends only on the coordinates and T – only on the derivatives of the coordinates, we can see that the equation of motion $\vec{F} = m\vec{a}$ of the particle is equivalent to the **Euler-Lagrange equation**:

$$\frac{\partial L}{\partial x_i} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{x}_i}, \quad i = 1, 2, 3, \tag{2.7}$$

where

$$L(\vec{r}, \dot{\vec{r}}, t) = T(\dot{\vec{r}}) - V(\vec{r}) \tag{2.8}$$

is called **Lagrangian** of the particle and $(x_1, x_2, x_3) = (x, y, z)$.

This equation has a new physical meaning – the particle follows the trajectory minimizing the **action** functional

$$S(\vec{r}(t)) = \int_{t_1}^{t_2} L(\vec{r}, \dot{\vec{r}}, t)\mathrm{d}t \tag{2.9}$$

with fixed ends t_1, t_2. Indeed, if we vary the functional S, we will get

$$\begin{aligned}
\delta S &= \int_{t_1}^{t_2} \delta L \mathrm{d}t = \int_{t_1}^{t_2} \sum_{i=1}^{3} \left(\frac{\partial L}{\partial x_i} \delta x_i + \frac{\partial L}{\partial \dot{x}_i} \delta \dot{x}_i \right) \mathrm{d}t \\
&= \int_{t_1}^{t_2} \sum_{i=1}^{3} \frac{\partial L}{\partial x_i} \delta x_i \mathrm{d}t + \int_{t_1}^{t_2} \sum_{i=1}^{3} \frac{\partial L}{\partial \dot{x}_i} \mathrm{d}\delta x_i \\
&= \int_{t_1}^{t_2} \sum_{i=1}^{3} \frac{\partial L}{\partial x_i} \delta x_i \mathrm{d}t + \sum_{i=1}^{3} \frac{\partial L}{\partial \dot{x}_i} \delta x_i \Big|_{t_1}^{t_2} - \int_{t_1}^{t_2} \sum_{i=1}^{3} \delta x_i \mathrm{d} \left(\frac{\partial L}{\partial \dot{x}_i} \right) \\
&= \int_{t_1}^{t_2} \sum_{i=1}^{3} \left(\frac{\partial L}{\partial x_i} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{x}_i} \right) \delta x_i \mathrm{d}t,
\end{aligned} \tag{2.10}$$

which is zero for any variation of $x_i(t)$ iff all three terms in the integrand are zero, leading to the Euler-Lagrange equations.

The particle chooses the trajectory with minimal action, but the action depends only on the trajectory, not on the specific parametrization of the trajectory. This means that the Euler-Lagrange equation is valid not only for the Cartesian coordinates (x_1, x_2, x_3), but for any **generalized coordinates** q_i:

$$\frac{\partial L}{\partial q_i} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{q}_i}. \tag{2.11}$$

2.3 Hamiltonian Mechanics

Using the generalized coordinates, we can also define generalized momenta:

$$p_i = \frac{\partial L}{\partial \dot{q}_i}. \tag{2.12}$$

Considering that due to Lagrange-Euler equation,

$$\dot{p}_i = \frac{\partial L}{\partial q_i}, \tag{2.13}$$

the total differential of the Lagrangian can be written as

$$\mathrm{d}L = \sum_{i=1}^{n} \left(\frac{\partial L}{\partial q_i} \mathrm{d}q_i + \frac{\partial L}{\partial \dot{q}_i} \mathrm{d}\dot{q}_i \right) = \sum_{i=1}^{n} (\dot{p}_i \mathrm{d}q_i + p_i \mathrm{d}\dot{q}_i). \tag{2.14}$$

The Lagrangian is a function of q_i and $\dot{q}_i$. However, we could apply a Legendre transformation to acquire a new function $H = H(p_i, q_i)$. Notice that $\mathrm{d}(p_i \dot{q}_i) = \dot{q}_i \mathrm{d}p_i + p_i \mathrm{d}\dot{q}_i$.

We define the **Hamiltonian** as

$$H(p_i, q_i) = \sum_{i=1}^{N} p_i \dot{q}_i - L. \tag{2.15}$$

This way,

$$\mathrm{d}H = \sum_{i=1}^{n} (\dot{q}_i \mathrm{d}p_i - \dot{p}_i \mathrm{d}q_i), \tag{2.16}$$

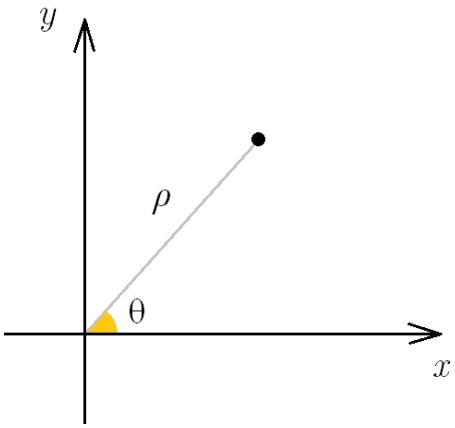

Figure 2.1: Polar coordinates

leading to the following **Hamilton's equations**:

$$\begin{aligned} \dot{p}_i &= -\frac{\partial H}{\partial q_i}, \\ \dot{q}_i &= \frac{\partial H}{\partial p_i}. \end{aligned} \tag{2.17}$$

These equations are equivalent to the original equations of motion. However, they are first-order differential equations, which means they are easier to solve numerically. We will be using these equations in our simulations. Any system, which obeys Hamilton's equation, is called a **Hamiltonian system**.

Suppose the particle is only subject to a constant force $\vec{F}$. Then, its potential energy can be defined as $V = -\vec{F} \cdot \vec{r}$, since its work from any position $\vec{r}_1$ to $\vec{r}_2$ is $W = \vec{F} \cdot (\vec{r}_2 - \vec{r}_1)$, not depending on the exact path of the particle. If the particle is subject both to a potential $V(\vec{r})$ and an external constant force $\vec{F}$, the total Hamiltonian would be

$$H = \frac{\vec{p}^2}{2m} + V(\vec{r}) - \vec{F} \cdot \vec{r}. \tag{2.18}$$

2.3.1 Polar coordinates

Consider a 2D space with Cartesian coordinates (x, y). The polar coordinates (ρ, θ) of a particle are defined as the distance to the origin of the coordinate system and the angle between the direction of the x-axis and the direction to the particle. We can transition from (x, y) to (ρ, θ) using

$$\begin{aligned} x &= \rho \cos\theta, \\ y &= \rho \sin\theta. \end{aligned} \tag{2.19}$$

The square of the velocity in polar coordinates becomes

$$\begin{aligned} \vec{v}^2 &= \dot{x}^2 + \dot{y}^2 = (\dot{\rho}\cos\theta + \rho(-\sin\theta)\dot{\theta})^2 + (\dot{\rho}\sin\theta + \rho(\cos\theta)\dot{\theta})^2 = \\ &= \dot{\rho}^2(\cos^2\theta + \sin^2\theta) + 2\dot{\rho}\rho\dot{\theta}(\sin\theta\cos\theta - \sin\theta\cos\theta) + \rho^2\dot{\theta}^2(\sin^2\theta + \cos^2\theta) = \\ &= \dot{\rho}^2 + \rho^2\dot{\theta}^2 \end{aligned} \tag{2.20}$$

This means that the Lagrangian has the form

$$L = \frac{m\dot{\rho}^2}{2} + \frac{m\rho^2\dot{\theta}^2}{2} - V(\rho, \theta). \tag{2.21}$$

The radial and angular momentum of the particle are defined using the standard procedure for generalized momenta (eq. (2.12)):

$$\begin{aligned} p_\theta &= \frac{\partial L}{\partial \dot{\theta}} = m\rho^2\dot{\theta}, \\ p_\rho &= \frac{\partial L}{\partial \dot{\rho}} = m\dot{\rho}. \end{aligned} \tag{2.22}$$

The Hamiltonian we acquire in polar coordinates has the form

$$H = \frac{p_\rho^2}{2m} + \frac{p_\theta^2}{2m\rho^2} + V(\rho, \theta) \tag{2.23}$$

Hamilton's equations in 2D polar coordinates. Now, an external force can again be added to the Hamiltonian as an $-\vec{F}.\vec{r}$ term. Let $\vec{F} = F_\rho\vec{e}_\rho + F_\theta\vec{e}_\theta$, where F_ρ and F_θ are the components of the force in the ρ and θ-directions respectively, and $\vec{e}_\rho$ and $\vec{e}_\theta$ are basis vectors with length 1. Then,

$$H = \frac{p_\rho^2}{2m} + \frac{p_\theta^2}{2m\rho^2} + V(\rho, \theta) - F_\rho\rho - F_\theta\rho\theta, \tag{2.24}$$

$$\begin{aligned} \dot{p_\rho}(t) &= \frac{p_\theta^2}{m\rho^3} - \frac{\partial V}{\partial \rho} + F_\rho, \\ \dot{p_\theta}(t) &= -\frac{\partial V}{\partial \theta} + \rho F_\theta, \\ \dot{\rho}(t) &= \frac{p_\rho}{m}, \\ \dot{\theta}(t) &= \frac{p_\theta}{mr^2}. \end{aligned} \tag{2.25}$$

Chapter 3

Overview of Quantum Mechanics

This chapter introduces the foundational notions in quantum mechanics. It is also an introduction to the ideas related to the two-level systems, which we will need in Chapter 5.2

Unlike classical mechanics, in quantum mechanics, we cannot know the precise positions and momenta of the particles simultaneously. Instead, we may know the probability ΔP of a particle being in a region of space ΔV.

The **probability density** function is defined as

$$f(\vec{r}) = \lim_{\Delta V \to 0} \frac{\Delta P}{\Delta V}, \tag{3.1}$$

the ratio between the probability of the particle being in the neighborhood of $\vec{r}$ to the volume of that neighborhood.

It is experimentally established that quantum particles on small lengthscales exhibit wave-like properties. A free particle with momentum p behaves like a wave with wavelength $\lambda = h/p$, where $h = 6.626 \times 10^{-34}$ J $\cdot$ s. Waves with frequencies ν propagate in quanta of energy $E = h\nu$. Supposing that we could determine the momentum of a particle exactly, it would end up having an equal probability of being everywhere in the Universe, behaving like a plane wave.

In the general case, we could know only the probabilities (or probability densities) of the different possible values of **observables** (the measurable physical quantities). The whole information we can know about a quantum system is given by its **quantum state** $|\psi\rangle$, which, for one quantum particle can be represented as a **wave function** $\psi(\vec{r}, t)$ evolving with time according to a partial differential equation making ψ propagate as a wave.

3.1 Hilbert space

All quantum states are elements of a vector space H, equipped with an inner product, making it a **Hilbert space**. In the Dirac notation, vectors in H are designated as $|a\rangle$ ("ket" vector) and $\langle a|$ ("bra" vector) designates the linear map from any vector $|b\rangle$ to the inner product between $|a\rangle$ and $|b\rangle$. The inner product is written as $\langle a|b\rangle$ ("bra-ket") and

obeys the following axioms:

$$\begin{aligned}
&\forall\,|a\rangle\,,|b\rangle \in H, \forall\lambda \in \mathbb{C} \Rightarrow \\
&\langle b|a\rangle = \langle a|b\rangle^*, \\
&\langle a+b|c\rangle = \langle a|c\rangle + \langle b|c\rangle, \\
&\langle a|\lambda b\rangle = \lambda\langle a|b\rangle, \\
&\langle a|a\rangle \geq 0, \langle a|a\rangle = 0 \iff |a\rangle \equiv |0\rangle\,.
\end{aligned} \tag{3.2}$$

Here, the symbol * stands for complex conjugation. The result of the inner multiplication is a scalar, meaning that it does not depend on the representation of the vectors in the Hilbert space. The norm of a vector in H is $|a| = \sqrt{\langle a|a\rangle}$.

In the **coordinate representation**, the state is represented by a complex function $\psi(\vec{r},t)$ called a wave function. The set of these wave functions can become a Hilbert space if we define the inner product between two wave functions as $\langle\phi_1|\phi_2\rangle = \int_V \phi_1^*(\vec{r})\phi_2(\vec{r})\mathrm{d}^3r$.

The probability density of a particle with wave function ψ to be found in a point $\vec{r}$ is $|\psi(\vec{r})|^2$. A plane wave with frequency ν and wavelength λ moving in the $\vec{n}$-direction has the form

$$\psi(\vec{r},t) = Ce^{-i2\pi\nu t + i\frac{2\pi}{\lambda}\vec{n}\cdot\vec{r}}, \tag{3.3}$$

where C is a normalization constant. If we substitute $2\pi\nu = 2\pi E/h = E/\hbar$ and $2\pi/\lambda\vec{n} = 2\pi|\vec{p}|\vec{n}/h = \vec{p}/\hbar$ in the above equation, we get

$$\psi(\vec{r},t) = Ce^{-\frac{i}{\hbar}Et + \frac{i}{\hbar}\vec{p}.\vec{r}}, \tag{3.4}$$

where $\hbar = h/2\pi$ is called the **reduced Planck constant**.

This is the wave function of a free particle with energy E and momentum $\vec{p}$, having an infinite uncertainty of the coordinates, as $|\psi|^2 = \text{const}$.

3.1.1 Operators and states

From this wave function, we could extract the energy and momentum by taking its derivatives with respect to time and coordinates:

$$\begin{aligned}
i\hbar\frac{\partial}{\partial t}\psi(\vec{r},t) &= i\hbar\left(-\frac{i}{\hbar}E\right)Ce^{-\frac{i}{\hbar}Et+\frac{i}{\hbar}\vec{p}.\vec{r}} = E\psi(\vec{r},t), \\
-i\hbar\vec{\nabla}\psi(\vec{r},t) &= -i\hbar\left(\frac{i}{\hbar}\vec{p}\right)Ce^{-\frac{i}{\hbar}Et+\frac{i}{\hbar}\vec{p}.\vec{r}} = \vec{p}\psi(\vec{r},t).
\end{aligned} \tag{3.5}$$

This means that the plane waves are eigenvectors of the linear operators $i\hbar\frac{\partial}{\partial t}$ and $-i\hbar\vec{\nabla}$ with eigenvalues E and $\vec{p}$. These operators are respectively called energy and momentum operators. In the coordinate basis,

$$\hat{E} = i\hbar\frac{\partial}{\partial t}, \qquad \hat{\vec{p}} = -i\hbar\vec{\nabla}. \tag{3.6}$$

Analogously to $\hat{E}$ and $\hat{\vec{p}}$, for any observable A, there is a corresponding operator $\hat{A}$. The **eigenfunctions** of $\hat{A}$ are states $|a\rangle$ where the physical quantity has a definite value. The **eigenvalue** of $\hat{A}$ corresponding to state $|a\rangle$ is the value α of the physical quantity.

The operator $\hat{\vec{r}}$ corresponding to the coordinates is simply multiplication by the coordinate. Indeed, if we knew the exact coordinates $\vec{r}_0$ of a particle, its probability density of being in $\vec{r}_0$ would be infinite, and outside of $\vec{r}_0$, the probability density is 0. Then, the wave function corresponding to $|\vec{r}_0\rangle$ in coordinate representation is $\delta^3(\vec{r}-\vec{r}_0)$ and the action of the operator on $|r_0\rangle$ would be

$$\vec{r}\delta(\vec{r}-\vec{r}_0) = \vec{r}_0\delta(\vec{r}-\vec{r}_0) \Rightarrow \hat{\vec{r}}|\vec{r}_0\rangle = \vec{r}_0\,|\vec{r}_0\rangle\,. \tag{3.7}$$

The expectation value of a random variable with probability density function $f(x)$, is $\mathbb{E}[X] = \int_{-\infty}^{\infty} xf(x)dx$. Therefore, the expected position of a particle in a quantum state $|\psi\rangle$ is

$$\mathbb{E}[\vec{r}] = \int_V \vec{r}|\psi|^2 d^3\vec{r} = \int_V \psi^*\vec{r}\psi d^3\vec{r} = \langle\psi|\hat{\vec{r}}|\psi\rangle. \tag{3.8}$$

In this notation, $\hat{\vec{r}}$ acts on $|\psi\rangle$, and $\langle\psi|$ acts on the result. Note that by using the bra-ket notation, we get a formula that will be valid in any basis.

The wave function itself can be derived from the ket-vector as $\psi(\vec{r}) = \langle\vec{r}|\psi\rangle$. Indeed, in coordinate representation, $\langle\vec{r}|\psi\rangle = \int_V \delta^3(\vec{r}'-\vec{r})\psi(\vec{r}')d^3r' = \psi(\vec{r})$. Therefore, the probability density of the particle being in $\vec{r}$ is $|\langle\vec{r}|\psi\rangle|^2$.

All quantum states are **normalized**, meaning that $\langle\psi|\psi\rangle = 1$. That is because the total probability to observe a particle somewhere is $1 = \int_V |\psi(\vec{r})|^2 d^3r = \langle\psi|\psi\rangle$.

All operators in quantum mechanics which correspond to observables are **self-adjoint**. The **adjoint** of the operator A is an operator $A^\dagger$ such that $\forall\,|\phi_1\rangle\,, |\phi_2\rangle \in H \Rightarrow \langle\phi_1|\hat{A}^\dagger|\phi_2\rangle = \langle\hat{A}\phi_1|\phi_2\rangle$. If the Hilbert space is finite-dimensional, the operator can be represented as a matrix, and $A^\dagger_{ij} = A^*_{ji}$ – the adjoint is the transposed and complex conjugated matrix. An operator is self-adjoint if $\hat{A}^\dagger = \hat{A}$.

It can be proven that all self-adjoint operators have *real* eigenvalues, which is important because all physical quantities must have real values. It can also be proven that eigenvectors corresponding to different eigenvalues are *orthogonal* and for every self-adjoint operator, we can use its eigenvectors to construct an orthonormal *basis* of the Hilbert space. This result is known as the spectral theorem [31]. This is important for the measurement in quantum mechanics to work properly, as described in the next section.

Another group of important operators in quantum mechanics are **unitary** operators. An operator $\hat{U}$ is unitary if $\hat{U}^\dagger\hat{U} = \hat{U}\hat{U}^\dagger = \mathbf{1}$, where $\mathbf{1}$ is the identity operator: $\mathbf{1}\,|\psi\rangle = |\psi\rangle$. In the finite-dimensional case, it is represented as a unitary matrix, $U^{-1}_{ij} = U^*_{ji}$. Unitary operators have the property of preserving the inner products between vectors (and respectively, the norm of the vectors: $\langle\hat{U}a|\hat{U}b\rangle = \langle a|b\rangle$). For example, if the Hilbert space is $\mathbb{R}^3$ (the space of the usual 3-dimensional vectors with real components), the unitary operators are the matrices of rotations and reflections.

In quantum mechanics, unitary operators can be used either for transformations between coordinate systems or as evolution operators. If we "rotate" the coordinate system using a unitary transformation, the new wave functions take the form $|\tilde{\psi}\rangle = \hat{U}\,|\psi\rangle$ and $\langle\tilde{\psi}| = \langle\psi|\,\hat{U}^\dagger$, and the operators are transformed as $\hat{\tilde{A}} = \hat{U}\hat{A}\hat{U}^\dagger$, preserving all inner products and physical quantities.

The evolution of a quantum system obeys the Schrödinger equation:

$$-i\hbar\frac{\partial}{\partial t}\,|\psi\rangle = \hat{H}\,|\psi\rangle\,, \tag{3.9}$$

where $\hat{H}$ is a self-adjoint operator called the **Hamiltonian**. We can evolve the wave function from an initial state $|\psi(t_0)\rangle$ to a state $|\psi(t)\rangle$ using the **evolution operator** $\hat{U}(t, t_0)$:

$$|\psi(t)\rangle = \hat{U}(t, t_0)\,|\psi(t_0)\rangle\,. \tag{3.10}$$

The Schrödinger equation will be satisfied if the evolution operator obeys the equation

$$-i\hbar\frac{\partial}{\partial t}\hat{U} = \hat{H}\hat{U}, \tag{3.11}$$

with the initial condition

$$\hat{U}(t_0, t_0) = \mathbf{1}. \tag{3.12}$$

If the Hamiltonian does not depend on time, this equation can be formally solved as

$$\hat{U} = e^{-\frac{i}{\hbar}\hat{H}t}, \tag{3.13}$$

which means that the evolution operator is indeed unitary, as $\hat{U}^{-1} = e^{i\hat{H}t/\hbar} = U^\dagger$. In the time-dependent case, one can arrive at a formal solution as a time-ordered exponential, which is again unitary [32].

3.1.2 Fidelities and measurement

In classical mechanics, if two waves interfere, the new wave will be the superposition of the two original waves. Let the displacements of the original waves at a certain point be $u_1 = A_1\cos(\omega t + \phi_1)$ and $u_2 = A_2\cos(\omega t + \phi_2)$. The intensity of the original waves is proportional to $\bar{u}_1^2 \sim A_1^2$ and $\bar{u}_2^2 \sim A_2^2$ respectively. However, the resulting wave has intensity, proportional to $\bar{u}^2 = A_1^2 + A_2^2 + 2A_1A_2\cos(\phi_1 - \phi_2)$ – instead of simply adding the two intensities, there also appears an interference term.

Similarly, quantum interference is experimentally observed. A quantum state $|\psi\rangle$ can be a superposition of some orthonormal quantum states $|\phi_1\rangle$ and $|\phi_2\rangle$: $|\psi\rangle = c_1\,|\phi_1\rangle + c_2\,|\phi_2\rangle$, $c_1, c_2 \in \mathbb{C}$. For the new state to be normalized, $1 = \langle\psi|\psi\rangle = \langle c_1\phi_1 + c_2\phi_2|c_1\phi_1 + c_2\phi_2\rangle = |c_1|^2 + |c_2|^2$. However, if a particle is in a superposition of two such states, the probability of observing the particle in a certain position will be

$$|\psi(\vec{r})|^2 = (c_1^*\phi_1^* + c_2^*\phi_2^*)(c_1\phi_1 + c_2\phi_2) = |c_1|^2|\phi_1|^2 + |c_2|^2|\phi_2|^2 + 2\mathrm{Re}(c_1c_2^*\phi_1(\vec{r})\phi_2^*(\vec{r})), \tag{3.14}$$

containing a sum of the probabilities of the old states with coefficients $|c_1|^2$ and $|c_2|^2$, which one may intuitively expect, as well as an interference term.

The probabilities of observing all other physical quantities follow a similar pattern. We saw that the probability density of the particle being in $\vec{r}$ is $|\langle\vec{r}|\psi\rangle|^2$, where $|\vec{r}\rangle$ is the eigenvector corresponding to the eigenvalue $\vec{r}$. For all observables A with operator $\hat{A}$, the probability to measure a value α for a quantum system in state $|\psi\rangle$, is $|\langle a|\psi\rangle|^2$, where $\hat{A}\,|a\rangle = \alpha\,|a\rangle$ (if $\hat{A}$ has a discrete spectrum, for operators with continuous spectrum such as $\hat{\vec{r}}$, this would be the probability density). As $\hat{A}$ is self-adjoint, this rule is consistent with the fact that if a system is in a state with eigenvalue α_1, there is no probability to observe another eigenvalue α_2 (its eigenvectors are orthonormal to the eigenvectors of α_1). Also, for all normalized states, the sum of all probabilities adds up to 1.

Similar to the coordinates of a particle, the expectation value of any observable $\hat{A}$ of the quantum system in a state $|\psi\rangle$ is $\langle\psi|\hat{A}|\psi\rangle$.

The quantity $|\langle\psi_1|\psi_2\rangle|^2$ is called **fidelity** of the states $|\psi_1\rangle$ and $|\psi_2\rangle$. The fidelity can be thought of as the degree of overlap or similarity between the two states. As the quantum states are normalized, the fidelity can range from 0 to 1. If the two states are orthogonal, $\langle\psi_1|\psi_2\rangle = 0$. If the states are identical, $\langle\psi_1|\psi_1\rangle = 1$. In the general case, $|\psi_2\rangle = c_1 |\psi_1\rangle + c_\perp |\psi_\perp\rangle \Rightarrow |c_1|^2 + |c_\perp|^2 = 1 \Rightarrow |\langle\psi_1|\psi_2\rangle|^2 = |c_1|^2 \leq 1$. If one of the states is an eigenstate of a certain observable, the fidelity gives us exactly the probability of measuring the corresponding eigenvalue of the observable.

After measuring an observable A, we have $|\langle a|\psi\rangle|^2$ probability of measuring α, where $\hat{A}|a\rangle = \alpha|a\rangle$. If the result of the measurement is α, the quantum system will **collapse** from $|\psi\rangle$ to the state $|a\rangle$. Then, any subsequent measurement of A will yield the same value α.

The measurement and collapse of the wave function are important concepts in quantum mechanics. We should note that performing a measurement in quantum mechanics means that the quantum system interacts with the outside world. This means that firstly, if we just think of a value of A and pretend that we measured the system to get this value, this would not constitute a measurement and will not cause a collapse of the wave function. Secondly, any interactions with the environment can lead to collapses if they are not part of an experimental setup created with the purpose of measuring the system.

3.2 Two-level system

The simplest possible quantum system is the two-level system (**qubit**). Its states are elements of a 2-dimensional complex Hilbert space. The "two levels" are an orthonormal basis containing two elements, which we will call $|0\rangle$ and $|1\rangle$. All other physical states can be represented as

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \tag{3.15}$$

where $c_0, c_1 \in \mathbb{C}$, and $\langle\psi|\psi\rangle = 1 \Rightarrow |c_0|^2 + |c_1|^2 = 1$.

Multiplying all kets by a global phase $e^{i\alpha}$ does not impact any physical quantities. Starting from the space of all complex tuples $\mathbb{C}^2$ and considering the restrictions that $|c_0|^2 + |c_1|^2 = 1$ and that (c_0, c_1) is identical to $(c_0 e^{i\alpha}, c_1 e^{i\alpha})$, the space of all distinct quantum states ends up being the complex projective space $\mathbb{CP}^2$.

Two-level systems describe vastly different cases. The spin of all spin-1/2 fermions is a two-level system with basis $|\uparrow\rangle$, $|\downarrow\rangle$ – "spin up" and "spin down". The polarization of light can be decomposed to basis states $|x\rangle$ and $|y\rangle$ – horizontal and vertical polarization. Qubits are the building blocks of quantum computers. Some quantum systems which are commonly used in quantum computers are trapped ions and superconducting qubits. We should note that such systems are not naturally occurring two-level systems. On their own, they would have multiple states, but pair of states is isolated artificially to form a two-level system.

3.2.1 Bloch sphere representation

There is a one-to-one correspondence between the complex projective space of the qubit states $\mathbb{CP}^2$ and the points on a sphere S^2. Consider a sphere with a center in the origin of the coordinate system. A point on the sphere is defined by its position vector $\vec{n}$. It can

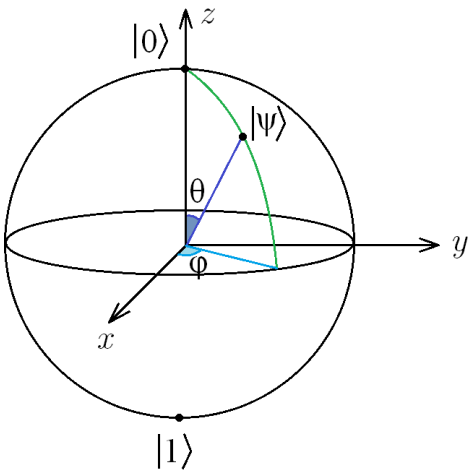

Figure 3.1: Bloch sphere used for representation of a two-level quantum system. The north and south pole of the Bloch sphere are the states $|0\rangle$ and $|1\rangle$ of the system, the rest of the points are superposions of these states. θ is measured with respect to the north pole, and ϕ - along the equator, starting from x and moving anticlockwise.

be described using two spherical coordinates (θ, ϕ). The polar angle θ is defined as the angle between the "north pole" (the z-axis) and P. The azimuthal angle ϕ is the angle between the x-axis and the projection of P on the "equator" (the circle in the xy-plane). With these coordinates,

$$\vec{n} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}. \tag{3.16}$$

The tuple $(c_1, c_2) = (\cos(\theta/2), e^{i\phi}\sin(\theta/2))$ satisfies the condition $|c_1|^2 + |c_2|^2 = 1$, making it a unique element of $\mathbb{CP}^2$. Moreover, every tuple $(c_1, c_2) \in \mathbb{CP}^2$ is equivalent to a tuple $(\tilde{c}_1, \tilde{c}_2) = e^{i\alpha}(c_1, c_2)$ where α can be chosen in such a way as to make $\tilde{c}_1$ a real number. Then, $\tilde{c}_2$ is a complex number with a certain polar form $\tilde{c}_2 = \tilde{\tilde{c}}_2 e^{i\phi}$ and $\tilde{c}_1^2 + \tilde{\tilde{c}}_2^2 = 1$, which means that we can write them in the form $\tilde{c}_1 = \sin(\theta/2)$ and $\tilde{\tilde{c}}_2 = \sin(\theta/2)$. This way we proved there is a one-to-one correspondence between $\mathbb{CP}^2$ and S^2.

The sphere S^2 which we can represent the qubit states on, is called a Bloch sphere. Using the above parametrization,

$$|\psi\rangle = e^{i\alpha}\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i(\alpha+\phi)}\sin\left(\frac{\theta}{2}\right)|1\rangle = e^{i\alpha}\begin{pmatrix} \cos(\theta/2) \\ e^{i\phi}\sin(\theta/2) \end{pmatrix}. \tag{3.17}$$

Here, we represented $|\psi\rangle$ as a column vector, where the basis vectors take the form

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{3.18}$$

On the Bloch sphere, $|0\rangle$ and $|1\rangle$ are positioned on the north and south pole, respectively. Note that although they look like they are pointing in "opposite" directions on the Bloch sphere, they are in fact orthogonal in the Hilbert space – their fidelity is $|\langle 0|1\rangle|^2 = 0$. The point on the x-axis, having state $|x\rangle$, looks like it is in the "perpendicular" direction with respect to $|0\rangle$, but its state is $|x\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, having fidelity $|\langle x|0\rangle|^2 = 1/2$ with $|0\rangle$.

3.2.2 Pauli matrices

Suppose our qubit is the spin state of a spin-1/2 particle. In this case, the points on the Bloch sphere are pointing in the "direction" of the spin of the particle. $|0\rangle = |\uparrow\rangle$ and $|1\rangle = |\downarrow\rangle$ are the "spin up" and "spin down" states. Each spin component s_x, s_y, s_z has an operator $\hat{s}_x, \hat{s}_y, \hat{s}_z$. The eigenvectors of $\hat{s}_x, \hat{s}_y, \hat{s}_z$ should be the states where the spin is pointing in the $\pm x$, $\pm y$, and $\pm z$ directions respectively. Their eigenvalues should be $\pm 1/2$ depending on the direction of the spin. In the basis of $|\uparrow\rangle$ and $|\downarrow\rangle$, these spin states have the form

$$\begin{aligned} |+x\rangle &= \frac{\sqrt{2}}{2}\begin{pmatrix}1\\1\end{pmatrix}, \quad |-x\rangle = \frac{\sqrt{2}}{2}\begin{pmatrix}1\\-1\end{pmatrix}, \\ |+y\rangle &= \frac{\sqrt{2}}{2}\begin{pmatrix}1\\i\end{pmatrix}, \quad |-y\rangle = \frac{\sqrt{2}}{2}\begin{pmatrix}1\\-i\end{pmatrix}, \\ |+z\rangle &= \begin{pmatrix}1\\0\end{pmatrix}, \qquad |-z\rangle = \begin{pmatrix}0\\1\end{pmatrix}. \end{aligned} \tag{3.19}$$

Therefore the operators of the spin components in this basis will be the matrices

$$\hat{s}_x = \frac{1}{2}\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad \hat{s}_y = \frac{1}{2}\begin{pmatrix}0 & -i\\i & 0\end{pmatrix}, \quad \hat{s}_z = \frac{1}{2}\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}, \tag{3.20}$$

where the matrices

$$\hat{\sigma}_x = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad \hat{\sigma}_y = \begin{pmatrix}0 & -i\\i & 0\end{pmatrix}, \quad \hat{\sigma}_z = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}, \tag{3.21}$$

are called Pauli matrices. We could combine the operators of the three spin components to a vector operator $\hat{\vec{s}}$, and we could combine the Pauli matrices into a Pauli vector $\hat{\vec{\sigma}}$.

The spin operators are self-adjoint: $\hat{\vec{s}}^\dagger = \hat{\vec{s}}$ and $\hat{\vec{\sigma}}^\dagger = \hat{\vec{\sigma}}$. The Pauli matrices are also unitary: $\hat{\sigma}_x\sigma_x^\dagger = \hat{\sigma}_y\sigma_y^\dagger = \hat{\sigma}_z\sigma_z^\dagger = \mathbf{1}$.

Spin magnitude and "direction"

Let us have an arbitrary spin state $|\psi\rangle$, whose Bloch sphere representation has spherical coordinates θ and ϕ and position vector $\vec{n}$. The component of the Pauli vector $\hat{\vec{\sigma}}$ in the direction of $\vec{n}$ is $\hat{\sigma}_n = \hat{\vec{\sigma}} \cdot \vec{n}$. If we act with this operator on $|\psi\rangle$ in the $|\uparrow\rangle, |\downarrow\rangle$ basis (see Eqs. (3.21),(3.16),(3.17)), we get

$$\begin{aligned} \hat{\vec{\sigma}} \cdot \vec{n}\,|\psi\rangle &= \left(\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \begin{pmatrix}0 & -i\\i & 0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}\right) \cdot \begin{pmatrix}\sin\theta\cos\phi\\\sin\theta\sin\phi\\\cos\theta\end{pmatrix} \begin{pmatrix}\cos\frac{\theta}{2}\\e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix} = \\ &= \begin{pmatrix}\cos\theta & \sin\theta(\cos\phi - i\sin\phi)\\\sin\theta(\cos\phi + i\sin\phi) & -\cos\theta\end{pmatrix} \begin{pmatrix}\cos\frac{\theta}{2}\\e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix} = \\ &= \begin{pmatrix}\cos\theta & \sin\theta e^{-i\phi}\\\sin\theta e^{i\phi} & -\cos\theta\end{pmatrix} \begin{pmatrix}\cos\frac{\theta}{2}\\e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix} = \\ &= \begin{pmatrix}\cos\theta\cos\frac{\theta}{2} + \sin\theta\sin\frac{\theta}{2}\\\sin\theta\cos\frac{\theta}{2}e^{i\phi} - \cos\theta\sin\frac{\theta}{2}e^{i\phi}\end{pmatrix} = |\psi\rangle\,. \end{aligned} \tag{3.22}$$

This spin state is an eigenstate to the $\hat{\sigma}_n$ operator, so the component of the spin in the direction of $\vec{n}$ has an exact value of 1/2. By illustrating the quantum spin on the Bloch

sphere, it appears like the spin has a definite direction. However, if we try to measure a different component of the spin, we could not tell with absolute certainty what the result will be, and the result will again be one of the two options $1/2$ and $-1/2$. For example, if we measure $\hat{s}_z$, we have probability $|\langle\uparrow|\psi\rangle|^2 = \cos^2\theta/2$ to measure spin up, and $|\langle\downarrow|\psi\rangle|^2 = \sin^2\theta/2$ to measure spin down. The expectation value of the spin is $\langle\psi|\hat{s}_z|\psi\rangle = 1/2(\cos^2\theta/2 - \sin^2\theta/2) = 1/2\cos\theta$, which means that by looking at the expectation values, the spin still looks like a vector pointing at a direction θ with respect to the z-axis.

Quantum gates

Suppose our two-level system is a qubit used in a quantum computer. To manipulate a qubit, the quantum circuit can perform unitary operators to the qubit state, which are called **quantum gates**. They are unitary in order to preserve the norms of all qubit states. Some examples of gates used in quantum circuits for single qubits are the Pauli matrices themselves. Another example is the phase shift gate

$$\hat{P}(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}, \tag{3.23}$$

which as a transformation on the Bloch sphere looks like a rotation of the azimuthal angle. The Hadamard gate

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.24}$$

performs a π-rotation around the point $\phi = 0, \theta = \pi/4$, transforming $|0\rangle \leftrightarrow |x\rangle$, $|1\rangle \leftrightarrow |-x\rangle$, and $|y\rangle \leftrightarrow |-y\rangle$. This gate has applications in different quantum computing algorithms. However, we will not use it elsewhere in the thesis, so the symbol $\hat{H}$ will be used for the Hamiltonian.

In Chap. 5.2, we will be using **rotation operator gates**, which are the exponentials of the Pauli matrices:

$$\hat{R}_x(\theta) = e^{-i\sigma_x\theta/2}, \hat{R}_y(\theta) = e^{-i\sigma_y\theta/2}, \hat{R}_z(\theta) = e^{-i\sigma_z\theta/2}. \tag{3.25}$$

We can calculate the explicit form of these matrices:

$$\begin{aligned} \hat{R}_k &= \exp\left(-i\frac{\theta}{2}\hat{\sigma}_k\right) = \sum_{n=0}^{\infty} \frac{-i^n\theta^n\hat{\sigma}_k^n}{2^n n!} = \\ &= \sum_{n=0}^{\infty} \frac{(-1)^n\theta^{2n}\mathbf{1}}{2^{2n}(2n)!} + \sum_{n=0}^{\infty} -i\frac{(-1)^n\theta^{2n+1}\hat{\sigma}_k}{2^{2n+1}(2n+1)!} = \\ &= \cos\left(\frac{\theta}{2}\right)\mathbf{1} - i\sin\left(\frac{\theta}{2}\right)\hat{\sigma}_k, \end{aligned} \tag{3.26}$$

where we have used that

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}; \quad \cos x = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}; \quad \sin x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}; \quad \sigma_k^2 = \mathbf{1}. \tag{3.27}$$

In the three cases (x, y, z), we obtain

$$\begin{aligned}
\hat{R}_x(\theta) &= \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \\
\hat{R}_y(\theta) &= \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \\
\hat{R}_z(\theta) &= \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix},
\end{aligned} \tag{3.28}$$

In Chap. 5.2, we will denote the inverse matrices as $\hat{R}_{-x}(\theta) = \hat{R}_x^{-1}(\theta) = \hat{R}_x(-\theta)$, $\hat{R}_{-y} = \hat{R}_y^{-1}$ and $\hat{R}_{-z} = \hat{R}_z^{-1}$.

Spinors On the Bloch sphere, these matrices perform rotations around the x, y, and z-axis, respectively. The parameter θ in $\hat{R}(\theta)$ corresponds exactly to the angle of rotation on the Bloch sphere. However, we can notice that these matrices, which are in the basis of $|0\rangle$ and$|1\rangle$, contain trigonometric functions with argument $\theta/2$, which looks like they perform only half of the necessary rotation. This is due to the fact that the spin states $|\psi\rangle$ are not vectors in the physical sense, but are instead **spinors**. If we apply a 2π rotation on such a spinor, the result would be

$$\hat{R}_k(2\pi)\,|\psi\rangle = \cos\pi\,|\psi\rangle - i\sin\pi\hat{\sigma}_k\,|\psi\rangle = -\,|\psi\rangle\,, \tag{3.29}$$

meaning that the spinor has transformed into the "opposite" spinor. In fact, the spinor acquired a global phase π, which does not change its observables.

Chapter 4

Reinforcement Learning Overview

4.1 Reinforcement Learning Framework

Machine learning is useful in optimization problems such as dynamic control. Its benefits compared to traditional control algorithms are the ability to generalize and therefore behave properly in different settings, and the ability to perform well in noisy environments [33]. We will introduce the basic terminology required to translate a dynamical control problem into a Reinforcement Learning (RL) problem.

In RL, the physical system we need to control is interpreted as an **environment**. The state of the physical system is called a **state** of the environment. The environment can proceed to different states based on the way we control it. In the RL language, there are different possible **actions** that the agent can take which would lead the environment to different states. The RL **agent** is a program that receives the state as an input and produces an action as an output according to a **policy**. The policy can be thought of as a strategy followed by the agent. The actions should be chosen in such a way that the physical system will exhibit certain behavior that would be of interest to us. In order to learn, the agent should also receive feedback in the form of a **reward**. We should choose the reward so that it would be maximal when the environment is close to the desired goal. During **training**, the agent updates its **policy** in such a way as to receive higher rewards [34]. This feedback loop is shown in Fig. 4.1. We will now give more strict definitions of these terms.

4.1.1 States, actions, rewards

The environment consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, reward space $\mathcal{R}$, and a set of rules specifying the state dynamics. The state space $\mathcal{S}$ is the set of all the possible states, where a state s_t comprises the values of all degrees of freedom of the system at a certain moment of time. For example, in a classical dynamical system, the state of the environment would be $(q_i(t), p_i(t))$. In a quantum system, it could be a representation of the quantum state $|\psi\rangle$, or if we do not possess the full information about the quantum system, it would be the density matrix.

The action space $\mathcal{A}$ consists of all the possible actions a which can be taken. The action

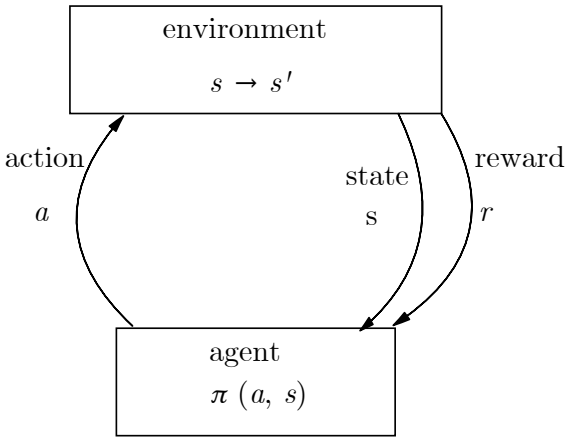

Figure 4.1: Typical feedback loop in Reinforcement learning. The RL-agents provides actions according to its policy, changing the state of the environment. The environment provides rewards so that the agent would refine its policy.

determines the probability

$$P(s'|s,a) \tag{4.1}$$

of entering a new state $s_{t+1} = s'$. The **Markov property** states that this probability can depend only on the previous state $s_t = s$ and the action $a_t = a$ taken at that time step.

The reward

$$r_{t+1} = r(s_t, a_t, s_{t+1}) \tag{4.2}$$

is a function of the current state of the environment (in the general case of a Markov Decision Process, it can also depend on the previous state and action). The reward function r is usually not inherent to a physical system. We add it by hand in a way that corresponds to our goal – when the system is near the desired state, the rewards should be higher.

The environment is interacted with during an interval of time called an **episode**. The episode is discretized in a finite number of time steps labeled by t. The total number of time steps T is called the **length** of the episode. The sequence

$$\tau = (s_0, a_0, s_1, a_1, ...s_T) \tag{4.3}$$

is called a **trajectory**.

Such a control process, containing the states $\mathcal{S}$, actions $\mathcal{A}$, rewards $\mathcal{R}$, and dynamics determined by the probabilities $P(s'|s,a)$, is called a Markov decision process (Fig. 4.2).

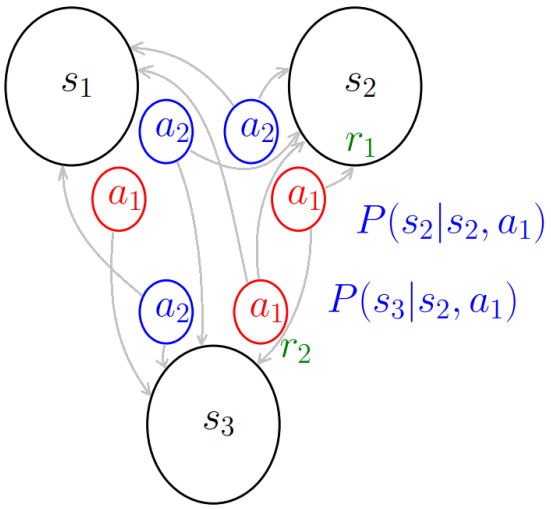

Figure 4.2: Structure of the Markov Decision Process. The action and current state determine the probabilities of entering the next state. Depending on the state the environment entered, it receives a certain reward.

4.1.2 Returns and policy

The agent will be trying to maximize the sum of all rewards it expects to receive after its action. Therefore, it is useful to define the **return** as

$$R_t = \sum_{t'=t}^{T} r_{t'+1}. \tag{4.4}$$

As we will set the environment in an **initial state** at $t = 0$ and expect the agent to obtain maximal rewards during the whole episode, the performance can be evaluated by the return at $t = 0$, which is often just called return. It is a function of all future states and actions and constitutes an average over the policy and the transition probability.

The agent is a program that produces actions according to a policy and updates the policy according to the received rewards.

The policy π is a function that inputs the environment state and outputs a probability distribution of the different actions. We will denote $\pi(a_t|s_t)$ the probability of choosing action a_t if the environment is in state s_t. The policy is optimal if it reaches the maximum of the expected return. In order to formalize this statement, we need a few more definitions.

The **discounted return** is

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k. \tag{4.5}$$

The discounting factor $\gamma \leq 1$ allows us to use RL in situations where the episode is arbitrarily long. The case $\gamma = 1$ corresponds to the usual return.

The **value function** is

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|S_t = s, A_t = a\right]. \tag{4.6}$$

This is the expected (discounted) return which would be achieved by using policy π starting from state s at time t. It is a function of the *state* we are in, so it is often called a **state-value function** [34, 35]. The idea is that $v_\pi(s)$ would be higher for states which are near the goal since staying near the goal would give us higher rewards.

Action-value function is

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|S_t = s, A_t = a\right]. \tag{4.7}$$

This is the expected return achieved by π if we start at s and take action a. We emphasize that it depends on the action. If an agent knows $q_\pi(s,a)$ perfectly for all actions in a given state, then it would know the best action from the given state, which would be the action with maximal q_π. If we have a discrete number of states and actions, then for a fixed strategy, v_π can be represented as a vector of the expected returns of all different states, and q_π – as a matrix of the expected returns of all states, taking all actions.

The **optimal policy** π_* is the one that achieves the highest possible expected return among all policies, for each initial state $s \in \mathcal{S}$. Then, the state-value function is the **optimal state-value function**

$$v_*(s) = \max_\pi v_\pi(s), \tag{4.8}$$

and the action-value function is the **optimal action-value function**

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{4.9}$$

for all $s \in \mathcal{S}, a \in \mathcal{A}$.

4.2 Neural Networks

Suppose we have to teach an agent to play a video game. The game is a complex environment whose state is a collection of pixels and its reward is the score of the game. In the ideal case, the agent would have infinite memory and time to try all possible actions and eventually construct a multiple-dimensional vector $v_\pi(s)$ containing the expected total score that the player would receive after settling on each of the states s. It would also construct a multi-dimensional matrix $q_\pi(s,a)$ with the expected scores achieved by taking all different possible actions. And it would construct a multi-dimensional matrix $\pi(a,s)$ containing probabilities for taking each possible action from each possible state. This would be very ineffective and infeasible in practice. Firstly, even if this discrete state space, new dimensions (e.g. a pixels) lead to exponential growth of the state space. This means that in practice, it would be impossible to visit the whole state space in order to calculate the whole state-value and action-value matrices. Moreover, in a continuous setting such as the phase space of a classical system or the Hilbert space in the case of a quantum system, all possible states would be infinite, making this approach even more absurd.

Hence, we need to approximate the policy (alternatively, the value function) using some type of a variational ansatz. We cannot just assume linear dependence between π, a, and s, since in practice the relation between the states, returns, and optimal actions is

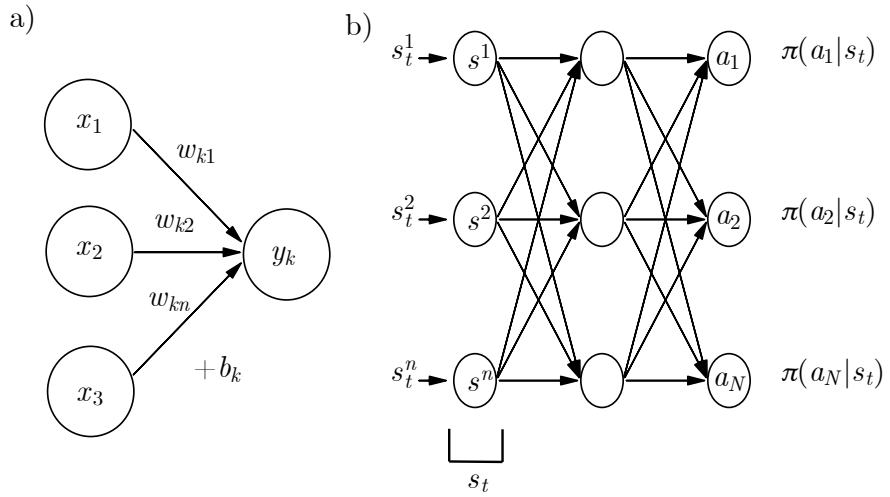

Figure 4.3: a) A single neuron. The outputs from the previous layers x_j are inputs of the neurons in the current layer. b) The policy $\pi(a|s)$ can be approximated with a neural network. Its input is a representation of the state s_t and the output is a vector of probabilities $\pi(a|s_t)$ for all actions a.

nonlinear. In order to approximate the complicated function $\pi(a, s)$ efficiently, we can use neural networks. Neural networks are known to be universal approximators [36].

A fully-connected neural network, as seen in Fig. 4.3b), contains multiple **layers**, each layer containing multiple **neurons**. The number of layers in the network is called **depth**. The number of neurons in a layer is called **width** of the layer. A neuron is a simple function of multiple parameters. For the k-th neuron in an arbitrary layer, this function has the form

$$y_k = \phi\left(\sum_{j=0}^{m} w_{kj}x_j + b_k\right). \tag{4.10}$$

where y_k is the output of the k-th neuron in a layer. It is produced from the input values x_j which are the outputs of the previous layer neurons (see also Fig. 4.3b). The width of the previous layers is m. The parameters w_{kj} and b_k are called **weights** and **biases**, on which ultimately the policy depends. The function ϕ is nonlinear, which is necessary in order to have any benefit in including multiple layers [37].

The most often used nonlinear functions ϕ are the logistic function $\phi(x) = (1 + e^{-x})^{-1}$ and the rectified linear unit (RELU) $\phi(x) = \max(0, x)$. Also, if we want the final layer to give us probabilities for different actions, it needs to have such function as to guarantee that the sum of all outputs will be equal to unity, and that no probabilities are negative. This is achieved by a softmax layer,

$$\sigma(x_1, ...x_m)_i = \frac{e^{x_i}}{\sum_{j=1}^{m} e^{x_j}}. \tag{4.11}$$

As some actions will gradually be discarded, their probabilities should become very small. The softmax layer will keep track of the logarithms of the probabilities, making it suitable for retaining small probabilities with good precision.

In the most general case, the neural network has an input vector $\mathbf{x}$ and an output vector $\mathbf{y} = \mathbf{y}(\mathbf{x}, \theta)$, where θ are all of the parameters (weights and biases) of the neural network. In order for the neural network to learn, we need to specify a **cost function** $C(\theta)$ that quantifies the performance of the neural network. We will then try to minimize it using gradient descent,

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta C(\theta_t), \tag{4.12}$$

where η is a small parameter. After starting from a random distribution of the parameters θ, we would slowly "descend" to the minimum of the cost function.

Typical gradient descent algorithms, called "optimizers", are Stochastic Gradient Descent (SGD), the Momentum method and Adam. SGD will always update the parameters θ in the direction of the current gradient using the same step size. By using momentum, gradients at past timesteps are also taken into account in an update of the parameters. This reduces the variance of the updates. The Adam algorithm also uses a second moment of the gradients in order to adapt its learning rate, leading to better convergence.

Neural networks allow gradients to be taken computationally easily. For more detailed explanations on gradient descent in neural networks we refer the reader to [36].

Ideally, the cost function depends on all different possible inputs and desired outputs. However, that would not be computable. In practice, the cost function is approximated using only a **batch** (a small number) of different inputs. If we have a measure of the performance of a single input-output pair $c_i = c_i(\mathbf{y}_{i,\text{goal}}, \mathbf{x}_i)$, then

$$C \propto \sum_{i \in \text{batch}} c_i. \tag{4.13}$$

This allows faster computation of the gradients. In order to use the information from a maximum number of input-output pairs, each update $\theta_t \to \theta_{t+1}$ is made using a different batch. Using batches has other advantages such as serving as regularization. We refer interested readers to [37].

4.3 Policy Gradient

There are different RL algorithms to find the optimal policy π_*. The common idea is that the ansatz policy is a function of a lot of parameters θ. The parameters can be varied during training to different values $\theta \to \theta'$ in such a way as to achieve a better policy π' (in the sense that the value function is higher) until gradually reaching the optimal policy.

We will now consider one specific instance of a Reinforcement Learning (RL) algorithm, Policy Gradient (PG), as we will be using it in our simulations. Some RL algorithms try to approximate the value function using neural networks. In Policy Gradient, the policy is learned directly using a neural network.

The input of the neural network is the state s, and the output is a policy π, as can be seen in Fig. 4.3. The objective that the neural network should optimize is $J(\theta) = v_\pi(\theta)$, where v is the action-value function (4.7). Note that the RL objective is to find the strategy with the *greatest* return, so if we use gradient *de*scent, the cost would correspond to $-J(\theta)$. The problem is that we do not know v_π directly, so we need to approximate it using a batch of trajectories. Note that we are not just seeking a good approximation for v_π. As we are doing optimizations with gradient descent, we are changing the policy in

the direction of its gradient. Therefore, we actually need a good approximation for the gradient of v_π.

Luckily, there is a neat way to approximate $\nabla J(\theta)$ using the Policy Gradient Theorem:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta), \tag{4.14}$$

where μ is the probability distribution of the states occurring in the trajectories if we use policy π. The proof of this theorem can be seen in Ref. [34].

4.3.1 The REINFORCE algorithm

The idea of the approximation will be to replace the expectation value over states $\sum_s \mu(s)$ with a Monte Carlo (MC) sample of the trajectories. We have

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &\propto \sum_s \mu(s) \sum_a q_\pi(s_t,a) \nabla_\theta \pi_\theta(a|s_t) && (4.15)\\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(s_t,a) \nabla_\theta \pi_\theta(a|s_t) \right] && (4.16)\\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(s_t,a) \pi_\theta(a|s_t) \frac{\nabla_\theta \pi_\theta(a|s_t)}{\pi_\theta(a|s_t)} \right] && (4.17)\\
&= \mathbb{E}[q_\pi(s_t,a_t) \nabla_\theta \ln \pi_\theta(a_t|s_t)].
\end{aligned}$$

We used that under the expectation sign, it does not matter if we use the expected action according to the policy or the actual action in the sample: $\mathbb{E}_\pi[\sum_a f(a)\pi_\theta(a|s_t,)] = \mathbb{E}_\pi[f(a_t)]$. Also, $\nabla \ln x = \dfrac{\nabla x}{x}$.

Now the expectation of the q-function can be replaced with the actual reward in the sample, and instead of taking expectation values, we can take an average:

$$\nabla_\theta J \approx \nabla_\theta \frac{1}{N_{\text{MC}}} \sum_{j=1}^{N_{\text{MC}}} \sum_{t=1}^{T} G_t(\tau^j) \ln \pi_\theta(a_t^j|s_t^j) = \nabla_\theta J'. \tag{4.18}$$

We have used that after the returns were generated, they are constant with respect to θ. Therefore, we can move the gradient in front of the averaging. In that way we now replace the real loss function $-J$ with a pseudo loss function $-J'$:

$$J' = \frac{1}{N_{\text{MC}}} \sum_{j=1}^{N_{\text{MC}}} \sum_{t=1}^{T} G_t(\tau^j) \ln \pi(a_t^j|s_t^j,\theta), \tag{4.19}$$

which has approximately the same gradient (up to a factor that can be absorbed in the step α of the gradient descent) and can be calculated directly.

This gives us the simplest working PG algorithm called REINFORCE [34]. For each episode, the agent will act with policy π on N_{MC} copies of the environment. After the environment produces the rewards r, we can calculate the returns G and update the policy by updating the parameters of the neural network, $\theta' = \theta + \alpha \nabla_\theta J'$.

Algorithm 1 Pseudocode for the REINFORCE algorithm [34].

initialize θ
for each episode **do**
  for each timestep t **do** ▷ Generate the episode
    generate action a_t following $\pi_\theta(\cdot|s_t)$
    act on the environment to get s_{t+1} and r_{t+1}
  end for
  for each timestep t **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ ▷ Calculate the returns
    $\theta \leftarrow \theta + \alpha\gamma^t \nabla_\theta \ln \pi_\theta(a_t|s_t)$ ▷ Update θ
  end for
end for

4.3.2 Adding Baseline to the REINFORCE algorithm

The REINFORCE algorithm can be further improved in several directions. First, the expectation value of the gradient would not change if we include an arbitrary function $b(s)$ called a **baseline**:

$$\sum_a b(s)\nabla_\theta \pi_\theta(a|s) = 0 \quad \Rightarrow \quad \sum_s \mu \sum_a q_\pi \nabla_\theta \pi_\theta = \sum_s \mu \sum_a (q_\pi - b)\nabla_\theta \pi_\theta. \tag{4.20}$$

This means that in the end, we could also add a baseline to the pseudo loss which would not change the expectation value of the gradients. However, we can choose b so that the variance of the gradient would be smaller. This would give better approximations of the gradients. A good practical baseline that we use in our algorithms is the averaged return over the sample $b_t = \frac{1}{N_{\mathrm{MC}}} G_t(\tau_j)$. The pseudo loss with baseline has this form:

$$J'(\theta) = \frac{1}{N_{\mathrm{MC}}} \sum_{j=1}^{N_{\mathrm{MC}}} \sum_{t=1}^{T} \ln \pi_\theta(a_t^j|s_t^j) \left(G_t(\tau_j) - b_t\right). \tag{4.21}$$

4.3.3 Regularization and Policy Entropy

The second improvement to the algorithm is adding regularization. This is necessary for stability issues – neural networks are prone to the *exploding gradients* or *vanishing gradients* problem. In our problems, we needed to combat mainly the exploding gradients problem by using L2 regularization [38].

Also, regularization is needed to increase *exploration* while learning. In order for the agent to find the optimal strategy in the long run, it has to observe various different strategies in the short run. By naively following the greatest ascent of the reward it would explore less. Suppose the agent performs with high probability relatively good (although not optimal) actions. By moving in the direction of the highest return, it will perform these actions with even higher probability until the policy converges to a delta function, meaning that the agent will always perform the same actions.

Optimizing exploration can be achieved by adding another term in the pseudo loss which is proportional to the **entropy** of the probability distribution π. By increasing the entropy of its strategy, the agent has the chance to encounter more different trajectories.

The Shannon entropy of the policy π (since π is a probability distribution) has the form

$$H(\pi_\theta(\cdot|s_t)) = -\sum_a \pi_\theta(a|s_t) \ln \pi_\theta(a|s_t). \tag{4.22}$$

Let us understand why increasing entropy leads to more exploration. Suppose we are interested in a fixed state s and we have n different possible actions $a_1, a_2, ...a_n$. An example "delta function" (also called **greedy**) strategy would be

$$\pi(a_i|s) = \delta_{i1} = \begin{cases} 1, i = 1 \\ 0, i \neq 1 \end{cases}, \tag{4.23}$$

which has entropy $H(\pi(\cdot|s)) \to 0$. However, If we perform an ε-greedy strategy, that is, having $1-\varepsilon$ probability to perform the first action and ε probability to perform a random action:

$$\pi(a_i|s) = \delta_{i1} = \begin{cases} 1 - \varepsilon\frac{n-1}{n}, i = 1 \\ \frac{\varepsilon}{n}, i \neq 1 \end{cases}, \tag{4.24}$$

it would have entropy

$$H(\pi(\cdot|s)) = -\left(1 - \varepsilon\frac{n-1}{n}\right)\ln\left(1 - \varepsilon\frac{n-1}{n}\right) - (n-1)\frac{\varepsilon}{n}\ln\left(\frac{\varepsilon}{n}\right). \tag{4.25}$$

$$\begin{aligned} \frac{\partial H}{\partial \varepsilon} &= \left(\ln\left(1 - \varepsilon\frac{n-1}{n}\right) + 1\right)\frac{n-1}{n} - \frac{n-1}{n}\left(\ln\left(\frac{\varepsilon}{n}\right) + 1\right) \\ &= \frac{n-1}{n}\ln\left(\frac{n - \varepsilon(n-1)}{\varepsilon}\right) > 0 \end{aligned} \tag{4.26}$$

when $0 < \varepsilon < 1$, which means that the entropy is always increasing if we increase the probability of choosing a random action.

Finally, if we choose each action with equal probability $\pi(a_i|s) = 1/n$, the entropy would be $H(\pi(\cdot|s)) = \ln(n)$ which is the highest possible entropy for n options.

We can add the entropy to the pseudo loss with a coefficient that would best suit the specific task. We may call this parameter **temperature**. The resulting pseudoloss will be

$$J'(\theta) = \frac{1}{N_{\text{MC}}}\sum_{j=1}^{N_{\text{MC}}}\sum_{t=1}^{T}\ln\pi(a_t^j|s_t^j,\theta)\left(G_t(\tau_j) - \beta^{-1}\sum_{t=0}^{T}\ln\pi_\theta(a_t^j|s_t^j) - b_t\right). \tag{4.27}$$

Chapter 5

Reinforcement Learning to Control Intrinsically Nonadiabatic Dynamical Systems

In this chapter, we apply Reinforcement Learning (RL), which we introduced in Chapter 4, to find the optimal control policy of different dynamical systems. More specifically, we use the Policy Gradient (PG) algorithm, defined in Section 4.3. Although RL is widely used in dynamical control, it is applied in situations where the dynamics of the system only depend on its coordinates and the external control. It is also applied in environments that change with time adiabatically, such as the "slowly changing environments" in Ref. [29]. Other examples of similar research are discussed in Sec. 1.3.2. The scenario in these cases is that the environment becomes slightly different in subsequent episodes of the training and the agent has to adapt to learning the new environment, leading to the ideas of the "optimistic" agent in Ref. [27] and the "prognosticator" in Ref. [28].

By contrast, here we extend the scope of Reinforcement Learning to systems whose dynamics change with time significantly within a single episode. We will see that the RL-algorithms can fail without intrinsic knowledge of the time. On the one hand, such environments cannot be considered Markov Decision Processes, as their dynamics do not solely depend on the state. Also, time-agnostic RL-agents are restricted to taking the same action (or distribution of actions) in the same environment state. This can make it impossible to follow the optimal policy if it is time-dependent.

We will first look at a simple situation where the environment changes with time but the optimal policy does not (cf. the potential well in Section 5.1.1). Then, we move on to more complicated environments where changes of the environment with time require that the optimal policy depends on time as well (cf. a double well in Sec. 5.1.2 and a two-dimensional potential in Sec. 5.1.3). We then consider a quantum two-level system in Sec. 5.2, whose optimal preparation policy is also time-dependent. Time-agnostic RL-agents are more prone to failing at such tasks, requiring modifications to the algorithms so that they will incorporate information about time.

All of the programs in this section were written in Python. The differential equations simulating the dynamics of the classical systems (considered in Sec. 5.1) were solved using a Runge-Kutta method of order 8(5,3) [39] in the SciPy library. The gradients in the neural networks were computed automatically using the JAX library [40]. The

full codes required for the simulations are stored on https://github.com/gmaleksand/time-dependent-rl.

5.1 Dynamical Control Case Studies in Classical Systems

We will be describing the classical systems with Hamiltonian Dynamics, as introduced in Chapter 2.3 and control them using external forces.

5.1.1 Single Potential Well

Problem Setup

A particle with mass m is contained in a field with potential energy

$$V(x,t) = -\frac{A(t)}{\cosh \frac{x}{w(t)}}, \tag{5.1}$$

where

$$A(t) = A_0 + A_1 \sin(\omega_1 t) \tag{5.2}$$

defines a sinusoidal time-varying strength of the potential with amplitude A_1 and frequency ω_1, about the constant offset value A_0;

$$w(t) = w_0 + w_1 \sin(\omega_2 t) \tag{5.3}$$

determines the width of the potential well which again varies periodically with time with frequency ω_2 and amplitude w_1, about the mean value w_0. An example plot of this potential for a fixed moment of time is given in Figure 5.1(a).

The shape of the potential is stretched in the V direction by changing $A(t)$, making the well deeper or shallower, and $w(t)$ changes the width of the potential well. To make the potential static, we can set $w_1 = 0$ and $A_1 = 0$. Setting $w_1 = 0$ excludes the time dependence of the width and $A_1 = 0$ excludes the time dependence of the height of the potential well, respectively. As we will see later, the static system would be easier to control using the RL framework.

In the static case ($A_1 = 0, w_1 = 0$) we can easily visualize the phase portrait of the particle, as shown in Fig. 5.1(b). Near the minimum, it looks like the phase portrait of a simple harmonic oscillator, and it has free solutions at high energies, resulting in open orbits.

We now consider the following control problem: By manipulating the system using an external constant force F directed leftwards or rightwards, get the particle to escape the potential well and approach infinity, as shown in Fig. 5.1.

Adding this control to the system, its dynamics are defined by the following equations of motion:

$$\begin{aligned} \dot{p}(t) &= -A\frac{\tanh(x/a)}{a\cosh(x/a)} + F, \\ \dot{x}(t) &= \frac{p}{m}. \end{aligned} \tag{5.4}$$

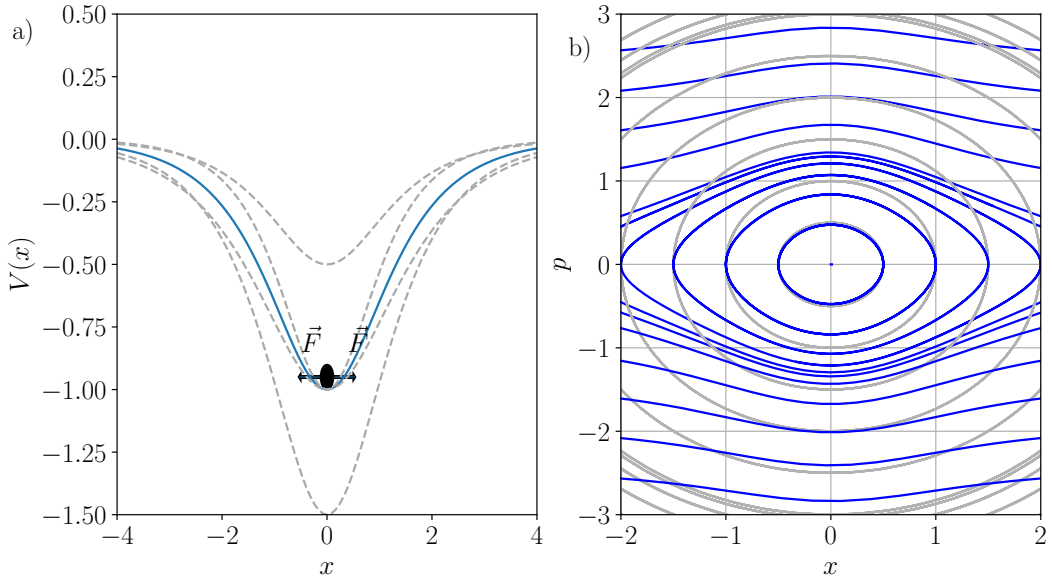

Figure 5.1: a) Plot of the potential energy of the particle at $t = 0$. The particle is initialized at rest at the minimum ($x = 0$) and the goal is to escape the potential well and move towards infinity using a constant force F directed leftwards or rightwards. The dashed lines show the potential at the extreme widths and amplitudes. b) Phase portrait of the simple potential well with no time dependence and without external control (plotted in blue). For comparison, the gray circles correspond to the phase portrait of a simple harmonic oscillator with frequency $\omega = 1\ \mathrm{s}^{-1}$. Constants in the plots: $A_0 = 1$ J, $w_0 = 1$ m, $A_1 = 0.5$ J, $w_0 = 0.2$ m, $m = 1$ kg.

The force F is chosen small enough so that we cannot make the particle escape by only applying it in one direction. However, it is useful to think of a **theoretical strategy** as a benchmark to the RL performance. The theoretical solution to the control problem we suggest is the following. In order for the particle to escape from the potential well, we can always apply forces in the direction of motion so that its energy would always increase. In that way, the particle would eventually have enough energy to reach infinity. This strategy has an additional benefit – it should work whether the width and height of the potential well change with time or not. A similar problem and solution are discussed in Ref. [34] as a "mountain car problem".

RL approach for the time-independent system

To cast this problem within the RL framework, we need to define the state space, action space, and reward space. The physical system described above is a deterministic environment, whose RL states can be defined as the physical states of the phase space, which means that a state is $s = (x, p)$ – the coordinate and momentum of the particle. Initially, the particle is at rest in the minimum of the potential well, i.e., the initial state is $s_0 = (0, 0)$.

The RL agent can take one of two actions: a constant force directed leftwards or rightwards, so $\mathcal{A} = \{-F, +F\}$. We set the reward to be $r = |x|$, so that the further away the agent is from the minimum, the higher reward it receives, which is in line with the goal of approaching an infinite distance from the center of the potential well. We are using

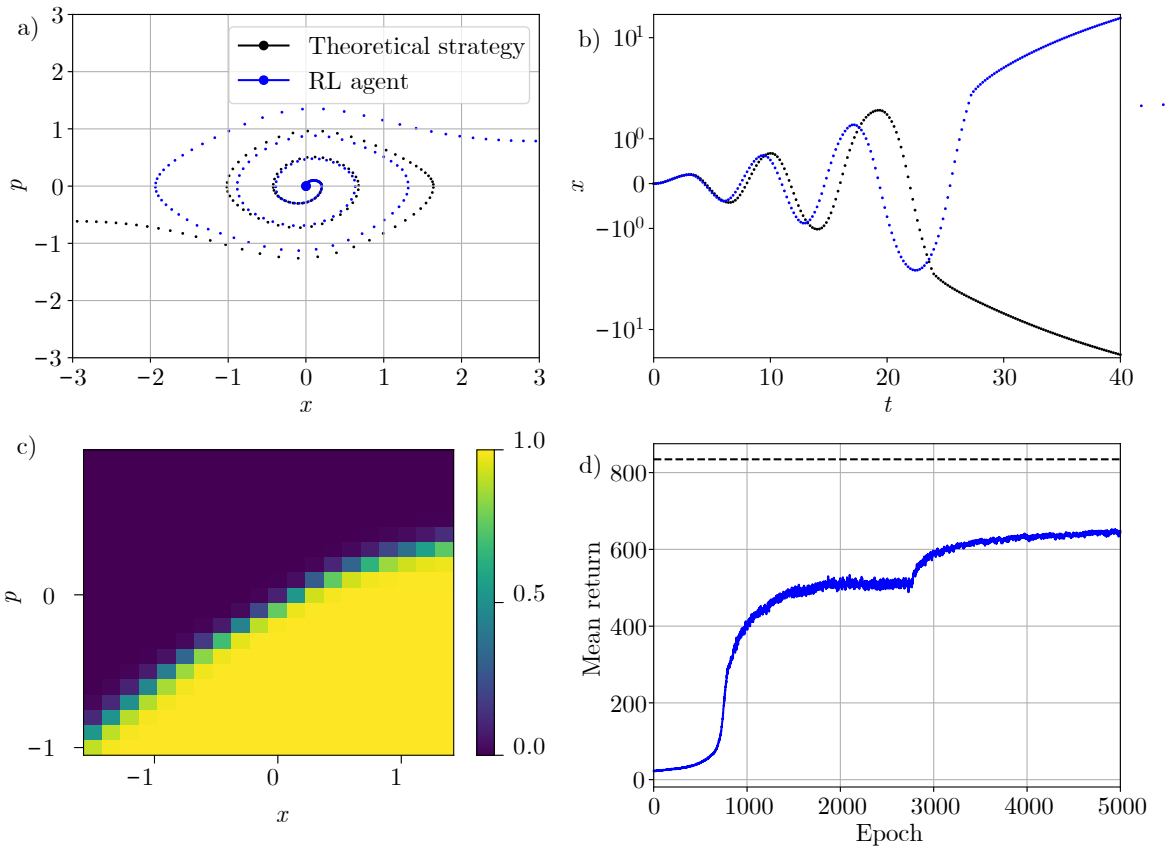

Figure 5.2: Performance of the RL agent in the time-independent simple potential well. a) phase diagram of the controlled particle, b) coordinate of the controlled particle as a function of time. The blue trajectory corresponds to a particle controlled by the RL agent, and the black trajectory - particle controlled by the theoretically suggested strategy. We notice that the agent behaves similar to the expectation, although the theoretical strategy is still better. c) strategy learned by the agent. The ploted value is the probability to act with force in leftwards direction. d) training curve of the agent and the dashed line is the return acquired by the theoretically optimal strategy. Constants for the simulation: $m = 1$ kg, $A_0 = 1$ J, $A_1 = 0, w_0 = 1$ m, $w_1 = 0, \omega_1 = 2$ s^{-1}, $\omega_2 = \sqrt{2}$ s^{-1}, $F = 0.1$ N.

discretized time steps and cut off the episode to a finite length which is chosen to be long enough so that the particle can reach a place where the potential energy will be negligible. However, the length of the episode should not be too long so that the trajectory will not be dominated by the free movement. The exact discretization is shown in Table A.2.

We are searching for the correct policy using the Policy Gradient algorithm with hyperparameters described in Table A.9. In the fixed case, we observe that the algorithm found a policy that resembles our physical intuition and received a similar return to the theoretical solution. We can see its performance in Fig. 5.2.

RL approach for the time-dependent problem

Now, let us add the time dependence to the system. Since the static policy works in the time-dependent environment, the same agent can learn to escape the potential well even

if we do not give it explicit information about time, as can be seen in Fig. 5.3. However, we can modify our algorithm in two different ways so that it can incorporate the time dependence. We do this by complementing the state $s = (x, p)$.

The simpler way we can modify the state is to include the phases $\omega_1 t$ and $\omega_2 t$ of the varying strength and width of the potential. We observe that if the agent receives states in the form $s = (p, x, \sin(\omega_1 t), \sin(\omega_2 t)$, this additional information helps it to learn a strategy that performs somewhat better than the strategy learned without any information about time. The result is shown in Fig. 5.3. However, we should be wary of this procedure. Although it is reasonable that a controlling agent should have information about time, the information about ω_1 and ω_2 is specific to the system we want to control and may not be directly available.

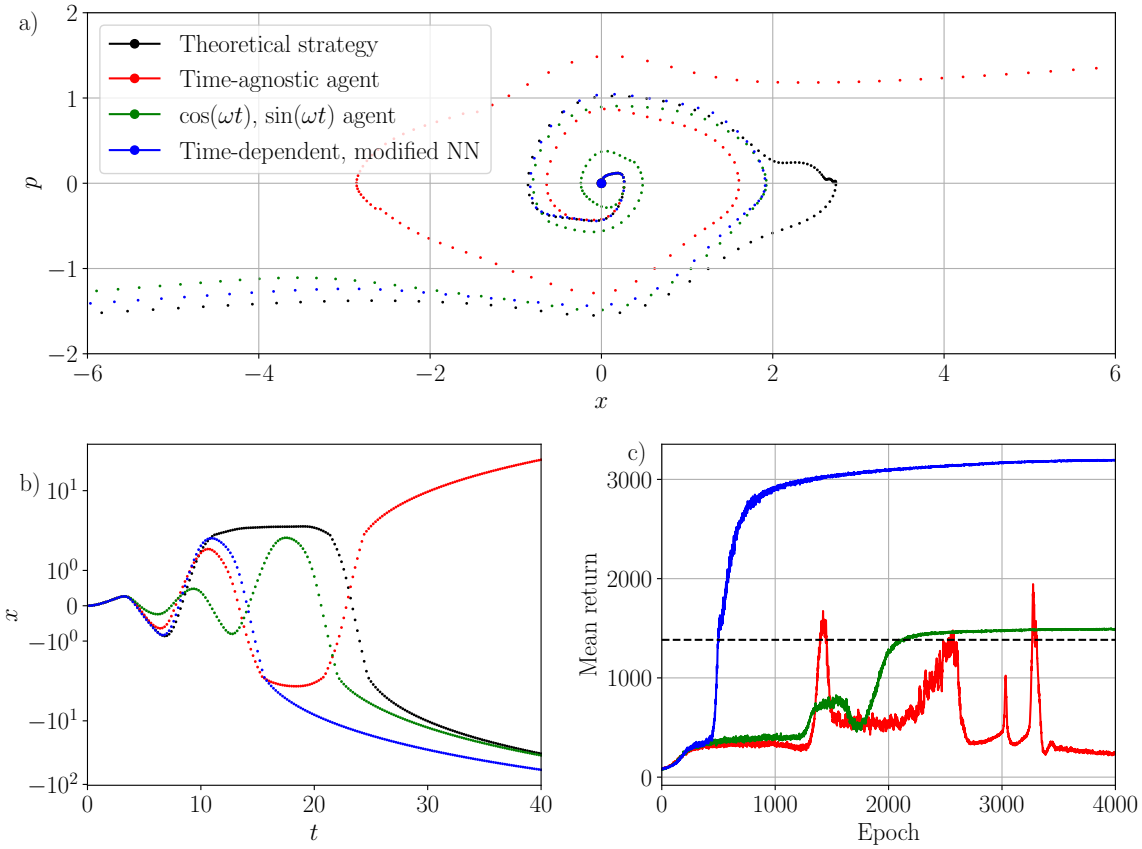

Figure 5.3: Performance of the different agents in the time-dependent simple potential well. Black: theoretical strategy ($F > 0 \iff p > 0$), red: agent receiving only the phase space state $s = (p, x)$, green: agent receiving time in the form $\sin(\omega_1 t)$ and $\sin(\omega_2 t)$, blue: agent receiving t through an extra cos-layer. a) shows the phase diagrams of the controlled particle, b) – the position of the particle as a function of time, and c) – the training curve of the different agents. As the time-agnostic agent (red) failed to learn in a stable way, the plotted trajectory is its most successful during training. The agent using our modified neural network (blue) managed to find the best possible control strategy among all agents as it escapes the potential well earliest. Constants for the simulations: $m = 1$ kg, $A_0 = 1$ J, $A_1 = 0.5$ J, $w_0 = 1$ m, $w_1 = 0.2$ m, $\omega_1 = 2$ s^{-1}, $\omega_2 = \sqrt{2}$ s^{-1}, $F = 0.1$ N.

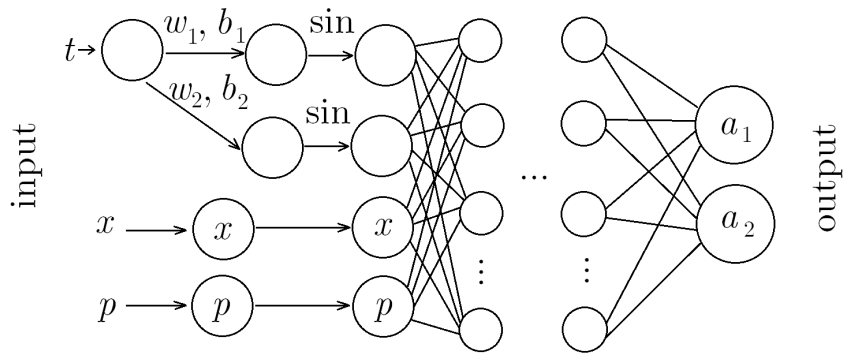

Figure 5.4: Structure of the modified neural network used for the time-dependent environment. Time is being preprocessed by an additional layer with two neurons and a sine function.

We now present a modification that would be more universal – it may be suitable for all cases where the system dynamics evolve non-adiabatically but periodically with time, assuming we do not know the period of the time variations. The RL state will now consist of time and the phase space state of the physical system $s = (t, s_{\text{physical}})$, in our case $s = (t, x, p)$. In addition to this, we extend the neural network with an additional layer through which we pass the time parameter. This layer consists of two neurons[1], which use the sine function as their non-linearity, so that their outputs are

$$\sin(w_i t + b_i), i = 1, 2, \tag{5.5}$$

where w_1, w_2, b_1, b_2 are parameters of this extended neural network. The resulting network can be seen in Fig. 5.4.

Using this approach, the RL agent finds an even better policy (in the sense of higher received reward) than the agent which was trained using the hard-wired $\sin(\omega_1 t)$ and $\sin(\omega_2 t)$ where ω_1 and ω_2 are the frequencies of the time variations, cf. Eqs. (5.2), (5.3). The result of the training can be seen in Fig. 5.3.

With this problem, we demonstrate that the need of using RL algorithms that take time into account in controlling the dynamical system depend on the specific control problem we have. In this case, the dynamics of the system vary with time. There exist good control policies that do not depend on time, so using time-agnostic control may work well. However, even in this case, the time-dependent agent finds a policy that is closer to the optimal.

5.1.2 Double-Well Potential

We will now move to a bit more complicated example where we can intuitively see that the time-independent control policy cannot be optimal. Our potential well is replaced with a double well with the goal to traverse from one of the minima to the other. This time, there will be a time-dependent shortcut between the two minima, which changes the optimal strategy. To set up the problem, first, we will consider our physical system.

[1] In our case we use two neurons as we have two characteristic time scales of the time-dependence. However, this method can also be used with a different number of neurons.

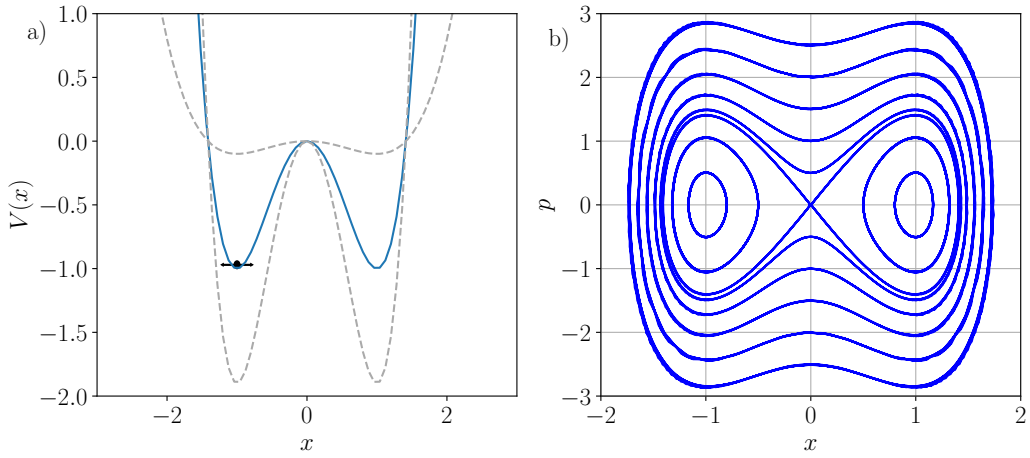

Figure 5.5: a) Plot of the potential energy of the particle at $t = 0$. The particle is initialized in the left minimum; the goal is to settle in the rightmost minimum using a constant force F in a desired direction. The blue plot is the value of the potential at $t = 0$, while the dashed lines show the extreme values of the potential. Constants for the plot: $A_0 = 1$ J, $A_1 = 0.9$ J, $w = 1$ m. b) Phase portrait of the double well with no time dependence and without external control. All particle trajectories are closed. However, in the lower-energy trajectories the particle orbits only one of the minima; therefore, it needs to first acquire higher energy, and then lower it when it approaches the second minimum. The model parameters are: $A = 1$ J, $w = 1$ m, $m = 1$ kg.

Problem Setup

A particle with mass m is contained in a field with potential energy

$$V(x,t) = A(t)\left(\left(\frac{x}{w}\right)^4 - 2\left(\frac{x}{w}\right)^2\right), \tag{5.6}$$

where

$$A(t) = A_0 + A_1 \sin(\omega t) \tag{5.7}$$

defines a time-varying height of the potential well with amplitude of the variations A_1 and frequency ω_1 about the value A_0, and w sets the positions of the minima: $x_{\min} = \pm w$. An example plot of this potential for a fixed moment of time is given in Figure 5.5(a).

We now consider the following control problem: Using a constant external force F directed leftwards or rightwards, if the particle is initially at rest at one of the minima ($x = -w, p = 0$), get the particle to escape from this well and settle in the other minimum ($x = w, p = 0$).

By adding the control to the system, our equations of motion become

$$\begin{aligned} \dot{p}(t) &= \frac{4A}{w}\left(\frac{x}{w} - \frac{x^3}{w^3}\right) + F, \\ \dot{x}(t) &= \frac{p}{m}. \end{aligned} \tag{5.8}$$

We can see the phase space portrait of the static double well potential in Fig. 5.5(b). The particle can oscillate near both minima and needs additional energy in order to traverse from one side of the potential to the other.

Again, the static case is obtained by setting $A_1 = 0$. This time, the intrinsic variation comes from $A(t)$ which can be interpreted as an overall height of the potential well. Note that the overall structure of the double-well potential stays the same by varying $A(t)$. The function x^4 quickly increases for large x and this behavior is not affected by $A(t)$. The most significant difference is produced in the space between the two minima. As A is the energy difference between the minima and the central maximum, we will interpret lower values of A as an "open shortcut" between the two minima, as can be seen in Fig. 5.5(a).

The expectation is that, in the static case, the force should again be applied in the direction of motion to accumulate energy, until the particle passes to the region around the second minimum. Then we should apply the force in the opposite direction so that the particle would lose energy and eventually stop. This will be referred to as a 'theoretical strategy'. However, in the dynamical case, it would be easier for the particle to pass from one valley to the other during the time interval when the barrier height is minimal.

RL approach

Now, let us translate the problem into the RL framework. In the static case, the RL-state can be defined again as $s = (x, p)$ – the phase space state of the particle, and the initial state would be $s_0 = (-w, 0)$.

In the time-dependent case, we will again test different algorithms with different definitions of the RL state. A time-agnostic algorithm would receive only the phase space state $s = (x, p)$. Extra-knowledge algorithms would use both the time t and the frequency ω of the potential variations in the form $\sin(\omega t)$ and $\cos(\omega t)$, so the total state consists of $s = (\sin(\omega t), \cos(\omega t), x, p)$. We also test an algorithm receiving unprocessed information about the time t, in which case $s = (t, x, p)$. In all of the above cases, the initial time is $t = 0$.

The action space again consists of a constant force directed leftwards or rightwards: $\mathcal{A} = \{-F, +F\}$. The reward has to be set in such a way as to be higher when the particle becomes closer to the goal $s_{\text{final}} = (w, 0)$. To treat all agents equally, we set the same reward

$$r = -4A\left(\frac{x}{w} - 1\right)^2 - k\frac{p^2}{2m} - u, \tag{5.9}$$

where $-4A(x/w - 1)^2$ approximates the potential energy around the desired minimum, giving higher rewards when the particle is closer to the minimum; $kp^2/(2m)$ is a kinetic energy term, giving a higher reward when the particle stops; and u is an additional constant penalty for "non-completed" trajectories, which is given when the particle is outside a small phase space 'ellipse', i.e., when $(x - w)^2/x_{\text{tol}}^2 + (p/p_{\text{tol}})^2 > 1$. We can see the values of the different reward weights used in the simulations in Table A.5.

All agents will use the same finite episode length containing T discretized time step of length Δt in terms of the time of the physical system. The length of the episodes is chosen to be somewhat longer than the time necessary for the particle to traverse from the first to the second minimum of the double well, so that the agents can be able to learn the optimal strategy gradually, first strategies where the particle approaches the goal in a longer time. The exact discretization can be seen in Table A.5.

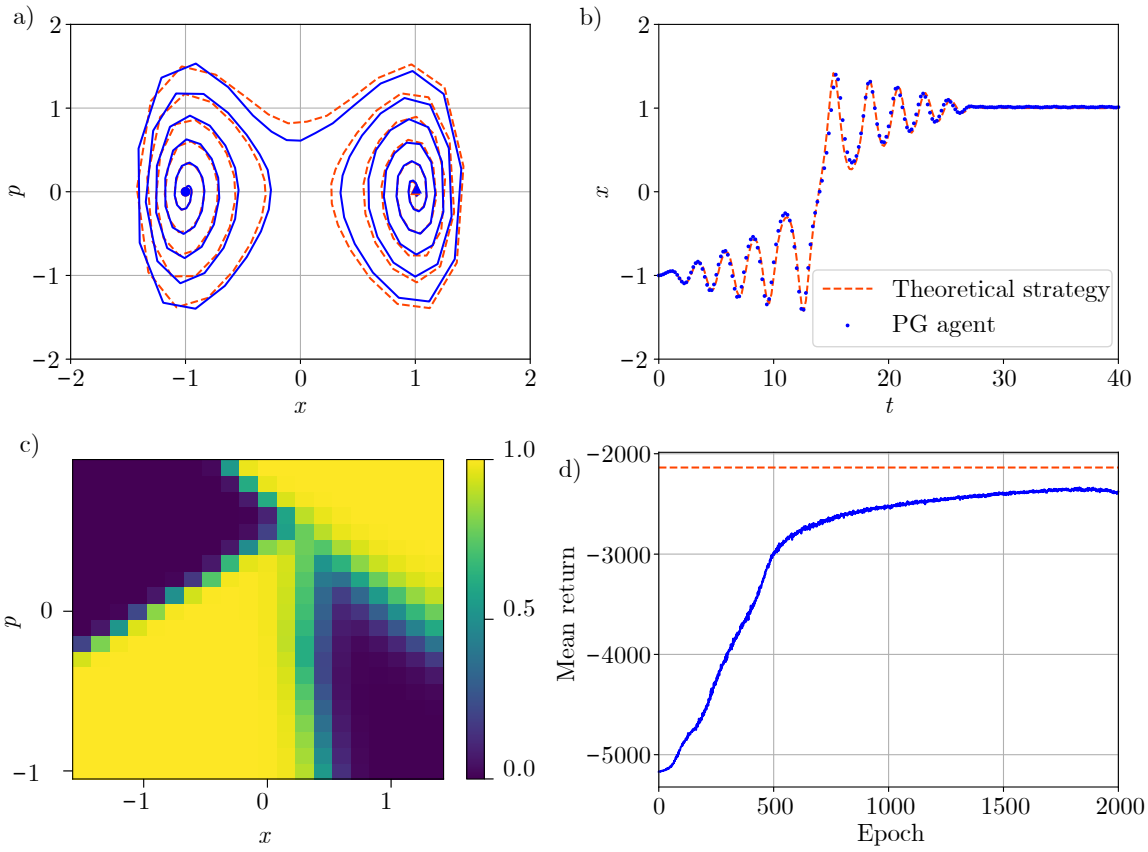

Figure 5.6: Performance of the RL agent in the time-independent double well. a) phase portrait of the controlled particle. b) coordinate of the controlled particle as a function of time. The blue trajectory corresponds to a particle controlled by the RL agent, and the orange – to a particle controlled by the theoretically suggested strategy [see text]. In this case, the agent moves along a nearly perfectly aligned trajectory with the theoretical. c) strategy learned by the agent. The colorbar shows the probability to act with force in the leftward direction. d) training curve of the agent; the dashed line is the return acquired by the theoretically optimal strategy. The constants for the simulation are as in Table A.4-A.6 except for $A_1 = 0$.

Results

We again use the PG-algorithm to solve this problem, with hyperparameters as shown in Table A.6. In the static case, we can that that the algorithm can find a policy which resembles our expectation, provided that the force used to control the particle is large enough so that the neural network has a high probability of observing trajectories that successfully exit the 'undesired valley' around $x = -w$. We can see its performance in Fig. 5.6.

It would be interesting to find neural networks or modifications of the reward function which can successfully control the particle with even smaller forces. However, this is currently beyond our focus as we will be trying to control the particle in the time-dependent case. We can speculate that this could be achieved by altering the initial state so that one acquires enough trajectories close to "seeing" the other side of the barrier, or by modifying the reward function in a way that would encourage exiting the valley as a first step.

Now, for the time-dependent scenario, we can again first try the same theoretical strategy: apply a force $+F$ whenever $p > 0, x < 0$ or $p < 0, x > 0$, and $-F$ otherwise. However, since the trajectory is sensitive to the time variations of the potential, this strategy may completely fail when the particle gets to pass the potential barrier at an inappropriate time, as shown in Fig. 5.7.1 (a,b).

We are using two time-dependent agents, which are structured similarly to the previous Sec. 5.1.1. The first one ("$\sin(\omega t), \cos(\omega t)$"-agent) receives $s = (p, x, \sin(\omega t), \cos(\omega t))$ as a state. This state is fed directly into the neural network. The second agent, which we will call the "Adaptive time agent", receives $s = (t, p, x)$ However, the time component is transformed through an additional two-neuron cos-layer as in Fig. 5.4. We observe that both time-dependent agents can learn to navigate the particle to the goal using a smaller force magnitude (as listed in Table A.5) than the static case. This can be explained by the fact that the particle passes through a lower barrier between the two minima as the agents successfully use the time window when the shortcut is open. This can be confirmed by looking at the particle position as a function of time in the controlled trajectories, cf. Fig. 5.7b). Both agents managed to receive very similar returns. The more universal agent (T2) performed a bit better. Therefore, it is sufficient to provide the agent with the raw time t. The agent can still learn a successful policy even without providing it with ω – which acts as additional information about the potential.

For comparison, we test a time-agnostic agent as well. In the same case when the time-dependent agents learn to reach the goal, it fails to learn in a stable way, as seen in Fig. 5.8. Moreover, even before "unlearning", the acquired policy still fails to reach the desired goal. However, it successfully escapes the first potential well, as seen in Fig. 5.7a.2. This agent may sometimes reach a good level of performance, but it has a fundamental problem with learning in a stable way.

The problem is that as the environment changes with time, it can no longer be considered a Markov decision process. The current state (p, x) and action do not give away the full information necessary to determine the next state. As the agent can visit the same state at different time steps, changing its action can be an improvement for one of the time steps and a deterioration for another. This phenomenon confuses the agent, leading to the destabilization of the agent's learning.

We should note this could not be the only reason for the failure. At other parameter choices (e.g. increasing A, making the wells deeper), the time-dependent agents fail to learn a successful policy as well.

5.1.3 Two-Dimensional Potential

Now, let us continue with a case where we would have less intuition. We now have a two-dimensional system, containing multiple potential barriers, a rotating shortcut and drag force. The extra dimension will bring a greater variety of trajectories. The shortcut-equipped barriers will present a maze-like environment, which is difficult to navigate. The drag force will prevent accumulating too much energy and will ensure that the potential remains relevant.

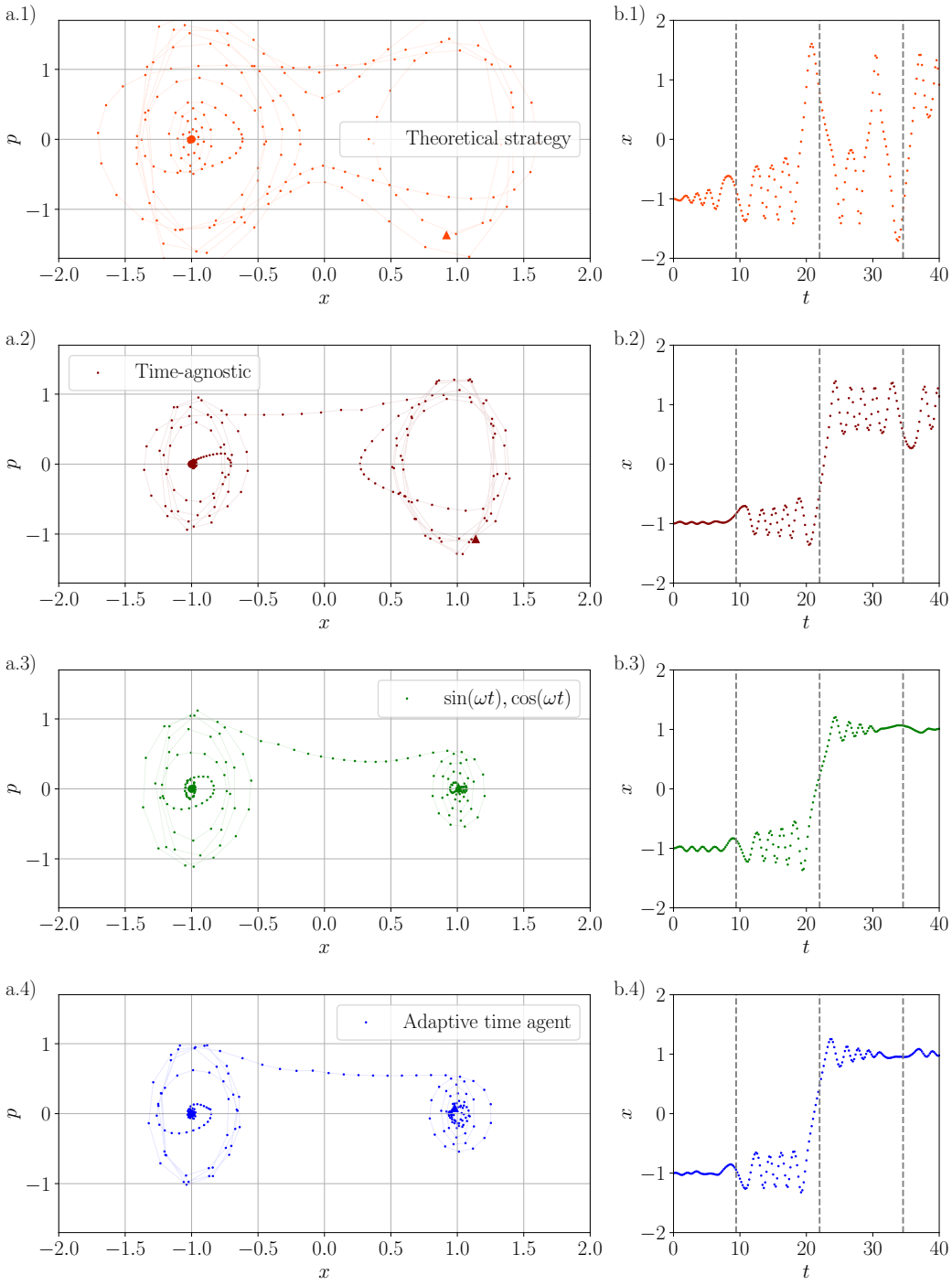

Figure 5.7: Performance of the RL agents in the time-dependent double well. a) phase diagram of the controlled particles, b) coordinate of the controlled particle as a function of time. The theoretical strategy may work well if the particle passes the barrier at the appropriate time. However, in this choice of parameters, it completely fails, as demonstrated in (1). For the time-agnostic agent (2), we are showing the best trajectory during training (at epoch 2500, see Fig. 5.8) which still fails to reach the minimum. The dashed vertical lines in (b) are placed at $t = 1.5\frac{\pi}{\omega}, t = 3.5\frac{\pi}{\omega}$ and $t = 5.5\frac{\pi}{\omega}$ when the potential height is lowest. The time-dependent agents (3,4) learn to pass when the shortcut is open. Constants as in Table A.4-A.6.

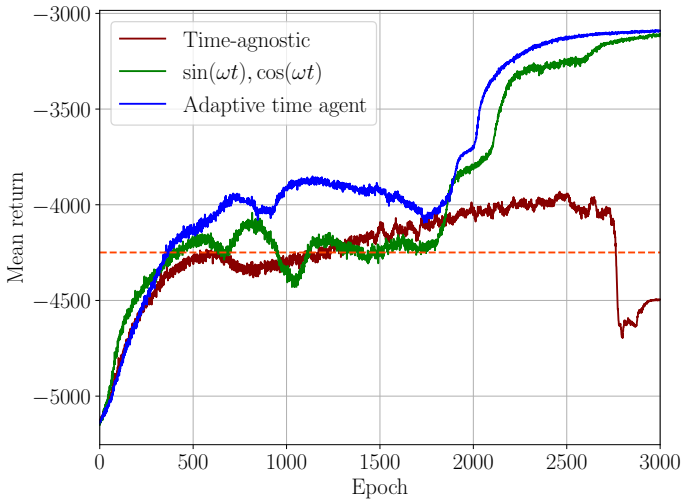

Figure 5.8: Training curves of the different RL agents in the time-dependent double well. The return received by the theoretical strategy is plotted by the dashed line. The time-agnostic agent fails to learn in a stable way. Constants as in Table A.4-A.6.

Problem Setup

A particle with mass m is contained in a field with potential energy

$$V(\rho, \theta, t) = AR(\rho)\Theta(\theta + \omega t), \tag{5.10}$$

where the radial profile of the potential has the form

$$R(\rho) = A|\rho| \sin^2(\rho/\rho_0), \tag{5.11}$$

which defines multiple potential barriers of increasing height as seen in Fig. 5.9 (a). The parameters A and r_0 control the height of the potential barrier and the distance between the barriers respectively. The angular part of the potential is given by the function

$$\Theta(\theta) = 1 - e^{-\kappa \sin^2(\theta)}, \tag{5.12}$$

which has the property of extinguishing the potential around $\theta = 0, \pi$, as seen in Fig. 5.9 (b), making the area around $\theta = 0, \pi$ a 'shortcut'. The parameter κ controls the width of the shortcut. Due to the time-dependence $\Theta(\theta + \omega t)$, the shortcut rotates at a constant angular velocity ω. The whole 2d structure of the potential can be seen in Fig. 5.9 (c).

If we ignore the shortcut, the potential is radially symmetric and contains a minimum at the center. The rest of the minima are equidistantly spaced in the form of concentric circles. Due to the symmetry of the potential, it will be easier to work with polar coordinates $\vec{r} = (\rho, \theta)$ and $\vec{p} = (p_\rho, p_\theta)$. The static equivalent of this system can be obtained by setting $\omega = 0$. This way we would still have a direction in space with a shortcut, but it would not be rotating, so unlike the previous problem in Sec. 5.1.2, the static version of the problem is now easier to control since the shortcut is still present.

There is also a friction force acting on the system, proportional to the velocity of the particle: $\vec{f}_{\text{friction}} = -\beta m \vec{v}$.

With this system, we will consider the following control problem: If the particle is initially at rest at the central minimum, $\vec{r}_{\text{init}} = (0, 0)$, $\vec{p}_{\text{init}} = (0, 0)$, using a force with constant

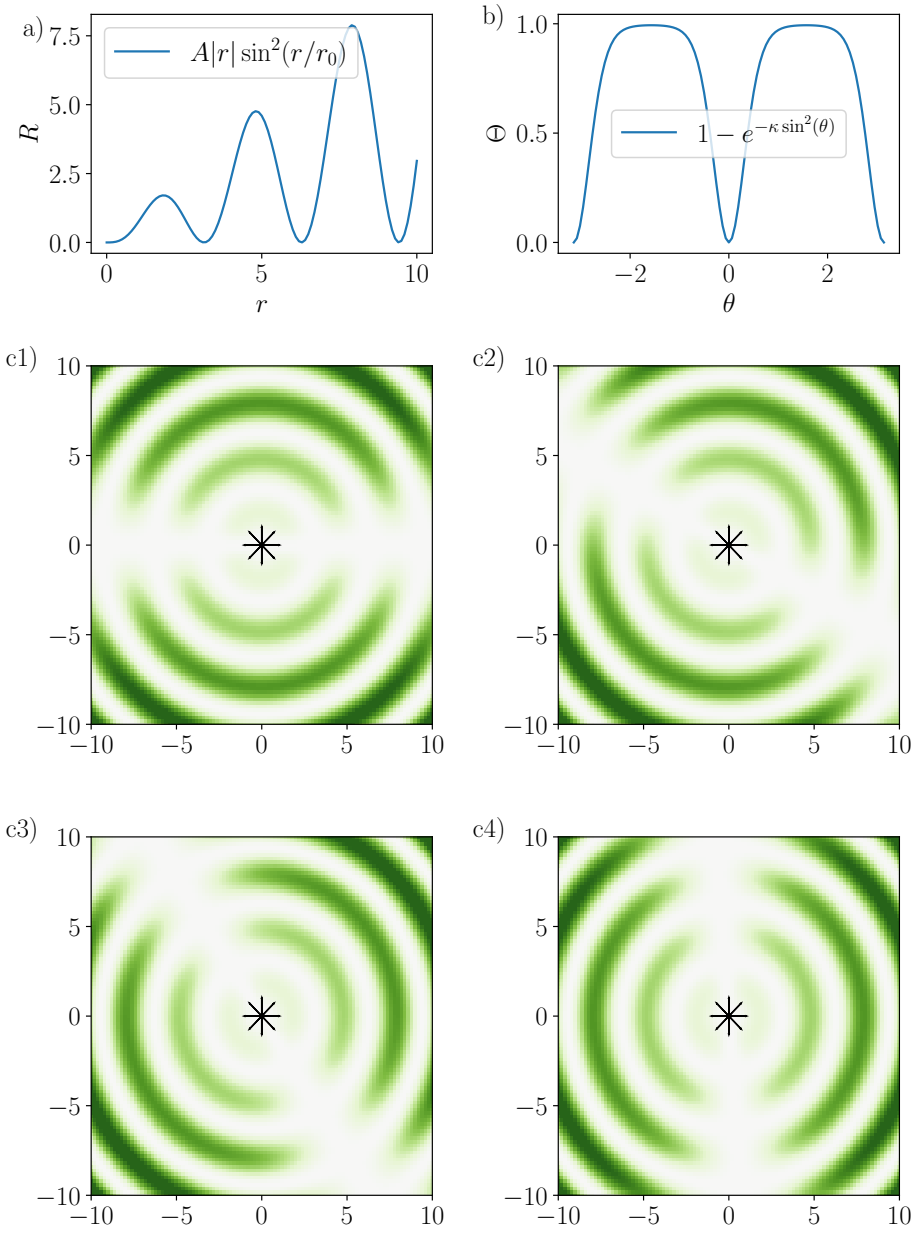

Figure 5.9: Plot of the potential energy of the particle at $t = 0$. (a) radial profile of the potential; (b) angular profile of the potential; (c) 1-4 – color plot of the two-dimensional potential at different time steps 0, t, $2t$, $3t$. The potential contains an infinite number of minima and maxima as well as a rotating shortcut. The particle is initialized at the central (0-th) minimum and the goal is to end up in the third minimum. Parameter values: $\kappa = 5$, $A = 1$ J, $r_0 = 1$ m, $\omega = 1$ s^{-1}, $t = 0.5$ s.

magnitude F, we should make the particle reach another desired minimum (e.g., the third minimum $|\vec{r}| = 3\pi\rho_0$). The particle is free to move in the θ-direction as it reaches the minimum.

In polar coordinates, the Hamilton equations of motion (see Eq. (2.25)) read:

$$\begin{aligned}
\dot{p_\rho}(t) &= \frac{p_\theta^2}{m\rho^3} - R'(\rho)\Theta(\theta+\omega t) + F_\rho - \beta p_\rho, \\
\dot{p_\theta}(t) &= -R(\rho)\Theta'(\theta+\omega t) + rF_\theta - \beta p_\theta, \\
\dot{\rho}(t) &= \frac{p_\rho}{m}, \\
\dot{\theta}(t) &= \frac{p_\theta}{m\rho^2},
\end{aligned} \tag{5.13}$$

where

$$\begin{aligned}
R'(\rho) &= \frac{A|\rho|}{\rho_0}\sin(2\rho/\rho_0) + A\text{sign}(\rho)\sin^2(\rho/\rho_0), \\
\Theta'(\theta+\omega t) &= e^{-\kappa\sin^2(\theta+\omega t)}\kappa 2\sin(\theta+\omega t)\cos(\theta+\omega t).
\end{aligned} \tag{5.14}$$

The rotating in time potential presents us with the possibility to transform the coordinate system to a co-rotating frame where the time dependence of the potential is removed. This means that our dynamic problem can be transformed into a static problem. We can do this using the transformation

$$\begin{aligned}
\vartheta &= \theta + \omega t, \\
p_\vartheta &= p_\theta,
\end{aligned} \tag{5.15}$$

which gives us the following Hamiltonian:

$$H' = H + \omega p_\vartheta \tag{5.16}$$

and new equations of motion

$$\begin{aligned}
\dot{p_\rho}(t) &= \frac{p_\vartheta^2}{m\rho^3} - R'(\rho)\Theta(\vartheta) + F_\rho - \beta p_\rho, \\
\dot{\rho}(t) &= \frac{p_\rho}{m}, \\
\dot{p_\vartheta}(t) &= -R(r)\Theta'(\vartheta) + \rho F_\theta - \beta p_\vartheta, \\
\dot{\vartheta}(t) &= \frac{p_\vartheta}{m\rho^2} + \omega.
\end{aligned} \tag{5.17}$$

Numerical simulation of the potential

In the numerical simulations, the radial coordinates present certain difficulties near the origin of the coordinate system ($\rho = 0$) as $\dot{p_r}\tilde{p}_\theta^2/\rho^3 \to \infty$, $\dot{\theta}\tilde{p}_\theta/\rho^2 \to \infty$. We cannot evade the divisions by ρ completely, but we will make the substitution

$$\tilde{p}_\theta = \frac{p_\theta}{\rho} \tag{5.18}$$

which will allow us to reduce the power of ρ in these divisions. As the potential barrier grows linearly with r, $\dot{\tilde{p}}_\theta$ will not contain division by zero. Also, in order to ensure numerical stability around $\rho = 0$, we replace the terms containing ρ^{-1} with "clipped" values between -10 and 10. With these substitutions, the equations of motion used in

the simulations are as follows:

$$\begin{aligned}
\dot{p_\rho}(t) &= \text{clip}\left(\frac{\tilde{p}_\theta^2}{mr}\right) - A\left(\frac{|\rho|}{\rho_0}\sin(2\rho/\rho_0) + \text{sign}(\rho)\sin^2(\rho/\rho_0)\right)(1 - e^{-\kappa\sin^2(\theta+\omega t)}) + F_\rho - \beta p_\rho, \\
\dot{\tilde{p}}_\theta(t) &= -2\kappa A \text{sign}(\rho)\sin^2(\rho/\rho_0)e^{-\kappa\sin^2(\theta+\omega t)}\sin(\theta+\omega t)\cos(\theta+\omega t) + F_\theta - \beta\tilde{p}_\theta, \\
\dot{\rho}(t) &= \frac{p_\rho}{m}, \\
\dot{\theta}(t) &= \text{clip}\left(\frac{\tilde{p}_\theta}{m\rho}\right),
\end{aligned} \tag{5.19}$$

where $\text{clip}(x) = \max(-10, \min(10, x))$.

RL approach

Let us again translate the problem to the RL framework. The RL state will coincide with the phase space state. For numerical reasons, we prefer to use $\tilde{p}_\theta$ instead of p_θ. As p_θ will become much larger than $\tilde{p}_\theta$ by increasing r, it could dominate in the neural network input.

The static RL state we use is $s = (\rho, \theta, p_\rho, \tilde{p}_\theta)$, and the RL state being used by the time-dependent agent will be $s = (t, \theta, \rho, \tilde{p}_\theta, p_r)$ with both t and θ being preprocessed through a cos-layer as in Fig. 5.4. The initial state is given by a distribution around $\rho = 0, \theta = 0, p_\rho = 0, p_\theta = 0$, which can be seen in Table A.7-A.9

The possible actions $\mathcal{A}$ are forces in either direction, which means they could be parameterized by an angle ϕ showing the direction of the force, so that in polar coordinates $\vec{F} = (\dot{p_\rho}, \dot{\tilde{p}}_\theta) = (F_\rho, \rho F_\theta) = (F\sin\phi, \rho F\cos\phi)$. We discretize the possible actions to angles $\phi = k\pi/4$, $k \in \{0, 1, ..., 7\}$. We define the reward function as $r = -|\rho - 3\pi\rho_0|$ for all agents.

Results

We are using the PG-algorithm with hyper-parameters as shown in Table A.9. In order for the agent to reach the goal quickly, it should find the position of the shortcuts. It would be harder for the particle to jump over the potential barriers, as it has a F/β cap on its momentum due to friction, making it difficult to gain enough energy to pass the barriers.

We notice that both agents – the one with knowledge of time and the one which does not know about time, managed to learn the successful policy. This can be seen in Fig. 5.10. The particle controlled by both agents passes the barriers at a zone around the shortcuts, where the potential energy is lower. As the initial states have small variations, the resulting passage occurs at different time steps, with different values of θ. However, when we transform the trajectories to the rotating frame of reference, we see that they all have similar $\theta - \omega t$-coordinate when their radial coordinate is on the barriers.

The reason why the behavior of the time-agnostic agent is similar to the time-dependent one is still unclear. One possible speculation is that although the potential and the trajectories depend on time, the successful control sequence might have a strong time dependence. Also, it could be the case that the time-agnostic agent, although receiving

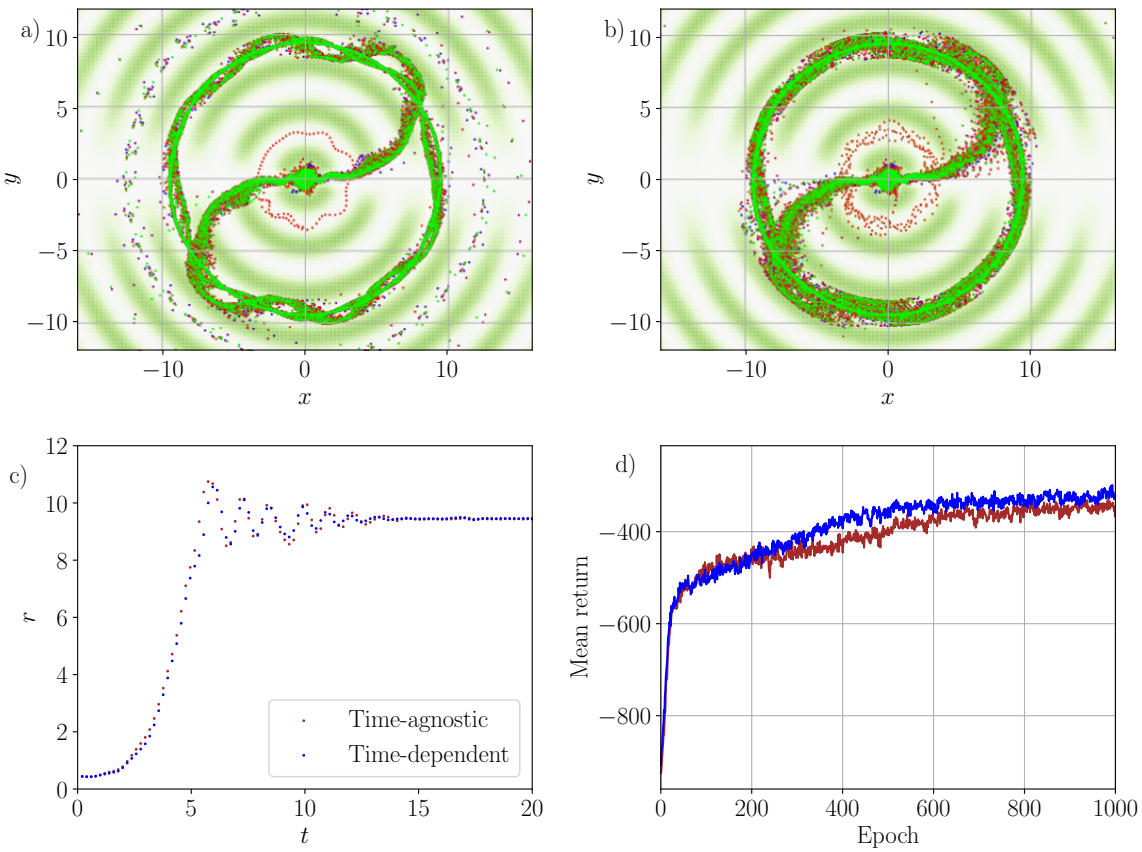

Figure 5.10: Performance of the RL agents in the time-dependent potential. (a), (b) – trajectories of the controlled particles; (a) – agent without knowledge of time, (b) – agent with knowledge of time; (c) – the coordinate of the controlled particles of an arbitrary trajectory in the batch as a function of time; (d) – training curves of different agents. Out of convenience, all of the diagrams are plotted in the rotating reference frame in order to see where the shortcuts are located with respect to the current position of the particle. Both agents learn to pass in the vicinity of the shortcuts, while doing so at different time steps, respectively at different angles θ. Both agents display similar training curves The constants for the simulation are as in Table A.9

partial information, still has enough data to deduce its action correctly. We have tried modifying the potential by rotating the shortcuts with different frequencies, which should prevent the possibility of a good time-independent strategy. However, we observed that the time-dependent agent also performed poorly in that situation. This experiment can be seen in App. B. The question of whether we do need to incorporate time in this higher dimensional problem is still open for further research.

5.2 Dynamical Control Case Studies in Quantum Systems

In this section, we apply Reinforcement Learning to prepare a two-level quantum system, e.g., a qubit, as described in Sec. 3.2, in a desired target state.

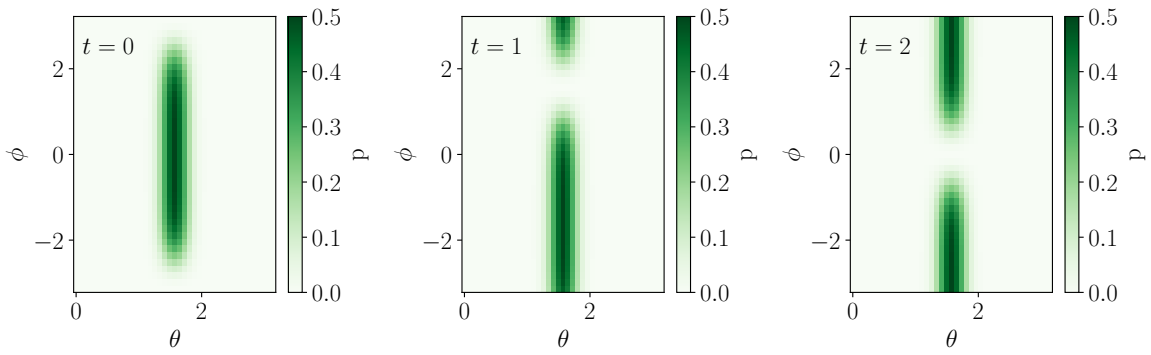

Figure 5.11: Color plot of the qubit decay probability from the current state $|\psi\rangle$ to $|1\rangle$. The green color shows the area where the decay probability is higher. It is located near the equator of the Bloch sphere. The three plots correspond to different time steps $t = 0$, $t = 1$, and $t = 2$. Model parameters: $p_0 = 0.5$ $\kappa_1 = 20$, $\kappa_2 = 1$, $\omega = 2.5$, $t = 1$.

Problem Setup

We consider a two-level quantum system. Its state is described by Eq. (3.15), which can be represented on the Bloch sphere (see Eq. (3.17)).

The qubit state can be controlled using quantum gates, which are represented by the unitary transformations

$$\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \hat{R}_{\pm x} = e^{\pm i\Delta t\sigma_x}, \quad \hat{R}_{\pm y} = e^{\pm i\Delta t\sigma_y}, \quad \hat{R}_{\pm z} = e^{\pm i\Delta t\sigma_z}, \tag{5.20}$$

where σ_x, σ_y, and σ_z are the Pauli matrices (see Sec. 3.2.2), and Δt is a time step.

The qubit will be initialized in a random state whose ϕ-coordinate is uniformly distributed in the interval $(-\pi, \pi]$ and θ – in the interval $[0.8\pi, \pi]$. The idea is that the initial states are on the southern hemisphere of the Bloch sphere, near $|1\rangle$.

We can apply one of the seven gates for each time step. The goal is to bring the qubit to the target state $|\psi_{\text{target}}\rangle = |0\rangle$ – the north pole of the Bloch sphere, in a fixed number of time steps T.

A difficulty for this state preparation problem will arise if the qubit starts interacting with an environment, leading to decoherence. We model the possible decay of the qubit as a decay from the current state $|\psi\rangle$ to $|1\rangle$. Due to the choice of environment, the decay probability is state and time-dependent, and we will model it by the following equation:

$$p_{\text{decay}}(\theta, \phi, t) = p_0 e^{-\kappa_1(\theta-\pi/2)^2}(1 - e^{-\kappa_2(\phi+\omega t+\pi)^2}), \tag{5.21}$$

which gives us a region near the equator on the Bloch sphere, where the probability of decay is higher (Fig. 5.11). The ϕ-coordinate of this region rotates with time at frequency ω. The constants κ_1 and κ_2 respectively determine the width of the high-decay region around the equator in the θ and ϕ direction, and p_0 is the amplitude of the decay probability in this region. Note that the initial state is chosen so that it is at a significant distance below the decay region.

The procedure we use for simulating the decay is the following: First, apply the action as if there is no decay and determine the state $|\psi\rangle$. Then, generate a uniformly distributed

random number between 0 and 1. If the random number is less than p_{decay}, send the state to $|1\rangle$. If the random number is larger than p_{decay}, do not make any additional changes to the state.

RL approach

Given this time-dependent quantum environment, we are trying to test whether by modifying the RL agent by taking time into account, we will achieve higher fidelity in the preparation of the qubit state.

We again compare the performance of two agents. One of them has knowledge only of an RL state, consisting of the angles (θ, ϕ) on the Bloch sphere. We will call it the **"static" agent**. The other agent will also use the time t as an input to the neural network. We will call it the **"dynamic" agent**. Both agents will operate on the same qubit subject to the time-dependent decay. They will also have identical neural network architecture (listed in Table A.12), with the only difference being that the "dynamic" agent has an additional input neuron containing the time t.

We will not manually transform the angles and time with sine and cosine functions with the appropriate periods. Instead, our first layer in the neural network will have a sine-nonlinearity, which will be applied to all neurons, so that the neural network could discover the periods on its own if necessary. Note that the transformation from quantum state $|\psi\rangle$ to the RL state is performed in such a way that $\theta \in [0, \pi], \phi \in (-\pi, \pi]$. This means that two identical quantum states cannot acquire different ϕ and θ coordinates.

The neural network has 7 output neurons corresponding to the 7 actions:

$$\mathcal{A} = \{\mathbf{1}, \hat{R}_{-x}\hat{R}_{-y}, \hat{R}_{-z}, \hat{R}_x, \hat{R}_y, \hat{R}_z\}. \tag{5.22}$$

The reward function we use is the fidelity $r = |\langle 0|\psi\rangle|^2$ (explained in Sec. 3.1.2).

Results

Both agents succeed in learning a policy that accomplishes a relatively high fidelity. However, the "dynamic" agent achieves a fidelity much closer to the maximum compared to the "static" agent. The fidelity of both agents over the training episodes is shown in Fig. 5.12.

By studying the specific policies of the agents, some of which are shown in Fig. 5.13, we notice that the "static" agent relies on chance, applying gates that try to rotate the qubit straightforwardly to the direction of the $|0\rangle$ state. On the other hand, the "dynamic" agent tries to rotate the qubit in such a way as to avoid the large-decay area, leading to a lower frequency of decay events.

It is interesting to understand what parameters control the observed difference between both fidelities – achieved by training the static and the dynamic agent.

First, we can widen the decay area by varying the parameter κ_1. By doing so, both agents find it harder to prepare the qubit, with the time-dependent agent keeping its advantage over the static agent. This can be observed in Fig. 5.14 (a), (b).

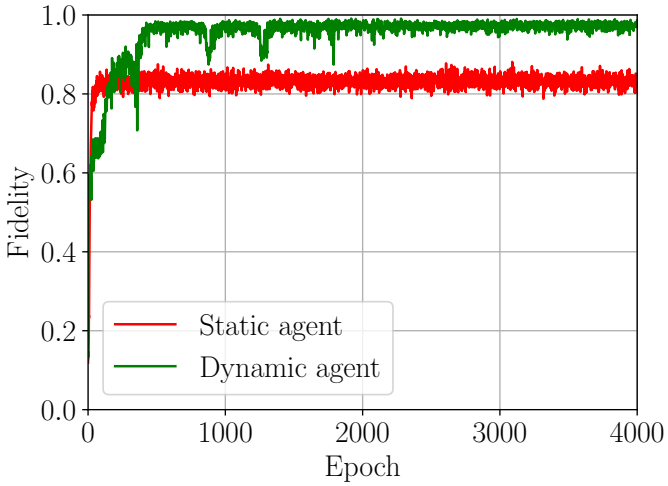

Figure 5.12: Fidelity achieved by the two agents in the qubit environment. Red: "static" agent, green: "dynamic" agent. The dynamic agent finds a control sequences which achieve nearly maximum fidelity. The static agent still manages to accomplish a relatively high fidelity. Model parameters as in Table A.10-A.12

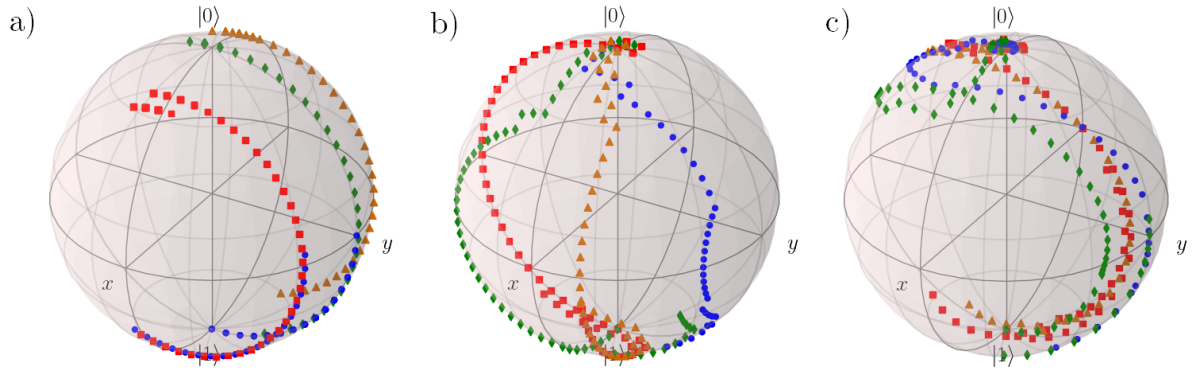

Figure 5.13: Path traced by the qubit states when controlled by: (a) – "static" agent, (b) – "dynamic" agent. The static agent almost always tries to move the qubit in the same way upwards and if it has luck, the qubit passes through the decay area and reaches the north pole. The dynamic agent offers more flexible solutions, which allow the qubit to pass through different sections of the decay area at different times. (c) plots the trajectory traced by the qubit states in a reference frame co-rotating with the dissipation, showing that the qubit passes through the no-decay area (which in this case is situated at $-x$).

Also, we can vary the total time $t = T\delta t$ that agents can operate on the qubit. Decreasing t leads to overall lower fidelities, with the static agent performing yet worse than the dynamic one. This is illustrated in Fig. 5.14 (c), (d). Note that we decrease t by keeping the total number of time steps T fixed, leading to a decrease of the length Δt of a single time step.

Finally, we should note that the results may vary due to random events in the training process. We observed the training of the same dynamic agents, starting from different seeds of the random number generators. We see that their learning curves are different from one another. Especially, their policies can be suddenly affected both in a positive and a negative direction, as can be seen in Fig. 5.15.

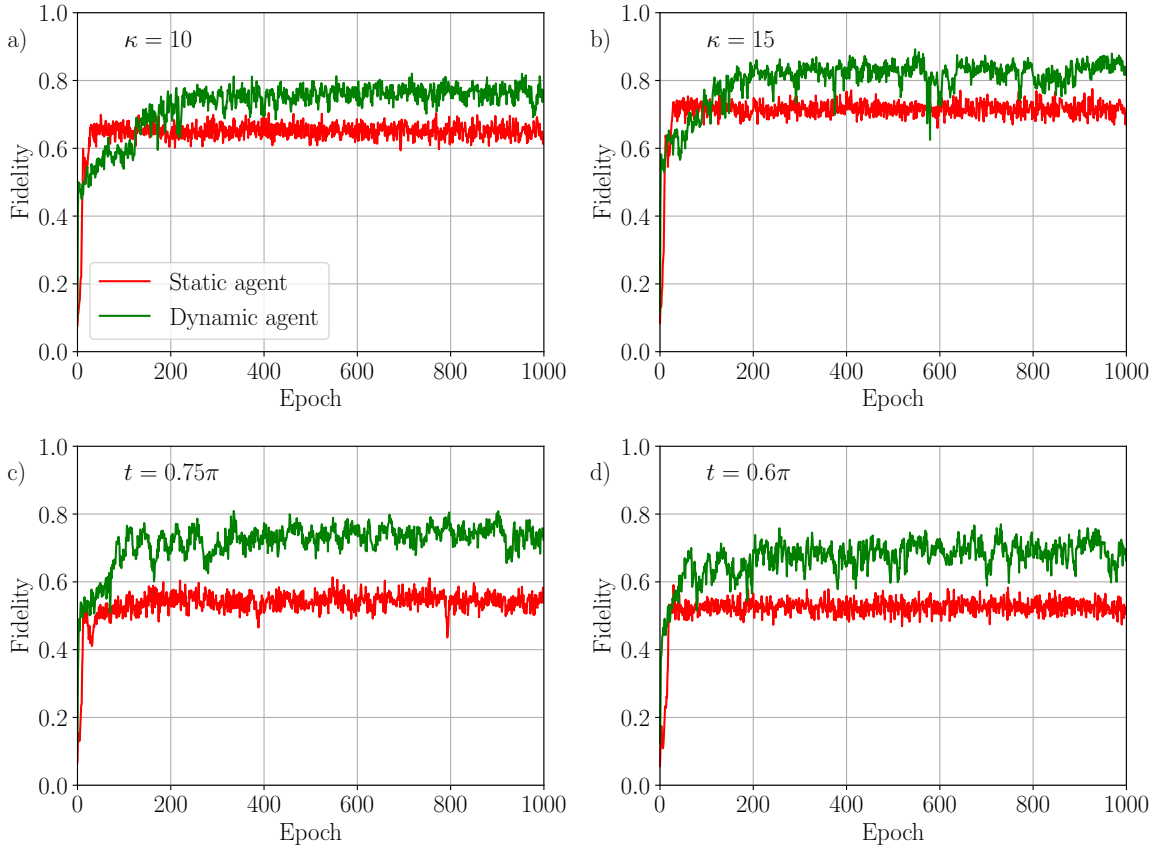

Figure 5.14: Fidelity achieved by the two agents in different settings. In (a) and (b), we vary the width of the decay area. In (c) and (d), we vary the time length. Both agents perform worse when given less time or having a wider decay area, with the time-dependent agent having advantage. The other model parameters as in Table A.10-A.12. We remind that the baseline values of κ and t we used are $\kappa = 20$ and $t = \pi$.

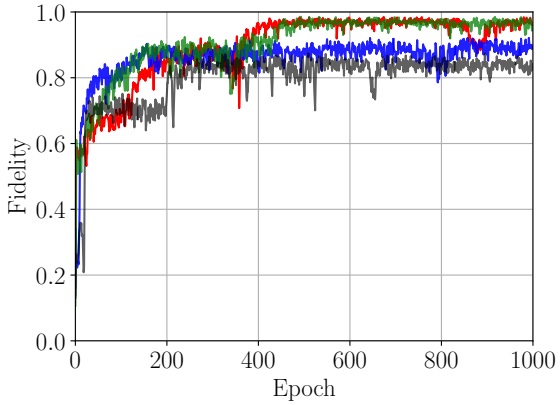

Figure 5.15: Training curves of dynamic agents with different seeds for the random number generators. Randomness affects the training of the RL agents. All training parameters are as in A.12.

Chapter 6

Conclusion

Reinforcement Learning (RL) is a powerful framework for searching optimal control policies in dynamical systems. Its advantages are that it can solve control problems autonomously, without requiring prior knowledge or an explicit model of the system in place. It is well-suited for controlling systems with continuous state spaces and systems whose dynamics exhibit nonlinearities. Also, it has the potential to generalize its knowledge to control modified or noisy environments.

In this thesis, we studied the problem of controlling time-dependent environments. In RL, the environment is required to be a Markov decision process. In that case, the agent determines its action based only on the state of the environment. When the environment itself is time-dependent, the RL framework needs to be generalized. We modified the RL framework by adding time as an additional element of the RL state. We used this approach to control three Hamiltonian systems and one quantum system.

In the first classical system, considered in Sec. 5.1.1 – a single potential well – a time-dependent RL agent managed to escape the quickest from the well by exploiting the time variations of the potential. Next, in Sec. 5.1.2, we moved on to a bit more complex system – a double well, where the time-agnostic agent could learn a policy that made the particle escape one of the potential minima and move to the vicinity of the second minimum, but as it passes through the same phase space state at different times, it is led to "unlearn" its policy. In contrast, the time-dependent agents learned to escape the first minimum and reach the second minimum successfully.

In the last classical system in sec. 5.1.3 – a two-dimensional potential, we failed to observe any significant benefits of including time to the RL state. This poses some new problems, which are interesting for further research. Firstly, it is interesting to understand whether some of the high-dimensional time-dependent systems have such intrinsic dynamics that knowledge of time would not be required at all in their control. Secondly, it would be interesting to research other ways in which we could extend the RL framework so that the agents could make better use of the time dependence.

In classical mechanics, there are other different control problems that we did not cover in this thesis but could be tackled by RL. A simple system that could be controlled in a very similar fashion to our example systems is the parametric oscillator. RL could be used in more complicated control problems, such as chaotic dynamical systems.

We also applied RL to prepare a two-level quantum system in ground state (see Sec. 5.2).

We created a simple model for a time-dependent decay of the system and we observed that our time-dependent agent could achieve significantly higher fidelity than an agent without knowledge of time. However, we should note that our agents are receiving rewards in the form of fidelity, which cannot be directly observed in a real quantum system. Although we have demonstrated the potential benefits of the time-dependent agent, it is currently operating in artificially created settings. The work on this problem could be developed in two main areas – to train agents using only information from measurable quantities, and to use environments with more realistic models of quantum dissipation.

Appendix A

Hyperparameters and parameters of the control problems considered

Mass of the particle	$m = 1$ kg
Mean depth of the potential well	$A_0 = 1$ J
Amplitude of depth oscillations	$A_1 = 0.5$ J
Frequency of depth oscillations	$\omega_1 = 2$ s^{-1}
Mean width of the potential well	$w_0 = 1$ m
Amplitude of width oscillations	$w_1 = 0.2$ m
Frequency of width oscillations	$\omega_1 = \sqrt{2}$ s^{-1}

Table A.1: Parameters of the physical system for the single potential well (Sec. 5.1.1)

Force applied	$F = 0.1$ N
Number of time steps	$T = 200$
Time length of the physical system	$t = 40$ s
Length of one time step	$\Delta t = 0.2$ s

Table A.2: Discretization and control parameters used by turning the single potential well into a control problem

Optimizer	Adam
Step size	lr $= 0.0001$
First momentum	$\beta_1 = 0.9$
Second momentum	$\beta_1 = 0.999$
Adam stability term	$\epsilon = 10^{-8}$
L2-regularization parameter	$\epsilon_{l2} = 0.01$
Returns discount factor	$\gamma = 1$
Initial temperature of the entropy term	$T_0 = 1$
Temperature decay constant	$N_{decay} = 400$
MC-sample	$N_{\text{MC}} = 512$
Neural network hidden layers sizes	128, 128, 64

Table A.3: Hyperparameters of the PG algorithm in the single potential well

Mass of the particle	$m = 1$ kg
Mean depth of the potential well	$A_0 = 1$ J
Amplitude of depth oscillations	$A_1 = 0.9$ J
Frequency of depth oscillations	$\omega = 0.5$ s^{-1}

Table A.4: Parameters of the physical system for the double well (Sec. 5.1.2)

Force applied	$F = 0.2$ N in the static case, $F = 0.15$ N in the dynamic case
Number of time steps	$T = 200$
Time length of the physical system	$t = 40$ s
Length of one time step	$\Delta t = 0.2$ s
Kinetic term weight	$k = 2$
Position tolerance	$x_{tol} = 0.112$ m
Momentum tolerance	$p_{tol} = 0.224$ kg.m/s

Table A.5: Discretization and control parameters used By turning the double well potential into a control problem

Optimizer	Adam
Step size	lr $= 0.0001$
First momentum	$\beta_1 = 0.9$
Second momentum	$\beta_1 = 0.999$
Adam stability term	$\epsilon = 10^{-8}$
L2-regularization parameter	$\epsilon_{l2} = 0.01$
Returns discount factor	$\gamma = 1$
Initial temperature of the entropy term	$T_0 = 1$
Temperature decay constant	$N_{decay} = 400$
MC-sample	$N_{\text{MC}} = 512$
Neural network hidden layers sizes	128, 128, 64

Table A.6: Hyperparameters of the PG algorithm in the double well problem

Mass of the particle	$m = 1$ kg
Amplitude coefficient of the potentials	$A = 2$ J/m
Potential width scale	$r_0 = 1$ m
Thinness of the shortcuts	$\kappa = 10$
Angular velocity of the rotating shortcut	$\omega = 1$ s^{-1}
Drag coefficient	$\beta = 0.5$ s^{-1}

Table A.7: Parameters of the physical system for the 2D potential (Sec. 5.1.3)

Force applied	$F = 1$ N
Number of time steps	$T = 100$
Time length of the physical system	$t = 20$ s
Length of one time step	$\Delta t = 0.2$ s

Table A.8: Discretization and control parameters used when turning the 2D potential into a control problem

Optimizer	Adam
Step size	lr $= 0.001$
First momentum	$\beta_1 = 0.9$
Second momentum	$\beta_1 = 0.999$
Adam stability term	$\epsilon = 10^{-8}$
L2-regularization parameter	$\epsilon_{l2} = 0.01$
Returns discount factor	$\gamma = 1$
Initial temperature of the entropy term	$T_0 = 0.01$
Temperature decay constant	$N_{decay} = 400$
MC-sample	$N_{\text{MC}} = 512$
Neural network hidden layers sizes	256, 128

Table A.9: Hyperparameters of the PG algorithm in the two-dimensional system

Probability amplitude	$p_0 = 0.5$
Decay region θ-distribution	$\kappa_1 = 20$
Decay region ϕ-distribution	$\kappa_2 = 1$
Angular velocity of the decay region	$\omega = 2.5\ \text{s}^{-1}$

Table A.10: Parameters characterizing the decay of the two-level system discussed in Sec. 5.2

Number of time steps	$T = 60$
Time length of the physical system	$t = \pi$ s
Length of one time step	$\Delta t = \pi/60$ s

Table A.11: Discretization and control parameters for the quantum two-level system

Optimizer	Adam
Step size	lr $= 0.001$
First momentum	$\beta_1 = 0.9$
Second momentum	$\beta_1 = 0.999$
Adam stability term	$\epsilon = 10^{-8}$
L2-regularization parameter	$\epsilon_{l2} = 0.001$
Returns discount factor	$\gamma = 1$
MC-sample	$N_{\text{MC}} = 512$
Neural network hidden layers sizes	512, 256, 64

Table A.12: Hyperparameters of the PG algorithm used in the control of the quantum two-level system

Appendix B

Additional Experiments with the Two-Dimensional Potential

We have compared the agents from Sec. 5.1.3 with and without knowledge of time in a modification of the two-dimensional potential.

The new potential takes the form

$$V(\rho,\theta) = \begin{cases} A_1 \sin^2(\rho/\rho_0)\,\Theta(\theta+\omega_1 t), & \rho \in [\pi\rho_0, 2\pi\rho_0], \\ A_2 \sin^2(\rho/\rho_0)\,\Theta(\theta+\omega_2 t), & \rho \in [2\pi\rho_0, 3\pi\rho_0], \\ 0, & \rho < \pi \cup \rho > 3\pi, \end{cases} \tag{B.1}$$

where Θ is defined by Eq. (5.12). This potential contains only two barriers. However, the shortcuts in different barriers rotate with different angular frequencies (ω_1 and ω_2). The task is the same as in Sec. 5.1.3 – the particle has to reach the zone $\rho = 3\pi\rho_0$. The RL agents have the same structure as in Sec. 5.1.3 and are being given the same reward $r = -|\rho - 3\pi\rho_0|$.

The result of the training is shown in Fig. B.1. We did not observe any benefit from adding time as a parameter to the agent. On the contrary, the time-agnostic agent achieved a final

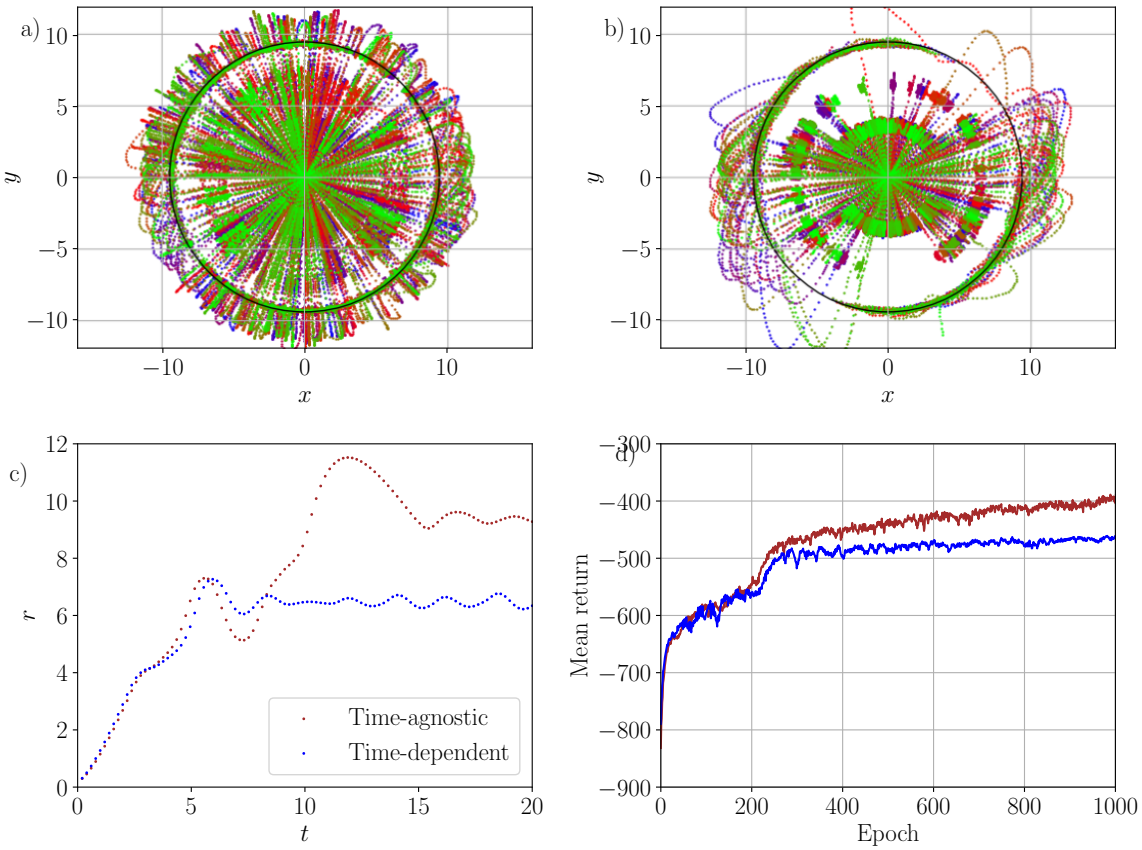

Figure B.1: Performance of the RL agents in the modified time-dependent potential. (a), (b) – trajectories of the controlled particles; (a) – agent without knowledge of time, (b) – agent with knowledge of time; (c) – the coordinate of the controlled particles of an arbitrary trajectory in the batch as a function of time; (d) – training curves of different agents. In (a) and (b) the goal $r = 3\pi$ is designated by a black circle. Both agents sometimes get stuck before the second barrier. The time-agnostic agent achieved somewhat better control over the particle with more trajectories reaching the goal, leading to a higher mean return. The constants for the simulation are given in App. B.1.

B.1 Constants for the simulations with the modified 2D potential

Mass of the particle	$m = 1$ kg
Amplitude of the first barrier	$A_1 = 3$ J
Amplitude of the first barrier	$A_2 = 5$ J
Potential width scale	$r_0 = 1$ m
Thinness of the shortcuts	$\kappa = 10$
Angular velocity of the first shortcut	$\omega_1 = 1$ s^{-1}
Angular velocity of the second shortcut	$\omega_2 = \sqrt{2}$ s^{-1}
Drag coefficient	$\beta = 0.75$ s^{-1}

Table B.1: Parameters of the physical system

Force applied	$F = 1.5$ N
Number of time steps	$T = 100$
Time length of the physical system	$t = 20$ s
Length of one time step	$\Delta t = 0.2$ s

Table B.2: Discretization and control parameters used when turning the modified 2D potential into a control problem

Optimizer	Adam
Step size	lr $= 0.001$
First momentum	$\beta_1 = 0.9$
Second momentum	$\beta_1 = 0.999$
Adam stability term	$\epsilon = 10^{-8}$
L2-regularization parameter	$\epsilon_{l2} = 0.01$
Returns discount factor	$\gamma = 1$
Initial temperature of the entropy term	$T_0 = 0.01$
Temperature decay constant	$N_{decay} = 400$
MC-sample	$N_{\text{MC}} = 512$
Neural network hidden layers sizes	256, 128

Table B.3: Hyperparameters of the PG algorithm in the modified two-dimensional system

Bibliography

[1] J. McCarthy. What is artificial intelligence. 2007. doi:10.1142/9789812816818_0009.

[2] J. Koomey, S. Berard, M. Sanchez, and H. Wong. Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing*, 33(3):46–54, 2010. doi:10.1109/MAHC.2010.28.

[3] H. Matsubara, H. Iida, R. Grimbergen, et al. Chess, shogi, go, natural developments in game research. *ICCA journal*, 19(2):103–112, 1996. doi:10.3233/ICG-1996-19208.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi:10.1038/nature16961.

[5] T. G. Brown, A. Statman, and C. Sui. Public Debate on Facial Recognition Technologies in China. *MIT Case Studies in Social and Ethical Responsibilities of Computing*, (Summer 2021), aug 10 2021. doi:10.21428/2c646de5.37712c5c.

[6] M. Westerlund. The emergence of deepfake technology: A review. *Technology innovation management review*, 9(11), 2019. doi:10.22215/timreview/1282.

[7] OpenAI. Gpt-4 technical report. 2023. doi:10.48550/arXiv.2303.08774.

[8] D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, T. Köppe, K. Millikin, S. Gaffney, S. Elster, J. Broshear, C. Gamble, K. Milan, R. Tung, M. Hwang, T. Cemgil, M. Barekatain, Y. Li, A. Mandhane, T. Hubert, J. Schrittwieser, D. Hassabis, P. Kohli, M. Riedmiller, O. Vinyals, and D. Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, June 2023. doi:10.1038/s41586-023-06004-9.

[9] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, February 2022. doi:10.1038/s41586-021-04301-9.

[10] M. Krenn, J. Landgraf, T. Foesel, and F. Marquardt. Artificial intelligence and machine learning for quantum technologies. *Physical Review A*, 107(1), January 2023. doi:10.1103/physreva.107.010101.

[11] F. Metz and M. Bukov. Self-correcting quantum many-body control using reinforcement learning with tensor networks. 2023. doi:10.48550/arXiv.2201.11790.

[12] J. Yao, M. Bukov, and L. Lin. Policy gradient based quantum approximate optimization algorithm. 107:605–634, 20–24 Jul 2020. https://proceedings.mlr.press/v107/yao20a.html.

[13] J. Yao, L. Lin, and M. Bukov. Reinforcement learning for many-body ground-state preparation inspired by counterdiabatic driving. *Phys. Rev. X*, 11:031070, Sep 2021. doi:10.1103/PhysRevX.11.031070.

[14] M. Bukov, A. G. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta. Reinforcement learning in different phases of quantum control. *Physical Review X*, 8(3):031086, 2018. doi:10.1103/PhysRevX.8.031086.

[15] V. V. Sivak, A. Eickbusch, H. Liu, B. Royer, I. Tsioutsios, and M. H. Devoret. Model-free quantum control with reinforcement learning. *Phys. Rev. X*, 12:011059, Mar 2022. doi:10.1103/PhysRevX.12.011059.

[16] X.-M. Zhang, Z. Wei, R. Asad, X.-C. Yang, and X. Wang. When does reinforcement learning stand out in quantum control? a comparative study on state preparation. *npj Quantum Information*, 5(1), October 2019. doi:10.1038/s41534-019-0201-8.

[17] R. Porotti, D. Tamascelli, M. Restelli, and E. Prati. Coherent transport of quantum states by deep reinforcement learning. *Communications Physics*, 2(1), June 2019. doi:10.1038/s42005-019-0169-x.

[18] M. Bukov. Reinforcement learning for autonomous preparation of floquet-engineered states: Inverting the quantum kapitza oscillator. *Phys. Rev. B*, 98:224305, Dec 2018. doi:10.1103/PhysRevB.98.224305.

[19] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven. Universal quantum control through deep reinforcement learning. *npj Quantum Information*, 5(1), April 2019. doi:10.1038/s41534-019-0141-3.

[20] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath. Quantum error correction for the toric code using deep reinforcement learning. *Quantum*, 3: 183, September 2019. doi:10.22331/q-2019-09-02-183.

[21] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert. Reinforcement learning decoders for fault-tolerant quantum computation. *Machine Learning: Science and Technology*, 2(2):025005, December 2020. doi:10.1088/2632-2153/abc609.

[22] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret. Real-time quantum error correction beyond break-even. *Nature*, 616(7955):50–55, mar 2023. doi:10.1038/s41586-023-05782-6.

[23] Y. Baum, M. Amico, S. Howell, M. Hush, M. Liuzzi, P. Mundada, T. Merkh, A. R. Carvalho, and M. J. Biercuk. Experimental deep reinforcement learning for error-robust gate-set design on a superconducting quantum computer. *PRX Quantum*, 2:040324, Nov 2021. doi:10.1103/PRXQuantum.2.040324.

[24] A. Bolens and M. Heyl. Reinforcement learning for digital quantum simulation. *Phys. Rev. Lett.*, 127:110502, Sep 2021. doi:10.1103/PhysRevLett.127.110502.

[25] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022. doi:10.1017/9781108380690.

[26] P. Hamadanian, M. Schwarzkopf, S. Sen, and M. Alizadeh. Demystifying reinforcement learning in time-varying systems. 2023. doi:10.48550/arXiv.2201.05560.

[27] Y. Han, R. Solozabal, J. Dong, X. Zhou, M. Takac, and B. Gu. Learning to control under time-varying environment. 2022. doi:10.48550/arXiv.2206.02507.

[28] Y. Chandak, G. Theocharous, S. Shankar, M. White, S. Mahadevan, and P. S. Thomas. Optimizing for the future in non-stationary mdps. 2020. doi:10.48550/arXiv.2005.08158.

[29] E. Lecarpentier and E. Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning, extended version. 2020. doi:10.48550/arXiv.1904.10090.

[30] I. Szita, B. Takács, and A. Lörincz. ϵ-mdps: Learning in varying environments. *J. Mach. Learn. Res.*, 3(null):145–174, mar 2003. ISSN 1532-4435. doi:10.1162/153244303768966148.

[31] B. C. Hall. *The Spectral Theorem for Unbounded Self-Adjoint Operators*, pages 201–226. Springer New York, New York, NY, 2013. ISBN 978-1-4614-7116-5. doi:10.1007/978-1-4614-7116-5_10.

[32] B. R. Desai. *Quantum Mechanics with Basic Field Theory*. Cambridge University Press, 2009. doi:10.1017/CBO9780511813979.

[33] S. Moe, A. M. Rustad, and K. G. Hanssen. Machine learning in control systems: An overview of the state of the art. pages 250–265, 2018. doi:10.1007/978-3-030-04191-5_23.

[34] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. Adaptive Computation and Machine Learning series. Bradford Books, Cambridge, MA, 2 edition, November 2018. http://incompleteideas.net/book/the-book-2nd.html.

[35] P. Tashev. Developing artificial intelligence agents to manipulate quantum entanglement. Master's thesis, Faculty of Mathematics and Informatics, Sofia University, 2022. https://quantum-dynamics.phys.uni-sofia.bg/files/master_thesis_tashev.pdf.

[36] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com.

[37] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, May 2019. doi:10.1016/j.physrep.2019.03.001.

[38] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. page 78, 2004. doi:10.1145/1015330.1015435.

[39] P. Prince and J. Dormand. High order embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75, 1981. ISSN 0377-0427. doi:https://doi.org/10.1016/0771-050X(81)90010-3.

[40] S. S. Schoenholz and E. D. Cubuk. Jax, m.d. a framework for differentiable physics*. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124016, dec 2021. doi:10.1088/1742-5468/ac3ae9.