UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
Control Systems Engineering

"Reinforcement Learning for Robust Quantum State Stabilization"

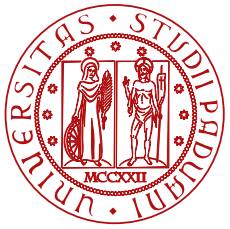
Relatore: Prof. / Dott Francesco Ticozzi

Laureando: Manuel Guatto

Correlatore: Prof. /Dott Gian Antonio Susto

ANNO ACCADEMICO 2022 – 2023

Data di laurea 21/04/2023



Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control Systems Engineering

Reinforcement Learning for Robust Quantum State Stabilization

Master Candidate

Manuel Guatto

Student ID 2022574

Supervisor

Prof. Francesco Ticozzi

University of Padova

Co-supervisor

Prof. Gian Antonio Susto

University of Padova

Academic Year
2022/2023

To everyone who wants to pursue
its passion

Abstract

This thesis work examines the application of state-of-the-art reinforcement learning algorithms to the quantum state stabilization problem, specifically focusing on the robustness w.r.t. different types of noise.

In the first part, this work aims to provide the basics of both the quantum mechanics formalism and the reinforcement learning framework. The second part of the thesis focuses on the development of three different reinforcement learning methods with the purpose of stabilizing a target state for a quantum system via feedback control. The algorithms that follows from these setups are trained without any noise acting on the system, but their robusteness is then tested by adding different types of quantum noise to the system dynamics. An in-depth numerical analysis of the performance of the presented methods for a significant case study is developed, and the results are compared in order to identify the approach that is least sensitive to the noise addition.

Our results provide some valuable insights about the robustness of the reinforcement learning framework applied to quantum state stabilization and clearly indicate that model-free approaches as the best candidates for a controlling quantum systems in uncertain environments. This work also suggests interesting new research directions, including the scalability of the performances when dealing with quantum systems of increasing size.

Sommario

Questo lavoro di tesi esamina l'applicazione dei più moderni algoritmi di apprendimento per rinforzo al problema della stabilizzazione degli stati quantistici, concentrandosi in particolare sulla robustezza rispetto a diversi tipi di rumore. Nella prima parte, questo lavoro si propone di fornire le basi sia del formalismo della meccanica quantistica che della struttura degli algoritmi di apprendimento per rinforzo. La seconda parte della tesi si concentra sullo sviluppo di tre diversi metodi di apprendimento con l'obiettivo di stabilizzare uno stato target per un sistema quantistico attraverso attraverso un controllo in retroazione. Gli algoritmi che ne derivano sono addestrati senza che alcun rumore agisca sul sistema. La loro robustezza viene poi testata aggiungendo diversi tipi di rumore quantistico alla dinamica del sistema. Un'analisi numerica approfondita delle prestazioni dei metodi presentati per un caso studio significativo, e i risultati, vengono confrontati al fine di identificare l'approccio meno sensibile all'aggiunta del rumore. I risultati forniscono alcune preziose indicazioni sulla robustezza del sistema di apprendimento per rinforzo applicato alla stabilizzazione degli stati quantistici e indicano chiaramente che gli approcci model-free sono i migliori candidati per il controllo dei sistemi sistemi quantistici in ambienti incerti. Questo lavoro suggerisce anche nuove interessanti direzioni di ricerca, tra cui la scalabilità delle prestazioni quando ci si interfaccia con sistemi quantistici di dimensioni crescenti.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 General Overview

In this thesis, we will explore the use of the reinforcement learning framework in quantum feedback control. The research aims to develop a model free Reinforcement Learning (RL) framework for the design of a feedback law and study the robustness of the latter to the presence of quantum noise. In order to pursue this goal we will develop a simulated (RL) environment in which we are able to train and test various setups. The study is significant and yields a novel viewpoint on the subject because, even if many quantum control problems have already been addressed with reinforcement learning setups in the existing literature, there are no direct comparisons about the robustness of different reinforcement setups. This work aims to provide a first contribute in this direction to the field of the quantum control using reinforcement learning algorithms.

In this introduction, we will provide a brief overview of the background, significance, and objectives of the study. We will first provide a perspective on the study of the quantum control and reinforcement learning, highlighting key milestones and notable findings. Finally, we will present the general structure of this thesis work.

1.2 Quantum Control

Quantum control refers to the ability to manipulate the behavior of quantum systems, which are inherently unpredictable and complex. In recent years, researchers have made significant progress in understanding and harnessing the potential of quantum control to improve technology and advance scientific discovery. The field of quantum control involves the design and implementation of techniques to manipulate quantum systems, such as atoms, molecules, and photons, with a high degree of precision and accuracy. It is a multidisciplinary field that draws on concepts from quantum mechanics, control theory, and information science, and has broad applications in areas such as quantum computing, quantum communication, and quantum sensing. The study of quantum control is therefore of great importance in the ongoing development of quantum technologies, and promises to revolutionize the way we process, store, and transmit information in the coming years.

1.3 Reinforcement Learning

Reinforcement learning is a type of machine learning that focuses on the interaction between an agent and its environment. In reinforcement learning, an agent learns to make decisions by receiving feedback from the environment in the form of rewards or penalties. The goal of reinforcement learning is to train an agent to make optimal decisions in a given environment by maximizing the total reward it receives over time.

Reinforcement learning has become increasingly popular in recent years due to its ability to solve complex problems that are difficult to tackle using traditional machine learning techniques. It has been successfully applied to a wide range of domains, including robotics, game playing, natural language processing, and recommendation systems.

One of the key advantages of reinforcement learning is its ability to learn from experience. As the agent interacts with the environment, it learns which actions lead to positive outcomes and which lead to negative outcomes. Over time, the agent becomes better at making decisions, and can even learn to adapt

to changing environments.

Reinforcement learning is a rapidly evolving field, with new algorithms and techniques being developed all the time. As the field continues to mature, it holds great promise for solving some of the most challenging problems in artificial intelligence and beyond.

1.4 Intersection Area: Quantum control with RL

RL can be used to design control protocols that optimize the performance of a quantum system based on a specific goal.

In quantum control with RL, the quantum system is modeled as an environment, and the control actions are taken by an RL agent. The agent receives feedback in the form of a reward signal that is computed based on the performance of the quantum system. The objective is to learn a control policy that maximizes the expected reward over time.

This field of study, although not much beaten down, presents important contributions to the world of quantum control, as examples we can cite some results in the quantum-state manipulations [8] [14] [21].

1.5 Structure of the thesis

This thesis work presents the following chapters:

1. Introduction
    - In this first chapter we will provide a general overview about the research topic and some insights about the main fields: Reinforcement Learning and Quantum control.

2. Quantum Mechanics
    - In this chapter we will present the formalism used to describe the quantum mechanics from a quantum information and computation point of view. Moreover we will deal with some types of noises used in the simulations.

3. Reinforcement Learning

    - In this chapter we will present the general RL framework. The chapter will include the mathematical definition of the environments, some concepts about the main algorithms and a section related to the Proximal Policy algorithm, the one used in the simulations.

4. Quantum State Stabilization: A discrete-time framework

    - In this chapter we will present both the mathematical model and the simulated one. We will delve deeper into the dynamics of the system and we will discuss the details about the simulations.

5. Simulations

    - In this chapter we will discuss both the train and the tested model. Moreover we will present and analyze the results of the simulations.

6. Conclusions

    - In this chapter we will summarize the work's results discussing some limitations and future steps.

2

Quantum Mechanics

"Quantum mechanics: Real Black Magic Calculus"

Albert Einstein

2.1 Introduction

Quantum mechanics is a fundamental theory of physics that describes the behavior of matter and energy at the smallest scales, such as atoms and subatomic particles. It was developed in the early 20th century to explain the strange and counterintuitive properties of quantum systems, such as superposition, entanglement, and wave-particle duality.

In recent decades, the field of quantum information and computation has emerged, which studies how these quantum properties can be harnessed to perform computational and communication tasks that are beyond the capabilities of classical computers. This has led to the development of quantum algorithms, quantum cryptography, and quantum communication protocols, among other applications.

However, harnessing the power of quantum systems is not without its challenges, and quantum control has become an increasingly important area of

research. This involves developing techniques to manipulate and control quantum systems, in order to achieve desired outcomes or mitigate unwanted effects such as decoherence.

The history of quantum mechanics and its related fields is rich and complex, involving many pioneering scientists such as Max Planck, Albert Einstein, Niels Bohr, Werner Heisenberg, Erwin Schrödinger, and Richard Feynman, among others. Their contributions have led to a profound understanding of the quantum world and opened up a wealth of possibilities for the future of technology and science.

The main references for this chapter are the followings [13][5].

2.2 Fundamentals concepts of Quantum mechanics

2.2.1 First postulate: State Space

In this section we will present the first postulate of the quantum mechanics and we will discuss some of its implications.

This postulate will set up the arena in which the quantum mechanics will take place. Firstly we recall fundamental concept of *Hilbert space*.
Mathematically, a Hilbert space is defined as a complete, separable inner product space that is also a vector space over the field of complex numbers. Formally:

Definition 2.2.1 (Hilbert space)**.** *A Hilbert space $\mathcal{H}$ is a complete metric space, with a metric defined by the inner product, that satisfies the following axioms:*

- *$\mathcal{H}$ is a complex vector space, with addition and scalar multiplication defined as usual.*
- *$\mathcal{H}$ is equipped with an inner product, denoted by $\langle \cdot, \cdot \rangle$, which is a complex-valued function that satisfies:*
  - *$\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle^*$, where * denotes the complex conjugate.*
  - *$\langle \alpha\mathbf{u} + \beta\mathbf{v}, \mathbf{w} \rangle = \alpha\langle \mathbf{u}, \mathbf{w} \rangle + \beta\langle \mathbf{v}, \mathbf{w} \rangle$, where α and β are complex numbers.*
  - *$\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$, with equality if and only if $\mathbf{u} = \mathbf{0}$.*
- *$\mathcal{H}$ is complete, meaning that every Cauchy sequence of vectors in $\mathcal{H}$ converges to a unique vector in $\mathcal{H}$.*

Now we can define the first postulate.

Postulate 2.2.1 (Postulate 1)**.** *Associated to any isolated physical system is a complex vector space with inner product (Hilbert space) known as state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.*

Once we have defined the formalism of a general state space, we can define the simplest quantum mechanical system which is called *qubit*. The qubit lives in a two dimensional complex Hilbert space $\mathcal{H}$ identified as $\mathbb{C}^2$. Without loss of generality we can define an orthonormal base in order to describe the state space:

$$|0\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad ,$$
$$|1\rangle \doteq \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad .$$

With this setup it is possible to built any vector in the state space:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad ,$$

where α and β are complex numbers ($\alpha, \beta \in \mathbb{C}$) which have to respect the *normalization condition*:

$$|\alpha^2| + |\beta^2| = 1 \quad .$$

2.2.2 Second postulate: Evolution

In this section we will present the second postulate of the quantum mechanics, moreover we will discuss its implications trying, at the same time, to clarify some concepts.

Postulate 2.2.2 (Postulate 2)**.** *The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time $t_1$is related to $|\psi'\rangle$ of the system at time t_2 by a unitary operator $\mathcal{U}$ which depends only on the times t_1 and t_2.*

The aim of this postulate is to describe how the quantum states of a *closed* quantum system are related. In order to understand as best as possible this

postulate we want to clarify some terminology, in particular we will focus on the definition of closed system and on the definition of unitary operator.

Definition 2.2.2 (Closed System)**.** *A system is said to be closed when does not interchange information (i.e. energy and/or matter) with another system.*

Definition 2.2.3 (Unitary Operator)**.** *A unitary operator is a bounded linear operator $\mathcal{U} : H \to H$ on a Hilbert space H that satisfies:*

- $\mathcal{U}^\dagger \mathcal{U} = I$,
- $\mathcal{U}\mathcal{U}^\dagger = I$,

where $\mathcal{U}^\dagger$ is the adjoint of $\mathcal{U}$, and $I : H \to H$ is the identity operator.

In reality all systems interacts at least with another system, except for the whole universe. Nevertheless there exist some systems which are described by a unitary evolution with a good approximation. This postulate (Post. 2.2.2) can be refined in order to describe the evolution of quantum systems in *continuous time*.

Postulate 2.2.3 (Postulate 2 (Continuous Time))**.** *The time evolution of the state of a a closed quantum system is described by the Schrödinger equation:*

$$i\hbar \frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle \quad .$$

The $\hbar$ is called *Planck's constant* whose values has to be determined experimentally, the H is a fixed Hermitian operator called Hamiltonian. From this second version of the Postulate we can clearly argue that if we know the Hamiltonian H of the system then we can understand its dynamics. Since the Hamiltonian is an Hermitian operator it has a spectral decomposition:

$$H = \sum_E E\,|E\rangle\langle E| \quad ,$$

with eigenvalues E and corresponding normalized eigenvectors $|E\rangle$. The states $|E\rangle$ are called *energy eigenstates* and the eigenvalues E are called *energies*. The lowest energy is called *ground state energy* and the associated eigenstate is called *ground state*.

To provide a link between the postulate Post 2.2.2 and postulate Post 2.2.3 we have to write the solution of the Schrödinger equation which is the following:

$$|\psi(t_2)\rangle = e^{\frac{-iH(t_2-t_1)}{\hbar}} |\psi(t_1)\rangle = U(t_1,t_2)\,|\psi(t_1)\rangle \quad ,$$

from this we can define:

$$e^{\frac{-iH(t_2-t_1)}{\hbar}} = U(t_1,t_2) \quad .$$

2.2.3 Third Postulate: Quantum Generalized Measurements

In this section we will present the third postulate of the quantum mechanics, it is about the quantum measurements. After having discussed the postulate we will present some measurement operators.

Postulate 2.2.4 (Postulate 3)**.** *Quantum measurements are described by a collection* $\{M_m\}$ *of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcome that may occur in the experiment. If the state of the quantum system is* $|\psi\rangle$ *immediately before the measurement then the probability that result m occurs is given by:*

$$p(m) = \langle\psi|\, M_m^\dagger M_m\, |\psi\rangle \quad ,$$

and the state of the system after the measurement is:

$$\frac{M_m\,|\psi\rangle}{\sqrt{\langle\psi|\, M_m^\dagger M_m\, |\psi\rangle}} \quad .$$

The measurement operators satisfy the completeness equation:

$$\sum_m M_m^\dagger M_m = I$$

This postulate is fundamental since it describe not just the effect of the measure on the state but also the constraint that a set of operators must respect in order to be a set of measurement operators. Pursuing in the understanding of this constraint we can notice that the completeness equation represents the fact

that the probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle \quad .$$

A simple but important example of a measurement is the measurement of a qubit in the *computational basis*. It is a measurement of a qubit with two possible outcomes defined by the two measurement operators:

$$M_0 = |0\rangle\langle 0| \quad , \quad M_1 = |1\rangle\langle 1| \quad .$$

We can notice that this type of measurement respects the completeness equation indeed:

$$M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1 = I \quad .$$

Another important class of measurements is called *Projective Measurements*.

Definition 2.2.4 (Projective Measurements)**.** *A projective measurement is described by an observable, M, a Hermitian operator on the state space of the system being observed. The observable has a spectral decomposition:*

$$M = \sum_m m P_m \quad ,$$

where P_m is the projector onto the eigenspace of M with eigenvalue m. The possible outcomes of the measurement correponds to the eigenvalues, m, of the observable. Upon measuring the state $|\psi\rangle$, the probability of receiving result m is given by:

$$p(m) = \langle\psi| P_m |\psi\rangle$$

Given that outcome m occured the state of the quantum system immediately after the measurement becomes:

$$\frac{P_m |\psi\rangle}{p(m)} \quad .$$

A key aspect of the projective measurements is that them drive the quantum state to the *collapse* in other words the measurement action will destroy the superposition. Let us consider a state $|\psi\rangle$ as initial state, from definition Def. 2.2.4 we know that the state after having measured the state and having received outcome m is:

$$|\psi_m\rangle = \frac{(P_m |\psi\rangle)}{\sqrt{\langle\psi| P_m |\psi\rangle}} \quad .$$

Applying P_m to $|\psi_m\rangle$ does not change it, so we have that $\langle\psi| P_m |\psi\rangle = 1$, and we have proved that by repeating the measurement the outcome will be always m each time without changing the state.
For the projective measurements it is easy to compute the expectation of the measurement:

$$\begin{aligned}\mathbb{E}(M) &= \sum_m m p(m) \\ &= \sum_m m \langle\psi| P_m |\psi\rangle \\ &= \langle\psi| (\sum_m m P_m) |\psi\rangle \\ &= \langle\psi| M |\psi\rangle \quad .\end{aligned}$$

2.2.4 Fourth Postulate: Composite systems

In this section we will present the fourth postulate of the quantum mechanics, moreover we will discuss its implications trying, at the same time, to clarify some concepts. Firstly we will define the tensor product, which will be very useful as we will dealing with this postulate.

Definition 2.2.5 (Tensor Product)**.** *Let V and W be two vector spaces over a field F. The tensor product $V \otimes W$ is the vector space generated by the formal symbols $v \otimes w$, where $v \in V$ and $w \in W$, subject to the following relations:*

$$\begin{aligned}(v_1 + v_2) \otimes w &= v_1 \otimes w + v_2 \otimes w \\ v \otimes (w_1 + w_2) &= v \otimes w_1 + v \otimes w_2 \\ (cv) \otimes w &= v \otimes (cw) = c(v \otimes w)\end{aligned}$$

where $v_1, v_2 \in V$, $w_1, w_2 \in W$, and $c \in F$.

Now we introduce the fourth postulate of quantum mechanics.

Postulate 2.2.5 (Postulate 4)**.** *The state space of a composite physical system is the tensor product of the state space of the component physical systems. Moreover, if we have systems numbered 1 to n, and system number i is prepared in the state $|\psi_i\rangle$, the the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \cdots \otimes |\psi_n\rangle$.*

The aim of this postulate is to describe how the quantum states interacts to built a composite system. The postulate argue that if we have a composite

system made up of multiple subsystems, the state space of the composite system is the tensor product of the state spaces of the individual subsystems. The tensor product of two state spaces represents all possible combinations of the states of the two subsystems.
The statement also describes how to find the joint state of the total system if we know the states of the individual subsystems. If we have n systems, each with its own state $|\psi_i\rangle$, the joint state of the total system is given by the tensor product of the individual states: $|\psi_1\rangle \otimes |\psi_2\rangle \cdots \otimes |\psi_n\rangle$. This joint state represents all possible combinations of the individual states, and it is used to calculate the probabilities of various measurement outcomes on the composite system.

2.3 Density operator

2.3.1 Definition

Until now we have considered just a single quantum system, but if we want to deal with real quantum systems we have to introduce the concept of *ensemble* of quantum states and further the concept of *density operator*.
Let us consider a quantum system that could be in one of n possible quantum states, and let us call these state vectors $|\psi_i\rangle$, where i is the index. To every possible quantum state is associated a real positive number p_i which represents the probability that the quantum system is in the state $|\psi_i\rangle$. Clearly all the p_i has to respect:

$$\sum_i p_i = 1 \quad .$$

We call the set of pairs $\{(p_i, |\psi_i\rangle)\}$ an ensemble of quantum states. It is possible to distinguish three main cases of ensembles:

- Pure State:
  - Exist state i of the ensemble whose associated probability is: $p_i = 1$
- Completely mixed state:
  - The probabilities of every state in the ensemble are all equal: $p_i = p_j \quad \forall i, j$
- Mixed State:
  - Whether the ensemble is not a pure state or a completely mixed state and the only constraint is that: $\sum_i p_i = 1$

Now we can define the density operator.

Definition 2.3.1 (Density Operator)**.** *The density operator is defined as:*

$$\rho = \sum_i p_i \, |\psi_i\rangle \langle\psi_i| \quad .$$

In the particular case in which the ensemble represents a pure state, we call it $|\psi\rangle$, then the density operator could be written as:

$$\rho = |\psi\rangle \langle\psi| \quad . \tag{2.1}$$

Going deeper we have to define some proprieties of the density operator:

- The density operator is an Hermitian operator:
$$\rho = \rho^\dagger$$
- The density operator is semi-positive:
$$\rho \geq 0$$
- The density operator is idempotent if it represents a pure state:
$$\rho = \rho^2$$
- The trace of the density operator is sums to 1:
$$tr(\rho) = 1$$

2.3.2 Evolution quantum postulates, density operator

After having defined the density operator we can reformulate the previous postulate in order to adapt them to the density operator formalism.
In order to rewrite the first postulate we consider the definition of the density operator and its proprieties.

Postulate 2.3.1. *Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its density operator, which is semi-positive operator ρ with trace one, acting on the state space of the system. If a quantum system is in the state ρ_i with probability p_i, then the density operator for the system is $\sum_i p_i \rho_i$.*

Next, let us consider a system which has initial state $|\psi_i\rangle$ with probability p_i, then following the Post.2.2.2, the state after the evolution is $U|\psi_i\rangle$ with probability p_i. Therefore the evolution of the density operator is described by the following equation:

$$\rho = \sum_i |\psi_i\rangle \langle\psi_i| \, p_i \xrightarrow{U} \sum_i p_i U |\psi_i\rangle \langle\psi_i| \, U^\dagger = U \rho U^\dagger$$

Postulate 2.3.2 (Postulate 2)**.** *The evolution of a closed quantum system is described by a unitary transformation. That is, the state ρ of the system at time t_1 is related to the state ρ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,*

$$\rho' = U\rho U^\dagger \quad .$$

Measurements are also easily described in the density operator language. Suppose we perform a measurement described by measurement operators M_m as indicated by the postulate Post.2.2.4. If the initial state was $|\psi_i\rangle$, then the probability of getting result m is:

$$p(m|i) = \langle\psi_i| M_m^\dagger M_m |\psi_i\rangle = tr(M_m^\dagger M_m |\psi_i\rangle \langle\psi_i|) \quad .$$

In order to compute the probability if the outcome m, by the law of total probability, we have to sum over all the possible i.

$$\begin{aligned} p(m) &= \sum_i p(m|i)p_i \\ &= \sum_i tr(M_m^\dagger M_m |\psi_i\rangle \langle\psi_i|)p_i \\ &= tr(M_m^\dagger M_m \rho) \quad . \end{aligned}$$

Moreover we suppose to obtain the outcome m, and as before if the initial state was $|\psi_i\rangle$ the state after the measurement, according to postulate Post 2.2.4 will be:

$$|\psi_i^m\rangle = \frac{M_m |\psi_i\rangle}{\sqrt{\langle\psi_i| M_m^\dagger M_m |\psi_i\rangle}}$$

Thus, after a measurement which yields the result m we have an ensemble of states $|\psi_i^m\rangle$ with respective probabilities $p(i|m)$. The corresponding density operator ρ_m is therefore:

$$\rho_m = \sum_i p(i|m) |\psi_i^m\rangle \langle^m_i| = \sum_i p(i|m) \frac{M_m |\psi_i\rangle \langle\psi_i| M_m^\dagger}{\langle\psi_i| M_m^\dagger M_m |\psi_i\rangle} \quad .$$

By elementary probability we obtain that:

$$\begin{aligned}\rho_m &= \sum_i p_i \frac{M_m |\psi_i\rangle \langle\psi_i| M_m^\dagger}{tr(M_m^\dagger M_m \rho)} \\ &= \frac{M_m \rho_m M_m^\dagger}{tr(M_m^\dagger M_m \rho)} \quad .\end{aligned}$$

Putting together this last equation with the postulate Post. 2.2.4 we can define the measurement postulate for the density operator.

Postulate 2.3.3 (Postulate 3)**.** *Quantum measurements are described by a collection* $\{M_m\}$ *of measurement operators. These are operators acting on the state space of the system being measured. The index* m *refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is* ρ *immediately before the measurement then the probability that result* m *occurs is given by:*

$$p(m) = tr(M_m^\dagger M_m \rho) \quad ,$$

and the state of the system after the measurement is

$$\frac{M_m \rho_m M_m^\dagger}{tr(M_m^\dagger M_m \rho)} \quad .$$

The measurement operators satisfy the completeness equation,

$$\sum_m M_m^\dagger M_m = I$$

Last but not least we write the equivalent Postulate 2.2.5 for the density operator.

Postulate 2.3.4 (Postulate 4)**.** *The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through* n*, and system number* i *is prepared in the state* ρ_i*, then the joint state of the total system is* $\rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$*.*

2.3.3 Bloch Sphere

In this section we want to introduce a "visualization tool" that will be useful later. Let us consider the basis $\{I, \sigma_1, \sigma_2, \sigma_3\}$ where I is the Identity matrix and the others are the Pauli matrices:

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad ,$$

$$\sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad ,$$

$$\sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad .$$

After this preamble we are able to define the most general density matrix for a qubit by a linear combination of the previous basis. In particular we recall that a propriety of the density matrix is the trace condition:

$$tr(\rho) = 1 \quad .$$

In order to respect this condition, since the Pauli matrices have all trace equal to 0:

$$tr(\sigma_i) = 0 \quad \forall i \in \{1, 2, 3\} \quad ,$$

it follows that the I has to have a coefficient equal to $\frac{1}{2}$. In this way remains only three degrees of freedom to complete the representation of the density matrix ρ, therefore by choosing a real three dimensional vector $\overrightarrow{\mathrm{r}} = r_1, r_2, r_3$, we can rewrite the density matrix in its most general form:

$$\rho = \frac{1}{2}(I + \overrightarrow{\mathrm{r}}\overrightarrow{\sigma}) = \frac{1}{2}\begin{pmatrix} 1 + r_3 & r_1 - ir_2 \\ r_1 + ir_2 & 1 - r_3 \end{pmatrix} \quad .$$

The only constraint over the vector $\overrightarrow{r}$ is that $||\overrightarrow{r}|| \leq 1$. This vector is known as *Bloch vector* for the state ρ.
We can notice that by the use of the vector $\overrightarrow{r}$ it is possible to indicate the point within the sphere that correspond to a given ensemble of states. We recall that

for the pure state ρ is idempotent, and from this follows that for pure states:

$$tr(\rho^2) = 1 \quad .$$

If we rewrite this condition for the general representation of the density matrix it follows that:

$$tr(\rho^2) = \frac{1}{2}(1 + ||r||^2) = 1 \leftrightarrow ||\overrightarrow{r}|| = 1$$

In a sphere this condition is satisfied for all the points of the surface, therefore to every point in the surface corresponds a pure state.

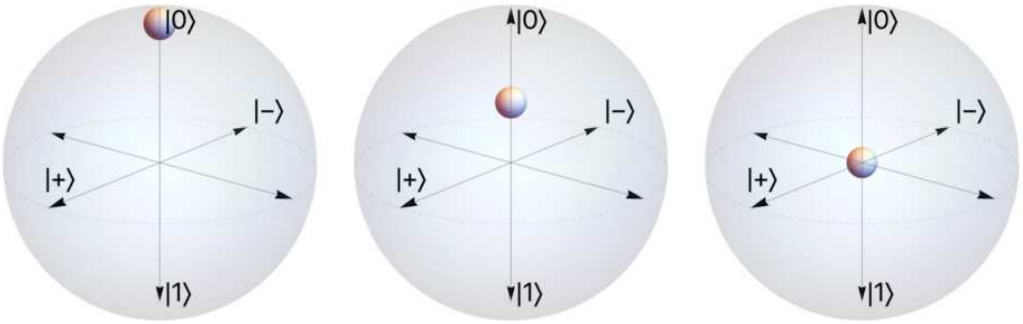


Figure 2.1: Examples of pure and mixed state in a Bloch Sphere. Left: a pure state. Center: an arbitrary mixed state. Right: Complitely mixed state

2.4 Quantum operation and noises

2.4.1 Introduction

The term *Quantum operations* refers to the mathematical operations that are performed on quantum systems in order to manipulate their state and extract information. These operations are fundamental to the field of quantum computing, which seeks to harness the unique properties of quantum mechanics to perform calculations that would be infeasible on classical computers. Quantum operations are also important in quantum communication and quantum cryptography, where they are used to encode and decode information in a secure way. Understanding and controlling quantum operations is essential for the development of practical quantum technologies, and has been the focus of intense research in recent years.

2.4.2 Quantum operations and Superoperators

Let us consider a bipartite system: system plus the environment. The system evolves unitarily, according to postulate 2.3.2. We want to describe the evolution just of the system, without the environment. Without loss of generality we can assume that the environment is in a pure state let us say state $|0\rangle$. We describe the density matrix of the whole system as:

$$\rho_{s+e} = \rho_s \otimes |0\rangle_e \langle 0|_e \quad .$$

The temporal evolution of the total system is related to the unitary operator U, which drives the system to:

$$\rho'_{s+e} = U\rho_{s+e}U^\dagger = U(\rho_s \otimes |0\rangle_e \langle 0|_e)U^\dagger \quad .$$

Since we are interested in the density matrix of just the single system without considering the environment, we have to trace out the environment component:

$$\begin{aligned}\rho'_s &= tr_e(\rho'_{s+e}) = tr_{s+e}[U(\rho_s \otimes |0\rangle_e \langle 0|_e)U^\dagger] \\ &= \sum_k \langle k| U |0\rangle_e \rho_s \langle 0| U^\dagger |k\rangle_e\end{aligned}$$

where $\{|k\rangle\}$ is a basis for the Hilbert space $\mathcal{H}_e$ associated with the environment subsystem and $\langle k| U |0\rangle_e$ is an operator acting on the Hilbert space $\mathcal{H}_s$ associated with the subsystem of interest. If we define *Kraus operators*:

$$E_k = \langle k| U |0\rangle_e \quad ,$$

we can also write that the evolution of our system of interest is:

$$\rho' = \sum_k E_k \rho_s E_k^\dagger \quad . \tag{2.2}$$

Moreover, since U is unitary follows that the kraus operators has to respect the following constraint:

$$\sum_k E_k E_k^\dagger = I \quad . \tag{2.3}$$

Since the equation 2.2 defines a linear map between linear operators and linear operators:

$$\mathcal{E} : \rho_s \rightarrow \rho_s' = \sum_k E_k \rho_s E_k^\dagger \quad ,$$

and if the completeness relation (Eq. 2.3) is satisfied then the map $\mathcal{E}$ is known as *quantum operation* or as *superoperator*. and the equation 2.2 is known as the *Kraus representation* of the superoperator $\mathcal{E}$.
The proprieties of a superoperator are the followings:

- Linearity:
$$\mathcal{E}(\alpha\rho_1 + \beta\rho_2) = \alpha\mathcal{E}(\rho_1) + \beta\mathcal{E}(\rho_2)$$
- Preserve Hermitianity:
$$\mathcal{E}(\rho)^\dagger = \mathcal{E}(\rho^\dagger) = \mathcal{E}(\rho)$$
- Trace Preserving:
$$tr(\mathcal{E}(\rho)) = tr(\rho) = 1$$
- Completely positive:
$$I \otimes \mathcal{E}(\rho_{A+e}) \geq 0 \quad .$$

Two superoperators $\mathcal{E}_A$ and $\mathcal{E}_B$ can be composed to give a new superoperator:

$$\mathcal{E} = \mathcal{E}_B \mathcal{E}_A \quad ,$$

defined by $\mathcal{E}(\rho_1) = \mathcal{E}_B(\mathcal{E}_A(\rho_1)) \quad .$

2.4.3 Operation and Noises

In this section we will present some useful noise and operations. We begin by defining the *depolarizing channel*.
The depolarizing channel is a type of quantum noise. In order to define its effect we consider a single qubit and we suppose that with probability p the qubit is *depolarized*, which means that is substituted with the completely mixed state. Moreover with probability $1 - p$ the qubit is left untouched. The state of the system after this noise is:

$$\rho' = \mathcal{E}(\rho) = p\frac{I}{2} + (1-p)\rho \quad .$$

The effect of the depolarization noise is illustrated in the following figure.

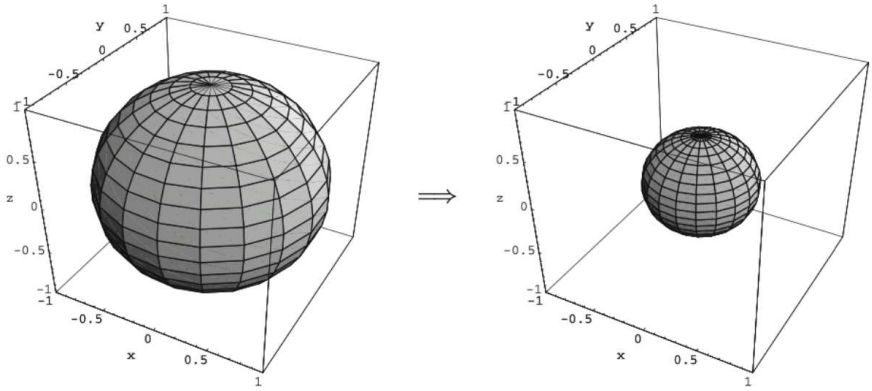


Figure 2.2: Effect of the depolarizing channel on the Bloch Sphere, for p = 0.5

Another fundamental type of noise is called *Amplitude damping channel*.
The amplitude damping channel is a quantum noise model that describes the loss of energy from a quantum system to its environment. In this model, the amplitude (or strength) of the quantum state is reduced over time due to the interaction with the environment. This can be caused by processes such as photon emission, which causes the quantum state to decay towards the zero state (or ground state). The amplitude damping channel can be described by a Kraus operator that maps the initial quantum state to a reduced state with lower amplitude. It is described by the following quantum operation:

$$\mathcal{E}_{AD}(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger \quad ,$$

where $E_k = \langle k| B |0\rangle$ are:

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix} \quad ,$$
$$E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix} \quad .$$

A very important type of quantum operation is the *bit flip* channel.
The bit flip quantum operation is a type of quantum gate that acts on a single qubit (quantum bit). It corresponds to a logical NOT operation in classical computing, which flips the value of a classical bit from 0 to 1, or vice versa. In the case of a qubit, the bit flip operation flips the quantum state from the $|0\rangle$ state to the $|1\rangle$ state, or vice versa, while leaving the relative amplitudes of the two states unchanged. The kraus representation of this operation is the following:

$$E_0 = \sqrt{p}I = \sqrt{p}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad , \quad E_1 = \sqrt{1-p}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad .$$

The effect of the bit flip is illustrated in the following figure.

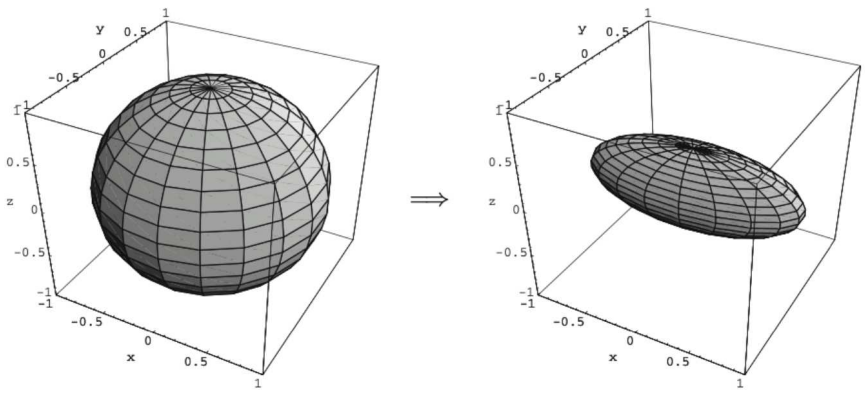


Figure 2.3: Effect of the bit flip channel on the Bloch Sphere, for p = 0.3

3

Reinforcement Learning

"Reinforcement learning is like a child learning to walk: it falls down, gets back up and tries again, gradually learning to balance"

Andrew Ng

3.1 Introduction

Reinforcement Learning (RL) is a type of machine learning that focuses on training agents to make a sequence of decisions in an environment. The agent learns to behave in an environment by performing certain actions and receiving rewards or penalties. The ultimate goal of the agent is to maximize the cumulative reward over time. RL algorithms are used to learn from the interactions between the agent and the environment, in order to improve the agent's decision-making abilities.

There are several types of RL algorithms, each with its own unique approach to solving RL problems. Some popular algorithms include Q-learning, SARSA, policy gradient and actor-critic methods.

In particular in this chapter we will focus on the policy gradient method. In

this approach, the agent learns to improve its decision-making by directly adjusting its policy, which is a function that maps states to actions. This approach is particularly useful when the action space is continuous or when the optimal policy is not easy to represent.

The aim of this chapter is to provide the reader the foundations to understand the environment definition. Moreover, we want to give a basic knowledge of the main RL algorithms in order to comprehend the PPO algorithm used in the simulations.

The main reference for this chapter is the Sutton and Burto's book [20].

3.2 RL FRAMEWORK

The main components of the general reinforcement learning (RL) framework are:

- Agent: The agent is the entity that interacts with the environment and takes actions.
- Environment: The environment is the collection of entities the agent interacts with. It can be either a physical or a virtual.
- Reward: The reward is a scalar value that represents the feedback the agent receives from the environment for its actions. The agent's goal is to maximize the cumulative reward over time.

When the agent interacts with the environment some other key concepts come into play:

- State: The state represents the current situation of the environment. The agent uses the state to decide on its next action.
- Action: The action is the decision made by the agent based on the current state of the environment.
- Policy: The policy is the strategy used by the agent to decide on its actions. It maps states to actions.

It is possible to formalize in a mathematical way these key concepts. In particular we define $\mathcal{S}$ the set of all possible states of an environment. $\mathcal{S}$ can be either a finite discrete or continuous set. The agent interacts with the environment by the chosen of actions. We call $\mathcal{A}$ the set of all possible actions

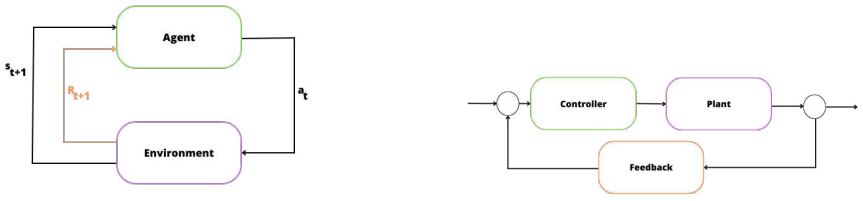
that an agent can perform. The agent have to choose an action every time-step t where $t \in \mathbb{N}$. At every time-step t an agent receives from the environment a state $s_t \in \mathcal{S}$. We define a *policy* as a conditional distribution:

$$\pi(a|s) = \mathbb{P}(A_t = a|S_t = s) \quad ,$$

which assign a probability $\pi(a|s)$ to the possible actions $a \in \mathcal{A}$. The agent at time-step t samples from $\pi(a|s)$ an action a_t. Depending on the selected action, the agent receives from the environment the reward $R_{t+1}(s_t, a_t)$ which is a measure of the goodness of the action performed. The environment returns also the next state s_{t+1}:

$$s_{t+1} = \mathbb{E}[s_t, a_t] \ ,$$

it can be either random or deterministic.
It is possible to compare this framework with the classical control theory framework. In particular we highlight the similarities between the agent and the controller, the action a_t chosen at time t by the agent can be seen as the control action u_t performed by the controller at time t, the environment can be interpreted as the plant to control and last but not least the reward R can be seen as a feedback action. In the following figures we tried to underline both the reinforcement learning framework and its similarity with control theory one 3.1.



(a) RL Scheme (b) Control Scheme

Figure 3.1: Analogy between RL and Control Scheme

3.3 Mathematical formalism for the environment definition

In this section we will dive deep into the mathematical formalism that let us define a reinforcement learning environment. First we will start by describing the path that from a Markov Chain leads to the Markov Decision Processes (MDP) which are the most used framework to describe the fully observable case. Then we will deal with the Partially Observable MDP which works with the partially observable case.

3.3.1 Markov Chain

A Markov Chain is a mathematical model for a sequence of events in which the probability of each event depends only on the state attained in the previous event. In this framework we define a stochastic process over the collection of random variables S_t with $t \in \mathbb{N}$. This collection of random variables $\{S_t\}$ are defined over the common alphabet $\mathcal{S}$ with associated probability distribution $\mathbb{P}$.

Definition 3.3.1 (Markov Propriety)**.** *A stochastic process defined over the random variables $S_t \in \mathcal{S}$ with associate probability distribution $\mathbb{P}$ is Markov if:*

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t] \quad .$$

In a Markov Chain the probability of transitioning from one state to another is determined by the probability of the current state and the transition probability between the current state and the next state. In mathematical terms, a Markov Chain is defined as a pair $< \mathcal{S}, \mathcal{P} >$ where $\mathcal{S}$ is a finite set of states and $\mathcal{P}$ is the transition probability matrix, where each element $p_{ij} \in \mathcal{P}$ represents the probability of transitioning from state i to state j. In conclusion we can define a Markov Chain as:

Definition 3.3.2 (Markov Chain)**.** *A Markov Chain (or Markov Process) is the Markov Stochastic Process specified by the pair $< \mathcal{S}, \mathcal{P} >$ such that:*

- *$\mathcal{S}$ is an alphabet*
- *$\mathcal{P}$ is a state transition probability matrix with entries:*

$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s] \quad .$$

3.3.2 MRP, State-Value function and Bellman Equation

A Markov Reward Process (MRP) is a variation of a Markov Chain that incorporates the notion of rewards. In an MRP, each state has an associated reward, and the goal is to determine the long-term expected reward of being in a particular state.

Definition 3.3.3 (Markov Reward Processes)**.** *A Markov Reward Process is a tuple* $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ *such that:*

- $\mathcal{S}$ *is a finite set of Markov States*
- $\mathcal{P}$ *is a state transition probability matrix with entries:*
$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s] \quad .$$
- $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$ *is a reward function that assigns a scalar to each state* s_i*:*
  - *We define the random variable* $R_{t+1} = \mathcal{R}(s_t)$ *and the expected reward as:*
$$\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$$
- $\gamma \in [0, 1]$ *is a discount factor.*

Through the use of the discount factor, we are able to represent the relative importance of future rewards w.r.t. current rewards. In particular when we have a discount factor $\gamma = 0$ we call that case *"myopic"*, on the other hand if $\gamma = 1$ the case is called *"far-sighted"*. In order to find a solution to the MRP we have to introduce the concepts of *state-value function* and the *BellMan Equation*.
The definition of both the state-value function and the BellMann equation has as foundation the concept of *Return*.

Definition 3.3.4 (Return)**.** *The Return* G_t *is the random variable representing the total future discounted reward from time-step* t*:*

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad , \tag{3.1}$$

where γ *is a discount factor,* $\gamma \in [0, 1]$.

The value function for a state is the expected sum of discounted rewards starting from that state. More formally, the state-value function $V(s)$ for an MRP is defined as:

Definition 3.3.5 (State-Value function). *The state value function $V(s)$ of a Markov Reward Process is the expected return starting from state s:*

$$V(s) = \mathbb{E}[G_t | s_t = s] \quad .$$

It is possible to highlight that the state-value function can be decomposed into two main parts:

- The immediate reward R_{t+1},
- The discounted value of successor state $\gamma V(S_{t+1})$.

Exploiting this concept we get that:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \quad . \end{aligned}$$

From the previous result and from the *law of iterated expectations* we can derive the Bellmann Equation:

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | s_t = s] = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | s_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})] \, . \end{aligned} \tag{3.2}$$

3.3.3 MDP, V-function and Q-function

A Markov Decision Process (MDP) is a variation of a Markov Reward Process that incorporates the notion of actions. To use this framework we suppose to have the full knowledge of the system, in other words everything is observable.

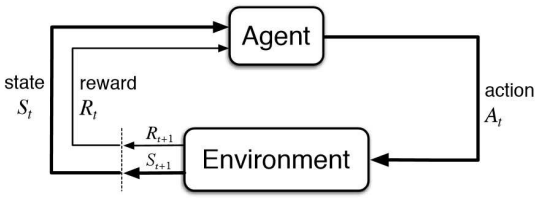


Figure 3.2: Reinforcement Learning Framework in an observable space [16]

Definition 3.3.6 (Markov Decision Process). *A Markov Decision Process is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ such that:*

- $\mathcal{S}$ *is a alphabet of a Markov Chain,*
- $\mathcal{A}$ *is a finite set of actions,*
- $\mathcal{P}^a$ *is a state transition probability matrix with entries:*
$$\mathcal{P}^a_{SS'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \quad , \tag{3.3}$$
- $\mathcal{R}^a : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a reward function that assigns a scalar to each state pair* (s_i, a_j)*:*
  - *We define the random variable* $R_{t+1} = \mathcal{R}^a(s, a)$ *and the expected reward as:*
$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \quad , \tag{3.4}$$
- γ *a discount factor,* $\gamma \in [0, 1]$.

As we can see the concept of MDP is strictly related to the concept of action. Indeed at every time t the agent chooses an action that will drive it to a new state. In particular the action performed by the agent is chosen accordingly to a certain policy $\pi(\cdot|\cdot)$.

Definition 3.3.7 (Policy)**.** *A policy is a distribution over actions given that we are in a state:*
$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \quad .$$

A policy fully defines the behaviour of a RL agent. MDP policies depend on the current state. Given a policy and a MDP we can define the dynamic of the state transition and the rewards as follows:

- The state sequence $S_1, S_2, \ldots$ is a Markov Process $< \mathcal{S}, \mathcal{P}^\pi >$ where:
$$\mathcal{P}^\pi_{\mathcal{S},\mathcal{S}'} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{\mathcal{S},\mathcal{S}'} \quad .$$
- The state-reward sequence $S_1, R_1, S_2, R_2, \ldots$ is, as in Def. 3.3.3, a Markov Reward Process $< S, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma >$ where:
$$\mathcal{R}^\pi_{\mathcal{S}} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_{\mathcal{S}} \quad .$$

Due to the fact that, as we have highlighted, in a MDP both the transition between states and the reward depend on the policy, we have to provide a different definition of state-value function.

Definition 3.3.8 (State-Value Function for MDP)**.** *The state-value function $V_\pi(s)$ of a MDP is the expected return from state s if we follow policy policy π:*

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \quad .$$

Moreover using the whole state-action $< s, a >$ pare we can define another useful function called the *Q-Function* or *action-value function*.

Definition 3.3.9 (Action-Value Function)**.** *The action-value function $Q_\pi(s, a)$ of an MDP is the expected return from state s if we take action a and then we follow policy π:*

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \quad .$$

As for the MRP is possible to rewrite both the *V-function* and the *Q-function* in an recursive way. Indeed the state-value function can be decomposed into two main parts:

- The immediate reward R_{t+1},
- The discounted value of successor state $\gamma V_\pi(S_{t+1})$.

This leads to a new expression for the V-function that is:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \\ &= \sum_{a\in\mathcal{A}} \pi(a|s)(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a V_\pi(s')) \quad . \end{aligned}$$

Similarly we can apply the same reasoning for the Q-function and we obtain:

$$\begin{aligned} Q_\pi(s) &= \mathbb{E}[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \\ &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a'\in\mathcal{A}} \pi(a'|s')Q_\pi(s', a') \quad . \end{aligned}$$

To solve a MDP we have to find the *optimal policy*, which is a mapping from states to actions that maximizes the long-term expected reward.
The optimal policy is the one that let us know the *optimal action-value function* Q_* which is the the maximum action-value function over all policies :

$$Q_*(s, a) = max_\pi Q_\pi(s, a) \quad ,$$

from the optimal action-value function it is possible to derive the *optimal state-value function*:

$$V_*(s) = max_a Q_*(s,a) \quad .$$

3.3.4 POMDP

A Partially Observable Markov Decision Process (POMDP) [7][4] is an extension of an MDP that takes into account the fact that the agent may not have full knowledge of the current state of the environment. In a POMDP, the agent receives an observation at each time step, rather than directly observing the state of the environment. The observation may not be enough to uniquely identify the true state, and the agent must use its previous observations and actions to infer the state.

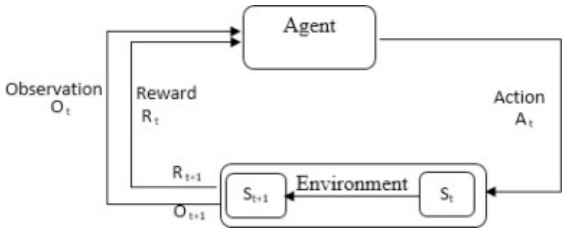


Figure 3.3: RL Framework in a partially observable space[3]

Definition 3.3.10 (Partially Observable Markov Decision Process)**.** *A Partially Observable Markov Decision Process (POMDP) is defined as a tuple* $< \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma, b_0 >$ *such that:*

- $\mathcal{S}$ *is a Markov State Space,*
- $\mathcal{A}$ *is a finite set of actions,*
- Ω *is a set of possible observations,*
- $\mathcal{P}$ *is a state transition probability matrix with entries:*

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \quad , \tag{3.5}$$

- $\mathcal{O}$ *set of observation probabilities*

$$\mathcal{O}^a_{os'} = \mathbb{P}[O_t = o | S_{t+1} = s', A_t = a] \tag{3.6}$$

  - *The probability of observing* $o \in \Omega$ *given that we take action* $a \in \mathcal{A}$ *and end up in* $s' \in \mathcal{S}$*,*

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is a reward function:*

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \quad , \tag{3.7}$$

- γ *a discount factor,* $\gamma \in [0, 1]$*,*
- b_0 *is a probability distribution over all possible states.*

In POMDPs, the agent can not tell for sure where it is in the state space, all it can have are *beliefs* on that probability distribution over states, this is usually called belief state b. It represents an agent's knowledge or uncertainty about the state of the system given the observations it has made up to that point. Formally it can be defined as:

Definition 3.3.11 (Belief State)**.** *the belief state at time* t*, denoted as* b_t*, is a probability distribution over the set of all possible states,* $\mathcal{S}$*:*

$$b_t(s) = \mathbb{P}[S_t = s|o_1 : t] \quad , \tag{3.8}$$

where s_t *is the true state of the system at time* t*,* $o_1 : t$ *is the sequence of observations made up to time* t*, and* $P(s_t = s|o_1 : t)$ *is the probability that the system is in state* s *at time* t *given the observations made up to that point.*

The belief state is updated at each time step based on the current observation and the transition dynamics of the system and it is computed via Bayes rule:

$$\begin{aligned} b(s_i) &= P(s_i|a, o, b) \\ &= \frac{P(o|s_i, a, b)P(s_i|a, b)}{P(o|a, b)} \\ &= \frac{P(o|s_i, a, b)\sum_j P(s_i|a, s_j)P(s|a, b)}{P(o|a, b)} \\ &= \frac{P(o|s_i, a)\sum_j P(s_i|a, s_j, b)P(s|a, b)}{P(o|a, b)} \\ &= \frac{\mathcal{O}(o|s_i, a)\sum_j P(s_i|a, s_j)b(s_j)}{P(o|a, b)} \quad , \end{aligned}$$

where as $P(o|a, b) = \sum_k \mathcal{O}(o|a, s_k)\sum_j \mathcal{P}(s_k|a, s_j)$.

The expected reward for the agent of taking action a in belief state b is:

$$r(b, a) = \sum_{s \in \mathcal{S}} R(s, a) b(s) \quad .$$

3.3.5 QOMDPs

A quantum observable Markov decision process (QOMDP) [4] generalizes a POMDP by using quantum states rather than belief states. In a QOMDP, an agent can apply a set of possible operations to a d-dimensional quantum system. The operations each have $\mathcal{K}$ possible outcomes. At each time step, the agent receives an observation corresponding to the outcome of the previous operation and can choose another operation to apply. The reward the agent receives is the expected value of some operator in the system's current quantum state.
A QOMDP is defined as a set $< \mathcal{S}, \Omega, \mathcal{A}, \mathcal{R}, \gamma, \rho_0 >$ such that:

- $\mathcal{S}$ is a Hilbert space. We allow pure and mixed quantum states so we will represent states in S as density matrices.
- $\Omega = \{o_1, \ldots, o_{|\Omega|}\}$ is a set of possible observations.
- $\mathcal{A} = \{A^1, \ldots, A^{|\mathcal{A}|}\}$ is a set of superoperators. Each superoperator $A^a = \{A^a_1, \cdots, A^a_{|\Omega|}\}$ has $|\Omega|$ Kraus matrices. Note that each superoperator returns the same set of possible observations; The return of o_i indicates the application of the ith-Kraus matrix so taking action a in state ρ returns observation o_i with probability:

  $$\mathbb{P}(o_i|\rho, a) = Tr(A_i^{a\dagger} A_i^a \rho)$$

  If o_i is observed after taking action a in state ρ, the next state is:

  $$\rho_{t+1|\rho,a,o_i} = \frac{A_i^a \rho A_i^{a\dagger}}{Tr(A_i^a \rho A_i^{a\dagger})}$$

- $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{A}|}\}$ is a set of operators. The reward associated with taking action a in state ρ is the expected value operator R_a on ρ,

  $$R(\rho, a) = Tr(\rho R_a)$$

- $\gamma \in [0, 1)$ is the discount factor.
- ρ_0 is the starting state as density operator.

Like an MDP or POMDP, a QOMDP represents a world in which an agent chooses actions at discrete time steps and receives observations. The world modeled by the QOMDP is a quantum system that begins in ρ_0, the starting state of the QOMDP. At each time step, the agent chooses a superoperator from the set $\mathcal{A}$, whereupon the corresponding operation is done on the system and the agent receives an observation from the set in accordance with the laws of quantum mechanics. The agent also receives a reward according to the state of the system after the operation and $\mathcal{R}$.

3.4 Algorithms

In this section, we will focus on the various RL algorithms that have been developed. We will begin by discussing the main categories of RL algorithms, firstly we will delve deeper into the concept of Model Free and Model Based RL. Then we will focus the reader attention on Model Free RL algorithms highlighting the two main categories:

- Value-Based,
- Policy-Based.

We will explore specific algorithms within each category, such as Q-learning, Reinforce (Policy Gradient), and actor-critic methods, while providing a detailed analysis of their strengths and weaknesses. By the end of this section, the reader will have a comprehensive overview of the different RL algorithms and their applications.

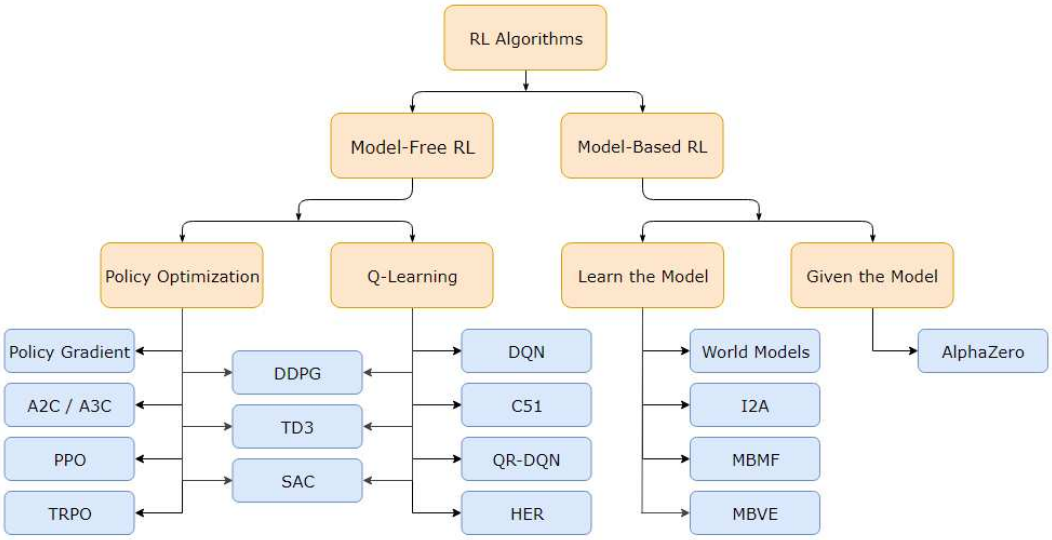


Figure 3.4: A non-exhaustive taxonomy of algorithms in modern RL. [1]

3.4.1 Model-Free vs Model-Based RL

A first partition in the Reinforcement Learning algorithms cluster consists in the definition of the following categories: Model-Free and Model based RL [1]. Model-free RL algorithms gather experience from the environment to learn, update and improve a value function, such that the V-function or the Q-function, or a policy $\pi(\cdot|\cdot)$. This process works just taking into account the rewards that follows from the interaction between the RL agent and the environment without any knowledge about the underlying environment model. Model-free RL algorithms are simpler to implement and can be more robust to changes in the environment, but they may require more data and experience to converge to a good solution. It is possible to split the algorithms in this category in two different type:

- Value-Based approach,
  - Main algorithms: Q-Learning, Sarsa, TD(λ).
- Policy-Based approach.
  - Main algorithms: Reinforce, TRPO, PPO.

Model-based RL algorithms involve learning a model of the environment, which the agent can use to plan and make decisions. In particular as model we mean anything that an agent can use to predict how the environment will react to the action performed by the agent. These algorithms typically use dynamic programming or tree search methods to determine the optimal action to take in each state. The agent updates its understanding of the environment as it interacts with it, and uses this information to plan future actions. Model-based RL algorithms can be very sample efficient, but can become impractical in large or complex environments.

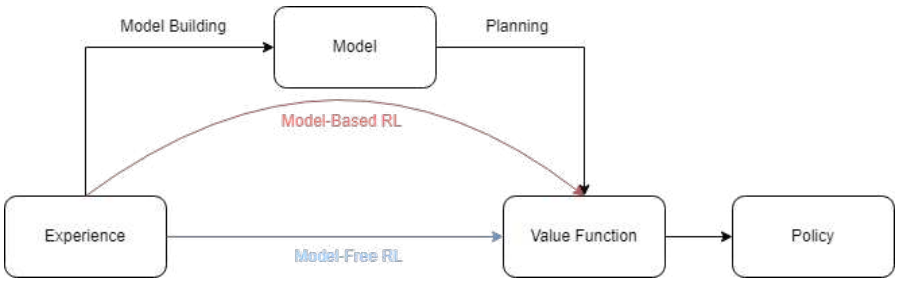


Figure 3.5: Block Scheme representing Model based and Model Free approaches

3.4.2 Exploration exploitation dilemma

Before going deeper in the algorithms we have to define a fundamental reinforcement learning problem called "*Exploration exploitation Dilemma*". Firstly we will provide some basic definitions:

- Exploration: is the process of trying out different actions to learn more about the environment.
- Exploitation: is the process of using the current knowledge of the environment to select actions that are likely to lead to the highest rewards.

The exploration exploitation dilemma arises when the agent has to decide whether to explore new actions to increase his knowledge about the environment or to exploit its current knowledge to receive a higher reward but losing the opportunity to learn something and even maybe get a better reward. Before delve deeper int this problem we recall that the action selected at time t is sampled from the current policy, which means:

$$a_t \sim \pi(a|s_t) \quad ,$$

therefore to deal with this problem a possible solution is to define particular policies. This type of approach is very useful in the case in which the environment has a *discrete action space* $\mathcal{A}$ and we consider the chosen algorithm belongs to the value based family. A possible policy to tackle the exploration exploitation dilemma is the ϵ-greedy policy. It consists into select with probability $1-\epsilon$ the action higher Q value and with probability ϵ a random action:

$$\pi(a|s_t) = \begin{cases} argmax_a Q(s_t, a) & \text{with probability } 1-\epsilon, \\ \text{a random action} & \text{with probability } \epsilon \end{cases} .$$

A slightly modified version of this policy is called *Decreasing ϵ-greedy* in which the value of the ϵ value decreases over time.
In the following figure we can see the mean reward in a typical RL problem. The reader can notice that using the greedy policy, which means choosing every time the action that returns the highest reward without gathering any knowledge of the environment, the return is lower than the other two cases with exploration.

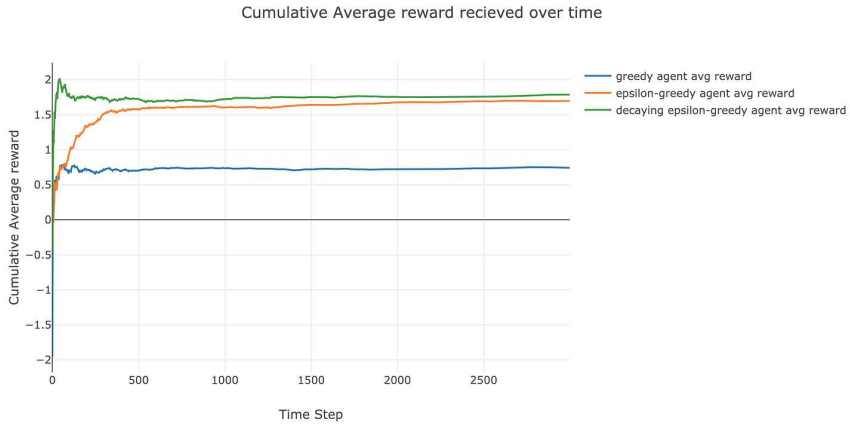


Figure 3.6: Typical behaviour of agent with and without exploration techniques

In the case of *continuous action space* $\mathcal{A}$ the agent does not choose directly any action but it returns two values in function of the current state s: $\mu(s)$ a mean value and $\sigma(s)$ a standard deviation. With these two values the agent builds a policy that is a gaussian distribution centered in μ with variance σ^2, then the action is sampled from this policy:

$$a_t \sim \pi(a|s) = \mathcal{N}(a|\mu(s), \sigma^2(s)) \quad .$$

With these kinds of policies is not necessary to introduce a rule to increase the exploration since the policy by itself is stochastic. Nevertheless in some cases is useful to adopt a strategy in order to avoid too quick variance reduction which can lead the agent to converge in a policy that is locally optimal, but not necessary globally optimal. With this focus a typical solution is to introduce a *"entropy bonus"* term in the agent cost function [11].
We recall that the entropy is defined as:

$$H(X) = -\int_{-\infty}^{\infty} p(x)\log(p(x))dx \quad .$$

Entropy in the Reinforcement Learning framework relates directly to the unpredictability of the actions which an agent takes in a given policy. Therefore greater the entropy, the more random the actions an agent takes.

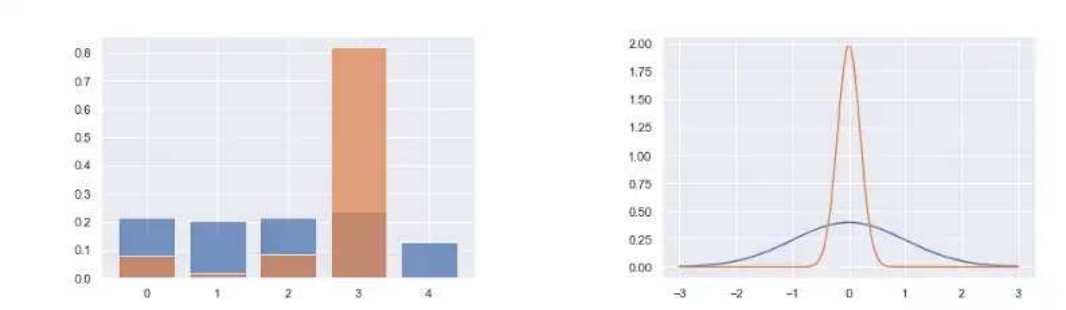


Figure 3.7: Categorical (left), and Gaussian (Right) distributions. Orange shows low-entropy distributions, while blue shows high-entropy distributions. [11]

3.4.3 Value-Based algorithms

The class of value-based algorithms is the class of reinforcement learning algorithms in which the agent tries to estimate the value of a given state or state-action pair in a given environment. These algorithms are able to do this using the concept of value functions, which, as seen before, these are a mathematical representation of the expected return that an agent will receive if it follows a particular policy $\pi(\cdot|\cdot)$. The value function is then used to guide the agent's decision-making process by indicating which actions are likely to lead to the highest rewards.
We recall that the two main value functions are the V-function and the Q-function which are defined by the Bellman equations as follows:

$$V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \quad ,$$
$$Q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \quad .$$

Every algorithm of the value based family performs at least one of this two key tasks:

- Policy Evaluation/Prediction: it consists in the derivation of the value function and the action-value function.
- Policy Improvement/Control: it consists in the research of the best policy.

Before delve deeper into the algorithms for the policy improvement we want to define an order relationship for the policies.

Definition 3.4.1 (Policy Order)**.** *Let us define two policy: π and π'. We say that the policy π is better than the policy π' and we write:*

$$\pi > \pi' \quad ,$$

if:

$$V_\pi(s) > V_{\pi'}(s) \quad \forall s \in \mathcal{S}$$

A first basic approach consists in considering the previous two tasks as separate, but more advanced approaches as the *Generalized Policy Iteration* (GPI) merge these two tasks combining prediction iterations to improvement iterations.

One of the most basic value-based methods is Monte Carlo (MC) method, which estimates the value of a state or state-action pair by averaging the rewards

obtained from following the current policy from that state or state-action. The update rule for the value function $V(\cdot)$ using MC method is given by the *average* of the returns or by the use of incremental updates (it can be proved that the two approaches are equivalent). The incremental updates approach is more efficient (in terms of memory) and it consists in the following equation:

$$V_\pi(s) = V_\pi(s) + \alpha[G - V_\pi(s)],$$

where G is the return (cumulative reward) obtained from following the current policy from the state s, and α is the stepsize. By the law of large numbers the sequence of averages of these estimates converges to the expected value of V_π. The first visit MC algorithm for the estimation of V_π is the following one.

Algorithm 1 First Visit MC prediction, for estimating V_π

Input a policy π to be evaluated
Initialize $V(s) \in \mathbb{R}$ arbitrary, for all $s \in \mathcal{S}$
Returns(s)$\leftarrow$ an empty list, for all $s \in \mathcal{S}$
for each episode **do**
  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  G = 0
  for each step of episode, t = T-1, T-2, ... **do**
    G = $\gamma G + R_{t+1}$
    while Unless S_t, appears in $S_0, S_1, \ldots, S_{t-1}$ **do**
      Append G to *Returns*(S_t)
      $V(s)$ = average($Returns(S_t)$)
    end while
  end for
end for

With a slightly modification of the previous algorithm is possible to estimate the Q-function w.r.t. π that is Q_π. The Monte Carlo estimation algorithm can be also used for the control goal. In particular it uses the GPI approach. The value function is iteratively altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved w.r.t. the current value function. In order to express these concepts in the MC algorithm, the key step is perform alternating complete steps of policy evaluation and policy improvement. The policy evaluation is done accordingly to the adaptation of

the previous algorithm w.r.t. the Q-function. The policy improvement instead is done by making the policy greedy w.r.t. the current action-value function that is:

$$\pi(s) \doteq argamax_a Q(s,a) \quad . \tag{3.9}$$

In this case we are defining a deterministic policy. Since we are using an *action-value function* we do not need any model to build the greedy policy. In order to justify the correctness of the algorithm we recall the *policy improvement* theorem:

Theorem 3.4.1 (Policy Improvement)**.** *Let π and π' be any pair of deterministic policies such that for all $s \in \mathcal{S}$,*

$$Q_\pi(s, \pi'(s)) \geq V_\pi(s) \quad .$$

Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in \mathcal{S}$:

$$V_{\pi'}(s) \geq V_\pi(s) \quad .$$

Moreover, if the first inequality is a strict inequality at any state, then the previous inequality must be a strict inequality too.

Applying this theorem to the idea of selecting a greedy policy we can prove that the algorithm with an infinite number of steps converges to an optimal policy. Let us consider the k-th and the $k+1$-th step in the algorithm and then according to the greedy policy we build it as $\pi_{k+1} = argamax_a Q_{\pi_k}$. Moreover we consider the quantity $Q_{\pi_k}(s, \pi_{k+1}(s))$ which represents the Q-value if the next action is taken by the policy π_{k+1} and all the other action are sampled from π_k. After this assumptions we can derive that:

$$\begin{aligned} Q_{\pi_k}(s, \pi_{k+1}(s)) &= Q_{\pi_k}(s, argamax_a Q_{\pi_k}(s,a)) \\ &= max_a Q_{\pi_k}(s,a) \\ &\geq Q_{\pi_k}(s, \pi_k(s)) \\ &\geq V_{\pi_k}(s) \quad . \end{aligned}$$

In this way we have proved that since the policy π_{k+1} is always better than π_k, or at least is it is equal to π_k, after an infinite number of episodes the policy will converge to an optimal one:

$$\pi_\infty = \pi^* \quad .$$

Algorithm 2 First-Visit Monte Carlo Reinforcement Learning

Initialize Q(s,a) for all s ∈ S and a ∈ A(s)
Initialize $\pi(s)$ for all s ∈ S
Initialize Returns(s,a) an empty list for all $s \in \mathcal{S}, a \in \mathcal{A}$
for each episode (forever) **do**
  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  for each pair s, a in the episode **do**
    G = G + R_{t+1}
    Append G to the *Returns(s,a)*
    Q(s, a) = $average(Returns(s, a)$
  end for
  for each s in the episode **do**
    $\pi(s) = argamax_a Q(s, a)$
  end for
end for

Another key value based algorithm is the *Temporal Difference learning* (TD-Learning). Since it is out of the focus of this thesis describe in deep this family of algorithms, in this section we will focus on the simplest case of the TD-Learning family that is the TD(0). The main difference with the Monte Carlo method is that the TD(0) do not wait until the next episode to update the value function, it just wait until the next time step. In order to define this concept in a mathematical way, first we recall that:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma V(S_t + 1)|S_t = s] \quad .$$

Now we recall that Monte Carlo methods uses an estimate of $\mathbb{E}_\pi[G_t|S_t = s]$ as target, whereas TD(0) uses exactly the last expression that is $\mathbb{E}_\pi[R_{t+1} + \gamma V(S_t + 1)|S_t = s]$ as the target, this is called *TD-target*. Since $V_\pi(S_t + 1)$ is not known, to iterate the algorithm we use the best estimate that we have of V_π which is $V_t(S_t+1)$ (the current estimate). One of the most famous TD-learning algorithms used to approximate the optimal action-value function Q^* is the Q-learning. The Q-learning algorithm starts with an arbitrary initial Q-function and repeatedly updates the Q-function using the current estimate, the observed rewards and next states. It converges to the optimal Q-function as the number of iterations increases. The Q-learning uses the concept of *Optimal BellMann Equation* to build

Algorithm 3 TD(0) algorithm to estimate V_π

Initialize V(s) for all $s \in \mathcal{S}$
Initialize $\pi(s)$ for all s ∈ S
for each episode **do**
  Initialize s_0
  repeat
    $a \sim \pi(a|s)$
    Take action a, observe R_{t+1} and next state s′
    $V(s) = V(s) + \alpha[R_{t+1} + \gamma V(s') - V(s)]$
    $s = s'$
  until s is final state
end for

the TD-target and therefore the value function update. In particular the Optimal BelMann equation is defined as follows:

Definition 3.4.2 (Bellmann Optimality Equation)**.** *The Bellmann optimality equation for Q^* is defined as:*

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma max_{a'}(Q(S_{t+1}, a'))|S_t = s, A_t = a] \quad .$$

The Q-function update is defined as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[R_{t+1} + \gamma max_a(Q(s_{t+1}, a')) - Q(s_t, a_t)] \quad .$$

Algorithm 4 Q-learning

1: Initialize $Q(s,a)$ arbitrarily for all s,a
2: **repeat**
3: Initialize s
4: **repeat**
5: Choose a from s using policy derived from Q (e.g., ϵ-greedy)
6: Take action a, observe r, s'
7: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'}(Q(s',a')) - Q(s,a)]$
8: $s \leftarrow s'$
9: **until** s is terminal
10: **until** convergence

The algorithm we have seen since now can be implemented by constructing a Q-table which is able to store all the values of every state-action pair (s,a).

Nevertheless there exist problems whose state-space and/or action-space are too large or continuous and therefore them can not be represented in a tabular form. To deal with these problems one possible solution is to use the *universal function approximator*: the neural networks. One popular algorithm for value function approximation is the Deep Q-Network (DQN) algorithm, which combines Q-learning with deep neural networks. The DQN algorithm uses a neural network to approximate the action-value function, denoted as $Q(s, a; \theta)$, where θ are the parameters of the network. The algorithm updates the network's parameters by minimizing the temporal-difference (TD) error between the predicted and target Q-values.

Algorithm 5 Deep Q-Network (DQN)

Initialize replay memory D to capacity N
Initialize Q-network with random weights θ
for $episode = 1$ to M **do**
    Initialize state s_1
    for $t = 1$ to T **do**
        With probability ϵ select a random action a_t, otherwise select $a_t = argmax_a Q(s_t, a; \theta)$
        Execute action a_t in emulator and observe r_t, s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        Sample a random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
        Set $y_j = r_j$ for terminal s_{j+1}
        $y_j = r_j + \gamma \max_a Q(s_{j+1}, a; \theta_{target})$ for non-terminal s_{j+1}
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to θ
        Every C steps, set $\theta_{target} = \theta$
    end for
end for

3.4.4 Policy-Based Algorithms

The family of the Policy-Based algorithms is a family of algorithms that finds its foundation in the idea of directly optimize the policy instead of building a value function [6] [10]. This approach has several advantages:

- Better convergence properties,
- Effective in high-dimensional or continuous action spaces,
- Can learn stochastic policies.

The main idea on how to improve the policy is to use a set of parameters $\boldsymbol{\theta}$ with the purpose to parameterize the policy π:

$$\pi(a|s, \boldsymbol{\theta}) = \mathbb{P}[A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}] \quad ,$$

the next step is to optimize this vector of parameters $\boldsymbol{\theta}$ w.r.t. a cost function $J(\boldsymbol{\theta})$. In particular the chosen cost function is the following:

$$J(\boldsymbol{\theta}) = V_{\pi_{\boldsymbol{\theta}}}(s_0)$$

where V_{π_θ} is the value function for policy $\pi_{\boldsymbol{\theta}}$ and s_0 is the starting state.
In order to maximize the cost function $J(\boldsymbol{\theta})$ is used the gradient ascent w.r.t. $\boldsymbol{\theta}$:

$$\nabla J(\boldsymbol{\theta}) = \nabla V_{\pi_{\boldsymbol{\theta}}} \quad .$$

In order to compute this gradient we refer to the fundamental *Policy Gradient Theorem*:

Theorem 3.4.2 (Policy Gradient Theorem)**.** *The policy gradient theorem for the episodic case establishes that*

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q_{\pi_\theta}(s_t, a_t)] \tag{3.10}$$

We recall to the reader that:

- $\nabla_\theta J(\theta)$ is the gradient of the expected cumulative reward with respect to the policy parameters θ
- π_θ is the current policy parameterized by θ
- τ is a trajectory, which is a sequence of states, actions, and rewards

- T is the length of the trajectory
- $Q_{\pi_\theta}(s_t, a_t)$ is the expected cumulative reward starting from state s_t, taking action a_t, and following policy π_θ thereafter.
- $\mathbb{E}_{\tau\sim\pi_\theta}$ denotes the expectation over all possible trajectories generated by the policy π_θ.

From the last theorem (3.4.2) is possible to derive the following quantity:

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\tau\sim\pi_\theta}[G_t \frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})}] \quad .$$

Now we can define the rule for the update of the $\boldsymbol{\theta}$ vector:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} = \boldsymbol{\theta}_t + \alpha G_t \nabla_{\boldsymbol{\theta}} ln\pi(A_t|S_t, \boldsymbol{\theta}_t)$$

Now we can define the REINFORCE algorithm that since it uses the complete return G is a Monte Carlo algorithm.

Algorithm 6 REINFORCE : Monte Carlo Policy Gradient

Initialize policy parameter $\boldsymbol{\theta}$
for each episode **do**
    Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    for each step of episode t = 0, . . . , T-1 **do**
        $G \leftarrow$ return from step $t(G_t)$
        $\boldsymbol{\theta} = \boldsymbol{\theta}_t + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} ln\pi(A_t|S_t, \boldsymbol{\theta}_t)$
    end for
end for

3.4.5 Actor Critic

One problem of the REINFORCE algorithm (Algo.6) is that as all the Monte Carlo algorithms leads to a high variance. This is due to the fact that trajectories can lead to different returns due to stochasticity of the environment and stochasticity of the policy, consequently, the same starting state can lead to very different returns.
A possible solution to this problem relates in the use of the combination of policy-based and value-based algorithm. This approach is known with the name of *Actor-Critic* approach [12].
The main idea under this technique is to use:

- an *Actor*: it defines and improves the policy $\pi_{\boldsymbol{\theta}}(s, a)$ in order to take actions,
- a *Critic*: it defines, in function of a set of parameters $\mathbf{w}$, a criterion (for example a value function) with the purpose to measure how good is the action taken.

A good criterion in order to both stabilize the learning and provide a good critic is the *Advantage function*:

Definition 3.4.3 (Advantage function)**.** *The advantage function w.r.t. a policy $\pi_{\boldsymbol{\theta}}$ is a measure of the relative value of a particular action compared to the average value of all actions in a given state:*

$$A_{\pi_{\boldsymbol{\theta}}}(s, a) = Q_{\pi_{\boldsymbol{\theta}}}(s, a) - V_{\pi_{\boldsymbol{\theta}}}(s) \quad .$$

For a true value-function $V_{\pi}\boldsymbol{\theta}$ the TD-error $\delta_{\pi_{\boldsymbol{\theta}}}$:

$$\delta_{\pi_{\boldsymbol{\theta}}} = R + \gamma V_{\pi_{\boldsymbol{\theta}}}(s') - V_{\pi_{\boldsymbol{\theta}}}(s) \quad ,$$

is an unbiased estimate of the advantage function:

$$\begin{aligned} \mathbb{E}[\delta_{\pi_{\boldsymbol{\theta}}}|s, a] &= \mathbb{E}[R + \gamma V_{\pi_{\boldsymbol{\theta}}}(s')|s, a] - V_{\pi_{\boldsymbol{\theta}}}(s) \\ &= Q_{\pi_{\boldsymbol{\theta}}}(s, a) - V_{\pi_{\boldsymbol{\theta}}}(s) \\ &= A_{\pi_{\boldsymbol{\theta}}}(s, a) \quad , \end{aligned}$$

so it is possible to use the it to compute the policy gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[\nabla_{\theta} ln\pi(a|s, \boldsymbol{\theta}) A_{\pi_{\boldsymbol{\theta}}}] = \mathbb{E}_{\pi}[\nabla_{\theta} ln\pi(a|s, \boldsymbol{\theta}) \delta_{\pi_{\boldsymbol{\theta}}}] \quad .$$

In practice since we can not know exactly V_{π_θ}, to compute the TD-error we will use an approximated version of the value function, parameterized by the vector of parameters $\mathbf{w}$:

$$\delta_{\mathbf{w}} = R + \gamma \hat{V}(s', \mathbf{w}) - \hat{V}(s, \mathbf{w}) \quad .$$

The equation used to update the $\boldsymbol{\theta}$ and $\mathbf{w}$ vectors of parameters are the following:

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \delta_{\mathbf{w}} \nabla_{\boldsymbol{\theta}} ln \pi(a|s, \boldsymbol{\theta})$$
$$\mathbf{w} = \mathbf{w} + \alpha^{\mathbf{w}} \delta_{\mathbf{w}} \nabla_{\mathbf{w}} \hat{V}(s, \mathbf{w})$$

It is possible to distinguish two main cases: the first one is when the advantage function is positive, the second one is when the advantage function is negative, both are analyzed in the following table.

Case	Effect
$A(s,a) > 0$	The gradient is pushed in its direction
$A(s,a) < 0$	The gradient is pushed in its opposite direction

Table 3.1: Effect of the advantage function on the optimization of $\boldsymbol{\theta}$ and $\mathbf{v}$

Algorithm 7 One-Step Actor-Critic (A2C) algorithm

Input: a differentiable policy parametrization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterized $\hat{V}(s, \mathbf{w})$
Parameters: step size $\alpha^{\boldsymbol{\theta}}$, $\alpha^{\mathbf{w}}$
Initialize $\boldsymbol{\theta}$ and $\mathbf{w}$
for each episode (forever) **do**
    Initialize S_0
    $S = S_0$
    I = 1
    while S is not terminal **do**
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta = R + \gamma \hat{V}(s', \mathbf{w}) - \hat{V}(s, \mathbf{w})$
        $\mathbf{w} = \mathbf{w} + \alpha^{\mathbf{w}} \delta_{\mathbf{w}} \nabla_{\mathbf{w}} \hat{V}(s, \mathbf{w})$
        $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta_{\mathbf{v}} \nabla_{\boldsymbol{\theta}} ln \pi(a|s, \boldsymbol{\theta})$
        $I = \gamma I$
        $S = S'$
    end while
end for

3.5 Proximal Policy Optimization

3.5.1 Introduction

The Proximal Policy Optimization (PPO) is an algorithm of the policy-based family. The intuition behind PPO is that we want to improve the training stability of the policy by limiting the change of the policy at each training epoch, in other words the purpose is to avoid too large policy updates.
This because of two main reasons:

- smaller policy updates during training are more likely to converge to an optimal solution,
- a too big step in a policy update can lead to a bad policy from which could be very hard or even impossible to recover.

In order to avoid large update it is essential to define how much the policy is changed w.r.t. the former one. To do so a possible way is to use the ratio between the current and the older policy, then clip it in a range $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter.

3.5.2 Clipped Objective Function

In this subsection we will define the objective function used by the PPO to build a good policy. To do this in this section we will use a top down approach, starting by the definition of the objective function and then analyzing every part. The clipped objective function ([6][17][18][2][10]) is the following:

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

First of all we define the ratio function that is:

$$r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} \quad .$$

It represents the "divergence" between the current policy and the former one. In particular this is a measure of how much is probable to take action a_t at state s_t in the current policy w.r.t. the previous one. We highlight two main cases:

- If $r_t(\boldsymbol{\theta}) > 1$, means that the action a_t at state s_t is more likely in the current policy than the older policy.

- If $0 \leq r_t(\boldsymbol{\theta}) \leq 1$, means that the action a_t at state s_t is less likely for the current policy than the older one.

Now we take care about the unclipped part of the expectation:

$$L^{CPI}(\boldsymbol{\theta}) = \hat{\mathbb{E}}[\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}old}(a_t|s_t)}A_t] = \hat{\mathbb{E}}[r_t A_t] \quad .$$

Here we can notice that without a constraint, the maximization of L^{CPI} would lead to an excessively large policy update. Hence we now introduce the second part of the expectation whose aim is to penalize changes to the policy that move $r_t(\theta)$ away from 1. Indeed the last part of the clipped objective function is:

$$L^{CL} = clip(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t \quad .$$

This clipped part is a version where $r_t(\boldsymbol{\theta})$ is clipped between $[1\epsilon, 1+\epsilon]$. With the Clipped Objective function, we have two probability ratios, one non-clipped and one clipped in a certain range depending on the ϵ hyperparameter (by heuristics $\epsilon = 0.2$).
Taking the minimum of the clipped and non-clipped objective means we'll select either the clipped or the non-clipped objective based on the ratio and advantage function.

Case	$r_t(\boldsymbol{\theta}) > 0$	A_t	Return Value of min	Objective is clipped	Sign of the Objective	Gradient
1	$r_t(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$	+	$r_t(\boldsymbol{\theta})A_t$	no	+	yes
2	$r_t(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$	-	$r_t(\boldsymbol{\theta})A_t$	no	-	yes
3	$r_t(\boldsymbol{\theta}) < 1-\epsilon$	+	$r_t(\boldsymbol{\theta})A_t$	no	+	yes
4	$r_t(\boldsymbol{\theta}) < 1-\epsilon$	-	$(1-\epsilon)A_t$	yes	-	0
5	$r_t(\boldsymbol{\theta}) > 1+\epsilon$	+	$(1+\epsilon)A_t$	yes	+	0
6	$r_t(\boldsymbol{\theta}) > 1+\epsilon$	-	$r_t(\boldsymbol{\theta})A_t$	no	-	yes

Table 3.2: Behaviour of the clipped objective function

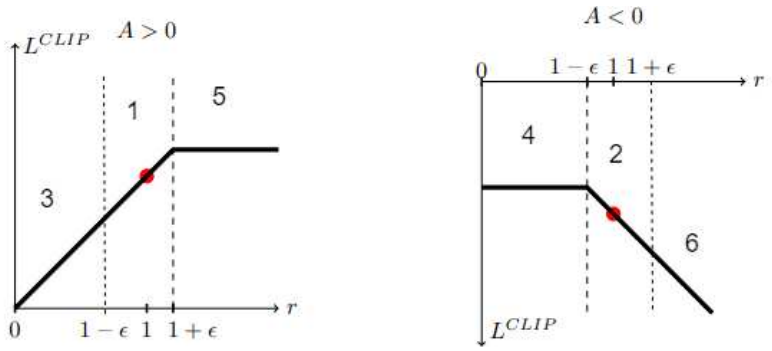


Figure 3.8: L^{CLIP} in function of the probability r_t

Looking to the previous table (Tab. 3.2) we can analyze the behaviour of the Clipped Objective function. It is possible to split the behaviour in six main cases divided into three clusters:

- In the first two cases the algorithm does not apply the clipping action since the ratio is between the range $[1-\epsilon, 1+\epsilon]$.

    1. Case 1: This case presents $A_t > 0$, this implies that the action is better than the average of all the actions in that state. Therefore the algorithm should encourage the agent to take that action in that state. Since $r_t(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$ the algorithm can increase its policy's probability of taking that action at that state.
    2. Case 2: This case presents $A_t < 0$, this implies that the action is worse than the average of all the actions in that state. Therefore the algorithm should discourage the agent to take that action in that state. Since $r_t(\boldsymbol{\theta}) \in [1-\epsilon, 1+\epsilon]$ the algorithm can decrease its policy's probability of taking that action at that state.

- In the second couple of cases (Case 3 and Case 4) we can notice that the ratio $r_t(\boldsymbol{\theta})$ is out of the range. Specifically the probability ratio is lower than $[1\epsilon]$, this means that the probability of taking that action at that state is much lower than with the old policy.

    3. Case 3: In this case $A_t > 0$ then the agent wants to increase the probability of taking that action at that state.
    4. Case 4: In this case $A_t < 0$ then the agent does not want to decrease more the probability of taking that action, therefore the gradient will be 0. Indeed as we can see by the left plot of fig. 3.8 the function is a flat line.

- In the last couple of cases (Case 5 and Case 6) we can notice that the ratio $r_t(\boldsymbol{\theta})$ is out of bounds either. However in these two cases we see that $r_t(\boldsymbol{\theta}) > 1+\epsilon$ which implies that the probability of taking that action in that state is much higher in the current policy than in the former one.

    5. Case 5: In this case we see that $A_t > 0$. This means that the action selected drives the agent to a higher reward w.r.t. the average one. Nevertheless we do not want that an agent acts in a too greedy way since the probability of taking that action in that state is already higher than the probability in the old policy. To do so the gradient is set to 0 as we can see from the left plot of the figure 3.8.
    6. Case 6: In this case we have $A_t < 0$ this means that the agent wants to decrease the probability of taking that action in that state, since the expected return by taking that action is lower than the average one.

3.5.3 Cost Function

Since the PPO is implemented as an Actor-Critic method, it is useful to add to the Clipped Objective function other two cost functions [17]. The first one is the value loss and it is related to the goodness of the approximation of the target value function:

$$L^{VF}(\boldsymbol{\theta}) = (V_{\pi_{\boldsymbol{\theta}}} - V^{target})^2 \quad .$$

It is a squared error value loss similar to the one used for the update of the policy network in the DQN (Algo. 5).The second term added is the "*bonus entropy term*" $S[\pi_{\boldsymbol{\theta}}](s_t)$ that is used to ensure sufficient exploration.
Finally the cost function used to improve the policy is the following:

$$L_t^{CLIP+VF+S}(\boldsymbol{\theta}) = \mathbb{E}_t[L_t^{CLIP}(\boldsymbol{\theta}) - c_1 L_t^{VF}(\boldsymbol{\theta}) + c_2 S[\pi_{\boldsymbol{\theta}}](s_t)]$$

3.5.4 Pseudocode

After this analysis we will exploit the pseudocode of the PPO written in the actor-critic style [17].

Algorithm 8 PPO Algorithm in Actor Critic style

Initialize the actor and critic networks with random weights
for iteration = 1 to max_iterations **do**
    Collect a batch of N transitions $(s_i, a_i, r_i, s_i + 1)$ from the environment
    Compute the advantage estimates A_i using the critic network
    $\pi_{old} \leftarrow$ actor network's policy distribution for states s_i
    for k = 1 to K **do**
        $\pi \leftarrow$ actor network's policy distribution for states s_i
        Ratio $\leftarrow \pi(a_i)/\pi_{old}(a_i)$
        Loss $\leftarrow$ -min$(Ratio * A_i, clip(Ratio, 1-\epsilon, 1+\epsilon) * A_i)$
        Update the actor network using the loss
    end for
    Update the critic network using the Mean Squared Error loss between the estimated V-values and the actual V-values
end for

4

Quantum State Stabilization: A discrete-time framework

4.1 Introduction

In this chapter we will present the model that we are going to use in order to both train and subsequently test the reinforcement learning agent. We will start the first section describing the mathematical model that underlines the system we are going to use. After the formal description in the same section we will try to extrapolate from the model the main elements to build the reinforcement learning framework.
The second section will describe the simulated system, highlighting the physical proprieties, and the matrices which are going to represent the more abstract concepts introduced in the previous section. In particular in this section we will put our focus: to the the quantum noises adopted in the simulations, to the definition of the control Hamiltonian and last but not least to the definition of the measurement operators.

4.2 Mathematical Model

In this simulation we consider a quantum system associated with finite dimension complex Hilbert space. We define $\rho(t)$, density operator, as the state of the system at time t. In order to describe our model we define the following notation.

Notation					
Name	**Variable**	**Type**	**Name**	**Variable**	**Type**
State of the system at time t	ρ_t	Density Operator	*Observation of outcome l at time t*	$o_{l,t}$	Real Number
Noise	$\mathcal{N}$	Superoperator	*Control Operation*	$\mathcal{U}_\beta$	Superoperator
Noise Kraus matrix	N_k	Matrix	*Unitary matrix*	U_β	Matrix
Control parameter	β	Complex Number	*Measurement operator at time t with outcome l*	$M^l_{o_{l,t}}$	Matrix
Quantum operation	$\mathcal{E}^\beta$	Superoperator	*Set of actions*	$A_o^{l,\beta}$	Seto of Superoperators

Table 4.1: Notation

The following block scheme describes the starting idea of the model.

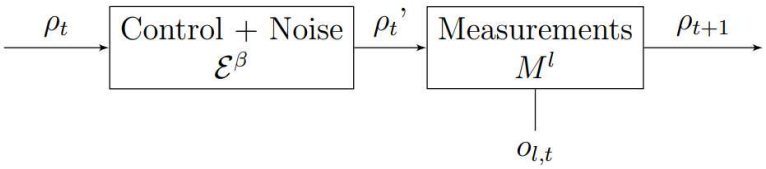


Figure 4.1: First idea of the model

We can notice that the model is composed by two main parts:

- Quantum operations $\mathcal{E}^\beta$
- Quantum Measurements M

Regarding the quantum operation, this is the part that leads the state of our system from ρ_t to ρ'_t. We can express this relationship via the following Kraus representation:

$$\rho'_t = \sum_k E_k^\beta \rho_t E_k^{\beta^\dagger} \quad .$$

The next step is to design the set of matrices which define the quantum superoperator: $\mathcal{E}^\beta = \{E_1^\beta, \ldots, E_n^\beta\}$.
The idea is to define the quantum operation as a combination of two actions: the noise and the control action. We define $\mathcal{N}$ as the noise and $\mathcal{U}_\beta$ the superoperator which describes the evolution of the quantum system in function of the control parameter β. The noise $\mathcal{N}$ is a superoperator defined by the Kraus operators N_k:

$\mathcal{N} = \{N_1, \ldots, N_n\}$.
The unitary matrix $U_\beta = e^{-iH_c(\beta)}$, where $H_c(\beta)$ is the control Hamiltonian, defines the unitary superoperator $\mathcal{U}_\beta$ which is:

$$\mathcal{U}_\beta(\rho) = U_\beta \rho U_\beta^\dagger \quad .$$

Under this considerations we can define the whole quantum operation as follows:

$$\mathcal{E}^\beta = \mathcal{U}_\beta \circ \mathcal{N} \quad .$$

From this we get that:

$$\mathcal{E}^\beta \implies \{E_k^\beta = U_\beta N_k, \quad \forall \beta, k\} \quad .$$

If we assume that $\mathcal{U}$ is independent from the noise $\mathcal{N}$ we can reinterpret the previous block scheme splitting the quantum operation block into two different blocks which are the *noise* $\mathcal{N}$ and the *control* $\mathcal{U}_\beta$.

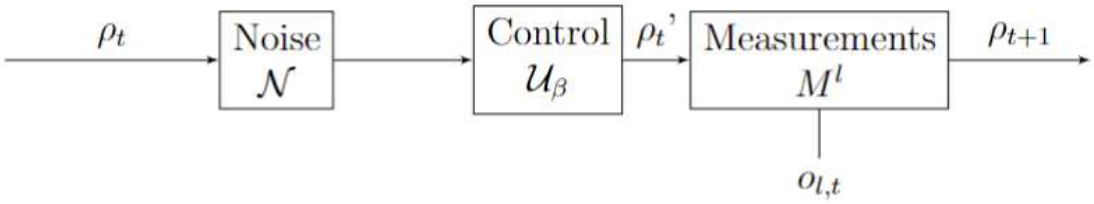


Figure 4.2: Model with split blocks

We define the quantum measurement M as the collection $\{M^l\}$ of measurement operators. Regarding the measurement block, it has two outputs:

- ρ_{t+1}, that is the state at time $t+1$,
- $o_{l,t}$, which is the observation outcome l measured at time t.

Both quantities can be obtained by the quantum mechanics postulates. In particular we know that the observation outcome is ruled by the following result:

$$p(l)_t = tr(M_{o_{l,t}}^\dagger M_{o_{l,t}} \rho_t') \quad .$$

Once we have the outcome result we can condition the state after the measure-

ment according to the measurement postulate (post. 2.3.3) which is:

$$\rho_{t+1|o_{l,t}} = \frac{M_{o_{l,t}} \rho'_t M^{\dagger}_{o_{l,t}}}{tr(M^{\dagger}_{o_{l,t}} M_{o_{l,t}} \rho'_t)} \quad .$$

In the following table we will have a summary of all the main parts of the whole system.

System			
System Part	**I/O**	**Variable**	**Name of the variable**
Noise $\mathcal{N}$	Input	ρ_t	State of the system at time t
	Output	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
Control U_β	Input	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
	Output	ρ'_t	State of the system after the control
Measurement $M^l_{o_{l,t}}$	Input	ρ'_t	State of the system after the control
	Output	ρ_{t+1}	State of the system at time t+1
	Output	$o_{l,t}$	Outcome l of observation at time t

Table 4.2: System Input/Output Table

Putting together all the three blocks we can compute directly ρ_{t+1}, the state of the system at time $t+1$.

$$\rho_{t+1} = \frac{M^l_{o_{l,t}} \rho'_t M^{l^{\dagger}}_{o_{l,t}}}{Tr(\cdot)} = \frac{M^l_{o\,l,t} U_\beta \mathcal{N}(\rho_t) U^{\dagger}_\beta M^{l^{\dagger}}_{o_{l,t}}}{Tr(\cdot)} \quad .$$

From this we can define a set of actions that is:

$$\mathcal{A}^{l,\beta} = \{A^{l,\beta}_o = M^l_o U_\beta\} \quad ,$$

with this last observation we can update our block scheme.

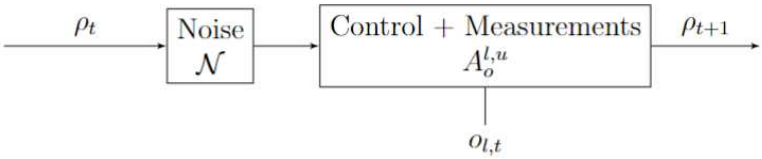


Figure 4.3: Final block scheme

As we can see, putting together the measure and the control blocks, we have

defined a set of actions. Now since the reward function $\mathcal{R}(s, a)$ is a map from $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, it is possible to derive from this setup a starting point to define the reinforcement learning framework.

4.3 Simulated System

First of all we start by defining the vector space in which our system lives. In this simulation we consider quantum system that is described in $\mathcal{H} = \mathbb{C}^3$. In particular we define:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, |2\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

as the spanning vectors. We defined $\rho(t)$, density operator, as the state of the system at time t, with $t \in \mathbb{N}$. The simulated system definition can be split in three main sub-problems: Noise Definition, Unitary evolution definition and Definition of the Measurement operators.

4.3.1 Noise Definition

In this section we will define the quantum noises for the qutrit system used in the simulations. In particular the type of noises adopted are the following:

- Depolarizing Channel
- Damping Channel
- Random Qutrit flip Channel

We start by analyzing the *depolarizing channel*. As for the qubit case treated in the section 2.4.3, it consists in replacing, with probability α the state ρ_t with the completely mixed state and instead with probability $1 - \alpha$ ρ_t is left untouched. The state of the quantum system after this noise is:

$$\mathcal{N}(\rho) = \alpha \frac{I}{3} + (1 - \alpha)\rho \quad .$$

From this notation is easy to see that the α parameter is used as an indicator about the effect of the noise on the evolution of the system.

The second noise we are going to describe is the *amplitude damping channel* for a qutrit system. As in explained in section 2.4.3 the amplitude damping channel reduces the energy of the system due to an interaction with the environment.

The amplitude damping channel is described by the Kraus representation w.r.t. the computational basis $\{|0\rangle, |1\rangle, |2\rangle\}$ by the following matrices [9]:

$$N_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{1-\gamma_1} & 0 \\ 0 & 0 & \sqrt{1-\gamma_2-\gamma_3} \end{pmatrix} \quad , \quad N_{01} = \begin{pmatrix} 0 & \sqrt{\gamma_1} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$N_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \sqrt{\gamma_2} \\ 0 & 0 & 0 \end{pmatrix} \quad , \quad N_{03} = \begin{pmatrix} 0 & 0 & \sqrt{\gamma_3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad ,$$

with the following constraint:

$$\begin{cases} 0 \leq \gamma_i \leq 1 \quad \forall i \in \{1,2,3\} \\ \gamma_2 + \gamma_3 \leq 1 \end{cases} .$$

It is possible to visualize the effect of this quantum noise in the following figure.

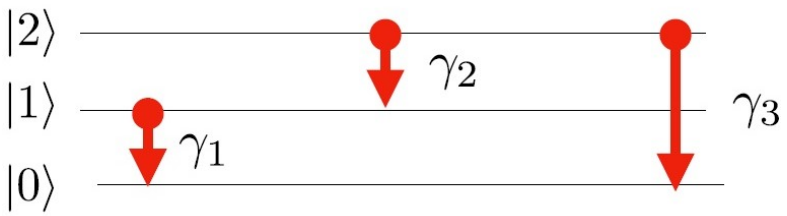


Figure 4.4: Schematic representation of the Amplitude Damping Channel for a qutrit system

In particular in the simulated model we will choose the γ_i parameters as follows:

$$\begin{cases} \gamma_1 = 0 \\ \gamma_2 + \gamma_3 = \alpha \\ \gamma_2 = \gamma_3 \end{cases} .$$

This choice will lead to the definition of the Kraus representation for the noise superoperator:

$$N_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sqrt{1-\alpha} \end{pmatrix} \quad , N_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{\alpha}{2}} \\ 0 & 0 & 0 \end{pmatrix} \quad , \quad N_{03} = \begin{pmatrix} 0 & 0 & \sqrt{\frac{\alpha}{2}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad ,$$

Also in this case we can see as the α parameter plays an important role in the simulated system, indeed it represents as before an index regarding the effect of the noise on the system dynamics.

The third type of noise that we have considered is the *random qutrit flip channel*. This type of quantum noise consists into exchange the energy of the states. In order to describe it for a qutrit system we will use a set of permutation matrices. In particular we write:

$$N_0 = \sqrt{1 - \frac{2\alpha}{3}} I \quad , \quad N_1 = \sqrt{\frac{\alpha}{3}} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad , \quad N_2 = \sqrt{\frac{\alpha}{3}} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

In this case the parameter α as in the depolarizing channel noise defines the probability of the qutrit system being flipped.

4.3.2 Unitary Evolution

For what concern the control, we choose to implement a impulse control, with which the evolution of the quantum system is only related to the controlled Hamiltonian H_c. In particular we define as unitary evolution matrix:

$$U_\beta(t) = e^{-iH_c(\beta(t))} \quad \text{with } \beta(t) \text{ the control parameter.}$$

Now we define the control Hamiltonian H_c as:

$$H_c = i[\beta(t)a - \beta(t)^* a^\dagger] \quad \text{with } \beta : t \to \mathbb{C} \text{ and}$$

$$a = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad .$$

At this point we are able to define the unitary superoperator $\mathcal{U}_\beta(\rho)$ which is:

$$\mathcal{U}_\beta(\rho) = U_\beta \rho U_\beta^\dagger \quad .$$

4.3.3 Definition of the quantum measurement

The measurement operator is defined as a set of matrices that respect the behaviour of a Kraus operator. The set of matrices is $\{M_0, M_1, M_2\}$ and has to fulfill the following constraint:

$$I = \sum_{i=0} M_i^\dagger M_i$$

In particular in the simulated system we take into account an "imprecise" version of the projective measurements [14] presented in the definition 2.2.4. The level of inaccuracy w.r.t. a projective measurement is represented by the parameter ϵ. This type of measurement provides some useful but uncertain information with the advantage of avoiding the state collapse. In order to represent this concept we define the following Kraus representation of the quantum measurement:

$$M_0^\dagger M_0 = \begin{pmatrix} 1-2\epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{pmatrix}, M_1^\dagger M_1 = \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1-2\epsilon & 0 \\ 0 & 0 & \epsilon \end{pmatrix} M_2^\dagger M_2 = \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & 1-2\epsilon \end{pmatrix} \quad ,$$

from this we can define the set $\{M_0, M_1, M_2\}$ as:

$$M_0 = \begin{pmatrix} \sqrt{1-2\epsilon} & 0 & 0 \\ 0 & \sqrt{\epsilon} & 0 \\ 0 & 0 & \sqrt{\epsilon} \end{pmatrix}, M_1 = \begin{pmatrix} \sqrt{\epsilon} & 0 & 0 \\ 0 & \sqrt{1-2\epsilon} & 0 \\ 0 & 0 & \sqrt{\epsilon} \end{pmatrix}, M_2 = \begin{pmatrix} \sqrt{\epsilon} & 0 & 0 \\ 0 & \sqrt{\epsilon} & 0 \\ 0 & 0 & \sqrt{1-2\epsilon} \end{pmatrix} \quad .$$

4.4 Control Task

The control problem we want to address consists in find a feedback law based on the measurements so that β, the control parameter, can be defined from the outcomes of the measurements in order to maximize the probability of reaching the target state $|2\rangle$.

A trivial solution to this problem can be defined in the special case in which we impose $\epsilon = 0$ and $\alpha = 0$. It consists in iterating a projective measurement and, if the outcome is not the expected one, a control action which is able to drive the target state probability far from zero. After a certain time it is possible to prove that this algorithm will converge to the target solution.

In order to find a policy for more complex problems we try to use the reinforcement learning framework.

5 Simulations

5.1 Introduction

In this chapter, we present the simulation results of a series of experiments aimed at addressing a complex control problem. The chapter is divided into three main subsections, each focusing on a different aspect of the experimental process. In the first subsection, we introduce the control problem at hand, highlighting its intricacies and the challenges it poses. We discuss the various approaches that have been taken to address it. In the second subsection, we present the training results, detailing the steps taken to prepare our models and the performance metrics used to evaluate them. We describe the various training algorithms used and the techniques employed to optimize model parameters. Finally, in the third subsection, we introduce the test results, which provide an evaluation of our models' performance under real-world conditions. We analyze the results and draw conclusions on the efficacy of our approach in solving the control problem. Overall, this chapter provides a resume of our experimental process and the results obtained, which can be useful in informing future research in this area.

5.2 Simulation Models and Training Results

5.2.1 Introduction to the problem

We introduce the *control problem* in the following way. Let us define ρ_0 as the density matrix at the beginning, time $t = 0$, and ρ_{goal} as the target density matrix that we want to reach. We define T as the maximum time in which we have to reach our target. We define two cases of control problem, which will influence the initial setup:

- *State Transfer*
  - ρ_0 Known or Estimated (for example $\rho_0 = \frac{1}{N}I$)
- *Stabilization*
  - ρ_0 unknown

The purpose for both cases is to start from ρ_0 and try to reach the target state ρ_{goal} in a maximum of T timesteps using the control action β which affects the system dynamics. In all the simulations we will consider as starting density matrix and target density matrix the following density operators:

$$\rho_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad , \quad \rho_{goal} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad .$$

In order to choose the control action β we will work with the reinforcement learning framework. In particular we will develop three main setups:

- MDP
  - In this setup we treat the quantum control problem as if it were a classical reinforcement learning problem. The agent is provided at each time-step with the state of the environment as if it was fully observable and a reward computed in function of the state. While this setup is not physically realizable it provides a reference benchmark for the model.
- POMDP

  - In this setup we begin to remove some "*quantum knowledge*" from the information provided to the agent. In particular we provide to the agent a new type of state that is different from the natural one, it receives just the action performed at the previous time-step and the previous outcome of the measurement. This model is not yet *totally* model free since during the training the agent receives a reward that is computed in function of the state of the environment (in function of the density operator) and therefore also in this case we need to exploit some knowledge about the quantum system. Moreover we want to specify that also in this case due to the reward, also this setup is not physically implementable.

- QOMDP

  - This is the model free setup in which we provide to the agent just classical information without any knowledge about the dynamic of the quantum system. Also in this case the state is built as in the POMDP setup: outcome of the measurement and previous control action. Further the reward is not computed in function of the state of the system. We want to highlight that this is the only setup that can be physically realized.

The main idea is to study and compare the behaviour of these three setups in order to understand the differences in performance, starting by providing all the possible information about the system and going further by reducing the quantity of *quantum information* about the system until we reach a completely Model Free framework. At the same time we will delve deeper into the interplay between the ϵ parameter which weights the measurement imprecision and the α parameter which corresponds to the amount of noise that affects the system dynamics.

5.2.2 Models and Train Results

Training MDP model

First of all we define the block scheme of the simulated model.

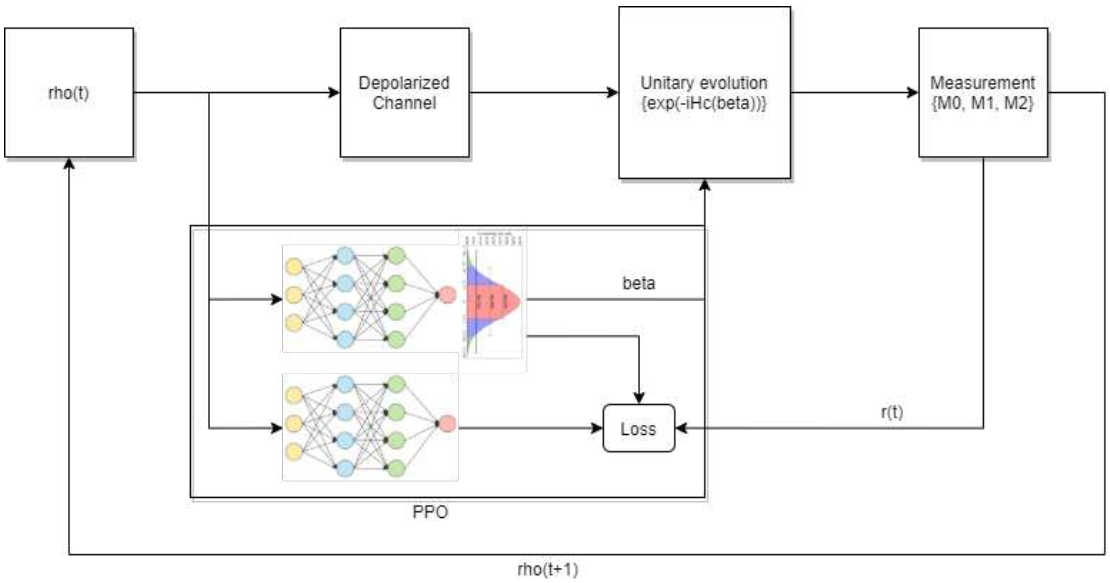


Figure 5.1: System Block Scheme

The main blocks of the entire simulated model are summarized in the following table.

System			
System Part	**I/O**	**Variable**	**Name of the variable**
Depolarized Channel	Input	ρ_t	State of the system at time t
	Output	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
Reinforcement Learning Agent (PPO)	Input	ρ_t	State of the system at time t
	Output	β	Control Action
Unitary Evolution	Input	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
	Input	β	Control Action
	Output	ρ'_t	State of the system after the control
Measurement $M^l_{o_{l,t}}$	Input	ρ'_t	State of the system after the control
	Output	ρ_{t+1}	State of the system at time t+1
	Output	$o_{l,t}$	Outcome l of observation at time t

Table 5.1: Entire System: Input/Output Table

It is possible to see from both the block scheme (Fig. 5.1) and from the below table (Table 5.1) that the agent is not aware of the noise, indeed the input fed

to the agent is *"captured"* before the noise block. We have chosen to train a different agent for every value of the parameter ϵ which describes how far is the measurement from a projective measurement. Moreover all the different agents are trained without the noise. In particular we recall that the noise is weight by the parameter α. During all the training session we will fix $\alpha = 0$. Another key point to set is the definition of the reward function. In this case we will consider the fidelity. The reward is equal to 0 if the agent has not reached the target fidelity or the maximum number of timesteps. Whenever one of this two condition is met the the agent will receive as reward the fidelity. This concept is better formalized in the following equations:

$$F(\rho, \sigma) = (tr[\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}])^2 \quad .$$

In particular the reward function is described by the following function:

$$R(\rho_t, \rho_{goal}, done) = \begin{cases} 0, & \text{if } F(\rho_t, \rho_{goal}) \leq F_{target} \text{ or if not reached maximum timesteps,} \\ F(\rho_t, \rho_{goal}), & \text{if } F(\rho_t, \rho_{goal}) > F_{target} \text{ or if reached maximum timesteps} \end{cases} .$$

The ϵ and α values used in the training are summarized in the following table.

Training Parameters	*Values*					
ϵ	0.1	0.15	0.175	0.2	0.25	0.3
α	0					

Table 5.2: Training Parameters

Regarding the control parameter, we can define the following constraint: $\beta \in [-1, 1]$.
The steps in the training are the following:

1. The state $\rho(t)$ is fed both to the Reinforcement Learning agent;
2. The agent outputs the control action β ;
3. The state evolves according to the unitary operator represented by the evolution block;
4. We measure the state;
5. We compute the reward accordingly to the fidelity;
6. We compute the loss through the fidelity;

7. We compute the state at time $t + 1$: $\rho(t + 1)$;

The parameters with which the agent is set up are summarized in the following table.

Parameter	**Learning Rate**	**Batch Size**	**N_steps**	**Value_func_coeff**	**Entropy_Coeff**
Values	$1e^{-4}$	512	512	0.5	0

Table 5.3: Default PPO hyperparameters

We let the agent learn for *timesteps* = 50000. In the following plots we can understand the evolution of the agent's knowledge.

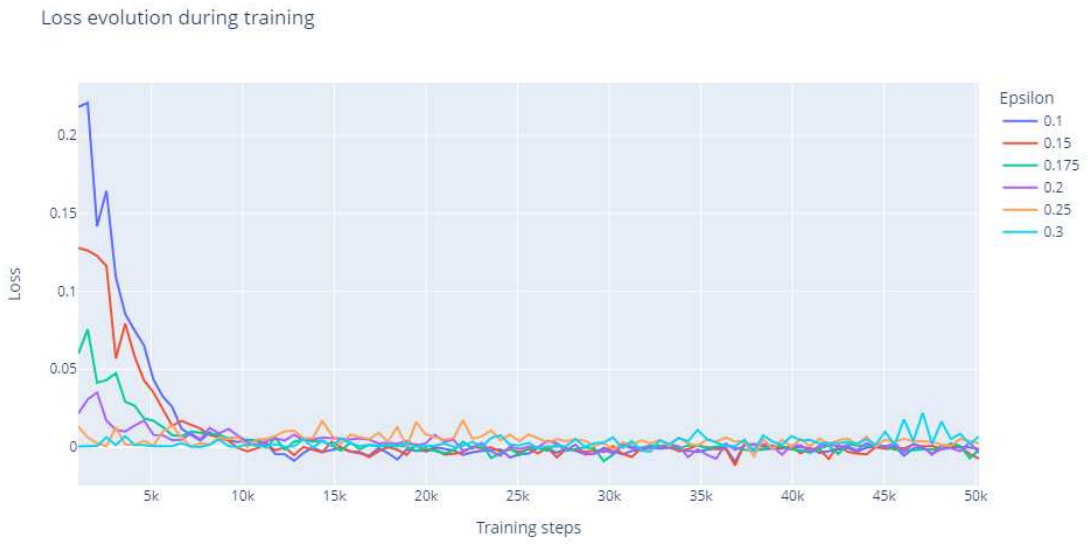


Figure 5.2: Loss evolution during training

Figure 5.3: Timesteps evolution in training

We can see from all of the plots that the algorithm is converging or can converge. In the figure (5.3) we can notice that the number of timesteps to reach the target fidelity is decreasing during the whole training for all the agents, even if the agent has still reached the target fidelity. This means that the agents want to find not only the policy that leads the state to the target, but the policy that is able to reach the target in the fewest steps possible.
In figure (5.2) we can see that during the training all the agents' loss are decreasing. Just after the training of each agent, we test its behaviour on the training environment for 100 episodes. For each environment we are going to store the mean fidelity reached, the fidelity standard deviation, the mean timesteps and its standard deviation.
All these results are summarized in the following table (5.4).

Parameters		**Results**			
ϵ	α	Fidelity	Fidelity Std	Timesteps	Timesteps Std
0.1	0	0.98	0.01	3.45	1.69
0.15	0	0.97	0.012	5.4	6.65
0.175	0	0.96	0.009	6.05	6.52
0.2	0	0.97	0.014	5.7	6.5
0.25	0	0.97	0.01	36.25	41.65
0.3	0	0.97	0.004	17.95	1.9

Table 5.4: Training Results

As we can see from the above table, in the training environment the various agents are reaching the fidelity target. Moreover as expected we can see that as far as the ϵ parameter grows, also the timesteps to reach the target grow too.

Training POMDP model

First of all we define the block scheme of the simulated model.

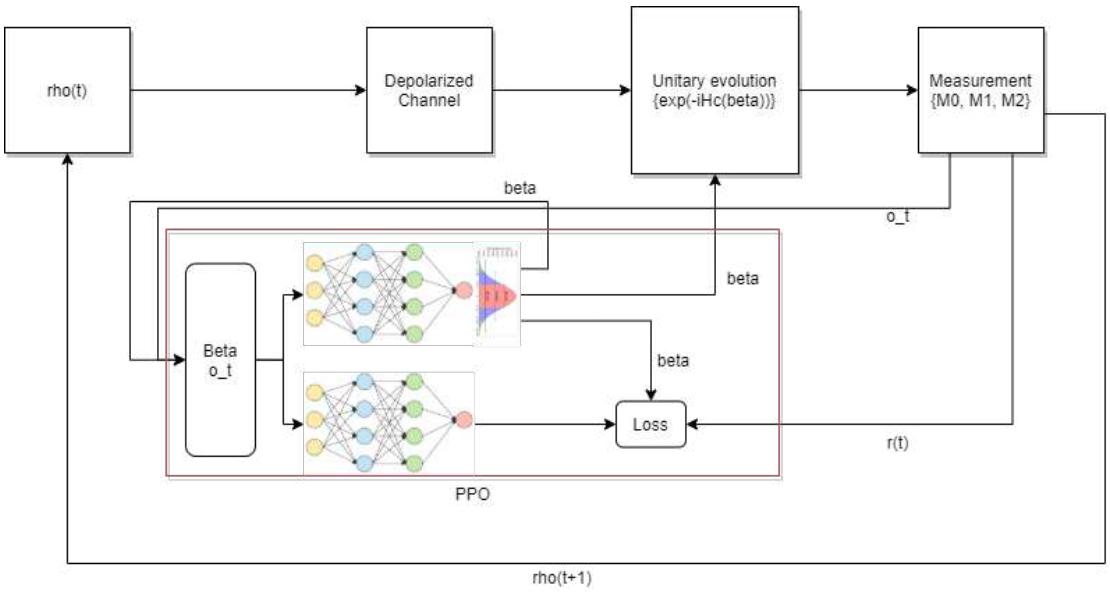


Figure 5.4: System Block Scheme

The main blocks of the entire simulated model are summarized in the following table.

System			
System Part	**I/O**	**Variable**	**Name of the variable**
Depolarized Channel	Input	ρ_t	State of the system at time t
	Output	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
Reinforcement Learning Agent (PPO)	Input	β_{t-1}	Control action at time $t-1$
	Input	o_{t-1}	Observation outcome at time $t-1$
	Output	β	Control Action
Unitary Evolution	Input	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
	Input	β	Control Action
	Output	ρ'_t	State of the system after the control
Measurement $M^l_{o_{l,t}}$	Input	ρ'_t	State of the system after the control
	Output	ρ_{t+1}	State of the system at time t+1
	Output	$o_{l,t}$	Outcome l of observation at time t

Table 5.5: Entire System: Input/Output Table

It is possible to see from both the block scheme (Fig. 5.4) and from the below table (Table 5.5) that the input that we are going to fed to the agent is composed by the outcome of the measurement at time t (o_t) and by the control action performed at time t (β_t). At time $t = 0$ we perform no action $\beta = 0$, this leads to have a unitary evolution equal to the identity $U(0) = I_{n\times n}$, then we perform the measurement. In this way the first state fed to the agent will be always $[0, o_{t=0}]^T$. In this setup, except for the computation of the reward we will never use the density operator ρ. The reward, as for the MDP case is computed accordingly to the fidelity between ρ and the target state ρ_{goal}:

$$F(\rho, \sigma) = (tr[\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}])^2 \quad .$$

In particular the reward function is described by the following function:

$$R(\rho_t, \rho_{goal}, done) = \begin{cases} 0, & \text{if } F(\rho_t, \rho_{goal}) \leq F_{target} \text{ or if not reached maximum timesteps,} \\ F(\rho_t, \rho_{goal}), & \text{if } F(\rho_t, \rho_{goal}) > F_{target} \text{ or if reached maximum timesteps} \end{cases} .$$

As for the MDP case we will train a different agent for every value of the ϵ parameter. Moreover all the different agents are trained without the noise. In particular we recall that the noise is weighted by the parameter α, so during all the training session we will fix $\alpha = 0$. The ϵ and α values used in the training are summarized in the following table.

Training Parameters	*Values*					
ϵ	0.1	0.15	0.175	0.2	0.25	0.3
α	0					

Table 5.6: Training Parameters

Regarding the control parameter, we can define the following constraint: $\beta \in [-1, 1]$.
The steps at a generic time t ($t \neq 0$) of the training procedure are the following:

1. The observation performed at time $t - 1$ and the previous control action β_{t-1} are fed to the Reinforcement Learning agent
2. The agent outputs the time t control action β_t
3. The state ρ_t evolves according to the unitary operator which depends on β_t
4. We measure the state and we get the outcome o_t
5. If the episode is ended we compute the reward accordingly to the fidelity on the other hand the reward is 0.
6. We compute the loss through the reward obtained
7. We compute the state at time $t + 1$: $\rho(t + 1)$

The parameters with which the agent is set up are summarized in the following table.

Parameter	**Learning Rate**	**Batch Size**	**N_steps**	**Value_func_coeff**	**Entropy_Coeff**
Values	$1e^{-4}$	512	512	0.5	0

Table 5.7: Recurrent PPO hyperparameters

We let the agent learn for *timesteps* = 30000. In the following plots we can understand the evolution of the agent's knowledge.

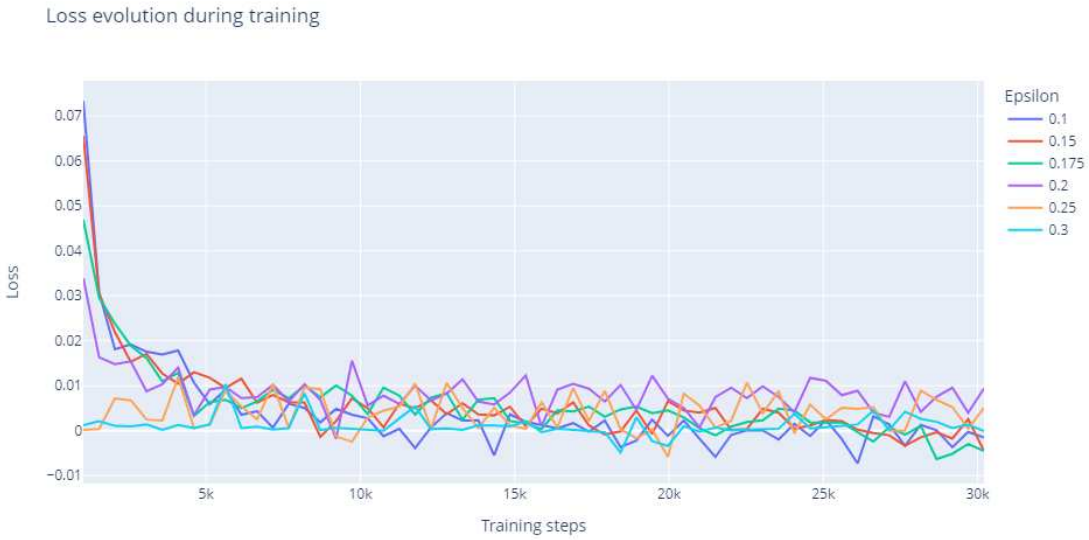


Figure 5.5: Loss evolution during training

Figure 5.6: Timesteps evolution in training

We can see from all of the plots that the algorithm is converging for most of the agents. In the figure (5.6) we can notice that the number of timesteps to reach the target fidelity is decreasing during the whole training for all the agents, even if the agent has still reached the target fidelity as we can see in table 5.8. This means that the agents want to find not only the policy that leads the state to the

target, but the policy that is able to reach the target in the fewest steps possible. In figure (5.5) we can see that during the training all the agents' loss are decreasing. Just after the training of each agent, we test its behaviour on the training environment for 100 episodes. For each environment we are going to store the mean fidelity reached, the fidelity standard deviation, the mean timesteps and its standard deviation.
All these results are summarized in the following table (5.8).

Parameters		Results			
ϵ	α	Fidelity	Fidelity Std	Timesteps	Timesteps Std
0.1	0	0.97	0.01	12.05	12.02
0.15	0	0.97	0.01	9.35	9.24
0.175	0	0.97	0.01	7.40	8.07
0.2	0	0.97	0.01	10.55	4.07
0.25	0	0.97	0.01	16.55	5.73
0.3	0	0.96	0.01	60.15	10.71

Table 5.8: Training Results POMDP setup

As we can see from the above table, we can see that in the training environment the various agents are reaching the fidelity target.

Train QOMDP agent setup [19][14]

First of all we define the block scheme of the simulated model.

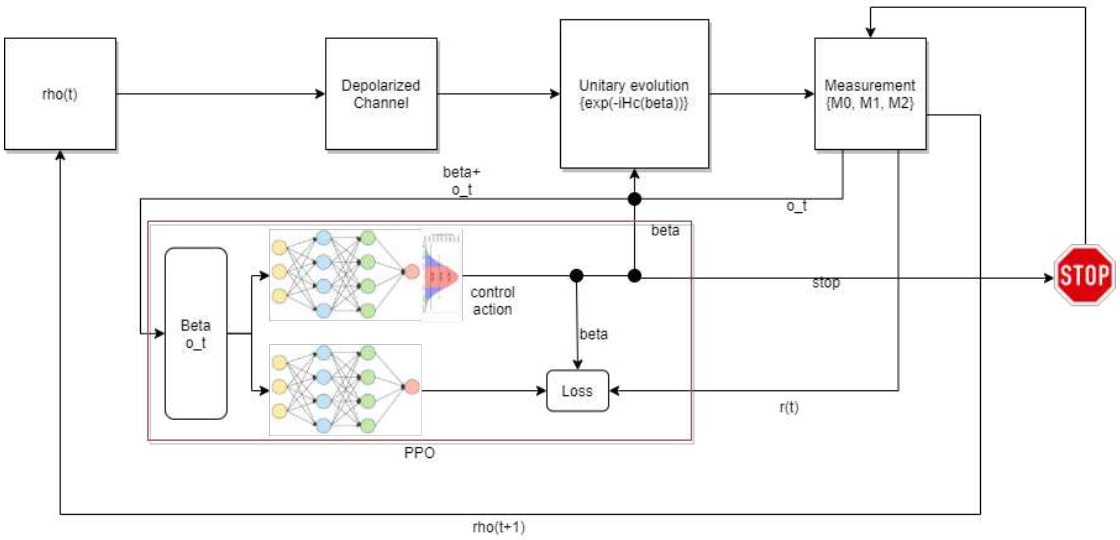


Figure 5.7: System Block Scheme

The main blocks of the entire simulated model are summarized in the following table.

System			
System Part	**I/O**	**Variable**	**Meaning**
Depolarized Channel	Input	ρ_t	State of the system at time t
	Output	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
Reinforcement Learning Agent (PPO)	Input	ρ_t	State of the system at time t
	Output	β	Control Action
	Output	Stop	Ends the Episode
Unitary Evolution	Input	$\rho_{t\|\mathcal{N}}$	State of the system after the noise
	Input	β	Control Action
	Output	ρ'_t	State of the system after the control
Measurement $M^l_{o_{l,t}}$	Input	ρ'_t	State of the system after the control
	Input	Stop	Perform the last measurement of the episode
	Output	ρ_{t+1}	State of the system at time t+1
	Output	$o_{l,t}$	Outcome l of observation at time t

Table 5.9: QOMDP: Input/Output Table

It is possible to see from both the block scheme (Fig. 5.7) and from the above table (Table 5.9) that the input that we are going to fed to the agent is composed by the outcome of the measurement at time t (o_t) and by the control action performed at time t (β_t). At time $t = 0$ we perform no action $\beta = 0$, $stop = 0$, this leads to have a unitary evolution equal to the identity $U(0) = I_{n\times n}$, then we perform the measurement. In this way the first state fed to the agent will be always $[0, o_{t=0}]^T$. With the aim of describing this setup for a time $t \neq 0$ we have first to introduce a new set of measurement operator. We will call *last observation* (o_last) the observation that is measured from the set of projective measurements $\{M_0^{end}, M_1^{end}, M_2^{end}\}$:

$$M_0^{end} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad M_1^{end} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad M_2^{end} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad .$$

Clearly this set of matrices respects the identity condition. The main idea is that when the agent decide to stop the episode, we perform one last measurement. This last measurement is indeed performed accordingly to the projective

measurement just described. This leads to a new idea of reward:

$$R(stop, o_{last}, done) = \begin{cases} 0, & \text{if } stop = 0 \text{ and done = False} \\ -1, & \text{if } stop = 0 \text{ and done = True} \\ r_{function}(o_{last}), & \text{if } stop = 1 \end{cases},$$

where for $r_{function}$ we mean the following function:

$$r_{function} : \Omega \rightarrow \{-1, +1\}$$

$$r_{function} = \begin{cases} +1, & \text{if } o_{last} \text{ correspond to the target state outcome} \\ -1, & \text{whatever else} \end{cases}$$

We recall that the probability of the outcome (m) of o_{last} is defined by:

$$p(m) = Tr(M^{end\dagger}_{m} M^{end}_{m} \rho_t) \quad \text{with } \rho_t \text{ density operator.}$$

For analysis purposes we will compute and take in memory also the fidelity (in particular the fidelity with which we will compare the results is computed when the stop action is performed but before the projective measurement), but as we can see the agent does not know the state of the quantum system and moreover the density operator is neither used to compute the reward. Acting in this way we have built a system that works using the QOMDP formalism. As for two previous cases we will train a different agent for every value of the ϵ parameter. Moreover all the different agents are trained without the noise. In particular we recall that the noise depends on the parameter α , so during all the training session we will fix $\alpha = 0$. The ϵ and α values used in the training are summarized in the following table.

Training Parameters	*Values*					
ϵ	0.1	0.15	0.175	0.2	0.25	0.3
α	0					

Table 5.10: Training Parameters

Regarding the control parameter, we can define the following constraint: $\beta \in [-1, 1]$.
To describe the main steps at a generic time t ($t \neq 0$) of the training procedure

we have to distinguish two cases:

1. In the timestep $t-1$ that we are considering the *flag* $stop = 0$

   (a) The observation performed at time $t-1$ and the previous control action β_{t-1} are fed to the Reinforcement Learning agent

   (b) The agent outputs the time t control action β_t and the $stop$ signal

   - if $stop = 1$ then we do not perform the action and the following measurement (we pass directly to the 2nd case of the list)
   - if $stop = 0$ then we can continue to perform the action

   (c) The state ρ_t evolves according to the unitary operator which depends on β_t

   (d) We measure the state and we get the outcome o_t

   (e) If the episode has not reached the maximum number of timesteps we return $reward = 0$

   (f) We compute the loss through the reward obtained

   (g) We compute the state at time $t+1$: $\rho(t+1)$

2. In the timestep $t-1$ that we are considering the *flag* $stop = 1$

   (a) The last observation o_last is performed accordingly to the set of matrices $\{M_0^{end}, M_1^{end}, M_2^{end}\}$

   (b) The final reward is computed accordingly to the previous functions.

The parameters with which the agent is set up are summarized in the following table.

Parameter	**Learning Rate**	**Batch Size**	**N_steps**	**Value_func_coeff**	**Entropy_Coeff**
Values	$1e^{-4}$	512	512	0.5	0

Table 5.11: Recurrent PPO hyperparameters

We let the agent learn for *timesteps* = 200000. In the following plots we can understand the evolution of the agent's knowledge.

Figure 5.8: Reward evolution in training

Figure 5.9: Timesteps evolution in training

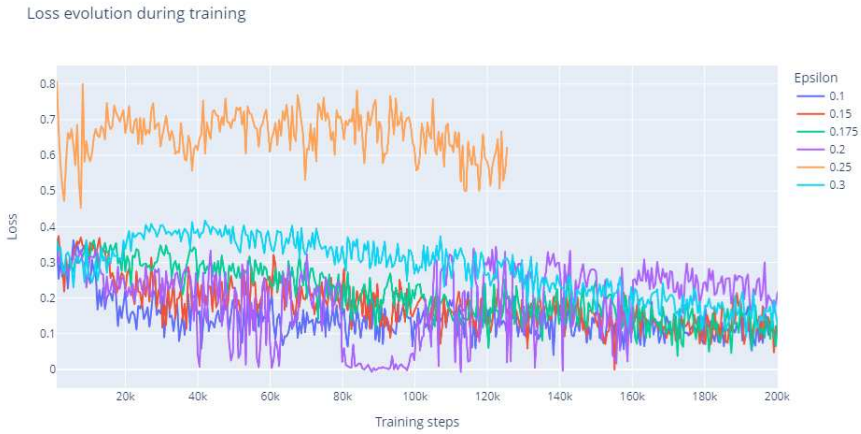


Figure 5.10: Loss evolution during training

We can see from almost all of the plots that the RL agents are converging in the reward, timesteps and also in loss. In particular in figure (5.8) we see that all the agents are converging to a neighbourhood of 1. We recall that in this setup the reward does not correspond to the fidelity. However from this plot we can see that the slope of almost all the curves is still a little bit positive, this means that probably the agents to converge to the optimal or sub-optimal policy could need some more training steps.
In the figure (5.9) we can notice that the number of timesteps to reach the target fidelity is not decreasing during the whole training. Instead we can observe that during the first 30k-40k of training steps the number of timesteps per episode is increasing, this is a clear sign of exploration. After the 40k of trianing steps we can see that for almost all the agents the number of timesteps per episode is decreasing or is stable.

In figure (5.10) we can see that during the training all the agents' loss are decreasing. For what concern the loss training data for the agent $\epsilon = 0.25$, we can observe that its data ended at training step 120k. This is probably due to an error of tensorboard, the platform managed to store the data. Just after the training of each agent, we test its behaviour on the training environment for 100 episodes. For each environment we are going to store the mean fidelity reached, the fidelity standard deviation, the mean timesteps and its standard deviation. All these results are summarized in the following table (5.12).

Parameters		**Results**					
ϵ	α	Reward	Reward Std	Timesteps	Timesteps Std	Fidelity	Fidelity Std
0.1	0	1.00	0.00	2.50	1.02	0.97	0.01
0.15	0	1.00	0.00	3.10	1.81	0.96	0.08
0.175	0	1.00	0.00	4.70	3.61	0.96	0.04
0.2	0	1.00	0.00	3.70	2.93	0.93	0.13
0.25	0	1.00	0.00	5.70	2.61	0.95	0.05
0.3	0	1.00	0.00	2.00	0.00	0.96	0.00

Table 5.12: Training Results

5.3 Test setup

5.3.1 Simulation parameter

First we define the test environment. We recall that during the training the α parameter that defines the probability of having noise in the environment was fixed to zero. During the test the α parameter will assume different values, in this way we will be able to compare the various results in order to identify how robust are the agents w.r.t. the noise. The α and ϵ parameters with which the various agents are tested are the resumed in the following table 5.13.

	α										
	0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
ϵ	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	0.175	0.175	0.175	0.175	0.175	0.175	0.175	0.175	0.175	0.175	0.175
	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Table 5.13: Test parameters

5.3.2 MDP Test Setup

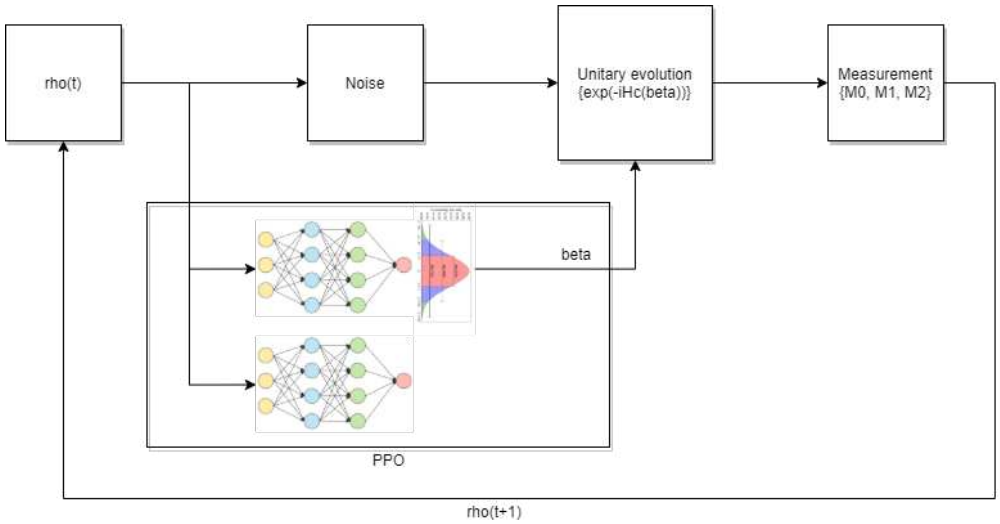


Figure 5.11: System Block Scheme (MDP)

As depicted in the table above (table 5.13) we will test the agents for every ϵ all and for all the α values. In particular for every combination of ϵ and α we are going to run 100 simulation. At the end of every simulation we will keep track of the fidelity/reward value achieved by the agent, another parameter which we will be stored and analyzed is the number of timesteps to reach the target state. In this setup we can interpret it as an index of how complex is for the agent solve the test environment (figure 5.11).

The main difference with the train procedure is that in the test the agent do not have to compute the reward to calculate the loss. The main steps in the test are the following:

1. The state $\rho(t)$ is fed both to:

   - Noise Block,
   - Reinforcement Learning agent;

2. At the same time:

   - The noise can either change or not the state;
   - The agent outputs the control action β ;

3. The state evolves according to the unitary operator represented by the evolution block;

4. We measure the state;

5. We compute the state at time $t + 1$: $\rho(t + 1)$;

5.3.3 POMDP Test Setup

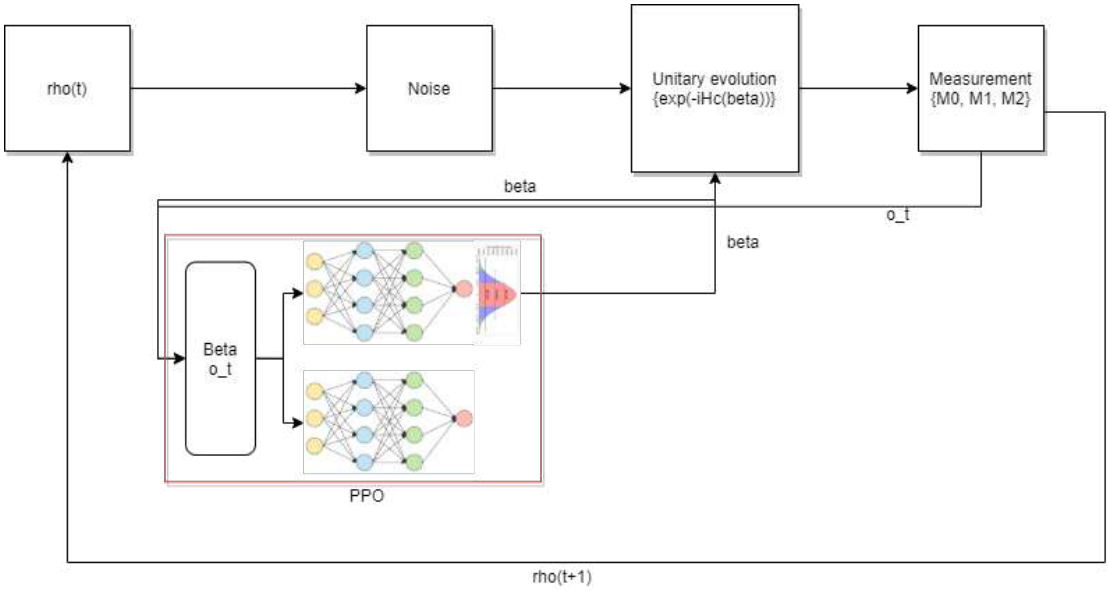


Figure 5.12: System Block Scheme (POMDP)

As depicted in the table above (table 5.13) we will test for every ϵ all the α values. In particular for every combination of ϵ and α we are going to run 100 simulation. At the end of every simulation we will keep track of the fidelity/reward value achieved by the agent, another parameter which we will be stored and analyzed is the number of timesteps to reach the target state. In this setup we can interpret it as an index of how complex is for the agent solve the test environment (figure 5.12).

The main difference with the train procedure is that in the test the agent do not have to compute the reward to calculate the loss. The main steps (for $t \neq 0$) in the test setting are the following:

1. At the same time:
   - The observation measured at time $t-1$ and the control action performed at time $t-1$, β_{t-1}, are fed to the Reinforcement Learning agent;
   - The state of the system represented by the density operator ρ_t enters in the noise block;

2. The evolution block receives in input the state of the system $\rho_{t|Noise}$ and the control action β_t given in output by the RL agent;

3. The state ρ_t is measured and the observation o_t is obtained;
4. We obtain the state at time $t + 1$ represented by the density operator ρ_{t+1};

5.3.4 QOMDP Test Setup

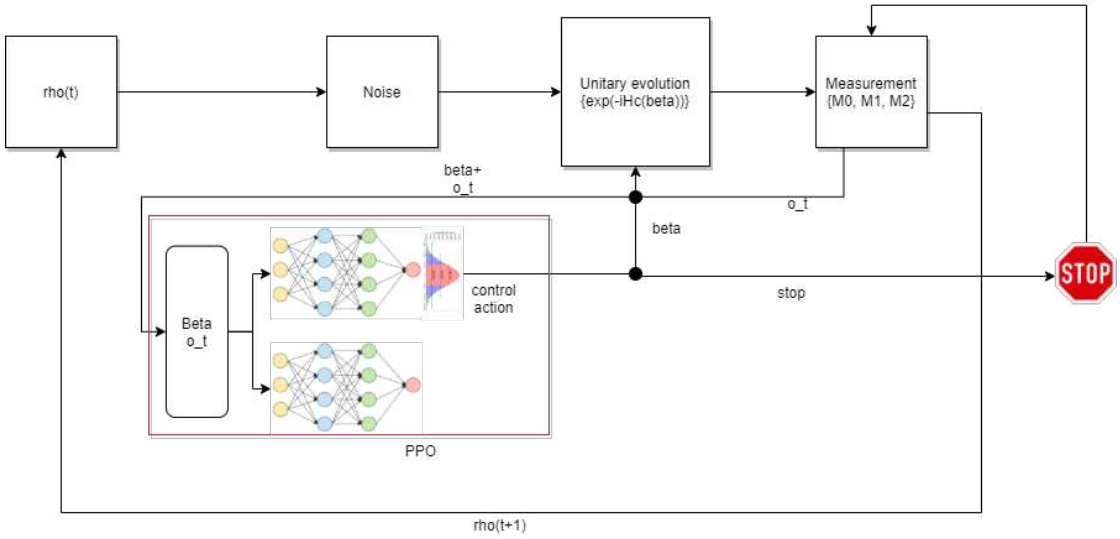


Figure 5.13: System Block Scheme (QOMDP)

As depicted in the table above (table 5.13) we will test for every ϵ all the α values. In particular for every combination of ϵ and α we are going to run 100 simulation. At the end of every simulation we will keep track of the fidelity, the reward values achieved by the agent, another parameter which we will be stored and analyzed is the number of timesteps that the agent chooses to *play*. In this setup we can interpret it as an index of how complex is for the agent solve the test environment (figure 5.13).

The main difference with the train procedure is that in the test the agent do not have to compute the reward to calculate the loss. The main steps (for $t \neq 0$) in the test setting are the following:

1. In the time-step $t - 1$ that we are considering the *flag* $stop = 0$
   (a) The observation performed at time $t - 1$ and the previous control action β_{t-1} are fed to the Reinforcement Learning agent;
   (b) The agent outputs the time t control action β_t and the $stop$ signal;

- if $stop = 1$ then we do not perform the action and the following measurement (we pass directly to the 2nd case of the list);
- if $stop = 0$ then we can continue to perform the action;

(c) The state ρ_t evolves according to the unitary operator which depends on β_t;

(d) We measure the state and we get the outcome o_t;

(e) We compute the state at time $t + 1$: $\rho(t + 1)$;

2. In the time-step $t - 1$ that we are considering the *flag* $stop = 1$;

(a) The last observation o_{last} is performed accordingly to the set of matrices $\{M_0^{end}, M_1^{end}, M_2^{end}\}$;

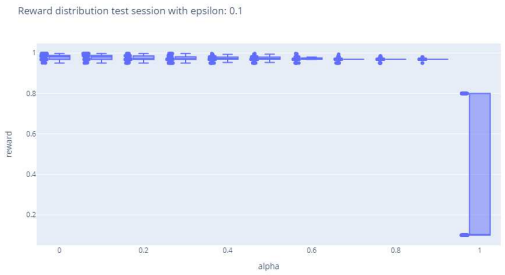
(b) We keep track of the fidelity w.r.t. ρ_{t-1};

5.4 Simulation Results

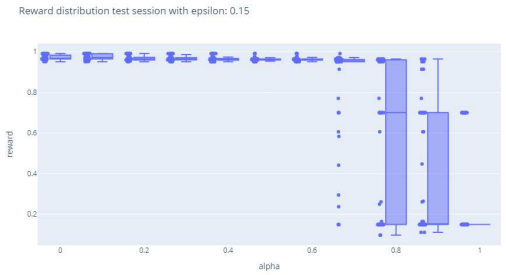
5.4.1 Mean Effect of Depolarizing Channel

MDP setup results

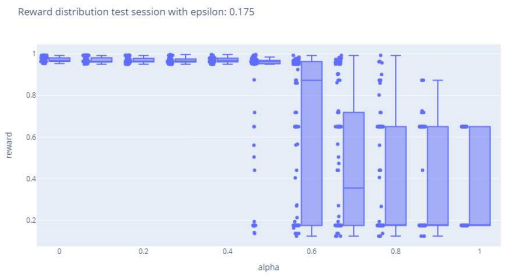
In the following plots (Figure 5.14) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



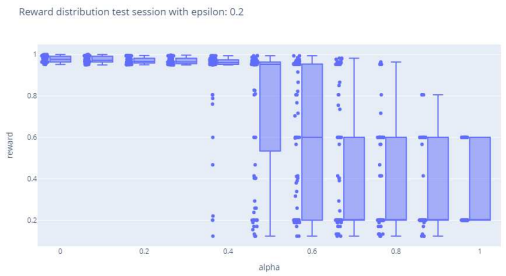
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



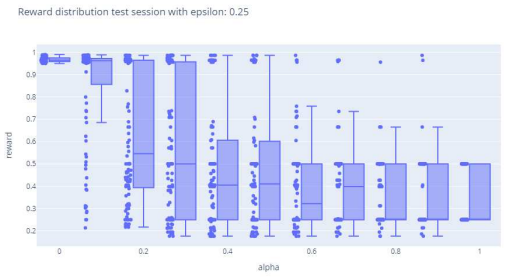
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



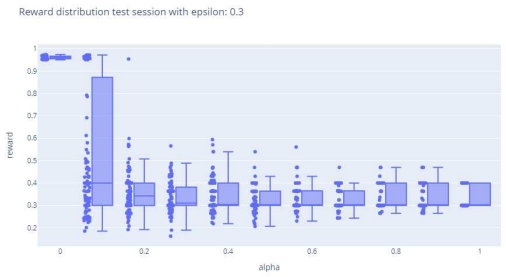
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



(e) Reward/Fidelity distribution with $\epsilon = 0.25$

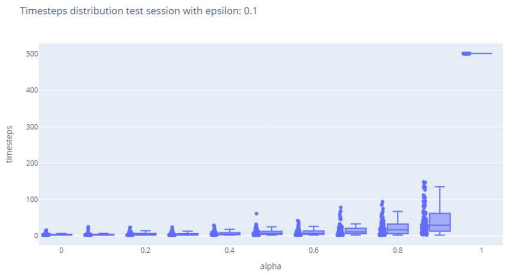


(f) Reward/Fidelity distribution with $\epsilon = 0.3$

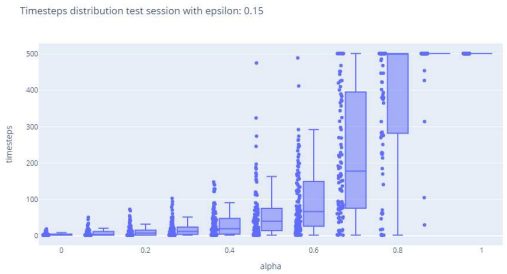
Figure 5.14: Reward/Fidelity distribution of different ϵ over the noise α (MDP)

We can notice that as far as the ϵ parameter grows, the distribution of the

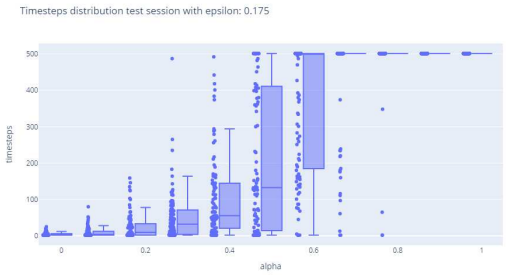
fidelity get worst when we test the agent for high α. Moreover as ϵ grows the *minimum* α with which our agent is able to reach 95% of fidelity decreases. Another key observation is that for high values of ϵ we can notice that the variance of the fidelity in the worst cases is reduced w.r.t. the same α value but with a less ϵ. In the following plots (Figure 5.15) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain quantity of noise α.



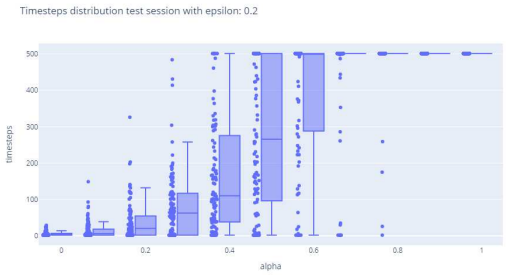
(a) Timesteps distribution with $\epsilon = 0.1$



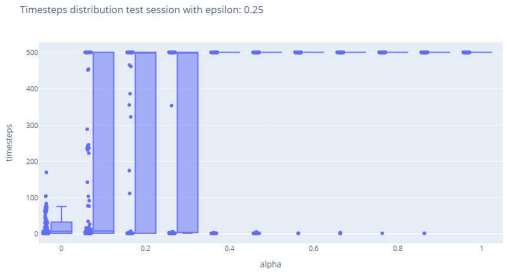
(b) Timesteps distribution with $\epsilon = 0.15$



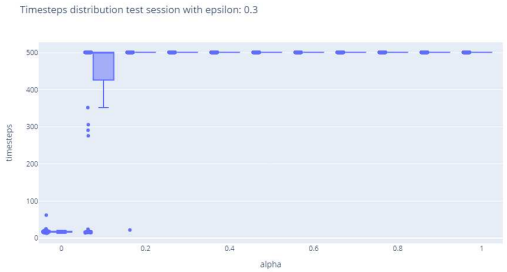
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

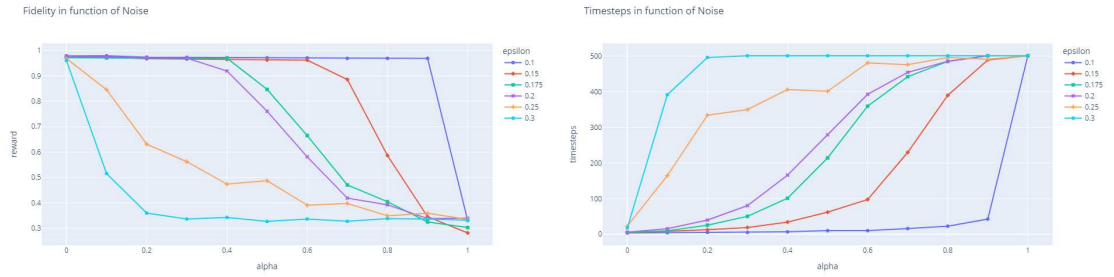


(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.15: Timesteps distribution of different ϵ over the noise α (MDP)

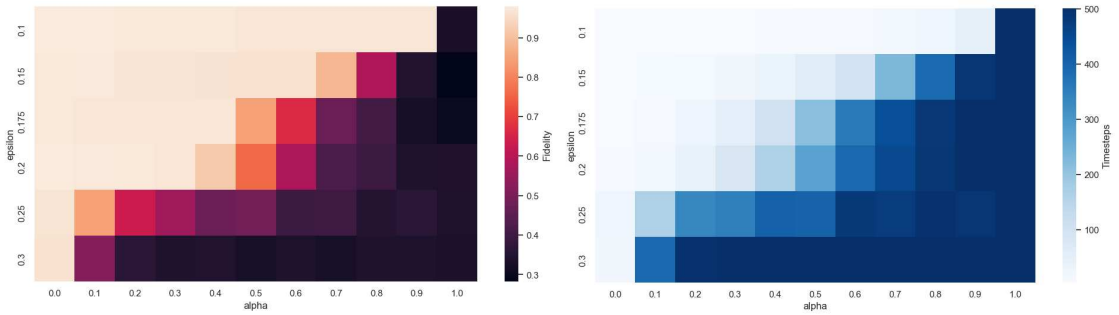
Looking in general (for almost all α) to the plot trends we can notice that to small values of ϵ corresponds a fewer number of timesteps. On the other

hand to high values of ϵ correspond a higher number of timesteps. In particular in the figure (5.15a), where we look to the behaviour of the agent trained with $\epsilon = 0.1$ we see that the distribution of the timesteps is very thin and the number is particularly low for all values of α except that for $\alpha = 1$. Taking a look to figure (5.15c), and comparing this one with the (5.15a) we can clearly see that the behaviour of the agent for high values of α is getting worst, but the variance of the number of timesteps is high. Looking to the case $\epsilon = 0.3$ we can see that the number of timesteps is still high, but in this case the variance is very low and for all the values of α except for $\alpha = 0.1$ the agent converges to 500 timesteps, this means that it is not able to solve the environment at any time.
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



(a) Reward/Fidelity distribution with $\epsilon = 0.1$

(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap

(d) Timetseps Heatmap

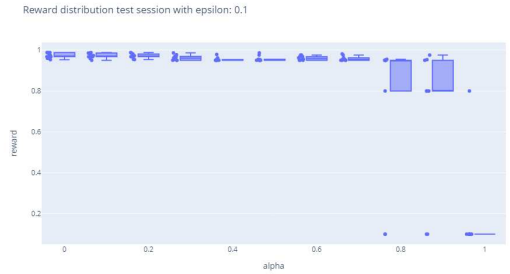
Figure 5.16: Mean fidelity and timesteps plots (MDP)

Looking to the previous plots we can clearly see how the various agents behave with the addition of the noise. In particular the heatmaps seem to be split along the diagonal, indeed in the up-diagonal we find the highest fidelity

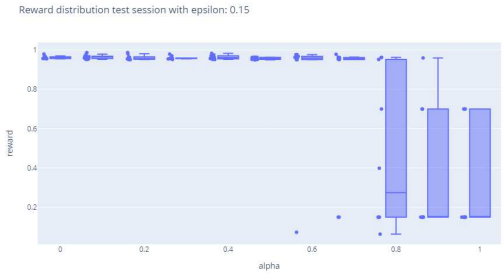
and the lowest number of timesteps. On the other hand in the lowest part of the heatmaps we find poor fidelities with an high number of timesteps. From figures (5.16a, 5.16b) we can clearly see that the agent trained with $\epsilon = 0.1$ is very robust to the noise. This robustness tends to decrease as the ϵ parameter increases its value. We can identify the breaking point in terms of noise robustness in the agent trained with $\epsilon = 0.175$, $\epsilon = 0.2$. In fact these two agents have very similar behaviour and are robust to noise until $\alpha = 0.3$. After these two agents we can clearly see that the performances of the other two are losing in quality in a significant way.

POMDP Setup results

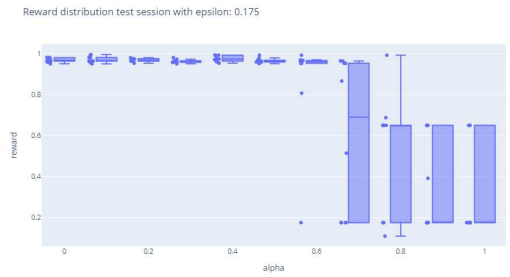
In the following plots (Figure 5.17) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



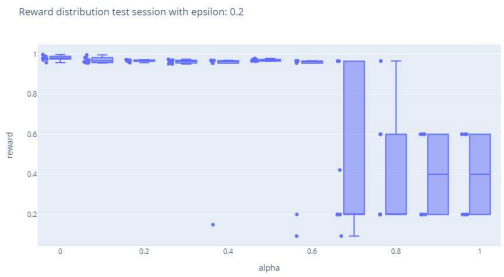
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



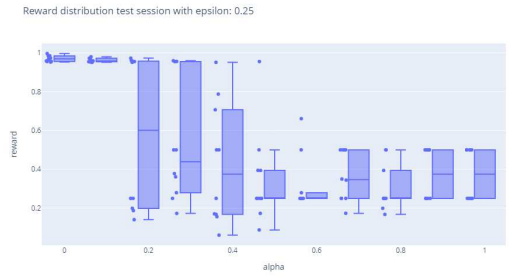
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



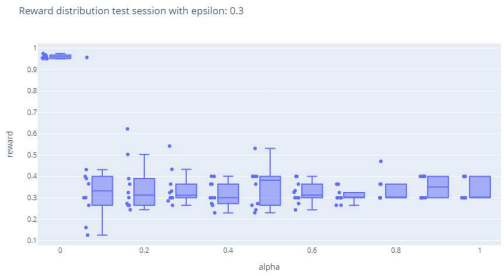
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



(e) Reward/Fidelity distribution with $\epsilon = 0.25$

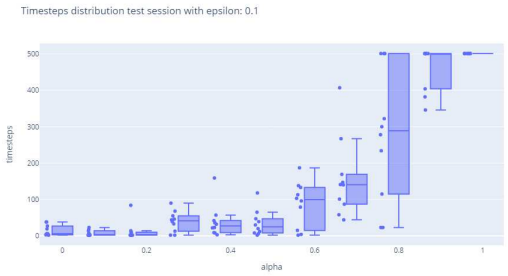


(f) Reward/Fidelity distribution with $\epsilon = 0.3$

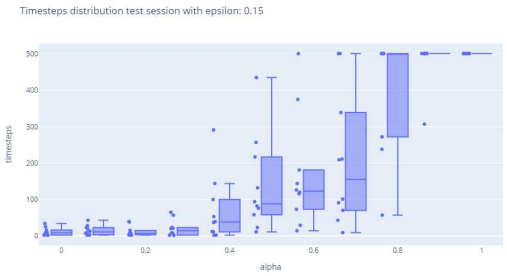
Figure 5.17: Reward/Fidelity distribution of different ϵ over the noise α (POMDP)

Exactly as expected the general trend consists in decreasing the performances by increasing both the ϵ and the α parameters. Indeed we can notice this behaviour by comparing figure (5.17a) with figure (5.17d). Looking to the two plots we can clearly see that for the plot with $\epsilon = 0.1$ the fidelity for the values

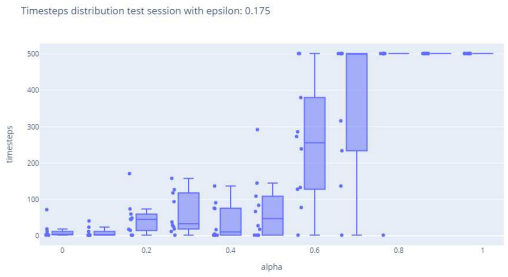
of $\alpha \leq 0.8$ is greater or equal to the target one, but for the other values we can notice that the performances decrease but the fidelity however a part for $\alpha = 1$, the performances are still good. Looking to the one with $\epsilon = 0.2$ we can see that the values of α, for which the fidelity is higher than the fidelity target, are fewer w.r.t. the ones in $\epsilon = 0.1$. Moreover in $\epsilon = 0.2$ we can notice that the distribution of the fidelity for high α is more concentrated away from the target fidelity. This means that increasing the value of ϵ, for high values of α, the probability of having a good fidelity decreases.
In the following plots (Figure 5.18) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the timestep distribution of every trained agent subject to a certain noise α.



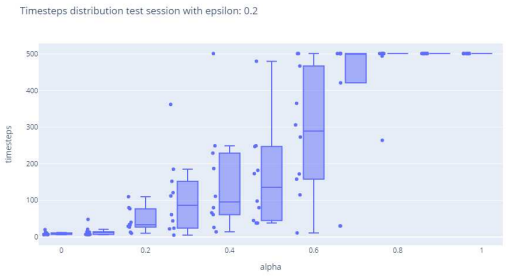
(a) Timesteps distribution with $\epsilon = 0.1$



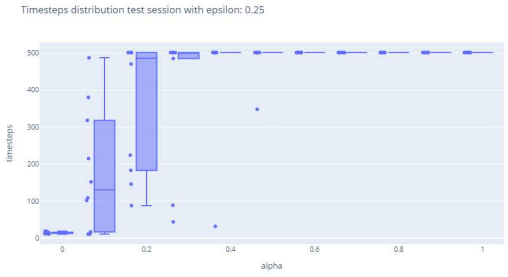
(b) Timesteps distribution with $\epsilon = 0.15$



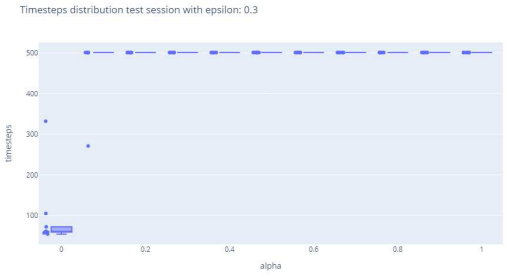
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

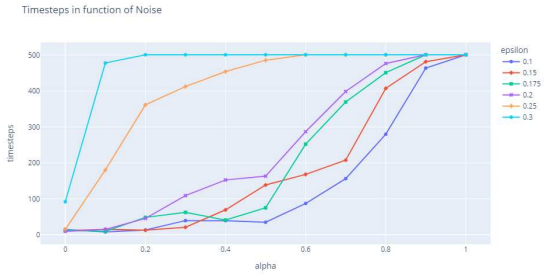


(f) Timesteps distribution with $\epsilon = 0.3$

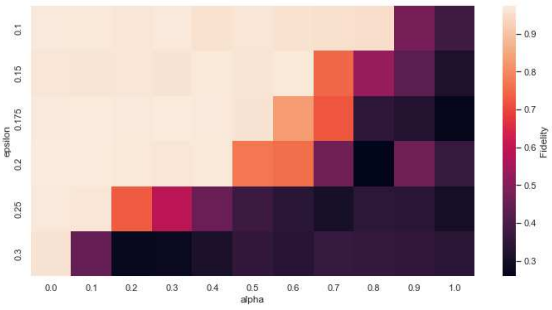
Figure 5.18: Timesteps distribution of different ϵ over the noise α (POMDP)

Looking in general (for almost all α) to the plot trends we can notice that to small values of ϵ corresponds a fewer number of timesteps on the other hand to high values of ϵ correspond a higher number of timesteps. In particular in the figure (5.18a), where we look to the behaviour of the agent trained with $\epsilon = 0.1$ we see that the distribution of the timesteps is very thin and the number is particularly low for all values of $\alpha \leq 0.8$. Taking a look to figure (5.18c), and comparing this one with the (5.15a) we can clearly see that the behaviour of the agent for high values of α is getting worst, but the variance of the number of timesteps is higher. Looking to the case $\epsilon = 0.3$ we can see that the number of timesteps is still high, but in this case the variance is very low and for all the values of α except for $\alpha = 0$ the agent converges to 501 timesteps, this means that it is not able to solve the environment at any time.
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.

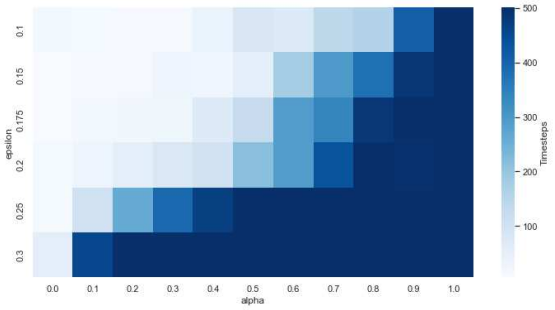
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap



(d) Timetseps Heatmap

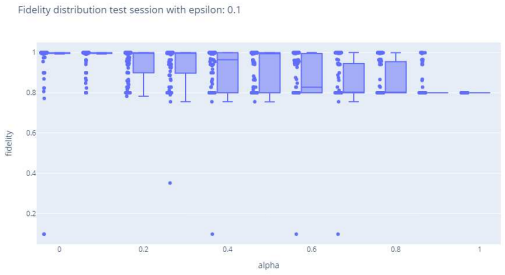
Figure 5.19: Mean fidelity and timesteps plots (POMDP)

Looking to the previous plots we can clearly see how the various agents

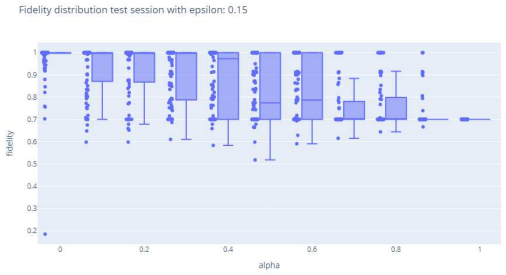
behave with the addition of the noise. In particular the heatmaps seem to be split along the diagonal, indeed in the up-diagonal we find the highest fidelity and the lowest number of timesteps. On the other hand in the lowest part of the heatmaps we find poor fidelities with an high number of timesteps. From figures (5.19a, 5.19b) we can clearly see that the agent trained with $\epsilon = 0.1$ is robust to the noise. This robustness tends to decrease as the ϵ parameter increase its value. We can identify the breaking point in terms of noise robustness in the agent trained with $\epsilon = 0.175$, $\epsilon = 0.2$. In fact these two agents have very similar behaviour and are robust to noise until $\alpha = 0.4$. After these two agents we can clearly see that the performances of the other two are losing in quality in a significant way.

QOMDP SETUP

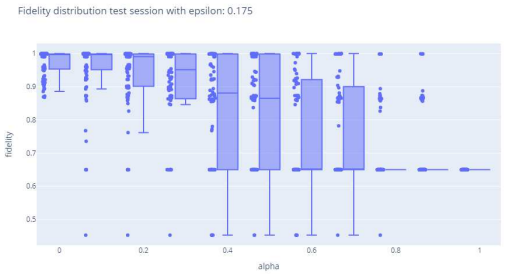
In the following plots (Figure 5.20) we can see the Fidelity distribution of every agent for different (α) noise parameters.



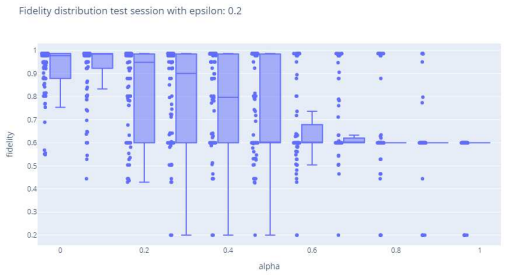
(a) Fidelity distribution with $\epsilon = 0.1$



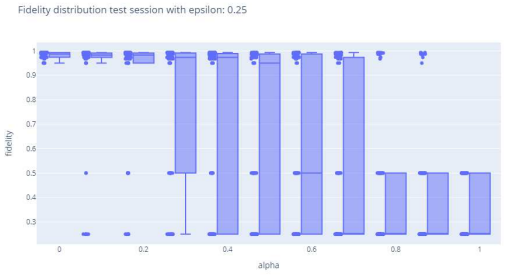
(b) Fidelity distribution with $\epsilon = 0.15$



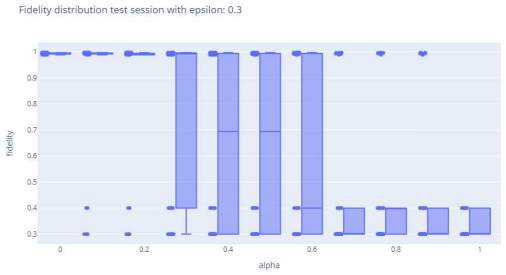
(c) Fidelity distribution with $\epsilon = 0.175$



(d) Fidelity distribution with $\epsilon = 0.2$



(e) Fidelity distribution with $\epsilon = 0.25$

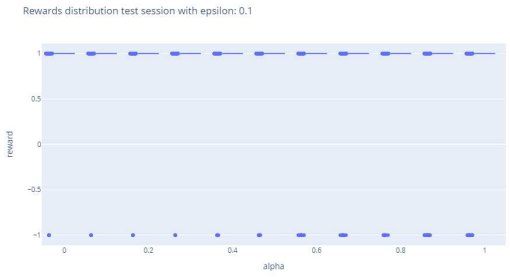


(f) Fidelity distribution with $\epsilon = 0.3$

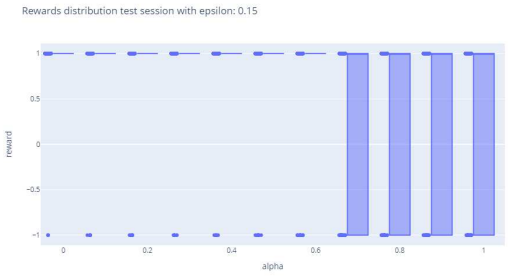
Figure 5.20: Fidelity distribution of different ϵ over the noise α (QOMDP)

From the plots in the previous Figure (5.20) we can see that every agent have a pretty different behaviour when deals with the noise. We start considering the first plot (Figure 5.20a). In this plot we can notice that the for $\alpha \in \{0, 0.1, 0.2, 0.3\}$ the agent can handle the noise, if we increase more the value of alpha the fidelity slightly decreases. This result can be justified by the fact that in spite of the noise the measurement performed with $\epsilon = 0.1$ is still a meaningful information, therefore the agent can handle also the noise. Looking to the Figure (5.20f) we

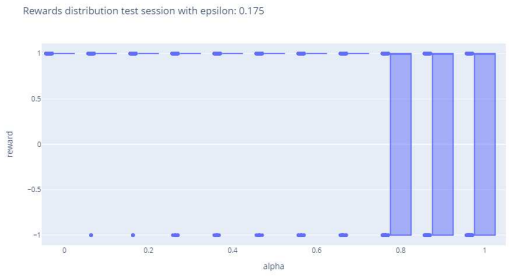
can notice that the a high fidelity is mantained only for $\alpha \in \{0, 0.1, 0.2\}$ but even in this cases we can see some episode to which the agent has achieved a very low fidelity. Moreover if we increase the noise the performances get worst in a very fast way. This is probably due to the fact that the measure is not much informative and with the addition of the noise, the system becomes uncontrollable.
In the following plots (Figure 5.21) we will see the effect of α on the reward distribution for the various agents.



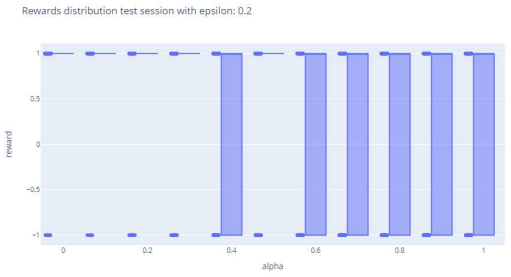
(a) Reward distribution with $\epsilon = 0.1$



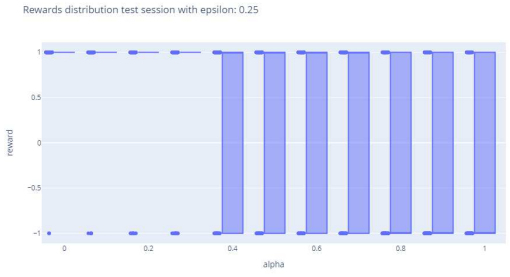
(b) Reward distribution with $\epsilon = 0.15$



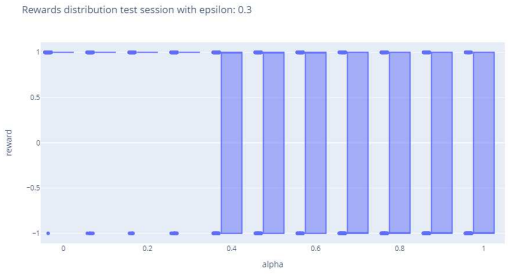
(c) Reward distribution with $\epsilon = 0.175$



(d) Reward distribution with $\epsilon = 0.2$



(e) Reward distribution with $\epsilon = 0.25$



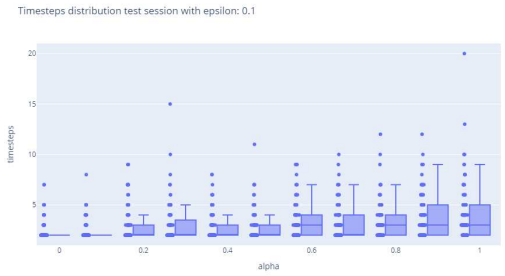
(f) Reward distribution with $\epsilon = 0.3$

Figure 5.21: Reward distribution of different ϵ over the noise α

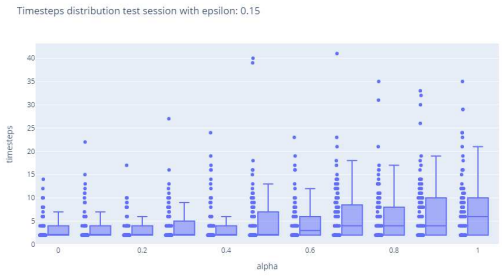
In this plots (Figure 5.21) we can clearly see the joint effect of the noise and the less informative measurements. We start by looking at the plot (Figure 5.21a)

where we can see that we have for all values of α, except for the last one, the reward distribution in +1. Going on, increasing the ϵ value we can see that the values of α for which the reward distribution is in +1, decreases. This is clearly an effect of the impact of the quality of the information carried out by the outcome measurement. Indeed when we deal with more informative measurements we have better results also when we deal with the noise, instead when we have less informative measurements also the noise management gets worst.

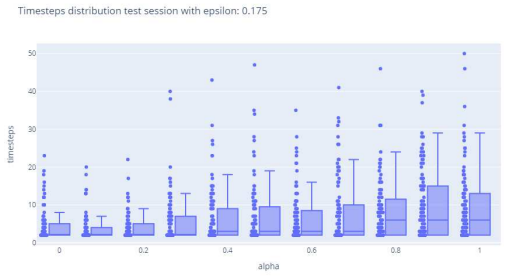
In the following plots (Figure 5.22) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the timesteps distribution of every trained agent subject to a certain noise α.



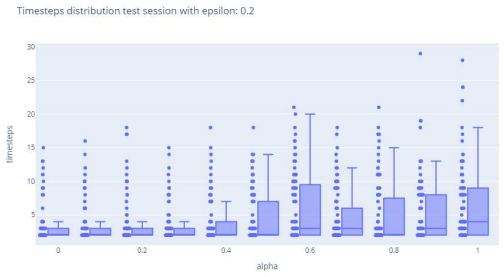
(a) Timesteps distribution with $\epsilon = 0.1$



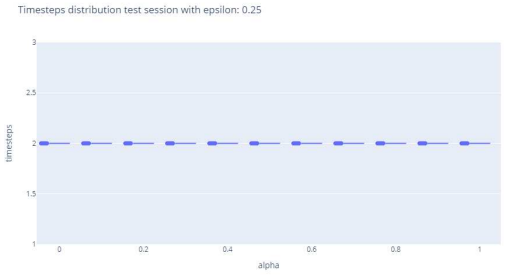
(b) Timesteps distribution with $\epsilon = 0.15$



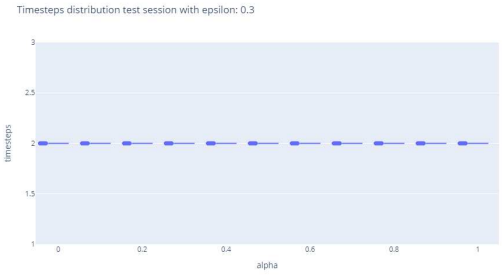
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



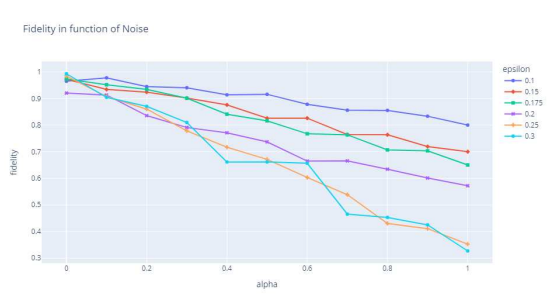
(e) Timesteps distribution with $\epsilon = 0.25$



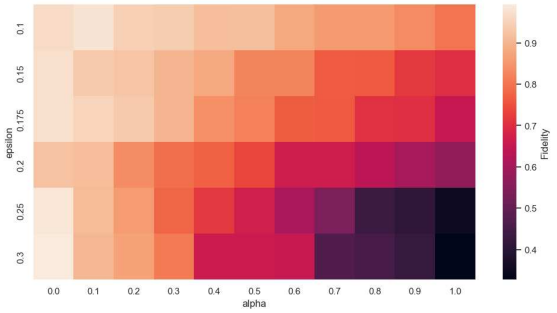
(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.22: Timesteps distribution of different ϵ over the noise α (QOMDP)

Looking in general (for almost all α) to the plot trends we can notice that to small values of ϵ corresponds a fewer number of timesteps on the other hand to high values of ϵ correspond a higher number of timesteps. In particular in the figure (5.22a), where we look to the behaviour of the agent trained with $\epsilon = 0.1$ we see that the distribution of the timesteps is very thin and the number is particularly low for all values of α and starts to grow for $\alpha \in \{0.7, 0.8, 0.9, 1\}$. Taking a look to the other box plots we can confirm this trend of increasing the number of timesteps for higher values of α. For $\epsilon = 0.25$, $\epsilon = 0.3$ we can notice a strange behaviour: the number of timesteps is 2 for every. This is probably due to the fact that the agent is not able to generalize and has developed a deterministic policy.
Last but not least after we are going to analyze the mean values of fidelity, reward and number of timesteps. In particular we will use two different points of view to study these values: plots and heatmaps.

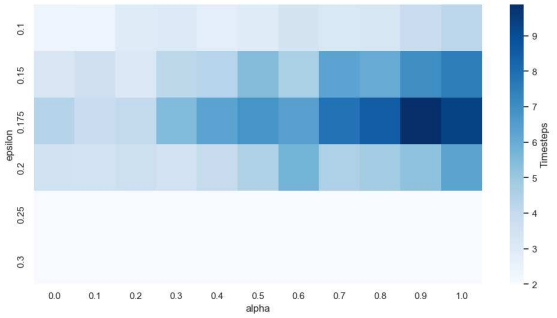


(a) Fidelity w.r.t. noise

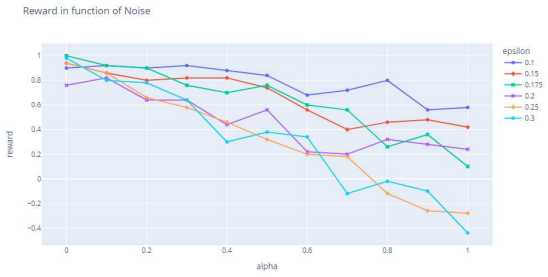


(b) Fidelity Heatmap

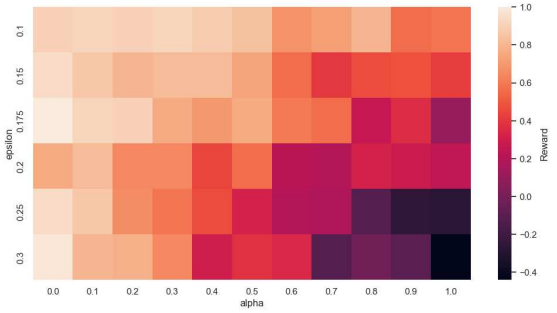
(c) Timestep w.r.t. noise



(d) Timetseps Heatmap



(e) Reward w.r.t. noise



(f) Reward Heatmap

Figure 5.23: Mean fidelity and timesteps plots (QOMDP)

Looking to the previous plots (Figure 5.23) we can clearly see how the various agents behave with the addition of the noise. In particular for the fidelity heatmap (Figure 5.23b) we can localize more light squares on the top left corner, this highlights that for $\epsilon \in \{0.1, 0.15, 0.175\}$ and for values of $\alpha \leq 0.3$ the agents are able to maintain a good fidelity. Going down in the diagonal from the top left corner to the right bottom corner we can see that the fidelity decreases a lot. From figures (5.23a, 5.23e) we can see that the agent trained with $\epsilon = 0.1$ is to one more robust to the noise. This robustness tends to decrease as the ϵ parameter increase its value.Looking to the behaviour of the agents trained with high values of ϵ ($\epsilon = 0.25, 0.3$) we can see that them are not able to generalize.

In the following table we can see compare the different behaviour of the various agents (for a more complete comparison consult B).

epsilon	alpha	MDP		POMDP		QOMDP	
		fidelity	std	fidelity	std	fidelity	std
0.1	0.1	**0.98**	**0.01**	0.97	0.01	0.98	0.05
0.1	0.5	**0.97**	**0.01**	0.96	0.01	0.91	0.09
0.1	1	0.33	0.33	0.39	0.35	**0.8**	**0.01**
0.175	0.1	**0.97**	**0.01**	0.97	0.01	0.92	0.09
0.175	0.5	0.85	0.27	**0.94**	**0.13**	0.81	0.14
0.175	1	0.30	0.21	0.32	0.22	**0.65**	**0.01**
0.3	0.1	0.51	0.27	0.37	0.18	**0.90**	**0.22**
0.3	0.5	0.33	0.05	0.32	0.05	**0.66**	**0.33**
0.3	1	0.33	0.05	0.33	0.05	**0.33**	**0.04**

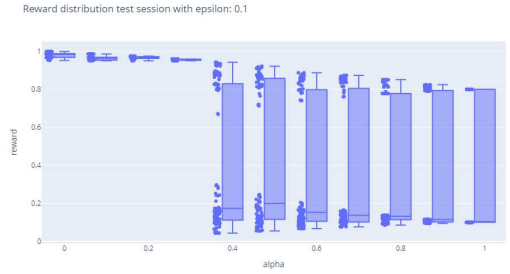
Table 5.14: Classical version of depolarizing noise comparison Table

As we can see from the previous table it seems with the classical version of the depolarizing channel the QOMDP setup performs better only when the noise is very high or we have a big inaccuracy in the measurements. In all the other cases the MDP or the POMDP setup outclass the Model Free one. In the next section we will compare these results with the quantum version of the depolarizing channel in order to see if the behaviour is the same.

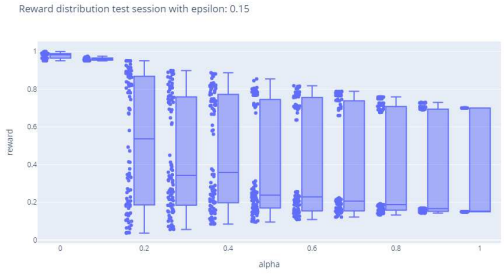
5.4.2 Test with Depolarizing Channel

MDP setup results

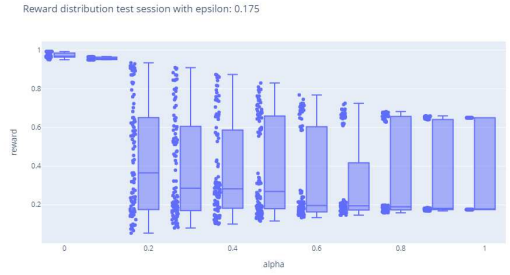
In this first section we will present the results for the MDP setup, as explained in the section 5.3.1, this is the setup where we provide to the maximum amount of information about the system. In the following plots (Figure 5.24) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



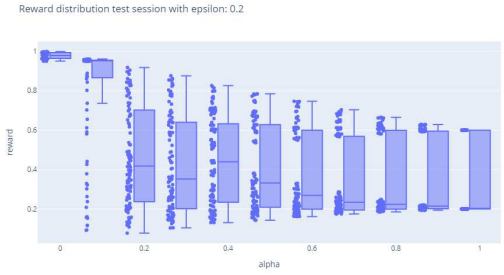
(a) Reward/Fidelity distribution with ϵ = 0.1



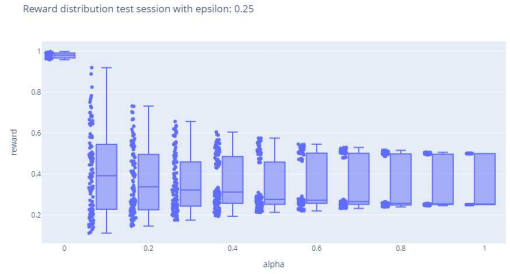
(b) Reward/Fidelity distribution with ϵ = 0.15



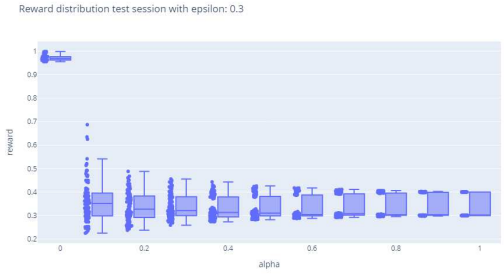
(c) Reward/Fidelity distribution with ϵ = 0.175



(d) Reward/Fidelity distribution with ϵ = 0.2



(e) Reward/Fidelity distribution with ϵ = 0.25



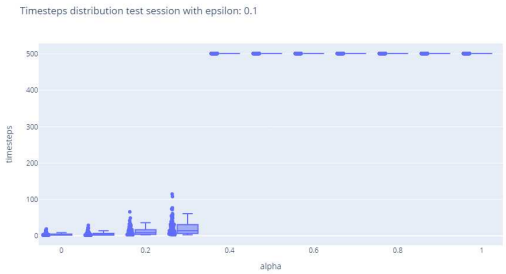
(f) Reward/Fidelity distribution with ϵ = 0.3

Figure 5.24: Reward/Fidelity distribution of different ϵ over the noise α (MDP)

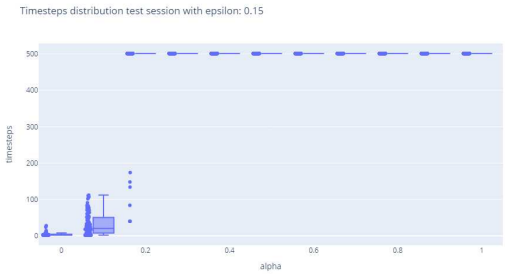
Comparing the previous plots we can see that as long as we increase the

values of ϵ the fidelity get worst and worst. Further starting with small values of ϵ we can see that the fidelity values are higher but as long as we increase α, the fidelity points become more scattered. Increasing the values of ϵ ($\epsilon = 0.25, 0.3$) the fidelity points present very low values with a very thin range of variation. This is an *empirical evidence* of the fact that increasing the values of ϵ and α the problem becomes harder and harder to handle for the agent.

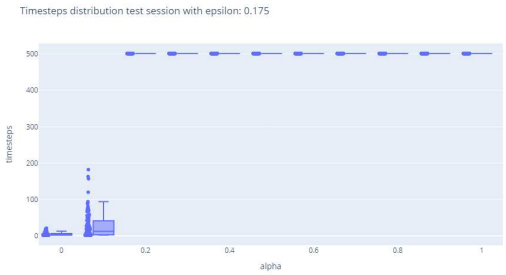
In the following plots (Figure 5.25) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain quantity of noise α.



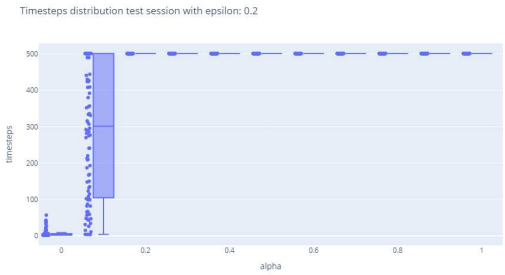
(a) Timesteps distribution with $\epsilon = 0.1$



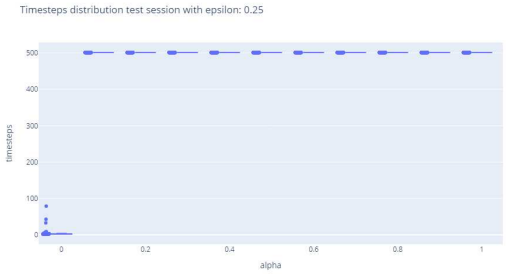
(b) Timesteps distribution with $\epsilon = 0.15$



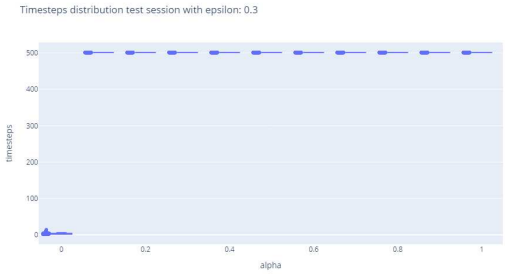
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



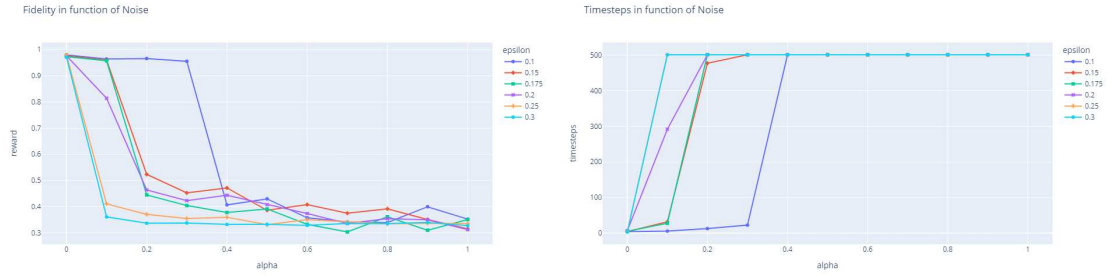
(e) Timesteps distribution with $\epsilon = 0.25$



(f) Timesteps distribution with $\epsilon = 0.3$

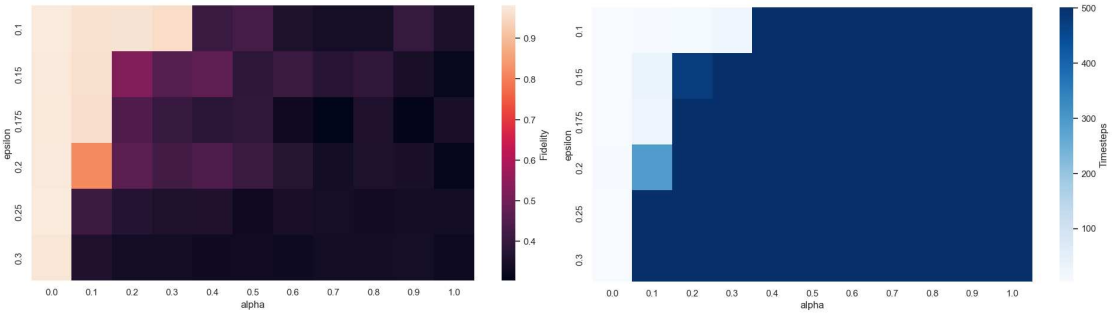
Figure 5.25: Timesteps distribution of different ϵ over the noise α (MDP)

Looking in general to the plot trends we can notice that the value of α with which the agents reach the time-step saturation gets smaller and smaller. In particular we can notice that for $\epsilon = 0.1$ (figure 5.25a) the saturation of the time-steps is reached with $\alpha = 0.4$. In the cases of $\epsilon = 0.15, 0.175$ (figures 5.25a, 5.25c) we can see that the α value with which the agents reach the saturation corresponds to $\alpha = 0.2$. It is possible to see the transition between the agents that are able to control a system with some noise that affects the dynamics and the agent that is not able to control the system with a bit of noise that affects the dynamics in the figure 5.25d, in which we can see that for $\epsilon = 0.1$ the number of time-steps is distributed between 14 and the saturation (500 time-steps). In the other two cases i.e. $\epsilon = 0.25, 0.3$, we can see that the agents do not saturate only for $\alpha = 0$, so only in the environment that is not affected by any noise.
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



(a) Reward/Fidelity distribution with $\epsilon = 0.1$

(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap

(d) Time-steps Heatmap

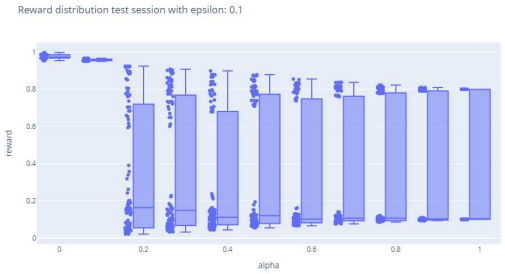
Figure 5.26: Mean fidelity and timesteps plots (MDP)

Looking to the previous plots we can clearly see how the various agents

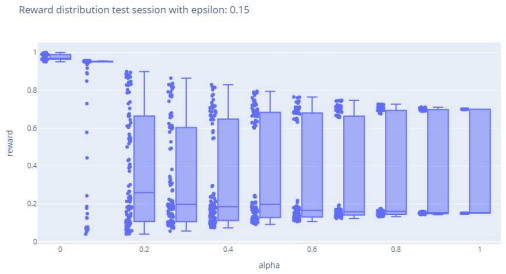
behave with the addition of the noise. We start by analyzing the two heatmaps: figure 5.26c, 5.26d. Comparing these two heatmaps we can see that them are very similar. In particular to a low number of time-steps corresponds higher fidelities, on the other hand to high number of time-steps corresponds lower fidelities. From this we can argue that every time the agents reach the time-steps saturation, they are not more able to control the system.
Looking to figure 5.24 we see what we expect, indeed the agents that work with higher values of ϵ, increasing the amount of noise in the system, behave in a worst way w.r.t. the agents that work with the same amount of noise but with a lower ϵ.
Last but not least we can notice that already with this approach the behaviour of the agents w.r.t. the one in which we have the effect of the classical version of depolarizing channel is different, therefore we will not continue to simulate classical version of quantum noise in order to train the agents.

POMDP Setup results

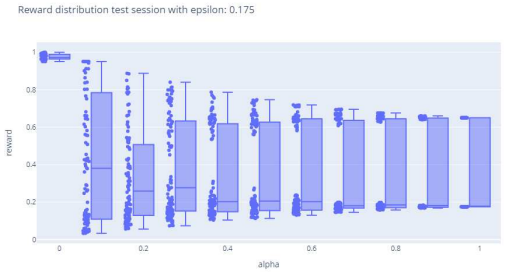
In this section we will present the results of the POMDP setup with the system affected by a depolarizing channel, as explained in the section 5.3.1,in this setup we start reducing the amount of *"quantum information"* that we provide to the agent. In the following plots (Figure 5.27) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



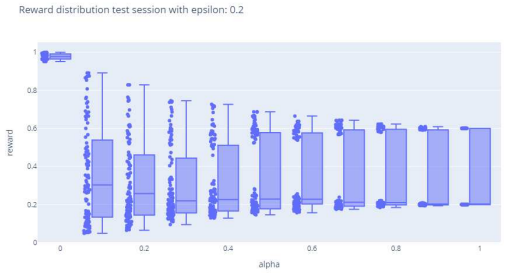
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



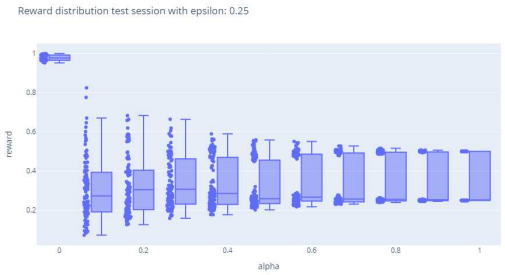
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



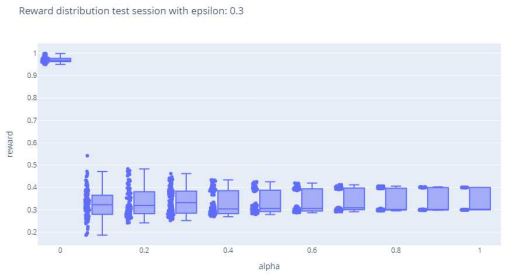
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



(e) Reward/Fidelity distribution with $\epsilon = 0.25$

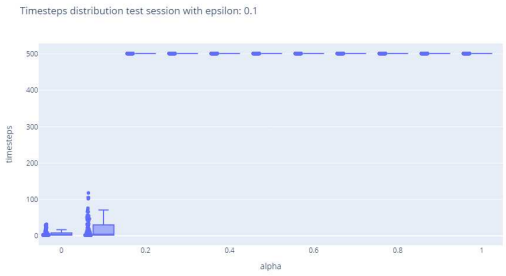


(f) Reward/Fidelity distribution with $\epsilon = 0.3$

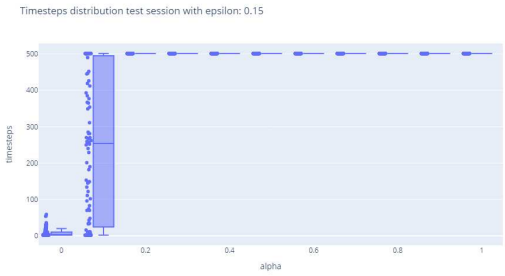
Figure 5.27: Reward/Fidelity distribution of different ϵ over the noise α (POMDP)

Comparing the general trend of the previous plots we can see that the trend

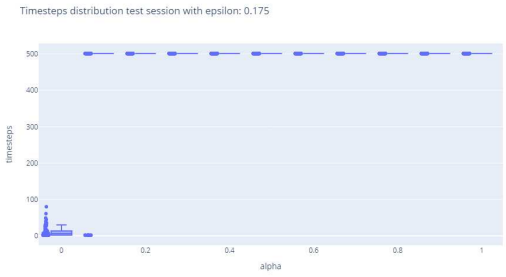
in which the fidelities get worst and worst by increasing ϵ and α is confirmed. In this setup w.r.t. the previous one we can see that the fidelities are more scattered. In particular comparing the plot 5.24a with figure 5.27a it is possible to observe that the behaviour is similar, but in this setup we have good fidelity with an acceptable variance just for $\alpha = 0$ and $\alpha = 0.1$. As expected increasing the value of ϵ we can see also in this setup that the performances of the agent get worse and the range of variation of the fidelity points gets progressively narrower.
In the following plots (Figure 5.28) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain noise α.



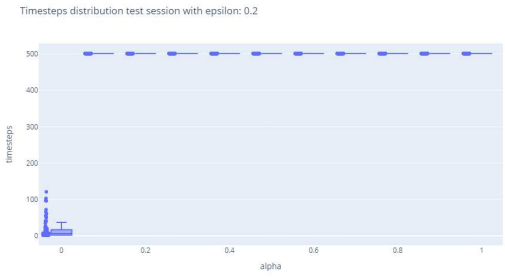
(a) Timesteps distribution with $\epsilon = 0.1$



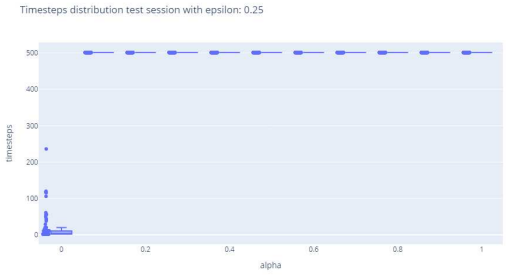
(b) Timesteps distribution with $\epsilon = 0.15$



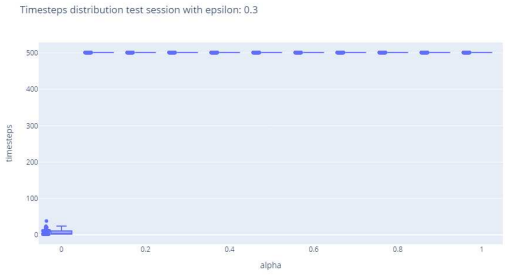
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

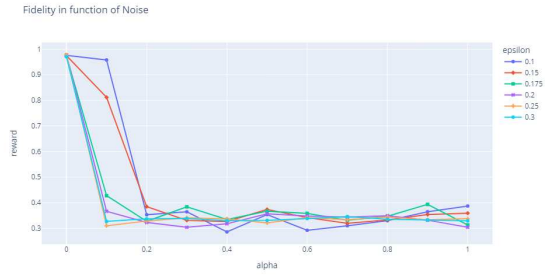


(f) Timesteps distribution with $\epsilon = 0.3$

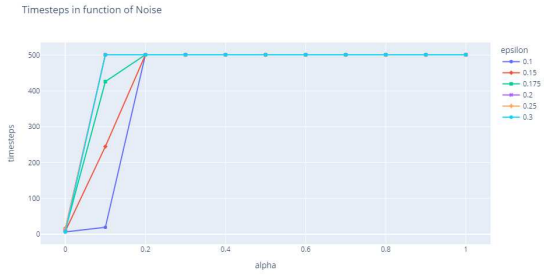
Figure 5.28: Timesteps distribution of different ϵ over the noise α (POMDP)

Looking to the previous plots we can see that these are much like to the ones in the figure 5.25. In particular we see that the behaviour is very similar, indeed the only difference is that the α value for which the time-steps saturate is lower with this setup. We notice that the number of time-steps used by the agent with ϵ parameter equal to $\epsilon = 0.1$ saturates for $\alpha = 0.2$. In this plot we can notice that for both the non-saturated α values and also for the saturated values, we have a very thin variance. In figure 5.28b we find the turnaround point, indeed we can see that for $\alpha = 0.1$ we see that the variance has grown, and the time-step values are distributed from a few time-steps to the saturation value of 500. Concerning all the other "ϵ-agents" we see that they saturate for every α value different from 0, this means that the agents are not able to reach the target fidelity.

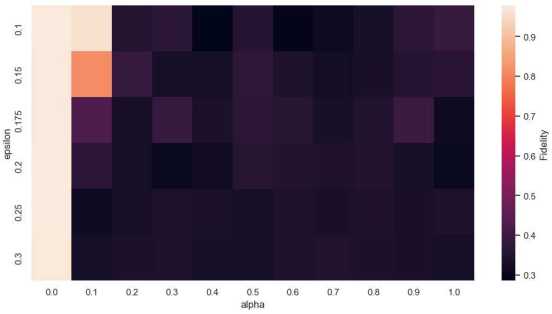
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



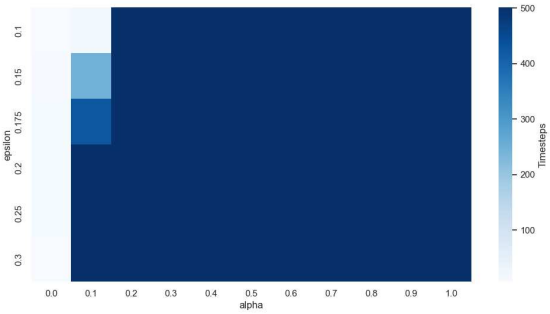
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap



(d) Timetseps Heatmap

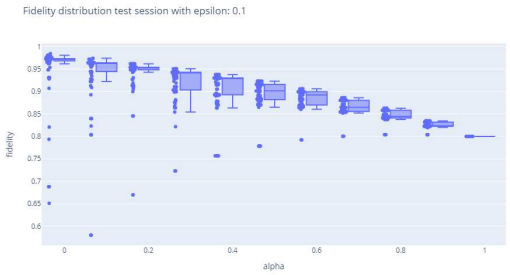
Figure 5.29: Mean fidelity and timesteps plots (POMDP)

We start the analysis focusing on the heatmaps (figure 5.29c, 5.29d). From

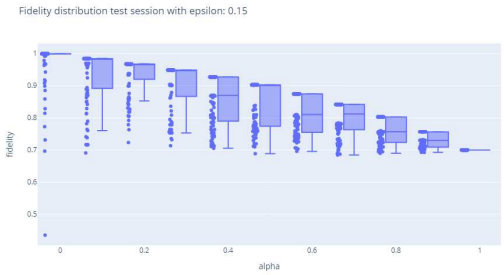
these latter is clear that even in this case there is a correspondence between number of timesteps used and fidelity achieved. In particular we notice that to a high fidelity corresponds a low number of time-steps and vice versa to a low fidelity corresponds a high number of time-steps.
From figure 5.29a we can see that when the system is not affected by the noise every agent is able to achieve a fidelity greater that 95%, instead just the agent with $\epsilon = 0.1$ is able to maintain a good fidelity also for $\alpha = 0.1$. We can clearly see that for $\alpha \geq 0.2$ no agent is able to control the system.

QOMDP SETUP

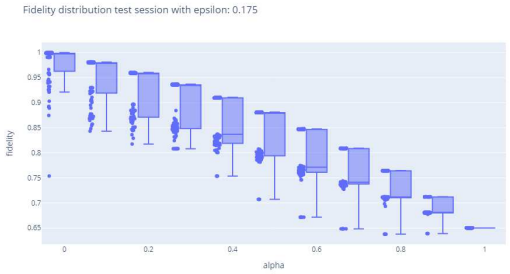
In this section we will present the results of the QOMDP setup with the system affected by a depolarizing channel, as explained in the section 5.3.1, this is the setup in which we built the *Model-Free* framework, indeed we recall that the state of the agent is represented by the previous control action and by the last observation and the reward is computed in function of the last observation. In the following plots (Figure 5.30) we can see the Fidelity distribution of every agent for different (α) noise parameters.



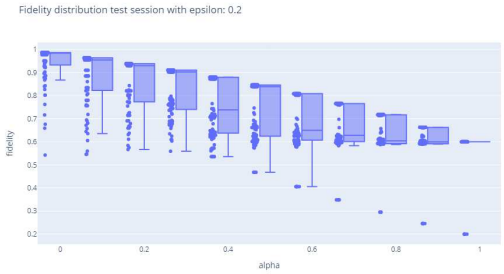
(a) Fidelity distribution with $\epsilon = 0.1$



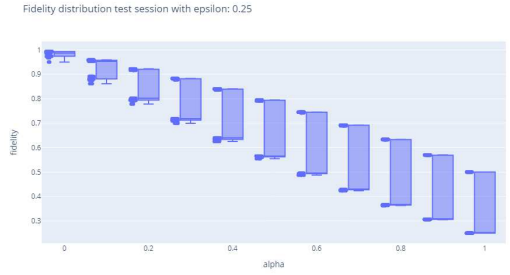
(b) Fidelity distribution with $\epsilon = 0.15$



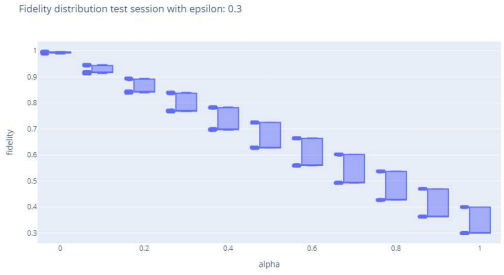
(c) Fidelity distribution with $\epsilon = 0.175$



(d) Fidelity distribution with $\epsilon = 0.2$



(e) Fidelity distribution with $\epsilon = 0.25$



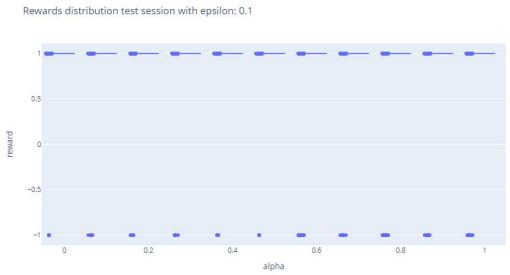
(f) Fidelity distribution with $\epsilon = 0.3$

Figure 5.30: Fidelity distribution of different ϵ over the noise α (QOMDP)

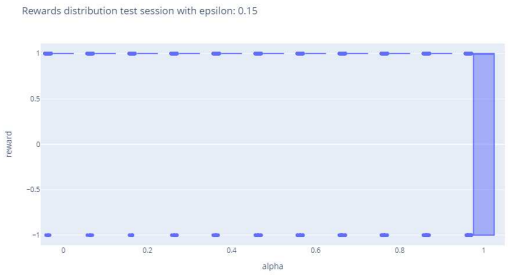
From the plots in the previous Figure (5.30) we can see also in this setup that as long as we increase the ϵ parameter we get worse results as expected,

indeed since the state depends on the measurement outcomes, increasing the inaccuracy on the measure will lead to worse fidelities. Comparing the plots 5.30 with the same plot in the previous setup (figures 5.24, 5.27) we can note that the variance for the higher values of α is lower in this setup, moreover even the fidelity values are higher too. This can be interpreted as an index of higher robustness to the depolarizing noise of this type of setup w.r.t. the previous ones.

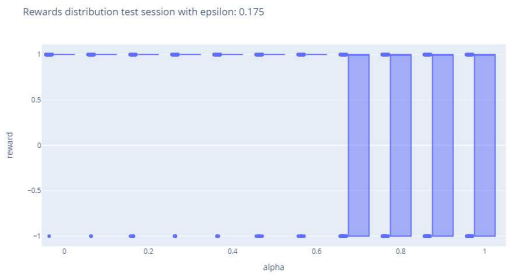
In the following plots (Figure 5.31) we will see the effect of α on the reward distribution for the various agents.



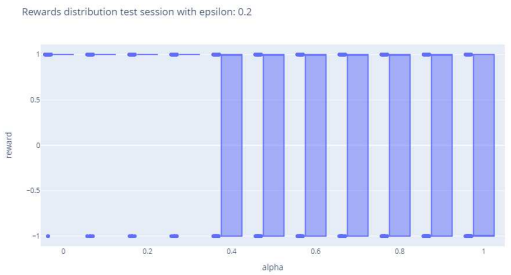
(a) Reward distribution with $\epsilon = 0.1$



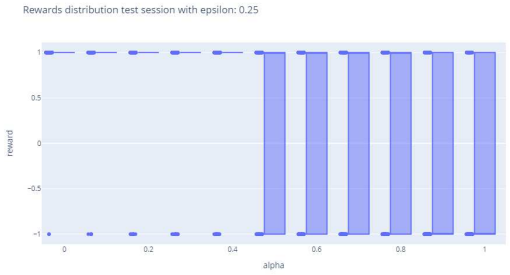
(b) Reward distribution with $\epsilon = 0.15$



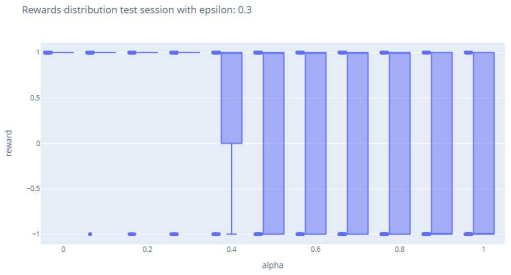
(c) Reward distribution with $\epsilon = 0.175$



(d) Reward distribution with $\epsilon = 0.2$



(e) Reward distribution with $\epsilon = 0.25$



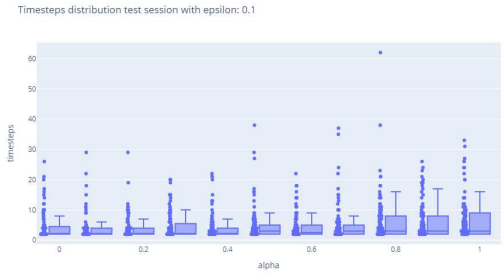
(f) Reward distribution with $\epsilon = 0.3$

Figure 5.31: Reward distribution of different ϵ over the noise α

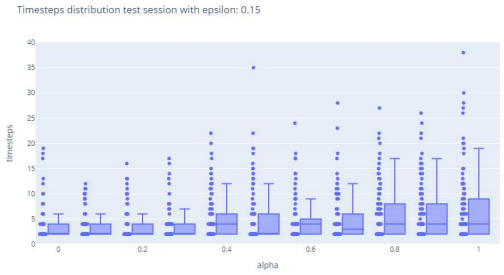
In this plots (Figure 5.31) we can clearly see the joint effect of the noise and

the less informative measurements. We start by looking at the plot (Figure 5.31a) where we can see that we have for all values of α, except for the last one, the reward distribution is +1. Going on, increasing the ϵ value we can see that the values of α for which the reward distribution is in +1, decreases. This is clearly an effect of the impact of the quality of the information carried out by the outcome measurement. Indeed when we deal with more informative measurements we have better results also when we increase the noise, instead when we have less informative measurements also the noise management gets worse.

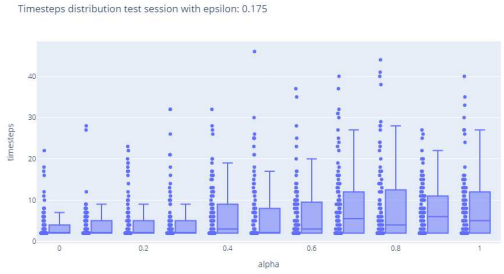
In the following plots (Figure 5.32) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the timesteps distribution of every trained agent subject to a certain noise α. We recall that in this setup it is the agent itself that has to decide when stop the control action.



(a) Timesteps distribution with $\epsilon = 0.1$

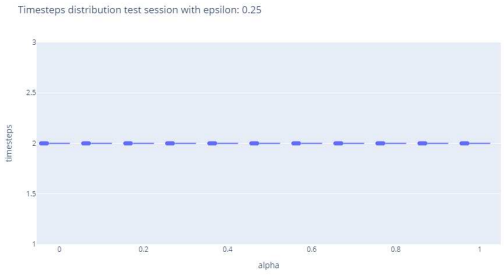


(b) Timesteps distribution with $\epsilon = 0.15$

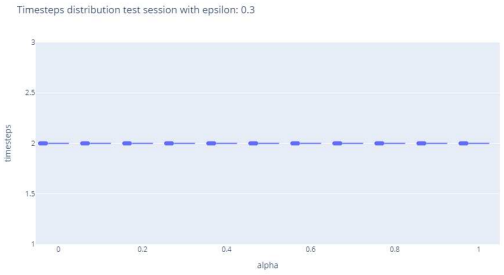


(c) Timesteps distribution with $\epsilon = 0.175$

(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

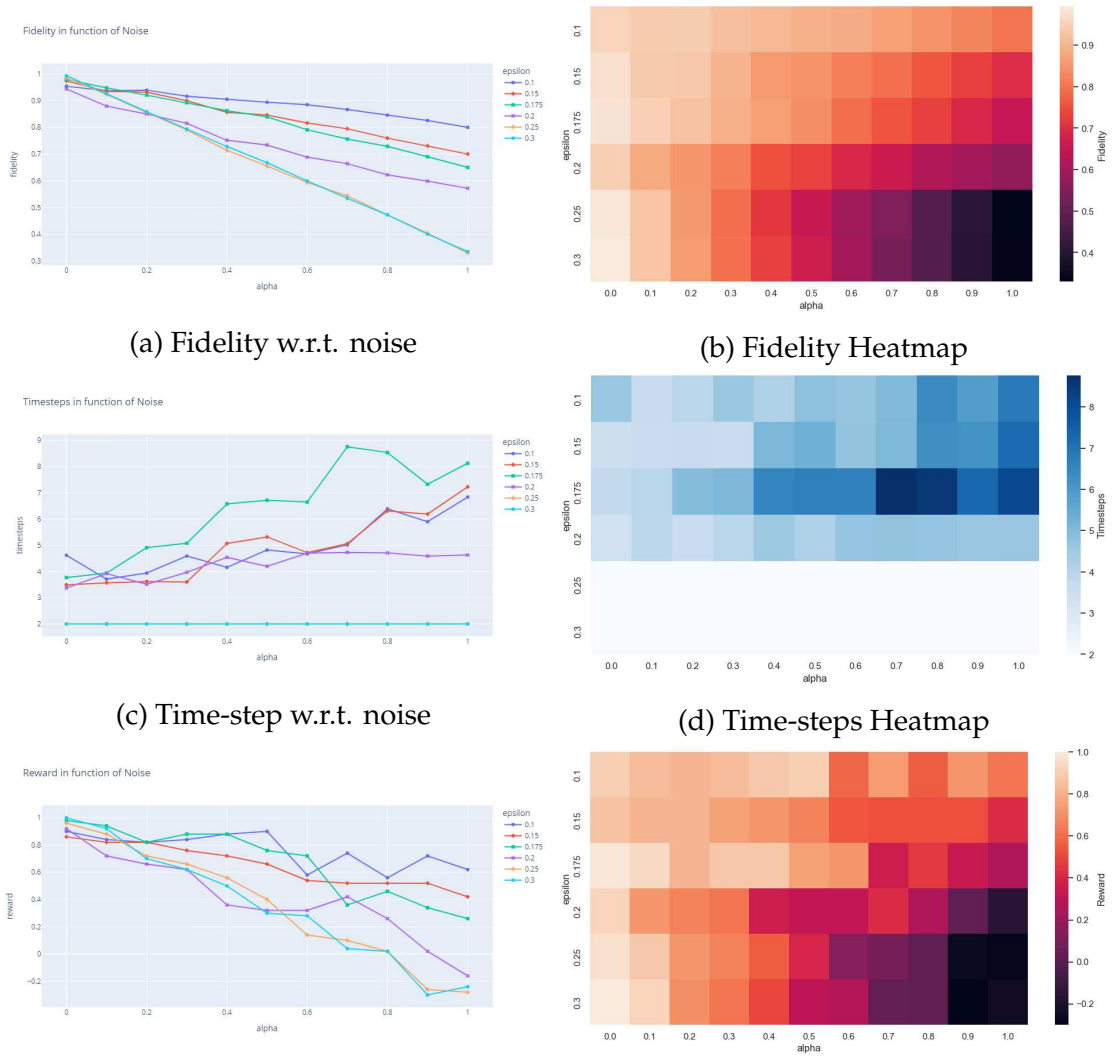


(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.32: Timesteps distribution of different ϵ over the noise α (QOMDP)

Comparing these plots to the ones of the previous setups we can clearly note that the number of control actions performed by this type of agents are very lower w.r.t. the agents derived from the other setups. A key aspect of this figure is that if we look carefully at the plots we can see that the number of the control actions taken as well as its range of variation tend to increase as long as we increase the α parameter (this result is valid for $\alpha \in \{0.1, 0.15, 0.175, 0.2\}$). This can be seen as an index of the fact that the agent recognize that by increasing the α parameter the problem becomes more complex. For the case related to the values of $\alpha = 0.25, 0.3$ we can conclude that during the training the agents were not able to find an adaptive policy and therefore they have chosen a semi-stochastic one with only two time-steps.

Last but not least after we are going to analyze the mean values of fidelity, reward and number of timesteps. In particular we will use two different points of view to study these values: plots and heatmaps.



(a) Fidelity w.r.t. noise

(b) Fidelity Heatmap

(c) Time-step w.r.t. noise

(d) Time-steps Heatmap

(e) Reward w.r.t. noise

(f) Reward Heatmap

Figure 5.33: Mean fidelity and timesteps plots (QOMDP)

We start the analysis by looking to the fidelity plot and heatmap (figures 5.33a, 5.33b). From these plots we can argue that the behaviour of the fidelity seems linear and the slope increases as long as the ϵ parameter increases. Looking to the heatmap it is clear that the worst fidelity is on the bottom right corner of the map instead the best one is on the top left corner, and this result represent our expectation indeed the performances of the agents get worst as long as we increase the quantity of noise in the system and increasing the inaccuracy on the

measurements. In the following table we can find a comparison between some results of this setup and the previous ones.

epsilon	alpha	MDP		POMDP		QOMDP	
		fidelity	std	fidelity	std	fidelity	std
0.1	0.1	**0.96**	**0.01**	0.95	0.05	0.94	0.1
0.1	0.5	0.43	0.35	0.35	0.33	**0.91**	**0.01**
0.1	1	0.35	0.33	0.39	0.34	**0.8**	**1.37E-16**
0.175	0.1	**0.95**	**0.05**	0.42	0.34	**0.95**	**0.05**
0.175	0.5	0.39	0.24	0.37	0.23	**0.84**	**0.04**
0.175	1	0.35	0.22	0.31	0.21	**0.65**	**1.37E-16**
0.3	0.1	0.36	0.08	0.32	0.06	**0.92**	**0.01**
0.3	0.5	0.33	0.05	0.33	0.04	**0.67**	**0.04**
0.3	1	0.32	0.04	0.33	0.04	**0.35**	**0.05**

Table 5.15: Depolarizing channel results comparison

From the table 5.15 we can argue that regarding the depolarizing noise the MDP setup with less noise and for higher values of ϵ behaves better than the other setup, but increasing the value of ϵ and adding some noise to affect the system dynamics, we notice that the best results are always obtained by this last setup.

By this result and looking also to the trend of the comparison plots of the various setups present in the Appendix A we can argue that for this type of noise the QOMDP setup is the more robust setup.

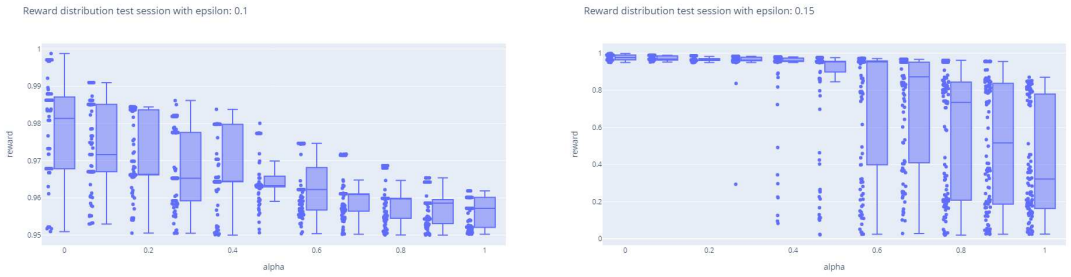
Looking to the time-steps plot we can see that as increasing α the number of timesteps increases too, this behaviour confirms the fact that the agent is aware about the higher complexity of the problem and tries to deal with it by increasing the number of control actions.

Regarding the rewards, we can see as expected that by increasing the noise and the inaccuracy in the measurements the agents obtain always less rewards.

5.4.3 Test with Damping Channel

MDP setup results

In this first section we will present the results for the MDP setup, as explained in the section 5.3.1, this is the setup where we provide to the maximum amount of information about the system.
In the following plots (Figure 5.34) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.

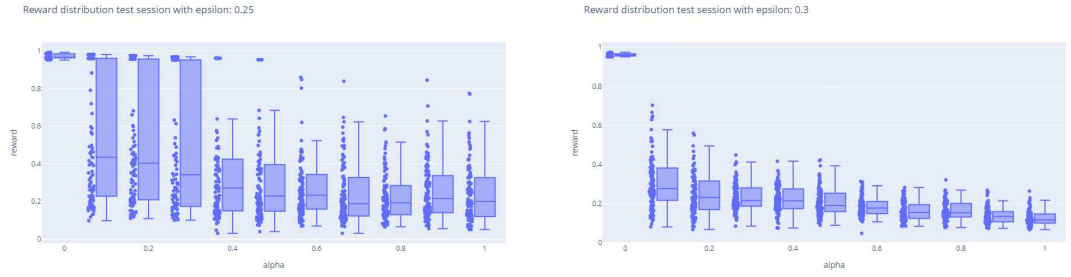


(a) Reward/Fidelity distribution with ϵ = 0.1

(b) Reward/Fidelity distribution with ϵ = 0.15

(c) Reward/Fidelity distribution with ϵ = 0.175

(d) Reward/Fidelity distribution with ϵ = 0.2

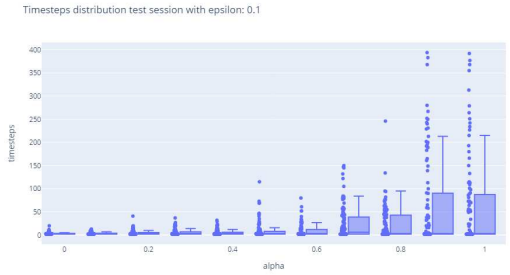


(e) Reward/Fidelity distribution with ϵ = 0.25

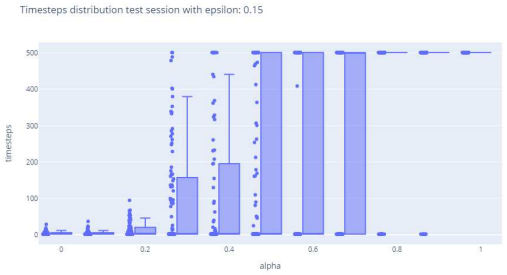
(f) Reward/Fidelity distribution with ϵ = 0.3

Figure 5.34: Reward/Fidelity distribution of different ϵ over the noise α (MDP)

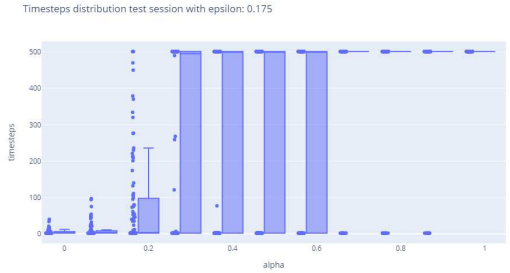
In figure 5.34 we can clearly note that the interplay between the ϵ and the α parameter is key for the control. Indeed as we can see in the plot 5.34a, where we have $\epsilon = 0.1$ even if by increasing the α value the fidelity goes down, the total decrease is still over the target fidelity of 95%. Nevertheless we note that with this type of quantum noise as long as we increase ϵ, the fidelity decreases for smaller and smaller alpha, moreover the values of the fidelity becomes more scattered. In the case of the plots 5.34e, 5.34f we can see index of the fact that the problem becomes very hard to handle since the fidelity is very low, but moreover also the fidelity points are more concentrated, this means that the agents are not more able to achieve the target result (95% fidelity) or some result near the target. In the following plots (Figure 5.35) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain noise α.



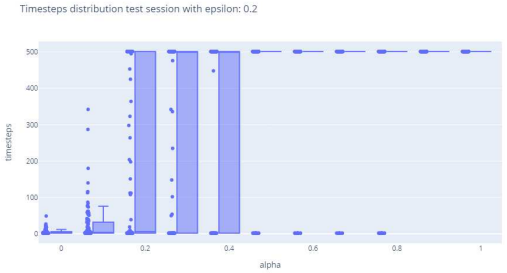
(a) Timesteps distribution with $\epsilon = 0.1$



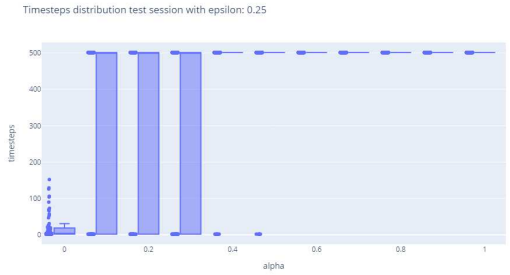
(b) Timesteps distribution with $\epsilon = 0.15$



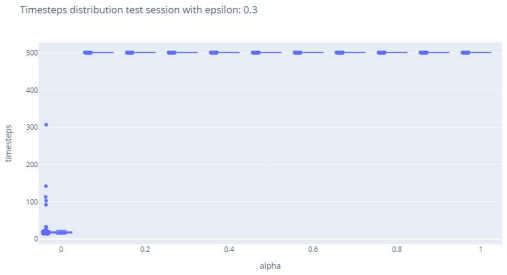
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

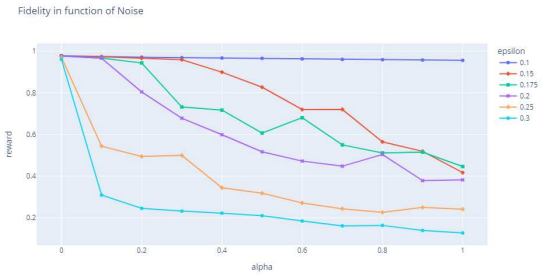


(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.35: Timesteps distribution of different ϵ over the noise α (MDP)

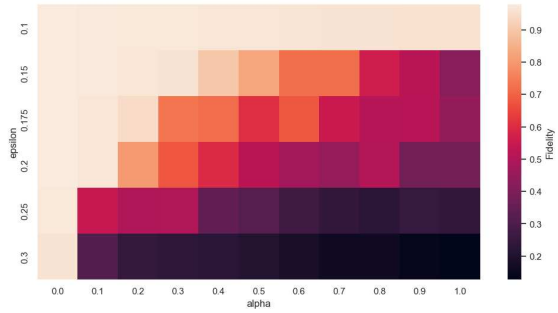
In figure 5.35 we can clearly see the effect of the interaction between the ϵ and α parameters, indeed, as expected, we can notice that increasing the inaccuracy in the measurement and also the quantity of noise, the total number of time-steps to achieve the target results grows. Analyzing the plot we can note that the first big step in which the agents start to perform worse is in the plot 5.35b. In this plot we can see that the agents start saturating from $\alpha = 0.8$, instead in the previous plot (5.35a) we can clearly see that the agent is still able to control the system without saturate the number of available time-steps. From the second plot onwards the agents start to saturate every time earlier until we reach the value of $\epsilon = 0.3$ in which the agent is able to control the system only without noise.

Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.

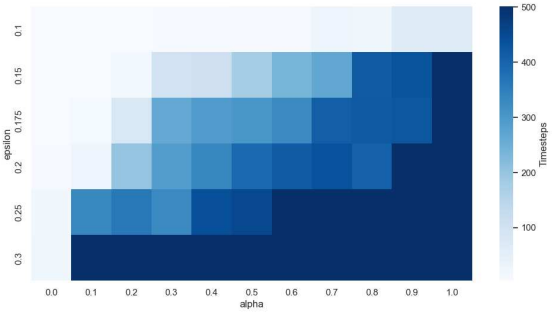


(a) Reward/Fidelity distribution with $\epsilon = 0.1$

(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap



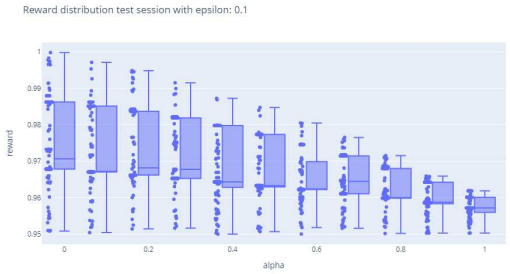
(d) Time-steps Heatmap

Figure 5.36: Mean fidelity and timesteps plots (MDP)

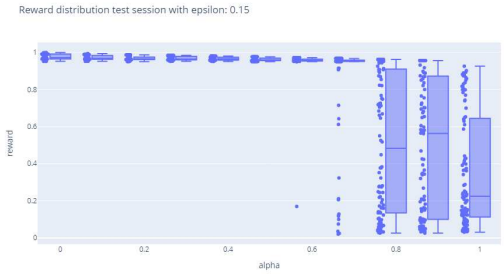
Looking to the previous plots we can clearly see how the various agents behave with the addition of the noise. Starting from the heatmaps (figures 5.36c, 5.36d) we can notice that also in this case there is a relationship between the fidelity and the number of time-steps, indeed in general to a high fidelity corresponds a low number of time-steps. Looking to the fidelity plot (fig 5.36a) we can see as expected that the more robust case is the one with the smaller ϵ, even if with few quantity of noise ($\alpha \leq 0.5$) we have good results also with the agents trained with $\epsilon = 0.15, 0.175$. Regarding the number of timesteps, we can see also in this case that as long as the problem becomes more complicated the number of control action taken in order to achieve the goal increases.

POMDP Setup results

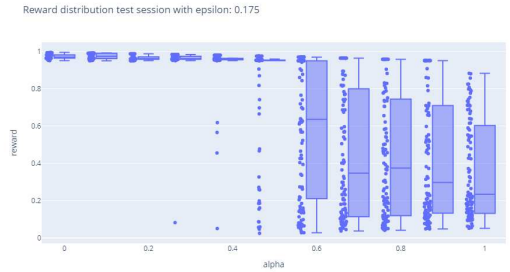
In this section we will present the results of the POMDP setup with the system affected by a damping channel, as explained in the section 5.3.1,in this setup we start reducing the amount of *"quantum information"* that we provide to the agent. In the following plots (Figure 5.37) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



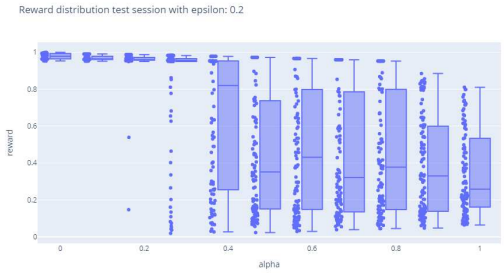
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



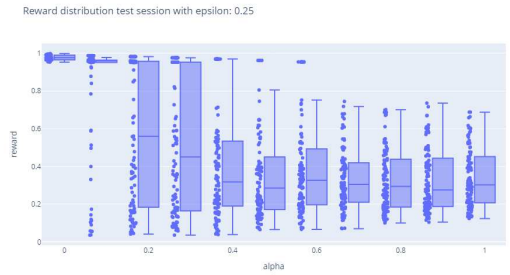
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



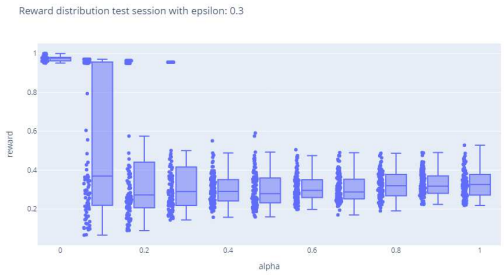
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



(e) Reward/Fidelity distribution with $\epsilon = 0.25$

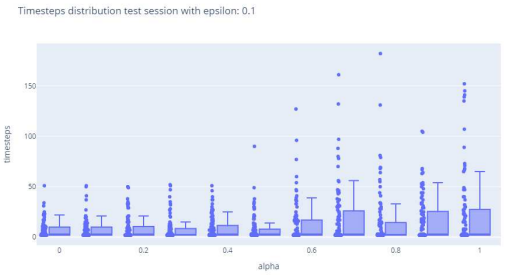


(f) Reward/Fidelity distribution with $\epsilon = 0.3$

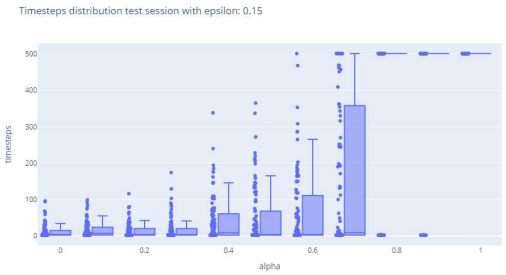
Figure 5.37: Reward/Fidelity distribution of different ϵ over the noise α (POMDP)

We can see in figure 5.34 that the interaction of the ϵ and α parameters is

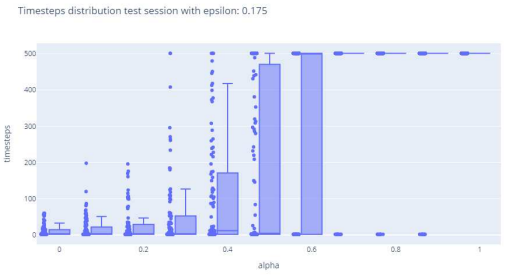
crucial for the control. In fact, as we can see in the figure 5.37a where we have $\epsilon = 0.1$, even if the fidelity decreases as the value of α increases, the overall drop is still greater than the desired fidelity of 95%. Yet, we observe that the fidelity declines for ever-smaller α with this kind of quantum noise as we increase ϵ, and the fidelity values also get more dispersed. Due to an extremely low fidelity and to the concentration of the fidelity points in the plots 5.37e and 5.37f , it is possible to argue that, as expected, with $\epsilon = 0.25, 0.3$ the problem becomes very hard to solve. In the following plots (Figure 5.38) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-steps distribution of every trained agent subject to a certain noise α.



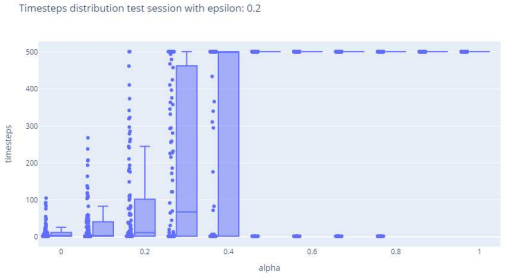
(a) Timesteps distribution with $\epsilon = 0.1$



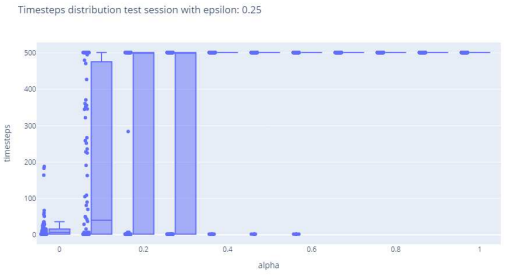
(b) Timesteps distribution with $\epsilon = 0.15$



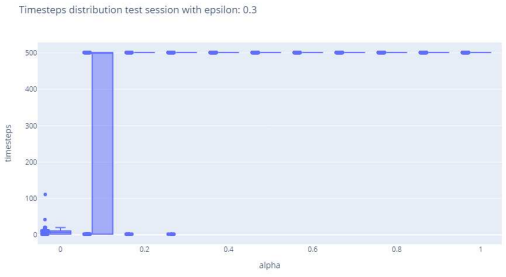
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



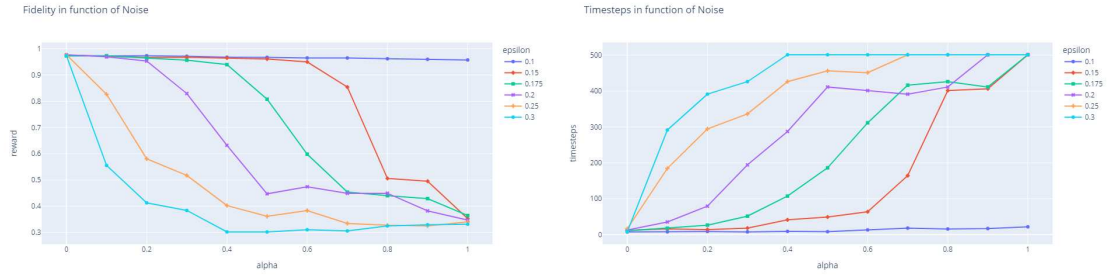
(e) Timesteps distribution with $\epsilon = 0.25$



(f) Timesteps distribution with $\epsilon = 0.3$

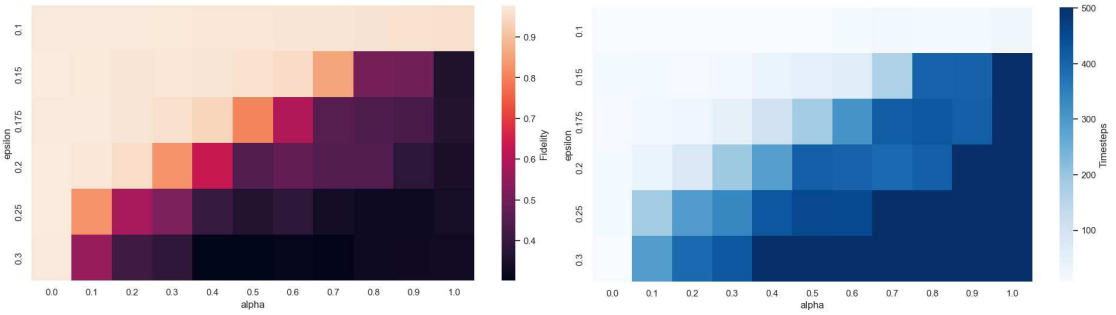
Figure 5.38: Timesteps distribution of different ϵ over the noise α (POMDP)

In figure 5.35 we can clearly see the effect of the interaction between the ϵ and α parameters, indeed, as expected, we can notice that increasing the inaccuracy in the measurement and also the quantity of noise, the total number of time-steps to achieve the target result grows. Analyzing the plot we can note that the first step in which the agents start to perform worse is in the plot 5.35b. Indeed in this plot d we can see that the agents start saturating from $\alpha = 0.7$, instead in the previous plot (5.35a) we can see that the agent is still able to control the system without saturate the number of available time-steps. We recall that in the MDP setup the saturation point was at $\epsilon = 0.15$, $\alpha = 0.8$. From the second plot onwards the agents start to saturate every time earlier until we reach the value of $\epsilon = 0.3$ in which the agent is able to control the system only for $\alpha = 0, 0.1$. In this case we find an improvement w.r.t. the MDP setup in which with $\epsilon = 0.3$ was able to control only the system without noise. Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



(a) Reward/Fidelity distribution with $\epsilon = 0.1$

(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap

(d) Timetseps Heatmap

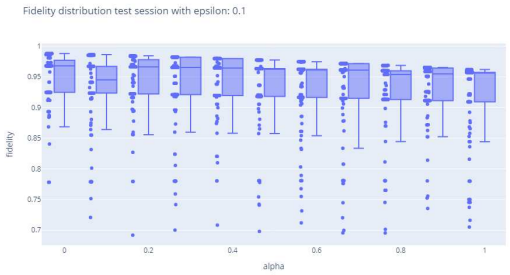
Figure 5.39: Mean fidelity and timesteps plots (POMDP)

Looking to the previous plots we can clearly see how the various agents

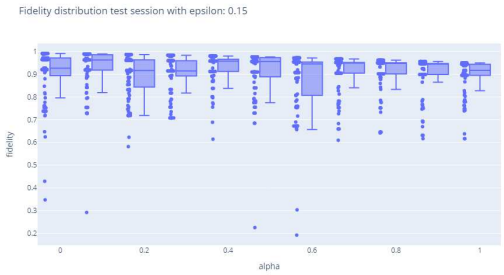
behave with the addition of the noise. In particular the heatmaps seem to be split along the diagonal, indeed in the up-diagonal we find the highest fidelity and the lowest number of timesteps. On the other hand in the lowest part of the heatmaps we find poor fidelities with an high number of timesteps. From figures (5.39a, 5.39b) we can clearly see that the agent trained with $\epsilon = 0.1$ is robust to the noise. This robustness tends to decrease as the ϵ parameter increase its value. In this plot we can find a very similar behaviour w.r.t. the MDP one (figure 5.36a), but in this case comparing the various curves we can note that this setup is more robust to the noise (Appendix A, Damping channel).

QOMDP SETUP

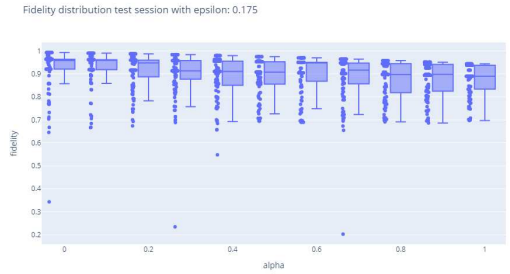
In this section we will present the results of the QOMDP setup with the system affected by a damping channel, as explained in the section 5.3.1, this is the setup in which we built the *Model-Free* framework, indeed we recall that the state of the agent is represented by the previous control action and by the last observation and the reward is computed in function of the last observation. In the following plots (Figure 5.40) we can see the Fidelity distribution of every agent for different (α) noise parameters.



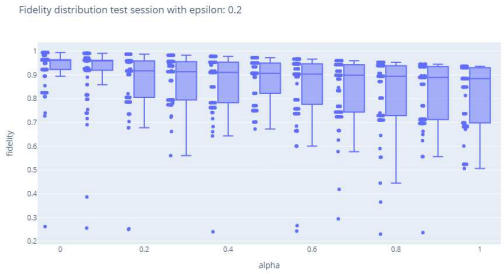
(a) Fidelity distribution with $\epsilon = 0.1$



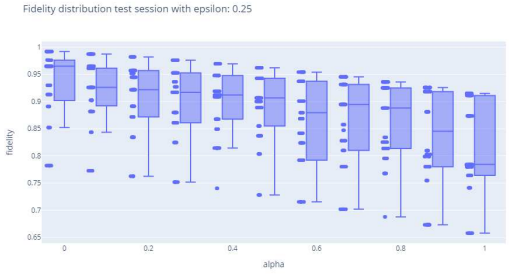
(b) Fidelity distribution with $\epsilon = 0.15$



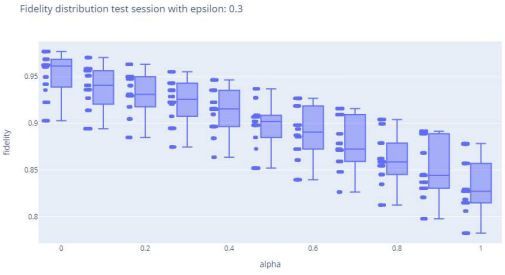
(c) Fidelity distribution with $\epsilon = 0.175$



(d) Fidelity distribution with $\epsilon = 0.2$



(e) Fidelity distribution with $\epsilon = 0.25$

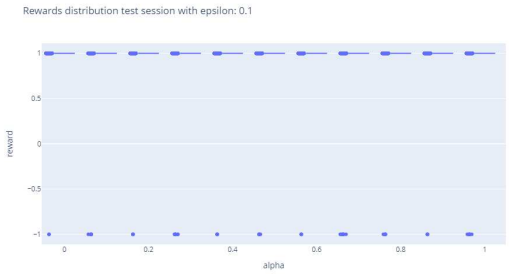


(f) Fidelity distribution with $\epsilon = 0.3$

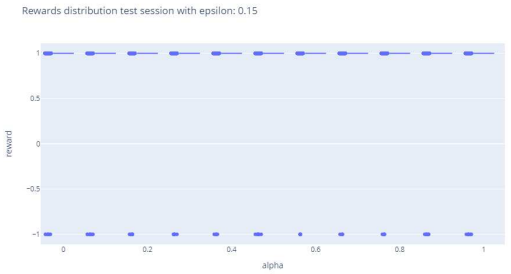
Figure 5.40: Fidelity distribution of different ϵ over the noise α (QOMDP)

From the plots in the previous Figure (5.40) we can see the evolution of the agents' behaviour due to the interaction of different α and ϵ parameters. Firstly

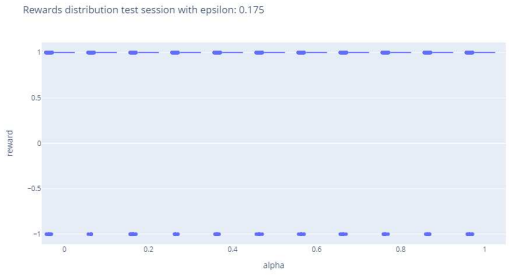
we note that as far as the ϵ parameter increases the fidelity points becomes more distributed and also reach lower fidelities. In particular we can see that in the first plot, the fidelity is greater than 95% and as long as we increase ϵ, the fidelity starts to get slightly worse. Then in the plots 5.40e, 5.40f, we notice that for $\epsilon = 0.25$ and $\epsilon = 0.3$ the fidelity goes down in a faster way than in all the other plots. In the following plots (Figure 5.41) we will see the effect of α on the reward distribution for the various agents.



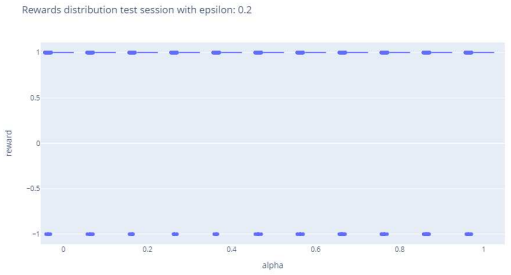
(a) Reward distribution with $\epsilon = 0.1$



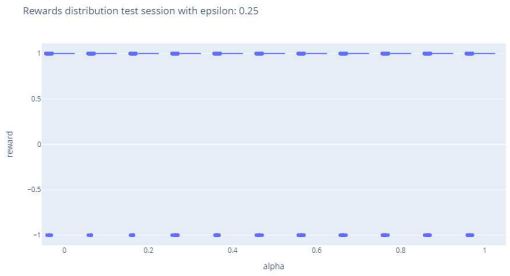
(b) Reward distribution with $\epsilon = 0.15$



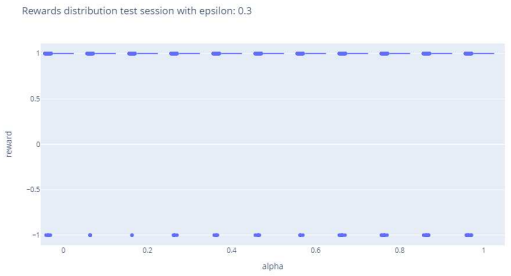
(c) Reward distribution with $\epsilon = 0.175$



(d) Reward distribution with $\epsilon = 0.2$



(e) Reward distribution with $\epsilon = 0.25$



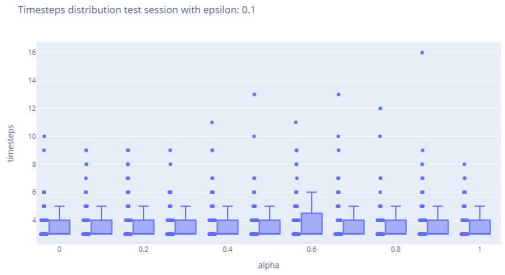
(f) Reward distribution with $\epsilon = 0.3$

Figure 5.41: Reward distribution of different ϵ over the noise α

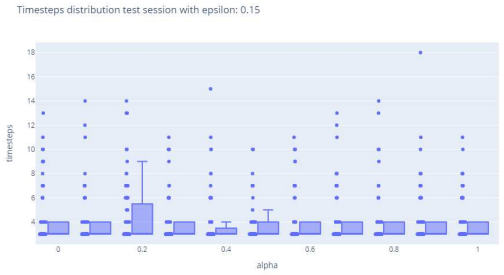
In this plots (Figure 5.41) we can observe the joint effect of the noise and the less informative measurements. In all the plots seems that the increase of the noise does not affect a lot the reward gathered from the agent, indeed as shown

the median is always +1.

In the following plots (Figure 5.42) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the timesteps distribution of every trained agent subject to a certain noise α.

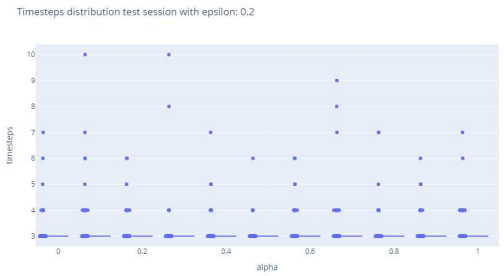


(a) Timesteps distribution with $\epsilon = 0.1$

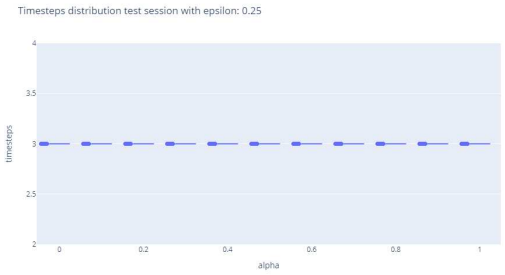


(b) Timesteps distribution with $\epsilon = 0.15$

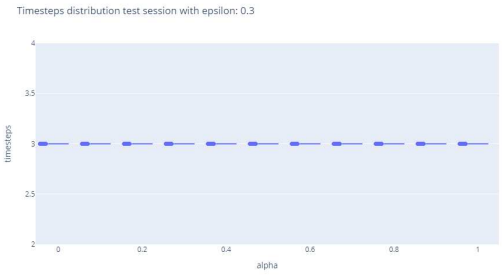
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

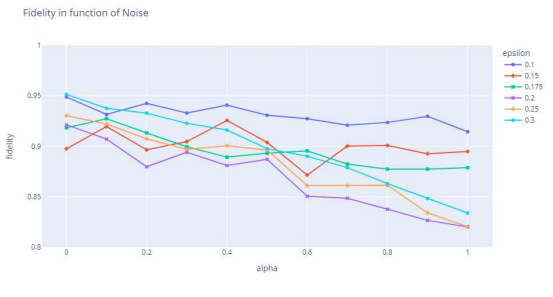


(f) Timesteps distribution with $\epsilon = 0.3$

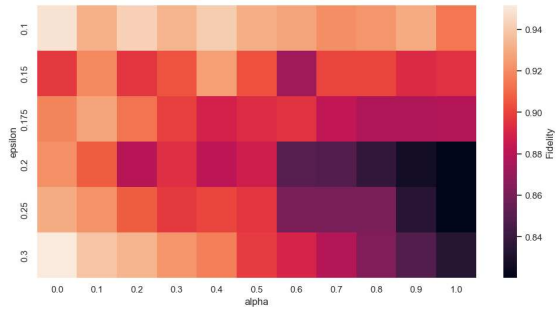
Figure 5.42: Timesteps distribution of different ϵ over the noise α (QOMDP)

Looking to the previous plots we can try to understand the relationship between the number of time-steps and the complexity of the problem. We start by looking at the first two plots (figure 5.42a, 5.42b) in which we find a very similar behaviour with a lot of cases in which the number of time-steps is between 2 and 4. Focusing on the third plot (figure 5.42c) we can observe as the number of time-steps starts to grow, probably since the agent starts to recognize

the increasing of difficulty in reaching the target. In the last three plots we can notice that the number of time-steps is almost fixed to 3. In the last two cases this result is probably due to the training in which the agent probably have converged in a semi-stochastic policy with a fixed number of time-steps. In the case of $\epsilon = 0.2$ we suppose that due to the noise the agent finds the state in a very improbable configuration and tries to tackle this problem with a quasi-fixed number of timesteps.
Last but not least after we are going to analyze the mean values of fidelity, reward and number of timesteps. In particular we will use two different points of view to study these values: plots and heatmaps.

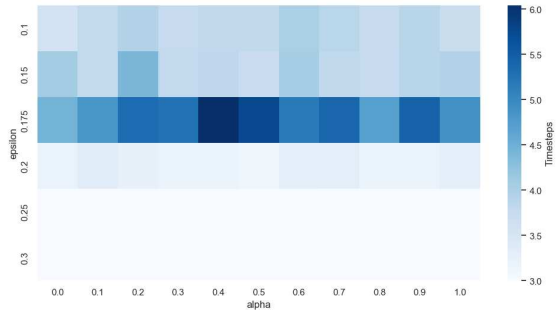


(a) Fidelity w.r.t. noise

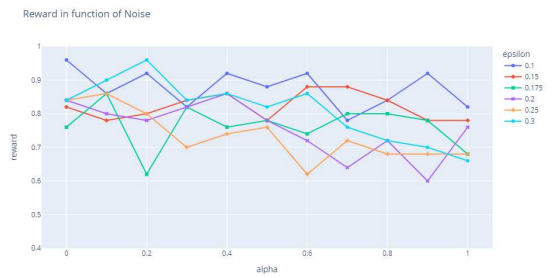


(b) Fidelity Heatmap

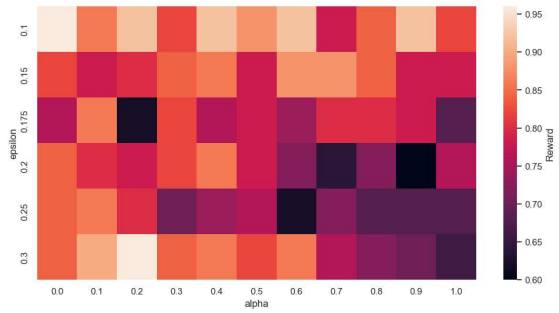
(c) Timestep w.r.t. noise



(d) Timetseps Heatmap



(e) Reward w.r.t. noise



(f) Reward Heatmap

Figure 5.43: Mean fidelity and timesteps plots (QOMDP)

We start the analysis by looking to the fidelity plot and heatmap (figures 5.43a, 5.43b). From the plots we can argue that the behaviour of the fidelity seems linear and the slope increases as long as the ϵ parameter increases. Looking to the heatmap it is clear that the worst fidelity is on the bottom right corner of the map instead the best one is on the top left corner, and this result represent our expectation indeed the performances of the agents get worst as long as we increase the quantity of noise in the system and increasing the inaccuracy on the measurements. In the following table we can find a comparison between some results of this setup and the previous ones.

epsilon	alpha	MDP		POMDP		QOMDP	
		fidelity	std	fidelity	std	fidelity	std
0.1	0.1	**0.98**	**0.01**	0.97	0.01	0.95	0.04
0.1	0.5	**0.97**	**0.01**	**0.97**	**0.01**	0.93	0.06
0.1	1	**0.96**	**0.10**	**0.96**	**0.01**	0.93	0.04
0.175	0.1	**0.97**	**0.01**	**0.97**	**0.01**	0.93	0.07
0.175	0.5	0.60	0.24	0.80	0.30	**0.9**	**0.07**
0.175	1	0.45	0.22	0.36	0.26	**0.88**	**0.08**
0.3	0.1	0.30	0.08	0.56	0.36	**0.93**	**0.02**
0.3	0.5	0.21	0.05	0.30	0.09	**0.90**	**0.03**
0.3	1	0.13	0.04	0.33	0.07	**0.83**	**0.03**

Table 5.16: Damping channel results comparison

From the table 5.16 we can argue that regarding the damping channel noise the MDP and the POMDP setup with less noise and for higher values of ϵ behaves better than the other setup, but increasing the value of ϵ and adding some noise to affect the system dynamics, we notice that the best results are always obtained by this last setup.

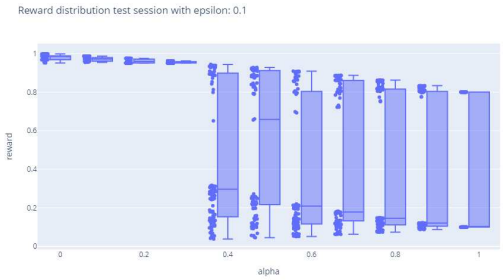
Looking to the time-steps plot we can see that as increasing α the number of timesteps increases too, this behaviour confirms the fact that the agent is aware about the higher complexity of the problem and tries to deal with it by increasing the number of control actions.

By this result and looking also to the trend of the comparison plots of the various setups present in the Appendix A we can argue that for this type of noise and for $\epsilon > 0.1$ the QOMDP setup is more robust. Instead, as shown in the comparison plots in the Appendix A, Damping Channel, we can argue that only for $\epsilon = 0.1$ the MDP and the POMDP setups are more robust even if without a big difference in the fidelity achieved by the QOMDP.

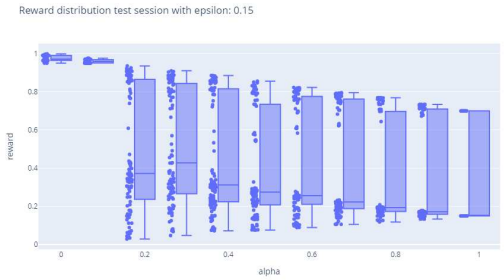
5.4.4 Test with Random Flip Channel

MDP setup results

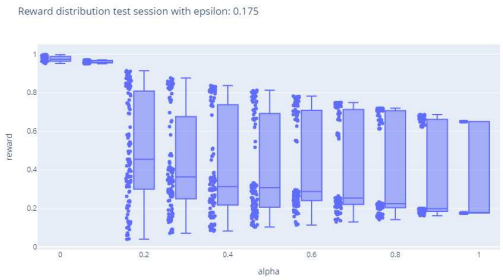
In this first section we will present the results for the MDP setup, as explained in the section 5.3.1, this is the setup where we provide to the maximum amount of information about the system.
In the following plots (Figure 5.44) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



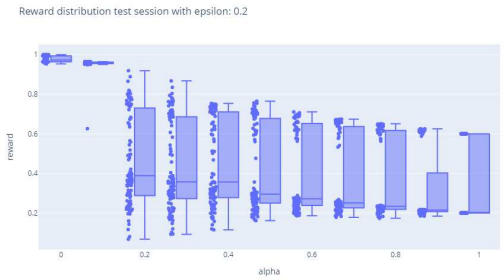
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



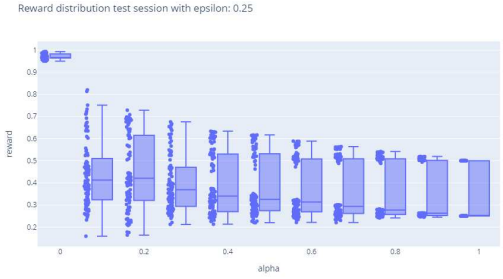
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



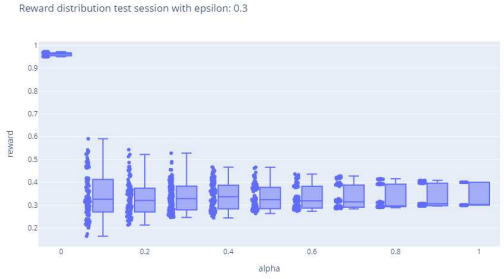
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



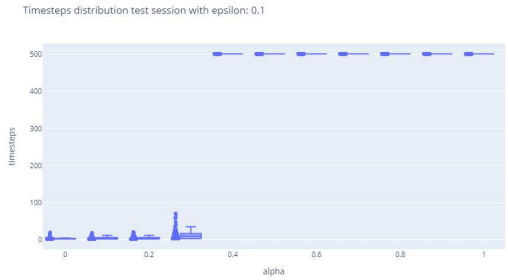
(e) Reward/Fidelity distribution with $\epsilon = 0.25$



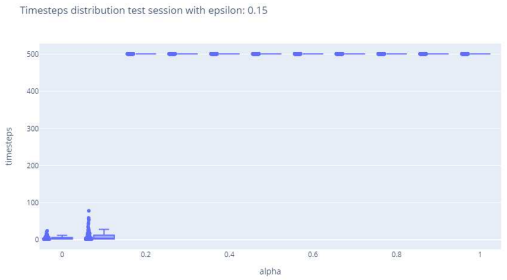
(f) Reward/Fidelity distribution with $\epsilon = 0.3$

Figure 5.44: Reward/Fidelity distribution of different ϵ over the noise α (MDP)

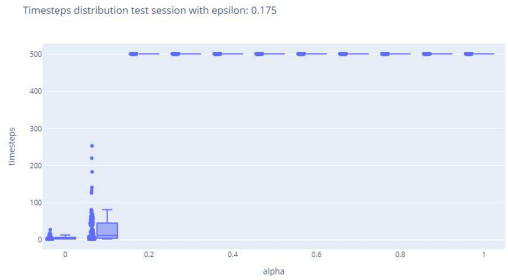
Comparing the previous plots we can see that as long as we increase the values of ϵ the fidelity gets worst and worst. We start the analysis from the first plot (figure 5.44a) in which we can note that the agent is able to achieve a fidelity greater than 95% with a very thin distribution for $\alpha \in 0, 0.1, 0.2, 0.3$. We observe than that increasing the value of α the fidelity points become more scattered, reaching also low fidelities. We note that in the other plots, as long as we increase the ϵ parameter, the fidelity points present a lower value. In the last plot (figures 5.44f) we can see that the fidelity points are stacked around a value of 0.3 with a narrowed range of variation.
In the following plots (Figure 5.45) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain noise α.



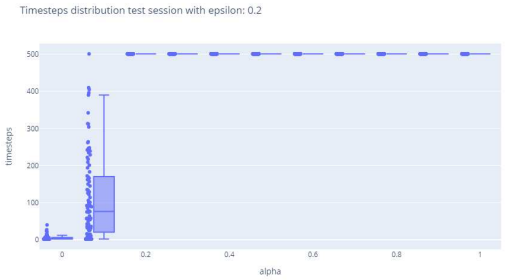
(a) Timesteps distribution with $\epsilon = 0.1$



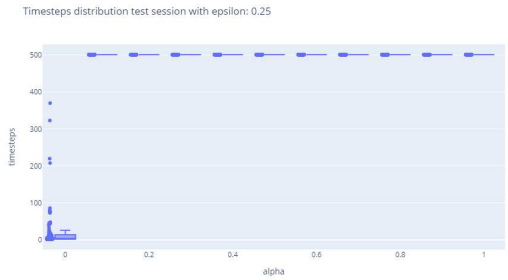
(b) Timesteps distribution with $\epsilon = 0.15$



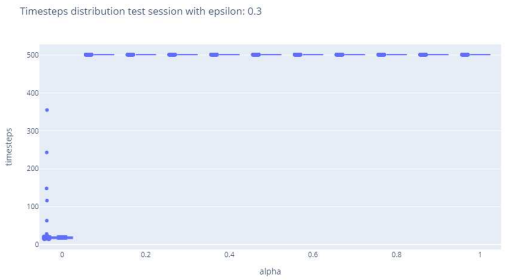
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



(e) Timesteps distribution with $\epsilon = 0.25$

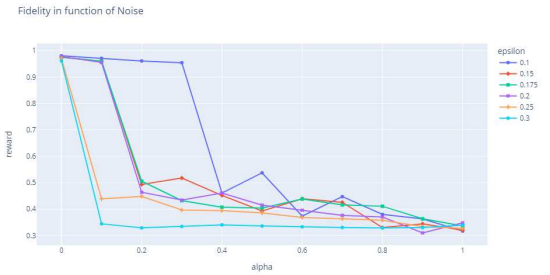


(f) Timesteps distribution with $\epsilon = 0.3$

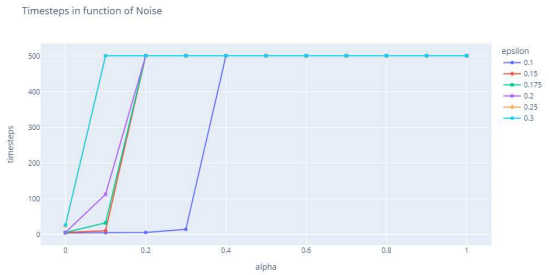
Figure 5.45: Timesteps distribution of different ϵ over the noise α (MDP)

Looking to the general trend of all the plots, as expected, we find that as long as we increase α, for every ϵ, the number of time-steps increases too, as an index of the fact that the problem becomes harder to handle. Starting from the first plot we can see that we have a very low number of time-steps until the agent reaches the value of $\alpha = 0.3$. Then we see that for all the other α the agent saturates the number of available time-steps, this can be interpreted as a sign of the fact that the agent is no more able to reach the target fidelity. As long as we increase the ϵ parameter we note that the value of α for which the agent saturates the number of time-steps becomes smaller and smaller until we reach the cases defined by $\epsilon = 0.25$ and $\epsilon = 0.3$ in which the agents can control the system only without the presence of the noise.

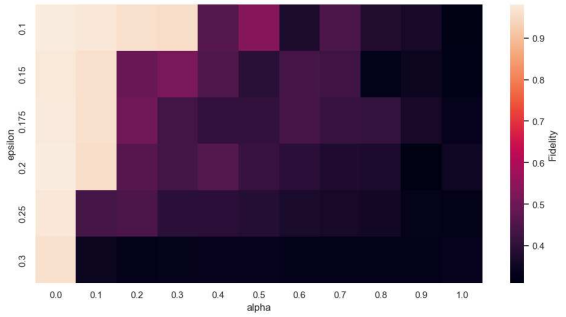
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



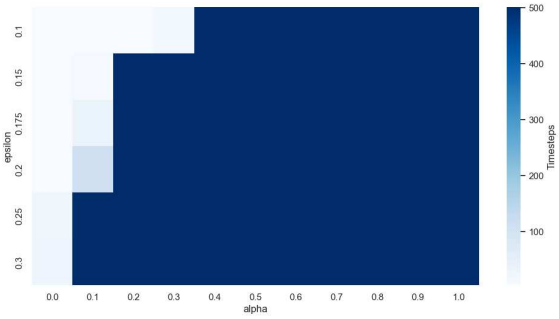
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap



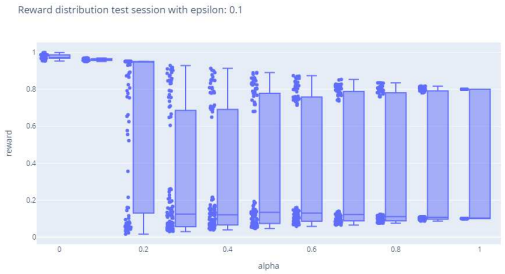
(d) Timesteps Heatmap

Figure 5.46: Mean fidelity and timesteps plots (MDP)

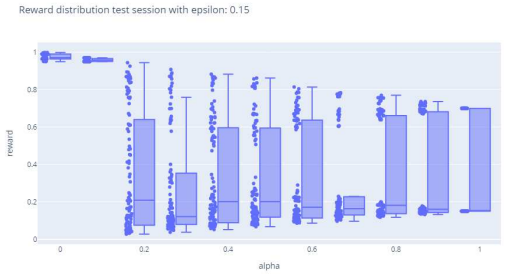
Looking to the previous plots we can clearly see how the various agents behave with the addition of the noise. In particular also in this case we can find in the heatmaps an index of the possible connection that there is between the number of time-steps and the fidelity that the agent is able to achieve. In particular we see that when the agent achieve high fidelities it never saturates the available number of time-steps. From figures (5.46a, 5.46b) we can clearly see that the agent trained with $\epsilon = 0.1$ is more robust to the noise w.r.t. the others, but also that agent it is not able to maintain a good performance for all the values of α.

POMDP Setup results

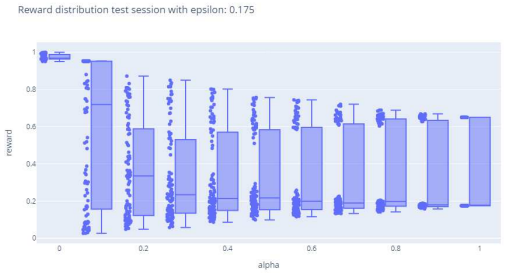
In this section we will present the results of the POMDP setup with the system affected by a random flip channel noise, as explained in the section 5.3.1,in this setup we start reducing the amount of *"quantum information"* that we provide to the agent. In the following plots (Figure 5.47) we can see the reward/fidelity distribution of every agent for different (α) noise parameters.



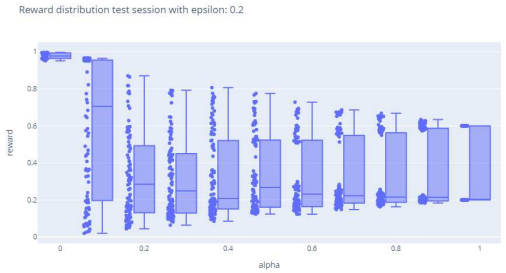
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



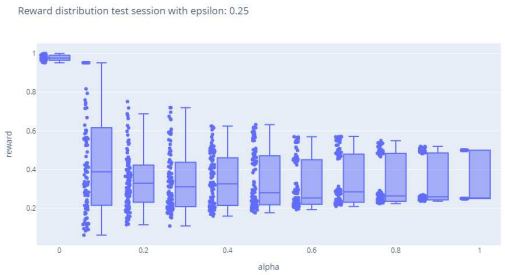
(b) Reward/Fidelity distribution with $\epsilon = 0.15$



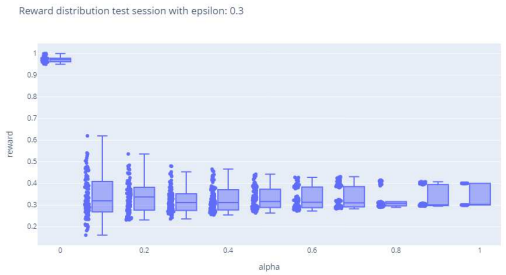
(c) Reward/Fidelity distribution with $\epsilon = 0.175$



(d) Reward/Fidelity distribution with $\epsilon = 0.2$



(e) Reward/Fidelity distribution with $\epsilon = 0.25$

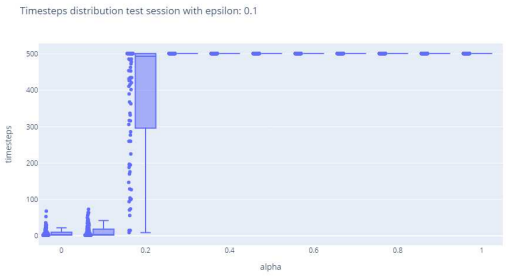


(f) Reward/Fidelity distribution with $\epsilon = 0.3$

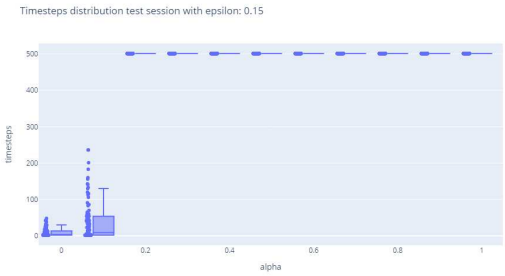
Figure 5.47: Reward/Fidelity distribution of different ϵ over the noise α (POMDP)

Exactly as expected, the general trend consists in decreasing the perfor-

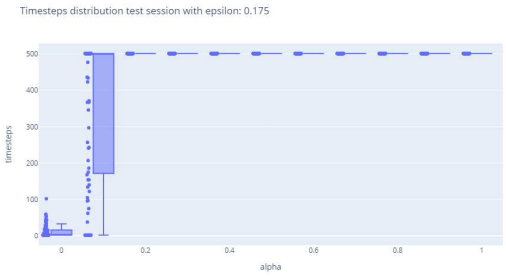
mances by increasing both the ϵ and the α parameters. Starting from figure 5.37a we can notice that the fidelity points present a high value and are close only for $\alpha \leq 0.1$. For all the other values of α the results present a points that becomes more scattered. Looking to the others ϵ-plots we can notice that the fidelity goes down and also the points become more concentrated. With $\epsilon = 0.3$ in figure 5.47f we can see that the range of variation for all the $\alpha \neq 0$ is very thin moreover the fidelity points present a very low fidelity. In the following plots (Figure 5.48) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the time-step distribution of every trained agent subject to a certain noise α.



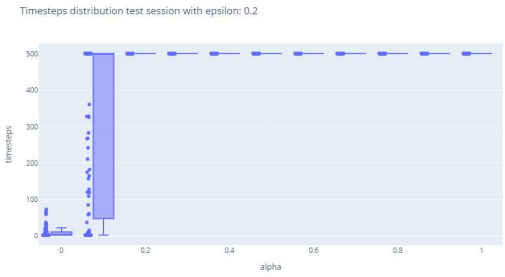
(a) Timesteps distribution with $\epsilon = 0.1$



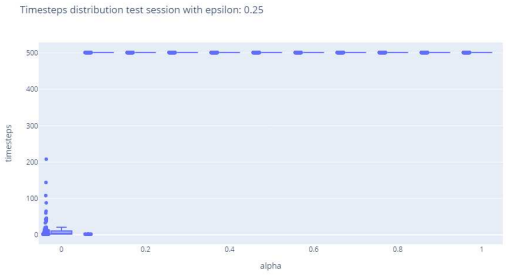
(b) Timesteps distribution with $\epsilon = 0.15$



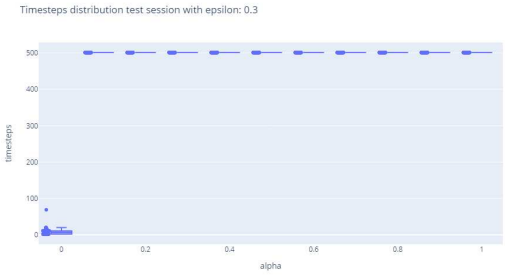
(c) Timesteps distribution with $\epsilon = 0.175$



(d) Timesteps distribution with $\epsilon = 0.2$



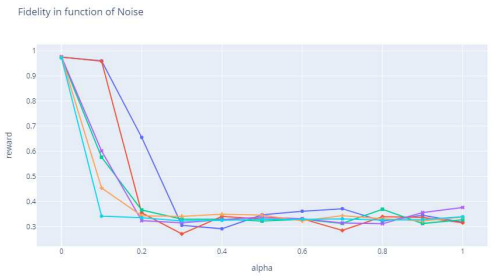
(e) Timesteps distribution with $\epsilon = 0.25$



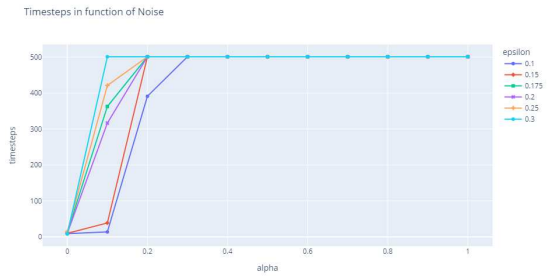
(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.48: Timesteps distribution of different ϵ over the noise α (POMDP)

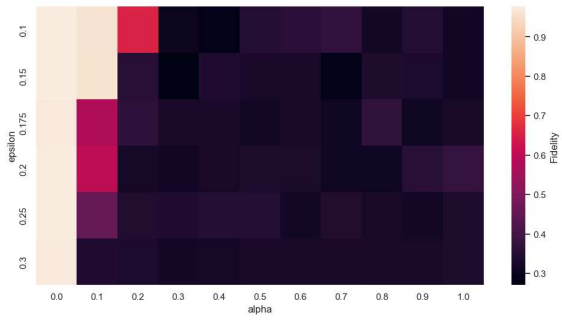
Looking to the general trend we can clearly see that for higher values of ϵ the value of α for which the various agents saturate the available timesteps decreases. In particular we observe the first plot (figure 5.48a) in which the saturation of the time-steps happens only for $\alpha = 0.3$, but already for $\epsilon = 0.15$ we see that the saturation is reached for $\alpha = 0.2$. In the last two cases (figures 5.48e 5.48f) we see that the agents are not able to reach the target for any value of $\alpha \neq 0$. Moreover comparing this figure (5.48) with the MDP-one (figure 5.45) we can identify a very similar behaviour.
Last but not least after we are going to analyze the mean values of both the fidelity and the number of timesteps. In particular we will use two different points of view to study these values: plot of the fidelity in function of α parameter and heatmaps.



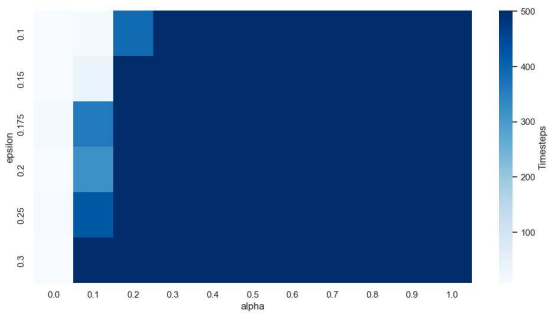
(a) Reward/Fidelity distribution with $\epsilon = 0.1$



(b) Reward/Fidelity distribution with $\epsilon = 0.15$



(c) Fidelity Heatmap



(d) Time-steps Heatmap

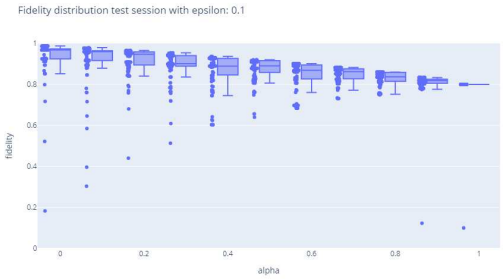
Figure 5.49: Mean fidelity and timesteps plots (POMDP)

We start the analysis focusing on the heatmaps (figure 5.49c, 5.49d). From these latter is clear that even in this case there is a correspondence between number of timesteps used and fidelity achieved. Also we notice that to a high fidelity corresponds a low number of time-steps and vice versa to a low fidelity corresponds a high number of time-steps.

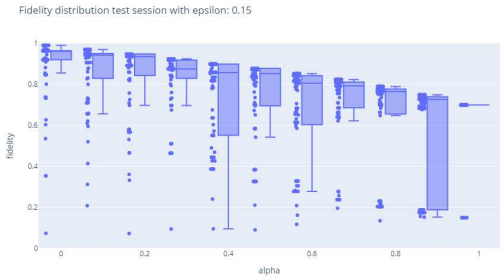
From figure 5.49a we can see that when the system is not affected by the noise every agent is able to achieve a fidelity greater that 95%, instead only the agents trained with $\epsilon = 0.1$ and $\epsilon = 0.15$ are able to maintain a good fidelity also for $\alpha = 0.1$. We can clearly see that for $\alpha \geq 0.3$ no agent is able to control the system. In summary we can argue that this setup is less robust to the noise w.r.t. the MDP-one.

QOMDP SETUP

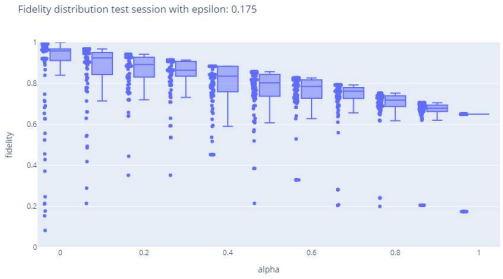
In this section we will present the results of the QOMDP setup with the system affected by a random flip channel, as explained in the section 5.3.1, this is the setup in which we built the *Model-Free* framework, indeed we recall that the state of the agent is represented by the previous control action and by the last observation and the reward is computed in function of the last observation. In the following plots (Figure 5.50) we can see the Fidelity distribution of every agent for different (α) noise parameters.



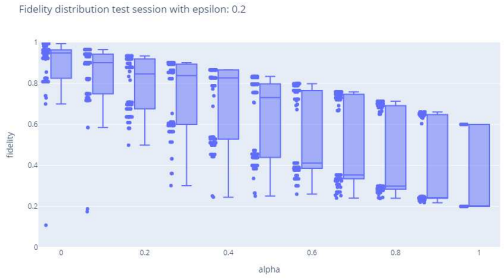
(a) Fidelity distribution with $\epsilon = 0.1$



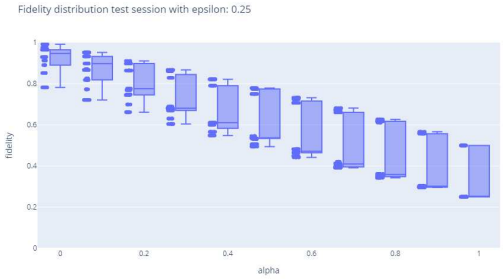
(b) Fidelity distribution with $\epsilon = 0.15$



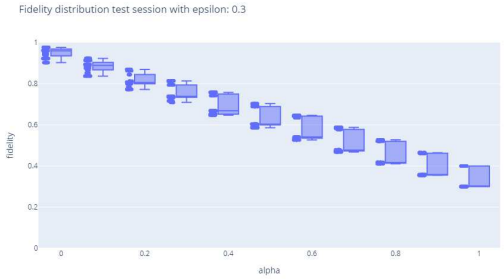
(c) Fidelity distribution with $\epsilon = 0.175$



(d) Fidelity distribution with $\epsilon = 0.2$



(e) Fidelity distribution with $\epsilon = 0.25$

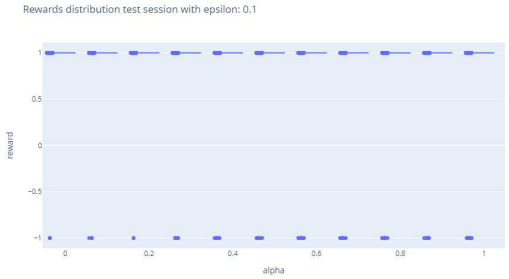


(f) Fidelity distribution with $\epsilon = 0.3$

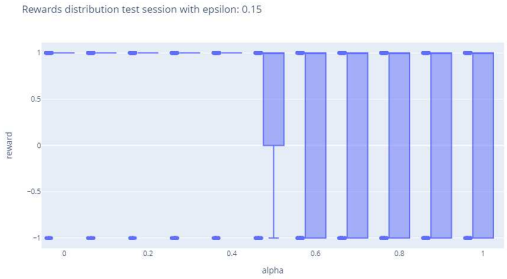
Figure 5.50: Fidelity distribution of different ϵ over the noise α (QOMDP)

From the previous figure 5.50 we can observe the effect of the interaction between the quantity of noise α and the inaccuracy in the measurement ϵ. In

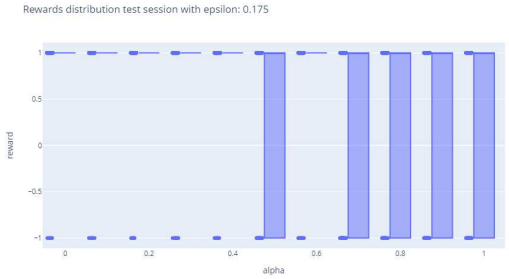
particular we notice that in the first plot the overall variation of the fidelity points is lower than the overall variation of the fidelity points in all the other two setups. As long as we increase the ϵ parameter, as expected, it is possible to notice that the fidelity points for larger α tend to decrease. In the following plots (Figure 5.51) we will see the effect of α on the reward distribution for the various agents.



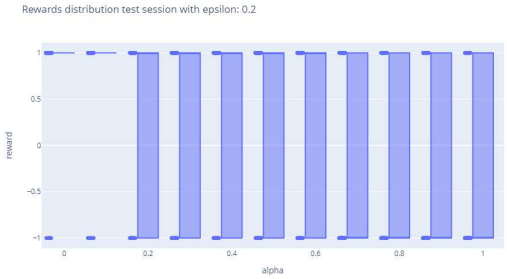
(a) Reward distribution with $\epsilon = 0.1$



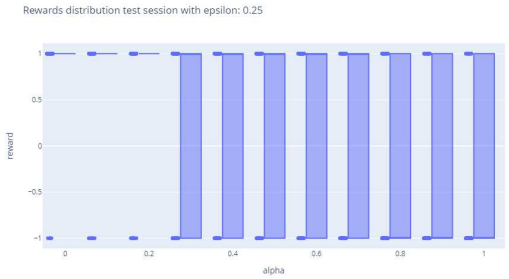
(b) Reward distribution with $\epsilon = 0.15$



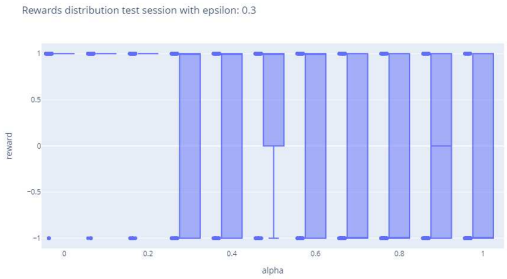
(c) Reward distribution with $\epsilon = 0.175$



(d) Reward distribution with $\epsilon = 0.2$



(e) Reward distribution with $\epsilon = 0.25$



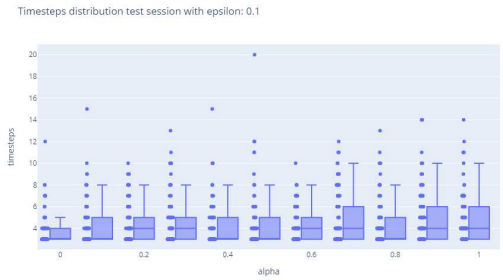
(f) Reward distribution with $\epsilon = 0.3$

Figure 5.51: Reward distribution of different ϵ over the noise α

In this plots (Figure 5.51) we can clearly see the joint effect of the noise and the less informative measurements on the reward dynamics. We start by looking at the plot (Figure 5.51a) where we can see that for all the values of α, the median of the reward distribution is in +1. Going on, increasing the ϵ value

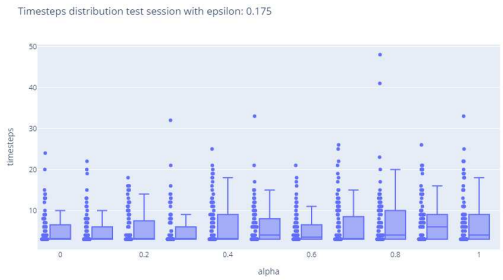
we can see that the values of α for which the reward distribution is concentrated in +1, decreases, in fact the variation starts to grow. This is clearly an effect of the impact of the quality of the information carried out by the outcome measurement. Indeed when we deal with more informative measurements we have better results even under a higher amount of noise, instead when we have less informative measurements and we add some noise, the reward values get worse.

In the following plots (Figure 5.52) we can see the distribution of the number of timesteps that each agent takes to solve the environment. We will plot the timesteps distribution of every trained agent subject to a certain noise α.



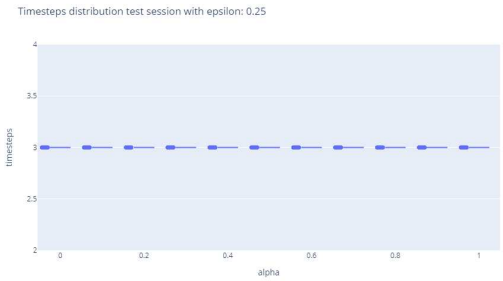
(a) Timesteps distribution with $\epsilon = 0.1$

(b) Timesteps distribution with $\epsilon = 0.15$

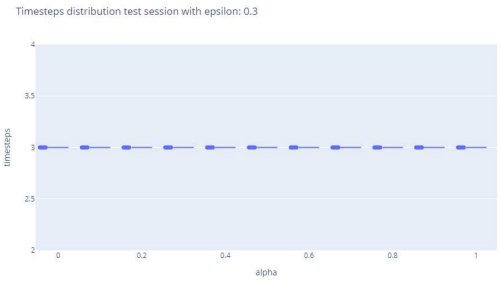


(c) Timesteps distribution with $\epsilon = 0.175$

(d) Timesteps distribution with $\epsilon = 0.2$



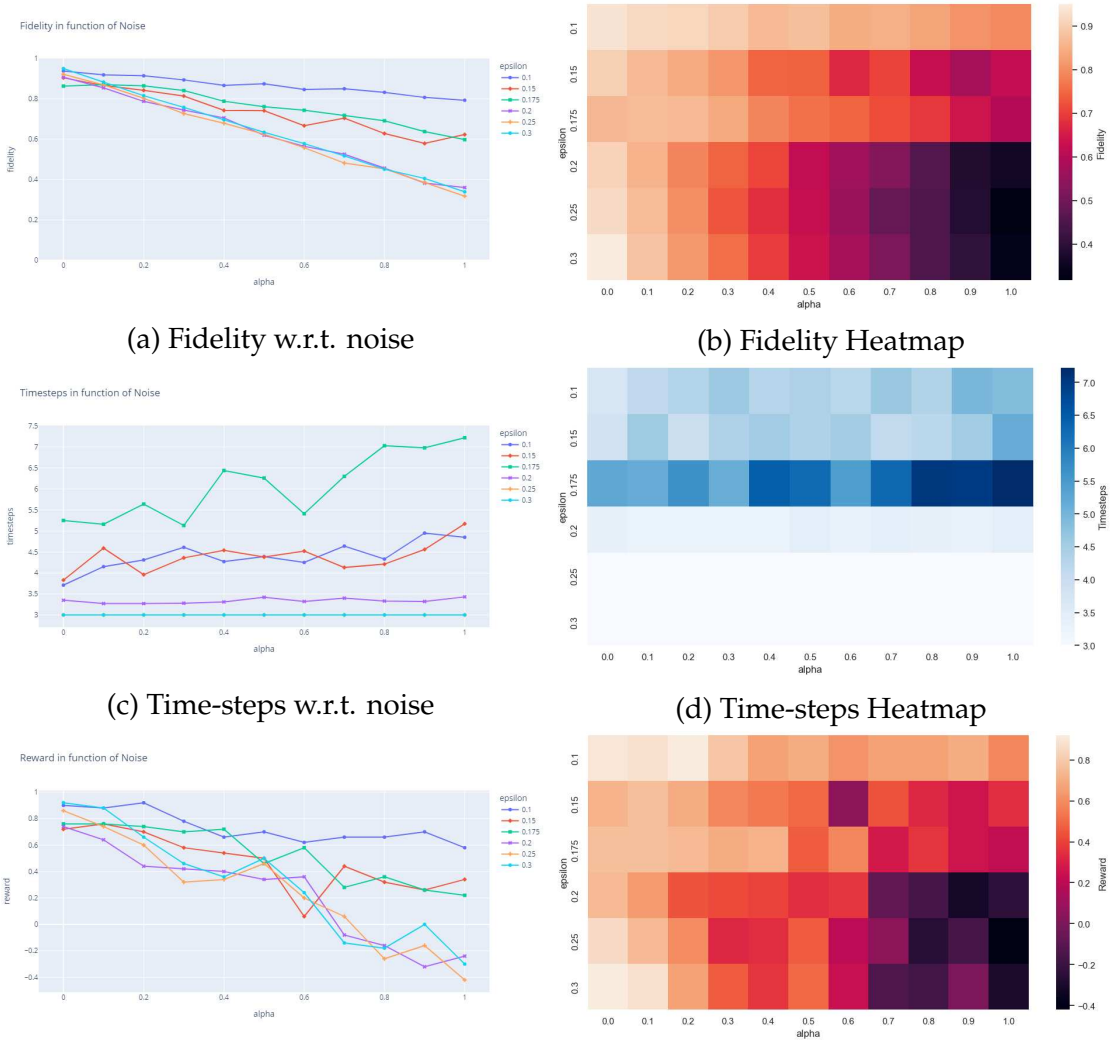
(e) Timesteps distribution with $\epsilon = 0.25$



(f) Timesteps distribution with $\epsilon = 0.3$

Figure 5.52: Timesteps distribution of different ϵ over the noise α (QOMDP)

Looking to the plots in the figures 5.52a, 5.52b, 5.22c, we can observe that as in the previous setups we find a higher number of time-steps for high values of α. This can be seen as an index of the fact that the agent is able to recognize the increased difficulty in the control problem and moreover is able to adapt its policy in order to deal with this further difficulty. Another key aspect of this setup is that w.r.t. the previous ones we can see that the total number of time-steps used by the agent to solve the problem is lower.
Last but not least after we are going to analyze the mean values of fidelity, reward and number of timesteps. In particular we will use two different points of view to study these values: plots and heatmaps.



(a) Fidelity w.r.t. noise

(b) Fidelity Heatmap

(c) Time-steps w.r.t. noise

(d) Time-steps Heatmap

(e) Reward w.r.t. noise

(f) Reward Heatmap

Figure 5.53: Mean fidelity and timesteps plots (QOMDP)

We start the analysis by looking to the fidelity plot and heatmap (figures 5.53a, 5.53b). From the plots we can argue that the behaviour of the fidelity seems linear and the slope increases as long as the ϵ parameter increases. Looking to the heatmap it is clear that the worst fidelity is on the bottom right corner of the map instead the best one is on the top left corner, and this result represent our expectation indeed the performances of the agents get worst as long as we increase the quantity of noise in the system and increasing the inaccuracy on the measurements. In the following table we can find a comparison between some results of this setup and the previous ones.

epsilon	alpha	MDP		POMDP		QOMDP	
		fidelity	std	fidelity	std	fidelity	std
0.1	0.1	**0.97**	**0.01**	0.96	0.01	0.92	0.10
0.1	0.5	0.53	0.35	0.35	0.34	**0.87**	**0.05**
0.1	1	0.31	0.32	0.32	0.32	**0.79**	**0.07**
0.175	0.1	**0.96**	**0.01**	0.58	0.38	0.87	0.14
0.175	0.5	0.40	0.26	0.32	0.20	**0.76**	**0.012**
0.175	1	0.34	0.23	0.33	0.22	**0.60**	**0.15**
0.3	0.1	0.35	0.1	0.34	0.09	**0.88**	**0.02**
0.3	0.5	0.34	0.05	0.33	0.08	**0.63**	**0.04**
0.3	1	0.34	0.05	0.34	0.05	**0.34**	**0.04**

Table 5.17: Random Flip channel results comparison

From the table 5.17 we can argue that regarding the depolarizing noise the MDP and the POMDP setup with less noise and for higher values of ϵ behaves better than the other setup, but increasing the value of ϵ and adding some noise to affect the system dynamics, we notice that the best results are always obtained by this last setup.

By this result and looking also to the trend of the comparison plots of the various setups present in the Appendix A section Random Flip Channel we can argue that for this type of noise the QOMDP setup is the more robust setup.

Looking to the time-steps plot we can see that as increasing α the number of timesteps increases too, this behaviour confirms the fact that the agent is aware about the higher complexity of the problem and tries to deal with it by increasing the number of control actions.

6

Conclusions and Future Works

6.1 Conclusions

This thesis work aimed to build and study a model-free reinforcement learning framework for robust quantum feedback stabilization problems, using a prototypical three-level system as a testbed system.

The central questions for this research were as follows:

1. Is it possible to devise a model-free reinforcement learning method for quantum stabilization when the parameters of the model or the dynamics are not accurately known?

2. How does the interplay between the inaccuracy grade in the measurements (parameterized by ϵ) and the amount of noise in the system (parameterized by α) affect the performance of the agents?

3. Among the available RL approaches, which one is more robust? How much the knowledge of the specific quantum dynamics affect the overall performance?

4. Is there a relationship between the number of control-horizon timesteps and the control accuracy?

With this purpose in mind we adapt to our setting three methods (MDP, POMDP, QOMDP) that rely on different levels knowledge of the underlying dynamics, in order to compare the performance and robustness between the behaviour of an agent that have access to complete "quantum information" (i.e.

has access to the full quantum state) and an agent that relies only on the (classical) measurement output. More in detail, in the first setup (MDP) the agent is provided at each time-step with the density operator of the system and with the actual fidelity as reward. In the second setup (POMDP) we provide to the agent only the outcome of the measurement and the previous control action, but we keep the reward as for the MDP case. We want to remark that these methods would be very hard to realize in a realistic physical setup, as they need precise information about the model parameters and the state to compute (at least) the current reward, and as such they are used mainly as benchmark for the QOMDP approach. In opposition, the third setup (QOMDP) is defined using only the measurement outcomes and the previous control action as the RL agent's input, and a reward function based on some measurement's outcomes. Since it only employs the classical information extracted from the system to be controlled, this setup is always realizable and does not need precise knowledge of the system's density operator. The last approach was inspired by the one used in [19], with a key modification: the addition of the *stop action* which lets the agent decide when stop the control action in order to evaluate the result via a dedicated measurement. We trained all the agents without noise and we tested the performance of the learned controllers with the addition of the noise. In particular we considered four types of noise: classical version of depolarizing channel, depolarizing channel, damping channel, random flip channel.

Before having access to the simulation results, we expected that a model free RL framework (as the QOMDP above) could address the desired task, but leveraging less information about the system, it would also have significantly worse performance than the other setups, at least when a good model of the dynamics was available.

Based on the numerical analysis, the comparison of the three strategy highlights a very similar behaviour irrespective of the specific noise model. In fact we can notice that the MDP and the POMDP approaches perform marginally better then the QOMDP for small noise perturbation, i.e. low values of α, but the performances get dramatically worse when we increase such parameter beyond a given threshold (around $\alpha \sim 0.3 - 0.4$). On the other hand the QOMDP method, while it is slightly outperformed for small values of α, it achieves significantly higher fidelities than the other two for larger values of the noise parameter. This

is clearly shown in the plots in Appendix A, and summarized in the following heatmaps (figure 6.1). This shows that the model-free QOMDP setup is more robust to the noise w.r.t. the MDP and POMDP ones. Moreover the addition of the stop action to the QOMDP setup lets the QOMDP-agents converge quicker than the agents built with the other two setups, allowing for the learning algorithm to adapt the simulation and control horizon to the intrinsic difficulty of the problem.

(a) Depolarizing Channel

(b) Damping Channel

(c) Random Flip Channel

Figure 6.1: Difference between QOMDP and MDP Fidelity Heatmaps

6.2 Future Work

Although this study has provided some valuable insights into the robustness associated with the use of reinforcement learning algorithms in order to stabilize a quantum system, there are several limitations that need to be addressed in future research. Firstly, in this work we only consider a small system. Future studies could explore the effect of dimensionality by considering a scalable system and studying the performance in terms of both robustness and number of time-steps as functions opf the overall dimension. Secondly, in this research we only considered PPO-based algorithms in the learning phase. Different reinforcement learning algorithms might behave differently, and studying and comparing the performances of other RL approaches, as well as current state-of-the-art optimal control algorithms.

In addition to the points above, future research could also explore new directions based on the findings of this study. For instance, it could be interesting to apply similar RL setups to problems like the one treated in the SOMA [15] paper. This could lead to new and more efficient methods with respect to existing open-loop optimal control approaches.

In conclusion, this study has provided some valuable insights about the robustness of model-free RL methods in the design of feedback control laws for quantum systems, opening the possibility for significant new developments. Related research directions, including the ones suggested above, could contribute to a deeper understanding of robust control for quantum systems, and ultimately, have important implications for the quantum control field in general.

Appendices

A

Comparison Plots

A.1 Classical Version of Depolarizing Channel

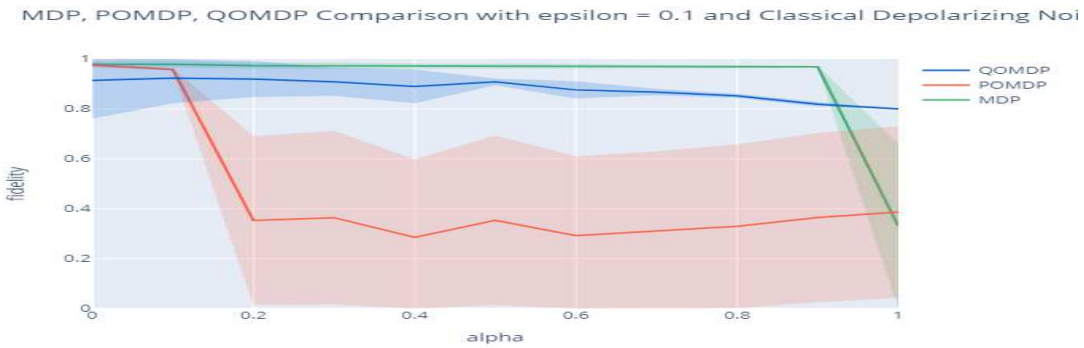


Figure A.1: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.1$

Figure A.2: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.15$

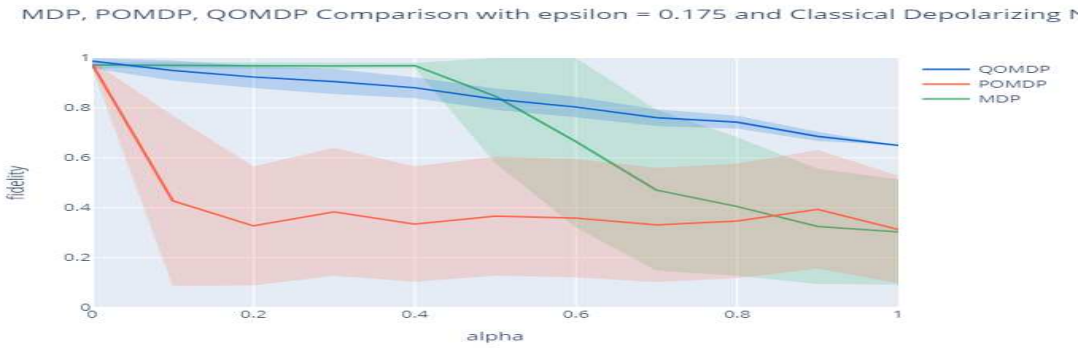


Figure A.3: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.175$

Figure A.4: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.2$

Figure A.5: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.25$

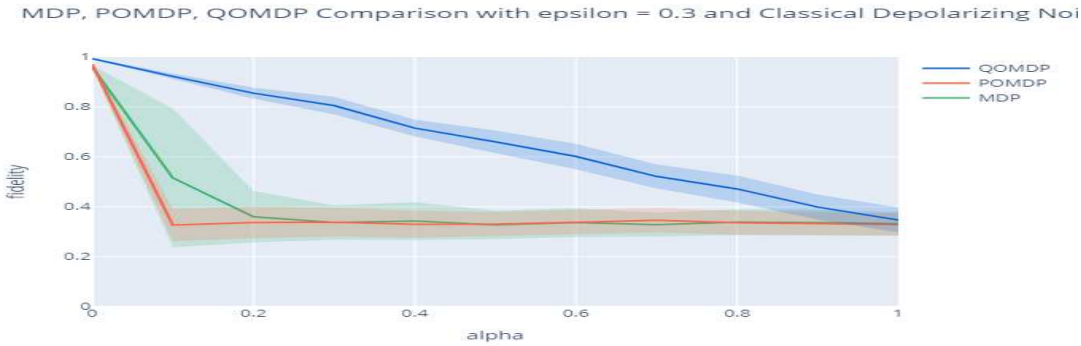


Figure A.6: MDP. POMDP, QOMDP Classical Depolarizing Noise Comparison with $\epsilon = 0.30$

A.2 Depolarizing Channel Noise

Figure A.7: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.1$

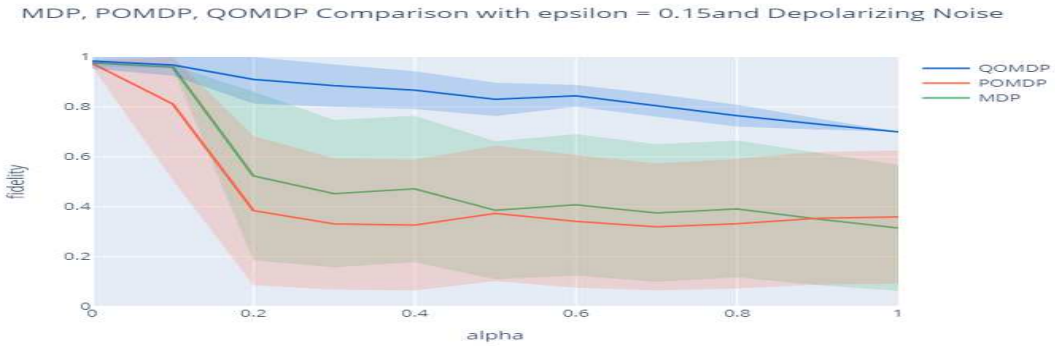


Figure A.8: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.15$

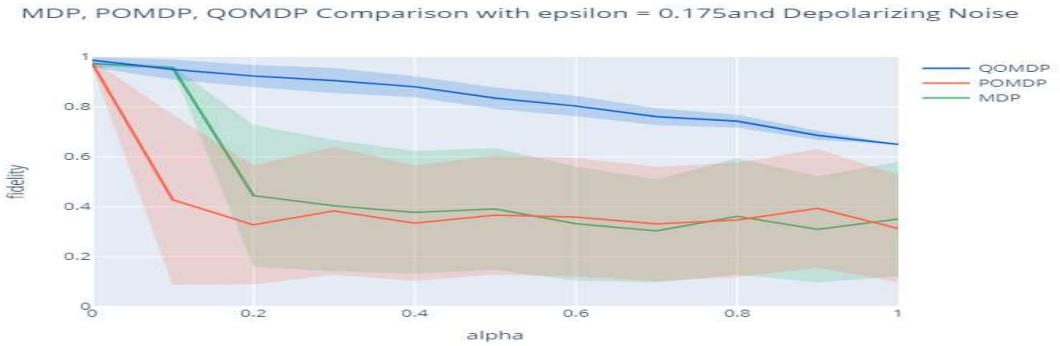


Figure A.9: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.175$

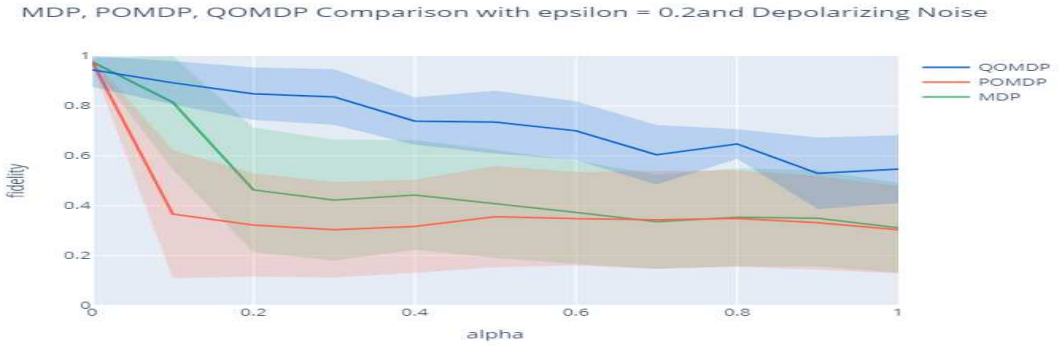


Figure A.10: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.2$

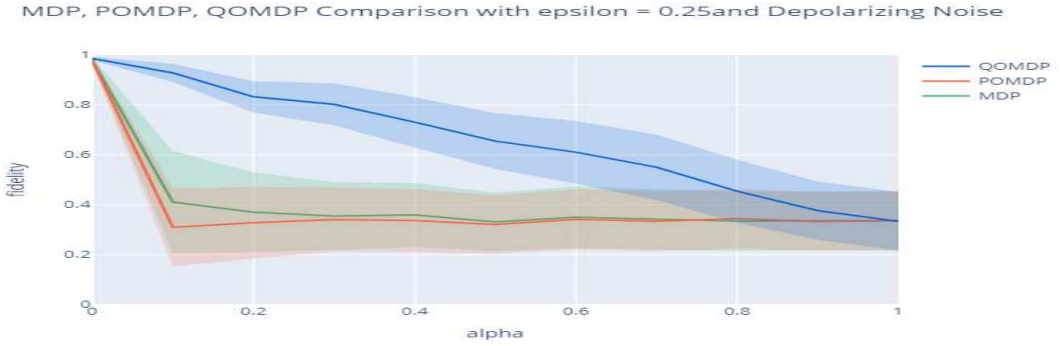


Figure A.11: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.25$

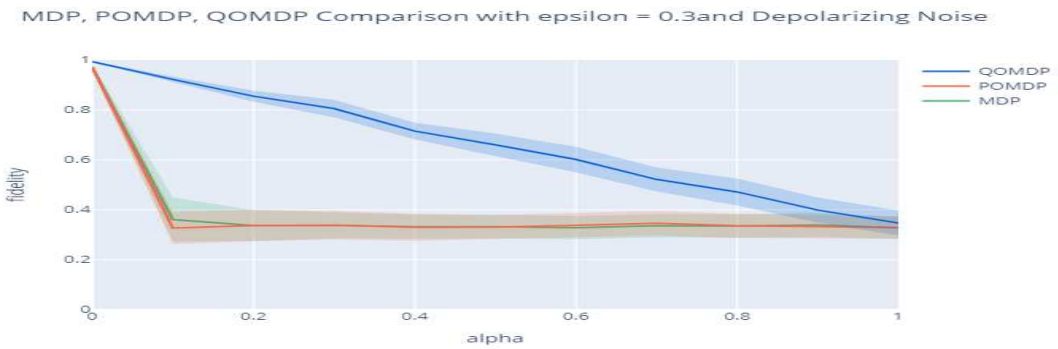


Figure A.12: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.30$

A.3 Damping Channel Noise

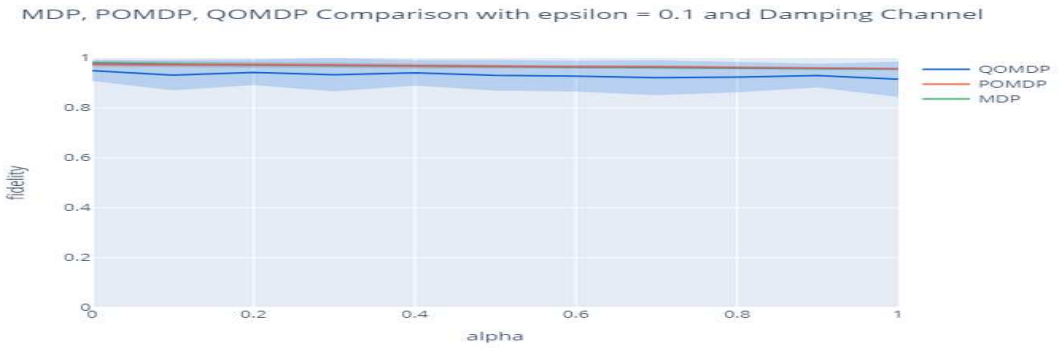


Figure A.13: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.1$

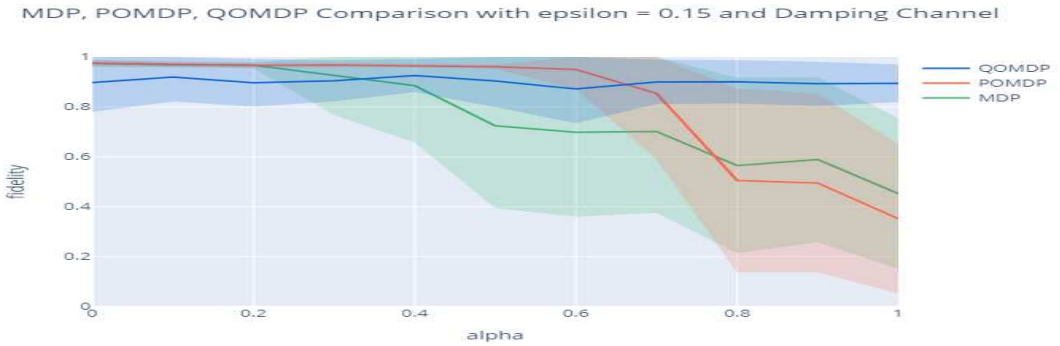


Figure A.14: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.15$

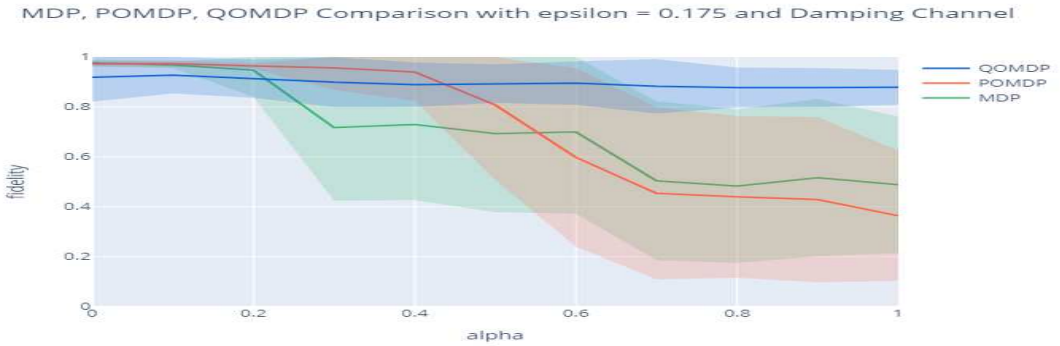


Figure A.15: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.175$

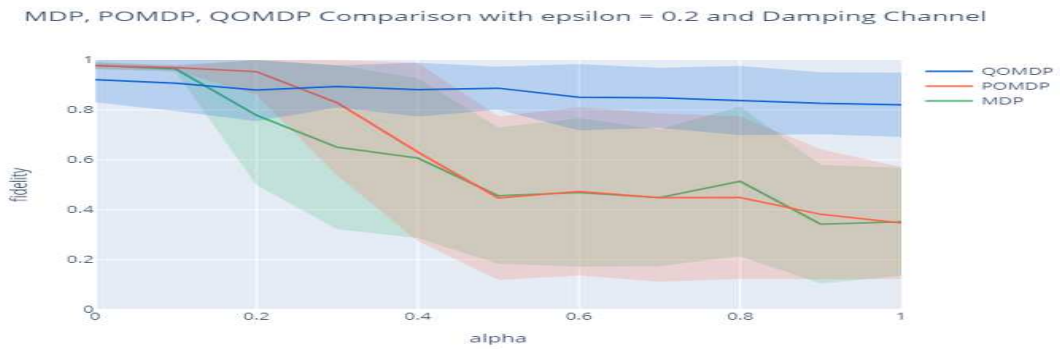


Figure A.16: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.2$

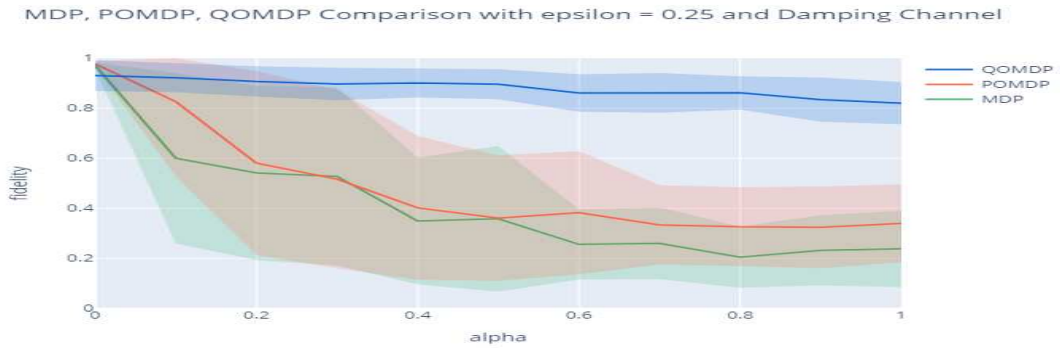


Figure A.17: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.25$

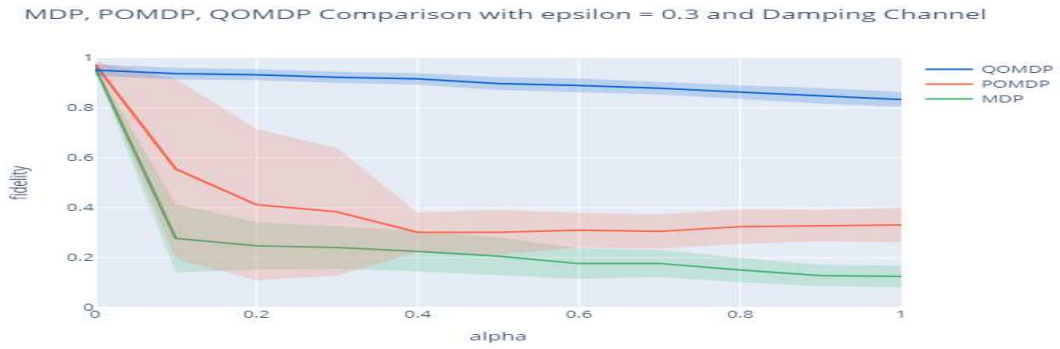


Figure A.18: MDP. POMDP, QOMDP Damping Noise Comparison with $\epsilon = 0.30$

A.4 Random Flip Channel Noise

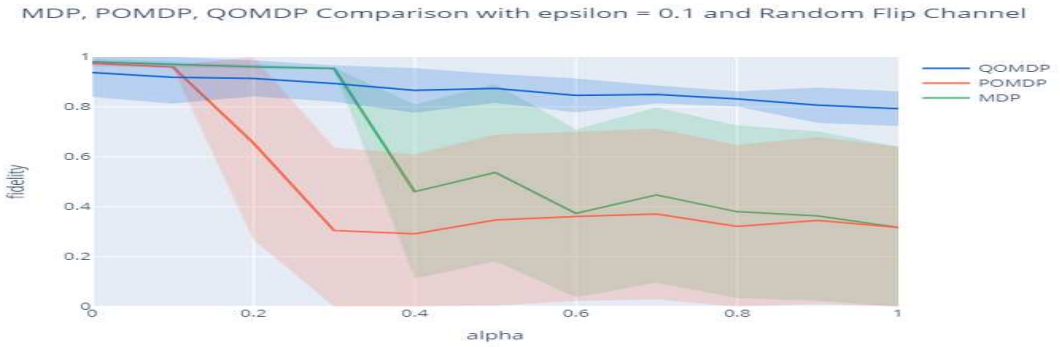


Figure A.19: MDP. POMDP, QOMDP Random Flip Channel Noise Comparison with $\epsilon = 0.1$

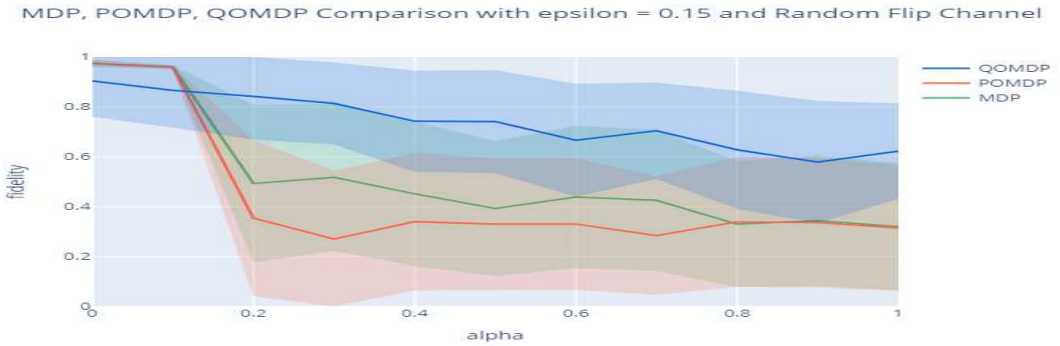


Figure A.20: MDP. POMDP, QOMDP Random Flip Channel Noise Comparison with $\epsilon = 0.15$

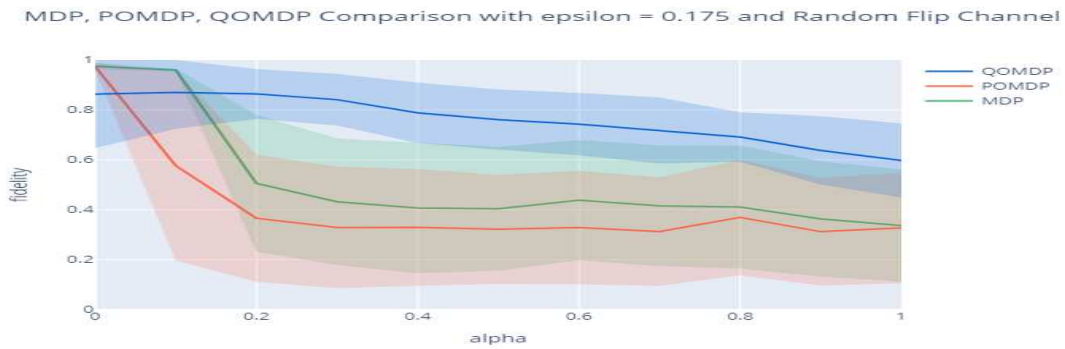


Figure A.21: MDP. POMDP, QOMDP Depolarizing Channel Noise Comparison with $\epsilon = 0.175$

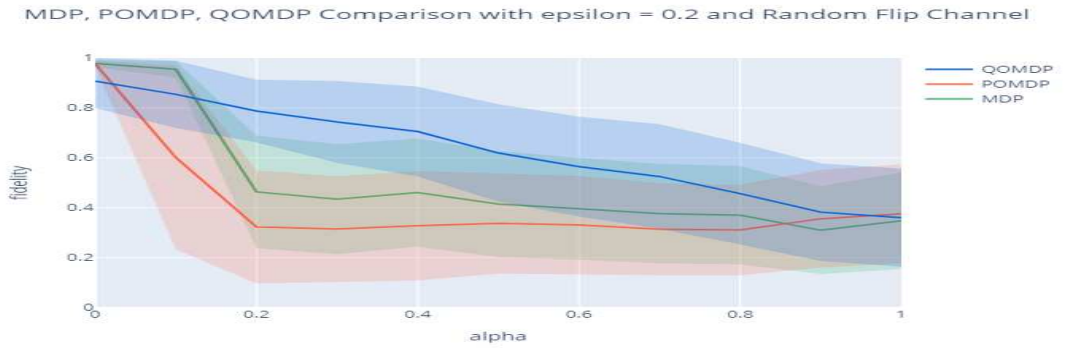


Figure A.22: MDP. POMDP, QOMDP Random Flip Channel Noise Comparison with $\epsilon = 0.2$

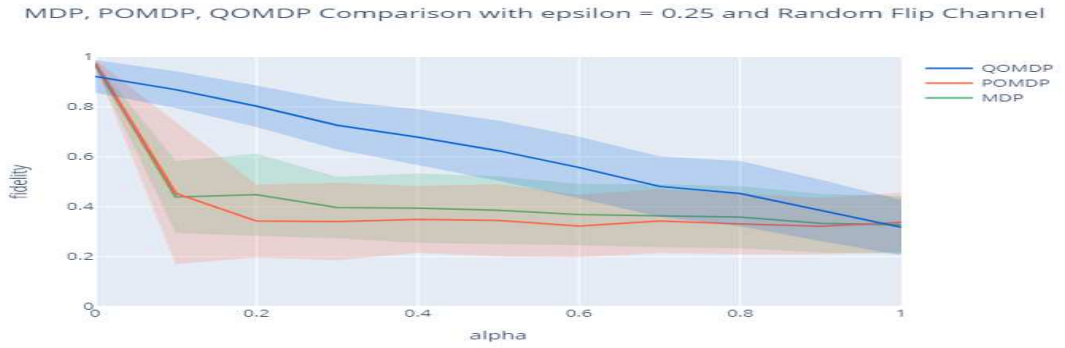


Figure A.23: MDP. POMDP, QOMDP Random Flip Channel Noise Comparison with $\epsilon = 0.25$

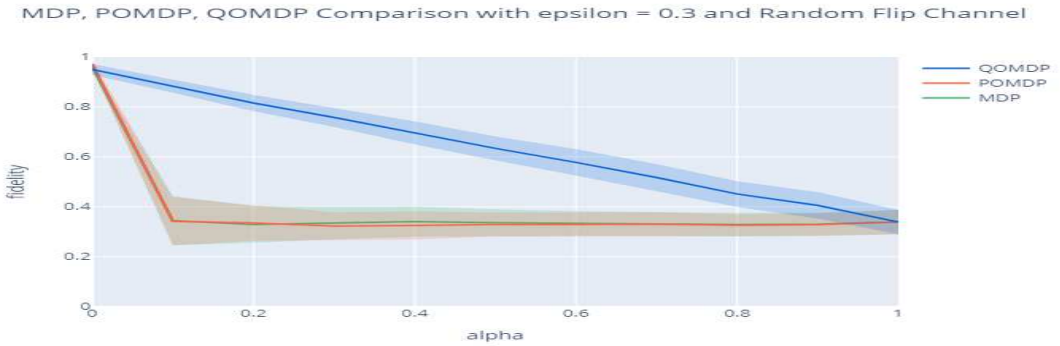


Figure A.24: MDP. POMDP, QOMDP Random Flip Channel Noise Comparison with $\epsilon = 0.30$

B

Results Tables

B.1 Classical Depolarizing Channel

MDP Classic Depolarizing Channel					
epsilon	alpha	reward	reward_std	timesteps	timesteps_std
0,1	0	0,98	0,01	3,76	2,68
0,1	0,1	0,98	0,01	4,34	3,99
0,1	0,2	0,97	0,01	5,26	4,23
0,1	0,3	0,97	0,01	5,17	4,07
0,1	0,4	0,97	0,01	6,87	5,25
0,1	0,5	0,97	0,01	10,15	9,11
0,1	0,6	0,97	0,01	10,12	8,85
0,1	0,7	0,97	0,01	16,07	15,36
0,1	0,8	0,97	0,01	22,68	21,27
0,1	0,9	0,97	0,00	42,66	37,75
0,1	1	0,33	0,33	501,00	0,00
0,15	0	0,97	0,01	4,37	3,73
0,15	0,1	0,97	0,01	8,28	9,53
0,15	0,2	0,97	0,01	12,68	14,44
0,15	0,3	0,97	0,01	18,84	21,18
0,15	0,4	0,96	0,01	34,26	36,24
0,15	0,5	0,96	0,01	62,02	74,80

Table B.1 continued from previous page

0,15	0,6	0,96	0,01	97,56	92,22
0,15	0,7	0,89	0,20	229,79	173,77
0,15	0,8	0,59	0,37	389,93	164,27
0,15	0,9	0,35	0,28	489,25	63,84
0,15	1	0,28	0,23	501,00	0,00
0,175	0	0,97	0,01	5,07	4,51
0,175	0,1	0,97	0,01	9,72	12,98
0,175	0,2	0,97	0,01	25,65	34,40
0,175	0,3	0,97	0,01	50,31	66,89
0,175	0,4	0,97	0,01	101,08	110,30
0,175	0,5	0,85	0,27	214,20	196,84
0,175	0,6	0,67	0,34	359,44	174,68
0,175	0,7	0,47	0,32	442,06	136,26
0,175	0,8	0,40	0,28	485,13	82,84
0,175	0,9	0,33	0,23	501,00	0,00
0,175	1	0,30	0,21	501,00	0,00
0,2	0	0,98	0,01	5,72	5,78
0,2	0,1	0,98	0,01	15,53	23,21
0,2	0,2	0,97	0,01	39,83	53,19
0,2	0,3	0,97	0,01	80,43	93,08
0,2	0,4	0,92	0,17	165,88	159,89
0,2	0,5	0,76	0,31	279,06	196,60
0,2	0,6	0,58	0,33	392,80	173,41
0,2	0,7	0,42	0,30	454,25	135,95
0,2	0,8	0,39	0,23	485,58	78,47
0,2	0,9	0,34	0,21	501,00	0,00
0,2	1	0,34	0,19	501,00	0,00
0,25	0	0,97	0,01	22,73	30,44
0,25	0,1	0,85	0,23	164,69	216,80
0,25	0,2	0,63	0,29	334,21	229,07
0,25	0,3	0,56	0,31	350,05	227,84
0,25	0,4	0,47	0,27	406,21	195,72
0,25	0,5	0,49	0,27	401,26	199,48
0,25	0,6	0,39	0,19	481,04	97,78

Table B.1 continued from previous page

0,25	0,7	0,40	0,19	476,07	108,67
0,25	0,8	0,35	0,14	496,01	49,65
0,25	0,9	0,36	0,15	491,02	69,86
0,25	1	0,34	0,12	501,00	0,00
0,3	0	0,96	0,01	17,77	4,93
0,3	0,1	0,52	0,28	391,61	196,82
0,3	0,2	0,36	0,10	496,21	47,66
0,3	0,3	0,34	0,07	501,00	0,00
0,3	0,4	0,34	0,08	501,00	0,00
0,3	0,5	0,33	0,06	501,00	0,00
0,3	0,6	0,34	0,06	501,00	0,00
0,3	0,7	0,33	0,05	501,00	0,00
0,3	0,8	0,34	0,05	501,00	0,00
0,3	0,9	0,34	0,05	501,00	0,00
0,3	1	0,33	0,05	501,00	0,00

Table B.1: MDP Classical version of Depolarizing Channel

POMDP Classical Depolarizing Channel					
epsilon	alpha	reward	reward_std	timesteps	timesteps_std
0,1	0	0,97	0,01	9,30	11,66
0,1	0,1	0,97	0,01	12,79	15,30
0,1	0,2	0,97	0,01	17,69	17,67
0,1	0,3	0,96	0,01	24,66	32,09
0,1	0,4	0,96	0,01	29,06	33,94
0,1	0,5	0,96	0,01	38,35	36,77
0,1	0,6	0,96	0,01	77,20	82,81
0,1	0,7	0,95	0,01	106,13	101,29
0,1	0,8	0,87	0,24	211,39	173,50
0,1	0,9	0,50	0,40	418,26	143,60
0,1	1	0,39	0,35	501,00	0,00
0,15	0	0,97	0,01	10,17	10,09
0,15	0,1	0,97	0,01	11,54	12,05
0,15	0,2	0,97	0,01	19,44	21,67

Table B.2 continued from previous page

0,15	0,3	0,96	0,01	30,21	31,13
0,15	0,4	0,96	0,01	47,66	52,89
0,15	0,5	0,96	0,01	83,24	67,51
0,15	0,6	0,95	0,08	125,03	122,58
0,15	0,7	0,81	0,30	278,58	176,19
0,15	0,8	0,57	0,37	401,98	159,78
0,15	0,9	0,40	0,31	475,15	85,26
0,15	1	0,33	0,26	501,00	0,00
0,175	0	0,97	0,01	8,46	10,55
0,175	0,1	0,97	0,01	15,11	17,98
0,175	0,2	0,97	0,01	26,62	33,40
0,175	0,3	0,97	0,01	48,52	59,12
0,175	0,4	0,96	0,02	86,98	99,14
0,175	0,5	0,94	0,13	143,44	143,34
0,175	0,6	0,82	0,28	271,16	188,67
0,175	0,7	0,59	0,35	392,48	168,31
0,175	0,8	0,47	0,31	452,72	117,56
0,175	0,9	0,31	0,24	494,20	42,72
0,175	1	0,32	0,22	501,00	0,00
0,2	0	0,98	0,01	10,75	5,01
0,2	0,1	0,97	0,01	21,93	18,46
0,2	0,2	0,97	0,01	40,66	37,55
0,2	0,3	0,97	0,01	67,70	48,33
0,2	0,4	0,95	0,12	145,44	131,19
0,2	0,5	0,88	0,22	198,40	164,46
0,2	0,6	0,73	0,34	313,33	187,24
0,2	0,7	0,58	0,35	399,94	152,67
0,2	0,8	0,46	0,32	441,03	132,11
0,2	0,9	0,32	0,19	500,96	0,40
0,2	1	0,32	0,18	501,00	0,00
0,25	0	0,97	0,01	16,67	11,77
0,25	0,1	0,95	0,11	121,79	122,46
0,25	0,2	0,72	0,31	319,68	182,26
0,25	0,3	0,50	0,31	440,33	122,81

Table B.2 continued from previous page

0,25	0,4	0,41	0,23	477,18	89,71
0,25	0,5	0,35	0,18	490,72	60,25
0,25	0,6	0,36	0,16	497,75	32,34
0,25	0,7	0,36	0,14	501,00	0,00
0,25	0,8	0,33	0,13	501,00	0,00
0,25	0,9	0,32	0,11	501,00	0,00
0,25	1	0,34	0,12	501,00	0,00
0,3	0	0,96	0,01	67,55	44,47
0,3	0,1	0,37	0,18	496,33	46,27
0,3	0,2	0,33	0,10	501,00	0,00
0,3	0,3	0,35	0,09	501,00	0,00
0,3	0,4	0,33	0,07	501,00	0,00
0,3	0,5	0,32	0,05	501,00	0,00
0,3	0,6	0,33	0,06	501,00	0,00
0,3	0,7	0,34	0,06	501,00	0,00
0,3	0,8	0,33	0,05	501,00	0,00
0,3	0,9	0,33	0,05	501,00	0,00
0,3	1	0,33	0,05	501,00	0,00

Table B.2: POMDP Classical Depolarizing Channel

QOMDP Classical Depolarizing Channel							
epsilon	alpha	reward	reward_std	timesteps	timesteps_std	fidelity	fidelity_std
0,1	0	0,90	0,44	2,39	0,96	0,96	0,13
0,1	0,1	0,92	0,39	2,41	0,97	0,98	0,05
0,1	0,2	0,90	0,44	2,97	1,65	0,94	0,08
0,1	0,3	0,92	0,39	3,02	1,94	0,94	0,10
0,1	0,4	0,88	0,47	2,82	1,40	0,91	0,12
0,1	0,5	0,84	0,54	2,96	1,59	0,92	0,09
0,1	0,6	0,68	0,73	3,43	1,86	0,88	0,12
0,1	0,7	0,72	0,69	3,17	1,70	0,86	0,11
0,1	0,8	0,80	0,60	3,28	1,84	0,86	0,08
0,1	0,9	0,56	0,83	3,83	2,11	0,83	0,07
0,1	1	0,58	0,81	4,24	2,78	0,80	0,00

Table B.3 continued from previous page

0,15	0	0,94	0,34	3,21	2,43	0,97	0,10
0,15	0,1	0,86	0,51	3,66	3,31	0,93	0,11
0,15	0,2	0,80	0,60	3,11	2,36	0,92	0,12
0,15	0,3	0,82	0,57	4,17	3,74	0,90	0,12
0,15	0,4	0,82	0,57	4,34	4,21	0,88	0,13
0,15	0,5	0,74	0,67	5,42	6,19	0,83	0,14
0,15	0,6	0,56	0,83	4,64	4,07	0,83	0,13
0,15	0,7	0,40	0,92	6,33	5,82	0,76	0,11
0,15	0,8	0,46	0,89	6,03	5,78	0,76	0,11
0,15	0,9	0,48	0,88	7,01	6,39	0,72	0,07
0,15	1	0,42	0,91	7,51	6,76	0,70	0,00
0,175	0	1,00	0,00	4,38	4,29	0,97	0,04
0,175	0,1	0,92	0,39	3,76	3,52	0,95	0,10
0,175	0,2	0,90	0,44	4,01	3,42	0,93	0,11
0,175	0,3	0,76	0,65	5,50	6,39	0,90	0,13
0,175	0,4	0,70	0,71	6,31	7,28	0,84	0,16
0,175	0,5	0,76	0,65	6,81	8,13	0,82	0,16
0,175	0,6	0,60	0,80	6,41	6,64	0,77	0,16
0,175	0,7	0,56	0,83	7,83	8,57	0,76	0,15
0,175	0,8	0,26	0,97	8,51	8,05	0,71	0,11
0,175	0,9	0,36	0,93	9,88	9,33	0,70	0,11
0,175	1	0,10	0,99	9,30	9,65	0,65	0,00
0,2	0	0,76	0,65	3,53	2,95	0,92	0,10
0,2	0,1	0,82	0,57	3,48	2,92	0,91	0,13
0,2	0,2	0,64	0,77	3,69	3,45	0,84	0,18
0,2	0,3	0,64	0,77	3,45	2,67	0,79	0,21
0,2	0,4	0,44	0,90	3,96	3,33	0,77	0,20
0,2	0,5	0,56	0,83	4,52	3,82	0,74	0,20
0,2	0,6	0,22	0,98	5,76	4,87	0,66	0,17
0,2	0,7	0,20	0,98	4,58	3,80	0,67	0,15
0,2	0,8	0,32	0,95	4,84	4,54	0,63	0,18
0,2	0,9	0,28	0,96	5,24	4,70	0,60	0,14
0,2	1	0,24	0,97	6,29	5,43	0,57	0,10
0,25	0	0,94	0,34	2,00	0,00	0,98	0,01

Table B.3 continued from previous page

0,25	0,1	0,86	0,51	2,00	0,00	0,91	0,22
0,25	0,2	0,66	0,75	2,00	0,00	0,86	0,25
0,25	0,3	0,58	0,81	2,00	0,00	0,78	0,31
0,25	0,4	0,46	0,89	2,00	0,00	0,72	0,34
0,25	0,5	0,32	0,95	2,00	0,00	0,67	0,33
0,25	0,6	0,20	0,98	2,00	0,00	0,60	0,34
0,25	0,7	0,18	0,98	2,00	0,00	0,54	0,33
0,25	0,8	-0,12	0,99	2,00	0,00	0,43	0,26
0,25	0,9	-0,26	0,97	2,00	0,00	0,41	0,23
0,25	1	-0,28	0,96	2,00	0,00	0,35	0,12
0,3	0	0,98	0,20	2,00	0,00	0,99	0,00
0,3	0,1	0,80	0,60	2,00	0,00	0,90	0,22
0,3	0,2	0,78	0,63	2,00	0,00	0,87	0,25
0,3	0,3	0,64	0,77	2,00	0,00	0,81	0,29
0,3	0,4	0,30	0,95	2,00	0,00	0,66	0,33
0,3	0,5	0,38	0,92	2,00	0,00	0,66	0,33
0,3	0,6	0,34	0,94	2,00	0,00	0,66	0,33
0,3	0,7	-0,12	0,99	2,00	0,00	0,47	0,27
0,3	0,8	-0,02	1,00	2,00	0,00	0,45	0,25
0,3	0,9	-0,10	0,99	2,00	0,00	0,42	0,23
0,3	1	-0,44	0,90	2,00	0,00	0,33	0,04

Table B.3: QOMDP Classical version of Depolarizing Channel

B.2 Depolarizing Channel

MDP Depolarizing Channel					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,98	0,01	4,02	3,21
0,1	0,1	0,96	0,01	5,63	5,25
0,1	0,2	0,97	0,01	12,78	11,10
0,1	0,3	0,96	0,00	22,46	21,25
0,1	0,4	0,41	0,36	501,00	0,00
0,1	0,5	0,43	0,36	501,00	0,00

Table B.4 continued from previous page

0,1	0,6	0,36	0,34	501,00	0,00
0,1	0,7	0,34	0,33	501,00	0,00
0,1	0,8	0,34	0,33	501,00	0,00
0,1	0,9	0,40	0,35	501,00	0,00
0,1	1	0,35	0,34	501,00	0,00
0,15	0	0,98	0,01	4,23	4,51
0,15	0,1	0,96	0,01	31,86	29,63
0,15	0,2	0,52	0,34	477,14	95,30
0,15	0,3	0,45	0,30	501,00	0,00
0,15	0,4	0,47	0,29	501,00	0,00
0,15	0,5	0,39	0,28	501,00	0,00
0,15	0,6	0,41	0,28	501,00	0,00
0,15	0,7	0,37	0,28	501,00	0,00
0,15	0,8	0,39	0,27	501,00	0,00
0,15	0,9	0,35	0,26	501,00	0,00
0,15	1	0,32	0,25	501,00	0,00
0,175	0	0,97	0,01	5,24	4,40
0,175	0,1	0,96	0,01	27,68	36,35
0,175	0,2	0,44	0,28	501,00	0,00
0,175	0,3	0,40	0,26	501,00	0,00
0,175	0,4	0,38	0,25	501,00	0,00
0,175	0,5	0,39	0,24	501,00	0,00
0,175	0,6	0,33	0,23	501,00	0,00
0,175	0,7	0,30	0,21	501,00	0,00
0,175	0,8	0,36	0,23	501,00	0,00
0,175	0,9	0,31	0,21	501,00	0,00
0,175	1	0,35	0,23	501,00	0,00
0,2	0	0,98	0,02	6,98	10,04
0,2	0,1	0,81	0,27	291,93	180,88
0,2	0,2	0,46	0,25	501,00	0,00
0,2	0,3	0,42	0,24	501,00	0,00
0,2	0,4	0,44	0,22	501,00	0,00
0,2	0,5	0,41	0,22	501,00	0,00
0,2	0,6	0,37	0,21	501,00	0,00

Table B.4 continued from previous page

0,2	0,7	0,34	0,19	501,00	0,00
0,2	0,8	0,35	0,20	501,00	0,00
0,2	0,9	0,35	0,19	501,00	0,00
0,2	1	0,31	0,18	501,00	0,00
0,25	0	0,98	0,01	4,36	9,11
0,25	0,1	0,41	0,20	501,00	0,00
0,25	0,2	0,37	0,16	501,00	0,00
0,25	0,3	0,35	0,14	501,00	0,00
0,25	0,4	0,36	0,13	501,00	0,00
0,25	0,5	0,33	0,12	501,00	0,00
0,25	0,6	0,35	0,12	501,00	0,00
0,25	0,7	0,34	0,12	501,00	0,00
0,25	0,8	0,33	0,12	501,00	0,00
0,25	0,9	0,34	0,12	501,00	0,00
0,25	1	0,34	0,12	501,00	0,00
0,3	0	0,97	0,01	3,36	1,98
0,3	0,1	0,36	0,09	501,00	0,00
0,3	0,2	0,34	0,06	501,00	0,00
0,3	0,3	0,34	0,05	501,00	0,00
0,3	0,4	0,33	0,05	501,00	0,00
0,3	0,5	0,33	0,05	501,00	0,00
0,3	0,6	0,33	0,05	501,00	0,00
0,3	0,7	0,34	0,05	501,00	0,00
0,3	0,8	0,33	0,05	501,00	0,00
0,3	0,9	0,34	0,05	501,00	0,00
0,3	1	0,33	0,04	501,00	0,00

Table B.4: MDP Depolarizing Channel

POMDP Depolarizing Channel					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,98	0,01	6,64	7,54
0,1	0,1	0,96	0,01	19,52	27,01
0,1	0,2	0,35	0,34	501,00	0,00

Table B.5 continued from previous page

0,1	0,3	0,36	0,35	501,00	0,00
0,1	0,4	0,29	0,31	501,00	0,00
0,1	0,5	0,35	0,34	501,00	0,00
0,1	0,6	0,29	0,32	501,00	0,00
0,1	0,7	0,31	0,32	501,00	0,00
0,1	0,8	0,33	0,33	501,00	0,00
0,1	0,9	0,36	0,34	501,00	0,00
0,1	1	0,39	0,34	501,00	0,00
0,15	0	0,97	0,01	8,62	11,62
0,15	0,1	0,81	0,30	244,99	198,78
0,15	0,2	0,38	0,30	501,00	0,00
0,15	0,3	0,33	0,26	501,00	0,00
0,15	0,4	0,33	0,26	501,00	0,00
0,15	0,5	0,37	0,27	501,00	0,00
0,15	0,6	0,34	0,27	501,00	0,00
0,15	0,7	0,32	0,25	501,00	0,00
0,15	0,8	0,33	0,26	501,00	0,00
0,15	0,9	0,35	0,27	501,00	0,00
0,15	1	0,36	0,27	501,00	0,00
0,175	0	0,97	0,01	10,99	14,08
0,175	0,1	0,43	0,34	426,15	178,18
0,175	0,2	0,33	0,24	501,00	0,00
0,175	0,3	0,38	0,26	501,00	0,00
0,175	0,4	0,33	0,23	501,00	0,00
0,175	0,5	0,37	0,24	501,00	0,00
0,175	0,6	0,36	0,24	501,00	0,00
0,175	0,7	0,33	0,23	501,00	0,00
0,175	0,8	0,35	0,23	501,00	0,00
0,175	0,9	0,39	0,24	501,00	0,00
0,175	1	0,31	0,22	501,00	0,00
0,2	0	0,98	0,01	15,79	24,38
0,2	0,1	0,37	0,26	501,00	0,00
0,2	0,2	0,32	0,21	501,00	0,00
0,2	0,3	0,30	0,19	501,00	0,00

Table B.5 continued from previous page

0,2	0,4	0,32	0,19	501,00	0,00
0,2	0,5	0,36	0,20	501,00	0,00
0,2	0,6	0,35	0,19	501,00	0,00
0,2	0,7	0,34	0,20	501,00	0,00
0,2	0,8	0,35	0,19	501,00	0,00
0,2	0,9	0,33	0,19	501,00	0,00
0,2	1	0,30	0,18	501,00	0,00
0,25	0	0,98	0,01	15,52	32,83
0,25	0,1	0,31	0,16	501,00	0,00
0,25	0,2	0,33	0,14	501,00	0,00
0,25	0,3	0,34	0,13	501,00	0,00
0,25	0,4	0,34	0,13	501,00	0,00
0,25	0,5	0,32	0,12	501,00	0,00
0,25	0,6	0,34	0,12	501,00	0,00
0,25	0,7	0,33	0,12	501,00	0,00
0,25	0,8	0,34	0,12	501,00	0,00
0,25	0,9	0,33	0,12	501,00	0,00
0,25	1	0,34	0,12	501,00	0,00
0,3	0	0,97	0,01	6,66	6,23
0,3	0,1	0,33	0,06	501,00	0,00
0,3	0,2	0,34	0,06	501,00	0,00
0,3	0,3	0,34	0,06	501,00	0,00
0,3	0,4	0,33	0,05	501,00	0,00
0,3	0,5	0,33	0,05	501,00	0,00
0,3	0,6	0,34	0,05	501,00	0,00
0,3	0,7	0,35	0,05	501,00	0,00
0,3	0,8	0,34	0,05	501,00	0,00
0,3	0,9	0,33	0,05	501,00	0,00
0,3	1	0,33	0,05	501,00	0,00

Table B.5: POMDP Depolarizing Channel

QOMDP Depolarizing Channel							
epsilon	alpha	reward	reward_std	timesteps	timesteps_std	fidelity	fidelity_std

Table B.6 continued from previous page

0,1	0	0,73	0,68	2,53	0,81	0,91	0,15
0,1	0,1	0,87	0,50	2,73	2,24	0,92	0,10
0,1	0,2	0,60	0,80	5,00	5,39	0,92	0,07
0,1	0,3	0,73	0,68	3,67	3,05	0,91	0,06
0,1	0,4	0,60	0,80	5,53	8,06	0,89	0,07
0,1	0,5	0,33	0,94	5,67	6,13	0,91	0,01
0,1	0,6	0,87	0,50	5,47	4,51	0,88	0,03
0,1	0,7	0,60	0,80	5,60	4,87	0,87	0,01
0,1	0,8	0,60	0,80	5,00	4,63	0,85	0,01
0,1	0,9	0,87	0,50	9,13	10,03	0,82	0,01
0,1	1	0,33	0,94	9,40	12,45	0,80	0,00
0,15	0	1,00	0,00	4,73	4,91	0,98	0,03
0,15	0,1	0,87	0,50	3,27	2,89	0,97	0,04
0,15	0,2	0,73	0,68	2,67	1,19	0,91	0,10
0,15	0,3	0,73	0,68	5,33	6,11	0,89	0,09
0,15	0,4	0,73	0,68	3,73	2,52	0,87	0,08
0,15	0,5	0,60	0,80	6,67	4,63	0,83	0,07
0,15	0,6	1,00	0,00	3,40	2,30	0,84	0,04
0,15	0,7	0,47	0,88	4,27	4,01	0,81	0,05
0,15	0,8	0,73	0,68	3,93	3,36	0,77	0,04
0,15	0,9	0,33	0,94	5,67	4,01	0,73	0,02
0,15	1	0,33	0,94	5,40	4,53	0,70	0,00
0,175	0	1,00	0,00	2,47	0,88	0,99	0,03
0,175	0,1	0,87	0,50	4,60	4,05	0,95	0,04
0,175	0,2	0,87	0,50	7,47	10,39	0,92	0,04
0,175	0,3	0,60	0,80	3,60	3,72	0,91	0,05
0,175	0,4	1,00	0,00	3,67	2,98	0,88	0,04
0,175	0,5	0,73	0,68	7,93	7,55	0,84	0,04
0,175	0,6	0,73	0,68	9,07	11,99	0,80	0,04
0,175	0,7	0,33	0,94	13,40	11,35	0,76	0,03
0,175	0,8	0,60	0,80	7,13	9,35	0,74	0,03
0,175	0,9	0,73	0,68	9,27	10,07	0,69	0,02
0,175	1	0,47	0,88	3,47	3,03	0,65	0,00
0,2	0	0,87	0,50	2,67	1,49	0,94	0,07

Table B.6 continued from previous page

0,2	0,1	1,00	0,00	3,80	3,67	0,89	0,09
0,2	0,2	0,87	0,50	3,87	3,26	0,85	0,11
0,2	0,3	0,73	0,68	3,27	2,11	0,84	0,11
0,2	0,4	0,47	0,88	7,13	8,79	0,74	0,09
0,2	0,5	0,73	0,68	3,20	1,87	0,74	0,13
0,2	0,6	0,47	0,88	6,73	7,75	0,70	0,12
0,2	0,7	0,20	0,98	5,60	5,25	0,60	0,12
0,2	0,8	0,07	1,00	4,93	3,59	0,65	0,06
0,2	0,9	0,07	1,00	3,87	2,55	0,53	0,14
0,2	1	-0,20	0,98	3,87	3,16	0,55	0,14
0,25	0	1,00	0,00	2,00	0,00	0,99	0,01
0,25	0,1	1,00	0,00	2,00	0,00	0,93	0,04
0,25	0,2	0,47	0,88	2,00	0,00	0,83	0,06
0,25	0,3	0,47	0,88	2,00	0,00	0,80	0,08
0,25	0,4	0,33	0,94	2,00	0,00	0,73	0,10
0,25	0,5	-0,20	0,98	2,00	0,00	0,65	0,11
0,25	0,6	-0,20	0,98	2,00	0,00	0,61	0,13
0,25	0,7	0,07	1,00	2,00	0,00	0,55	0,13
0,25	0,8	-0,20	0,98	2,00	0,00	0,45	0,13
0,25	0,9	0,07	1,00	2,00	0,00	0,38	0,12
0,25	1	-0,47	0,88	2,00	0,00	0,33	0,12
0,3	0	1,00	0,00	2,00	0,00	0,99	0,00
0,3	0,1	0,87	0,50	2,00	0,00	0,92	0,01
0,3	0,2	0,33	0,94	2,00	0,00	0,86	0,02
0,3	0,3	0,47	0,88	2,00	0,00	0,81	0,03
0,3	0,4	0,33	0,94	2,00	0,00	0,72	0,03
0,3	0,5	0,07	1,00	2,00	0,00	0,66	0,05
0,3	0,6	0,47	0,88	2,00	0,00	0,60	0,05
0,3	0,7	-0,20	0,98	2,00	0,00	0,52	0,05
0,3	0,8	-0,07	1,00	2,00	0,00	0,47	0,05
0,3	0,9	-0,20	0,98	2,00	0,00	0,40	0,05
0,3	1	-0,33	0,94	2,00	0,00	0,35	0,05

Table B.6: QOMDP Depolarizing Channel

B.3 Damping Channel

MDP Damping					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,98	0,01	3,25	2,74
0,1	0,1	0,98	0,01	3,61	2,98
0,1	0,2	0,97	0,01	4,62	4,44
0,1	0,3	0,97	0,01	5,77	6,50
0,1	0,4	0,97	0,01	5,45	6,33
0,1	0,5	0,96	0,01	15,06	24,46
0,1	0,6	0,96	0,01	16,13	27,29
0,1	0,7	0,96	0,01	19,98	32,97
0,1	0,8	0,96	0,01	29,03	47,27
0,1	0,9	0,96	0,01	58,47	97,39
0,1	1	0,96	0,00	51,63	75,67
0,15	0	0,97	0,01	5,30	4,93
0,15	0,1	0,97	0,01	8,01	8,69
0,15	0,2	0,97	0,01	15,94	25,35
0,15	0,3	0,93	0,16	105,03	156,97
0,15	0,4	0,89	0,23	167,82	200,03
0,15	0,5	0,72	0,33	240,40	238,58
0,15	0,6	0,70	0,34	279,28	246,62
0,15	0,7	0,70	0,33	266,52	249,00
0,15	0,8	0,57	0,35	376,29	216,00
0,15	0,9	0,59	0,33	411,18	191,71
0,15	1	0,45	0,30	501,00	0,00
0,175	0	0,98	0,01	5,92	5,92
0,175	0,1	0,97	0,01	13,13	21,73
0,175	0,2	0,95	0,11	67,51	114,69
0,175	0,3	0,72	0,29	315,45	236,80
0,175	0,4	0,73	0,30	250,12	248,35
0,175	0,5	0,69	0,32	286,48	246,99
0,175	0,6	0,70	0,33	231,54	248,70
0,175	0,7	0,50	0,32	421,16	182,94

Table B.7 continued from previous page

0,175	0,8	0,48	0,31	421,16	182,94
0,175	0,9	0,52	0,31	421,16	182,94
0,175	1	0,49	0,27	501,00	0,00
0,2	0	0,98	0,01	6,05	7,34
0,2	0,1	0,96	0,01	36,52	56,27
0,2	0,2	0,78	0,28	241,75	238,76
0,2	0,3	0,65	0,33	295,89	240,37
0,2	0,4	0,61	0,32	326,35	238,01
0,2	0,5	0,46	0,27	441,12	162,16
0,2	0,6	0,47	0,30	426,15	178,18
0,2	0,7	0,45	0,27	436,13	167,82
0,2	0,8	0,51	0,30	411,18	191,71
0,2	0,9	0,34	0,24	501,00	0,00
0,2	1	0,35	0,22	501,00	0,00
0,25	0	0,97	0,01	18,86	49,58
0,25	0,1	0,60	0,34	286,43	247,04
0,25	0,2	0,54	0,35	316,37	240,92
0,25	0,3	0,53	0,36	311,38	242,21
0,25	0,4	0,35	0,25	441,12	162,16
0,25	0,5	0,36	0,29	421,16	182,94
0,25	0,6	0,26	0,14	501,00	0,00
0,25	0,7	0,26	0,14	501,00	0,00
0,25	0,8	0,20	0,12	501,00	0,00
0,25	0,9	0,23	0,14	501,00	0,00
0,25	1	0,24	0,15	501,00	0,00
0,3	0	0,96	0,01	28,82	49,88
0,3	0,1	0,28	0,14	501,00	0,00
0,3	0,2	0,25	0,09	501,00	0,00
0,3	0,3	0,24	0,09	501,00	0,00
0,3	0,4	0,23	0,08	501,00	0,00
0,3	0,5	0,21	0,08	501,00	0,00
0,3	0,6	0,18	0,06	501,00	0,00
0,3	0,7	0,18	0,05	501,00	0,00
0,3	0,8	0,15	0,05	501,00	0,00

Table B.7 continued from previous page

0,3	0,9	0,13	0,04	501,00	0,00
0,3	1	0,12	0,04	501,00	0,00

Table B.7: MDP Damping Channel

POMDP Damping					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,97	0,01	7,50	8,43
0,1	0,1	0,97	0,01	7,76	9,56
0,1	0,2	0,97	0,01	8,76	10,98
0,1	0,3	0,97	0,01	7,70	10,96
0,1	0,4	0,97	0,01	9,10	11,03
0,1	0,5	0,97	0,01	8,29	12,86
0,1	0,6	0,97	0,01	13,26	20,48
0,1	0,7	0,96	0,01	18,17	28,02
0,1	0,8	0,96	0,01	15,71	28,14
0,1	0,9	0,96	0,00	16,94	22,12
0,1	1	0,96	0,00	21,82	35,37
0,15	0	0,98	0,01	12,15	18,26
0,15	0,1	0,97	0,01	15,93	21,63
0,15	0,2	0,97	0,01	14,20	20,14
0,15	0,3	0,97	0,01	18,06	30,07
0,15	0,4	0,97	0,01	41,23	60,93
0,15	0,5	0,96	0,01	49,12	78,70
0,15	0,6	0,95	0,08	63,56	101,91
0,15	0,7	0,85	0,26	164,38	202,35
0,15	0,8	0,51	0,37	401,26	199,48
0,15	0,9	0,49	0,36	406,19	195,76
0,15	1	0,35	0,30	501,00	0,00
0,175	0	0,97	0,01	10,59	14,00
0,175	0,1	0,97	0,01	17,89	30,05
0,175	0,2	0,96	0,01	26,41	43,29
0,175	0,3	0,96	0,09	51,46	99,88
0,175	0,4	0,94	0,11	107,43	152,13

Table B.8 continued from previous page

0,175	0,5	0,81	0,30	186,10	217,66
0,175	0,6	0,60	0,36	311,38	242,21
0,175	0,7	0,45	0,34	416,17	187,44
0,175	0,8	0,44	0,32	426,15	178,18
0,175	0,9	0,43	0,33	411,18	191,71
0,175	1	0,36	0,26	501,00	0,00
0,2	0	0,98	0,01	12,72	20,79
0,2	0,1	0,97	0,01	35,15	57,46
0,2	0,2	0,95	0,09	79,37	124,15
0,2	0,3	0,83	0,29	194,24	212,57
0,2	0,4	0,63	0,36	287,23	239,12
0,2	0,5	0,45	0,33	411,18	191,71
0,2	0,6	0,47	0,34	401,20	199,60
0,2	0,7	0,45	0,34	391,22	206,71
0,2	0,8	0,45	0,33	411,18	191,71
0,2	0,9	0,38	0,26	501,00	0,00
0,2	1	0,35	0,22	501,00	0,00
0,25	0	0,98	0,01	16,35	31,42
0,25	0,1	0,83	0,30	184,57	214,06
0,25	0,2	0,58	0,37	294,34	244,45
0,25	0,3	0,52	0,36	336,33	234,64
0,25	0,4	0,40	0,29	426,15	178,18
0,25	0,5	0,36	0,25	456,09	142,80
0,25	0,6	0,38	0,25	451,10	149,70
0,25	0,7	0,33	0,16	501,00	0,00
0,25	0,8	0,33	0,16	501,00	0,00
0,25	0,9	0,32	0,16	501,00	0,00
0,25	1	0,34	0,16	501,00	0,00
0,3	0	0,97	0,01	8,16	12,13
0,3	0,1	0,56	0,36	291,42	246,29
0,3	0,2	0,41	0,30	391,22	206,71
0,3	0,3	0,38	0,26	426,15	178,18
0,3	0,4	0,30	0,08	501,00	0,00
0,3	0,5	0,30	0,09	501,00	0,00

Table B.8 continued from previous page

0,3	0,6	0,31	0,07	501,00	0,00
0,3	0,7	0,31	0,07	501,00	0,00
0,3	0,8	0,32	0,07	501,00	0,00
0,3	0,9	0,33	0,06	501,00	0,00
0,3	1	0,33	0,07	501,00	0,00

Table B.8: POMDP Damping Channel

QOMDP Damping Channel							
epsilon	alpha	reward	reward_std	timesteps	timesteps_std	fidelity	fidelity_std
0,1	0	0,96	0,28	3,57	1,15	0,95	0,04
0,1	0,1	0,86	0,51	3,79	1,32	0,93	0,06
0,1	0,2	0,92	0,39	3,93	1,51	0,94	0,05
0,1	0,3	0,82	0,57	3,77	1,15	0,93	0,07
0,1	0,4	0,92	0,39	3,81	1,45	0,94	0,05
0,1	0,5	0,88	0,47	3,81	1,49	0,93	0,06
0,1	0,6	0,92	0,39	4,02	1,66	0,93	0,06
0,1	0,7	0,78	0,63	3,87	1,67	0,92	0,07
0,1	0,8	0,84	0,54	3,76	1,62	0,92	0,06
0,1	0,9	0,92	0,39	3,90	1,73	0,93	0,05
0,1	1	0,82	0,57	3,70	1,10	0,91	0,07
0,15	0	0,82	0,57	4,09	2,12	0,90	0,12
0,15	0,1	0,78	0,63	3,80	1,86	0,92	0,10
0,15	0,2	0,80	0,60	4,40	2,40	0,90	0,10
0,15	0,3	0,84	0,54	3,79	1,72	0,90	0,08
0,15	0,4	0,86	0,51	3,82	1,97	0,93	0,07
0,15	0,5	0,78	0,63	3,71	1,57	0,90	0,10
0,15	0,6	0,88	0,47	4,06	2,08	0,87	0,14
0,15	0,7	0,88	0,47	3,81	1,90	0,90	0,09
0,15	0,8	0,84	0,54	3,77	1,95	0,90	0,09
0,15	0,9	0,78	0,63	3,90	2,09	0,89	0,09
0,15	1	0,78	0,63	3,93	1,65	0,89	0,08
0,175	0	0,76	0,65	4,45	2,94	0,92	0,10
0,175	0,1	0,86	0,51	4,84	4,18	0,93	0,07

Table B.9 continued from previous page

0,175	0,2	0,62	0,78	5,32	4,31	0,91	0,08
0,175	0,3	0,82	0,57	5,25	3,75	0,90	0,10
0,175	0,4	0,76	0,65	6,04	4,76	0,89	0,09
0,175	0,5	0,78	0,63	5,76	4,48	0,89	0,08
0,175	0,6	0,74	0,67	5,17	4,21	0,90	0,09
0,175	0,7	0,80	0,60	5,41	4,30	0,88	0,11
0,175	0,8	0,80	0,60	4,71	3,35	0,88	0,08
0,175	0,9	0,78	0,63	5,44	4,06	0,88	0,08
0,175	1	0,68	0,73	4,89	3,39	0,88	0,07
0,2	0	0,84	0,54	3,21	0,60	0,92	0,09
0,2	0,1	0,80	0,60	3,35	0,95	0,91	0,11
0,2	0,2	0,78	0,63	3,23	0,56	0,88	0,12
0,2	0,3	0,82	0,57	3,18	0,88	0,89	0,08
0,2	0,4	0,86	0,51	3,18	0,57	0,88	0,11
0,2	0,5	0,78	0,63	3,12	0,41	0,89	0,09
0,2	0,6	0,72	0,69	3,29	0,62	0,85	0,13
0,2	0,7	0,64	0,77	3,30	0,91	0,85	0,12
0,2	0,8	0,72	0,69	3,19	0,64	0,84	0,14
0,2	0,9	0,60	0,80	3,19	0,50	0,83	0,12
0,2	1	0,76	0,65	3,29	0,62	0,82	0,13
0,25	0	0,84	0,54	3,00	0,00	0,93	0,06
0,25	0,1	0,86	0,51	3,00	0,00	0,92	0,06
0,25	0,2	0,80	0,60	3,00	0,00	0,91	0,06
0,25	0,3	0,70	0,71	3,00	0,00	0,90	0,06
0,25	0,4	0,74	0,67	3,00	0,00	0,90	0,06
0,25	0,5	0,76	0,65	3,00	0,00	0,90	0,06
0,25	0,6	0,62	0,78	3,00	0,00	0,86	0,08
0,25	0,7	0,72	0,69	3,00	0,00	0,86	0,08
0,25	0,8	0,68	0,73	3,00	0,00	0,86	0,07
0,25	0,9	0,68	0,73	3,00	0,00	0,83	0,09
0,25	1	0,68	0,73	3,00	0,00	0,82	0,08
0,3	0	0,84	0,54	3,00	0,00	0,95	0,02
0,3	0,1	0,90	0,44	3,00	0,00	0,94	0,02
0,3	0,2	0,96	0,28	3,00	0,00	0,93	0,02

Table B.9 continued from previous page

0,3	0,3	0,84	0,54	3,00	0,00	0,92	0,02
0,3	0,4	0,86	0,51	3,00	0,00	0,92	0,02
0,3	0,5	0,82	0,57	3,00	0,00	0,90	0,03
0,3	0,6	0,86	0,51	3,00	0,00	0,89	0,03
0,3	0,7	0,76	0,65	3,00	0,00	0,88	0,03
0,3	0,8	0,72	0,69	3,00	0,00	0,86	0,03
0,3	0,9	0,70	0,71	3,00	0,00	0,85	0,03
0,3	1	0,66	0,75	3,00	0,00	0,83	0,03

Table B.9: QOMDP Damping Channel

B.4 Random Flip Channel

MDP Random Flip Channel					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,98	0,01	3,80	3,22
0,1	0,1	0,97	0,01	4,49	3,74
0,1	0,2	0,96	0,01	5,10	4,18
0,1	0,3	0,95	0,00	13,88	14,21
0,1	0,4	0,46	0,35	501,00	0,00
0,1	0,5	0,54	0,36	501,00	0,00
0,1	0,6	0,37	0,34	501,00	0,00
0,1	0,7	0,45	0,35	501,00	0,00
0,1	0,8	0,38	0,35	501,00	0,00
0,1	0,9	0,36	0,34	501,00	0,00
0,1	1	0,32	0,32	501,00	0,00
0,15	0	0,97	0,01	4,89	4,22
0,15	0,1	0,96	0,01	10,07	13,93
0,15	0,2	0,49	0,32	501,00	0,00
0,15	0,3	0,52	0,29	501,00	0,00
0,15	0,4	0,45	0,29	501,00	0,00
0,15	0,5	0,39	0,27	501,00	0,00
0,15	0,6	0,44	0,29	501,00	0,00
0,15	0,7	0,43	0,28	501,00	0,00
0,15	0,8	0,33	0,25	501,00	0,00
0,15	0,9	0,34	0,26	501,00	0,00
0,15	1	0,32	0,25	501,00	0,00
0,175	0	0,98	0,01	5,04	4,48
0,175	0,1	0,96	0,01	32,08	44,61
0,175	0,2	0,51	0,27	501,00	0,00
0,175	0,3	0,43	0,25	501,00	0,00
0,175	0,4	0,41	0,26	501,00	0,00
0,175	0,5	0,40	0,25	501,00	0,00
0,175	0,6	0,44	0,24	501,00	0,00
0,175	0,7	0,42	0,24	501,00	0,00

Table B.10 continued from previous page

0,175	0,8	0,41	0,25	501,00	0,00
0,175	0,9	0,36	0,23	501,00	0,00
0,175	1	0,34	0,23	501,00	0,00
0,2	0	0,98	0,01	5,10	5,79
0,2	0,1	0,95	0,03	112,26	113,83
0,2	0,2	0,46	0,23	501,00	0,00
0,2	0,3	0,43	0,22	501,00	0,00
0,2	0,4	0,46	0,22	501,00	0,00
0,2	0,5	0,42	0,21	501,00	0,00
0,2	0,6	0,40	0,20	501,00	0,00
0,2	0,7	0,38	0,20	501,00	0,00
0,2	0,8	0,37	0,20	501,00	0,00
0,2	0,9	0,31	0,18	501,00	0,00
0,2	1	0,35	0,19	501,00	0,00
0,25	0	0,97	0,01	23,26	57,76
0,25	0,1	0,44	0,14	501,00	0,00
0,25	0,2	0,45	0,17	501,00	0,00
0,25	0,3	0,40	0,12	501,00	0,00
0,25	0,4	0,39	0,14	501,00	0,00
0,25	0,5	0,39	0,14	501,00	0,00
0,25	0,6	0,37	0,12	501,00	0,00
0,25	0,7	0,36	0,13	501,00	0,00
0,25	0,8	0,36	0,12	501,00	0,00
0,25	0,9	0,33	0,12	501,00	0,00
0,25	1	0,33	0,12	501,00	0,00
0,3	0	0,96	0,01	26,14	43,20
0,3	0,1	0,34	0,10	501,00	0,00
0,3	0,2	0,33	0,07	501,00	0,00
0,3	0,3	0,33	0,06	501,00	0,00
0,3	0,4	0,34	0,06	501,00	0,00
0,3	0,5	0,34	0,05	501,00	0,00
0,3	0,6	0,33	0,05	501,00	0,00
0,3	0,7	0,33	0,05	501,00	0,00
0,3	0,8	0,33	0,05	501,00	0,00

Table B.10 continued from previous page

0,3	0,9	0,33	0,05	501,00	0,00
0,3	1	0,34	0,05	501,00	0,00

Table B.10: MDP Random Flip Channel

POMDP Random Flip Channel					
epsilon	alpha	fidelity	fidelity_std	timesteps	timesteps_std
0,1	0	0,97	0,01	8,58	11,16
0,1	0,1	0,96	0,01	13,65	17,52
0,1	0,2	0,66	0,39	390,92	152,68
0,1	0,3	0,31	0,33	501,00	0,00
0,1	0,4	0,29	0,32	501,00	0,00
0,1	0,5	0,35	0,34	501,00	0,00
0,1	0,6	0,36	0,34	501,00	0,00
0,1	0,7	0,37	0,34	501,00	0,00
0,1	0,8	0,32	0,33	501,00	0,00
0,1	0,9	0,35	0,33	501,00	0,00
0,1	1	0,32	0,32	501,00	0,00
0,15	0	0,98	0,01	9,61	10,63
0,15	0,1	0,96	0,01	38,47	54,50
0,15	0,2	0,35	0,31	501,00	0,00
0,15	0,3	0,27	0,27	501,00	0,00
0,15	0,4	0,34	0,28	501,00	0,00
0,15	0,5	0,33	0,26	501,00	0,00
0,15	0,6	0,33	0,26	501,00	0,00
0,15	0,7	0,28	0,24	501,00	0,00
0,15	0,8	0,34	0,26	501,00	0,00
0,15	0,9	0,34	0,26	501,00	0,00
0,15	1	0,32	0,25	501,00	0,00
0,175	0	0,97	0,01	11,96	16,55
0,175	0,1	0,58	0,38	362,58	194,25
0,175	0,2	0,37	0,25	501,00	0,00
0,175	0,3	0,33	0,24	501,00	0,00
0,175	0,4	0,33	0,23	501,00	0,00

Table B.11 continued from previous page

0,175	0,5	0,32	0,22	501,00	0,00
0,175	0,6	0,33	0,23	501,00	0,00
0,175	0,7	0,31	0,22	501,00	0,00
0,175	0,8	0,37	0,23	501,00	0,00
0,175	0,9	0,31	0,22	501,00	0,00
0,175	1	0,33	0,22	501,00	0,00
0,2	0	0,98	0,01	9,71	15,21
0,2	0,1	0,60	0,37	316,46	218,12
0,2	0,2	0,32	0,23	501,00	0,00
0,2	0,3	0,31	0,21	501,00	0,00
0,2	0,4	0,33	0,22	501,00	0,00
0,2	0,5	0,34	0,20	501,00	0,00
0,2	0,6	0,33	0,20	501,00	0,00
0,2	0,7	0,31	0,18	501,00	0,00
0,2	0,8	0,31	0,18	501,00	0,00
0,2	0,9	0,36	0,20	501,00	0,00
0,2	1	0,38	0,20	501,00	0,00
0,25	0	0,98	0,01	14,34	29,52
0,25	0,1	0,45	0,28	421,16	182,94
0,25	0,2	0,34	0,15	501,00	0,00
0,25	0,3	0,34	0,16	501,00	0,00
0,25	0,4	0,35	0,13	501,00	0,00
0,25	0,5	0,35	0,14	501,00	0,00
0,25	0,6	0,32	0,12	501,00	0,00
0,25	0,7	0,34	0,13	501,00	0,00
0,25	0,8	0,33	0,12	501,00	0,00
0,25	0,9	0,32	0,11	501,00	0,00
0,25	1	0,34	0,12	501,00	0,00
0,3	0	0,97	0,01	7,96	7,92
0,3	0,1	0,34	0,10	501,00	0,00
0,3	0,2	0,34	0,07	501,00	0,00
0,3	0,3	0,32	0,06	501,00	0,00
0,3	0,4	0,32	0,06	501,00	0,00
0,3	0,5	0,33	0,05	501,00	0,00

Table B.11 continued from previous page

0,3	0,6	0,33	0,05	501,00	0,00
0,3	0,7	0,33	0,05	501,00	0,00
0,3	0,8	0,33	0,04	501,00	0,00
0,3	0,9	0,33	0,05	501,00	0,00
0,3	1	0,34	0,05	501,00	0,00

Table B.11: POMDP Random Flip Channel

QOMDP Random Flip Channel							
epsilon	alpha	reward	reward_std	timesteps	timesteps_std	fidelity	fidelity_std
0,1	0	0,90	0,44	3,71	1,43	0,94	0,10
0,1	0,1	0,88	0,47	4,15	1,96	0,92	0,11
0,1	0,2	0,92	0,39	4,31	1,74	0,91	0,07
0,1	0,3	0,78	0,63	4,61	2,20	0,89	0,07
0,1	0,4	0,66	0,75	4,27	2,00	0,87	0,09
0,1	0,5	0,70	0,71	4,39	2,57	0,87	0,06
0,1	0,6	0,62	0,78	4,25	1,59	0,85	0,07
0,1	0,7	0,66	0,75	4,64	2,28	0,85	0,04
0,1	0,8	0,66	0,75	4,33	2,03	0,83	0,03
0,1	0,9	0,70	0,71	4,95	2,32	0,81	0,07
0,1	1	0,58	0,81	4,85	2,21	0,79	0,07
0,15	0	0,72	0,69	3,83	1,90	0,90	0,14
0,15	0,1	0,76	0,65	4,59	3,07	0,87	0,15
0,15	0,2	0,70	0,71	3,96	2,17	0,84	0,17
0,15	0,3	0,58	0,81	4,36	2,88	0,81	0,16
0,15	0,4	0,54	0,84	4,54	3,29	0,74	0,20
0,15	0,5	0,50	0,87	4,38	2,71	0,74	0,21
0,15	0,6	0,06	1,00	4,52	2,53	0,67	0,23
0,15	0,7	0,44	0,90	4,13	2,10	0,70	0,19
0,15	0,8	0,32	0,95	4,21	2,76	0,63	0,24
0,15	0,9	0,26	0,97	4,56	3,04	0,58	0,24
0,15	1	0,34	0,94	5,17	2,98	0,62	0,19
0,175	0	0,76	0,65	5,25	3,85	0,86	0,22
0,175	0,1	0,76	0,65	5,16	3,90	0,87	0,15

Table B.12 continued from previous page

0,175	0,2	0,74	0,67	5,64	4,06	0,86	0,10
0,175	0,3	0,70	0,71	5,13	4,39	0,84	0,10
0,175	0,4	0,72	0,69	6,44	5,15	0,79	0,12
0,175	0,5	0,46	0,89	6,26	4,75	0,76	0,12
0,175	0,6	0,58	0,81	5,41	3,77	0,74	0,12
0,175	0,7	0,28	0,96	6,30	5,31	0,72	0,13
0,175	0,8	0,36	0,93	7,03	6,91	0,69	0,10
0,175	0,9	0,26	0,97	6,98	5,23	0,64	0,14
0,175	1	0,22	0,98	7,22	6,11	0,60	0,15
0,2	0	0,74	0,67	3,35	1,01	0,91	0,11
0,2	0,1	0,64	0,77	3,27	0,63	0,85	0,13
0,2	0,2	0,44	0,90	3,27	0,85	0,79	0,13
0,2	0,3	0,42	0,91	3,28	0,72	0,74	0,16
0,2	0,4	0,40	0,92	3,31	0,74	0,71	0,18
0,2	0,5	0,34	0,94	3,42	0,80	0,62	0,20
0,2	0,6	0,36	0,93	3,32	0,84	0,56	0,20
0,2	0,7	-0,08	1,00	3,40	0,99	0,53	0,21
0,2	0,8	-0,16	0,99	3,33	0,98	0,46	0,20
0,2	0,9	-0,32	0,95	3,32	0,96	0,38	0,20
0,2	1	-0,24	0,97	3,43	0,96	0,36	0,20
0,25	0	0,86	0,51	3,00	0,00	0,92	0,07
0,25	0,1	0,74	0,67	3,00	0,00	0,87	0,07
0,25	0,2	0,60	0,80	3,00	0,00	0,80	0,08
0,25	0,3	0,32	0,95	3,00	0,00	0,73	0,10
0,25	0,4	0,34	0,94	3,00	0,00	0,68	0,11
0,25	0,5	0,46	0,89	3,00	0,00	0,62	0,12
0,25	0,6	0,20	0,98	3,00	0,00	0,56	0,12
0,25	0,7	0,06	1,00	3,00	0,00	0,48	0,12
0,25	0,8	-0,26	0,97	3,00	0,00	0,45	0,13
0,25	0,9	-0,16	0,99	3,00	0,00	0,39	0,12
0,25	1	-0,42	0,91	3,00	0,00	0,32	0,11
0,3	0	0,92	0,39	3,00	0,00	0,95	0,02
0,3	0,1	0,88	0,47	3,00	0,00	0,88	0,03
0,3	0,2	0,66	0,75	3,00	0,00	0,82	0,03

Table B.12 continued from previous page

0,3	0,3	0,46	0,89	3,00	0,00	0,76	0,04
0,3	0,4	0,36	0,93	3,00	0,00	0,70	0,05
0,3	0,5	0,50	0,87	3,00	0,00	0,63	0,05
0,3	0,6	0,24	0,97	3,00	0,00	0,58	0,05
0,3	0,7	-0,14	0,99	3,00	0,00	0,52	0,05
0,3	0,8	-0,18	0,98	3,00	0,00	0,45	0,05
0,3	0,9	0,00	1,00	3,00	0,00	0,41	0,05
0,3	1	-0,30	0,95	3,00	0,00	0,34	0,05

Table B.12: QOMDP Random Flip Channel

References

[1] Open AI. *Part 2: Kinds of RL Algorithms*. 2018. URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.

[2] Open AI. *Proximal Policy Optimization*. 2018. URL: https://spinningup.openai.com/en/latest/algorithms/ppo.html.

[3] Shirin Akbarinasaji et al. "Partially observable Markov decision process to generate policies in software defect management". In: *Journal of Systems and Software* 163 (2020), p. 110518. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110518. URL: https://www.sciencedirect.com/science/article/pii/S0164121220300017.

[4] Jennifer Barry, Daniel T. Barry, and Scott Aaronson. "Quantum partially observable Markov decision processes". In: *Physical Review A* 90.3 (Sept. 2014). DOI: 10.1103/physreva.90.032311. URL: https://doi.org/10.1103%2Fphysreva.90.032311.

[5] G. Benenti, G. Casati, and G. Strini. *Principles Of Quantum Computation And Information - Volume Ii: Basic Tools And Special Topics*. World Scientific Publishing Company, 2007. ISBN: 9789814365550. URL: https://books.google.it/books?id=Its7DQAAQBAJ.

[6] Daniel Bick. *Towards Delivering a Coherent Self-Contained Explanation of Proximal Policy Optimization*. 2021.

[7] Caleb M. Bowyer. *Partially Observed Markov Decision Processes (POMDPs) for Reinforcement Learning (RL)*. 2022. URL: https://medium.com/mlearning-ai/partially-observed-markov-decision-processes-pomdps-for-reinforcement-learning-rl-437b8ae412dd.

[8] Marin Bukov et al. "Reinforcement Learning in Different Phases of Quantum Control". In: *Physical Review X* 8.3 (Sept. 2018). DOI: 10.1103/physrevx.8.031086. URL: https://doi.org/10.1103%2Fphysrevx.8.031086.

[9] Stefano Chessa and Vittorio Giovannetti. "Quantum capacity analysis of multi-level amplitude damping channels". In: *Communications Physics* 4.1 (Feb. 2021). DOI: 10.1038/s42005-021-00524-4. URL: https://doi.org/10.1038%2Fs42005-021-00524-4.

[10] Wouter van Heeswijk. *Proximal Policy Optimization (PPO) Explained*. The journey from REINFORCE to the go-to algorithm in continuous control. 2022. URL: https://towardsdatascience.com/proximal-policy-optimization-ppo-explained-abed1952457b.

[11] Arthur Juliani. *Maximum Entropy Policies in Reinforcement Learning Everyday Life*. 2018. URL: https://awjuliani.medium.com/maximum-entropy-policies-in-reinforcement-learning-everyday-life-f5a1cc18d32d.

[12] Dhanoop Karunakaran. *The Actor-Critic Reinforcement Learning algorithm*. 2020. URL: https://medium.com/intro-to-artificial-intelligence/the-actor-critic-reinforcement-learning-algorithm-c8095a655c14.

[13] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[14] Riccardo Porotti et al. "Deep Reinforcement Learning for Quantum State Preparation with Weak Nonlinear Measurements". In: *Quantum* 6 (June 2022), p. 747. DOI: 10.22331/q-2022-06-28-747. URL: https://doi.org/10.22331%2Fq-2022-06-28-747.

[15] Francesco Preti, Tommaso Calarco, and Felix Motzoi. "Continuous Quantum Gate Sets and Pulse-Class Meta-Optimization". In: *PRX Quantum* 3.4 (Oct. 2022). DOI: 10.1103/prxquantum.3.040311. URL: https://doi.org/10.1103%2Fprxquantum.3.040311.

[16] *Reinforcement Learning : Markov-Decision Process (Part 1)*. 2019. URL: https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da.

[17] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[18] Thomas Simonini. *Proximal Policy Optimization (PPO)*. The journey from REINFORCE to the go-to algorithm in continuous control. URL: https://huggingface.co/blog/deep-rl-ppo.

[19] V. V. Sivak et al. "Model-Free Quantum Control with Reinforcement Learning". In: *Physical Review X* 12.1 (Mar. 2022). DOI: 10.1103/physrevx.12.011059. URL: https://doi.org/10.1103%2Fphysrevx.12.011059.

[20] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[21] Xiao-Ming Zhang et al. "When does reinforcement learning stand out in quantum control? A comparative study on state preparation". In: *npj Quantum Information* 5.1 (Oct. 2019). DOI: 10.1038/s41534-019-0201-8. URL: https://doi.org/10.1038%2Fs41534-019-0201-8.

Acknowledgments

Al termine di questa tesi volevo dedicare un breve ringraziamento a tutte le persone che in questi anni mi sono state vicine e mi hanno supportato in questo percorso.

Vorrei ringraziare i miei relatori, Francesco Ticozzi e Gian Antonio Susto per la professionalità, la costante disponibilità e i suggerimenti preziosi che hanno guidato l'intera stesura dell'elaborato.

Ringrazio di cuore i miei genitori che in questi anni hanno sacrificato molto per permettermi di intraprendere questa strada, e, che per ogni mia necessità sono sempre stati presenti. Vorrei ringraziare i nonni tutti, che mi hanno sempre sostenuto e dimostrato molto affetto trattandomi come un figlio. Ringrazio gli zii che anche con una sola parola sono riusciti a fornirmi supporto.

Un ringraziamento molto speciale va ad Alessia, per esserci sempre stata, sia nei momenti più facili che in quelli più difficili aiutandomi a non mollare mai, standomi sempre accanto e ascoltandomi ogni qualvolta avessi qualche problema. Infine vorrei ringraziare i miei amici che mi hanno spesso offerto momenti di svago e spensieratezza alleggerendo il percorso.